Position Papers for the ASCR Workshop on Reimagining Codesign

https://www.orau.gov/ASCR-CoDesign/

James A. Ang[1], Andrew A Chien[2], Si Hammond[3], Adolfy Hoisie[4], Ian Karlin[5], Scott Pakin[6] John Shalf[7], and Jeffrey S. Vetter[8] (eds.)

March 2021

[1]Pacific Northwest National Laboratory
[2]Argonne National Laboratory and the University of Chicago
[3]Sandia National Laboratories
[4]Brookhaven National Laboratory
[5]Lawrence Livermore National Laboratory
[6]Los Alamos National Laboratory
[7]Lawrence Berkeley National Laboratory
[8]Oak Ridge National Laboratory

Disclaimer

U.S. Department of Energy Point of Contact:
Hal Finkel, hal.finkel@science.doe.gov

https://doi.org/10.2172/1843574

Contents

Thomas Flynn	Real-time data-driven codesign of hardware and software for analysis of high-throughput scientific experiments
Ana Gainaru	A Data Management codesign of Deep Neural Networks scientific applications on next generation resources
Andreas Gerstlauer	Algorithm-Architecture Codesign for Irregular and Sparse Problems
Roberto Gioiosa	ARIAA: Center for co-design of ARtificial Intelligence focused Architectures and Algorithms
Maya Gokhale	The need for speed: fast feedback for heterogeneous architecture Design Space Exploration
Ryan Grant	Co-design of System Software for Compute Accelerators and SmartNICs
Marcus Hanwell	Edge computing at synchrotrons: co-design for data reduction, monitoring, and real-time analysis
William Harrison	A Codesign Approach to Addressing End-to-End Security, Safety, and Integrity Within Systems of Heterogenous Chiplets
Oscar Hernandez	User-driven Co-design Tools for full System-level Deployments
Graham Heyes	Real-time streaming data acquisition and analysis for Nuclear Physics experiments
Seth Hitefield	Co-Design of Streaming Workflows to Drive Experiments-in-the-Loop
Gang Huang	From Quantum to Computing: Codesign for quantum bit control
Clay Hughes	Compilation Infrastructure for Extremely Heterogeneous Systems
Travis Humble	Heterogeneous Computing Co-design and Quantum Information Processing
Costin Iancu	Opportunities for Co-Design in the Hardware/Software Control Stack for Quantum Processors
Terry Jones	Reconceptualizing Quantitative Codesign
Balint Joo	Future Opportunities for Facilities in Co-design
Mahmut Kandemir	Architecture-Aware Modernization of HPC Applications
Gokcen Kestor	Co-Design through Domain-Specific Compilation Framework
Ryan King	Extracting Maximum Information from Exascale Simulations
Shruti Kulkarni	Neuromorphic Co-Design Towards Linking Emerging Materials with Algorithm Development

Ignacio Laguna	Challenges and Opportunities to Co-design Approximate Computing Scientific Applications
Jack Lange	Hardware Partitioning Support for Discrete Resources
Cannada Lewis	Codesign for the Masses
Lingda Li	A Co-designed Data Layout Transformation Infrastructure for Scientific Computing
Chunhua Liao	A Common Machine-Readable Vocabulary and Knowledge Base Supporting Multiple Programming Models
Jason Lowe-Power	Leveraging open source simulators for HPC codesign
Sandeep Madireddy	Application-Driven Codesign for Online and Continual Learning using Neuromorphic Architectures
Addi Malviya-Thakur	Towards Hybridizing AutoML and Codesign for Next-Generation Scientific Applications
Andres Marquez	Rethinking Codesign Tools for Data-movement
George Michelogiannakis	Compiler, OS, and Scheduler Support for Future Disaggregated HPC Systems
John Mitchell	Application Specific Machine Learning Accelerators Using Digital and Mixed-Signal Reconfigurable Computing
Matthew Norman	Confident, Adaptable, and Robust Machine Learning to Augment Traditional Modeling and Simulation
Peter Nugent	Co-Design for Radiation Transport in Astrophysics
Maryam Parsa	Bayesian Artificial Intelligence Governing Software-Hardware Codesign
Sean Peisert	ASCR Workshop on Co-Design White Paper: Hardware/Software Co-Design to Enable Trustworthy Data Domains for HPC
Ivy Peng	Rethinking Performance Monitoring for Co-Design Computer Systems with Heterogeneous Nodes
Priyanka Raina	Automated Codesign of Domain-Specific Hardware Accelerators and Compilers
Siva Rajamanickam	Vision for Co-designing a Unified-Memory Centric Heterogeneous Node Architecture
Sanjay Rajopadhye	Domain Specific CoDesign
Nageswara Rao	Co-Design for Edge-Core Stretched Computing with Integrated Instruments
Robert Robey	Codesign and Computational Methods
Arun Rodrigues	Unified, Open, Multi-Fidelity, Scalable Architectural Simulation
Tim Rogers	Raising the Level of Accelerator Abstraction
Robert Ross	Composition as a tool for HPC service development
Daniel Sanchez	A Full-Stack Architecture for Efficient Sparse Computation

Catherine Schuman	Top-Down Neuromorphic Hardware Co-Design via Machine Learning and Simulation
Sadasivan Shankar	Co-designing from Atoms to Architectures
Sameer Shende	Observing and Optimizing Performance of Coupled MPI Applications
Matthew Sinclair	Co-designing Power Management with Job Scheduling for Efficient Exascale Computing
Matthew Sottile	Automatic derivation of performance models for programs with data dependent behavior
Christine Sweeney	Codesign for Experimental Workflows at User Facilities
Keita Teranishi	Resilience-Aware Codesign: A Full-Stack Approach
Antonino Tumeo	Reimagining Hardware/Software Codesign Tools for Domain Specialization
John Paul Walters	Unifying the HPC and Edge Programming Models
Tim Warburton	Co-Designing Scalable Hardware and Algorithms with Application-Driven Benchmarks
Tim Williams	Systemwide Codesign with Community Software Stack
Matthew Wolf	Toward Machine-Actionable and Reusable Codesign Decisions
Nicholas Wright	Co-design for Whole System Architectures
Xingfu Wu	End-to-End PowerStack Codesign for Energy Efficient HPC
Angel Yanguas-Gil	Codesign approaches to enable and optimize computing at the extreme edge
Kazutomo Yoshii	Co-design Streaming Processing Hardware using Chisel Hardware Construction Language
Aaron Young	Emerging Heterogeneous Systems Provide Great Opportunities for Codesign
Andrew Younge	Containerization Should Influence New Hardware-assisted Mechanisms for Resource Multiplexing and Isolation
Victor Zhirnov	CoDesign 2030 Agenda

Call for Position Papers: Workshop on Reimagining Codesign

Important Dates

- 16th February 2021: Deadline for position paper submission
- 2nd March 2021: Notification of position acceptance
- 16–18th March 2021: Workshop
- SUBMISSION URL: ASCR Workshop on Co-Design - Position Papers
- WORKSHOP URL: https://www.orau.gov/ASCR-CoDesign/

Motivation

On behalf of the Advanced Scientific Computing Research (ASCR) program in the US Department of Energy (DOE) Office of Science, we are organizing a Workshop on Reimagining Codesign (ReCoDe). Codesign is the process of jointly designing interoperating components of a computing system—in particular: applications, algorithms, system software, programming models, and the hardware on which they run. The goal is to maximize the overall performance, efficiency, and other desirable qualities of the system as a whole. Codesign is a standard methodology in the embedded-systems community, where space, power, and cost constraints are commonly pitted against execution speed for a tightly constrained feature set. Over the last decade, the DOE has invested in codesign efforts to foster the development of exascale computing systems for broad classes of scientific and engineering applications. The ReCoDe workshop hopes to explore how scientific applications of interest to the DOE can be accelerated through close interactions with hardware designers and software-stack developers, in which all components adapt to each other's requirements and constraints. We want to answer the question of what are the key tools and methodologies for accomplishing codesign in today's computing landscape, and what will be the highest impact targets for meeting DOE's emerging mission requirements.

This workshop aims to bring together DOE, industry, and academia to identify opportunities to build on past codesign successes and identify new areas that are either emerging or that may need reimagining for the future. We want to continue to find opportunities that can be pursued as a joint effort and continue to break down the traditional customer/vendor dichotomy with true partnerships. From this work, DOE will benefit from increased application performance relative to what stock hardware or existing general-purpose roadmaps can provide, and vendors will benefit from expanding their hardware's capabilities to address needs they might have not otherwise anticipated and thereby create more widespread interest in their products.

The workshop will be structured around a set of breakout sessions, with every attendee expected to participate actively in the discussions. Afterward, workshop attendees—from DOE, industry, and academia—will produce a report for ASCR that summarizes the findings made during the workshop.

Invitation

We invite community input in the form of two-page position papers that identify and discuss key challenges and opportunities in the area of hardware/software codesign for scientific computing. In addition to providing an avenue for identifying workshop participants, these position papers will be used to shape the workshop agenda, identify panelists, and contribute to the workshop report. Position papers should not describe the authors' current or planned research, contain material that should not be disclosed to the public, nor should they recommend specific solutions or discuss narrowly focused research topics. Rather, they should aim to improve the community's shared understanding of the problem space, identify challenging research directions, and help to stimulate discussion.

One author of each selected submission will be invited to participate in the workshop.

By submitting a position paper, authors consent to have their position paper published publicly.

Authors are not required to have a history of funding by the ASCR Computer Science program.

Submission Guidelines

Position Paper Structure and Format

Position papers should follow the following format:

- Title
- Authors (with affiliations and email addresses)
- Topic: one or more of the following: architectures, applications, modeling and simulation, programming systems, emerging technologies, codesign methodologies
- Challenge: Identify aspects of current codesign that show the limitations of state-of-the-art practice with examples
- Opportunity: Describe how the identified challenges may be addressed, whether it is through new tools and techniques, new technologies, or new groups collaborating in the codesign process
- Timeliness or maturity: Why now? What breakthrough or change makes progress possible now where it wasn't possible before? What will be the impact of success?
- References

Each position paper must be no more than two pages including figures and references. The paper may include any number of authors but contact information for a single author who can represent the position paper at the workshop must be provided with the submission. There is no limit to the number of position papers that an individual or group can submit. Authors are strongly encouraged to follow the structure previously outlined. Papers should be submitted in PDF format using the designated page on the workshop website.

Areas of Emphasis

We are seeking submissions along the following general directions:

- Key aspects of codesign across the entire hardware/software stack to include applications, algorithms, system software, and system architecture
- Insights into codesign for workflows arising from scientific experiments, on supercomputers, or to support large-scale scientific instruments
- Methods and tools for quantitative codesign, including both those that inform high-level decision-making and those impacting low-level aspects of the codesign process.
- New codesign challenges anticipated over the next decade

Notional Questions

Position papers should present a view on how hardware, firmware, the software stack, and scientific applications can be codesigned for maximal effectiveness, perhaps taking inspiration from some of the following:

- How to balance breadth of applications versus customization benefit? Does this vary by system type?
- What are the tools and techniques that enable successful codesign interactions and where are the gaps?
- With artificial intelligence and machine learning becoming more widely used in scientific workflows and applications, what new challenges and opportunities does this present?
- New accelerator technologies and chiplets are increasing the possible design space. What is the potential of these new technologies and how can codesign be used to take maximum advantage of them?
- How can we further the state of the art in efficient and flexible open-source hardware, modeling, and simulation tools that can underpin hardware codesign activities?
- Is there a performance benefit to codesigning scientific applications and the computer systems (hardware and software) they run on, or will the additional time and cost outweigh the benefits observed relative to more-or-less portable applications running on stock supercomputers?
- How do scientific applications and supercomputer codesign differ fundamentally from the codesign employed regularly for embedded systems (such as in automobiles and home appliances)? Can either area learn from the other?

- What are the lessons learned from DOE’s last decade of exascale codesign collaborations among national laboratories, industry, and universities? What should be continued? What should be avoided?
- What are the roles that academia can play in codesign benefiting DOE systems?
- What do application developers and hardware designers need to do differently when beginning to think about codesign?
- What is the basic timeline for the constituent activities in a scientific-application codesign process? How can it be improved?
- What emerging hardware or software technologies will facilitate rapid development of both scientific applications and hardware platforms with sufficiently-flexible components to help produce successful outcomes?
- If codesign of scientific applications and hardware platforms becomes the norm, what new skills will be required of application/hardware developers? How will this future workforce be trained?
- What role does system software play in a codesign process?
- Open Source Software is supported by long-standing DOE policy. How can Open Source Hardware support codesign?
- What is the role of DOE in codesign for edge computing?
- What programming systems concepts facilitate effective codesign and why?

Selection

Submissions will be reviewed by the workshop’s organizing committee using criteria of overall quality, relevance, likelihood of stimulating constructive discussion, and ability to contribute to an informative workshop report. Unique positions that are well presented and emphasize potentially-transformative research directions will be given preference.

Re-Imagining Codesign

https://www.orau.gov/ASCR-CoDesign/

Version: 1.0

Organizing Committee:

- John Shalf, LBL
- Andrew A Chien, ANL
- Jeffrey Vetter, ORNL
- Jim Ang, PNNL
- Adolfy Hoisie, BNL
- Si Hammond, SNL
- Ian Karlin, LLNL
- Scott Pakin, LANL

DOE Point of Contact: Hal Finkel, DOE/ASCR

Introduction

The DOE has successfully employed codesign as a methodology to improve both the software and hardware in advanced high-performance computing (HPC) systems; thereby increasing both the mission-relevant capabilities of these computing systems and accelerating the advance of science. The reimagining of codesign contemplates the expansion of codesign in scope and ambition to reflect the clear and increasing importance of computing as a foundation for science and the broadening scope of where it is critical from HPC platforms to edge computing and data processing for experimental sciences.

History

The computing community has come to expect the doubling of the number of transistors on a chip every two years (Moore's Law), which has contributed immensely to improvements in computer performance over the past five decades. This expectation has led to a relatively stable ecosystem (e.g. electronic design automation tools, emulators, simulators, system software, compilers, algorithms, applications) built around general-purpose processor technologies (x86, Power, Arm, etc.). This stability meant that developers of each layer in the deep technology stack between application users and hardware designers only needed to focus on interface requirements for adjacent layers. This unified view of the computing-technology roadmap supported incredible growth in design efficiencies and the development of very high-performance general-purpose processors.

Scope

However, silicon-based transistors cannot get much smaller than they are today, and pushing these physical limits is driving large-scale disruption of the computing ecosystem[1]. New approaches must be explored to ensure continued growth in system performance. Transformational technologies are emerging that provide the opportunity for powerful new approaches to hardware/software codesign to specialize future computing environments to emerging DOE applications in HPC, edge computing, and data processing for experimental sciences.

Opportunity

In response to the flagging of traditional approaches to scaling, trends in hardware technology within the HPC ecosystem are fueling a trajectory toward extreme heterogeneity [1]. Advances in fabrication, packaging, and other areas continue to create opportunities [3] for codesigning hardware along with software in a wide variety of different domains [2,4]. Large-scale scientific computing continues to drive the leading edge in HPC and codesigning the next generation of HPC hardware along with scientific software discovery promises to further accelerate scientific delivery. Not only have modern packaging technologies, such as chiplets, widened the ways in which codesigned hardware components can be integrated into larger systems, but the available ecosystem of open-source hardware, open instruction-set architectures, and open-source software toolchain components, have changed the potential structure of future codesign activities. Simultaneously, the increasing complexity of software, incorporation of AI, and use of modern programming languages and parallel-programming models, are evolving the software side of the codesign equation. This workshop will explore opportunities to reimagine our future codesign methodologies for hardware and software relevant to scientific computing.

[1] Chipmaking is being redesigned. Effects will be far-reaching, Economist, January 23, 2021.

Trends and Opportunities

Heterogeneous acceleration and architectural specialization — whether they are commercial designs (evolutions of GPU or CPU technologies), emerging reconfigurable hardware or bespoke architectures that are customized for specific science applications — optimize hardware and software for particular tasks or algorithms and enable performance and/or energy efficiency gains that would not be realized using general-purpose approaches. These long-term trends in the underlying hardware technology (driven by the physics) are creating daunting challenges for maintaining the productivity and continued performance scaling of HPC codes on future systems. However, the opportunity is for powerful new approaches to hardware/software codesign to specialize future computing environments to emerging DOE applications in both HPC, edge computing, and data processing for experimental sciences.

Overall, there is strong consensus that the tapering of Moore's Law will lead to a broader range of accelerators or specialization technologies than we have seen in the past three decades. Examples of this trend exist in smartphone technologies, which contain dozens of specialized accelerators co-located on the same chip; in hardware deployed in massive data centers, such as Google's Tensor Processing Unit (TPU), which accelerates the Tensorflow programming framework for ML tasks; in field-programmable gate arrays (FPGAs) in the Microsoft Cloud used for Bing search, AI/ML and other applications; and a vast array of other deep learning accelerators. The industry is already moving forward with production implementation of diverse acceleration in the AI and ML markets (e.g. Google TPU, Nervana's AI architecture, Facebook's Big Sur) and other forms of compute-in-network acceleration for mega-data centres (e.g. Microsoft's FPGA Configurable Cloud and Project Catapult for FPGA-accelerated search). Even before the explosive growth in the AI/ML market, system-on-chip (SoC) vendors for embedded, Internet of things (IoT), and smartphone applications were already pursuing specialization to good effect. Shao et al. from Harvard University tracked the growth rate of specialized accelerators in iPhone chips, and found a steady growth rate for discrete hardware accelerator units, which grew from around 22 accelerators for Apple's 6th-generation iPhone SoC to well over 40 discrete accelerators in their 11th-generation chip, and Apple's recent move to the Arm-based M1 chip that offers seamless use of many heterogeneous accelerator cores that outperforms the leading-edge Intel x86 designs that it replaced.

There have also been demonstrated successes in creating science-targeted accelerators such as D.E. Shaw's Anton, which accelerates molecular dynamics (MD) simulations by more than 180 times that of contemporary high-performance computing (HPC) systems, and the GRAPE series of specialized accelerators for cosmology and MD. A recent International Symposium on Computer Architecture workshop on the future of computing research beyond 2030 (http://arch2030.cs.washington.edu/) concluded that heterogeneity and diversity of architecture are nearly inevitable given current architecture trends. This trend toward co-packaging of diverse 'extremely heterogeneous' accelerators is already well under way.

On the application side, proxy apps are the tool most commonly used to facilitate codesign. HPC applications have been well represented in this space. However, complex workflows are becoming more prevalent. Often mixing many HPC applications, data analysis, and machine learning. In addition, AI and edge computing is growing in importance and making sure these applications are represented will be key to optimizing hardware and software tradeoffs at the core of codesign engagements. The methodologies used to program portable applications, such as using C++ frameworks like Kokkos and RAJA, along with advances in programming languages and tools, are changing how even large-scale applications can evolve. These changes are also key to understanding future codesign opportunities.

Emerging Transformative Technologies

There are a number of emerging technologies that have the potential to dramatically change the face of codesign by making it more accessible, affordable, and faster.

- **Chiplets**: Conventional SoCs co-integrate heterogeneous circuits onto a single die, which is very expensive as die-sizes grow. A chiplet integration strategy breaks the components into pieces that can be fabricated onto much smaller dies and then co-integrates those smaller dies onto a common interposer using 2.5D integration methods. The recently emerging chiplet approach offers a faster and less expensive way to assemble various types of third-party chips, such as I/Os, memory and processor cores, in a package (a silicon motherboard). This approach can dramatically reduce the costs of codesigned specialized designs assembled from a library of pre-made building blocks.
- **Licensable IP for Server-class processors (the rise of Arm):** The dominant cost for chip production is the design and verification of the logic design, which traditionally must be amortized over the sales of many COTs chips (a large market where the chip is the commodity). However, in the embedded market, the intellectual property (IP) is the commodity (the IP being the verified design, which is the most expensive part) and creating products involves combining these IP blocks together onto a single chip or package. Traditionally SoC design methods have focused on low-power consumer electronics or high-performance embedded applications, but the emergence of server-class processor IP (ArmV8 & SVE) is offering capable double-precision floating point, 64-bit address capability, and options for high performance I/O and memory interfaces. The growing server-class licensable IP ecosystem could enable a new path to affordable flexibility to customize hardware for government processing needs.
- **Open Source Hardware and Open Silicon Compilers:** Most hardware design is proprietary, so the commodity is the chip. Licensable hardware IP (such as Arm and Cadence) has enabled 3rd parties to create their own customized designs, but licensed IP can still be expensive, and even with low-cost academic licensing, the results cannot be openly distributed to a broad research community. The emergence of open source hardware/IP such as RISC-V and open-source silicon compilers such as OpenRoads offer a path to reducing licensing costs for hardware design, accelerating hardware development, and democratizing hardware design[2]. With open source hardware,

[2] https://www.eetimes.com/open-source-its-not-just-for-software-anymore/#

non-commercial entities such as labs and academia can participate in hardware design and development.

- **Photonic Resource Disaggregation**: Disaggregated architectures that decouple memory from processors and accelerators allow for flexible node designs and represent a promising architecture shift that can meet the demands of next generation HPC and AI workloads. Photonics enables run-time specialization customization by disaggregating resources at the node/system scale and enabling custom assembly of nodes at application execution time from pools of system-scale resources. This capability is driven by recent technology advances such as Ayar Labs TeraPhy, ARPAe's ENLITENED, and DARPA PIPES -- continued evolution of these technologies could enable disaggregation that is efficient enough for state-of-the-art HPC systems.
- **Standardized Accelerator Interfaces:** The emergence of CCIX, Coherent PCIe, and CXL as an industry standard for co-integration of diverse heterogeneous accelerators offers an opportunity for multi-vendor heterogeneous integration to become main-stream. This offers HPC integrators the ability to tailor the delivery of heterogeneous accelerators at the system integration level rather than having to produce their own custom silicon -- reducing cost, time, and risk for delivering specialized hardware.
- **Advanced Hardware Description Languages and Hardware Generators**: One of the dominant costs for hardware development is design and verification. Emerging advanced hardware description languages, such as Chisel, PyRTL, and PyMTL bring modern programming language techniques such as inheritance, polymorphism, and strong type systems. Frameworks such as Aladdin[6] and others use automation enable more targeted accelerator design. By bringing these modern techniques to hardware design, languages can dramatically lower the cost of hardware design for architectural specialization for science. More importantly, these new expressive languages might enable hardware designers to bring applied mathematicians into the loop, which is essential for guiding targeted specializations.
- **Coarse Grained Reconfigurable Arrays** (the re-emergence of static-dataflow / reconfigurable computing): Examples include Samba Nova, GraphCore, Cerebras, and FPGA technologies in general. The underlying fabrics are all essentially static dataflow graphs, and in some cases, operate at clock-rates and area efficiency of custom silicon. However, programming this kind of hardware requires new ways to think about algorithm design (e.g., superpipelining).
- **Advanced Packaging Technologies for Heterogeneous Integration (HIR)**: The Heterogeneous Integration Roadmap (HIR) is an industry-led initiative for delivering performance improvements for electronic devices in the absence of transistor scaling (https://eps.ieee.org/technology/heterogeneous-integration-roadmap/2019-edition.html). Heterogeneous Integration refers to the integration of separately manufactured components into a higher-level assembly (System in Package – SiP) that, in the aggregate, provides enhanced functionality and improved operating characteristics. Heterogeneous Integration through SiP follows naturally from the conceptual vision to build large, complex systems out of smaller functions separately packaged, as described in Gordon Moore's 1965 paper. Heterogeneity and associated integration is far-reaching and can relate to materials, component type, circuit type, node, interconnect method,

and source or origin. The current roadmap shows a credible path to deliver the next five generations of performance doublings through heterogeneous packaging of diverse technologies.

- **Open Source and Extensible Compiler Frameworks**: the emergence of open-source compiler frameworks with intermediate representations that can be more easily extended (e.g. LLVM with MLIR), offer the potential for vendors to more easily accept complex HPC programs and transform these into optimized binaries for their heterogeneous hardware. Extensions such as MLIR further enable higher-level optimizations that can target accelerators while working with existing languages (C, C++, Fortran) that dominate the DOE’s HPC code portfolio.
- **Artificial-Intelligence-Integrated Applications**: science applications are increasingly integrating AI technologies, especially machine learning, both as part of “outer loop” workflows and as part as the “inner loop” of core algorithms. The merging of data-driven modeling into core aspects of HPC applications, fed by both experimental and simulation results, enables the use of new hardware-acceleration paradigms, such as those using reduced-precision arithmetic. Moreover, this integration is opening up new possibilities for multiscale modeling, adaptive simulation, and in-situ analysis.
- **Programming Abstractions and Languages**: modern application development is increasingly reliant on techniques made possible by modern programming languages. Many DOE applications now use C++ frameworks, such as Kokkos and RAJA, to enable portability across different accelerator technologies. Increasingly, application teams are investigating new paradigms and languages, from Legion to Julia, and some of these include use of just-in-time compilation to provide dynamic, high-performance specialization. These trends change what kind of design choices are practical in the context of future codesign activities.

Potential Approaches

Specialization is the most promising hardware technique for continuing to provide the year-on-year performance increases required by all users of scientific computing systems, but specialization needs to have a well-defined application target to specialize for and may not be applicable to all domains. This creates a particular need for the sciences to focus on the unique aspects of scientific computing for both analysis and simulation. Recent communications with computing industry leaders suggest that post-exascale HPC platforms will become increasingly heterogeneous environments. Heterogeneous processor accelerators — whether they are commercial designs (evolutions of GPU or CPU technologies), emerging reconfigurable hardware, or bespoke architectures that are customized for specific science applications — optimize hardware and software for particular tasks or algorithms and enable performance and/or energy efficiency gains that would not be realized using general-purpose approaches. These long-term trends in the underlying hardware technology (driven by the physics) are creating daunting challenges for maintaining the productivity and continued performance scaling of HPC codes on future systems.

Co-development of hardware and algorithms creates transformative new opportunities by cooperatively designing algorithms and hardware together. This requires tighter integration of applied mathematics experts with the hardware design teams, but tools that enable them to productively collaborate with the software and hardware designers are essential. The rapid disruption and change of the hardware and computing ecosystem admits new opportunities for codesign. These opportunities combine new hardware and algorithms, and can present transformative opportunities. In an era where specializing hardware to the application is the only means of performance improvement, the economic model for the design of future systems is going to need to change dramatically to lower design and verification costs for the development of new hardware. An enabling technology for this is the emergence of agile hardware production methods such as using chiplets. Rather than have a single large piece of silicon that integrates together all of the diverse accelerators comprising the customized hardware, the chiplets break each piece of functionality into a very tiny tile. These chiplets/tiles are then stitched together into a mosaic by bonding them to a common silicon substrate. This enables manufacturers to rapidly piece together a mosaic of these chiplets to serve the diverse specialized applications at a much lower cost and much faster turn-around. However, this approach falls down if the desired functionality does not already exist in the available chiplets. Perhaps in the future the ‘algorithm-driven hardware design’ and this chiplets approach might be able to meet in the middle to bring forth a new economic model that can enable productive architecture specialization for small markets, such as Shao’s vision for her Aladdin [5] integrated hardware specialization/design environment.

System Level Design for New Workflows. Photonics, compute in the network, new node types, and emerging complex workflows allow for innovation in system architectures. As emerging workflows incorporate, AI, multiscale approaches, and multiple applications into a single workflow machines with a single set of nodes may be suboptimal. Disaggregation enabled by photonics allow more flexibility in allocating compute needs. System architectures using nodes with multiple types, e.g. CPU only, CPU+GPU and AI accelerated all in the same system allow pieces of the workflow to run on nodes most efficient for them. These are just two examples of many of how high level system design could adopt new technologies for greater efficiency. While more complex, system design could lead to greater efficiencies, it will also require developing a software stack and understanding workflow requirements and high level system usage to properly balance a system. Codesign opportunities between system designers, performance modelers, and end users for how best to leverage new technologies could lead to more efficient systems of tomorrow.

Improvement in Performance Portability Approaches. Application developers are already struggling to keep their codes running well on multiple flavors of CPUs and GPUs, while enabling flexibility to move to other machines. Improvements in programming techniques to efficiently retarget codes with good performance to an increasingly diverse set of architectures is a likely emerging issue. Current approaches, such as RAJA and Kokkos struggle with long compile times and rely on programming techniques that require experts to implement. While large teams can afford experts, smaller teams often can not. Maintaining developer and scientist productivity and allowing all users to take advantage of the potential performance gains from new accelerators, is a challenge that will require rethinking or retooling performance portability strategies. Such strategies might include more-comprehensive abstraction frameworks; new programming tools and languages; just-in-time compilation technology and other mechanisms for dynamic specialization; intelligent runtime systems; and, advances in integrated monitoring and profiling.

Quantitative Tools of Codesign (ModSim). There is general consensus that the development of critical hardware/software technologies require quantitative tools of codesign along multiple dimensions. To be successful, such tools have to be applicable to design-ahead (in advance of implementation), and follow through to assisting optimizations during execution of complex workflows on the target systems. The ModSim capabilities have to be practical (in that the time to solutions for full system simulation under a realistic application workload has to be reasonably low), be accurate over a broad range of hardware/software architectures, and scalable as the system complexity and size increases. Successful tools of codesign need to consider the triad performance/power/reliability in an integrated fashion, and capture many of the boundaries up and down the hardware/software stack. Model generation done (semi)automatically is a lofty research goal, as is the interoperability of models. The nature of the workloads that are increasingly data-driven dynamic and irregular, require continued emphasis on dynamic modeling methods and tools used for optimization at runtime, introspection for runtime systems, or system control, monitoring, and optimization. Development of ModSim tools that are scalable, have common interfaces, use models interchangeably, and operate on a scale of time-to-solution vs accuracy is a key codesign technology.

Possible Research Directions (Why and How)

Here, we call out some of the key outstanding research issues in this space that can be addressed via an ASCR research program that would be needed to put these emerging technologies to work for DOE science. This list is not exhaustive, and the purpose of this workshop is to solicit community input to refine the research opportunities and their potential impacts on DOE computing for science.

- What programming systems (languages, compilers, runtime systems, hierarchical abstraction of software systems, libraries, etc.) facilitate effective codesign and why? Are there programming abstractions and paradigms that fundamentally improve the process of codesign?
- What kind of quantitative modeling, simulation and performance prediction tools are required in order to guide design choices in order to facilitate the codesign process?
- How can applied mathematicians be more tightly integrated into the design teams? What tools and methodologies can be employed to enable productive collaboration in this multi-disciplinary design cycle.
- How to co-design hardware and software security mechanisms that will allow HPC centers to implement security policies such that data providers can give usable access to their sensitive data sets to data scientists.
- What are the open source and community tools that will enable scalable hardware development, evaluation and optimization (including performance simulation, verification

and testing)? To what extent do the tools need to support flexible design strategies, the ability to mix/match different components, and be sufficiently easy to use for a broader range of the HPC community?

- What are the key prospective system performance shortcomings for critical DOE applications (e.g. memory capacity, microsecond real-time inference, ...) given the projected technology roadmap for the next decade? What exploration is helpful in ensuring there will be viable options? How can they be addressed with codesign?
- What are the new ways that codesign can create technologies, system designs, and work with technology vendors, integrators, and the ecosystem to enable the DOE to purchase and deploy systems that better meet its needs? That is, what are the new ways that codesign can be productive for DOE ends, given the rapidly changing landscape?
- How does the broader system and application scope that comprise DOE's advanced scientific computing needs (including stream-based computation, machine learning, facilities, etc.) affect the appropriate scope for codesign, critical parties, and opportunity for transformative impact?
- How can methods for rapid hardware generation / integration enable new system-design paradigms?
- Edge-computing application cases such as distributed sensor arrays, processing-in-sensor, and processing-in-network may well demand more intensive specialization than traditional general-purpose computing due to environmental constraints. How can we create an environment that balances rapid codesign/specialization of systems and a productive end-user environment?
- How does one take advantage of resource disaggregation to serve the needs of HPC workloads, and what are the underlying requirements in terms of resource managers/scheduling, programming environments and security that is necessary to make disaggregation usable and productive for DOE applications?
- How best do we pull together multi-disciplinary teams to execute codesign successfully?
- How should we evolve ModSim capabilities to enable this new re-imagined codesign with tools that are accurate over a broad range of hardware/software architectures, and scalable as the system complexity and size increases?

There are also ancillary meta-questions that are not themselves research issues, but speak to the viability and sustainability of the future of computing.

Alternative Hardware Realizations and their Impact

The traditional DOE computing model is that DOE laboratories procure and maintain large HPC systems with users from other laboratories as well as non-DOE users often able to gain access. The "capability" systems at the NNSA laboratories, e.g., Trinity and Sierra, tend to run less varied workloads than the NNSA "capacity" systems or any of the supercomputers at the SC laboratories. In general, the more focused the workload, the more opportunity there is to employ codesign techniques. In the limit, systems like Anton and GRAPE target exclusively a single domain — molecular-dynamics — and therefore are able to codesign applications, system software, and hardware to great benefit.

Although DOE is not in the business of fabricating HPC components, there are numerous existing opportunities within DOE data centers to exploit codesign techniques. At the procurement stage, supercomputers can be selected based on how well they match applications' needs, taking into consideration the ability for applications to adapt to the hardware. At the configuration stage, firmware and system software — and sometimes hardware configuration such as I/O node placement or network topology — can be tweaked to improve the performance of critical applications, and applications can adjust their parameters to the capabilities and limitations of the underlying hardware and software. Looking to the future, a number of transformative opportunities exist for DOE to collaborate much closer with hardware vendors, such as via processors based on chiplets, for which DOE could design or help design a small unit of DOE-centric logic that could be incorporated into a base processor.

DOE operates a number of experimental facilities such as accelerators and light sources. Measurements taken on these facilities increasingly require substantial computational power for filtering and analysis. This suggests additional avenues for codesign that have not yet been thoroughly explored, involving experimental facilities, supercomputers and data storage, and scientific applications. Similarly, edge technologies such as smart sensors are becoming more widely used and may eventually work their way into core DOE usage models. The resource-limited and distributed nature of edge computing implies that this is another form of hardware that may be of interest to DOE and that can benefit from codesign of all system components: edge devices, back-end supercomputers, applications, and, where possible, networking.

References

[1] Extreme Heterogeneity 2018 - Productive Computational Science in the Era of Extreme Heterogeneity: Report for DOE ASCR Workshop on Extreme Heterogeneity. https://doi.org/10.2172/1473756

[2] Basic Research Needs for Microelectronics: Report of the Office of Science Workshop on Basic Research Needs for Microelectronics, 2018. https://doi.org/10.2172/1616249

[3] Heterogeneous Integration Roadmap (HIR) 2019 edition. https://eps.ieee.org/technology/heterogeneous-integration-roadmap/2019-edition.html

[4] "There's Plenty of Room at the Top: What will drive computer performance after Moore's Law," C. Leiserson, N. Thompson, et al. , Science 05 Jun 2020: Vol. 368, Issue 6495, eaam9744, DOI: 10.1126/science.aam9744

[5] Shao, Yakun Sophia, et al. "Co-designing accelerators and soc interfaces using gem5-aladdin." 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2016.

[6] Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, David M. Brooks: The Aladdin Approach to Accelerator Design and Modeling. IEEE Micro 35(3): 58-70 (2015)

Lessons from Application and Programming Model Codesign

Alex Aiken Stanford/SLAC aaiken@stanford.edu

Pat McCormick LANL pat@lanl.gov

Elliott Slaughter SLAC eslaught@slac.stanford.edu

Topics: Applications and Programming Systems

Over the last several years the authors have worked on a number of projects to codesign applications with programming models. The applications have ranged from simulations (e.g., S3D, a combustion chemistry simulation [1]), to data analytics (ExaFEL, a project to build a realtime data analytics software stack for future light sources [2]), to machine learning [3]. The programming models have all been tasked-based, and mostly (but not exclusively) versions of the Legion programming model [4]. While we are fans of task-based programming models, most of our points below are more generally applicable. We relate a number of lessons we have learned from our codesign experiences, grouped together into three broad themes: *The Level of Abstraction, Specialization,* and *Interfaces.* The first two outline the codesign challenges we see, while the last highlights an opportunity to make progress.

The Level of Abstraction. One characteristic of DOE applications is the desire for performance portability across multiple platforms. As a result, a degree of flexibility is needed in codesign solutions not found in, say, embedded systems. Perhaps counterintuitively, codesign for portability is usually more successful if the targeted user interface is at a higher rather than lower level of abstraction. Done well, a higher level of abstraction means that fewer details of the implementation are fixed in the user specifications (which typically take the form of application code), enabling more scope for optimization and more opportunities to codesign among the different software and hardware components sitting below that interface. As an example, Liszt is a high-level DSL for mesh computations [5]. One of the key features of Liszt is that instead of emphasizing traditional array indexing to express relationships among different data elements (e.g., cell[2*i + 3]), Liszt encourages using user-defined fields (e.g., cell.neighbor). The resulting code is easier to write, to read, and for an optimizing compiler to analyze, which makes it possible to generate efficient code for both CPUs and GPUs despite their significant architectural differences. It is not obvious that higher levels of abstraction and better performance should go together, but in our experiences with codesigning the elements of a software stack to work on a variety of very different platforms, raising the level of abstraction has been the only approach we have used that has successfully lead to portable performance.

Specialization. The goal of codesign is to co-specialize multiple pieces of a system for maximum mutual advantage. CUDA has been codesigned with NVIDIA GPUs, and to greater or lesser degrees OpenMP has been codesigned with multicore processors, MPI has been codesiged with high performance networks, FPGA software stacks have been codesigned with FPGAs, and all of these have also been codesigned with applications. Each is a success within its domain, but there are cautionary lessons. First, a major investment was required in each case, which implies that the user community needed to be large for these efforts to be successful. For codesign efforts benefiting smaller groups, as would be the case with most codesign opportunities in the DOE, codesign costs must be lower; we expand on this point below. Second, the resulting specialization has created a situation where the individual components of a full supercomputer (network, nodes, and accelerators) have well-tuned interfaces, but because codesign was rarely done across components the interface to the complete system is low-level and complex to use. Thus, the natural evolution of supercomputer systems, where each component's interface has been codesigned only with its associated hardware, has resulted in an overall design that applications still struggle to fully exploit.

Interfaces. How can we reduce the cost of codesign to make it realistic to routinely cover at least some aspects of an application and the associated system software and hardware?   The only answer is to limit the scope. That does not necessarily mean limiting the potential upside of such projects. We believe that many codesign problems, at least within the DOE community, share requirements. In particular, a set of standardized system software interfaces, at a higher level than today's conglomeration of MPI/OpenMP/CUDA/etc. would both simplify application development while expanding the scope for optimization and codesign at lower levels and preserving future portability. Now is the right time to develop those abstractions, as it has become clear that a distributed system of multicore nodes with multiple accelerators will be the standard computing platform for many years to come. Task-based abstractions, which are already widespread in the data analytics and machine learning communities (e.g., TensorFlow, Pytorch, and Spark are all task-based), are one way to provide a higher-level and more uniform interface for applications that would also enable aggressive codesign of the layers underneath and provide applications with improved portability and performance. While there may be other viable options besides task-based approaches, we believe software abstractions that are independent of or at least agnostic to specific underlying hardware technologies are needed to allow more codesign across different parts of systems rather than only narrowly within a specific component vendor's hardware and software.

[1] S3D-Legion: An Exascale Software for Direct Numerical Simulation of Turbulent Combustion with Complex Multicomponent Chemistry. S. Treichler, M. Bauer, A. Bhagatwala, G. Borghesi, R. Sankaran, H. Kolla, P. McCormick, E. Slaughter, W. Lee, A. Aiken and J. Chen. In T. Straatsma, et al., editors, Exascale Scientific Applications: Scalability and Performance Portability, CRC Press, 2017.

[2] Fluctuation X-ray Scattering Real-Time App. A. Dujardin, E. Slaugther, J. Donatelli, P. Zwartk, A. Perazzo and C. H. Yoon. Proceedings of the Python in Science Conference, 2020.

[3] Beyond Data and Model Parallelism for Deep Neural Networks. Z. Jia, M. Zaharia and A. Aiken. Proceedings of the SysML Conference, 2019.

[4] Legion: Expressing Locality and Independence with Logical Regions. M. Bauer, S. Treichler, E. Slaughter and A. Aiken. Proceedings of the Conference on Supercomputing, 2012.

[5] Liszt: A Domain Specific Language for Building Portable Mesh-based PDE Solvers. Z. DeVito, N. Joubert, F. Palacios, S. Oakley, M. Medina, M. Barrientos, E. Elsen, F. Ham, A. Aiken, K. Duraisamy, E. Darve, J. Alonso, P. Hanrahan. Proceedings of the Conference on Supercomputing, 2011.

Missing pieces for accelerating co-design to meet the growing compute demand post Moore's Law

Tim Ansell
tansell@google.com
Software Engineer
Google

Aaron Cunningham
aacunningham@google.com
Open Hardware Policy
Google

Kevin Jameson
kajameson@google.com
Field Representative
Google Cloud

Miles Euell
mileseu@google.com
Field Representative
Google Cloud

Scott Frohman
sfrohman@google.com
Federal Sales Executive
Google Cloud

The Challenge

Nathan Myhrvold's Four Laws of Software [1] state that software expands to consume any available compute power. This phenomenon has enabled the computer industry to create ever more complex and compelling products with fewer software engineering resources [2, 3]. Recently, Moore's Law which has enabled this increasing expansion has slowed while the demand for ever more computing power has not.

The negative impact of previous slowdowns around Moore's Law on unprepared organizations is documented at [4]. Thus, forward looking organizations have already started to invest in the creation of hardware accelerator ASICs to tackle this growing demand. However, the growing cost of IC creation, as shown in Figure 1, and the gap between theoretically achievable transistor density and the actual realized transistor density, as shown in Figure 2, represent a huge challenge.

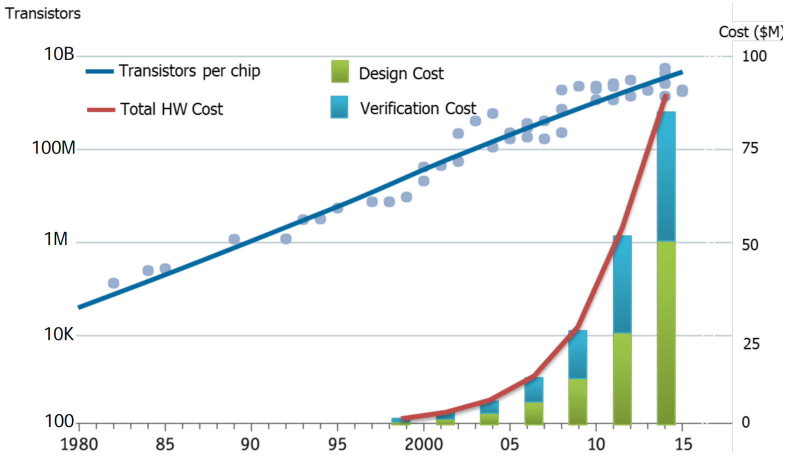


Figure 1: *The growing cost of IC creation.* [5]

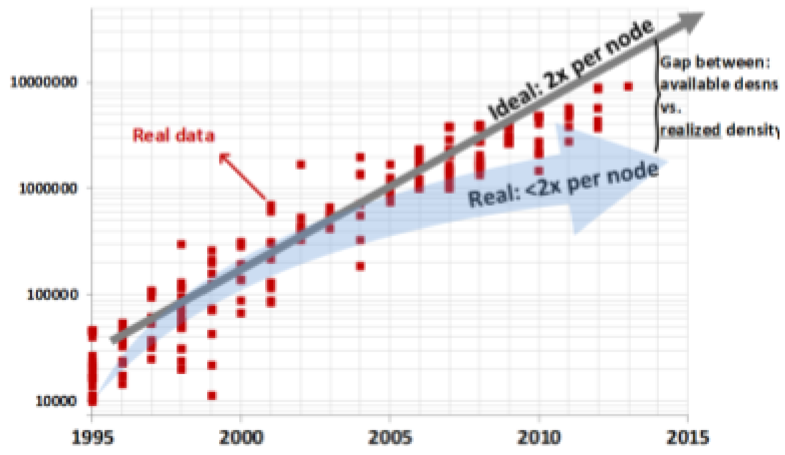


Figure 2: *The growing EDA and Design capability gap.* [6]

The Opportunity

Figure 3 and figure 4 demonstrate how Google's TPU, an ML hardware accelerator ASICs, is able to tackle the increasing compute demand for ML training at scale [7] and are up to 27x faster at 38% lower cost than GPUs[8].

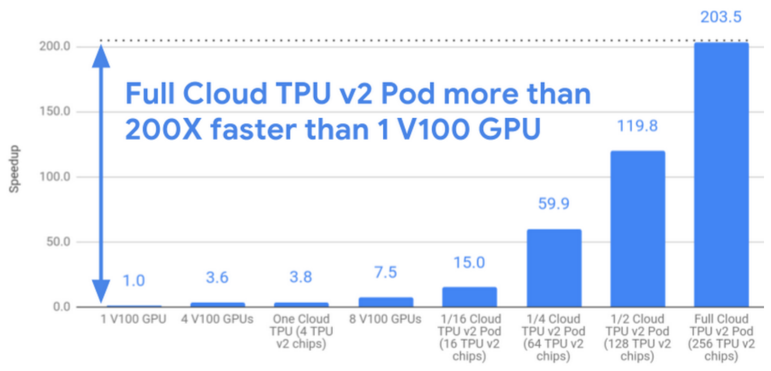


Figure 3: *ResNet-50 Training Speedup (Baseline: 1 V100 GPU).* [8]

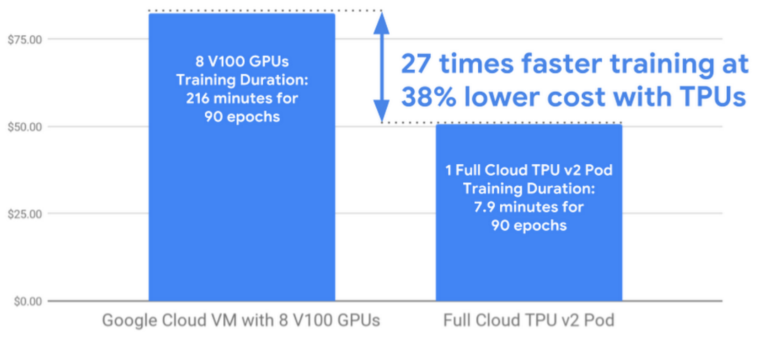


Figure 4: *ResNet-50 Training Cost Comparison.* [8]

Being able to reproduce and build on top of other researchers' results is a fundamental aspect required for a reliable knowledge sharing ecosystem [9] required to tackle challenges in this space. All of [10], [11], and Google's own research philosophy [2], reinforce this principle. The fact that research and development in the IC creation space typically has restrictive

NDAs around data and closed source proprietary tools was highlighted by Andrew Kahng in [6, 12]. Even groups like Google are mostly unable to publish in a reproducible way, making the creation of hardware accelerators very inaccessible.

To solve this issue, Google is working to fill in the current missing pieces required to develop fully open source IC development through the release of and collaboration with fully open source code bases and datasets as documented in [13]. Examples of the powerful effect on both research activity and the ability to directly transfer research to productionzation, has been previously found recently in the ML and ISA spaces.

The current explosion of research in the machine learning space [14] shows the effectiveness of fully open source code and dataset releases. With Google releasing TensorFlow in 2015 [15] and the open investments from companies like Facebook, Nvidia and Microsoft, it became possible to use the same technology for research and ML across numerous products. This type of high speed technology transfer and integration of research and development teams are two areas that the IC design industry has struggled to achieve as highlighted in [12, 16]. Parallels can be drawn between EDA tooling and TensorFlow, PDK Data and ML datasets, and IP & Libraries as ML fields of research.

RISC-V is another eloquent example of the success of the aforementioned open dataset and code trend [17]. Freed from the previous restrictions around proprietary licenses to the instruction set architecture (ISA), legions of designers have contributed to defining novel implementations and have built on top of each others' work. This has also led to direct technology transfer from the research space to industry adoption at an outstanding pace [18].

Why now?

With the slowdown of Moore's Law and the recent investments to bring IC manufacturing back to the US [19][20], it is the right time for the Department of Energy to extend the existing policy of supporting open source software solutions to also focus on open hardware acceleration ICs. The global effort Google is spearheading includes investigators from all of industry, academia, start-ups, and foundries, and would welcome contributions and collaboration from entities like the US Department of Energy.

Impact of success?

This investment in fully open IC design will help provide a needed push towards the democratization of hardware acceleration development at bleeding edge manufacturing nodes. This renewed open-source wave to break down the barriers of EDA tooling and ultimately hardware design will result in a thriving ecosystem with multiple blossoming projects that enable the continued acceleration of both scientific and commercial possibilities.

References

[1] Nathan Myhrvold. The future of software, the software industry, and Windows '47. In *ACM '97: The Next 50 Years of Computing*, ACM '97, New York, NY, USA, 1997. Association for Computing Machinery.
[2] A. Spector et al. Google's hybrid approach to research. *Commun. ACM*, 55(7): 34–37, July 2012.
[3] L. Hatton et al. The long-term growth rate of evolving software: Empirical results and implications. *Journal of Software: Evolution and Process*, 29(5):e1847, 2017.
[4] Neil Thompson. The economic impact of Moore's Law: Evidence from when it faltered. *SSRN Electronic Journal*, 01 2017.
[5] Andreas Olofsson. Intelligent Design of Electronics Assets (IDEA) & Posh Open Source Hardware (POSH), September 2017. URL https://www.darpa.mil/attachments/eri_design_proposers_day.pdf
[6] Andrew B. Kahng. Reducing time and effort in ic implementation: A roadmap of challenges and solutions. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, New York, NY, USA, 2018. Association for Computing Machinery.
[7] N. P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. 2017. URL https://arxiv.org/pdf/1704.04760.pdf
[8] Urs Hölzle. MLPerf benchmark establishes that Google Cloud offers the most accessible scale for machine learning training. Google Cloud Blog. December 2018. URL https://j.mp/google-ml-perf-benchmark
[9] Roger D. Peng. Reproducible research in computational science. *Science*, 334 (6060):1226–1227, 2011.
[10] National Academies of Sciences Engineering and Medicine. *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC, 2019. URL https://www.nap.edu/catalog/25303/reproducibility-and-replicability-inscience
[11] Victoria Stodden. Enabling reproducible research: Open licensing for scientific innovation. *Science*, 13, 03 2009.
[12] Andrew B. Kahng. Looking into the mirror of open source. In *2019 IEEE/ACM International Conference on Computer-Aided Design,* ICCAD '19, pages 1–8. IEEE, 2019
[13] Tim Ansell et al. The Missing Pieces of Open Design Enablement: A Recent History of Google Efforts: Invited paper, IEEE/ACM, *International Conference on Computer-Aided Design*, ICCAD '20, November 2–5, 2020, Virtual Event, USA. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3400302.3415736
[14] Y. Duan et al. Artificial intelligence for decision making in the era of big data - evolution, challenges and research agenda. *International Journal of Information Management*, 48:63 – 71, 2019. URL http://www.sciencedirect.com/science/article/pii/S0268401219300581
[15] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org
[16] Andrew B. Kahng. Open-source eda: If we build it, who will come? In *Proc. 28th IFIP/IEEE International Conference on Very Large Scale Integration,* VLSI-SoC, 2020.
[17] S. Hoover. Unleashing the potential of open-source silicon. Invited talk, *VSDOpen Conference*, 2019.
[18] Kim McMahon. RISC-V International Reports Another Strong Year of Growth with New Technical Milestones, Educational Programs, RISC-V Adoption and More. December 2020. URL https://j.mp/risc-v-report-2020
[19] DARPA Electronics Resurgence Initiative. April 2020. URL https://www.darpa.mil/work-with-us/electronics-resurgence-initiative
[20] CHIPS for America Act. H.R. 7178, 116th Congress. June 2020. URL https://www.congress.gov/bill/116th-congress/house-bill/7178

HW/SW Codesign for Disaggregated Memory Architectures: Opportunities and Challenges

Amro Awad
North Carolina State University
ajawad@ncsu.edu

Simon Hammond
Sandia National Laboratories
sdhammo@sandia.gov

Clayton Hughes
Sandia National Laboratories
chughes@sandia.gov

***Abstract*—Resource disaggregation, enabled by fast optical interconnects, allows novel design options and flexibility for future computing systems. Future DOE computing systems can potentially benefit from disaggregated memory architectures where various processing elements, perhaps from different vendors, can be seamlessly integrated and use shared memory pools. Memory disaggregation has the potential to enable interesting programming models, heterogeneous compute architectures, and minimize the memory under-utilization issue in existing conventional HPC systems.**

Leveraging disaggregated memory architectures in DOE systems requires a co-design strategy that includes run-time libraries, software interfaces, operating system and hardware. Only with that, future DOE systems can exploit the performance, reliability and energy efficiency promised by disaggregated memory architectures. In this white paper, we reflect on our experience in exploring disaggregated memory systems and identify major gaps where a hardware/software codesign strategy is most needed for such a new architecture.

Topic: architectures, emerging technologies, codesign methodology

I. Introduction

Conventional high-performance computing (HPC) systems closely couple memory modules with compute nodes. These architectures are most effective when an application's data can be partitioned across nodes, each partition fits well in the nodes' local memories, and minimal communications are needed between nodes. Typically, communication between nodes is carried over the network interface or fast customized interconnects, such as InfiniBand, through explicit message passing. For instance, in today's HPC systems, message passing interface (MPI) is commonly used to achieve synchronization and collective operations in applications run over multiple nodes. However, conventional HPC architectures suffer from (1) memory underutilization as other applications cannot effectively leverage unused memory space in nodes reserved for applications with low memory footprint (2) applications with memory demand (per node) exceeding node's memory capacity will fail to run unless expensive and slow swapping to/from network-attached storage is enabled (3) data-dependant workloads such as data analytics are difficult to partition across nodes and thus can cause significant communication overheads and data redundancy in conventional HPC systems (4) Finally, integrating accelerators or different types of processing elements in conventional HPC systems needs to leverage explicit message passing to other nodes or tightly couple it with a CPU that acts as its gateway to other nodes.

The emergence of high-bandwidth and ultra-fast interconnect technologies, such as optical networks, bring in new opportunities to disaggregate memory systems and resources, and connect them to processing elements (PE) through shared fabrics. As shown in Figure 1, such architecture allows various types of processing elements to access shared physical memory pools. Such memory pools could be petabytes of non-volatile memories (NVMs), as envisioned by HPE Labs in "The Machine" project, or simply a combination of pools from different memory technologies with various density, performance and data retention characteristics.

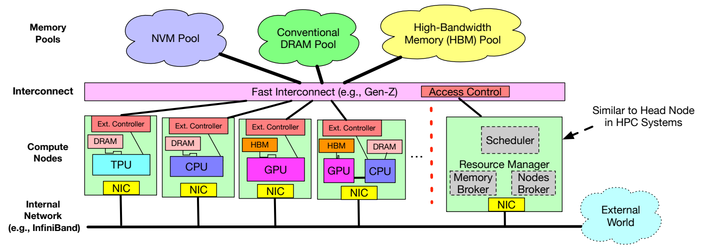


Fig. 1: Example disaggregated memory system.

Several memory-semantic protocols and standards, e.g., Gen-Z and CXL, define the memory interfaces that need to be implemented at each PE to allow it to access/request shared memory pools. Obviously, such architecture enables seamless ways to integrate different types of accelerators and processors in new programming models and interfaces that leverage shared memory pools. In other words, any PE that implements the memory semantic protocol used can be integrated and leverage the shared memory pools to cooperatively operate on shared data. Applications and use cases of such architecture can vary from data analytics operating on terabytes of data (e.g., large social graphs) to regular HPC workloads that can benefit from simple shared-memory programming model across heterogeneous compute nodes. Finally, operations such as job migration, snapshotting (for checkpointing), etc., can effectively exploit such architecture to boost their performance and reduce the resulting traffic.

However, such disaggregated memory architectures introduce several challenges that are yet to be solved. First, scalable memory management for such large shared memory pools is a key for any successful deployment of such systems. Second, it is important to understand how to port existing DOE workloads to such a new architecture to best leverage the large amount of memory each job can be provided with, and what heuristics to use to identify the best configuration and set of PEs to use. Third, how to effectively allocate and manage memory pages from different technologies, and what selection criteria to use. Fourth, with such shared physical memory shared between different PEs, possibly from different vendors, we need to carefully implement external access control to enforce isolation between nodes such that a malicious/buggy node (or running malicious/buggy code) cannot compromise the whole system. Fifth, how to ensure quality of service (QoS) and guarantee some level of performance isolation between different PEs, without underutilizing bandwidth or memory bandwidth due to static partitioning schemes. Finally, with such an architecture that integrates memory pools with storage characteristics (i.e., NVMs), how can we leverage these pools for dual use, i.e., checkpointing/filesystem and regular memory accesses with the able to distinguish between these traffics (memory accesses might be of higher priority).

In this white paper, we briefly discuss these challenges and how a co-design strategy can help exploiting the otherwise untapped potential of disaggregated memory architectures.

II. Challenge

In this section, we discuss the unique challenges of disaggregated memory systems, and how a co-design strategy can help overcome them.

Scalable Run-time Memory Management: designing a flexible memory management scheme that allows memory sharing between different PEs, allocation from different pools, and run-time monitoring of contention points, requires hardware support to furnish such run-time information and software support or library for managing shared pools in a scalable fashion. It is imperative to provide the ability to parallelize the implementation of the memory manager and make it scale for larger rate of allocation requests from different nodes. Moreover, the memory manager will be partly responsible for page migration and insertion/eviction policies from scarce but fast memory pools (e.g., high-bandwidth memory).

Access Control: due to the integration of different types of PEs, possibly from different vendors, relying on internal (inside each PE) access control enforcement can significantly increase the attack surface and the impact of potential bugs or security vulnerabilities, mainly due to the shared directly-accessible physical memory[1]. Therefore, it is important to employ external system-level access control enforcement that is managed through the memory broker/manager. Moreover, it is important to guarantee that no PE can directly change its own access control metadata.

Porting DOE Workloads: to be able to leverage disaggregated memory systems, it is critical to understand the atomicity guarantees (e.g., to implement barriers and locks) across nodes, memory consistency model, and coherence support in disaggregated memory systems. The ability to run DOE workloads on systems with partial support for such essential primitives requires codesigning the workloads in the context memory access parallelism in each PE and across PEs. For instance, the impact of memory ordering around critical sections that could be run by different PEs, perhaps with limited hardware support for coherence, is yet to be understood.

Rethinking Existing Checkpointing Mechanisms: with the availability of NVM memory pools in such systems, it is important for future DOE workloads to leverage such persistence features to optimize current checkpointing mechanisms for fast recovery from crashes or errors. By leveraging a careful co-design strategy, checkpointing can occur immediately by leveraging hardware support that support run-time remapping for memory pages, in a way similar to copy-on-write (CoW) in current systems. By doing that, checkpointing can be achieved with much lower number of writes and hence much faster execution.

Quality-of-Service (QoS): in addition to the memory-side support for QoS, run-time environment and resource allocation frameworks should accurately set the priorities and the class of services anticipated for particular workloads/tasks. Such class of service should be considered in both scheduling, and run-time performance isolation.

Application-Specific Performance Optimizations: applications with particular needs, e.g., high memory bandwidth, can potentially provide hints to the memory allocation library. Similarly, hints for prefetching, access granularity from fabric-attached memory, cache bypassing, etc., all can be leveraging the application down to the PE and the disaggregated memory system.

III. Opportunity

The development of disaggregated memory has the potential to profoundly impact DOE's scientific delivery. The most valuable assets that DOE owns are our experimental data sets and the knowledge of its research staff. Disaggregated memory systems would allow DOE users to build complex, high performance workloads that can operate on even the largest experimental data sets without requiring large-scale filesystem operations that are major bottleneck in existing HPC systems. By building a new approach to large-scale data storage including fine-grained access models, advanced programming approaches and completely rethinking checkpoint mechanisms, DOE users will be able to see order of magnitude or higher improvements in end-to-end scientific workflows, not just the compute/simulation phases that many users will think of when discussing codesign. Reimagining hardware systems means reimagnining how we store, process and compute on data, disaggregated memory systems are the future for how we will process vast increases in data from experimental facilities into the next decade.

IV. Timeliness

The emergence of open-source architectures, open standards for memory semantic protocols (e.g., Gen-Z and CXL), and fast interconnect technologies make such new architecture promising. Moreover, due to the heterogeneous nature of workloads and the wide use of domain-specific accelerators, flexible integration and codesign approaches together can make the best use of such a new architecture.

References

[1] Vamsee Reddy Kommareddy, Clayton Hughes, Simon David Hammond, and Amro Awad. Deact: Architecture-aware virtual memory support for fabric attached memory systems. *HPCA*, 2021.

Microarchitecture-Centric Codesign for Reduced TCO and High-Performance DoE Systems

Amro Awad
North Carolina State University
ajawad@ncsu.edu

Eric Rotenberg
North Carolina State University
ericro@ncsu.edu

***Abstract*—The emergence of open-source instruction set architectures (ISA), such as RISC-V, present new opportunities for rethinking microarchitecture innovations in a hardware/software co-design fashion. In this white paper, we discuss several microarchitectural innovations that can significantly benefit from codesign activities to lower the total cost of ownership (TCO), improve the performance, reliability and flexibility in future Department of Energy (DoE)'s workloads. In particular, we discuss how several ideas that improve system's reliability, minimize control flow hazards, and reduce average memory access time can benefit from co-design approach, and how can this be tuned for DoE workloads.**

I. Introduction

Today's processors overlap the execution for large window of instructions to hide the latency of cache misses (memory accesses). However, with such increase in the window of instructions, the impact of branch mispredictions on performance becomes significant. Additionally, with the increasing gap between processor speed and memory latency, the impact of memory accesses on performance becomes significant[6]. Prior works in the computer architecture community, such as *Slipstream processors*[6], [7] and *dual-core execution (DCE)* [9], advocate for a pre-execution strategy which allows early identification of taken control flow paths and resolve load/store addresses long before their execution. By relying on pruning techniques that allow the creation of a lightweight accurate leader thread, the original/follower thread can leverage the resolved branch predicates and pre-loaded data to avoid long delays, which would have occurred otherwise.

The idea of pre-execution can potentially bring in significant performance advantages for several DoE workloads, especially those with irregular memory accesses and hard-to-predict branches (i.e., as in deeply-nested for-loops in scientific workloads). However, such pre-execution often requires additional cores to run leader threads, and thus can reduce the total computational capacity of the system. Understanding the impact of run-ahead/pre-execution schemes in the context of high-performance computing (HPC) systems and DoE workloads is imperative. However, since such approaches are mainly optimized for the common case with a single-node in mind, applying such schemes for DoE workloads would benefit from a co-design strategy. For instance, the ability to opportunistically leverage idle cores in the node by enabling/disabling pre-execution where beneficial requires strong coordination between the run-time environment, compiler generated hints/primitives, and hardware. Moreover, rethinking the hardware such that it additionally integrates low-power tiny cores which can be leveraged for pre-execution requires support from the run-time library and resource allocation manager.

As demonstrated in prior work for control flow decoupling[5], using the compiler to decouple predicates' pre-calculation, and expose such predicates to the hardware can effectively eliminate significant portion of the performance overheads due to branch mispredictions. We believe that similar opportunities exist for various runahead schemes, as also demonstrated in prior work[1], [3]. It is also important to understand how the compiler optimizations, run-time environment, and hardware can be co-designed such that it improves the performance without underutilizing resources or incurring unacceptable power/area overheads. Moreover, investigating the unique challenges for implementing such approaches in open-source instruction set architectures (ISAs) and the required new instructions (if any), in addition to the needed support from the software stack (including compiler) is both timely and important.

In addition to performance improvement techniques, the reliability of memory systems is perhaps among the most challenging aspects of DoE's HPC systems. However, the reliability guarantees has been always restricted by (1) existing support in the processor chip; and (2) the reliability support in the memory module itself (additional chip). Generally, memory modules equipped with reliability support are typically much more expensive than their conventional (no reliability support) counterparts. Most processor vendors avoid introducing any additional logic for managing and handling reliability metadata, and thus limiting reliability support only to memory modules readily provide corresponding metadata along with the data through additional chips and wide buses. Such limitation from the processor side limits the options system integrators can choose the memory modules from; they must support extra chips for reliability. However, with the opportunity to redesign hardware, memory reliability can be provided by processor-side support even when using conventional memory modules. Prior work explored how processor-only support can provide memory reliability for unmodified memory modules[8], however without exploiting system-level optimizations that can be leveraged by the operating system, e.g., relaxing ECC for fault-tolerant applications. With co-design strategy, we can explore a software-defined memory reliability support that can be leveraged by the operating system and run-time environment to flexibly choose the capacity/reliability/performance marks.

In this white paper, we will discuss a systematic approach to rethink the aforementioned optimizations and innovations in a co-design approach that enables flexibility, low power, high performance, and reduced total-cost of ownership (TCO).

II. Research Challenges

Most DoE workloads are multi-threaded and highly-parallel, thus adopting run-ahead/pre-execution techniques can potentially reduce the number of the threads can run on a node, while improving the performance of each thread individually. Thus, a co-design approach can potentially leverage run-time information and compiler heuristics to decide upon reduced number of threads for higher performance of each thread (due to run-ahead/pre-execution) or to disable run-ahead/pre-execution for the sake of running the maximum number of threads a node can run efficiently. Moreover, in a heterogeneous core micro-architecture, e.g., ARM's big-little, deciding upon using tiny cores for running actual threads or helper threads (in case of pre-execution) requires heuristics about the potential improvement from pre-execution, and also an understanding of potential contentions can result from running other threads/applications in such tiny cores.

While shifting the memory reliability support to the processor side allows high-flexibility in choosing memory modules, hence reducing TCO, it comes at the cost of potential performance degradation and storage overheads. Similar to ECC-memory, there must be error correction and detection codes associated with the protected data, and thus managing such metadata can add latency and storage overheads that are proportional to the strength of the reliability support scheme. Thus, such reliability support should be flexible such that it adjusts to the level of protection needed (e.g., Raw Bit Error Rate (RBER) of the protected memory modules), and the application's reliability requirement. For instance, if the OS or run-time environment can adjust the type of protection (hence its storage and performance overheads) for certain nodes or workloads, that can bring in significant improvements while retaining the ability to provide strong reliability when needed.

III. Research Approach

To effectively evaluate codesign options, while able to explore the design space in a fast manner, we can use a highly-flexible microarchitecture model in fast simulators, such as Vanadis in SST simulator[4], to understand the system-level aspects and policies to manage and adjust such mircoarchitecture features. Moreover, it will help us understand the impact of compiler or run-time hints enabled through our codesign strategy. From there, we can adjust our support at system software and identify the needed information (and effective heuristics) to expose to the hardware. By modeling heterogeneous core architectures, and different allocation policies and threading options, we can better understand the impact of pre-execution in such architectures and how to tune it for DoE workloads. We can also leverage existing frameworks for automatic generation of synthesizable superscalar cores, such FabScalar[2], to optionally enable support for pre-execution, and use such low-level model to calibrate our timing model in the high-level simulator. In all these evaluations and investigations, DoE workloads will be used.

For processor-side memory reliability support, we will leverage existing memory models in high-level simulators such as SST, and investigate different scenarios for co-running workloads with different memory reliability requirements. We will also investigate how to architect software-defined memory reliability support that can be seamlessly configured and adjusted by the operating system. Finally, we will show how the compiler or run-time library can optionally provide hints for relaxed reliability for particular workloads/functions, which can lead to higher performance by bypassing the error correction/detection algorithm, which could be slow for high reliability systems as in multi-level phase-change memory (PCM) where strong ECC algorithms are typically used. We will augment existing DDR IPs with flexible and transparent memory reliability support.

Note that building new DDR IPs that implement the low-level timing details is very time consuming, and will be technology specific, thus a wrapper that implements the reliability support below the DDR IP will be pursued in this research.

References

[1] Robert S Chappell, Jared Stark, Sangwook P Kim, Steven K Reinhardt, and Yale N Patt. Simultaneous subordinate microthreading (ssmt). In *Proceedings of the 26th International Symposium on Computer Architecture (Cat. No. 99CB36367)*, pages 186–195. IEEE, 1999.

[2] Niket K Choudhary, Salil V Wadhavkar, Tanmay A Shah, Hiran Mayukh, Jayneel Gandhi, Brandon H Dwiel, Sandeep Navada, Hashem H Najaf-abadi, and Eric Rotenberg. Fabscalar: Composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template. *ACM SIGARCH Computer Architecture News*, 39(3):11–22, 2011.

[3] Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N Patt. Runahead execution: An alternative to very large instruction windows for out-of-order processors. In *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.*, pages 129–140. IEEE, 2003.

[4] Arun F Rodrigues, K Scott Hemmert, Brian W Barrett, Chad Kersey, Ron Oldfield, Marlo Weston, Rolf Risen, Jeanine Cook, Paul Rosenfeld, Elliot Cooper-Balis, et al. The structural simulation toolkit. *ACM SIGMETRICS Performance Evaluation Review*, 38(4):37–42, 2011.

[5] Rami Sheikh, James Tuck, and Eric Rotenberg. Control-flow decoupling. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 329–340. IEEE, 2012.

[6] Vinesh Srinivasan, Rangeen Basu Roy Chowdhury, and Eric Rotenberg. Slipstream processors revisited: exploiting branch sets. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 105–117. IEEE, 2020.

[7] Karthik Sundaramoorthy, Zach Purser, and Eric Rotenberg. Slipstream processors: Improving both performance and fault tolerance. *ACM SIGPLAN Notices*, 35(11):257–268, 2000.

[8] Doe Hyun Yoon and Mattan Erez. Virtualized and flexible ecc for main memory. In *Proceedings of the fifteenth International Conference on Architectural support for programming languages and operating systems*, pages 397–408, 2010.

[9] Huiyang Zhou. Dual-core execution: Building a highly scalable single-thread instruction window. In *14th International Conference on Parallel Architectures and Compilation Techniques (PACT'05)*, pages 231–242. IEEE, 2005.

Title(s):

Utilizing Recent Advances in NLP-ML for HW/SW Codesign
or
Automatic Acceleration of Algorithms using ML Techniques

Abdel-Hameed A. Badawy, New Mexico State University, badawy@nmsu.edu
Pat McCormick, Los Alamos National Laboratory, pat@lanl.gov

Topic: architectures, programming systems, emerging technologies, codesign methodologies

The growing prevalence of open-source hardware efforts and the ability to combine them with open-source software toolchains (e.g., programming languages, compilers, runtime systems, tools, etc.) provides a unique opportunity for many different approaches to multi-disciplinary codesign activities. For example, there are a growing number of proposals for RISC-V-based CPUs, GPUs, FPGAs, and other various forms of accelerators. Building on this broader trend, we can envision a more cost-effective approach that leverages a chiplet-based design methodology to enable customization, heterogeneity, and use in full-scale systems previously unachievable [1, 6].

Such designs could consider traditional processor cores, vector units, scratchpad-like sector caches, accelerators, processors-in-memory, etc. A fundamental challenge is selecting components for a given set of applications and a set of constraints (e.g., cost, power, programmability, available chip area, etc.). This is similar to how embedded systems are designed. One significant difference is that the applications running on an embedded system are finite and not general-purpose. Furthermore, embedded systems do not have proxy applications. Proxy apps hide the real applications that will run on the hardware. Working with the application(s) that will run on the system themselves is a significant advantage for embedded system codesign. In addition, the breadth of this exploration space is costly in terms of design choices, development and deployment timelines, and in general, the expertise and size of a team required to consider the cross product of all combinations for an application space as broad and complex as DOE's. Even though the chiplet approach can simplify aspects of the design, these costs are likely well beyond DOE budgets in terms of both funding and the required "*time to solution*". However, if these costs can be reduced, there is significant value in this approach to improving and optimizing architecture design choices across many mission-critical areas.

Genetic programming (GP) was once used to write programs or search for efficient programs to perform specific functions. With Machine Learning models that can learn through billions of parameters, researchers ask whether humans will need to write code again [2, 3]. GP, by definition, searches for the software. In the context of codesign, we can further push the envelope and explore whether we can co-search for (*i.e.,* codesign) the hardware and the software together. In other words, we can use tools like GPT-3 [4] or Microsoft Turing NLG [5] for the codesign of SW and HW together. The idea is that we train a machine learning model with as much code and parameters as needed so that we would ask it to produce code with the associated hardware that best fits that code. Since the system would be composed of various accelerators and components, the software would be written for specific components. We could use an intermediate language representation so that it is architecture-independent. Machine learning has come a long way, and probably it is time to use the full potential of machine learning for the benefit of codesign.

This idea is probably a long shot and maybe too far for the current technology. One way to think about it is to scale it down. Let us reason about it in the context of taking a particular algorithm that we want to accelerate using some reconfigurable architecture such as an FPGA, a CGRA, or even a GPU. The problem can be formulated as follows: Given the original algorithm and the particular configurable hardware or set of accelerators possible, is there a software solution or design of the algorithm one of the accelerators that can beat all the other choices. Alternatively, which accelerator could maximize the performance gain, and what should the algorithm look like to run on the selected accelerator. This is being solved with the following constraints: choose from the set of accelerators available and optimize the accelerator's algorithm to maximize the performance gain.

References

[1] Leiserson, Charles E., Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, and Tao B. Schardl. "There's plenty of room at the Top: What will drive computer performance after Moore's law?." *Science* 368, no. 6495 (2020).
[2] Billings, Jay Jay, Alexander J. McCaskey, Geoffroy Vallee, and Greg Watson. "Will humans even write code in 2040 and what would that mean for extreme heterogeneity in computing?." *arXiv preprint arXiv:1712.00676* (2017).
[3] Lee, Rubao, Hao Wang, and Xiaodong Zhang. "Software-defined software: a perspective of machine learning-based software production." In *2018 IEEE 38th international conference on distributed computing systems (ICDCS)*, pp. 1270-1275. IEEE, 2018.
[4] Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020).
[5] Rajbhandari, Samyam, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. "Zero: Memory optimization towards training a trillion parameter models." *arXiv preprint arXiv:1910.02054* (2019).
[6] Extreme Heterogeneity 2018 - Productive Computational Science in the Era of Extreme Heterogeneity: Report for DOE ASCR Workshop on Extreme Heterogeneity.

https://doi.org/10.2172/1473756

An AI System for AI Codesign

Prasanna Balaprakash*, Murali Emani, Venkat Vishwanath, Rob Ross, Stefan Wild
Argonne National Laboratory
pbalapra@anl.gov

Topic: Codesign methodologies
Challenges: Heterogeneous nodes, many-core processors, deep memory hierarchies, power and energy demands, and wafer-scale integration make application and system management on existing and emerging AI computing platforms an increasingly daunting task. Current codesign strategies for software stack and application development on AI hardware are mostly static and are often optimized based on a narrow set of commercial AI workloads. Consequently, such a one-size-fits-all approach will render several existing, emerging, and future DOE AI applications less performant—a significant hurdle for the data-driven scientific discovery. To that end, we focus on solving two critical AI codesign research challenges faced by DOE AI community:

1. How to map the diverse set of DOE AI applications on existing and emerging AI hardware, i.e. how to offload computations to appropriate hardware with respective tuned configurations?
2. How to design appropriate AI hardware to effectively run these DOE AI applications?

Opportunity: We envision an AI system for AI codesign that leverages reinforcement and supervised learning, architectural simulator, analytical performance models, and DOE leadership-class systems. An automated AI approach for AI codesign will be an ambitious effort that takes a radically new route compared to the traditional codesign approaches attempted so far within DOE complex. The proposed technology has the potential to revolutionize ways in which DOE AI applications will be run on different hardware to maximize scientific impact, hardware will be procured for leadership-class computing facilities and edge computing needs, and design choices of the hardware vendors will be influenced by the DOE AI application needs.

We have to develop codesign-centric and vendor-agnostic DOE AI application benchmarks for evaluating the performance of different hardware platforms and software stacks. While there are recent community developments such as MLCommons [1], they lack the capabilities and flexibility required for DOE AI application codesign methods. We can use MLPerf benchmarks from MLCommons as a starting point and expand them to include representative benchmarks from DOE AI applications. For example, the extended benchmarks will range from storage-intensive, simulation-enabled training to compute-intensive inference with uncertainty quantification, and involve varying degrees of retraining, power/energy demands, mixed precision, error tolerance, dense and sparse datasets with diverse data types such as images, text, time series, graphs, and point clouds.

Existing hardware accelerators such as GPUs, TPUs, FPGAs and emerging AI systems such as Cerebras, SambaNova and Graphcore provide unique capabilities and exhibit different characteristics with respect to training time, inference time, latency, power, and energy demands. To map the diverse set of DOE AI application benchmarks on existing and emerging hardware, we envision a codesign-aware reinforcement learning (RL) approach, where an agent evaluates different mapping strategies and learns to maximize the user-defined, application-specific reward. However, given the complexity and the degrees of freedom of the decision space of mapping strategies, a naive RL method is unlikely to be successful. Therefore, we need to develop a hierarchical RL method [2] that leverages node-level and system-level models, to obtain the optimal mapping strategy. At the node level, we need multimetric performance models (e.g., for runtime, memory footprint, data movement, power, energy) based on the hardware/software/application configuration options. Traditional analytical modeling approaches should be adopted wherever possible because of their data efficiency and extrapolating power. However, sufficiently rich analytical models are difficult due to interactions between different components within a node (e.g., pipelines, special instructions, order of executions, memory hierarchy, frequency scaling). At the system level, models for communication, data movement, load balancing, and I/O are vital, but to-date, such models have proven too complex to model analytically. Therefore, when analytical performance models become too restrictive for both node and system level models, we have to develop supervised-learning-based performance modeling to develop surrogate models. In hierarchical RL, at each level of the hierarchy, agents try to optimize their level-specific reward functions using the models. Once trained on the benchmarks with different accelerators, the RL agent can be leveraged to map the real AI applications on a given hardware.

To design appropriate hardware for the diverse set of DOE AI applications, we envision architectural-simulation-based RL approach in which agents try different hardware designs and learn to maximize user-defined application-specific reward function. Scalable architectural simulators that emulate novel system designs and that support modification of both programming models and hardware organization are critical. These architectural simulators need to provide extensible and parameterized models for the node, interconnect, and storage. In the RL-based design approach, the RL agent will orchestrate the parameters of the node, interconnect, and storage to maximize the reward function. We envision a codesign-specific constrained hierarchical RL in which an agent's actions are restricted using domain and user constraints. Consequently, agents will be forced to find design parameters that are not only maximally impactful on diverse set of AI application benchmarks but also close to vendor baseline and feasible to design/modify. We need to develop multiobjective RL methods [4] that can generate diverse hardware designs with different performance, accuracy, energy tradeoffs for the same application benchmark. Finally, we have to scale the RL methods on leadership-class systems to explore the combinatorial hardware design space using the hardware simulator as the environment. A promising research avenue is explainable and interpretable RL methods for codesign that can provide insights on the high-performing RL policies.
Timeliness: The diversity in AI hardware accelerators, DOE AI applications, and programming models make the AI codesign an expert-driven, iterative, and time-consuming process. This issue has the potential to hinder the development of AI hardware, software and applications, which directly affects the promise of AI-enabled scientific discovery within DOE and elsewhere. Recent advancements in AI methods, in particular, data-efficient domain-aware RL for design problems and demonstrations of AI methods for hardware design [3] make the envisioned AI system a novel and realistic approach to tame the complexities of AI codesign. The proposed approach has the potential to revolutionize codesign for DOE AI applications by automating the process of mapping diverse set of scientific applications to emerging AI hardware and test beds and designing hardware for diverse set of scientific applications.

References

[1] ML Commons. `https://mlcommons.org/en/`.

[2] Frans et al. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

[3] Mirhoseini et al. Device placement optimization with reinforcement learning. In *International Conference on Machine Learning*, pages 2430–2439, 2017.

[4] Nguyen et al. A multi-objective deep reinforcement learning framework. *Engineering Applications of Artificial Intelligence*, 96:103915, 2020.

Advanced Architecture Assessment Within a Codesign Methodology

Kevin J. Barker (kevin.barker@pnnl.gov), Pacific Northwest National Laboratory
David Manz, Pacific Northwest National Laboratory

February 16, 2021

Topics: Architectures, Emerging Technologies, Codesign Methodologies

Challenge: The slowdown in technology node scaling has pushed computer architects to explore new opportunities and methods in order to provide continually increasing levels of computational performance. Architectural specialization enables emerging systems to provide high performance within reasonable power and area budgets. Specialization has made inroads in the marketplace, first with dedicated Graphical Processing Units (GPUs) and more recently with the emergence of dedicated Machine Learning accelerators. While these architectures, particularly GPUs, have penetrated the high-performance computing landscape to great effect, it is apparent that they are not silver bullets providing a single solution amenable to all application types. As we move into an era in which scientific discovery will be driven by workloads containing computational components that span traditional HPC, data analytics, and AI/ML, increasingly heterogeneous architectures will be required to support such diversity.

As purpose-designed architectures are conceived and fabricated, potentially using reconfigurable programmable logic, assessment will become a driving concern. Such assessment will have axes of performance, power efficiency, and security. A number of challenges will need to be addressed, including developing methods to evaluate composable IP blocks developed by third parties and assessment of pre-implementation designs.

Vision: Performance evaluation of high-performance computing systems has long been used in various stages of the design life cycle, from pre-production prototyping to large-scale deployment and has been effectively used in large-scale system design and the evaluation of proposed systems. Various methods have been used to great effect, including low-level architectural simulation and coarse-grained analytical modeling. Both approaches have benefit and are appropriate to address different concerns.

However, reasoning about performance in the context of emerging heterogeneous architectures raises new challenges. The proliferation of device architectures and capabilities makes insight into workload behavior difficult. To address these concerns, the Department of Energy Office of Science established the Center for Advanced Technology Evaluation (CENATE) at Pacific Northwest National Laboratory (PNNL) in 2016. CENATE encompasses an advanced architecture testbed capability that enables researchers to quantitatively explore performance, power efficiency, and security of emerging processing and system architectures.

Position Statement: A lack of concerted and focused effort dedicated to the integrated evaluation of emerging technologies and their impact on DOE mission areas is a notable gap that CENATE is designed to address. We envision CENATE taking a leading role in an evaluation and assessment capability that spans the DOE national laboratory complex, as well as partners in academia and industry. With the rapid emergence of new computing technologies, building an understanding of their capabilities and potential impacts requires coordination among all players. Thrust areas in which CENATE has built capabilities that can be applied to this challenge include:

1. Advanced technology evaluations that examine state-of-the-art advances in hardware and associated software on a range of testbeds, comprising both component technologies and scalability platforms.
2. Measurement instrumentation for power and performance encompassing state-of-the-art measurement, tools, and methods.
3. Testbed infrastructure that affords rapid evaluation of technologies.
4. Predictive exploration combining empirical evaluation with modeling and simulation to quantify the potential impact of technologies on future systems and workloads.

Opportunities: A unified assessment and evaluation capability, even one with multiple components led by different organizations, can be a mechanism to unify a myriad of point-to-point interactions between the DOE, industry, and academia. A common framework for the evaluation of advanced architecture designs, including those for extremely heterogeneous systems, will enable the rapid evaluation of architectural concepts developed within the DOE as well as by industrial and academic technology developers. Computing testbeds and tools, including reconfigurable architectures, can serve as simulation and emulation platforms for advanced architecture concepts and enable direct evaluation of critical DOE workloads. The definition of key metrics of interest will enable an exchange of assessment information and allow organizations across the DOE to benefit from the experiments and assessments conducted at a single site. Finally, the development of methodologies to assess architectural concepts from the design stage through implementation will foster collaborations between national laboratory and university computer scientists and hardware architects.

Timeliness: For nearly two decades, the DOE and larger computing industry have grappled with the implications of a slowdown in technology node scaling, leading first to multi-core processing architectures and later into the widespread adoption of GPU architectures as primary compute engines of large-scale HPC systems. More recently, the DOE Exascale Computing Project (ECP) demonstrated the benefit of leveraging vendor technology roadmaps toward the integration of CPU and GPU processors into heterogeneous system architectures. However, with the emergence of new application workloads combining aspects of HPC, data analytics, and AI/ML, architectures are being driven toward a finer granularity of heterogeneity leveraging emerging and open System-on-Chip ecosystems that are driving much of the innovation seen in other computing spaces. This raises an urgent need to deploy an assessment capability that can provide a quantitative understanding of how these emerging architectures can best be leveraged by current and future DOE workloads.

[1] Tallent N.R., K.J. Barker, R. Gioiosa, A. Marquez, G. Kestor, S. Song, and A. Tumeo, et. al., "Assessing Advanced Technology in CENATE", in Proc. of the IEEE International Conference on Networking, Architecture, and Storage (NAS), 2016.
[2] Li A., S. Song, J. Chen, X. Liu, N.R. Tallent, and K.J. Barker, "Tartan: Evaluating Modern GPU Interconnect via a Multi-GPU Benchmark Suite", in IEEE International Symposium on Workload Characterization (IISWC), 2018.
[3] Castellana V.G., M. Minutoli, A. Tumeo, M. Lattuada, P. Fezzardi, and F. Ferrandi, "Software Defined Architectures for Data Analytics", in Proc. of the 24th Asia and South Pacific Design Automation Conference (ASPDAC), 2019.
[4] Li A., T. Geng, T. Want, M. Herbordt, S. Song, and K.J. Barker, "BTSC: A Novel Binarized-Soft-Tensor-Core Design for Accelerating Bit-Based Approximated Neural Nets", in International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), 2019.
[5] Firoz J.S., A. Li, J. Li, and K.J. Barker, "On the Feasibility of Using Reduced-Precision Tensor Core Operations for Graph Analytics", in IEEE High Performance Extreme Computing Conference (HPEC), 2020.

Design Space Exploration For Machine Learning Architectures

Michael Barrow, Oliva Weng, Ryan Kastner
barrow9@llnl.gov, oweng@ucsd.edu, kastner@ucsd.edu

TOPIC: HARDWARE SOFTWARE CO-DESIGN TOOLS FOR MACHINE LEARNING

Hardware designers and software developers operate in fundamentally different ways. Hardware designers deal with highly parallel programming constructs, optimize their designs heavily, and spend a majority of their efforts on verification. Software developers focus on usability, fast incremental development, and integration into complex systems. By and large, hardware and software communities have operated in isolation by using hardware/software interfaces and abstractions (e.g., ISAs).

In recent years, the line between hardware and software has blurred. “Software” companies are seeing the advantages of developing their own custom hardware, e.g., Microsoft’s SQ2 chip. Since the hardware is designed for the application at hand, it provides advantages in performance, power consumption, security, and cost. Furthermore, the emergence of open-source hardware designs, languages, and abstractions have made hardware design easier. Despite all these in-roads, hardware design still remains challenging.

Machine Learning (ML) applications exemplify these challenges. ML is a must-have for software companies. For example, 70% of Google revenue is from ML targeted advertisements. ML algorithms are highly parallel and good targets for specialized parallel hardware. But, creating optimal ML hardware designs is difficult because the hardware design space is not considered during ML algorithm development. The ML algorithm design space is high in dimensions, with many viable network architectures, hyperparameter settings, and optimization approaches (e.g., pruning and quantization). Traditionally, software companies meticulously search this space using high level ML frameworks like PyTorch and Tensorflow to meet their accuracy requirements. These frameworks however largely abstract away and ignore the multi-dimensional space of hardware architectures (e.g., memory bandwidth and hardware precision). ML developers leave it to the hardware specialists to take their finished ML applications as they are and optimize their performance for hardware. This approach is problematic. Because ML frameworks do not take the hardware design space into account, many of the ML models are developed at the expense of hardware. It is only possible to efficiently design hardware for a ML application if the ML algorithm design space is co-explored with possible hardware mappings.

The traditional approach of isolated software and hardware development suffers from this major challenge: *Since software developers have already fixed the dimensions of the ML algorithm design space by the time hardware developers receive them, the hardware design space cannot be effectively explored to achieve hardware-efficient designs.*

CHALLENGE: EFFECTIVE DESIGN SPACE EXPLORATION

State-of-the-art ML applications are complicated, with thousands of dimensions and millions of weight parameters. It is difficult for application developers to understand the impact of their design decisions on hardware performance. Similarly, it is difficult for hardware developers to understand the impact of their decisions on algorithm accuracy. Without feedback between each other, it is not possible to explore the application design space with respect to both the software and hardware intelligently.

Recent developments have sought to effectively and efficiently map neural networks to hardware. Proposals have been made by both hardware and software oriented groups, with mixed results. FINN is a open source framework initially developed by the hardware company Xilinx [5]. FINN is an end-to-end tool that generates hardware accelerated inference networks for Xilinx FPGA platforms, and is focused on quantized neural networks. The tool is implemented using Python and Vivado HLS and has interfaces to high level ML frameworks such as pytorch. The benefit of this is that hardware performance is exposed to the algorithm development environment. Although FINN is capable of generating highly optimal designs, it is not well suited to design space exploration. Modifying a network requires domain specific hardware knowledge. In our experience, any modifications took several months for PhD and Postdoctoral students to make.

hls4ml is another end-to-end Python tool for generating accelerated inference networks, but was developed by a consortia of software-centric academics [1]. hls4ml has a broader support for ML network designs compared to FINN. It is also better documented, leading to shorter development times. However, hls4ml relies on Vitis®to map its generated quantized network IP cores to a Xilinx FPGA and is less optimal. Conversely, FINN expects the user to manually configure a network mapping on the FPGA.

Although these state-of-the-art ML frameworks bridge the gap between application and hardware design, the status quo end-to-end paradigm is limited. Existing tools lack feedback cues between the ML and hardware mapping steps, or lack knobs to tune hardware mappings. Consequently, the end-to-end flow cannot inform iterative exploration of the complete design space, costing many development hours on both the software and hardware side. What is needed is a framework with a more informative feedback loop from the hardware tools to the algorithm developer, with more intuitive (ML-specific) hardware map tuning knobs.

OPPORTUNITY: APPLICATION-SPECIFIC FEEDBACK TOOL FOR DESIGN SPACE EXPLORATION

State-of-the-art ML application hardware mapping is done using High Level Synthesis (HLS) compilers. Although HLS has long sought to bridge high level application specifications to hardware mapping, HLS compilers are unwieldy tools. HLS primitives are highly granular so that HLS can be applied to a broad spectrum of application classes. Unfortunately, the consequence is that domain-specific hardware design knowledge is required to describe and schedule ML primitives using HLS tools. *This presents an opportunity to develop ML-focused tools that allow co-exploration of the algorithm and hardware design spaces.*

We want to “lift” the hardware design-space exploration to the software or even the algorithmic level to make hardware software co-design more commonplace. Without exploring the hardware architecture implementation specifics, design choices made using high level ML frameworks to improve model performance can turn out to be unexpectedly costly during hardware deployment. For example, although Resnet-a-like networks have good training speed thanks to gradient highways, they can cause major challenges when trying to realize efficient hardware implementations. Naive skip connections waste resources due to large buffers required to accurately represent the network, but the connections may not even be necessary to achieve acceptable results.

Such pitfalls could be avoided with a tool that gives machine learning developers access to various algorithmic, software, and hardware knobs that they can tune whilst they develop their ML algorithms using an enhanced end-to-end flow. Some example knobs follow in Table 1:

ML Knobs	hyper-parameters	model parameters / topology	time to convergence	model sparsity	quantization
HW Knobs	memory bandwidth	power requirements	resource utilization	throughput	

Table 1: Hardware Software Co-design knobs that could be exposed to ML algorithm developers

MATURITY: REMAINING GAPS IN ML SPECIFIC DESIGN SPACE EXPLORATION APPROACHES

Recent advances in Neural Architecture Search and adjacent research areas offer steps towards such an all-encompassing tuning tool [3, 4, 6]. For example, [4] introduced Adaptive Threshold Non-Pareto Elimination (ATNE), which is a design space exploration framework that uses machine learning to tune various OpenCL-to-FPGA knobs (such as unroll factor and number of SIMD lanes) to automate finding Pareto-optimal designs for a given high level application. This demonstrates how a smart Hardware Abstraction Layer (HAL) can optimize hardware using a set of abstract design knobs and insulate the application developer from needing to understand how to make progress towards an optimal hardware mapping.

Although promising, these works are hampered in ML applications because the HAL is not capable of optimizing the ML network topology. In other words, they do not provide ML network optimization hints to help the application developer explore the hardware design space as well.

Neural Architecture Search has recently been extended to find hardware and resource friendly ML network designs, as seen in [2]’s Lamarckian evolutionary algorithm for multi-objective neural architecture design (LEMONADE) algorithm. In Elsken’s work, hardware architecture is assumed fixed, and a subset of ML design knobs from Table 1 are explored by the LEMONADE algorithm. This work represents the “top down” counterpart to “bottom up” approaches such as ATNE that conversely assume the ML design is fixed and explore the HW knobs in Table 1.

Bridging the gap between works such as LEMONADE and ATNE is an exciting prospect that would enable co-exploration of the ML algorithm and hardware design space. Success in this area would streamline the development of some of the most widely used applications today. ML application developers would be provided with the tools they need to develop hardware-friendly networks, and hardware specialists would be able to make informed optimizations to these networks.

References

[1] Duarte, J., Han, S., Harris, P., Jindariani, S., Kreinar, E., Kreis, B., Ngadiuba, J., Pierini, M., Rivera, R., Tran, N., et al. Fast inference of deep neural networks in fpgas for particle physics. *Journal of Instrumentation 13*, 07 (2018), P07027.

[2] Elsken, T., Metzen, J. H., and Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081* (2018).

[3] Liu, H.-Y., and Carloni, L. P. On learning-based methods for design-space exploration with high-level synthesis. In *Proceedings of the 50th annual design automation conference* (2013), pp. 1–7.

[4] Meng, P., Althoff, A., Gautier, Q., and Kastner, R. Adaptive threshold non-pareto elimination: Re-thinking machine learning for system level design space exploration on fpgas. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2016), IEEE, pp. 918–923.

[5] Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., and Vissers, K. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2017), pp. 65–74.

[6] Zuluaga, M., Sergent, G., Krause, A., and Püschel, M. Active learning for multi-objective optimization. In *International Conference on Machine Learning* (2013), PMLR, pp. 462–470.

Codesign Beyond Exascale

Bo Begole, Gabe Loh, Mike Schulte, Mike Ignatowski, Valentina Salapura, Keith Lowery, Andrew Kegel, John Kalamatianos, Nuwan Jayasena, Alan Lee

Advance Micro Devices, Inc.

***Abstract*—This position paper proposes enhancements to the successful codesign methods employed during the last decade of DOE programs including the Exascale Compute Project (ECP) and related research programs (DesignForward, FastForward, PathForward).**

I. Introduction

Computing systems ranging from low-power sensors to supercomputers require significant advancements in performance, power efficiency, and programmability to extend innovations in science, manufacturing, medicine, and national security into the post-exascale era. The country's future scientific leadership, security, and economic growth depend directly on continuing the advancement of our computational capabilities and maintaining high-performance computing (HPC) leadership. Although computing systems have seen significant advancements over the decades, the future of American computing leadership has been called into question due a combination of the growth of foreign computing capabilities, scarcity in the necessary technical human skills, exploding software complexity, rapidly increasing power consumption, and fundamental physical limits with transistors reaching atomic scale [1].

The U.S. Department of Energy (DOE) exascale program investments were critical in fostering codesign methods that address these challenges and maintain rapid innovation. Scientists and system architects in DOE and industry have shared knowledge, allowing industry specialists to gain deep insights into DOE problems at the national labs and to design systems that accelerate the execution of unique scientific applications. Specialization will continue to be one of the methods used for advancing the performance of scientific applications in the face of declining performance gains due to slowing of Moore's Law and increasing system complexity. Increased investment in codesign can maintain performance gains of prior decades by addressing these challenges.

II. Challenges to Codesign

After nearly a decade of successfully employing codesign with members of DOE, we have identified several challenges that need to be addressed to handle the increasing complexity of DOE missions and to maintain US HPC leadership into the coming decades.

A. *Proxy application overhead and inaccuracy*

Proxy applications have been used to characterize the behaviors of scientific applications and datasets without exposing potentially sensitive details about individual actual applications or their data. Unfortunately, developing proxy applications is time consuming and the result has inherent inaccuracies as they alter certain characteristics of the actual applications. In some cases, a proxy application needs to be ported to new accelerators, which comes on top of the cost of porting the actual application itself to such accelerators.

Because the proxy is not the actual application, it will not fully characterize the behaviors of the original application. Proxy applications focus on the high-cost characteristics of an application, and are often biased toward the computationally heavy aspects, potentially under-representing the data movement, code footprint, and communication overheads seen in actual applications. A proxy is intended to quantify performance enhancements that would potentially benefit the actual application. However, a proxy is only designed to measure improvement in the bottlenecks it embodies and will fail to show enhancements outside of its implementation.

B. *Pairing expertise*

Furthermore, the needs of application developers are increasingly varied, such that no single computing architecture can optimally address the diverse set of application requirements. Architects need to work with application developers tightly and iteratively to design semi-custom systems or influence general purpose systems to meet the developers' needs. Pairing the specific application developers with engineers who can design solutions becomes challenging due to the size and complexity of government and industry organizations.

III. Future of Codesign

Addressing these challenges requires increasing the investment in codesign to deepen the understanding of application behaviors on current and future compute architectures. New analytic tools and artificial intelligence (AI) can be employed to increase the efficiency of scientists and engineers in the codesign of high performing solutions.

A. *Instrumentation of runtime systems*

Rather than investing time and resources into developing inherently inaccurate proxy applications, future HPC systems can include additional hardware instrumentation and analytics to increase insights without diminishing the performance in deployed systems. Through instrumentation within systems that are currently deployed, the DOE can capture the actual behaviors of existing applications including their computation and memory movement patterns and actual resource contentions.

In the past, hardware monitoring via performance counters has provided limited insights into "hot spots" where performance is noticeably high (or low), but it has been left to human architects and developers to hypothesize the cause. Increasing the information collected and performing statistical and machine intelligence-enabled analysis within the hardware itself can increase the fidelity of observed performance and identify likely causes and relationships of the observed performance phenomena.

Codesign is needed so that scientists can validate that certain behaviors are captured by increased embedded instrumentation. In addition, alleviating identified bottlenecks will continue to require codesign to weigh the tradeoffs of prospective solutions.

B. *Intelligent application behavior analysis*

Codesign should be facilitated by intelligent software that can identify performance problems and likely causes. Tools that can predictively characterize an application across a variety of heterogeneous architectures would allow codesigners to rapidly narrow in on the most promising approaches, and perhaps even consider design directions that would not have been initially imagined.

Ultimately, the optimizations that codesigners identified may become automated into the runtime of systems. Developing intelligent system management will require collaboration between component suppliers (processors, memory, networks), system integrators, datacenter operators, and possibly 3rd-party software operating system and system management software suppliers. No single entity can do this alone.

To keep track of the current state of the hardware and software operations, self-managing systems will likely require feedback from various levels of the system. We envision a hierarchical flow comprising various sensors for fast-changing attributes like temperature, power, performance counters, compute and data movement intensity, network performance, as well as slower loops for reliability, aging, etc.

C. *Simplifying programming complexity*

As systems increasingly combine heterogeneous processers and accelerators, the task of programming becomes increasingly complex. Developers need to map their applications to data models, instruction sets, and compute paradigms for a multitude of processors. Codesign teams can alleviate some of the complexity by designing micro-architectures, software abstractions, and intermediate representations that parallel the problem domain optimally. In some cases, instruction set architectures (ISA) can be extended to map more closely to application paradigms and provide optimal performance or facilitate the development of reusable libraries that can benefit multiple application domains. For other cases, the introduced abstraction layer could map well to the problem domain. For certain application domains, the data structures and operations of application developers can be facilitated in software as well as mapped to hardware via intelligent compilers and runtime layers.

AI tools can also provide a boost in automating or semi-automating the transformations between a domain representation of an application and the machine execution of the solution across a variety of different architectures. Such AI support systems need to be codesigned to combine the expertise of application developers, runtime software and system architects.

D. *Open Innovation*

Another element of successful innovation is to embrace open innovation across public, commercial, academic, and non-profit institutions to expand the pool of ideas that contribute to innovation. Open innovation involves the fostering of open and non-proprietary standards and interfaces by which components can interoperate, allowing suppliers to focus on innovations in their specific value-added specializations. Open standards help amortize public investments because compatibility with open standards enables solution providers to sell into a larger ecosystem than a government-specific or otherwise propriety standard would provide. Related to open standards, open-source software provides an ad hoc form of standardization that can be steered by a community of contributors including public entities.

IV. Conclusion

Codesign has been highly effective in the recent design of HPC systems by providing a user-centered approach to identify requirements and to prioritize tradeoffs that drive innovation.

To build on the successful foundation of codesign over the last decade, it is vital that funding programs focus on ongoing, stable, and long-term efforts. Government funding and planning should intercept the planning cycles of both government and industry entities to maintain institutional knowledge and continuous innovation. The increasing difficulty of achieving performance gains in the post-Moore's Law era will require ever more complex solutions and necessitate increased investment to produce a codesigned series of increasingly advanced systems on a competitive timeline.

References

[1] A. Lee, B. Begole, M. Ignatowski, K. Lowery, G. Loh, "AMD Advanced Research's Response to DOE Request for Information: Basic Research Initiative for Microelectronics," https://www.regulations.gov/document?D=DOE-HQ-2019-0031-0023

Improving Energy Efficiency of Edge Devices Using Asynchronous Design Paradigm

Kshitij Bhardwaj
Lawrence Livermore National Laboratory, Livermore, CA
bhardwaj2@llnl.gov

Topic areas: heterogeneous architectures, emerging technologies

From mobile phones to supercomputers, we are seeing a rise in the use of heterogeneous architectures. The modern systems combine general-purpose processors with GPUs, FPGAs, application-specific chiplets, and specialized hardware accelerators [1, 2]. Application-specific acceleration in these systems has led to improved performance. However, power consumption continues to be a major challenge for chip designers [3], and there is a need for non-traditional innovative design technologies which can reduce power consumption while maintaining or improving performance. These approaches will help design not just energy-efficient application-specific chiplets for HPC but also resource-constrained edge devices such as near-sensor processors, both of which align well with the DOE's vision.

Majority of the computing systems that we see around us use a *conventional synchronous design approach,* as shown in Figure 1. In these systems, a single clock drives all the processing elements and the memories. For years, the designers have relied on increasing the clock rate to deliver high performance but now we have hit a *power wall,* where the clock rates have saturated [3]. The reason the clock speeds cannot be increased any further is that it leads to significant increase in chip power ($power \propto clock\_frequency$) which in turn can cause thermal and reliability issues. Another limitation of conventional synchronous approach is that the clock rate is determined by the slowest component of the system, which may not be optimal, especially for heterogeneous systems where the processing speeds of different units vary significantly. Finally, as the system complexity increases, connecting a single clock to all the cores of a system complicates physical design process and drives up the cost.

An alternative to synchronous technology is *asynchronous design.* Systems using this technology eliminate clock and instead use handshaking between the various processing elements, as shown in Figure 2. Such handshaking signals use request and acknowledgement signals for data transfer. These designs have several potential advantages [4]: (i) low power as there is no clock switching power as well as due to their data-driven operation that leads to dissipating power only when the design is active and not when it is idle. In contrast, synchronous designs can burn power even when idle; (ii) performance is average-case as it is not limited by the slowest component; and (iii) allows object-oriented style of integration of a variety of synchronous processing units, each operating at different clocks, using an asynchronous interconnect. This paradigm is commonly called as *globally-asynchronous locally-synchronous (GALS) systems,* as shown in Figure 3.

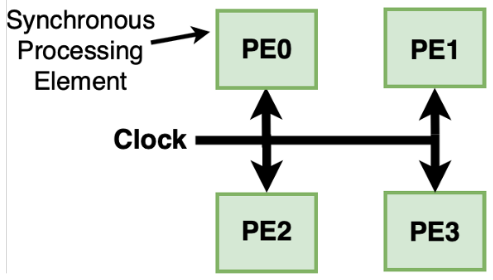


Figure 1: Conventional synchronous design

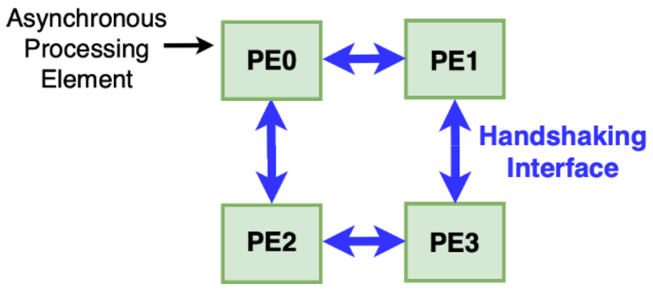


Figure 2: Alternative asynchronous design

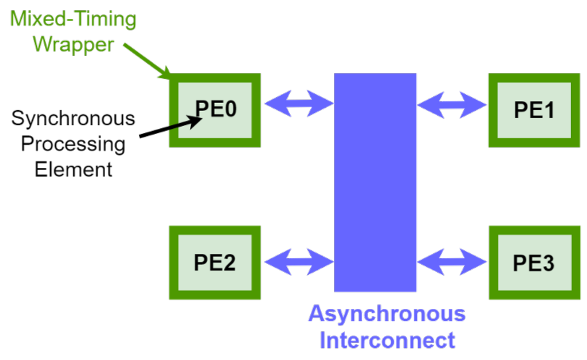


Figure 3: A GALS system

Asynchronous and GALS systems have seen an increased interest recently from both academia and industry. Nvidia's introduced a $4mm^2$ GALS chip for internet-of-things (IoT) applications [5]. Qualcomm showed that an asynchronous on-chip network can lead to 24% reduction in system latency compared to a synchronous network [6]. A collaborative study between AMD, Columbia University, and University of Ferrara demonstrated that an asynchronous on-chip network router outperforms a state-of-the-art AMD synchronous router in 14nm FinFET technology: 55% area reduction, 28% lower latency, and 58% lower power [7]. Fulcrum (acquired by Intel) has produced high-speed asynchronous ethernet switches that support up to 640 Gbps bandwidth [8]. Furthermore, asynchronous has also shown promise for emerging neuromorphic chips for spiking neural networks, such as Intel's Loihi asynchronous chip integrates 128 cores, each modeling 1024 neurons [9]. Loihi supports real-time learning and has been used for a variety of applications such as robotics, and sniffing out hazardous chemicals. Asynchronous and GALS technology has also been used to develop ultra-low-power brain-computer interfaces [10]. Finally, a recent work also showed asynchronous deep neural network accelerators outperform synchronous DNN accelerators by 4.7× in terms of energy efficiency [11].

Given the advantages of asynchronous technology in terms of power and performance, and the recent uptick in its use in industrial chips, we envision that the future edge devices, such as near-sensor processors, as well as computing and networking nodes of the next-generation heterogeneous HPC systems will benefit from adopting this design technique. There is a need for the national labs, industrial vendors, and academia to come together and explore asynchronous design to develop highly energy-efficient systems to accelerate a variety of scientific applications.

REFERENCES

[1] Apple M1 SoC Specifications. https://www.tomshardware.com/news/Apple-M1-Chip-Everything-We-Know.
[2] AI gets a boost via LLNL, SambaNova collaboration. https://www.llnl.gov/news/ai-gets-boost-llnl-sambanova-collaboration.
[3] Oreste Villa, Daniel R. Johnson, Mike O'Connor, Evgeny Bolotin, David W. Nellans, Justin Luitjens, Nikolai Sakharnykh, Peng Wang, Paulius Micikevicius, Anthony Scudiero, Stephen W. Keckler, and William J. Dally. Scaling the Power Wall: A Path to Exascale. In *SC*, pages 830–841, 2014.
[4] Steven M. Nowick and Montek Singh. Asynchronous Design - Part 1: Overview and Recent Advances. *IEEE Design and Test*, 32(3):5–18, 2015.
[5] Matthew Fojtik, Ben Keller, Alicia Klinefelter, Nathaniel Ross Pinckney, Stephen G. Tell, Brian Zimmer, Tezaswi Raja, Kevin Zhou, William J. Dally, and Brucek Khailany. A Fine-Grained GALS SoC with Pausible Adaptive Clocking in 16 nm FinFET. In *ASYNC*, pages 27–35, 2019.
[6] Yvain Thonnart, Pascal Vivet, Shikhanshu Agarwal, and Ramesh Chauhan. Latency Improvement of an Industrial SoC System Interconnect using an Asynchronous NoC Backbone. In *ASYNC*, pages 46–47, 2019.
[7] Davide Bertozzi, Gabriele Miorandi, Alberto Ghiribaldi, Wayne P. Burleson, Greg Sadowski, Kshitij Bhardwaj, Weiwei Jiang, and Steven M. Nowick. Cost-Effective and Flexible Asynchronous Interconnect Technology for GALS Systems. *IEEE Micro*, 41(1):69–81, 2021.
[8] Mike Davies, Andrew Lines, Jon Dama, Alain Gravel, Robert Southworth, Georgios D. Dimou, and Peter A. Beerel. A 72-Port 10G Ethernet Switch/Router Using Quasi-Delay-Insensitive Asynchronous Design. In *ASYNC*, pages 103–104, 2014.
[9] Mike Davies et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 38(1):82–99, 2018.
[10] Ioannis Karageorgos, Karthik Sriram, Ján Veselý, Michael Wu, Marc Powell, David Borton, Rajit Manohar, and Abhishek Bhattacharjee. Hardware-Software Co-Design for Brain-Computer Interfaces. In *ISCA*, pages 391–404, 2020.
[11] S. Xiao et al. A Data-Driven Asynchronous Neural Network Accelerator. In *IEEE TCAD*, 2020.

Auto-DSE4HPC: Towards an Automated Design Space Exploration Tool for Next-Generation HPC Systems

Kshitij Bhardwaj, Maya Gokhale
Lawrence Livermore National Laboratory, Livermore, CA
bhardwaj2@llnl.gov, gokhale2@llnl.gov

Topic areas: Heterogeneous architectures, design space exploration

High performance computing (HPC) systems are the backbone for cutting-edge scientific research. These systems are used for a diverse set of applications, such as first principles physics simulations, quantum algorithms, environment research, and drug discovery, e.g. [1]. HPC applications, already diverse by the nature of the scientific phenomena being modeled, have become even richer in algorithmic diversity by incorporating machine learning (ML) models, for example to to approximate complex physics models [2] or to predict the candidate antiviral agents for vaccines [1].

To enable fast simulation of a variety of scientific applications, including ML-integrated workloads, we envision the next generation of HPC systems to embrace heterogeneity. These systems are expected to encompass not just server-grade processors but also leaner general-purpose cores, GPUs, FPGAs, as well as specialized hardware accelerators for ML [3]. These systems will also benefit from integration of different memory technologies, such as non-volatile memory that can deliver more storage capacity at lower power than DRAMs [4], beneficial in data-rich ML workloads. In a step towards heterogeneity, HPC clusters commonly integrate GPUs [5], and in some cases, FPGAs [6] and ML accelerators [7]. Some of these systems also integrate non-volatile memories and high-bandwidth memories (HBMs) with mainstream DRAMs [5].

Designing the next-generation heterogeneous HPC systems is a challenging task due to their intractably large design space. This space will include many design parameters: number of processing cores per compute node, cache sizes, number of GPUs and number of streaming processors in a GPU, GPU memory size, number of multiply-accumulate units and scratchpad memory size in ML accelerators, amount of non-volatile vs. volatile memory, intra-node bus bandwidth, and many more. These systems also need to be optimized in terms of multiple objectives: performance, power, area, as well as cost. Manually tuning the design parameters to find the optimal HPC configurations is not tractable and would require an intelligent and efficient early architecture design space exploration (DSE) approach.

While there has been prior research on DSE for HPC systems, these works have only targeted a limited design space. The design parameters focused are either for processing architecture such as number of cores and cache sizes [8], or for memory technology that tries to find an optimal combination of emerging memories for HPC [9, 10]. However, to design the next-generation heterogeneous HPC systems, a comprehensive DSE is required that combines design parameters from compute processors, memories, as well as interconnects.

Given this huge design space and multiple target objectives, we envision an ML-based DSE approach, which we call *Auto-DSE4HPC,* as shown in Figure 1. This framework will automatically explore the comprehensive heterogeneous HPC design space and rapidly converge to optimal hardware configurations. We believe that ML algorithms, such as *Bayesian optimization (BO)* and *reinforcement learning (RL)* that have been widely-used for systems-on-chip DSE [11–13], will be good candidates for Auto-DSE4HPC. These algorithms can efficiently tune the various design parameters in order to co-optimize multiple objective functions such as application throughput, system power, and area. These objectives can be evaluated using an HPC system architecture simulation toolkit (such as structural simulation toolkit or SST [14]) that can run the target scientific applications for the different ML-selected hardware configurations. Both BO and RL have been shown to be very

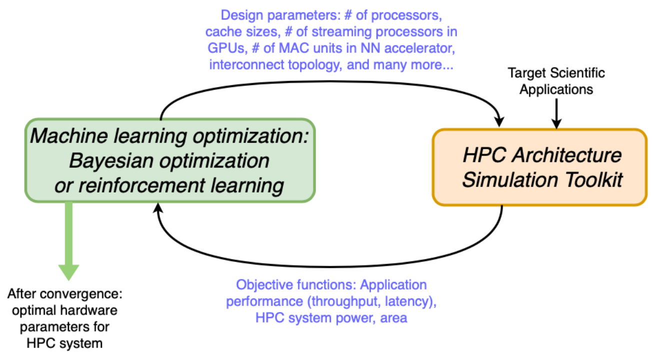

Design parameters: # of processors, cache sizes, # of streaming processors in GPUs, # of MAC units in NN accelerator, interconnect topology, and many more...
Target Scientific Applications
Machine learning optimization: Bayesian optimization or reinforcement learning
HPC Architecture Simulation Toolkit
After convergence: optimal hardware parameters for HPC system
Objective functions: Application performance (throughput, latency), HPC system power, area

Figure 1: Envisioned Auto-DSE4HPC framework: a conjunction of ML-based optimization algorithms with state-of-the-art HPC architecture simulation toolkit. ML algorithms such as Bayesian optimization and reinforcement learning will tune the various design parameters and learn their effect on the target objective functions, eventually converging to a set of optimal HPC hardware configurations.

effective for black-box optimization, and can achieve faster convergence (i.e., requiring less number of expensive HPC architecture simulations in the loop) compared to other traditional approaches such as Genetic algorithms [11, 15].

BO typically uses probabilistic models such as Gaussian processes as the learning model to learn the effects of different design parameters on the HPC power and performance, followed by predicting the optimal design configurations. RL, on the other hand, uses a reward-based system that learns a policy that can predict the optimal configurations. Recently, there has been a rise in the use of deep neural networks as RL policies. It will be interesting to compare BO and RL in Auto-DSE4HPC in terms of amount of training data required, quality of solutions found, and convergence time. To build Auto-DSE4HPC, as a first step, several industrial and academic BO and RL tools [16–18] are publicly available that can be interfaced with the state-of-the-art HPC simulation toolkits.

We expect Auto-DSE4HPC will greatly simplify the DSE cycle, and help answer the following questions about the designs of the next-generation HPC systems for the target scientific applications: (i) what collection of processors, GPUs, specialized hardware accelerators in the compute nodes lead to best performance-per-watt?; (ii) what are the optimal micro-architectures for these processing units?; (iii) what is the best network architecture for intra-node, inter-node, and inter-rack connections; and (iv) does integrating different memory technologies in the memory hierarchy (both at the node-level and rack-level) improve application performance while lowering power and area? To build this comprehensive Auto-DSE4HPC framework and use it to design the future generations of HPC systems, we will need close collaboration between national labs, academia, and industry.

REFERENCES

[1] T. Desautels et al. Rapid in silico design of antibodies targeting SARS-CoV-2 using machine learning and supercomputing. *bioRxiv*, 2020.
[2] A. Sanchez-Gonzalez et al. Accurate prediction of X-ray pulse properties from a free-electron laser using machine learning. *Nature*, 2017.
[3] S. G. Cardwell et al. Truly heterogeneous HPC: co-design to achieve what science needs from HPC. *SMC*, 2020.
[4] Building an ecosystem of heterogeneous memory supercomputing. https://www.nextplatform.com/2020/07/27/building-an-ecosystem-for-heterogeneous-memory-supercomputing/.
[5] Sierra. https://computing.llnl.gov/computers/sierra.
[6] German university will deploy FPGA-powered Cray supercomputer. https://www.top500.org/news/german-university-will-deploy-fpga-powered-cray-supercomputer/.
[7] AI gets a boost via LLNL, SambaNova collaboration. https://www.llnl.gov/news/ai-gets-boost-llnl-sambanova-collaboration.
[8] C. Gomez et al. Design space exploration of next-generation HPC machines . *IPDPS*, 2019.
[9] I. Peng et al. Siena: exploring the design space of heterogeneous memory systems. *SC*, 2018.
[10] S. Sen et al. Machine learning based design space exploration for hybrid main-memory design. *MemSys*, 2019.
[11] B. Reagan et al. A case for efficient accelerator design space exploration via bayesian optimization. *ISLPED*, 2017.
[12] K. Bhardwaj et al. A comprehensive methodology to determine optimal coherence interfaces for many-accelerator SoCs. *ISLPED*, 2020.
[13] W. Jiang et al. Hardware/software co-exploration of neural architecture. *IEEE TCAD*, 2020.
[14] Structural simulation toolkit (sst). http://sst-simulator.org.
[15] K. Settaluri et al. AutoCkt: deep reinforcement learning of analog circuit designs. *DATE*, 2020.
[16] Bayesian optimization in Pytorch. https://github.com/pytorch/botorch.
[17] Boa - a multi-objective bayesian optimization program for the gem5-Aladdin SoC simulator. https://github.com/gncs/boa.
[18] Applied reinforcement learning at Facebook. https://github.com/facebookresearch/ReAgent.

Including Operations, Analytics, & Communication In Next Generation CoDesign: It Just Makes Sense!

Jim Brandt (brandt@sandia.gov), Jeremy Enos (jenos@illinois.edu), Ann Gentile (gentile@sandia.gov), William Kramer (wtkramer@illinois.edu)

Topics: architectures, applications, emerging technologies, codesign methodologies

Challenge:
Over the last decade codesign has largely been an engagement between the software (e.g., applications, compiler, OS) and hardware architecture communities. The middleware that handles system and resource management, operational data analytics, and resiliency processes, though extremely critical to efficient system and application operation, continues to face challenges with respect to scale, performance, and interoperation with the rest of the system. These types of critical middleware need to be included as an integral part of the codesign process to ensure that next generation systems can best exploit the advanced features presented by new hardware technologies and application software.

One such example is resource management whose somewhat static nature coupled with the typically homogeneous nature of HPC computing resources has been largely left out of codesign processes. An increasingly urgent problem, articulated in the 2018 Heterogeneity ASCR report, is that the heterogeneity of current and future architectures does not lend itself to efficient utilization when the characteristics of available resources and the low-level needs of user applications, which may change over execution time, are not known by the scheduling and resource management software. Current codesign approaches do not focus on exposing the dynamic needs of applications or resource conditions. Nor do they provide hooks for feedback that would enable dynamic reconfiguration in the face of faults, degraded performance, or changes in application demands.

Another example is fault handling which could vastly improve overall system performance and efficiency if included in codesign efforts. An example might be thresholding timeouts between middleware components in a sensible way, or allowing them to communicate effectively: If a filesystem high availability component (e.g. Lustre) is in the process of failing over, there is no guarantee that the next layer up (e.g. MPI) will wait appropriately before deciding that it will never return. Thus, a mechanism for the filesystem to indicate "hey - I'm failing over, and will be right back" could preempt a full application run failure. While this describes just two adjacent layers, such communication and response could extend down to the hardware and up to the application itself.

Addressing the problems depicted by such examples has predominantly been the domain of the operations community. Their inclusion in the codesign process would bring unique insights of production scenarios and approaches to solutions. Improving system management through inclusion of operations domain knowledge into codesign efforts can yield great returns since even the best hardware-informed application design will be less performant when impacted by poor resource allocation or suboptimal resiliency mechanisms. Since typical systems run at 100% capacity, substantially increased throughput on existing systems can be gained through more intelligent operations.

Opportunity:
Research challenges in this area span a broad range of middleware and edge tools and their interactions, so there is broad opportunity to innovate new services, APIs, and standards that better help these layers communicate with each other, the system, and services relaying the system state. Though much effort has already been poured into capability enhancements of each of these layers independently, it is, in many cases, blunted by an adjacent layer making ineffective use of a feature, a divergent standard for communication, or even just poor assumptions coded in that worked at the time of development. The opportunity exists to take a more holistic approach to improving a system already comprising interdependent software components that can be made more capable, reliable, and even performant through more effective communication among these components. This could take the form of earlier examples such as application-to-resource steering and sensible fault-handling logic, or leveraging metric and event data to drive intelligent service responses. An example might be as simple as having a

scheduling component pause itself during certain system conditions or events, or as complex as Machine Learning (ML) pattern identification to help pinpoint misbehaving applications or hardware resources. New tools and new collaborations among application developers, analytics developers, the modsim community, hardware developers, and, newly, operations staff will be required.

Significant new capabilities for interaction and response across a broader set of system components and software than has been addressed in previous codesign efforts will need to be developed. Research is needed to innovate on designs and to assess tradeoffs. A set of monolithic services may be simple to design, but may have little flexibility. A loosely-coupled federation of “self-aware” distributed services that interact directly with each other have more flexibility but will be far more complex to design and validate.

Enabling dynamic adaptation of resource allocation to meet application needs will require a complete redesign of scheduling/resource management and of application data exposure and feedback potentials. Resource components do not currently have a system view and have no concept of their inherent capabilities. Characterizations of resources’ capabilities need a shared language with applications and may need external measurements to provide them with a view of current headroom and possible contention through affiliations (e.g., a host may have its resources completely free but be connected to the system through a congested network switch; a host may have free GPUs but those in use are fully utilizing other subsystems and thus sharing additional GPUs would result in contention and degraded performance; likewise, memory capacity may be available while memory bandwidth is completely subscribed).

Timeliness or maturity:
The scenario of increased component heterogeneity and the desire for a richer environment with support for dynamic interaction between services, resources, and applications has existed for some time. What makes this an exciting time for tackling the problem through new and expanded codesign efforts is that hardware vendors and application programmers/users alike have accepted that we can no longer run these systems and components as black boxes due to complexities of hardware, system software, middleware, and application software, especially when coupled with extreme scale. Increases in scale have also magnified some small, ignorable problems of the past into large problems of the present.

To enable improvements in operational efficiency, both hardware and software developers have vastly increased the level of instrumentation across all components. HPC monitoring has correspondingly advanced its capabilities in lightweight and efficient methods of gathering, transporting, and exposing raw and processed information. These can be leveraged for additional purposes as well. For example, application data can be exposed by programming abstraction layers and transported by these mechanisms, as can be feedback directives to new hooks within the layers. Placement of such communications within the abstraction layer can enable interactivity with a wide variety of applications without requiring per-application changes. New open source hardware development efforts will provide new opportunities for the wider community to develop and expose telemetry data and hooks for feedback.

Advances of recent years in analytic methods, including ML, can enable distributed run time determination of best-fit resources for concurrent applications and of when to most effectively trigger new application-to-resource mappings from richer and larger datasets. Such analyses can be placed more broadly throughout the system due to increased capabilities for processing, such as high core counts and low-latency local storage. Moreover, new ML techniques like transfer learning, may make development of analytics for different architectures and applications tractable, and increasing emphasis on explainable ML may make automated architectural and high consequence resource decisions acceptable.

These current and potential advances comprise all of the elements required to create a self-managing and self-optimizing HPC ecosystem. Success will require developers in all of these areas to work together to iterate on the details of instrumentation, analysis, communication, and configuration. The impact of success in this codesign effort will be truly scalable and resilient orchestration of system components, of applications and workflows that are becoming increasingly complex and dynamic in their demands, and of resources that have additional constraints, possibly artificial. It will also enable deep insight into how to optimize resource procurements for next generation systems.

AI-enabled Analysis and Control for Enhancing Data Transition and Movement

Patricia Gonzalez-Guerrero, Anastasiia Butko,
George Michelogianniakis, John Shalf
{lg4er, abutko, mihelog, jshalf}@lbl.gov
Lawrence Berkeley National Laboratory, Berkeley, CA

A. *Topics: architectures, applications, codesign methodologies.*

B. *Challenge: data transition and movement in complex heterogeneous systems.*

Coherent cache memory has been indispensable in multi-core architectures for many decades. The cache subsystem facilitates multi-core system programming and allows developers to focus on other crucial aspects of parallel system performance. However, due to the extensive protocol-related traffic and lack of explicit data movement management, cache memory scalability as well as its use in extremely heterogeneous systems is a big concern. Moreover, the complexity of coherence protocols makes it very hard to evaluate data transition and movement using the existing tools, and inhibits our ability to make significant advances in the future.

Figure 1 shows an example of the data state transition and movement from a small scale (Figure 1.a) to a large scale (Figure 1.c) scenario. Even in the single-CPU system, data can exist in mutiple states across the system and constantly changes over time depending on algorithm nature, hardware resource utilization and various data movement protocols. Tracking these changes to better understand the patterns and use it to improve system performance and efficiency is an extremely challenging task that cannot be done without complex codesign tools.

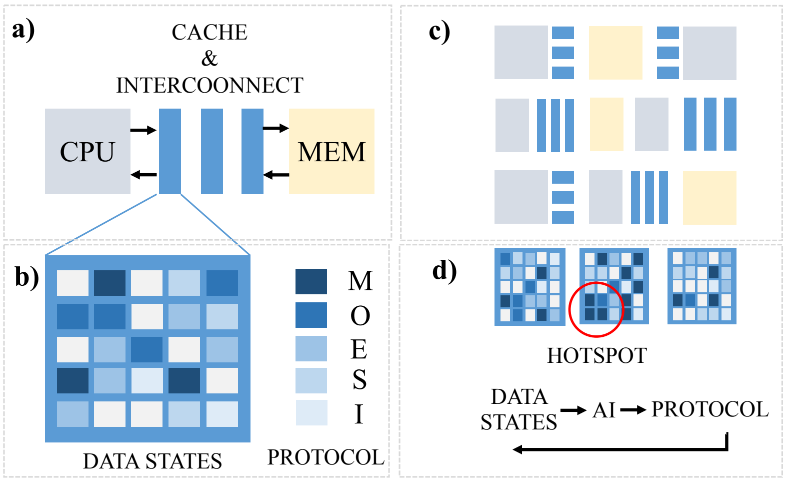


Fig. 1: AI-enabled control for Data State Transition and Movement: **a)** Simple single-CPU system. **b)** Cache Data States with MOESI coherency protocol. **c)** Complex heterogeneous multi-PU system. **d)** Data States changes over time, AI analysis and control for enhancing protocol and interconnect traffic for complex systems.

C. *Opportunity: open-source hardware and software, advanced ML and AI techniques.*

Over the past decade, various open-source hardware projects, e.g. the Rocket Chip Generator [1], appeared and changed the landscape of the architecture research and academia-industry engagement. Namely, Open Cache Coherency (Open2C) [3], demonstrates a novel methodology for architecture exploration of cache coherence mechanisms. Open-source hardware gives access to the low-level data for detailed analysis and control. Thus, the data transition and movement challenge in the era of extreme heterogeneity and large systems can be addressed using novel approaches, i.e. Machine Learning (ML) and Artificial Intelligence (AI), creating new opportunities for software-hardware codesign.

ML and AI is a known method to finx patterns in complex data sets. Moreover, novel AI techniques, such as deep reinforcement learning (DRL) [5] can be used for complex decision-making in an uncertain environment. An artificial agent may learn by interacting with its environment, similarly to a biological agent. Using the experience gathered, the artificial agent should be able to optimize some objectives given in the form of cumulative rewards. Recent studies describe how ML can improve on-chip communication [6] that shows growing interest and potential for significant impact of ML/hardware combination in the future.

Figure 1.d illustrates an example of AI-enabled analysis and control for data transition and movement with the cache data state (e.g. tag array cache line state according to the coherency protocol) that change over time. AI can detect hotspots in data states and provide feedback to the protocol improving inter-unit communication and hardware resource utilization.

D. Timeliness or maturity

The development and codesign of open-source hardware only recently become possible thanks to recent advances in high-level Hardware Description Languages (HDL) such as Berkeley Chisel [2] or PyRTL [4]. Until recently, HDL models have been considered extremely time-consuming because of the language complexity and amount of effort required for model implementation and debugging. New languages offer additional functionalities coming with functional- and object-oriented programming paradigms. With the novel capabilities, these tools can be integrated into complex software-hardware codesing workflows together with the ML and AI tools.

References

[1] Krste Asanović, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. The rocket chip generator. Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.

[2] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R Avižienis, J. Wawrzynek, and K. Asanović. Chisel: Constructing hardware in a scala embedded language. In *DAC Design Automation Conference 2012*, pages 1212–1221, June 2012.

[3] Anastasiia Butko, Albert Chen, David Donofrio, Farzad Fatollahi-Fard, and John Shalf. Open2c: Open-source generator for exploration of coherent cache memory subsystems. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '18, page 311–317, New York, NY, USA, 2018. Association for Computing Machinery.

[4] D. Dangwal, G. Tzimpragos, and T. Sherwood. Agile hardware development and instrumentation with pyrtl. *IEEE Micro*, 40(4):76–84, 2020.

[5] Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *arXiv e-prints*, page arXiv:1811.12560, November 2018.

[6] Parisa Khadem Hamedani. Improving communication in chip multiprocessors using emerging technologies and machine learning. 2017.

Planes, Trains, and Automobiles: Making Your Data Available in Time for Discovery

Quincey Koziol (koziol@lbl.gov) and Suren Byna (sbyna@lbl.gov)

Topic: Architectures, Emerging technologies

Challenge

Data movement in high performance computing (HPC) continues to be a challenging issue today and is on track to become even more complex in future systems. HPC systems are becoming increasingly heterogeneous in their compute elements with CPUs, GPUs, and special-purpose FPGAs. As a result, the memory locations that buffer for data for processing are intrinsically heterogeneous. In addition, storage devices themselves are becoming heterogeneous, with new non-volatile and memory-class memories, although traditional parallel file systems are here to stay as capacity storage. Software that manages each of these memories and storage devices are diverse, i.e., memories are managed by processor hardware and operating systems, whereas storage devices are managed by parallel file systems.

Applications are not currently designed with this complex memory/storage hierarchy in mind and typically offload data to persistent storage in a generic manner, very similar to their traditional interactions with POSIX-based file systems. Needless to say, the lack of information from the application about how important and persistent its data is presents challenges to storage system stack. High- and low-level I/O middleware, such as HDF5 and MPI-I/O, attempt to gather enough information to perform I/O operations efficiently, but are not able efficiently leverage the storage system without more information about the data from the application, as well as more information about the storage stack from the system. All these layers, applications, memory management, I/O middleware, parallel file system, and the HPC system itself, must collaboratively build better models of where, when, and how to move data to satisfy the needs of application users. A co-design effort is essential to utilize the capabilities of heterogeneous memory and storage devices to their full potential.

Opportunity

Future application co-design must focus on data movement as much as compute efficiency. Energy costs for data movement are a growing factor in system design, and are projected to be the dominant factor in the future as benefits from Moore's Law decrease density increases per silicon die and future system performance gains will likely be made through adding more cores, with corresponding increases in interconnections and therefore data movement [1, 2].

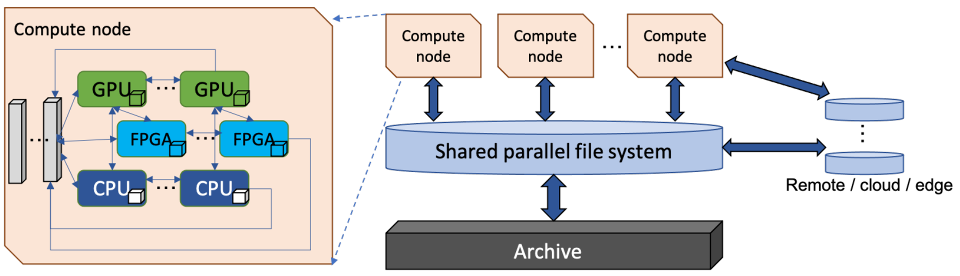
We propose three areas of focus for data movement co-design:

1. Application Guidance and Input - Applications, whether simulation, experimental / observational data focused, or AI-based, must add more information to their data movement requests. No longer can an application just make a POSIX 'write' call to send a buffer to persistent storage. The application should describe the importance and persistence of the buffer to the data movement middleware and participate in negotiating shared ownership of system resources like compute-local memory, system interconnect, node-local storage, and in-network compute with the storage middleware. Interfaces are needed for applications to express the data movement irrespective of heterogeneous end-points.

2. Data Movement Middleware - High- and low-level data movement middleware, similar to today's I/O middleware such as HDF5 and MPI-I/O, must expose a more flexible set of options for applications to choose from. These could include synchronous vs. asynchronous I/O, borrowed, shared, or transferred ownership of memory buffers, mechanisms for recognizing data importance and persistence priorities, etc. Storage middleware must also query a system's configuration: What layers are there in the memory and storage hierarchy? How much space is available in them? How are those layers connected and how fast are those connections? Only

when these questions can be answered will the storage middleware be able to optimize data movement for the application.

3. HPC Runtime Systems - Runtime systems that schedule data movement, manage available space, and place data closer to analysis capabilities can achieve high efficiency, but the systems' memory/storage hierarchy must also become "introspectable" and "programmable". Data movement middleware must be able to retrieve the system's configuration, both statically (to know maximum capabilities) and dynamically (to adjust data movement according to current system load). In addition, flexible and programmable capabilities to transfer data between all layers of the system's memory/storage hierarchy must be exposed for the storage middleware to leverage as needed.



As the diagram above shows, future HPC systems' memory/storage hierarchy will have many "moving pieces" and will need careful optimization in a concerted and organized way. Application developers alone will not wish to invest the required effort involved to wring the best performance possible from complex storage systems as their changes will require rework when porting to a new system. System vendors also don't have a vested interest in building portable data movement middleware that might benefit competitors in the field.

Instead, this era will see the rise of data movement middleware that provides a rich and flexible interface for application developers to express their data movement desires and will also query the system's memory/storage hierarchy for all the information needed to fully carry out those requests. Indeed, the "data movement middleware" that is required to help application teams and system vendors extract the maximum benefit from the underlying system will no longer be focused on merely accessing "stored" data - it will instead be tasked with the entire, complex, data movement process, from RAM to tape and the cloud, and all steps along the way.

Timeliness: These data movement codesign efforts will turn the typical compute-focused codesign effort on its head - compute may play a secondary role to the overall optimization of getting data back to science teams in an efficient, timely, and easy-to-manage way. Only now, in the last days of Moore's Law, are the costs and importance of optimizing for data movement being revealed and coming to the forefront of codesign.

References

1. P. Kogge and J. Shalf, "Exascale Computing Trends: Adjusting to the New Normal for Computer Architecture," in Computing in Science & Engineering, vol. 15, no. 6, pp. 16-26, Nov.-Dec. 2013, DOI: 10.1109/MCSE.2013.95.
2. J.S. Vetter, et al., "Extreme Heterogeneity 2018: DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity", US Department of Energy, Office of Science, Advanced Scientific Computing Research, 2018, DOI:10.2172/1473756.

Author : Allan Cantle : February 12, 2021
CEO : Nallasway, and Technical Leader : Open Compute Project, OCP HPC Sub Project
Email : a.cantle@nallasway.com Website : www.nallasway.com

Open, Modular, Flexible and Composable Domain Specific Computing

TOPIC : Hardware Architectures

Challenge

As the computing industry transitions to "Domain Specific Architectures"[1] we've seen the industry explore disaggregation and composability at both the 19" Rack Level and the Chiplet Level. However, innovation is seriously lacking in between these two extremes.

The 19" rack will be 100 Years old next year after AT&T established the standard in 1922[2] and during those 100 years, we've seen technology relentlessly shrink. Today we see the "Domain Specific Architecture" challenge addressed with disaggregation of the Compute, Memory, Networking and Storage at the 1 or more Rack Unit, U, level.

At the other end of the scale we are seeing the Silicon industry tackle the "Domain Specific Architecture" challenge with multiple Chiplets in a substrate with 2D, 2.5D and now 3D innovations. These devices are significantly increasing Power In and Thermal Management requirements, as well as delivering IO speeds >100Gbits/s/lane with their associated signal integrity and power challenges.

It has become increasingly stark that our industry requires new system level modular solutions that truly allow for full composability with the highest performance, best efficiency and lowest cost possible. The biggest challenge the industry faces in achieving this is in how to collaborate across the entrenched industry standard Silo's that have been established with today's large incumbent players. Our industry needs to Reimagine how to CoDesign across these Silo's in order to innovate revolutionary hardware architecture solutions that are correctly tailored to the needs of a "Domain Specific Architecture" world.

Opportunity

We have an opportunity to restate the physical building blocks of computing such that we can dramatically improve Performance while simultaneously reducing Power and Cost. We can address this by taking a step back and analyzing the modularity, flexibility and composability needs from a holistic, clean slate, Systems Engineering perspective. To be successful, a CoDesign approach will be required between experts in the following areas : Systems, Silicon, Packaging, Signal Integrity, Power, Thermal and Mechanical.

The ideal architecture would allow a Systems Architect to quickly design and build architectures with the right balance of resources that suit any particular application in a Lego Brick fashion. Ideally, one could easily construct architectures that would have any mix of the classic compute ratio of Ops : Bytes/s : Bytes Capacity, be it 100 : 1 : 0.01 or 1 : 1 : 1 or 0.01 : 1 : 100 etc.

A measure of success would be that a range of differing applications with different resource needs could all be specifically built and would deliver Best In Class Performance, Efficiency and Cost for every build scenario. Additionally, Systems could be easily constructed such that they were easily tailored to Data Center, On-Prem, or Edge implementations.

The architectural building blocks would be fully Open so that providers of the computing components from startups to incumbents could provide the interoperable modules that would allow the systems architects to construct the most efficient solutions from a multitude of "best in class" suppliers.

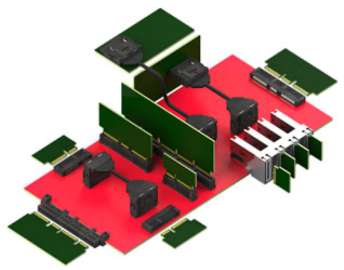
When it comes to composability, the SerDes Transceiver has become the most common level of interconnect that all components support. The SerDes is also the point of transition from electrical to optical communications. So true composability at the interconnect level is now within reach.

The Open Compute Project[4], OCP, who's core focus is open hardware architectures would be the ideal home for this effort.

Timeliness

The timing is excellent as the industry sees a proliferation of innovation in all aspects of computing from a diverse range of processors to new memory and storage technologies. The latest Top500 HPC Machines, Fugaku and Summit, demonstrate very different implementations[5] that could potentially be built out of a common set of open and modular building blocks. Modularization is happening at the Rack Unit level but the higher speeds are needing retimers to be used like confetti in order to build reliable and stable communications channels even for the shortest of hops. e.g. the latest 112Gbps SerDes channels don't have the budget to reach across a traditional Server Motherboard PCB[3] without using a Retimer that consumes power; adds latency and cost without providing any utility!

The theme of modularization has been around for a while and Module / interconnect standards like OAM[6] and EDSFF[7] are gaining traction, while still being constrained by the 19" rack traditions of our industry. We have an opportunity to both be inspired by these efforts as well as leveraging them to Reimagine and CoDesign them for a significantly improved Industry Standard Modular, Flexible and Composable Computing Architecture for the future.


EDSFF Interconnect Possibilities

References

1) John L. Hennessy, and David A. Patterson. "Domain Specific Architectures," in Computer architecture: a quantitative approach, Sixth Edition, Elsevier, 2018.

2) Wikipedia - 19-Inch Rack - https://en.wikipedia.org/wiki/19-inch_rack

3) DesignCon 2019 112-Gbps Electrical Interfaces: An OIF Update on CEI-112G - Slide 38 - https://www.slideshare.net/LeahWilkinson3/designcon-2019-112gbps-electrical-interfaces-an-oif-update-on-cei112hg

4) OCP Website - https://www.opencompute.org

5) OCP HPC Subproject : Slides 3 and 6 - https://drive.google.com/file/d/1PUoiXrK62pNwqtp_y2lRt11_Bx9CZwZ3/view

6) OAM - https://www.servethehome.com/facebook-ocp-accelerator-module-oam-launched/

7) EDSFF Interconnect - https://www.molex.com/molex/products/family/sliver_edgecard_connector_and_cable_assemblies

Achieving Extreme Heterogeneity: CoDesign using Neuromorphic Processors

Suma George Cardwell, Frances S. Chance, Craig M. Vineyard and James ‘Brad’ Aimone
Cognitive and Emerging Computing, Sandia National Laboratories
Email: sgcardw@sandia.gov

Topic: Architectures, Applications, Emerging technologies, Codesign methodologies

As conventional systems saturate in power efficiency, innovations in both architectures and algorithms are required to meet the computing needs of the future. With the slowing of Moore’s law and the recent popularity of deep neural networks there has been renewed focus on emerging technologies, such as neuromorphic computing. Inspired by the brain, neuromorphic architectures leverage properties such as massive parallelism, sparse activity, and event-driven computing. Neuromorphic computing has the potential to be impactful for machine learning, scientific computing, modeling cognitive tasks as well as applications at the edge. Codesign tools are critical for the adoption of such novel technologies. If designed into a heterogeneous system with other accelerators and conventional computing platforms, this technology has the potential to augment the capabilities of High Performance Computing (HPC) platforms (1). This paper highlights the need for new heterogeneous tools and architectures through the lens of neuromorphic computing.

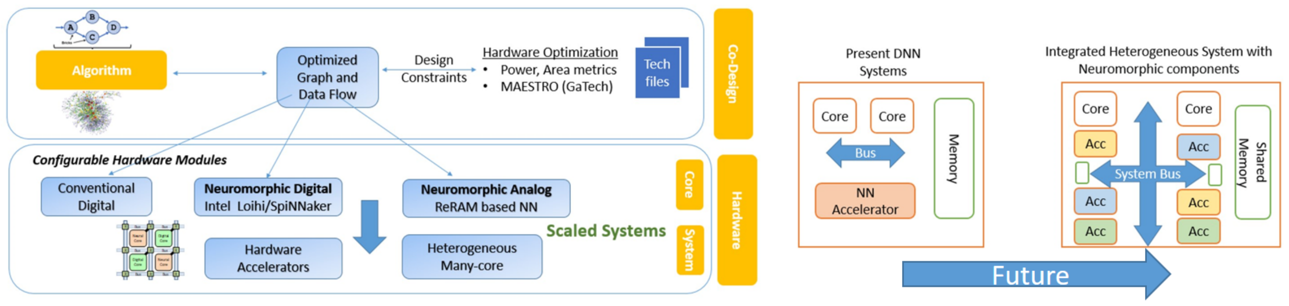


Figure 1: Developing Heterogeneous Architectures will require codesign tools that span algorithms and hardware. The future of computing will likely be extremely heterogeneous with different accelerator types. Image reproduced from (1)

Incorporating different classes of processors on single HPC node has been key to moving towards exascale computing. The scientific computing ecosystem is also changing with data collection outpacing theory in many fields like neuroscience, medicine, and, climatology. There has been explosion of different accelerator approaches in industry like TPUs, Cerebras (wafer-scale), Mythic (analog), each with a unique approach to overcoming performance bottlenecks. It is evident that future HPC approaches will be highly heterogeneous, where HPC system could include both conventional (CPUs, GPUs), and non-conventional approaches (neuromorphic hardware, Processing-In-Memory). Digital neuromorphic chips like Intel’s Loihi have shown 100x efficiency gains compared to GPUs and CPUs and can be scaled to build larger systems (2). Analog neuromorphic architectures promise even further savings in energy efficiency, area, and latency than their digital counterparts (3; 4).

Current Challenges While a lot of progress has been made in codesign methodologies and their adoption, a major gap exists in the integration of novel computing paradigms like neuromorphic computing in heterogeneous computing. This is in part due to the lack of a cohesive codesign tool and also due the diverse nature of neuromorphic backends which range from digital, analog, mixed-signal to beyond-CMOS approaches.

Challenge#1: Codesign tools that support novel architectures. A lot of work is being currently done to incorporate GPUs, FPGAs to build heterogeneous systems. This is in part due the API and prototyping tools that are made available, as well as the ease of access to these devices for testing and validation. Such a framework is missing for neuromorphic processors– which are still evolving and have diverse approaches to the architecture and devices used.

Challenge#2: Developing applications for neuromorphic from HPC to the edge. Challenge 1 plays into challenge 2 in that, lack of codesign tools and ease of usability limits the different applications users can develop for these novel architectures. Thus, the barrier of entry is high, which hinders adoption.

Challenge#3: Exploration of next-generation heterogeneous neuromorphic architectures. We need to explore complex neurons and connectivity mechanisms that would make neuromorphic systems even more capable and apply to diverse set of problems. This will also include exploring novel devices, new integration techniques (3D architectures, photonics) and novel algorithms that exploit their characteristics.

Opportunity: The identified challenges present many opportunities through new tools and techniques, new technologies, and groups collaborating through open-source tools in the codesign process.

Codesign tools for Heterogeneous Neuromorphic Architectures: Accelerator tools have democratized the ability to test and validate different dataflow architectures. The neuromorphic field needs such open-access codesign tools available to the larger community that supports varied backends. This could entail analytical tools, cycle-accurate tools, as well as tools that enable exploring integration of neuromorphic accelerators with conventional processors. We can leverage deep learning accelerator modeling tools like Timeloop(5), MAESTRO (6), and NVDLA to explore heterogeneous architectures by extending them with different analog and digital neuromorphic kernels. Analytical tools work through computing energy related operations (number of memory read/write, number of MAC, NOC communications) given a certain technology node. These tools are beneficial for rapid testing and architecture prototyping. Such tools can feed into cycle-accurate explorations, e.g. Sandia's Structural Simulation Toolkit (SST) that can yield cycle-accurate simulations. Other tools that account for the performance of emerging device technologies are CrossSim and PUMA (3; 7). High level tools like Sandia's Fugu, are also needed to enable designing spiking neural networks while being hardware agnostic. A modular approach to codesign tools is also important, especially tools that enable integration of new architectures and device characteristics. This requires not just looking at non-Von Neumann architectures but also novel non-CMOS devices. While the software space is constantly evolving, building tools that can be re-usable, open, adaptable will be crucial to adoption.

Impact diverse set of applications: Many applications have been demonstrated for neuromorphic systems. Examples include solving PDEs using random walks on a neuromorphic platform (8), and a spiking implementation of Locally-competitive algorithms (LCA) (9) that implicitly solves the LASSO optimization problem with improved energy costs compared to conventional solvers (10). Our hypothesis is that multi-precision networks using neuromorphic processors will perform better than conventional computing approaches for scientific computing and machine learning algorithms. Recent programs like DARPA FENCE, focus on neuromorphic event sensor and processors that will bring low SWaP advantages to the edge. Thus, diverse applications will facilitate the development of new architectures that support a diverse set of algorithms and create an eco-system where users inform neuromorphic hardware developers. There is a tradeoff in codesigning applications and hardware, but applications are no longer immune to the hardware they run on, to gain performance benefits. This requires a strategic investment in codesign tools and approaches. Perhaps, the development of useful mini-apps for such heterogeneous architectures will be a good first step in this direction. Collaboration between industry, academia, and research laboratories is also important. Efforts like Intel's Neuromorphic Research Community and outreach by IBM (TrueNorth) for academic and research partners are good examples of this.

Next-generation extremely heterogeneous architectures: While we need novel neuromorphic devices to accelerate computation, we also need novel algorithms and architectures. A lot of current neuromorphic hardware uses simplified neuron models that can be scaled to billions of neurons. However, we hypothesize that designing complex neurons will augment the capabilities these systems currently offer. For example, introducing dendritic processing will introduce nonlinear summation, spatio-temporal processing, and increased connectivity. Techniques to do brain-inspired local learning is another area of active research that could impact the use of neuromorphic processors as not just inference but training engines. Next-generation neuromorphic circuits and systems based upon nonlinear dendritic processing and local learning will balance the trade-off between scalability and the biological complexity. Novel approaches in fabrication like three-dimensional architectures and wafer-scale technology as well as in-memory computing devices could further alleviate current communication and connectivity bottlenecks. This would require synergistic collaboration across devices, architectures, software, and algorithms.

Timeliness or Maturity Neuromorphic architectures have the potential to have an impact in the next 5-10 years as implementations in silicon exist today (Loihi, TrueNorth, SpiNNaker, offerings from BrainChip, GrAI Matter Labs). Non-CMOS approaches are promising and industry trends (Imec/Global Foundries) show that these architectures will be available for mass production soon. Neuromorphic accelerators can impact the efficiency of machine learning, scientific computing, and edge applications with two-three orders of magnitude improvement in energy and speed. Codesign tools will enable algorithm and hardware designers to account for novel accelerators in their design flows better. Hence, developing open-source codesign tools that include a wide variety of novel backends like neuromorphic processors is imperative to achieve extreme heterogeneity in the future.

Acknowledgments Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] S. G. Cardwell, C. Vineyard, W. Severa, F. Chance, F. Rothganger, F. Wang, S. Musuvathy, C. Teeter, and J. B. Aimone, "Truly heterogeneous HPC: Co-design to achieve what science needs from HPC," *Smoky Mountain Computational Sciences and Engineering Conference*, Accepted for Talk, Publication 2020.

[2] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*, 2019, pp. 1–8.

[3] S. Agarwal, A. Hsia, R. Jacobs-Gedrim, D. R. Hughart, S. J. Plimpton, C. D. James, and M. J. Marinella, "Designing an analog crossbar based neuromorphic accelerator," in *2017 Fifth Berkeley Symposium on Energy Efficient Electronic Systems & Steep Transistors Workshop (E3S)*. IEEE, 2017, pp. 1–3.

[4] J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in neuroscience*, vol. 7, p. 118, 2013.

[5] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2019, pp. 304–315.

[6] H. Kwon, M. Pellauer, and T. Krishna, "Maestro: an open-source infrastructure for modeling dataflows within deep learning accelerators," *arXiv preprint arXiv:1805.02566*, 2018.

[7] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-m. W. Hwu, J. P. Strachan, K. Roy *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 715–731.

[8] W. Severa, R. Lehoucq, O. Parekh, and J. B. Aimone, "Spiking neural algorithms for markov process random walk," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[9] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural computation*, vol. 20, no. 10, pp. 2526–2563, 2008.

[10] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

Bridging the Co-Design Gap

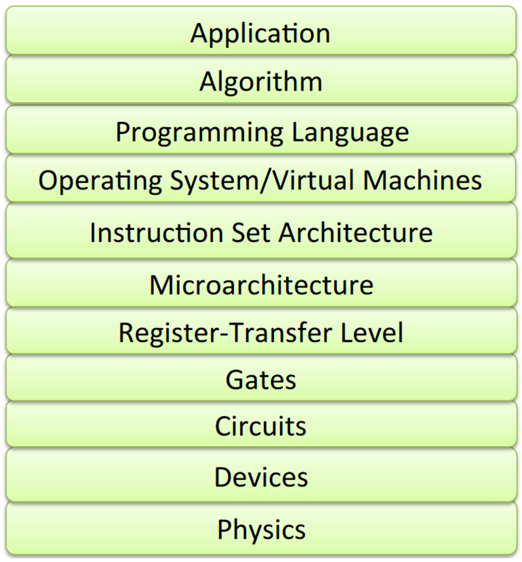
Cy Chan, Maximilian Bremer, Aydin Buluc, Xiaoye Li, James Sethian, John Shalf

Topics: codesign methodologies, applications, programming systems, modeling and simulation

Challenge: The current trend towards hardware specialization has created an increasingly diverse landscape of architecture choices for co-design [1]. With this complexity, it is becoming difficult for domain experts to effectively reason about the hardware design space without first developing a deep knowledge of computer hardware and architecture. Similarly, architecture experts do not always understand all of the algorithmic degrees of freedom and constraints in each application domain.

A review of co-design strategies found that they can typically be categorized as either architecture-centric or application-centric efforts [2]. **Architecture-centric** efforts are commonly framed around optimizing specific mini-applications, where the algorithms' performance characteristics remain relatively static. However, such an approach significantly limits the flexibility of design decisions at the level of the application. Since algorithmic advancements have been as important as hardware and architecture advancements in driving long-term computational performance [3], it is critical to develop co-design methodologies that are flexible enough to incorporate different algorithmic approaches to solving computational problems. In contrast, **application-centric** co-design efforts typically consider optimizing applications over a subset of potential architectures. An example of this would be high order finite element discretizations which arose from the need for higher arithmetic intensity algorithms to match the flop/s-bandwidth balance of current supercomputers. Work in this area has made tremendous advances in improving the stability of these algorithms, but not all applications exhibit the required solution smoothness to fully profit from these approaches. Future co-design approaches should consider a wide variety of hardware accelerator archetypes, balancing algorithmic advances against practical application-driven constraints to further optimize scientific output.

Ideally, co-design would involve optimization across all layers of computing abstraction (see Figure [4]) to maximize performance; however, since it is very difficult for any human to simultaneously master all layers, we require new tools and techniques to facilitate, automate, and accelerate those interactions. The challenge for the co-design community is to provide a mechanism to *bridge the gap* between the *architecture-centric* and *application-centric* co-design patterns, providing a path for domain experts who understand the application design space to navigate the architectural design landscape in a meaningful way that does not overconstrain the algorithmic approach.



Opportunity: The envisioned research would build methodologies and tools to make architectural co-design exploration easier for domain scientists to investigate application or algorithmic reformulations. A key research topic is to identify the appropriate programming models and software abstractions that can both enable scientists to express their algorithms at a high level and also interface with hardware performance modeling and simulation tools to estimate their performance with various hardware parameterizations (speeds, capacities, etc.) With such an interface established, the hardware acceleration techniques could then be cleanly hidden behind software abstractions, e.g. hash-tables for vertex matching or task-based programming models for message queues.

An accelerated hardware simulator could then apply the appropriate cost models to each of the primitives as they are executed. The abstractions would provide interception points for the modeling tools, and may occur at the individual instruction level (e.g. fast atomic support), library call level (e.g. accelerated hash table operations), or even implicitly at the programming model or runtime level (e.g. accelerated task dependency handling or message queues). The methods used to estimate performance may rely on a combination of hardware simulation and performance modeling techniques, depending on the scale of the problem. This approach would also yield the benefit of having runnable programs that the developer can use to verify program correctness and evaluate numerical accuracy in conjunction with estimated performance.

Timeliness: Recent successes in algorithmic optimization, performance modeling and simulation, and architectural optimization will help guide the identification of these flexible, productive programming abstractions between the domain experts and the underlying hardware design space. For example, using content-addressable memory to implement index matching in hardware for sparse graph applications [5]; using software-assisted hardware prefetching to accelerate iterative solvers commonly used in conservation laws [6]; and using hardware message queues to accelerate DAG-based programming models for sparse direct solvers [7] have shown substantial speed-ups over state-of-the-art implementations for current general-purpose computers.

Given their demonstrated benefits, we believe that accelerator technologies have enormous potential to improve the performance of many applications, and by allowing domain experts to more easily explore new algorithms, we will maximize their impact and increase their rate of adoption. The end of Moore's law foreshadows the emergence of domain-specific architectures, and with chiplet-based hardware designs, it may be possible from a techno-economic standpoint to assemble these architectures in the near future. By giving domain scientists the means to express their application-specific hardware needs and conveying what tools are available, they will be able to improve scientific output through novel algorithms that efficiently leverage future specialized hardware.

References:

1. William J. Dally, Yatish Turakhia, Song Han, "Domain-Specific Hardware Accelerators," Communications of the ACM, July 2020.
2. Ang, James A. "High performance computing co-design strategies." *Proceedings of the 2015 International Symposium on Memory Systems*. 2015.
3. Leiserson, Charles E., et al. "There's plenty of room at the Top: What will drive computer performance after Moore's law?." Science 368.6495 (2020).
4. Wentzlaff, David. "Introduction and Instruction Set Architectures." Lecture Notes, Computer Architecture ELE 475, Princeton University, delivered Sept. 14 2016.
5. Zhu, Qiuling, et al. "Accelerating sparse matrix-matrix multiplication with 3D-stacked logic-in-memory hardware." *2013 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2013.
6. Zhang, Chao, et al. "RnR: A Software-Assisted Record-and-Replay Hardware Prefetcher." *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020.
7. Ding, Nan, et al. "Leveraging One-Sided Communication for Sparse Triangular Solvers." *2020 SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 2020.

The Case for CoDesign

David Donofrio
Tactical Computing Laboratories
ddonofrio@tactcomplabs.com

Topics: Architectures, modeling and simulation, codesign methodologies

1 Challenge

As we continue to approach the end of traditional performance scaling we see a rise in specialization as key to maintaining continued performance gains. To influence the direction of this specialization significant investments have been made in commercial vendors, such as the PathForward and FastForward programs, for development of features beneficial to science and HPC. However, these programs are still strongly tied to existing vendor roadmaps and do not provide sufficient differentiation from what is available as standard COTS offerings. These programs demonstrate the recognition that the HPC community needs solutions that differ from what is available in the commercial market. While the PathForward and FastForward programs have produced demonstrably positive outcomes they also represent a missed opportunity as the HPC community should be *driving* hardware innovation not reacting to it. The scientific community has, arguably, already driven innovations that are now critical parts of commercial products. An argument can be made that GPGPU was first driven by the HPC market and perhaps more importantly, was deployed, debugged and used by a broad community of expert developers who worked closely with vendors to tailor the software needs and act as effective beta-testers.

Even with these investments in vendors, we see certain applications, such as climate and bioscience, are not benefiting from current system architecture trends and are increasingly left behind by HPC centers focused on utilizing GPU-based accelerators to achieve performance. The trend towards performance optimization of lower-precision arithmetic driven by the rise of AI algorithms is at odds with the needs of many science applications. As fewer customers request double precision support, or become concerned with the performance of double precision processors, vendors will need incentives to keep these features in place.

If the HPC community wishes to continue to enjoy the year-over-year performance gains that have continued over decades it is no longer sufficient to simply fund the existing vendors to produce tweaked versions of their existing products, the DoE and the larger HPC community must begin to take on the co-design challenge in earnest.

2 Opportunity

There is an opportunity to create not just one, but multiple HPC systems each tailored to a subset of HPC applications. Much like the commercial cloud vendors are beginning to differentiate themselves through various hardware offerings, future HPC centers could distinguish themselves by being particularly optimized to a subset of applications. This diversification of hardware architectures may result in a decrease in code portability, but these trends are already emerging. At a simple level, there is the CPU vs. GPU code base, but, due to differences in both hardware and software architectures, code that runs well on one vendor of CPU, and even more so on GPUs, will not always seamlessly port to the hardware of another. By tailoring systems to classes of applications we may lose code portability but this loss can be more than offset by gains in code efficiency, programmability and developer productivity.

The DoE should invest in leading edge technology - both software and hardware, to drive innovation and performance - especially for applications and users who have been left behind by current technology scaling. This is not to say application developers should not continue to take advantage of the latest technology, but more that the DoE should engage in *true* co-design. In a more aggressive co-design strategy the DoE can first articulate the scientific challenge(s) to be addressed then procure targeted machines to address each of the science areas. While representative benchmarks are certainly used in present day machine procurements, many systems outside of those targeted at specific national security interests are of a *one-size-fits-most* design, inevitably leaving users with diminished performance scaling.

This shift towards specialization should be accompanied by a model that views the various DoE datacenters as an aggregate compute resource rather than a collection of independent centers. This aggregate model allows each center to deploy hardware solutions that target a subset of application domains. This trend away from a *one-size-fits-most* model will place additional demands on connectivity between centers to allow data to freely move between sites. This specialization by center is already being observed in the cloud space with Google offering TPU-accelerated nodes as well as the FPGA and ARM offerings from Azure and AWS. The DoE has an advantage over the siloed commercial cloud providers in that users can be encouraged to utilize resources across centers; a further extension to the *Superfacility* concept.

3 Why now?

Many are familiar with the ASCI Program where the DoE co-designed systems with vendors such as Intel and IBM. The world is certainly different than what existed 20 years ago, however, specialization of computing remains one of the best methods for maximizing performance-per-watt. It is possible *today* to begin engaging with vendors, such as those producing ARM or RISC-V based cores, to begin designing systems better tailored to the HPC community. The DoE can be a driving force for computing innovation through targeted investments in both science optimized hardware and the software required to effectively utilize these new machines.

There is an explosion in availability of open source hardware led by the RISC-V foundation. Even more importantly, there is a robust software ecosystem to support these hardware designs. Going further there are now complete open source hardware design and synthesis flows that allow a designer to go from concept to GDSII and fabrication for little to no cost. We can begin to leverage these flows to redefine the traditional relationship DoE has had with system vendors. Investment in robust, community supported, open source technologies will strengthen their capabilities and can allow the DoE to *drive innovation* by producing prototype systems- possibly serving as a demonstration vehicle and leading to more productive relationships between vendors and HPC system architects.

As impressive as these existing open source flows are, processor design, simulation and manufacture remain daunting tasks. Few open source flows rise to the standards of performance and reliability that a production HPC center would feel comfortable deploying racks of such a device. Furthermore, processor design remains a hard problem with challenges going beyond the many technical hurdles to include legal/IP issues and basic economics. We are still relatively early in the open source hardware movement and there is ample opportunity to begin to engage with this community and improve these design flows and drive features important to the scientific community.

The DoE and larger scientific community is no stranger to building custom electronics. One only needs to look at the custom sensors created for the cosmology or high energy physics community. We know how to build incredibly complex and bleeding edge solutions tailored to the needs of scientific discovery, however, we have considered the commercial offerings to be sufficient for our scientific computing needs. We are not alone in needing custom computing solutions and an opportunity exists for the DoE to once again become a leader in computing by leveraging a robust, community supported, open source hardware ecosystem to produce multiple tailored, co-designed systems suited to the needs of the scientific community.

Let Co-design be About Co-design

A. Dubey
Mathematics and Computer Science Division
Argonne National Laboratory, Lemont, IL 60439

February 16, 2021

The word co-design in my mind implies that persons with diverse knowledge and expertise come together to design systems that are interdependent on one another in such a way that the outcome is superior to what it would have been had they not "designed" together. From all that I know of the embedded systems community this is how their co-design works. There is design on both sides, there are negotiations over trade-offs carried out in good faith, and the final goal is well defined. That implies that the constraints on the design space are understood. The operative words here are "design", "trade-off", "known outcome", and "constraints". In the context of DOE, co-design has meant different things at different times, but the words above could not fully be applied to any of them. Not least because we have yet to acknowledge the need to investment in software "design" without which co-design becomes more of a one way process.

Additionally, In a real co-design all parties come to the table prepared to give up something important, and that is decision making autonomy. They start from a place of trust that the other parties have similar commitment to the process. Obviously, they also assume that all parties are prepared to do their homework in terms of understanding each other's work, taxonomy, and possibilities and constraints. These concepts are at cross purpose with the siloed culture that prevails in research environments in general, and to a large extent in the DOE labs in particular.

The first iteration of co-design funded by the DOE, between specific applications and platforms, revealed that clever hardware features do not become available to applications because layers of system software and other supporting software sit between the application and the hardware. And little is to be gained by trying too much specialization at the level of what application wants from the hardware. In my mind the main lesson learned was that co-design is best between two adjacent layers in the computational ecosystem. Perhaps others agreed with this, because the second iteration, in the ECP project, focused on software infrastructure co-designed with the applications that use it. There are no intermediate layers between the two so the concerns of the applications can be directly translated into features that the corresponding infrastructure implements. The developers of the infrastructure ensure that these desired features also make good use of the underlying hardware.

The contrast between the two iterations of co-design also points to path forward. If one views the entire computational eco-system as composed of layers, then deep co-design makes sense between two consecutive layers. While that is a necessary condition, it is not sufficient. Information also needs to percolate out to the layers that are further apart. For example, while the ECP co-design centers have successfully abstracted out some platform details from their client applications, they cannot completely eliminate the interaction between the application and the hardware. This is because the services provided by the co-design infrastructure may not cover all needs of an application. Therefore co-design is not a single overarching process, instead, there may be a place for hierarchy and nesting within the co-design process.

To fully unlock the potential of co-design, however, the entire culture of research in DOE must adapt to the changing world of scientific discovery. The old structure of siloed research is fast becoming untenable in many areas of mission critical work. Breadth of knowledge in a research team is becoming at least as important as depth has been in the past and continues to be. However, breadth is more difficult to achieve because it requires that a subset of team members have cross-cutting knowledge and expertise and can contextualize research discussions for those who do not. Additionally, team leaders have to be cognizant of different cultures that team members come from and work towards keeping the team coherent. The essential elements of a successful co-design endeavor require that DOE teams begin to resemble the teams in embedded systems. While it is true that an end goal cannot be defined as concretely in research environments, greater clarity of where the co-design effort is headed will certainly help. Similarly, trust building among people working together is as important as the technical knowhow in the team to ensure that genuine give and take can flourish. It is important to keep in mind that teams fail more often for sociological reasons than due to lack of technical expertise. As long as individuals have incentive not to share their best ideas, or be prepared to give up on them on them if needed, co-design will continue to give suboptimal outcomes.

Resilience by Codesign (and not as an Afterthought)

Christian Engelmann: engelmannc@ornl.gov
Oak Ridge National Laboratory, Oak Ridge, TN, USA

Topics: Modeling and simulation, codesign methodologies

Challenge: Resilience, i.e., obtaining a correct solution in a timely and efficient manner, is one of the key challenges in extreme-scale high-performance computing (HPC). Extreme heterogeneity, i.e., using multiple, and potentially configurable, types of processors, accelerators and memory/storage in a single computing platform, will add a significant amount of complexity to the HPC hardware/software ecosystem.

The notion of correct computation and program state assumed by users and application developers today, which has been based on binary bit-level correctness, will no longer hold for processing elements based on some emerging technologies, such as neuromorphic computing elements. The diverse set of compute and memory components in future heterogeneous systems will require novel hardware and software resilience solutions. Errors and failures reported by heterogeneous hardware will need to be handled by the appropriate software component to enable efficient masking, recovery, and avoidance with little burden on the user. Similarly, errors and failures reported by the software running on heterogeneous hardware need to be equally efficiently handled with little burden on the user.

This requires a new approach, where resilience is holistically provided by the HPC hardware/software ecosystem. The key challenge is to codesign extreme heterogeneous HPC systems with (1) wide-ranging resilience capabilities in architecture, system software, programming models, libraries, and applications, (2) interfaces and mechanisms for coordinating resilience capabilities across diverse hardware and software components, (3) appropriate metrics and tools for assessing resilience, and (4) an understanding of the performance, resilience and energy trade-off that eventually results in well-informed system design choices.

The current state of practice for HPC resilience is global application-level checkpoint/restart, a single-layer approach that burdens the user with employing a resilience strategy at extreme coarse granularity (the job level). Part of the current state of practice for HPC resilience are also hardware solutions at extreme fine granularity, such as SECDED ECC for main memory, caches, registers and architectural state, Chipkill for main memory, and redundant power supplies and voltage regulators. RAS management systems are deployed for monitoring and control. The state of resilience research is more advanced and includes a number of technologies, such as fault-tolerant programming (fault-tolerant MPI, re-execution of failed tasks and containment domains), proactive fault tolerance using migration of computation away from components that are about to fail, and resilient solvers with recovery, compensation or self-stabilization properties. Recent work made inroads in understanding the fault, error and failure models of HPC systems. Some work in understanding the performance/energy and performance/resilience trade-offs exists as well. Other recent work pioneered the concept of design patterns for a structured approach to HPC resilience.

Hardware/software HPC codesign for resilience is mostly nonexistent at this point! There are a few concepts, models and tools, investigating and comparing individual resilience technologies and their performance/resilience trade-offs, such as for checkpoint/restart and redundancy. There are no design space exploration tools investigating the performance, resilience and energy trade-offs of different compute node or HPC system hardware/software designs. As a result, HPC resilience research solutions are not adopted in practice, as it is unclear if their benefits warrant adoption costs. Another result is the inability to mitigate unexpected reliability issues in HPC systems with the employed resilience technologies. A prime example is the impact of unexpected GPU failures on ORNL's Titan. The system was never designed to handle the resulting MTBF of 2 hours, requiring replacement of 11,000 out of 18,688 GPUs.

Opportunity: Coordinated cross-layer and adaptive resilience solutions can offer efficient error and failure masking, recovery, and avoidance at the appropriate hardware or software component and compute or data granularity. While the various heterogeneous compute and memory components will have hardware resilience mechanisms, software-based solutions to fill gaps in detection, masking, recovery, and avoidance of errors and failures will require coordination between the multiple layers of the system by design. Based on the underlying execution model and intrinsic resilience features of the hardware, the various components in an extreme heterogeneous system may be organized into predefined protection domains. Coordinated resilience solutions will handle errors and failures in specific components and granularities where it is most appropriate to do so and in coordination with the rest of the system, which prevents errors from propagating and failures from cascading beyond the protection domains.

This approach also removes some of the complexity that is introduced by extreme heterogeneity. Adaptive strategies can leverage the unique capabilities of heterogeneous protection domains, since the performance, resilience and energy profiles of each domain are different. Programming models and runtime environments may dynamically configure an application to use specific components in the heterogeneous system based on performance, resilience and energy costs. For example, critical computation may be executed on more resilient components; computation on less resilient components may be checked for errors with computation on more resilient components. Critical data may be stored solely or at least backed up on more resilient storage. Holistic cross-layer and adaptive resilience essentially provides efficient end-to-end resilience by design for computation and data.

Resilience needs to become an integral part of the HPC hardware/software ecosystem through codesign, such that the burden for resilience is on the system by design and not on the operator or user as an afterthought. Understanding the performance, resilience and energy trade-off is key to solving the resilience challenge for extreme heterogeneity, which is to design a reliable system within a given cost budget to achieve an expected performance. Design choices are based on a detailed understanding of this trade-off, which is HPC system and HPC application specific. Future research in hardware/software HPC codesign for resilience needs to address the following aspects:

- Develop an understanding of the error and failure characteristics of hardware and software components.
- Identify protection domains, interfaces and mechanisms of resilience capabilities in hard- and software.
- Design interfaces and mechanisms for coordinating resilience capabilities and quality of service requirements across hardware and software components.
- Define uniform metrics for assessing performance, resilience and energy across heterogeneous components to enable design trade-offs.
- Create design space exploration tools to understand the performance, resilience and energy trade-offs between different node and system designs.

Timeliness or maturity: The state of research for HPC resilience is rich in mechanisms that can be utilized. However, a longer-term and coordinated codesign effort is required to enable wide-ranging resilience capabilities in practice and to make them an integral part of the HPC hardware/software ecosystem. Research in defining, communicating and matching HPC resilience capabilities with quality of service requirements is required as we transition to extreme heterogeneity, including creating best practices and standards for resilience. Recent work in fault models, trade-offs and resilience design patterns can form the basis for solving the challenges. However, more research in (1) uniform metrics, (2) performance/resilience/energy trade-offs and (3) design space exploration tools is still required.

Simply put, if resilience by design is not done now, in the early stages of extreme heterogeneity, the current state of practice for HPC resilience, global application-level checkpoint/restart, will remain the same for decades to come due to the high costs of adoption of alternatives later on. The prime example for this is MPI, for which, 25 years after its first standardization, resilience is still not part of the MPI standard, despite 20 years of research in fault-tolerant MPI, numerous research prototypes and a 10-year discussion in the MPI standardization body. PVM, MPI's predecessor, was fault tolerant in 1993! In contrast to the MPI standardization effort 25 years ago, the current state of research for HPC resilience is far beyond the current state of practice. The existing knowledge, experience and prototypes serve as a foundation for making resilience an integral part of the HPC hardware/software ecosystem.

Real-time data-driven codesign of hardware and software for analysis of high-throughput scientific experiments

Thomas Flynn (tflynn@bnl.gov) Adolfy Hoisie (ahoisie@bnl.gov)

Topics: codesign methodologies, architecture, runtime systems, modeling and simulation

Challenge: The latest-generation scientific imaging systems generate data at an exceedingly high volume and velocity. These trends are due to increasing spatial and temporal resolutions achievable with new imaging technology. For instance, recent scanning electron microscopes can produce up to 50 GB/s of data [1]. To make the most of these new technologies, scientists must be able to process imaging data quickly, extract scientific insights, and rapidly incorporate them into new imaging settings or experiments. This means data need to be ingested, analyzed, and results returned back to the user in near real time. These analysis tasks can range from simple denoising or image processing to machine learning (ML) workloads to numerical simulations. For example, in electron microscopy, it is of interest to employ experimental data as an input or boundary condition to molecular dynamics simulations used to monitor sample degradation during imaging.

To reach the level of throughput necessary for this tight coupling of experiments and analyses, there must be close coordination between experimental hardware (edge computing), storage, and computing systems—at both lower (systems/hardware) and higher (algorithmic) levels of the hardware-software (HW-SW) stack. This can range from tuning buffers and storage placements on high-performance storage systems to mapping compute kernels to cores of artificial intelligence (AI) accelerators to choosing the right size of ML model that can meet latency constraints placed on the analysis by the interested scientist. Furthermore, experiments can be dynamic and involve changing conditions, which have important implications for analysis. Different experiments, or even specific parts of a single experiment, may call for disparate imaging settings (e.g., spatial resolution or frame rate) and distinct algorithm settings (e.g., accuracy) that place diverse requirements on latency. Thus, a single mapping of SW onto HW will not suffice. Although there has been some work on data-driven approaches to mapping ML workloads onto HW for optimal performance, existing approaches, such as device placement [2] or methods of resizing neural networks [3], remain mostly static in nature, i.e., they determine a mapping of SW to HW *a priori*. However, dynamic approaches that can find new mappings or placements on the fly as conditions change or as new criteria (e.g., latency, accuracy, or criteria of scientific interest) are introduced are needed.

Opportunity: Codesign should not be interpreted only as a static process, where we, for example, design a processor that optimizes the performance of a tailored application workload. Instead, the need for dynamic (or real-time) codesign is an imperative for the now dominant data-driven workloads and heterogeneous architectures that require optimal mapping to system resources for optimality. Under this definition of codesign, there is a significant opportunity for model-based codesign of the HW and SW architectures. Such model-based codesign would extend from the experimental apparatus to analysis and storage system and back to the experiment through feedback loops that are actionable as they allow the reconfiguration and optimization of the experiment *in vivo*. A key opportunity that may be enabled by pushing the envelope from the current state of the art is the ability to codesign in a dynamic regime. One ingredient of the codesign methodology is a dynamic runtime system that is model-driven and relies on information gleaned from available monitoring data from the HW path and the SW. In a nutshell, this complex codesign can coordinate HW, SW, and experimental equipment to meet the shifting requirements present during scientific experiments. To illustrate, consider three factors in the case of monitoring degradation of a protein sample in an electron microscopy experiment. In this case, higher electron doses give better imaging fidelity but can lead to sample damage. However, a scientist wants to use as high a dose as possible while incurring minimum damage. To assess damage, one approach would be to leverage simulations to compare observed data with data from an ideal undamaged specimen. The HW settings that impact simulation performance may include sizes of buffers, storage placement, and assignments of threads to cores. Relevant algorithm settings can include time step intervals and spatial resolution. If the workflow involves ML, settings such as model complexity and device placement are also important. Finally, the runtime system also can control performance by providing inputs to the experimental HW, including frame rates and resolution. These factors all impact performance, yet they have complex interactions that imply a dynamic model-based control system that can tune them rapidly is needed to extract the best performance. Notably, performance in this case is not limited to computational criteria, like latency or throughput, but also includes the experimental requirements, such as image quality, sample integrity, and beam stability. Therefore, we posit there is a place for a dynamic codesign to coordinate HW, SW, and experiment settings in real time by reacting to recent performance data and adjusting the various settings appropriately.

The envisioned model-based codesign system potentially can be developed using ML by observing performance data for different algorithmic and system settings. The resulting models then will be used to select actions judged to produce good performance. Importantly, this training can occur online while experiments are being conducted, and actions can take effect immediately, leading to a coupling of the experiment and codesign system.

Timeliness: Success in this area means scientists can perform more informative experiments, and process the results quicker, leading to a faster research cycle. They also will be able to glean insights from experiments at higher spatial and temporal resolution than ever before. This can motivate new theoretical models and generate more accurate quantitative information in important problems. Even small changes in latency can mean huge advantages, enabling rapid iterate-and-test research cycles.

Of course, the general approach of dynamic codesign to jointly control HW, SW, and experiments can be applied to many optimal experimental design settings beyond electron microscopy. Intelligent resource management and new methods of performance prediction and modeling are both important for getting optimal performance out of heterogeneous systems [4]. With next-generation high-throughput detectors already being put into production, development of a dynamic codesign to coordinate HW, SW, and detector is an urgent problem that can lead to immediate benefits. The same frameworks and accelerators that run the scientific ML workloads supplementing the experiment also will be suitable for implementing a model-based codesign. Recent advances in ML approaches, such as deep reinforcement learning and maturing SW frameworks, mean the tools for this problem already may be close at hand.

References

[1] "K3 IS Camera" *Gatan, Inc.* https://www.gatan.com/products/tem-imaging-spectroscopy/k3-camera-0.
[2] "Device Placement Optimization with Reinforcement Learning" *Mirhoseini et al.* Proceedings of the 34th International Conference on Machine Learning (ICML 2017) https://arxiv.org/abs/1706.04972.
[3] "Correctness-Preserving Compression of Datasets and Neural Network Models" *Joseph et al.* Correctness 2020: 4th International Workshop on Software Correctness for HPC Applications (SC21 workshop) https://sc21.supercomputing.org/presentation/?id=ws_corr107&sess=sess205.
[4] Extreme Heterogeneity 2018 - Productive Computational Science in the Era of Extreme Heterogeneity: Report for DOE ASCR Workshop on Extreme Heterogeneity. https://doi.org/10.2172/1473756.

Title: A Data Management codesign of Deep Neural Networks scientific applications on next generation resources

Authors: Ana Gainaru (gainarua@ornl.gov), Scott Klasky (klasky@ornl.gov), Tahsin Kurc (tahsin.kurc@stonybrook.edu), Joel Saltz (joel.saltz@stonybrookmedicine.edu)

Topic: Applications

High performance computing has continued to advance infrastructure with multi-core heterogeneous nodes connected by fast networks with extensive storage capabilities in order to enable the ever-growing resource demands of large-scale applications. However current HPC systems were envisioned to mostly run large scale parallel calculations, and the emerging trends where scientists both augment their applications with machine learning techniques and run large scale analytics for processing experimental and observational data has often been overlooked. The next-generation systems need to be co-designed with these new types of applications to reduce the data management effort within this new class of machine learning applications. One of the co-design goals is to create semi-autonomous computational experiments which can keep up with the unpresented volumes and velocities of data being generated at science facilities by co-designing data management middleware with next-generation hardware.

The scientific community is experiencing unprecedented amounts of data sets in the orders of 1 TB of data per second and within the next 10 years, facilities are already predicting rates of up to 1 PB/s. Due to the speed of data, analysis must be automated by the use of machine learning techniques, such as deep learning methods. There is no surprise that deep neural networks have gained more and more traction in the scientific community since this accumulation of varied and detailed data creates great learning opportunities. This is the case for a wide range of scientific areas: bioinformatics, fusion energy, medical research, computational fluid dynamics, lattice quantum chromodynamics using deep learning networks for a large variety of projects, such as classifying galaxies types [1], analyzing large image datasets to characterize disease structure and function in biomedical research [2], reconstructing the 2D plasma profile [3] and predicting protein structure properties [4].

Development of deep learning models often involves iterative processes and neural architecture searches in which datasets are analyzed multiple times with different deep neural network configurations to find accurate, reliable and efficient deep learning models. In addition, an application may carry out analyses of a dataset with multiple deep learning models in order to detect, classify and predict features and patterns at multiple scales. These analyses may execute a chain of deep learning workflows where data is streamed from one to another or run multiple independent workflows each working on the same input data mapped on different spaces or sliced in different formats. This is the case, for example, for the workloads used by the Stony Brook digital Pathology group where DNN models are trained on whole slide images in research efforts that target cancer biomarker development [7]. These tasks involve complex workflows that can be carried out in several different ways involving multiple DNN branches with classification applications that require different parts of the same input image divided in patches of different shapes and sizes. While several specialized hardware and software have been developed to optimize DNN performance, they are all focused on computation and not data management.

The data management co-design challenge is two folded:

1) On the training side, massive quantities of multi-dimensional training data are needed for DNN models, which increase the accuracy and allow for greater amounts of parallelism. In many cases the same dataset needs to be traversed many times during model selection and training. Hyper parameter optimization and network architecture search can both involve the need to make multiple sweeps through the same training data. The training applications are often composed of many tightly coupled processes that are distributed across the large-scale HPC system and that need to exchange data throughout their execution. Workflow modifications, dataset restructuring and systems software that supports caching of 3D data subsets in quickly accessed memory hierarchy layers is likely to yield major improvements in performance. Because different models may work with different data sizes, partitioning data into smaller batches and moving these across the HPC resources should be done carefully to minimize I/O and

computation overheads. Moreover, predictions in the form of labels and segmentation masks should be collected and written out to disk efficiently to minimize I/O overheads and the overall execution time of the neural architecture search. As new technologies allow for efficient reduction, a careful co-design can be done with these use-cases to move data to the network, reduce data on the network with a given accuracy, and move the data. Furthermore, the speed of the reduction plays a crucial role in this method, so algorithms need to be designed to allow for higher latencies.

2) On the analysis side, the same model is read by all the applications while the access patterns for the input data vary greatly with workflows often passing data in some form between each other and to/from the storage system. Typically, the usage of the input data is done in a non-uniform fashion with some portions being of more importance than others based on predefined parameters well known in advanced or on results generated by other processes in the pipeline. Under such circumstances, and due to the massive amount of data together with the complexity of the application behavior, the fundamental data management abilities required by these applications become highly challenging to implement in a scalable and efficient manner in existing data management solutions. In this case, it is clear that Software Designed Storage can play a central role to reduce, re-organize, and possible refactor the data from storage to allow for it to stream faster to the application. In addition, the repetitive nature of data access patterns for model prediction has great potential to optimize preplanned data movement. A co-design study can allow in-transit computations to prepare the data to the format and accuracy required by the application and to adapt to requirements determined by the network architecture or discovered at runtime.

The data access patterns for both training and analysis can be leveraged by data management systems to fetch in a more efficient way the required input data in the format and order given by the needs of the application and optimize the data path between collaborative processes. For this to happen, the community needs to rethink how to represent and manipulate these scientific datasets at large-scale in a scalable and efficient manner to be able to adapt to the new access patterns introduced by the current and future generation large-scale applications that are starting to heavily rely on machine learning algorithms. Co-designing data management middleware with next-generation hardware is key in achieving this goal. With compute power increasing at a much higher rate than storage or network technology, new hardware technologies are already being developed to speed-up common access patterns over the network. One such example is the Mellanox Bluefield set of switches that enable reduction in-the-network [5] or the Seagate RISC-V system on a chip technology design primarily to optimize IO for edge computing [6]. Fetching the data directly in the required format, adapting the reduction and/or precision of each piece of requested data, staging or streaming data where/when is needed as well as deciding what resources in the software/hardware stack (or what specialized hardware) to use for each IO task need to be reevaluated based on the new patterns introduced by future scientific applications. Understanding the needs of these applications and the benefits and limitations of their interaction with current specialized hardware systems can shape the future hardware and middleware IO landscape.

References

[1] S. Dieleman et al. Rotation invariant convolutional neural networks for galaxy morphology prediction. Monthly notices of the royal astronomical society, 450(2):1441– 1459, 2015
[2] R. Gupta et al. Characterizing Immune Responses in Whole Slide Images of Cancer With Digital Pathology and Pathomics. Curr Pathobiol Rep 8, 133–148 (2020).
[3] D. Ferreira. Applications of Deep Learning to Nuclear Fusion Research. *ArXiv, abs/1811.00333*. 2018
[4] J. Zhou et al. Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. arXiv preprint arXiv:1403.1347, 2014.
[5] B. Williams et al. Exploring Mellanox Bluefield SmartNICs as Accelerators for Heterogeneous Architectures. SemanticScholar, DOI:10.2172/1565824, 2019
[6] Sagar Karandikar, et al. 2020. FirePerf: FPGA-Accelerated Full-System Hardware/Software Performance Profiling and Co-Design. ASPLOS 2020
[7] Kurc T et al. From Whole Slide Tissues to Knowledge: Mapping Sub-cellular Morphology of Cancer. Lecture Notes in Computer Science, vol 11993. Springer, 2019

Title: Algorithm-Architecture Codesign for Irregular and Sparse Problems
Authors: Andreas Gerstlauer, George Biros, Lizy John - The University of Texas at Austin
Topics: architectures; emerging technologies. codesign methods; modeling and simulation

Challenges: With increasing acceleration of computations, memory accesses are rapidly becoming the primary bottleneck even in traditionally compute-bound applications. This is even more the case for irregular and sparse computations that underlie many problems in science and engineering: solutions of PDEs, graph algorithms, and machine learning. There have been tremendous strides in algorithms for such problems. However because sparse computations have irregular data access patterns, they are challenging on both CPUs and GPUs, FPGAs or other accelerators. Recent efforts on co-designing architectures, systems, and algorithms for both dense and sparse computations have focused mostly on specialized computational kernels and less on end-to-end workloads that require composition of such kernels at variable granularity levels. At the full, end-to-end system level, data movement among system components becomes an even greater challenge. Addressing these algorithmic and architecture challenges requires algorithm designers and architects to work closely. However, for them to do so, corresponding tools need to be available. Specifically, there is a need for full-system, end-to-end modeling and simulation tools to enable such codesign research.

Opportunities: As a highlight of the codesign challenges, we report on our experience in [1] in which we studied performance and energy saving benefits of hardware acceleration for N-body methods. Our results showed that unless carefully managed at the software level, acceleration benefits are limited by data movement costs, where far or near memory placement of accelerators makes little difference. Only with careful algorithm and architecture co-optimizations can acceleration benefits be unlocked. But even then, a performance floor set by the data-intensive processing steps around the accelerated kernel remains. Our experiences also showed the limitations of existing simulation tools for such studies [2]. Similar research is needed to generalize co-design approaches and tool development for a broader range of algorithms and architectures specifically targeting data-intensive aspects.

Algorithms: Most DOE applications involve sparse computations. Examples include multiphysics and multiscale partial differential equations, atomistic simulations in materials science, and machine learning methods like construction of nearest neighbor graphs, clustering, graph analytics, and graph neural networks. Many of these applications employ sparse computation kernels like domain decomposition and multigrid methods, sparse tensors, Fast Fourier Transforms, stencil computations, finite-element and finite volume methods, N-body and hierarchical matrix algorithms, and pointwise nonlinear physics kernels. For example, the workflow in [3], involves sorting, projection, sparse and dense factorizations, and data reshuffling kernels in the context of building a hierarchical sparse approximation to a neural network Hessian. A novel computer architecture should enable the efficient composition of such kernels. A key question is, *what technologies will allow us to achieve an ASIC-like performance* for real end-to-end applications? Research is needed to identify core fundamental primitives, e.g., sparse matrix-vector and matrix-matrix multiplication, exact and approximate sparse factorizations, data operations (reduction sort, gather, scatter) that have to be composed efficiently. Furthermore, research is needed for communication avoiding algorithms, again in the context of composing several such kernels to achieve performance. A set of benchmarks, workloads, and metrics need to be established to guide algorithmic and architectural codesign.

Architectures: Heterogeneous, accelerator-rich architectures will integrate a variety of general-purpose and specialized components including CPUs, GPUs, FPGAs and ASIC accelerators. A key question, however, is how such systems should be composed for a certain application domain. What are the right combinations of accelerators and memory hierarchies, and, more importantly, how can data movement between heterogeneous components and memories in the system be optimally orchestrated and supported? The latter is crucial for memory-bound irregular and sparse workloads. To reduce data movement, accelerators can be moved into or near memories. However, this presumes that data is already stored there. If data is produced or consumed by other components, it may be more beneficial to keep accelerators closer to data sources/sinks but provide support for direct dataflow in hardware, e.g. through the caches. Research is needed to identify optimal system architectures including hardware support for computing in-memory (CiM) and efficient orchestration or elimination of data movement (Figure 1). This includes programming and OS abstractions for efficient offloading and management of data and computations. Such architectures will increasingly look like distributed systems within a node, and corresponding algorithms need to be developed to optimally map onto and exploit architecture capabilities.

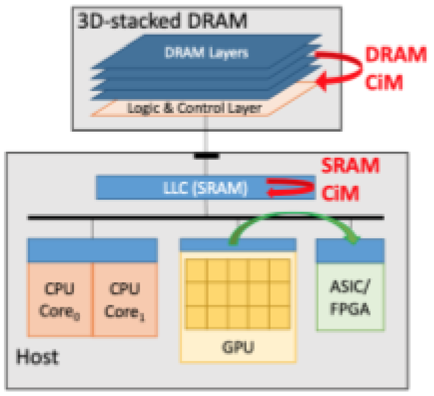


Figure 1: Envisioned architecture for sparse computations.

Modeling and Simulation Tools: Co-design necessitates a common tooling platform for algorithm developers and architects to collaborate. Traditional analytical or simulation-based modeling and simulation (ModSim) tools are too inaccurate or too slow to support end-to-end evaluation of architectural decisions for complete applications. Research is needed to develop full-system ModSim tools that support modular composition of different component models at varying speed/accuracy tradeoffs depending on the use case, including the ability to easily integrate new component models, e.g. CiM simulators as well as new ModSim technologies, e.g. machine learning-based models.

Timeliness or maturity: There are many reasons why this is the right time for this research. There have been many advances on sparse computation algorithms that explore memory hierarchies; new approaches for specific kernels like sparse matrix-vector multiplication; new programming tools like Kokkos [4] that enable performance-portable algorithm development and several main codes are redesigned to support such applications. Similarly, a large number of activities have appeared at the architecture level including new accelerator designs for specific sparse operations as well as new in-memory processing and in-memory circuit designs. Finally, new modular tools for modeling and simulation such as SST that allow assembling full-system simulations with plug-and-play of different component models have appeared. This is the right time to pull together such algorithm, architecture and tool approaches to enable end-to-end co-design research for this domain.

References:

[1] M. Asri, D. Malhotra, J. Wang, G. Biros, L. K. John, A. Gerstlauer, “Hardware Accelerator Integration Tradeoffs for High-Performance Computing: A Case Study of GEMM Acceleration in N-Body Methods,” *IEEE TPDS*, March 2021. https://doi.org/10.1109/TPDS.2021.3056045
[2] M. Asri, A. Pedram, L. K. John, A. Gerstlauer, “Simulator Calibration for Accelerator-Rich Architecture Studies,” *SAMOS*, July 2016. https://doi.org/10.1109/SAMOS.2016.7818335
[3] C. Chao, S. Reiz, C. Yu., H. J. Bungartz, and G. Biros, “Fast Approximation of the Gauss-Newton Hessian Matrix for the Multilayer Perceptron,” *SIAM JMAA*, 42 (1), February 2021. https://doi.org/10.1137/19M129961X
[4] H. C. Edwards, and C.R. Trott, “Kokkos: Enabling performance portability across manycore architectures,” *Extreme Scaling Workshop (XSW)*, 2013. https://www.osti.gov/servlets/purl/1143842

ARIAA: Center for co-design of ARtificial Intelligence focused Architectures and Algorithms

Roberto Gioiosa*
roberto.gioiosa@pnnl.gov
Pacific Northwest National Lab
Richland, Washington, USA

Sivasankaran Rajamanickam
srajama@sandia.gov
Sandia National Laboratories
Albuquerque, New Mexico, USA

Tushar Krishna
tushar@ece.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Topics: Co-design methodologies, architectures, prog. systems

CHALLENGES

artificial intelligence (AI) has the potential to revolutionize the way scientists solve challenging problems and accelerate the rate of scientific discoveries. For example, AI/machine learning (ML) techniques are being used in domains of interest to U.S. Department of Energy (DOE), including power grids and cyber- and national-security applications. As general-purpose architectures (CPUs, GPUs) face challenges due to the trade-off between performance and power consumption, ML for image recognition tasks are beginning to use special-purpose accelerators such as NVIDIA Deep Learning Accelerator (NVDLA), Cerebras, Samba Nova and field-programmable gate array (FPGA)s (Xilinx Versal). Data-flow has emerged as a common paradigm to program many of these accelerators, as it allows for significant reduction in data movement and requires considerably lower energy through the reduction of instruction fetch and decode hardware stages. However, employing ML/deep learning (DL) accelerators in scientific and engineering domains brings additional complexity. Current AI/ML accelerators are designed for dense, regular computations and do not provide efficient support to execute sparse, irregular computations common in ML for DOE applications (e.g., weight pruning, physics-constrained ML), graph analysis, and sparse linear algebra. Also, the DOE software ecosystem needs to be adapted to leverage these heterogeneous resources effectively. Understanding how novel data-flow architectures can be leveraged by DOE workloads and their impact on performance and power efficiency is paramount.

OPPORTUNITY

The Center for co-design of ARtificial Intelligence focused Architectures and Algorithms, ARIAA, is focused on developing data-flow accelerators for sparse, irregular computations that can enable novel, scalable solutions to DOE applications. ARIAA brings together hardware architects, programming model designers, and applications and algorithms experts to explore co-design of (1) novel hardware capabilities, (2) programming models and runtimes, and (3) algorithms and applications. A major focus of ARIAA is ML, including computational kernels and mini-applications for data-flow architectures, and ML applications for cyber-security and electric power grid operations. To achieve full hardware/software co-design, ARIAA leverages and extends several key technologies, including simulators and emulators for data-flow architectures and ModSim tools, runtime and programming models, compilers and languages, and algorithms and motifs. As shown in Figure 1, we believe when combined together, these technologies enable scientific applications to adopt to new architectures.

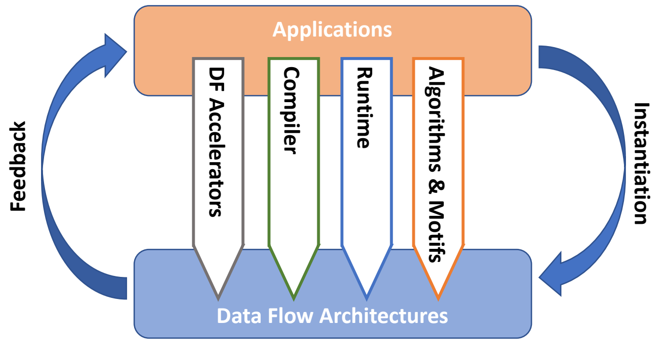


Figure 1: ARIAA co-design technologies

Data-flow architectures and ModSim Tools Data-flow architectures have been studied for many years and some emerging data-flow accelerators are already available to the scientific community. When designing novel data-flow architectures it is important to 1) evaluate the novel hardware concepts in the context of full applications and software stack, 2) understand similarities and differences with current and near-future technologies, and 3) follow an agile process to quickly evaluate new concepts but retain the possibility of deeper evaluation. ARIAA leverages and is developing several technologies to simulate and emulate data-flow accelerators: Structural Simulation Toolkit (SST) [10] is highly-efficient suite of hardware component models designed to work together and permit extension with ARIAA hardware models; MAESTRO is an analytical model to determine the performance and energy-efficiency of deep neural network (DNN) data-flows; Marvel is an optimizer for searching through and recommending optimal mappings of the workload on hardware; STONNE is a cycle-level simulator for flexible data-flow accelerators; FPGAs provide a convenient way to evaluate hardware prototype in a context of a system-on-chip (SoC). Several building blocks for data-flow accelerators have also been developed, including MAERI [6] (a dense GEMM/CONV2D engine), SIGMA [9] (a sparse GEMM engine) and MINT [8] (a sparsity format converter). Additionally, through the DOE Center for Advanced Technology Evaluation (CENATE) [1], ARIAA can access novel data-flow architectures developed for AI workloads.

Compiler Extracting data flow computational graphs from complex applications and re-configuring data-flow accelerators require a sophisticated compiler. ARIAA leverages the COMET compiler for tensor algebra [7]. COMET consists of a Domain-Specific Language

*Corresponding Author

(DSL) for tensor algebra computations, a progressive lowering process to map high-level operations to low-level architectural resources, a series of optimizations performed in the lowering process, and various Intermediate Representation (IR) dialects to represent key concepts, operations, and types at each level of the multi-level IR. COMET is based on the Multi-Level Intermediate Representation (MLIR) framework, which supports the compilation of high-level abstractions and domain-specific constructs and provides a disciplined, extensible compiler pipeline with gradual and partial lowering. COMET extracts data=flows from scientific applications developed with DSL and/or ML frameworks (e.g. TensorFlow) and produces various IRs. At each level or the IR stack, COMET performs different kind of optimizations: high-level, domain-specific optimizations are performed at the highest levels, where operations resemble language-level constructs (e.g., convolutions, tensor contractions) while low-level architecture-specific optimizations are performed at the lowest levels. Currently, COMET can generate binary code for CPU and GPU architectures as well as synthesizable Verilog for FPGA. Work is in progress to produce code for novel architectures, such as Samba Nova and Xilinx Versal architectures.

Parallel Runtime ARIAA enables the execution of complex scientific workflows on extremely heterogeneous systems through Minos Computing Library (MCL) [5]. MCL is a modern task-based, asynchronous programming model for extremely heterogeneous systems. MCL aims at abstracting the low-level hardware details of a system, supporting the execution of complex workflows that consists of multiple, independent applications (e.g., scientific simulation coupled with in-situ analysis or ML frameworks that analyze the results of a physic simulation), and performing efficient and asynchronous execution of computation tasks. In the context of ARIAA MCL enables execution on 1) current accelerators and 2) emulated/simulated novel hardware designs. MCL seamlessly orchestrates the execution of application tasks on CPU and GPU devices, but also on NVDLA and FPGA accelerators. Users need not manually manage data transfer to/from accelerators nor have to port their applications. MCL can also drive the execution of computational tasks on custom hardware designs implemented using hardware simulators or emulators, such as SST or FPGA implementations. Importantly, MCL facilitates execution on emerging data-flow accelerators, such as Samba Nova and Cerebras.

Applications and Motifs ARIAA mainly focuses on four application areas: cyber-security, chemistry, power grid, and graph and optimization kernels. Within these domain areas, several kernels and application motifs have been extracted and mapped to data-flow architectures. Examples includes graph convolutional neural networks, physics-informed neural networks, linear algebra, graph, and optimization kernels. ARIAA scientists developed a hybrid approach to solve the Alternating Current Optimal Power Flow (AC-OPF) problem in powergrid simulation [4]. This approach leverages multi-task neural network to provide better initial conditions for the computational solver used in the back end and achieves up to 3.28x improvement over traditional solvers. ARIAA has also developed a set of mini-applications called Mantevo-DF to evaluate emerging data-flow hardware.

TIMELINE AND MATURITY

As DOE and the scientific community at large embrace novel data-flow architectures, it is paramount to understand how those emerging technologies can be used in scientific applications and evaluate their expected impact before these accelerators become mainstream and difficult to modify. ARIAA seeks to provide informed answers to these new challenges and support the execution of complex scientific workflows on future systems that feature AI accelerators. While the work in ARIAA is still ongoing, several technologies are already available to the scientific community. SST [3] and MAERI/MAESTRO are publicly available and the new hardware designs produced in ARIAA will be progressively uploaded to the respective repositories. The COMET compiler will soon be released as open source and several of the lowering steps are been pushed to the standard MLIR. MCL [2] have been open-sourced and are available to the community while Mantevo-DF will soon be released.

ACKNOWLEDGEMENS

The authors would like to thank the former and current DOE program managers, Dr. Laura Biven and Dr. Steven Lee, for their support. The authors would also like to acknowledge all the project collaborators. Kevin Barker, Gokcen Kestor, Ryan Friese, Mahantesh Halappanavar, Edoardo Aprá, Feng Pan, Burcu Mutlu, Juan Escobedo, Lenny Guo, Rizwan Ashraf (PNNL); William Hart, Clayton Hughes, Matthew Martinez, Jonathan Berry, Simon Hammond, Carl Laird, Cynthia Philips, Mike Eydenberg, Cannada Lewis and Gordon Moon (SNL[1]); Vivek Sarkar, Prasanth Chatarasi, Hyoukjun Kwon, Eric Qin, Geonhwa Jeong, Raveesh Garg (GT).

REFERENCES

[1] 2021. Center for Advanced Technology Evaluation. https://cenate.pnnl.gov.
[2] 2021. Minos Computing Library. http://minos-computing.github.io.
[3] 2021. The Structural Simulation Toolkit. http://sst-simulator.org.
[4] Wenqian Dong, Zhen Xie, Gokcen Kestor, and Dong Li. 2020. Smart-PGSim: Using Neural Network to Accelerate AC-OPF Power Grid Simulation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC '20)*. IEEE Press, Article 63, 15 pages.
[5] Roberto Gioiosa, Burcu O. Mutlu, Seyong Lee, Jeffrey S. Vetter, Giulio Picierro, and Marco Cesati. [n.d.]. The Minos Computing Library: efficient parallel programming for extremely heterogeneous systems. In *Proceedings of the 13th Annual Workshop on General Purpose Processing using Graphics Processing Unit* (New York, NY, USA, 2020-02-23) *(GPGPU '20)*. ACM, 1–10.
[6] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna. 2018. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Programmable Interconnects. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 461–475.
[7] Erdal Mutlu, Ruiqin Tian, Bin Ren, Sriram Krishnamoorthy, Roberto Gioiosa, Jacques Pienaar, and Gokcen Kestor. [n.d.]. COMET: A Domain-Specific Compilation of High-Performance Computational Chemistry. In *Workshop on Languages and Compilers for Parallel Computing (LCPC'20)*. Springer.
[8] E. Qin et al. 2021. Extending Sparse Tensor Accelerators to Support Multiple Compression Formats. In *2021 IEEE International Parallel Distributed Processing Symposium (IPDPS)*.
[9] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna. 2020. SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 58–70.
[10] A.F. Rodrigues, K.S. Hemmert, B.W. Barrett, C. Kersey, R. Oldfield, M. Weston, Cook Risen, R., Rosenfeld J., CooperBalls E. P., and B. Jacob. 2011. The structural simulation toolkit. *SIGMETRICS Performance Evaluation Review* 38 4 (2011), 37–42.

[1]Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

The need for speed: fast feedback for heterogeneous architecture Design Space Exploration

Maya Gokhale (LLNL gokhale2@llnl.gov)

Topics areas: emerging technologies, architectures, modeling and simulation

The reduced impact of Moore's Law and end of Dennard scaling has led to an explosion of special purpose acceleration architectures that gain performance by utilizing narrow, purpose-built circuits tailored to high payoff applications such as deep learning. Examples abound such as the Google Tensor Processing Unit, Intel offerings such as Gaudi, and Cerebras and Samba Nova, to name just a few of interest to DOE. While attention in the commercial sector focuses on acceleration architectures for deep learning and ML workloads, opportunities for incorporating heterogeneous architectures likewise exist in scientific and national security applications. Architectures exhibiting heterogeneity and specialization appear to offer promise to the seemingly conflicting goals of energy reduction and high performance. However, selecting, designing, and evaluating purpose-built acceleration units is particularly challenging in the ASCR space, due to the wide diversity of scientific and national security applications. Even for a particular problem there are multiple approaches to simulating the physics as exemplified by the collection of mini and proxy apps that have been provided to vendors and the research community.

There is a vast design space not only of potential acceleration units of interest but concomitantly of potential insertion points in a computing system. Examples of design choices include specialized processing element arrays; multiple forms of floating point units and their controllers; inclusion, size, and placement of scratchpads; placement of units on-CPU die, near memory, storage, network endpoint, router. In addition to selection, design, and configuration of hardware Intellectual Property (IP) blocks, interaction of cache hierarchy and heterogeneous function units is subtle and complex. The memory/storage hierarchy includes on-chip (cache/scratchpad), on-package, on-node, and remote (network-attached) memories.

Simulating and modeling heterogeneous architectures is actively being conducted by vendors and the research community. While proposed architectures can be transcribed relatively quickly into software simulation modules, conducting meaningful, comprehensive simulations is slow, laborious, and time-consuming. Not only is software simulation slow when capturing the low level cycle accurate behavior of a hardware block, but evaluating that block in context leads to many orders of magnitude increase in turnaround. If the proposed unit is to sit at a non-traditional point such as memory, storage, network, and sensor, it is necessary to model the effects of distributing and partitioning compute among these disparate elements. With the surge of heterogeneous accelerator designs currently being proposed, there is a compelling need for tools that can provide fast turnaround in addition to quantitative, comprehensive evaluation of potential processing units.

As a concrete example, consider a proposal to include a custom floating point unit in an architecture. Questions to arise include: how many units; how are they interconnected; how are they connected to the rest of the compute infrastructure, be it a core, a CPU, cache, scratchpad, network on chip, external memory, interconnection network element or even scientific instrument. These are the factors that ultimately determine the worth with respect to performance, cost, programmability, and overall utility of the proposed new hardware element. Even when the design space is trimmed down through analytic modeling, individual software simulation runs can take hours to days. As an example, a recent experiment in our group to simulate GPU traces captured from a run of XSBENCH (small, default parameters) through AccelSim had to be terminated after two days when the job's allocation expired.

Design space exploration is maddeningly slow, particularly to those of us who have come to expect instant turnaround from our snapchats, instagrams, tweets, and Jupyter notebooks. Unfortunately, there isn't a silver bullet to this problem, but there are a collection of thrusts that can speed up the evaluation cycle.

The first line of inquiry is measurement of potential impact. Probes and instrumentation from a representative collection of algorithms run on existing hardware provide a baseline to gauge potential impact. Next comes analytic modeling to extrapolate parameters and establish bounds. These tools provide a reality check that might result in discarding an idea or greatly modifying a unit's design.

The next step is fast simulation through very high level behavioral simulation. Here it is important to have a simulation infrastructure with a large library of support hardware units. There should be a choice of level of fidelity of a unit, with all presenting a consistent API and simulated timing. For example one module could simulate the whole internals of a floating point unit while another equivalent module could simply do a bit accurate floating point calculation, but they would have the same data format and (model of) latency and pipeline characteristics. It's important to promote open source tools with well defined and copiously documented interfaces, lowering the effort for external contributors to expand library elements. Clang/llvm is an excellent example of such a tool in the compiler community.

Investigation typically starts with evaluation in local compute complex, but has to move quickly to more and more encompassing system level evaluation. This includes expanding scope to accurately model in a realistic environment, i.e. with full software stack including OS and drivers, and out to network. This may be infeasible with software simulation, but is particularly valuable to uncover interface, configuration, and compatibility issues. For example, we discovered a 2X – 5X slowdown between using the same accelerator under Linux compared to "bare metal no OS" mode. Through deeper investigation, we discovered an OS interaction disabling cache when using the accelerator in a Linux environment, whereas cache was enabled in bare metal mode.

For heterogeneous compute unit evaluation at a system level it is particularly advantageous to eliminate the dichotomy of hardware and software by using the same language to describe both. C++-based hardware description languages are well suited to system level evaluation of scientific workloads since most scientific applications are written in C++ or C. This can be accomplished through supporting industry standard hardware simulation languages such as SystemC, which is widely used for ASIC development, and through the use of High Level Synthesis tools that translate suitably annotated C/C++ to hardware pipelines and state machines. Following the iterative refinement path, high level behavioral hardware descriptions can be lowered to RTL in the same language and the same tool framework. In contrast, switching from software behavioral description to a different Hardware Description Language requires development of new testbenches for the hardware version and breaks the ease of switching between equivalent functionality hardware modules that interact with the software application.

In previous work, we have found targeted emulation on FPGA to be very effective for full system emulation, reducing run time from 1000'sX slowdown typical of software simulation to 20X over real time. FPGAs or specialized hardware emulators are typically employed by vendors for CPU emulation. We have found that multiprocessor System on Chip FPGAs are particularly effective in providing cache coherent interface from multi-core ARM CPU to proposed accelerators in programmable logic. For certain types of acceleration units, mapping the software stack to the hard CPU complex allows fast turnaround in the order of seconds to minutes, enabling greatly enlarged design space exploration over either soft processors or software simulation.

For the success of heterogeneous hardware research, fast turnaround of design space exploration is needed. A combination of techniques, including multiple levels of description with consistent interfaces, seamless path to hardware, and targeted use of FPGA for fast emulation, will contribute to realizing this goal.

Relevant tools:
AccelSim, gem5, ZSim, Dynamo Rio, SST, RISC-V related extensions, Logic in Memory Emulator, Open Cores, SystemC

Co-design of System Software for Compute Accelerators and SmartNICs

Ryan E. Grant
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM
regrant@sandia.gov

Scott Levy
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM
sllevy@sandia.gov

Whit Schonbein
Center for Computing Research
Sandia National Laboratories
Albuquerque, NM
wwschon@sandia.gov

I. Topic

Topic: Architectures, Emerging Technologies, Codesign Methodologies

II. Challenge

Current codesign approaches focus on aligning applications, system software and hardware to make the most efficient use of a supercomputer possible. However, the base architectures of the hardware are heavily influenced by markets such as commercial cloud computing. Rarely is there an opportunity to design a system component from the ground up for a new supercomputer. For example, GPUs were not designed with the supercomputing market in mind; they were originally a consumer market product for graphics display. GPUs have also evolved to an important component for large markets such as machine learning/AI and cyptocurrencies. In practice, this limits codesign opportunities because many markets are competing to use the spare silicon area of these devices to accelerate their own particular use cases. As a result, designers of supercomputers for scientific computing are forced to figure out how to exploit existing hardware products and features to get the best possible performance out of scientific and engineering applications..

The one component that has been consistently designed specifically for supercomputers is network hardware. The performance of scientific and engineering applications is undoubtedly bound by the network performance of a system; poor network performance means poor application performance. Fortunately, high-performance network codesign has been a successful endeavour for many years (e.g., Bull's BXI interconnect [1] incorporates Portals 4 [2]). However, there are new challenges facing network codesign for the future. Network codesign is not only about high performance data movement, it is also critical that the network be designed to efficiently support the system software interface for applications. Practically, this means that networks must efficiently support MPI message passing semantics.[1] We have seen several recent efforts to codesign network hardware to improve MPI performance, including explicit hardware support for MPI message matching [3] and collectives [4]. However, major new challenges that will impact the entire approach to codesign are on the horizon.

Codesign efforts have historically focused mainly on application needs, system software, and specific hardware support that are operating within a central control point in the system (i.e. the CPU). Recent developments have led to a new challenge: integrating intelligent networks into supercomputers by introducing general-purpose programmable NICs (i.e., SmartNICs and Data Processing Units (DPUs)). SmartNICs and DPUs are a byproduct of the increase in network performance over the last few decades. The package size of a NIC has expanded to be much larger than the NIC logic requires because the silicon die size is determine by electrical engineering concerns about pin pad sizes and support for very high serializer-deserializer (SERDES) speeds. Therefore, we are presented with a unique codesign challenge and opportunity. Network hardware vendors [5] have begun releasing new NICs that use this extra silicon area to build additional compute resources. Given the emergence of these new devices, we are faced with an important question: *How can we effectively exploit the compute resources on SmartNICs and DPUs to accelerate scientific and engineering applications?*

The novel challenge presented by the codesign of SmartNICs is that they present the rare case where we are not attempting to incorporate existing hardware designed for another primary purpose (e.g., GPUs). Instead, SmartNICs present a blank slate that we have the opportunity to codesign specifically for scientific applications; an opportunity unlike any we have seen since the vector computing era. However, with this great opportunity comes great challenges. We have not had recent opportunities to influence hardware direction. Therefore, our existing approach to codesign must adapt to account for this change. However, it is not yet clear how we can effectively exploit SmartNICs to improve the performance of scientific and engineering applications. For example, should we use them to offload elements of system software from the compute units in the system? Should we use them for intelligent data steering and filtering? Can we use them for AI acceleration? Should application developers design their programs to explicitly target parts of their code for SmartNICs? Will SmartNICs become the new central control point for system software and even hardware in a system, usurping

[1]Support for SHMEM semantics is desirable, but perhaps not required.

the CPU's long held role? Will they have separate operating systems (OS) or will they integrate with the OS on the CPU? Will CPUs even be needed in the future or will SmartNICs attached to GPUs become the new node architecture?

The vast possibilities for SmartNICs are part of the challenge for codesign. As we explore possible roles for the SmartNIC, we will be able to assess what roles it is most capable of performing and what roles are needed most in future generation systems.

III. Opportunity

The challenges facing the codesign of SmartNICs and DPUs present numerous opportunities. For the first time in many years, we have the chance to tackle known problems that supercomputer users have had to learn how to live with. For example, with SmartNICs we can help address some of the problem of performance predictability by offloading known sources of OS/System "noise" from the compute units to a environment that is not executing application code. Significant effort has been devoted to minimizing the impact of OS noise on supercomputers. However, offloading services to a SmartNIC or CPU may further reduce noise. Moreover, moving system software tasks so that they are not competing for resources with the application code may provide opportunities for complex, run-time optimization of system software tasks where it was not previously possible due to the cost of computing the optimizations.

SmartNICs are the ideal platform to use to optimize data movement off of the node by scheduling movement to and from the I/O subsystems and other nodes during intervals when the network is not needed for latency-sensitive applications communication. For example, SmartNICs provide the opportunity to intelligently schedule and independently perform application checkpoints to burst buffers and the parallel file system. The application merely needs to note the location of the checkpoint data in memory and when it can be copied. The SmartNIC can then copy the data locally and move it out on the network when it will not interfere with application data traffic. SmartNICs also offer the possibility of seamlessly coordinating data movement on heterogeneous nodes (e.g., to and from GPU memory) without perturbing application code executing on the host CPUs.

SmartNICs also provide opportunities to integrate first-of-a-kind hardware into supercomputing systems. For example. SmartNICs could easily be designed with Tensor Processing Units (TPUs), or custom matrix-multiply units to enhance application performance or provide high performance ML/AI compute capabilities to the system as a whole. SmartNICs offer the possibility of a new kind of distributed system software that can be augmented with AI capabilities to offer a truly distributed, globally-coordinated system management ecosystem. This capability would provide the opportunity for entirely new types of distributed system software that were not previously possible or practical to be developed. For example, a distributed system software solution for network scheduling could be developed that would help applications that were completely unaware it existed. By coordinating with the system resource manager/scheduler, applications can be run with their communication schedules in mind to avoid network congestion based on previously observed behavior. An online distributed system running on SmartNICs could use AI to predict future traffic during application execution and adapt network settings on-the-fly to optimize performance. PVFS systems can be speculatively primed to deliver data to intermediate buffers in anticipation of the application's needs. The opportunities for new approaches to system software design in this space are vast and intriguing.

IV. Timeliness

This work is ideally timed as industry is undergoing efforts to find the best uses for SmartNICs and infrastructure support is minimal. As such the hardware is at a prime point to be influenced as are opportunities to influence system software and the interfaces for such hardware. Common interface efforts have been started [6] but as the architectures are still widely varied and in flux, a standard interface still needs guidance as to what role the SmartNIC will play in future supercomputers before it can be used to unify the interface in a useful manner.

Acknowledgment

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] S. Derradji, T. Palfer-Sollier, J.-P. Panziera, A. Poudes, and F. W. Atos, "The BXI interconnect architecture," in *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*. IEEE, 2015, pp. 18–25.

[2] B. W. Barrett, R. Brightwell, S. Hemmert, K. Pedretti, K. Wheeler, K. D. Underwood, R. Reisen, A. B. Maccabe, and T. Hudson, "The Portals 4.0 network programming interface," *Sandia National Laboratories*, 2012.

[3] W. P. Marts, M. G. Dosanjh, W. Schonbein, R. E. Grant, and P. G. Bridges, "MPI tag matching performance on ConnectX and ARM," in *Proceedings of the 26th European MPI Users' Group Meeting*, 2019, pp. 1–10.

[4] R. L. Graham, D. Bureddy, P. Lui, H. Rosenstock, G. Shainer, G. Bloch, D. Goldenerg, M. Dubman, S. Kotchubievsky, V. Koushnir *et al.*, "Scalable hierarchical aggregation protocol (SHArP): A hardware architecture for efficient data reduction," in *2016 First International Workshop on Communication Optimizations in HPC (COMHPC)*. IEEE, 2016, pp. 1–10.

[5] Mellanox. (2018) Mellanox BlueField SmartNIC. [Online]. Available: https://www.mellanox.com/products/bluefield-overview

[6] B. K. Williams, W. K. Poole, and S. W. Poole, "OpenSNAPI: Toward a unified API for SmartNICs," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2020.

Edge computing at synchrotrons: co-design for data reduction, monitoring, and real-time analysis

Marcus Hanwell[1], Line Pouchard[2], Yihui Ren[2], Wei Xu[2], Anthony DeGennaro[2] and Christine Sweeney[3]

1: NSLS-II, Brookhaven National Laboratory 2: CSI, Brookhaven National Laboratory
3: Los Alamos National Laboratory

{mhanwell, pouchard, yren, xuw, adegennaro}@bnl.gov and cahrens@lanl.gov

Topics: Applications, Codesign methodologies

Challenge

Synchrotrons, such as the NSLS-II, are large user facilities, funded at the national level, to serve the scientific community with some parallels to supercomputing facilities. They offer extremely bright beams of photons impossible to produce at smaller facilities. These national user facilities enable scientific discoveries from designing new quantum materials to revealing protein structures. Beamlines are positioned around the electron storage ring and use the high intensity beam at specialized end stations. These beamlines are equipped with different types of sample mounts and a variety of sensors that range from diffraction, imaging, and computed tomography increasingly with dynamic measurements. These sensors are improved with each iteration, producing higher resolutions often at greater acquisition rates that couple with accelerator upgrades that enhance beam brightness, size and other characteristics. This represents a huge downstream data and computation challenge for synchrotron facilities. It is estimated that data rates will be on the order of exabytes in the next 10 years!

Thanks to brighter light sources, improved sensors and heightened user interest, it is increasingly important that facilities operate around the clock. Recent events around the global pandemic have highlighted the growing need for greater automation at user facilities and increased the urgency of doing so. Particularly, beyond real-time remote monitoring, we would like to improve automatic sample movement and sensor adjustment by utilizing robotic arms in increasingly difficult experimental arrangements. As robotic arms operate, samples must be tracked, and all relevant information, such as beam position, sample-stage alignment and anomalies (e.g., possible collisions), must be recorded. As multi-modal experiments, where the same sample is moved and analyzed across multiple beamlines, become more important, such autonomous sample moving, tracking and registration will be a key goal for national user facilities.

As the challenges of increased data rates and fully autonomous operation become more pronounced—to enable better science at national user facilities [1, 2]—it is critical that we seize this co-design opportunity to tackle these challenges.

Opportunity

We have identified three main challenges at our synchrotron: 1) high data rate that requires real-time compression and online analysis; 2) remote experiments monitor and autonomous control; 3) a holistic and modularized system design and integration of hardware, software and users. Although these challenges are motivated by day-to-day operations at our synchrotron, we believe they are ubiquitous to other facilities. We envision the co-design has at least three levels from top to bottom: 1) modularized abstraction and API design to boost reusability and portability, 2) integration of low-latency artificial intelligence (AI) solutions at the edge close to detectors, and 3) selection and programming suitable compute devices.

Abstractions—Software-Hardware Interfaces

Co-design at the edge for experiments at user facilities requires a common understanding between experimental scientists and electrical engineers in addition to the traditional teams of computational scientists, computer scientists and computational hardware designers. Well-designed abstractions allow edge device, sensor, network, storage and compute hardware designers to understand needed capabilities. These abstractions may involve one or more middleware layers that are intermediate (usually) between Python or facility control languages (e.g., EPICS) and lower-level languages or libraries.

Some important abstractions include those for robotics, metrics and workflow. Interfaces can generalize robotic capabilities (movements, goals, constraints) across devices that may differ between experimental setups locally or across facilities. Cross-cutting definitions of metrics can be applied to data collection and quality monitoring (noise characteristics, data throughput and latency, ranges of data values, data value variance, timing) that can be used at the edge, online or for offline replay. Workflow, processing and/or analytics pipeline abstractions can allow for ordering operations to help co-design in optimizing or parallelizing sequences or groups of tasks.

Experiment Visualization and Inspection Co-design

On one hand, experiment data that are usually heterogeneous and multidimensional introduce the visualization and evaluation challenge that demands the co-design with users. A visualization framework with user interactions presents potential opportunities including: 1) the pre-screening of those data before delivering to the followup analysis; 2) the evaluation of the results generated by the adopted analysis modules; and 3) the understanding of the analysis modules themselves, especially the deep learning approaches.

On the other hand, the front-end visualization and back-end imaging framework, once combined, can enable coupled analysis, processing, display and visualization acceleration on various hardware, such as GPUs/TPUs. This presents another co-design opportunity with hardware that could significantly improve interactive and real-time interaction as data are acquired.

Low-latency AI Inference on New Hardware

Deep learning approaches have been very effective for complex pattern recognition tasks, such as image classification, pedestrian tracking, object detection and instance segmentation in computer vision. Scientific communities are embracing AI solutions in many aspects, from replacing expensive simulations with neural network surrogate models to segmenting electron microscope images. Most of these achievements have been reached in an offline fashion, either on a powerful workstation equipped with GPU cards or sometimes on high-performance computing (HPC) systems. Integrating a trained model with experimental facilities and impacting the way scientists conduct experiments are the natural next steps. The computation in neural network inference differs from that in training. Unlike neural network training, which consists of both forward phase (to compute activations) and backward phase (to compute gradients and update the model), neural network inferencing only has the forward phase and can operate in a streaming fashion. Therefore, a co-design problem emerges. On the hardware level, besides GPU-like devices (streaming compute unites with small local cache but vast global memory) such as AGX, newer hardware designs invoke non-von Neumann architecture, such as GraphCore IPU, Cerebras CS-1, SambaNova, Xilinx AI-Core, Intel Loihi, etc. How to pick the right inference hardware for specific scientific applications and design a neural network that works well on the specific hardware and meets the bandwidth or latency requirement is a co-design opportunity.

Specific to the challenges in our synchrotron applications, can we design and fit traditional algorithms on new AI-motivated hardware, so we can repurpose the same hardware at different stages of experiments, for example, data reduction and compression during the experiment and AI-driven collision detection and sample tracking during the sample staging period by robotic arms? Will industrial edge devices with the design goals of wireless connectivity and low power consumption fit into a scientific environment where devices are exposed to strong radiation and energy consumption is not a concern? Should the scientific community define their own requirements for ideal "edge" devices?

Timeliness

The light sources are brighter, and the sensors used are faster with greater pixel densities, creating huge challenges as the data rates produced by these facilities increase. Existing solutions are showing their flaws, and the area would benefit greatly from increased collaboration using co-design to develop specialized solutions to improve capabilities at these national user facilities.

References

[1] John Hill et al. "Future trends in synchrotron science at NSLS-II". In: *Journal of Physics: Condensed Matter* 32.37 (June 2020), p. 374008. DOI: 10.1088/1361-648x/ab7b19.

[2] Nicholas Schwarz et al. "Enabling Scientific Discovery at Next-Generation Light Sources with Advanced AI and HPC". In: *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*. Ed. by Jeffrey Nichols et al. Cham: Springer International Publishing, 2020, pp. 145–156. ISBN: 978-3-030-63393-6.

Title: A Codesign Approach to Addressing End-to-End Security, Safety, and Integrity Within Systems of Heterogenous Chiplets

Authors & Affiliations.

Name	Affiliation	Email
Bill Harrison	Cyber Security Research Group/NSSD	harrisonwl@ornl.gov
Chris Hathhorn	Cyber Security Research Group/NSSD	hathhorncr@ornl.gov
Michela Becchi	ECE Dept./North Carolina State University	mbecchi@ncsu.edu
David Andrews	CSCE Dept./University of Arkansas	dandrews@uark.edu

Topics. Programming Systems; Codesign Methodologies.

Challenge: Combine Chiplet Configurability with High Levels of Trust

Intellectual Property (IP) reuse has emerged as a way to reduce device development's time and cost. With the slowdown of Moore's law, building systems by integrating smaller hardware functions can be cheaper and more viable than manufacturing complex systems in a single chip. This idea is at the basis of the "chiplets" technology. Chiplets are modular chips that can be assembled in a package and connected using a die-to-die interconnect. Recently Intel has developed an open-source, royalty-free standard called the Advanced Interface Bus (AIB) for connecting multiple semiconductor dies within a single package (Fig. 1, left). The AIB specification has been adopted by DARPA's Common Heterogeneous Integration and IP Reuse Strategies (CHIPS) program. Chiplets and the AIB interconnect technology are also used in the Intel Stratix 10 and Agilex FPGA families, where the AIB interface is used to couple an FPGA die to multiple transceiver and HBM2 stacked-DRAM tiles.

While offering significant opportunities, integrating 3^{rd}-party chips in a single system creates security risks, calling for validation and information assurance measures. We propose leveraging FPGA fabric to add separation kernel-like and other assurance guarantees to chiplets-based Systems in Packages (SiPs), and deploying our solution on an Intel chiplet-based device (Fig. 1, center). Our framework will allow adding security domains with specific information flow constraints to an existing system, and including the existing chiplets to them.

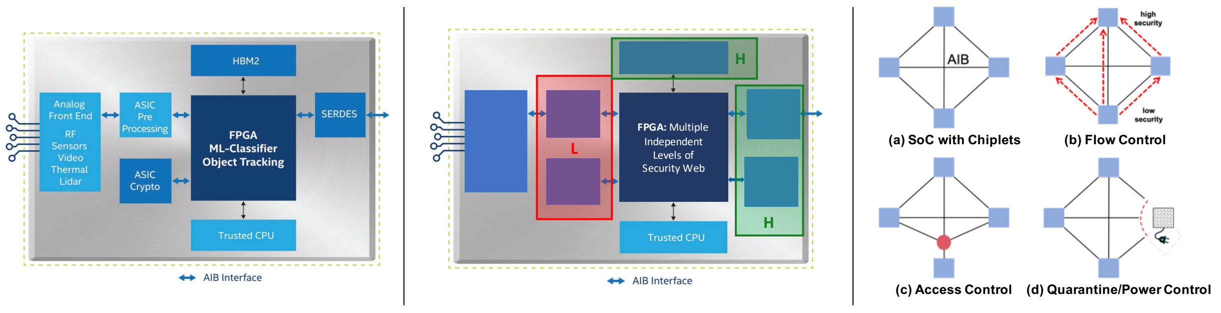


Figure 1: (Left) Intel's Advanced Interface Bus (AIB) [7]. (Center) Spider can express Separation Kernel-like Assurance Guarantees to Intel's Chiplet Ecosystem. (Right) Flexible Security Policy Enforcement for (a) Chiplet-based SoCs with policies including (b) information flow control, (c) access control/monitoring, and (d) quarantining of faulty/malicious chiplets.

Opportunity: Functional Language HLS for Configurable, Verifiable Chiplet Security.

High-level synthesis can support *both* software process levels of configurability and high assurance capabilities [4, 6, 8, 5], and the toolchain we envision exploits this duality to full advantage. For this reason, we propose an HLS language, Spider, for programming security envelopes to manage Chiplet applications. Spider will be an HLS language whose programs ("spider webs") enforce security properties through the configuration of the AIB interconnect technology. Furthermore, Spider language design will support high assurance through the re-purposing of tools and techniques from software formal methods to FPGA fabric—specifically those based in functional programming languages like Haskell, Coq, and Agda. We envision a reconfigurable system where security constraints can be configured using a high level programming interface, which can be also used to perform formal reasoning on the system's properties.

AIB interconnect technology can support configuration of security policies (e.g., flow and access controls) within reconfigurable FPGA fabric, although there are currently no high-level programming abstractions for defining, implementing, and enforcing such security policies. The goal or the proposed research is to design and implement a security policy language to simplify and accelerate the development of flexible, effective, and correct security envelopes for the Chiplet ecosystem. Powerful policy language abstractions can take full advantage of AIB configurability to lower the development cost of security infrastructure while increasing its capabilities. We will call this security policy language *Spider* and refer to the security envelopes it produces as *webs*.

We envision Spider as analogous to the P4 network programming DSL, although webs will be implemented within the AIB interconnect rather than on network devices. The P4 domain-specific language is for programming the controlling packet forwarding planes of networking devices (e.g., routers and switches). P4 is implementation- and protocol-independent, meaning that it abstracts aways from details of particular target architectures and protocols to provide a uniform approach to programming network devices. A programming language-based approach can yield both software engineering benefits (e.g., comprehensibility, modularity, and ease of development/refactoring) and high assurance by leveraging of software verification techniques [9, 2]. The proposed research combines P4 like programmability for the AIB interconnect as a functional high-level synthesis language.

Multiple Independent Levels of Security (MILS) systems [1] (sometimes called *separation kernels*) are a US Department of Defense architecture for securing DOD mission-critical embedded systems amid increasing levels of resource sharing. MILS kernels enforce multiple security "domains" so that all communication/interaction between processes at different security levels obeys the prescribed security discipline. Major vendors provide MILS-like security enforcement in hardware [3]; these systems include Apple's Secure Enclave Processor (SEP), ARM TrustZone technology, and Intel's Trusted Execution Technology (TXT). These vendor technologies all focus on security enforcement at the hardware-software boundary to support cryptographic algorithms, access control to memory subsystems, and the integrity of system software. By contrast, the proposed research has an analogous, albeit purely hardware, goal of enforcing verified MILS-like security for the Chiplet ecosystem. Fig. 1 (right) illustrates the kinds of security capabilities this research will provide to the Chiplets ecosystem. These include MILS-style ("no-write-down, no-read-up") flow policies (Fig. 1 (right, b)) in which the permissible flows are programmed directly into the AIB interconnect. Access control policies (Fig. 1 (right, c)) such as conventional memory protection capabilities are possible as are memory scrubbing and dynamically set bandwidth limitations. Isolation primitives (Fig. 1 (right, d)) will support graceful degradation of functionality in the presence of faulty or malicious chiplets. These primitive include power-control capabilities to either conserve energy and/or turn-off malicious/faulty chiplets.

Timeliness/Maturity.

There is no such thing as a secure system without secure hardware. To produce trustworthy hardware/software systems, it is our position that security must be a first-class concern throughout the entire codesign process. Our position is that recent advances [4, 6, 8, 5, 9, 2] in applying functional languages to high assurance hardware development enables codesign of systems with whole-system security guarantees and that exploiting these advances will revolutionize the trustworthiness of codesigned systems over the next decade.

References.

[1] J. Alves-Foss, P. Oman, C. Taylor, and S. Harrison. The MILS architecture for high-assurance embedded systems. *Int. Jour. of Emb. Sys.*, 2:239–247, 09 2006.

[2] R. Doenges, M. Arashloo, S. Bautista, A. Chang, N. Ni, S. Parkinson, R. Peterson, A. Solko-Breslin, A. Xu, and N. Foster. Petr4: Formal foundations for p4 data planes. *Proc. ACM Program. Lang.*, 5(POPL), January 2021.

[3] A. Ehret, K. Gettings, B. R. Jordan, and M. A. Kinsy. A survey on hardware security techniques targeting low-power SoC designs. In *High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2019.

[4] W. Harrison and G. Allwein. Verifiable security templates for hardware. In *Proceedings of the Design, Automation, and Test Europe (DATE) Conference*, 2020.

[5] W. Harrison and C. Hathhorn. Expanding the high-level synthesis trust zone. ORNL LDRD Program, 2020.

[6] W. Harrison, C. Hathhorn, and G. Allwein. High Assurance is FIRRTL Ground. In *Interactive Theorem Proving (ITP) Conference (submitted)*, 2021.

[7] D. Kehlet. Accelerating innovation through a standard chiplet interface: The advanced interface bus (aib). Technical report, Intel Corporation, 2020.

[8] A. Procter, W. Harrison, I. Graves, M. Becchi, and G. Allwein. A principled approach to secure multi-core processor design with ReWire. *ACM TECS*, 16(2):33:1–33:25, January 2017.

[9] C. Skalka, J. Ring, D. Darias, M. Kwon, S. Gupta, K. Diller, S. Smolka, and N. Foster. Proof-carrying network code. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1115–1129, New York, NY, USA, 2019. Association for Computing Machinery.

Title: User-driven Co-design Tools for full System-level Deployments
Authors: **Oscar Hernandez**, Matt Baker (Oak Ridge National Laboratory), Stephen Poole (Los Alamos National Laboratory), Kiran Ravikumar, P. K Yeung, Jeffrey Young (Georgia Tech)
Topic: applications, modsim, programming systems, codesign methodologies, architectures

As we approach the first exascale computer deployments, it is imperative that we reexamine the concepts behind co-design for the next generation of systems. Codesign for the last decade has focused on an effort to “redesign” applications to conform to a predetermined path in hardware and systems designs which are almost solely focused on floating point performance, power consumption and costs. Applications and their figure of merits (FOM) are defined at procurement time, which may not reflect the latest changes on the application and the programming environment, the new algorithms developed, and challenges encountered by domain scientists on new emerging workloads like machine learning or converged HPC+AI applications. The process of co-design lacks user interactivity and doesn’t have the right tools and methodologies to best communicate application requirements to hardware manufacturers. Most of the work done by the vendors focuses on predicting performance of applications on a new system, but domain scientists often don’t understand the methodology used to verify these projections.

We propose that co-design efforts should support tools that include a user-driven and code-focused methodology. Application developers and domain scientists should play an active role in co-design via a set of tools that allows them to transparently do design space exploration for upcoming systems with a focus on overlooked features like data movement and network communication. This proposed set of tools would require characterizing and capturing application requirements that can be passed to system software and hardware designers, and it would also necessitate multi-precision or multi-modal simulation techniques to allow for simulating and tweaking future architectures to match application characteristics.

Challenge: As an example of the current crisis for co-design, the GESTS application[2], is a turbulence code that has been ported to multiple supercomputers including ORNL’s Titan and Summit with good performance and in the near future to Frontier. GESTS is a code that makes extensive use of large 3D FFTs that require transpose operations and all-to-all communications between processors. While the core FFT kernel is well-represented by existing mini-apps and benchmarks, the usage of these kernels at scale causes complex issues when porting to new architectures. Their choice of algorithm depends on the number of accelerators per node, the amount of memory available on the devices, the compute vs network bandwidth availability, and asynchronous executions strategies to hide data movement costs. Early access systems might include a relatively close approximation of system accelerators, memory, and FLOPs but they have no way to represent cluster-scale networking, data movement requirements, and application mappings across the near and far communication domains of a large supercomputer.

Existing profiling tools like NVProf, Vampir, and Tau are designed to help application developers tune their applications to a current set of hardware. They were not designed to guide the developer to focus their code-design efforts other than telling them where the bottlenecks are on an existing system. This limits the design space exploration options for architects and limits the ability of domain scientists to improve applications or to determine which regions of code are important to co-design for other platforms. Tools like NVProf are also vendor specific and do not capture any porting effort knowledge. Also, if a site changes vendors, the old tools are no longer useful, which limits continuity of tool knowledge and of techniques used in the porting experience itself. Currently, there is limited guidance for these experts to decide how to algorithmically change their code to obtain the best performance on a future system. Future co-design tools must be able to handle applications *at system-level scale*, which requires a much different approach to the design of related simulation and emulation tools to guide users of their regions of interests and the design exploration space for system software and hardware.

Opportunity: One approach could be to use trace-driven or application-driven simulators (as with gem5, MUSA, or DynamoRio tools) for specific components like memory or CPUs while focusing on more accurate modeling of key system-level components such as the network or accelerator. While these approaches can work well for well-known workloads and current-generation hardware models, they are currently limited in terms of mixing models with different fidelities (e.g., combining an ML cache model with a detailed CPU). One potential opportunity is to better target these tools towards application users and to provide user-driven feedback to target specific code regions or specific architectural features of interest. Other features could then be replaced with a lower-fidelity model.

Definitions of these code regions of interest (ROI) are currently limited by the complexity and speed of finding significant ROIs and relating them to common figures of merit (FOM) like time to solution, simulation cycles/real-world seconds, and data movement (B/FLOP). A second huge opportunity is the use of newly developed compiler and runtime tools that can improve the characterization process for these ROI, detect code patterns and develop a classification system to allow for user-based scoping of ROI, and help map these ROI to scalable system-level simulations.

Why now? It takes 10 years to do a true application-centric co-design effort, as is evidenced by RIKEN's efforts with the development of Fugaku. We see recent improvements in compilers, runtime analysis tools, and simulation that enable us to fully address our ideal set of tools that 1) identify patterns that contribute to specific application FOM, 2) allow for mixed-fidelity simulation, and 3) incorporate learning and user feedback. Compiler analysis tools built on LLVM now allow for similarity analysis and classification of complex code patterns [1] and machine learning techniques to classify similar code regions [4]. At the same time, runtime analysis tools like DynamoRio and the OpenMP and Kokkos Tools API allow users to design simple tools to analyze code at runtime and to provide more robust information about ROI for an entire application. Finally, we are starting to see improved efforts for mixed-fidelity simulation using full application stacks with simulators like MUSA or gem5 with ROI-based techniques like BarrierPoint [3].

While these new techniques and tools allow for improved co-design, we suggest that further work is needed to create a comprehensive system-level characterization and simulation framework that can target challenging application patterns that are not addressed by today's co-design and application porting efforts.

References:

[1] Wei Ding, Oscar Hernandez, and Barbara Chapman. "A Similarity-Based Analysis Tool for Porting OpenMP Applications." *Facing the Multicore-Challenge III: ANPTPC, 2013.*
[2] Kiran Ravikumar, David Appelhans, P.K. Yeung, "GPU acceleration of extreme scale pseudo-spectral simulations of turbulence using asynchronism", *SC 2019.*
[3] M. Tairum Cruz, S. Bischoff and Roxana Rusitoru, "Shifting the Barrier: Extending the Boundaries of the BarrierPoint Methodology," *ISPASS 2018.*
[4] Fangke Ye, et al. "MISIM: An End-to-End Neural Code Similarity System." *arXiv preprint arXiv:2006.05265, 2020.*
2018.

Real-time streaming data acquisition and analysis for Nuclear Physics experiments

Graham Heyes and Amber Boehnlein
Thomas Jefferson National Accelerator Facility
12000 Jefferson Ave, Suite # 10, Newport News, VA 23606

Topic: architectures, applications

1. CHALLENGE

With very few exceptions data acquisition systems used for nuclear physics experiments are based on assumptions and architectures that have changed very little over the last quarter century. The primary assumption is that the raw data rate from the detector is too large to acquire without reduction at source. This is achieved by using custom electronics and firmware to make real-time decisions on which data contains interesting physics. This trigger is fed back to the digitizing electronics and selected data is read out and passed to the data acquisition chain. Typically, an "event builder", implemented in hardware or software, merges data belonging to the same trigger to form an "event" which contains all the data associated with the physics interaction that the trigger chose. The output of the process is a time ordered sequence of events, that contain the raw output of the digitizers, typically written to files. This impacts the workflow by which the data is processed to generate physics results. There are several challenges:

- The trigger throws away data and may bias an experiment.
- The trigger is implemented in hardware, is inflexible and hard to debug.
- The event builder is an intrinsic bottleneck, complex, and a point of failure.
- Data generation is driven by the trigger, which has random timing.
- The output of the data acquisition is raw digitized signals from the electronics. It must be converted to physical values such as positions, times, energies, in a software process called reconstruction.
- Reconstruction requires detector calibration which requires running the detector under well understood conditions and comparing the data taken with simulations.

An example is the CLAS12 detector at Jlab. This detector is read out by approximately a hundred VXS chassis of electronics [1]. The trigger is implemented in custom built hardware using the serial lanes of the VXS backplane and optical links to transfer trigger data to the decision-making global trigger hardware [2]. Data from all 100+ data sources is merged in real-time in software at data rates of around 1 GByte/s and 90kHz event rate. The output is a series of data files with approximately one file generated every minute. The experiment runs 2/3 of the year with a daily uptime of 60-80%. This generates thousands of files of data each of which must be run through reconstruction using a detector calibration that was valid when the data was taken. Calibration drifts over time and special calibration data taking runs must be repeated several times each day. Converting this data into an accurate calibration is a time-consuming process, often reconstruction can only start weeks or months after the data is taken.

2. OPPORTUNITY AND PROGRESS:

The previous section began with the statement that data acquisition architecture and analysis workflow are driven by assumptions that have not changed in a long time. It was impossible to acquire all the data without a trigger to cut down the rate, even if it could all be acquired the dataset would be too big to store, even if it could be stored the computational cost of the reconstruction step would be prohibitive. In parallel several groups at different laboratories are independently realizing that these assumptions are no longer true [3]. The key factor that breaks these assumptions is rapid technology advance in FPGAs, optical network links, memory devices, and computing over the past years. It is now possible to use FPGAs to read out detectors with minimal filtering to remove background noise but without resorting to a physics-based trigger. The data can then be streamed in parallel from the various component detectors over optical links into memory. The data flow will be deterministic since the detector will be read at fixed rate rather than on a random trigger. Given adequate fast memory to temporarily buffer the data, there is no need for a complex real-time event builder. These last two factors eliminate many of the failure modes faced by triggered systems. These streaming readout systems are expected to be the dominant architecture in the future.

A streaming system makes no physics-based trigger cut on the data. The data is clean and unbiased allowing the possibility of calibration and reconstruction in real time. To reduce computation and storage cost it is likely that, post acquisition if not post reconstruction, the experimenter would opt to implement a software trigger that would reject uninteresting data. A software trigger would be more flexible than one implemented in custom electronics and firmware. A streaming system acquires time sequenced data from all parts of the detector in parallel resulting in a multi-dimensional dataset, with one dimension being time. It is possible to implement software that resolves interactions that occurred very close together in time that a traditional trigger could not resolve. There is also the possibility of novel data processing techniques that are not possible with the files of linear sequences of events output by a traditional NP data acquisition. An example is use of AI/ML to infer the physics either at the event level, by recognizing events, or at a higher level by recognizing hit patterns. In the former case we skip the calibration and reconstruction step by using pattern recognition to recognize particle tracks through the detector and infer the parameterization of the track. In the latter case we skip most of the traditional NP workflow, calibration, reconstruction, and statistical analysis, and infer the physics directly from patterns in the raw data.

To realize the full potential of streaming detector readout we require a heterogeneous and flexible computing environment. We need to integrate a data source using a mix of software and firmware into a high-performance data processing system. This system must provide an environment that can evolve as new techniques and technologies appear. For example, if we follow the path of using AI/ML to infer physics it may be appropriate to make use of hardware inference accelerators. Also, existing trigger systems have led to the development of firmware that very efficiently processes NP data. This could be taken advantage of by making FPGAs available in the data processing systems. Currently NP makes minimal use of GPUs but this is changing, not only through user written code but also use of GPU enabled AI/ML tools and other packages.

If successful, the effort to optimize compute for a streaming NP data source will greatly accelerate the workflow between running an experiment and publishing a result. It will also tighten the loop between a development in theory and testing it with experiment. The use of a streaming readout is critical to experiments planned at JLab as well as to the joint BNL-JLab EIC project [4], where a traditional trigger system would be expensive to implement and possibly compromise the science results.

3. References

[1] R. B. e. a. Boyarinov S, "The CLAS12 Data Acquisition System," *Nucl.Instrum.Meth.A 966 (2020) 163698.*

[2] B. Raydo, S. Boyarinov and et al, "The CLAS12 Trigger System," *Nucl.Instrum.Meth.A 960 (2020) 163529.*

[3] "Streaming Readout VI," JLab, 2020. [Online]. Available: https://indico.jlab.org/event/378/.

[4] National Academies of Sciences, "An Assessment of U.S.-Based Electron-Ion Collider Science.," The National Academies Press., [Online]. Available: https://doi.org/10.17226/25171.

Co-Design of Streaming Workflows to Drive Experiments-in-the-Loop

Seth Hitefield*, Thomas Naughton, Bogdan Vacaliuc, Nageswara S. Rao, Neena Imam, Teja Kuruganti
Oak Ridge National Laboratory
{hitefieldsd,naughtont,vacaliucb,raons,imamn,kurugantipv}@ornl.gov

Topics: Applications, Programming systems, Emerging technologies

I. Challenge

Many state of the art scientific and engineering workflows follow the familiar pattern of batch computation for running codes and processing data. The sheer amount of data combined with complex computations necessitate large, centralized supercomputers and other high-performance-computing (HPC) systems at remote or cloud facilities. This requires a multi-stage approach of collecting all data for a given experiment or application, transferring it to compute facilities, executing the analysis codes, and interpreting results. Traditional science workflows at Department of Energy (DOE) experimental facilities such as the Spallation Neutron Source (SNS) require co-designs that effectively and collaboratively utilize computing facilities like Oak Ridge Leadership Computing Facility (OLCF) [1]. Other DOE scenarios, including modernized smart grids, must also process significant amounts of sensor data on large scale systems.

This batch processing paradigm limits the overall productivity of users due to the constraints of remote compute and instrument resources and is inefficient in both time and cost. For example, users at the SNS must estimate duration of data collection, hoping it is sufficient to produce adequate results. This is inefficient in two ways: either 1) too little data is collected requiring users to later re-run the experiment, or 2) too much data is acquired that consumes expensive instrument time that could be used for finer granularity or additional sample environment collections. In another example, monitoring the effect on the smart grid of distributed power generation from consumer-owned systems, users are limited to post-processing and can only derive conclusions for historical events; there is no ability to enable real-time analytics that drive active control decisions for power generation services to improve grid stability. The objective of our approach is to re-imagine batch-based processing into streaming-based systems and develop frameworks that enable persistent, high-throughput, and low-latency scientific and engineering workflows.

II. Opportunity

The rise of edge computing drives a fundamental shift in thinking within the scientific community towards deploying computing capabilities at the instrument edge rather than only centrally located facilities, reducing the gap between the data and the code [2], [3]. Connecting instruments, edge computing, and HPC facilities into larger science federations will be a key approach that enables the future of cutting edge scientific discovery and will result in more efficient experiments. These federations are possible due to improvements in embedded and edge hardware and as high-bandwidth connections such as 5G and fiber become ubiquitous [4]. They will also support brand new capabilities such as streaming workflows that will lead to smarter experiments with real-time control and inference, enabling the discovery of new paradigms to derive understanding from data[5]. This shift to streaming closes the feedback loop for scientists as details of experiments are available *immediately* as the instrument collects data, and for engineers as the sensor data of entire smart grids are integrated for state assessment and control. Building this capability of streaming and real-time control will require re-imagining how workflows are formulated since current workflows and leadership class computing facilities are not designed with this type of use-case in mind [6].

We propose a co-design framework for supporting streaming workflows that enables experiments-in-the-loop running over software federations consisting of distributed, heterogeneous compute resources and wide-area networks (Fig. 1) [7]. Future science experiments and smart grids will produce increasing amounts of data; workflows must support high-throughput pipelines while maintaining overall low-latency for real-time control and analytics. As computational resources become geographically dispersed, the complexity of these workflows increases and wide-area networks become an integral part of design. The federations' resources must be persistently available and connected for the lifetime of the experiment, requiring significant coordination between the data generation, data transfers, and computation throughout the entire federation. Enabling these capabilities requires rethinking the traditional scientific workflow from end-to-end and developing the infrastructure and software to best use resources throughout the entire federation, including edge computing, data reduction, and heterogeneous architectures.

A streaming workflow co-design will require three main focuses: 1) understanding the limitations of existing batch workflows and re-mapping them into discrete processing blocks that can be computationally distributed in a stream, 2) orchestrating processing over the entire workflow on distributed resources in a federation, and 3) automating the creation of persistent network pipelines to provide high-bandwidth, low-latency connections between nodes in the streams and enabling pre-processing and data reduction to accommodate bandwidth restrictions while still meeting application requirements.

A. Re-mapping Workflows to Streaming Architectures

The key to enabling streaming workflows is identifying the smallest unit of experiment data that can be streamed through the system and decomposing the overall workflow into individual computational blocks that process those units of data. Once identified, existing workflows can be re-imagined

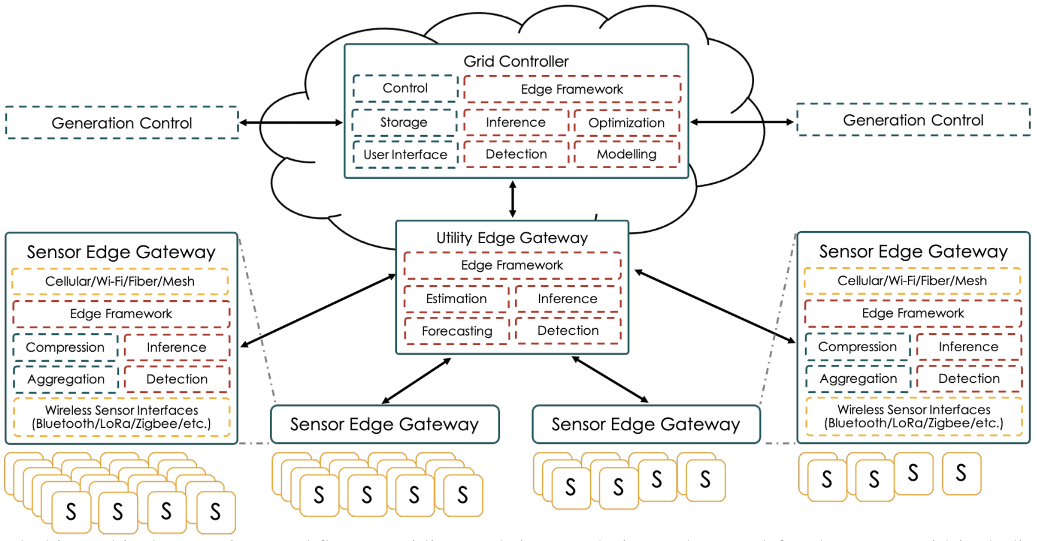


Fig. 1. An example hierarchical streaming workflow providing real-time analytics and control for the smart grid including smart sensors, heterogeneous sensor gateways, utility edge systems, and the overall grid controller and power generation control interfaces.

as a series of data flows where processing is done in-situ while data is transferred from the source to computational resources. This is a cross-disciplinary effort between experimental scientists, computer scientists, and engineers to analyze existing workflows and co-design these streaming computations.

B. *Orchestration and Scheduling*

Shifting towards streaming architectures also requires rethinking how computational resources are scheduled and policies defined to support these persistent applications and streams. Unlike cloud environments that are designed for multi-tenancy and easily support persistent applications, traditional HPC systems do not readily fit this use-case. Careful thought is needed to understand how to adapt traditional accounting, policy enforcement, check-pointing, and failover on leadership HPC systems to support these workflows.

Heterogeneous system architectures pose interesting challenges for streaming workflows. Accelerators like graphical processing units (GPU), field programmable gate arrays (FPGA), and neural processing units (NPU) increasingly used in supercomputing systems are not integrated and tuned for stream processing. Alternate and more intelligent scheduling systems must be considered for optimally scheduling these resources based on pending tasks and availability.

C. *Persistent Network Pipelines and Data Reduction*

Streaming workflows will heavily rely on persistent network connections and can benefit greatly from innovations such as SmartNICs [8]. These incorporate onboard compute cores that can process data either in or out of band. Possibilities for pre-processing data in the stream (such as compression, reduction, and translation) exist before it is transmitted over a connection. Additionally, SmartNICs can directly transfer data to other accelerator cards without requiring host processor interaction. This combined with Software Defined Networking (SDN) would be a key contributor for enabling streaming workflows.

III. Timeliness

Advances in different disciplines provide the building blocks that can eventually enable streaming workflows. Improvements in embedded and edge hardware, network connectivity, and distributed computing frameworks are the main enabling technologies that are beginning to provide the structure required for streaming workflows. This connectivity in turn increases the demand for real-time control over experiment and engineering workflows.

Our approach will build on these advances including: edge computing, SDN, Digital Signal Processing (DSP), Software Defined Radio (SDR), and science federations. Both DSP and SDR applications are two domains that map well to the streaming paradigm and require high-throughput, low-latency processing and as such provide a strong basis for which to build streaming scientific workflows. The vision for streaming workflows is to provide immediate visualization of experiments, real-time control for sensor systems, and to usher in the fourth paradigm of science: obtaining *understanding* from data.

References

[1] "Software defined networking for extreme-scale science: Data, compute, and instrument facilities," DOE ASCR Workshop, Bethesda, MD, 2014.
[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[3] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.
[4] P. Beckman *et al.*, "5G Enabled Energy Innovation: Advanced Wireless Networks for Science, workshop report," Mar. 2020.
[5] M. Gahegan, "Fourth paradigm GIScience? Prospects for automated discovery and explanation from data," *International Journal of Geographical Information Science*, vol. 34, no. 1, pp. 1–21, 2020.
[6] J. E. McClure *et al.*, "Toward real-time analysis of synchrotron micro-tomography data: Accelerating experimental workflows with AI and HPC," in *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, pp. 226–239, Springer, 2020.
[7] T. Naughton *et al.*, "Software framework for federated science instruments," in *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, pp. 189–203, Springer, 2020.
[8] "BlueField SmartNICs," accessed Feb. 15, 2021. www.mellanox.com/products/BlueField-SmartNIC-Ethernet.

From Quantum to Computing: Codesign for quantum bit control

Gang Huang[1] , Anastasiia Butko
Lawrence Berkeley National Laboratory

Topic: architectures, emerging technologies and codesign methodologies
Challenge:

As the silicon based computing system approaches its physics limitation, the quantum computer is one of the possible routes to provide next generation high performance computing to advance the science and DOE mission. The overall quantum computing is still in its early stages and the scalable qubit control system is one of the challenges and opportunities for the hardware/software codesign.

Quantum bit (qubit) control system is the classical electronics together with the firmware and software running on them to bridge the quantum algorithm and low level quantum hardware. The control system generates and measures the radio frequency (RF) pulse, DC signal or laser pulse to implement the quantum gates and readout for different low level quantum hardware.

Typical qubit control system consists of analog hardware, digital hardware and software. The analog hardware determines the signal performance seen by the quantum hardware, including the noise, the bandwidth etc. The digital hardware is typically implemented on/around a field programmable gate array (FPGA). On the FPGA, domain specific soft core or hard core can be utilized to accomplish more flexible signal processing. Basic software API is essential to interact with the upper layer code or operators. As the technologies emerge, the boundaries among the layers are getting blurry, e.g. more and more logic can be implemented on the FPGA to achieve lower latency, the system on chip (SoC) design simplify the system design. For an integrated system, the overall system architecture needs to trade off among multiple dimension requirements. The system needs to be expandable to control hundreds to thousands of qubits in the near future while maintaining flexibility, performance and also a reasonable budget and footprint.

The control system complexity is proportional as the number of qubits increase. Tens of qubits or even hundreds of qubits control systems can be achieved by bruteforce duplicating current control systems, but beyond that, the system architecture must be reconsidered to accommodate the emerging technologies. A full stack understanding of the overall design is essential to the architecture level design and optimization crossing the analog domain and the digital domain.

The major industry leaders, such as IBM[1], google[2], ionQ[3], rigetti[4] etc, built their qubit control system for their quantum hardware. These systems successfully operate tens of qubits and demonstrate quantum supremacy[5]. The low level control stacks of these are not open source to the community. Research groups developed smaller systems based on the standard lab instruments and in-house developed software to support advanced research. These systems are flexible to implement experiments on the small scale while expanding them to the next stage is a challenge.

[1] ghuang@lbl.gov

Opportunity:

Distributed compact control modules allocated right next to the device will be ideal for the system integration. That is still far from the current reality of racks of electrons controlling few qubits, so more development and evaluation are required to properly specify the system in the right architecture. Along the path, the R&D will generate hardware, firmware and software design as well as develop the market of quantum computing.

Several emerging transformative technologies have the potential to change the qubit control system. The chiplets approach might integrate the analog and digital modules faster and less expensively. The open source hardware such as RISC-V provides an opportunity to develop instruction set architectures suitable for classical quantum integration. RTL level design and verification with the systemverilog to further modularize the hardware and the digital signal processing.

To meet the flexibility and performance requirements together, the system can be further modularized using the codesign methodologies. Combining the open hardware and open source gateware modules, together with the upper level software to build a system.

It is a great opportunity to team up the national laboratory, the university research group and the industry partners, to design and build open source stack, from the hardware to the firmware and software to support multiple quantum platforms.

Timeliness of maturity:

In the last decade, the quantum bit coherence time and achievable circuit depth has been significantly improved on the superconducting, trapped ion quantum hardware platform. The continuous improvement of quantum hardware makes it possible to develop larger scale quantum systems and create the requirement of larger, expandable control systems.

Reference

1. https://www.ibm.com/quantum-computing/
2. https://quantumai.google/
3. https://ionq.com/
4. https://www.rigetti.com/
5. Arute, F., Arya, K., Babbush, R. et al. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5

Compilation Infrastructure for Extremely Heterogeneous Systems

Clayton Hughes, chughes@sandia.gov
Siva Rajamanickam, srajama@sandia.gov
Kevin Pedretti, ktpedretti@sandia.gov
Simon Hammond, sdhammo@sandia.gov

Topics: architectures, applications, emerging technologies, codesign methodologies

Challenge

The post-exascale era in computing will look much different from today's landscape, requiring heterogeneous node and system architectures to achieve power, cost, reliability, and usability requirements while maintaining the rate of increase of application performance. Investments made through the Department of Energy's (DOE) Advanced Scientific Computing Research (ASCR) and Advanced Simulation and Computing (ASC) programs have produced strategies for porting applications to heterogeneous nodes comprised of well-understood components (CPUs/GPUs). However, the slowing of Moore's law is driving the computing community toward more specialized forms of compute to achieve performance, leading to an explosion of accelerator designs from both industry and academia. At the same time, open source hardware is democratizing co-design of domain optimized silicon. Leveraging the new types of compute that these devices bring will require new co-design methods to realize their potential and will be critical to the success of DOE's computing mission.

The current co-design approach assumes that new systems will be well-aligned with previous technologies. However, application- and domain-specific accelerators have been shown to provide significant speedups [1, 2]. If the design space is less constrained, when there are options to incorporate novel types of compute, such as dataflow, neuromorphic, analog or application-specific accelerators, then we must rethink how we approach co-design and system procurements. It's likely that there will be options to incorporate multiple types of these accelerators at both the node and system level. The question of how to effectively exploit these resources is an open question. Moreover, power and cost constraints will limit purchases to a subset of the available accelerators, making it critical for DOE to identify architecture requirements that represent the largest possible subset of its workload.

Identifying common patterns across the DOE application space and automatic offload will lead to better co-design efforts and more targeted procurement activities, becoming an integral part of the co-design feedback loop. Many of the emerging accelerators require domain specific languages (DSLs) or vendor-optimized libraries to achieve performance [3, 4]. Recent experiences migrating applications to GPUs have shown how difficult it is porting even a few kernels to a new architecture. Isolating profitable offload targets in million-line applications or across the breadth of the DOE application space for multiple accelerators is not feasible. And although the DOE has invested heavily into performance portability tools, like Kokkos and RAJA, these would still require input from domain experts for each accelerator. Moreover, compiling for all possible machine targets can still lead to long compile times and binary bloat. Compile-time substitution and runtime JIT compilation [5] can solve this problem if common application patterns can be identified and automatically mapped to the appropriate heterogeneous hardware.

Opportunity

Addressing the problem of mapping low-level application behavior to heterogeneous accelerator functionality is a non-trivial problem that spans multiple expertise domains – the fundamental physics required for simulation informs the algorithms and data structures; algorithm scalability, multi-physics coupling, and workflows inform the communication and storage requirements; and together these determine the optimum hardware configuration. The DOE national labs, with their industry and academic partners, may be the only stakeholders capable of addressing this problem in a way that will benefit the nation, from national security to commercial products.

Combining expertise in devices, hardware, software, and compilers, it is possible to develop a framework capable of performing in-depth static and dynamic analysis of application behavior, mapping those kernels to the appropriate accelerator or even developing accelerators customized for a given set of behaviors. The reduced development time provided by such a framework would save the DOE billions of dollars in development costs and open the accelerated compute space up to a wider range of applications, leading to greater scientific discoveries. Providing facilities and procurement teams this information will lead to more targeted procurements and to better direct NRE funds to influence vendor roadmaps.

Timeliness:

The DOE must position itself to reap the benefits of the availability of a wide range of computational accelerators and the opportunities presented by open source hardware. The multi-layer co-design stack that has served the DOE in the past are no longer sufficient. A multidisciplinary approach to extend HPC computing into post-Moore's Law technologies, is critical for applications spanning larger demands on scientific and embedded computing capabilities and increased use of specialized compute accelerators enabled by:

- The widespread adoption of open-source compiler technologies, such as LLVM, is broadening the research into high- and low-level optimization
- The RISC-V ecosystem and assorted RTL development tools are democratizing co-design of domain-optimized silicon
- Advances in material science and fabrication technology is opening the door to the integration of exotic computing devices with traditional von Neumann architectures

References

[1] A. Sanaullah, A. Khoshparvar and M. C. Herbordt, "FPGA-Accelerated Particle-Grid Mapping," in *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2016.

[2] D. E. Shaw and e. al, "Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on*, 2014.

[3] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand and S. Amarasinghe, "Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines," in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2013.

[4] M. Steuwer, T. Remmelg and C. Dubach, "Lift: a functional data-parallel IR for high-performance GPU code generation," in *Proceedings of the 2017 International Symposium on Code Generation and Optimization*, 2017.

[5] H. Finkel, D. Poliakoff, J.-S. Camier and D. F. Richards, "ClangJIT: Enhancing C++ with Just-in-Time Compilation," in *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, 2019.

Position Paper for ASCR Workshop on Reimagining Codesign

Heterogeneous Computing Co-design and Quantum Information Processing

Travis S. Humble, Alex McCaskey, Eugene Dumitrescu, and Raphael Pooser
Quantum Computing Institute, Oak Ridge National Laboratory

Topics: architectures, programming systems, emerging technologies, codesign methodologies

Challenge: Conventional codesign methodologies face many challenges integrating quantum information processing (QIP) into a large-scale, scientific computing infrastructure. QIP, which uses the principles of quantum mechanics to design new algorithms, relies on fundamentally distinct devices to offer dramatic advantages in computing performance. Early examples of quantum advantage have found quantum computing devices can already outperform modern supercomputing systems in terms of time, accuracy, and power [1,2]. These niche demonstrations underscore the potential of QIP to yield computational speed ups for problems in chemistry, materials science, high-energy physics, and artificial intelligence among many others [3].

Over the next decade, we expect QIP to mature rapidly from the current noisy, intermediate-scale quantum devices to fully fault-tolerant quantum computers. These transitions in technology will afford new opportunities for the interplay between hardware capabilities and application goals. QIP, including quantum computing, will support several foreseeable goals in the future of computing [4,5]. As special-purpose computing systems, quantum computers can support new use cases of edge computing for modeling and simulation. As computational accelerators, quantum computers can offer novel platforms to design HPC systems. As networked nodes, fault-tolerant quantum computing systems can enable new paradigms of large-scale data processing capabilities.

How will QIP transition from today's disconnected HPC and experimental, prototype quantum devices to tomorrow's integrated quantum-HPC systems? Integration alongside heterogenous technologies will require well-articulated memory and execution models, each imposing requirements on devices, languages, and algorithms. Reimagining codesign must account for the changes brought by the new quantum computational model and identify the frameworks by which to build and evaluate such systems.

Opportunity: The effort to reimagine codesign affords an opportunity to include quantum computing in the design and eventual development of the most performant computing systems. By including quantum computing in codesign now, we can foster the development of the technology itself and ensure future compatibility with industry standards and practices. Quantum computing technology embraces the principles of heterogeneous computing by necessity and separate, independent development would splinter the remarkable capabilities offered by combining multiple computational models.

Several efforts are already underway to identify early co-design opportunities for QIP. This includes multiple research programs from DOE ASCR to build 1) new software and programming frameworks for quantum programming [6], 2) test and evaluate the performance of isolated quantum computing devices [7], and 3) prototype quantum-accelerated applications for multiple scientific domains [8,9,10]. Additional industrial development efforts from IBM, Google, Honeywell, Quantum Brilliance and Rigetti among many others are enabling early commercialization opportunities for these technologies that are generating demand for better interfaces and applications.

New tools to design and evaluate the integration of QIP with future computing systems are needed. This includes tools to support "Open Source and Extensible Compiler Frameworks" through the design and evaluation of programming such as compilers, numerical simulators, program debuggers, and syntax checkers. Co-design efforts that integrate QIP with future heterogeneous architectures will require the ability to quickly prototype and adapt compiler, debugging, and framework software technologies. As quantum hardware continues its technological advance toward available fault-tolerant architectures, quantum software must quickly adapt and enable research and development of quantum algorithmic implementations. New tools to design and evaluate the execution of quantum-accelerated applications using program simulators, system simulators, timing, and resource estimation. This includes the development of "Standardized Accelerator Interfaces" and "Quantitative Tools of Codesign (ModSim)". Finally, new tools and standards are needed to design and evaluate quantum-accelerated applications including performance measures and benchmarks.

At ORNL, we have begun prototype work that leverages the extensible LLVM and MLIR frameworks for relevant quantum compilation tasks [11]. We anticipate that the MLIR will play a vital role in future quantum-classical hardware-software co-design efforts due to its unique language dialect extensibility, modular and reusable design, and connection to lower-level assembly language dialects, thereby promoting tight integration with classical heterogeneous architectures. We also have a growing repertoire of methods for the evaluation of quantum computing devices that can start the development of full-fledge benchmarks for quantum computing systems [12].

Timeliness or maturity: Although current QIP devices generally lack the resilience and performance to directly rival conventional computing systems, they are expected to mature rapidly. This emphasizes the urgency of early engagement that naturally addresses the "Co-development of hardware and algorithms" using quantum computing devices. It is also timely to pursue "System Level Design for New Workflows" for quantum computing that clarifies the quantum advantage that may be afforded to future computing systems. These approaches are also certain to enable new opportunities for DOE user facilities to integrate the advantage of quantum computing more quickly and efficiently into future designs, but only if sufficient early insights are available for planning that advances.

References

1. F. Arute et al. "Quantum supremacy using a programmable superconducting processor." *Nature* 574 (2019): 505-510.
2. H.-S. Zhong et al. "Quantum computational advantage using photons." *Science* 370.6523 (2020): 1460-1463.
3. Extreme Heterogeneity 2018 - Productive Computational Science in the Era of Extreme Heterogeneity: Report for DOE ASCR Workshop on Extreme Heterogeneity. https://doi.org/10.2172/1473756
4. K A. Britt and T. S. Humble. "High-performance computing with quantum processing units." ACM Journal on Emerging Technologies in Computing Systems 13.3 (2017): 1-13.
5. K. E. Hamilton et al. "Accelerating scientific computing in the post-Moore's era." ACM Transactions on Parallel Computing 7.1 (2020): 1-31.
6. Advanced Scientific Computing program office, Advanced Research for Quantum Computing
7. Advanced Scientific Computing program office, Quantum Testbeds
8. Advanced Scientific Computing program office, Quantum Algorithm Teams
9. Advanced Scientific Computing program office, Quantum Computational Application Teams
10. Basic Energy Science program office, Materials and Chemical Sciences Research for Quantum Information Science
11. A. McCaskey, and T. Nguyen, "A MLIR Dialect for Quantum Assembly Languages," https://arxiv.org/abs/2101.11365
12. A. J. McCaskey et al. "Quantum chemistry as a benchmark for near-term quantum computers." *npj Quantum Information* 5.1 (2019): 1-8.

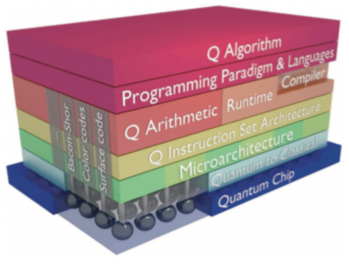
Opportunities for Co-Design in the Hardware/Software Control Stack for Quantum Processors

Anastasiia Butko (abutko@lbl.gov), Gang Huang (ghuang@lbl.gov), ***Costin Iancu*** *(cciancu@lbl.gov)i*
Lawrence Berkeley National Laboratory

Topic: architectures, applications, modeling and simulation, programming systems, emerging technologies, codesign methodologies

Challenge: Quantum computing has the potential to provide transformational computing capability to many scientific domains. Our view is based on developing the control stack for the Berkeley Advanced Quantum Testbed[7] and our in-house quantum processors. Programming and controlling a quantum processor require deploying a complex hardware execution pipeline in conjunction with a complex software stack. Figure 1 depicts the interaction between components in a holistic view of the programming and execution environment. On the top of the figure is software flow of program representations at different levels of abstraction while the program is compiled and executed. Note that intermediate steps are optional, e.g. an application can be compiled directly into pulses.

Hardware flow on the bottom depicts the hardware processing pipelines that generates pulse level representation of the circuit. The first observation is that due to these highly complex interactions, co-designing the software/hardware control stack can provide large benefit. The main challenge here is that we still do not have one good qubit "implementation" or QPU architecture able to scale to a "quantum supremacy" system size. Thus, we are still developing a sophisticated control stack for "under-specified" hardware, which increases the importance of codesign in the development of software/hardware control for quantum computing.



MICRO-50, October 14-18, Cambridge, MA, USA

Figure 1. Overview of the quantum computer system stack [2]

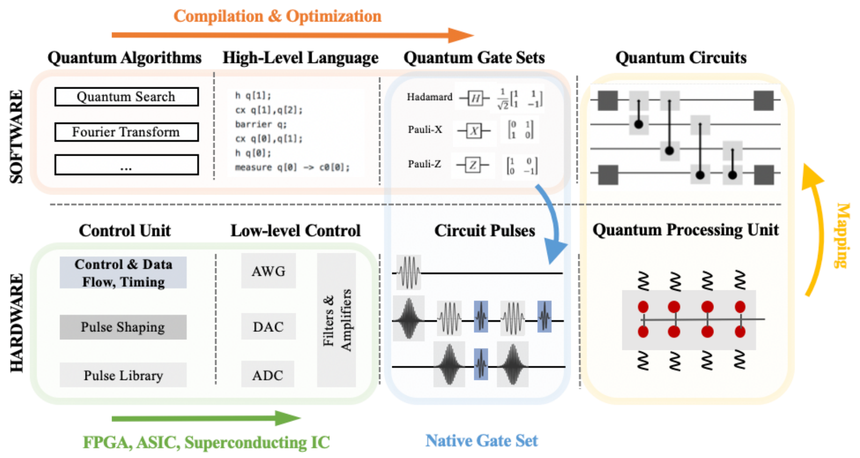


Figure 2. Our vision: Quantum Software-Hardware Co-design

Opportunity: Opportunities are abundant and feedback/codesign loops can be identified between components at any level of abstraction. Examples of **co-development of hardware and algorithms** include:

- *Algorithm-QPU codesign for error mitigation* [1]. When hardware operations are noisy, ideal algorithms need to be reimplemented as noise aware. On the other hand, the structure of the algorithm (circuit) itself determines hardware noise. Can we build chips that respond better to the noise profiles for applications in a given science domain? Can we build approximate implementations for an algorithm given a noise profile? Note that this can be also viewed as "**codesign for improvement in performance portability**".
- *Hardware design exploration* [3]. Our recent work in circuit synthesis provides methods to generate an optimal depth (depth = performance) circuit from an algorithm description. Optimality enables predictive power for

hardware design exploration and we've been using this approach to propose hardware gate-sets and chip topology to maximize "performance" for classes of algorithms.

- *Algorithm-ISA-QPU co-design for performance [4].* Our recent work in characterizing space-time circuit structure introduces metrics that guide hardware control pipeline (ISA) and QPU design in terms of required internal parallelism when executing gates.

Compiling and optimizing quantum programs provides a rich opportunity for "**system level design for new workflows**". Given the short chip coherence times, we have severe restrictions on circuit depth and optimal implementations are paramount. Optimal compilation is compute-intensive and generates interactive and elastic (large scale) workloads and workflows that can benefit from hardware acceleration. Our recent work [5] showcases the (high) computational needs for circuits of 100 qubits or less. This need is likely also to be perceived by applications that use hybrid quantum-classical approaches.

Furthermore, the field is rife with opportunities for "**quantitative tools of codesign (ModSim)**" in the area of building [2][4] the hardware control pipeline.

Besides the obvious benefit for feedback loops in the hardware/software control stack for quantum processors, there are many other feedback loops enabled and synergy with methods employed in other domains. For example, in our recent work [3,5,6] we've been using numerical optimization in synthesis, as well as for hybrid quantum-classical algorithms. This is an example of "**co-development of algorithms**", as we can see huge benefits from specializing the numerical methods for the quantum domain specific objective functions. We have several other inter-disciplinary examples.

Timelines and maturity: Quantum computing is one of the few emerging technologies that requires a deeply embedded codesign development process in order to succeed. The qubit technology keeps improving and current demonstrations on systems containing tens of qubit show promise. The challenge is building larger scale systems while tracking somewhat rapid changes in qubit technology. For example, we are working with commercial companies (arbitrary waveform generators manufacturers) that embraced the concept and update their hardware/software environment based on feedback for our physics experimentalists. The field is young, has tremendous potential and we expect the need for codesign only to increase for the upcoming decade and after.

References:

[1] *Algorithm-Centric Error Mitigation*. Ed Younis et al. APS 2019
[2] *An Experimental Microarchitecture for a Superconducting Quantum Processor*. X. Fu et al. IEEE Micro 2017
[3] *Towards Optimal Topology Aware Quantum Circuit Synthesis*. IEEE QCE 2020.
[4] *Understanding Quantum Control Processor Capabilities and Limitations through Circuit Characterization*. A Butko et al. 2020
[5] *QFAST: Quantum Synthesis Using a Hierarchical Continuous Circuit Space*. Ed Younis et al. IWQC 2020
[6] *Classical Optimizers for Noisy Intermediate-Scale Quantum Devices*. W Lavrijsen et al.
[7] Berkeley Advanced Quantum Testbed. `https://aqt.lbl.gov/`

Reconceptualizing Quantitative Codesign

Terry Jones[1]*, Jim Brandt[2] , David Montoya[3], Oscar Hernandez[1], Michael Jantz[4], Ann Gentile[2]
[1]ORNL, [2]Sandia, [3]Trenza, [4]University of Tennessee
* corresponding author: trj@ornl.gov

Topic: architectures, programming systems, codesign methodologies

Challenge: Collection and quantitative analysis of appropriate metrics across the High Performance Computing (HPC) ecosystem is of critical importance to codesign. Its impact spans the whole spectrum of stakeholders. Whether providing hardware architectures, system software, application programming environments, or production run-time environments, having the appropriate knowledge to optimize the interaction of all of these critical components as well as the evolution of the HPC ecosystem is critical to continued growth.

Today, HPC computing centers collect a wealth of information on the health, usage, and efficiency of our machines, workflows and programming environments. While collection and analysis of this information has evolved and improved over the years, there are still severe gaps that have left us unable to provide the knowledge that is needed by hardware and software vendors, system operations staff, application developers, and user groups to create and operate highly efficient large scale HPC systems. Would-be users of this information face significant challenges in obtaining effective analysis in a timely manner and efforts to provide it are currently fragmented across centers. The infrastructure to collect, store, share and analyze the volumes of available information is a core capability — yet many challenging barriers remain due in large part to the many stakeholders and insufficient coordination. With many new potential information sources in future systems, it is urgent that we identify and address critical requirements and gaps across the various ASCR stakeholders. Doing so will enable us to create collective and collaborative solutions that address both existing challenges and emerging needs and effectively support our upcoming ASCR environments. Design solutions that rely on intelligence derived from the data collection and analysis processes described above are henceforth referred to as ***Quantitative Codesign***.

Opportunities abound. Making progress at the highest end of HPC without access to the needed data can be compared to being asked to fly an airplane at night without sufficient instrumentation. Vendors are provided with example applications to target but often lack a true understanding of where inefficiencies manifest on full scale workloads. Furthermore, architectural simulators do not incorporate the critical performance-killing attributes of current generation technologies and their integration. Hence the vendors miss opportunities for improvement. Moreover, users often only have feedback on operating efficiency at the granularity of total application execution time. Low-level interactions frequently cause substantial performance degradations that users are unable to explain. Likewise, operations staff often lack knowledge of application resource utilization and cannot diagnose the longer run times experienced by the users. Since root causes go undiagnosed, next generation systems also fail to address the problems.

Opportunity: First and foremost, we seek a coordinated effort to bring together the pertinent data from each shareholder in the codesign space into a framework where data discovery and access is straightforward regardless of data source. The envisioned Quantitative Codesign environment would pull together data traditionally held by disjointed communities (e.g., sysadmins, application teams, vendors, and so on) into a framework where the needed data is easily accessible. This framework would provide mechanisms for data providers who wish to share their data with others including application teams, vendors, facilities, operations, and system software researchers. In many cases, we seek to bring together data that is currently being produced although not generally known or utilized for a variety of reasons; in a few instances, we seek to extend and provide new data collection capabilities.

For example, one area that is ripe for integration with Quantitative Codesign processes is the intersection of application development and run-time environments. In the past few years Continuous Integration (CI) has been widely adopted by development teams to continuously test development efforts. As part of these CI efforts, developers test across a variety of platforms on a daily basis and typically provide a pass/fail result for each. Introducing targeted run-time data collection (e.g., mem, mpi, program counters, omp, gpu, IO) and quantitative analysis into this process would enable feedback to users and identify issues within applications, compiler capabilities, runtimes, and differences across platform architectures that ultimately would drive improvements across the spectrum of stakeholders.

Integrating Quantitative Codesign capabilities with existing design processes will enable more effective solutions across the computing stack. Information derived from monitoring and analysis would provide valuable insight for users, application developers, and system architects as to how, and why, applications make use of the underlying system resources. Furthermore, by identifying the appropriate stakeholders and introducing them to information originating from diverse collection regimes, this initiative would facilitate the discovery and sharing of potentially useful intelligence among larger teams and communities. In doing so, this approach also has the potential to spark further discussions and research on how to collect and employ this information more effectively. Thus, there is significant opportunity for discoveries that will not only increase application performance, but also benefit the broader HPC and scientific communities.

Timeliness or maturity: Over the past decade, there has been a growing awareness of the multi-faceted benefits we can derive from data-driven strategies like Quantitative Codesign. This increasing awareness, along with improvements in Machine Learning (ML) technologies, have driven vendors, operations staff, and application developers to espouse integrating an ever-increasing level of instrumentation into their products. The time is ripe for turning this vast trove of available information and the incredible advances in analysis technologies it represents into appropriate knowledge and understanding. Doing so would create a feedback loop that could assist vendors and software developers in their designs. The recent National Strategic Computing Initiative Update Report has recommended that we promote timely access for developers of technologies, architectures, and systems to carry out the research needed to create the future computing software ecosystem [1], and Quantitative Codesign provides a solution to the 'access problem' of these extremely rare machines. If the future envisioned by the CSESSP report [2] is to be realized, our software base will require significant investment in both modified and new code — an activity enormously assisted by Quantitative Codesign. There is no disagreement that more knowledge is good though there is still lack of concurrence across HPC stakeholders as to the cost/benefit tradeoff for varying fidelities of information collection and long term storage. The benefits of Quantitative Codesign will come through integrating design processes with more detailed knowledge of the interactions of the various components within the HPC ecosystem.

Quantitative Codesign is also essential for addressing challenges brought about by the recent trend of increasing heterogeneity in HPC architectures [3]. For example, many HPC machines now incorporate alternative types of memory alongside conventional DDR SDRAM. Technologies such as "on-package" or "die-stacked" DRAM as well as non-volatile RAMs can provide distinct advantages compared to conventional DRAM, including higher performance as well as cheaper and more energy efficient storage per byte. Each of these technologies also comes with its own limitations, such as smaller capacity or less bandwidth for reads and writes. Further complications arise because some of these new technologies can interface directly with processor caches, while others can only be accessed through peripheral devices, such as GPUs or other accelerators.

Quantitative Codesign could mitigate many of the current problems with allocating and managing such heterogeneous resources effectively. Detailed knowledge of application demands will enable architects to make better decisions about how to select and organize computing hardware. This approach can also help system software, including operating systems, compilers, and runtime software, distribute the available hardware resources among applications more effectively. Integrating high-level profiling and analysis with low-level resource management routines will enable these systems to implement new policies that respond flexibly to changes in application demands, and could potentially expose powerful new efficiencies on platforms with heterogeneous hardware.

References

[1] National Strategic Computing Initiative Update: Pioneering the Future of Computing. https://www.nitrd.gov/news/National-Strategic-Computing-Initiative-Update-2019.aspx

[2] Workshop Report: Computational Science And Engineering Software Sustainability And Productivity (CSESSP) Challenges. https://www.nitrd.gov/news/CSESSP_Workshop_Report.aspx

[3] Vetter, Jeffrey S., et al. "Extreme Heterogeneity 2018 - Productive Computational Science in the Era of Extreme Heterogeneity: Report for DOE ASCR Workshop on Extreme Heterogeneity". United States. https://doi.org/10.2172/1473756. https://www.osti.gov/servlets/purl/1473756.

Future Opportunities for Facilities in Co-design

Balint Joo, joob@ornl.gov, Oak Ridge Leadership Computing Facility

Topics: architectures, applications, programming systems

Introduction

A highly successful co-design effort from 2000 was the design of the QCDOC supercomputer [1] co-designed for efficient simulations of lattice Quantum Chromodynamics (LQCD) between Columbia University, the RIKEN-Brookhaven Center, and by the UKQCD Collaboration and IBM corporation. The system was designed alongside the BlueGene/L supercomputer and shared several common components including the structure of the serial internode fabric, and the 3-way memory controller. Very similar in spirit to the solicitation for this proposal, the QCDOC utilized: an embedded PowerPC-440 core customized with SoC components from a library of intellectual property components from IBM including an eDRAM scratch pad memory, floating point processors, high speed serial links, and an ethernet adaptor. The software stack included a customized version of the GNU C and C++ compilers, a cut down version of GLIBC, a minimal operating system that allowed the system to be booted, and programs to be executed, and a lean runtime that provided access to the communications links, and calls to allocate memory in the scratchpad. QCDOC was successful in the domain of LQCD with systems at Brookhaven National Laboratory, and Edinburgh University. The BlueGene series of supercomputers also became highly successful and lasted through two further generations with the BlueGene/P and BlueGene/Q models, which were legendary for their low latency, high bandwidth fabrics.

Challenges of Customized Systems

A key challenge for custom built systems such as QCDOC is the software stack which was purposefully lean, both to not get in the way of the application and because QCD applications of the time needed no other major dependencies, and also due to resource constraints. There was no real queueing system, nor a full MPI[2] stack, nor other common libraries, or high level languages such as Python [3].

User facilities require more generality, especially considering a future where simulations are expected to be combined with machine learning, analytics and possibly quantum and neuromorphic computing. Architecturally, several classes of system appear to be required: strong scaling systems, weak scaling systems and analysis farms and Edge nodes featuring different tradeoffs between communications capability, memory-size and bandwidth, floating point performance and I/O. As an example strong scaling systems emphasize the highest performing fabrics while weak scaling systems can trade off network performance and emphasize higher performance on-node (e.g. using accelerators).

In terms of software stacks, bringing together the future of simulation, data analysis, and AI methods poses challenges for application developers. Many AI oriented applications utilize high-level frameworks in languages such as Python or Julia[4], while simulations tend to be carried out using C++ and Fortran. Accelerator programming models vary from corporately owned solutions (e.g. CUDA[5]) to open source copies (HIP) to ones based on open standards (SYCL[6] and OpenMP[7]). Interoperability solutions exist (e.g. Python and C interfaces, Julia and C/C++ interfaces) to bring these models and languages together.

When considering a future chip the following key questions arise for application and library developers: a) which programming models (PMs) will be available? b) will there be a single PM to control all the heterogeneous hardware or will separate PMs need to be used? c) will the PMs interoperate? d) what is the timeline for the availability of the PMs? e) are there good proxy systems to develop on right now f) where will there be performance bottlenecks? The answers can influence the application design and the algorithms used.

A facility may have other questions: a) Is it better to operate several different systems or an extremely heterogeneous system in a variety of modes? Who will provide the software infrastructure (compilers, libraries for simulation, analysis, learning etc.) for such novel hardware. Who has the capability? Does the facility need to contract out e.g. PM development?

Opportunities for Facilities

Facilities have a crucial role to play in co-design activities. Knowing their application base they can brainstorm with vendors on what heterogeneous components on a chip would be most useful to the facility, and would have a good idea of the impact of bottlenecks. Facilities can also proactively partner with applications in co-design activities providing staff and expertise. They can provide the vendors with both traditional mini-applications, and potentially 'mini-workflows' that can simulate how an application plans to utilize different systems (e.g. for simulation at one stage or for machine learning at another). Liaisons at facilities can usefully engage with vendors to test and harden early PM implementations as well as to explore interfaces between the PMs. Facilities already have experience in these areas.

The SYCL programming model and the Julia programming language hold some future promise as do re-targetable models such as Kokkos and Raja. SYCL is a parallel programming model standard from the Khronos Group based in C++ and can currently support a variety of accelerators (GPUs, FPGAs). Julia is a high level, high productivity language used in data intensive applications and AI. Kokkos and Raja provide portability to C++ codes via back ends (e.g. in SYCL, CUDA, HIP or OpenMP). Implementations of Julia are tied in heavily with the LLVM compiler suite[8] for Just-in-Time compilation whereas one of the prominent SYCL implementations (the open source one from Intel) is also based on LLVM. The support in LLVM for heterogeneity comes through its Internal Representation (IR). Compiler front-ends generate IR, which after optimization passes is lowered by the 'back-end' into the target representation. The latter is either a binary or another kind of IR such as NVIDIA PTX[9], or SPIR-V[10] which can be executed (e.g. by just-in-time compiling into binary and loaded) by the device driver. The efficacy of a driver consuming an intermediate assembly (such as PTX) has been amply demonstrated by the success of CUDA. Facilities could play a role in a) fostering the continued development of LLVM working with vedors on the back-end for new accelerators and build up relevant expertise, and in the area of Fortran (FLANG) b) foster the development of IR standards, again especially for new accelerators c) serving on standards bodies (C++, Khronos SYCL, OpenMP, MPI etc.) and encouraging vendors to adopt these standards.

[1] Boyle, P. et. al. Overview of the QCDSP and QCDOC Computers, *IBM J. Res. Dev., 49*(2), 351–365, (2005)
[2] MPI: https://www.mpi-forum.org/
[3] Python Programming Language, https://www.python.org/
[4] Julia Programming Language, https://julialang.org
[5] NVIDIA CUDA:
[6] SYCL 2020 Standard, https://www.khronos.org/registry/SYCL
[7] OpenMP: https://www.openmp.org
[8] Lattner,C and Adve V., "LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation". Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04), Palo Alto, California, Mar. 2004, https://llvm.org
[9] NVIDIA PTX: https://docs.nvidia.com/cuda/parallel-thread-execution/index.html
[10] SPIR-V: https://www.khronos.org/registry/spir-v/

Architecture-Aware Modernization of HPC Applications

Mahmut Taylan Kandemir (Presenter), George Kesidis, Bhuvan Urgaonkar
{mtk2,gik2,buu1}@psu.edu
School of EECS, Pennsylvania State University

Topics– Architectures, Programming Systems, Emerging Technologies, Co-design Methodologies.

Challenge– Given an application program, achieving optimum performance/energy-efficiency requires a careful selection of hardware-mix to employ and corresponding code-data layout restructuring. We identify four important challenges that have to be addressed when developing a co-design framework that satisfies the performance and energy-efficiency constraints of applications that are of interest to DOE:

Hardware Heterogeneity Challenge: The problem of 'hardware heterogeneity' is how best to exploit the significant diversity likely to exist in tomorrow's data centers and supercomputers for the benefit of complex, data-intensive HPC applications. Such diversity will be found along multiple dimensions. For example, how might we best exploit the strengths of special-purpose accelerators (e.g., GPUs, TPUs and FPGAs) as well general-purpose CPUs? As another example, how might we combine emerging non-volatile memories (e.g., Intel Optane) alongside conventional DRAM, flash SSDs, and magnetic drives? Solving the hardware heterogeneity problem would allow HPC applications to be easily compiled to effectively exploit diverse hardware types. Thus, a critical challenge is to address this growing hardware heterogeneity.

Systems Software Challenge: Over the years, system software has become increasingly more complex to offer more functionality. However, this comes at a price. Many resource management decisions taken by systems software are application agnostic and can even, in some cases, conflict with individual application-level objectives, as they optimize for system-level objectives. In particular, from an application viewpoint, systems software tends to be a black box, which makes it difficult for the application to fully control its runtime behavior in desirable ways, leading eventually to inefficiencies. Therefore, an important challenge is to retain the programming/management benefits enabled by systems software while removing sources of inefficiency arising from the information gap between applications and systems software.

Disconnect Between Hardware and Software: Current application optimization and mapping frameworks do not adequately take critical architecture-specific information into account in optimization/mapping, e.g., important characteristics of CPU/GPU cores, interconnect information, and memory system parameters. This prevents applications from fully exploiting the potential offered by the underlying hardware with this problem becoming even more acute for environments with higher hardware heterogeneity and specialization. Hence, an important challenge is to bridge this (growing) gap between hardware and software.

Mapping/Remapping Challenge: As both the degree and magnitude of heterogeneity increase, a key challenge will be to port HPC applications written originally for today's heterogeneous hardware (e.g., CPU+GPU) using currently-popular programming paradigms (e.g., MPI+OpenMP+CUDA) into versions suitable for the even higher diversity possessed by tomorrow's hardware (e.g., CPU+GPU+FPGA+ASIC) and programming paradigms (e.g., MPI+OneAPI+serverless functions). Unfortunately, this challenge is magnified by the previous three challenges and, as a result, porting applications (and performance) for each generation of new hardware becomes increasingly difficult. This problem is further magnified by emergence of task-specific hardware (ASICs) that can be used to expedite select portions of an application.

If all these four challenges can be addressed properly, we would be able to lower the burden of re-targeting a given application for new hardware significantly. To our knowledge, there is no existing framework that responds to all these challenges when targeting large-scale DoE workloads.

Opportunity– We envision three pieces that can be integrated to address these four challenges:

Hardware Abstraction: Identifying the right hardware abstraction to expose to software is key to enabling software to be hardware-aware. In one extreme, one can imagine a very high level abstraction that hides most of the hardware details (as most current approaches do). While this option certainly makes application porting easier, it will leave a lot of performance on the table, given the increase in variety of complexity of heterogeneous hardware components. In the other extreme, one can consider providing every detail of architecture to hardware (e.g., number and types of CPUs and GPUs, details of on-chip and chip-to-chip networks, number of memory controllers, memory system details). In addition to the difficulty of providing this information, it is not clear whether it is possible to design any code optimizer to take advantage of it. We believe a right abstraction is between these two extremes, and needs to be explored.

Systems Software Abstraction: Exposing the details of system software to application is as equally important as exposing hardware details to application. As in the case of hardware, deciding the right level of system software abstraction for application is key to achieving the right performance-energy efficiency tradeoffs.

Vertical Integration: With the right level of architecture and systems software abstractions in place, a three-way integration would be needed to achieve a successful *application-architecture co-design*: (i) user input (in the form of code annotations/pragmas capturing application/execution environment constraints), (ii) an optimizing compiler that takes application code, architectural description, system software description, and user annotations as input and can generate optimized output code, and (iii) a malleable systems software stack including cloud software (virtualized services) that can interact with the compiler-generated code. In other words, vertical integration is about defining interfaces using which application, systems software, and hardware can effectively cooperate with one another. It is important to emphasize the role of the compiler here. The compiler will perform all major co-design decisions which will at the end result in the minimum amount of carefully selected heterogeneous hardware-mix to run the application on (e.g., select GPU types, right amount of DRAM/NVM capacity, and right FPGA resources) and accompanying code and data layout restructuring, that satisfy the specified performance and energy-efficiency constraints. It will do so by considering the information provided by the systems software and architecture abstractions.

A Novel Co-design Strategy Based on Architecture Abstraction, Systems Software Abstraction, and Vertical Integration: We propose to explore a novel application-architecture co-design strategy, combining architecture abstraction, systems software abstraction, and vertical integration, described above. More specifically, one needs to (i) decide the minimum amount and types of hardware to use (to address energy-efficiency constraint), (ii) restructure the application program code and it data layout in memory and storage to adapt to the decided hardware (to address performance constraint), and (iii) choreograph interactions among hardware, application, and system software. If the targeted/specified performance and/or energy-efficiency constraint could not be satisfied in the first attempt, (i), (ii), and (iii) should be *iteratively* re-invoked.

Timeliness or Maturity– The vision presented above is possible due to the significant strides achieved in algorithms, architecture, compilers, and system software over the last decade, including some of our own work, e.g., [4, 3, 6, 1, 2, 5, 7]. We are also encouraged by the confluence of expertise, we developed over the years, in systems, performance, machine learning, architecture, and compilers. Successful execution of the proposed research will facilitate effective porting of complex, data-intensive HPC applications to different platforms whose underlying platform has different types/amounts of special purpose devices. More importantly, the framework will be flexible enough to target new accelerators that the DOE systems can incorporate in the future.

References

[1] A.F. Baarzi, T. Zhu, and B. Urgaonkar. BurScale: Using Burstable Instances for Cost-Effective Autoscaling in the Public Cloud. In Proc. ACM SOCC, 2019.

[2] A. Jain, A.F. Baarzi, N. Alfares, G. Kesidis, B. Urgaonkar, and M. Kandemir. SplitServe: Efficient Splitting Complex Workloads across FaaS and IaaS. In *Proc. ACM/IFIP Middleware, 2020.*

[3] Mustafa Karakoy, Orhan Kislal, Xulong Tang, Mahmut Taylan Kandemir, Meenakshi Arunachalam. Architecture-Aware Approximate Computing. In Proc. ACM SIGMETRICS), 2019.

[4] Ashutosh Pattnaik, Xulong Tang, Onur Kayiran, Adwait Jog, Asit Mishra, Mahmut T. Kandemir, Anand Sivasubramaniam, Chita R. Das. Opportunistic Computing in GPU Architectures. In Proc. ACM ISCA, 2019.

[5] Y. Shan, A. Jain, G. Kesidis, B. Urgaonkar, J. Khamse-Ashari, and I. Lambadaris. Scheduling distributed resources in heterogeneous private clouds. In Proc. IEEE MASCOTS, Milwaukee, Sept. 2018.

[6] Xulong Tang, Mahmut Taylan Kandemir, Mustafa Karakoy, Meena Arunachalam. Co-Optimizing Memory-Level Parallelism and Cache-Level Parallelism. In Proc. ACM PLDI, 2019.

[7] C. Wang, B. Urgaonkar, N. Nasiriani, and G. Kesidis. Using Burstable Instances in the Public Cloud: When and How? In Proc. ACM SIGMETRICS, 2017.

Co-Design through Domain-Specific Compilation Framework

Gokcen Kestor*
gokcen.kestor@pnnl.gov
Pacific Northwest National Lab
Richland, Washington, USA

Erdal Mutlu
erdal.mutlu@pnnl.gov
Pacific Northwest National Lab
Richland, Washington, USA

Roberto Gioiosa
roberto.gioiosa@pnnl.gov
Pacific Northwest National Lab
Richland, Washington, USA

Topics: Co-design, architectures, modeling and simulation.

CHALLENGES

Custom hardware accelerators may significantly increase performance, area, and energy efficiency of high-end compute systems. It comes as no surprise that hardware acceleration is widely employed in many domains, including mobile, automotive, high-performance computing, and machine learning. In U.S. Department of Energy (DOE) environments, GPUs have been widely used to speedup scientific computation but, as new workloads, computation paradigms, and power/energy challenges arise, researchers are exploring new energy-efficient accelerator designs to place next to CPU and GPU cores in the same system-on-chip (SoC). However, designing custom accelerators traditionally requires computationally-intensive simulations using software or FPGA-based tools, which may limit the scope to very small kernels. On the other hand, DOE applications are complex, large, and distributed. To understand the impact of novel hardware designs on DOE applications it is paramount to evaluate the performance, area, and power/energy impacts on the entire application at scale. New tools are required to perform an agile co-design cycle where novel hardware designs can be easily swapped in and out while executing a full-scale application.

OPPORTUNITY

Compiler technologies have considerably evolved during the last decades and now support sophisticated code analysis and translation methodologies as well as efficient code generation for target heterogeneous systems. In the context of co-design, a compiler-based tool is desirable because it offers the possibility of automatically generate code that leverages new hardware concepts without or with little code modifications in the applications. We developed COMET [3, 7], a tensor algebra compiler and Domain-Specific Language (DSL) for chemistry applications and extended the original design to support co-design of application algorithms and hardware accelerators.

COMET consists of a DSL for dense and sparse tensor algebra computations, a progressive lowering process to map high-level operations to low-level architectural resources, a series of optimizations performed in the lowering process, and various Intermediate Representation (IR) dialects to represent key concepts, operations, and types at each level of the multi-level IR. COMET is based on the Multi-Level Intermediate Representation (MLIR) framework [2], a compiler infrastructure to build reusable and extensible compilers and IRs. MLIR supports the compilation of high-level abstractions and domain-specific constructs and provides a disciplined, extensible compiler pipeline with gradual and partial lowering. Users can build domain-specific compilers and customized IRs (called *dialect*),

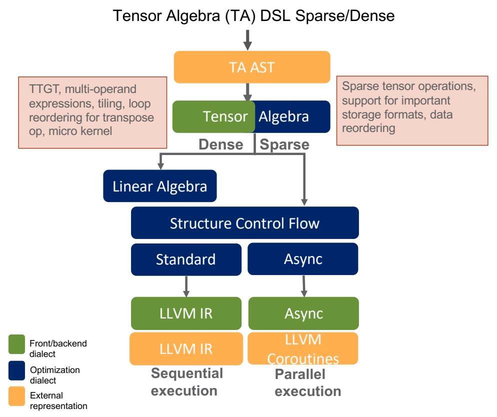


Figure 1: COMET execution flow and compilation pipeline

as well as combining existing IRs, opting into optimizations and analysis. Figure 1 shows the COMET's compilation pipeline and the lowering through the various MLIR dialects.

Tensor contractions are high-dimension analogs of matrix multiplications widely used in many scientific and engineering domains, including deep learning, quantum chemistry, and finite-element methods. For example, the perturbative triples correction in couple cluster CCSD(T) [4] methods used in the NWChem computational chemistry framework [1] originates a 6D output tensor from two 4D inputs tensors. Tensor contractions are computationally intensive and dominate the execution time of many computational applications. A way to perform the above computation is to directly lower to a nested-loop implementation of the problem. Such implementations have been shown to be inefficient due to poor data locality. A more efficient approach, commonly used in modern high-performance tensor libraries, leverages highly optimized GEMM hardware engines and/or architecture-specific implementations. This approach, often referred as transpose-transpose-GEMM-transpose (TTGT), performs the permutations of the input tensors followed by a high-performance matrix-matrix multiplication and a final permutation to reconstruct the output tensor.

COMET provides an opportunity to perform hardware/software co-design and design space exploration (DSE) efficiently and to assess the performance of the entire application, instead of only the innermost kernel. For example, researchers may want to explore the use of hardware GEMM accelerators and repalce the architecture-specific GEMM kernels in the TTGT reformulation of a tensor contraction. In order to perform co-design for a target accelerator and measure the impact of realistic tensor contractions, we replace the micro-kernel with a timing model of the hardware accelerator and execute the entire contraction at native speed. To this extend, we pair COMET with Aladdin [5], a pre-register-transfer

*Corresponding Author

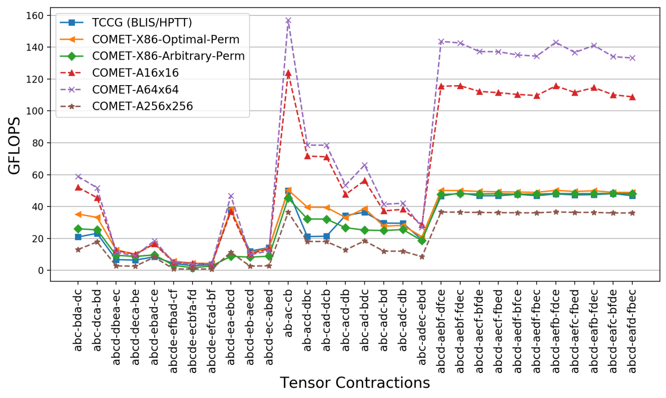


Figure 2: COMET's performance comparison against the hand-optimized TCCG [6] benchmarks on x86 (solid lines) and emulated platforms (dashed lines).

level (RTL), power-performance simulation framework that targets rapid prototyping of data parallel accelerators. Aladdin specifications are essentially C representations of the functionalities that need to be implemented in hardware. From these representations, an LLVM-based tracer extracts a dynamic data dependence graph (DDDG) that describes the accelerator. Next, Aladdin applies various optimizations and resource constraints, therefore generating a realistic design. Finally, Aladdin estimates power and performance from dynamic traces obtained from a driver program. Performing DSE for the hardware accelerator design with Aladdin takes order of minutes (instead of hours as in traditional FPGA-based DSE) while executing tensor contraction with COMET is performed at native speed. The entire process, thus, can be completely automated and executed within minutes.

As an example, Figure 2 shows performance comparison of various tensor contractions from multiple scientific and engineering domains. The main goal of this experiment is to identify the best data-parallel accelerator to perform GEMM operation in the TTGT method to solve tensor contractions. We leverage COMET modeling capabilities and combine the code generated by our compiler framework with the timing estimates produced by Aladdin models of the GEMM designs. In particular, we replace the x86 micro-kernel used for normal code generation with a delay that represents the execution time of the innermost GEMM computation on the hardware accelerator. We analyze three possible scenarios: small (16×16), medium (64×64), and large (256×256). Table 1 reports the hardware characteristics of the three designs in terms of performance, area, and average power. For comparison, consider that an Intel Ivy Bridge measures 160 mm^2 while an NVIDIA Volta GPU die measures 815 mm^2, which are 2,867x and 14,606x bigger than the 16x16 accelerator. Figure 2 reports the performance of a system that features custom GEMM hardware accelerators (dashed lines). The plot shows that hardware accelerators may substantially increase performance for compute-bound tensor contractions, such as the last 11 contractions, and achieve up to 156 GFLOPS, 3.1× speedup over the same code employing a "soft" AVX512 accelerator.

The plot also shows a critical point for hardware/software co-design analysis: while it seems intuitive that larger accelerators

Table 1: Characteristics of the emulated GEMM hardware.

	16x16	64x64	256x256
Perf. (cyc)	131	1026	32770
Avg. Power (mW)	5.077	13.639	73.7972
Avg. Area (uM^2)	55827	224068	4.097e+06

provide higher performance, this is not always the case in our experiments. There may be several reasons for this behavior, including large carry-over loops, computing GEMM for non-square matrices, caches that are not large enough to contain all the data, etc. Figure 2 does show that tensor contractions that are compute-bound with smaller hardware accelerators become memory-bound with the largest GEMM design. We infer that the lowest-level cache does not have sufficient storage to feed such large accelerators or to support data reuse. The actual point of co-design is, indeed, to figure out those trade-offs and select the best accelerator for the particular workload (64×64) instead of the best accelerator from the single operation (256×256).

TIMELINE AND MATURITY

While the COMET compiler was initially designed as a research compiler for chemistry applications, it has evolved into a more general compiler framework for other domain areas, including artificial intelligence (AI), graph analytics, and powergrid, as well as including capabilities for hardware/software co-design. COMET currently supports various projects, such as the PNNL Data Model Convergence Initiative, the DOE co-design of ARtificial Intelligence focused Architectures and Algorithms (ARIAA), and the PNNL High-Performance Data Analytics project. COMET is now looking to expand to support additional programming framework beyond the native DSL as an embedded DSL for host languages such as Julia, Python, and Rust. COMET will soon be open sourced and available to the scientific community.

ACKNOWLEDGEMENT

This research is supported by PNNL Laboratory Directed Research and Development Program (LDRD), Data-Model Convergence Initiative, project DuoMO: A Compiler Infrastructure for Data-Model Convergence, and project Hybrid Advanced Workflows.

REFERENCES

[1] E. Aprá et al. 2020. NWChem: Past, present, and future. *The Journal of Chemical Physics* 152, 18 (2020), 184102. https://doi.org/10.1063/5.0004997
[2] Chris Lattner et al. 2020. MLIR: A Compiler Infrastructure for the End of Moore's Law. *arXiv preprint arXiv:2002.11054* (2020).
[3] Erdal Mutlu, Ruiqin Tian, Bin Ren, Sriram Krishnamoorthy, Roberto Gioiosa, Jacques Pienaar, and Gokcen Kestor. [n.d.]. COMET: A Domain-Specific Compilation of High-Performance Computational Chemistry. In *Workshop on Languages and Compilers for Parallel Computing (LCPC'20)*. Springer.
[4] Krishnan Raghavachari et al. 1989. A Fifth-Order Perturbation Comparison of Electron Correlation Theories. *Chemical Physics Letters* 157 (05 1989), 479–483.
[5] Yakun Sophia Shao et al. 2014. Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures. In *ISCA*. IEEE Press, 97–108.
[6] Paul Springer and Paolo Bientinesi. 2018. Design of a High-Performance GEMM-like Tensor–Tensor Multiplication. *ACM Trans. Math. Softw.* 44, 3, Article 28 (2018), 29 pages.
[7] Ruiqin Tian, Luanzheng Guo, Jiajia Li, Bin Ren, and Gokcen Kestor. 2021. A High-Performance Sparse Tensor Algebra Compiler in Multi-Level IR. arXiv:2102.05187 [cs.DC]

Extracting Maximum Information from Exascale Simulations

Ryan King (NREL, ryan.king@nrel.gov), Andrew Glaws (NREL, andrew.glaws@nrel.gov), Kenny Gruchalla (NREL, kenny.gruchalla@nrel.gov), Marc Henry de Frahan (NREL, Marc.HenrydeFrahan@nrel.gov)

Topic
Modeling and simulation

Challenge
To date, DOE investments in exascale computing codes and high performance computing resources have largely focused on growing the size or complexity of problems that can be simulated. Recent advancements in differentiable programming, in-situ learning, and visualization offer new pathways for scientific insight beyond running ever larger simulations; however, obtaining these additional outputs introduce new hardware challenges regarding coordinating heterogeneous computing resources for simulation and learning, careful communication and inspection of training data, and new programming paradigms for differentiable computing that must be overcome in a coordinated fashion.

Opportunity
We propose to expand conventional notions about the outputs of scientific simulations. We believe that design gradient information, trained machine learning models, compressed data representations, and streaming visualization are all equally important outputs that ought to be considered as first class citizens when designing future computational workflows and computing architectures. With careful hardware co-design, these additional outputs would make insights from exascale simulations portable through the use of ML surrogates, interpretable through the use of advanced visualization, reproducible from compressed representations, and directly translatable to gradient-based design optimization.

Differentiable Programming
A new progamming abstraction, called *differentiable programming (dP)*, is emerging to help automate the process of obtaining outer loop design gradients in a scalable fashion[1]. This approach is inspired by the success of backpropagation in deep learning, where neural networks are a composition of many smaller differentiable operations. Exemplified by emerging languages like Julia, each function in a piece of scientific software is constructed to be differentiable, allowing for backpropagation or reverse mode automatic differentiation across entire program by applying the chain rule to each object. This paradigm shift has important ramifications for hardware, compilers, memory management, and floating point precision. Depending on if the *dP* framework uses static or compiled graphs, new compilers, operator overloading, callbacks, checkpointing schemes and a library of differentiable functions are required. Furthermore, even basic reading and writing to memory must be reinterpreted to be a differentiable operation where all memory locations are accessed instead of a single address. Incorporating design gradients will maximize the information extracted from exascale simulations, and provide many ancillary benefits to optimization, UQ, machine learning, surrogate modeling, model calibration, and sensitivity analysis. Having readily available design gradients is a key breakthrough as physics-informed deep learning can then be embedded seamlessly into scientific software and trained on the fly, providing a pathway to true hybrid methods that fuse data-driven and physics-informed computing[2].

In-situ Machine Learning
The growth in deploying Machine learning (ML) models to provide a range of insights into scientific data and simulations has profound implications on computing hardware if these are to be used efficiently in exascale systems. These challenges revolve around dealing with large increases in data storage requirements, enabling efficient utilization, and minimizing communication between compute,

visualization and ML nodes. In-situ transfer training during exascale simulations can adapt a pretrained machine learning (ML) model to recover critical insights from these large computational studies and decrease the need for saving large datasets. FPGAs and other specialized hardware inform by ML sampling techniques can enable efficient utilization by selecting the appropriate data to be streamed between nodes and processed rapidly at the compute endpoints. Mixed precision communication arithmetic from novel hardware, e.g., NVIDIA's A100, has the potential to decrease the number of bytes communicated throughout the HPC system.

Compressive ML frameworks, e.g., autoencoders, learn reduced embeddings of data into latent space representations by leveraging the so-called *information bottleneck*. Computing architectures mimicking this framework can reduce communication overhead by aggregating distributed data tensors to fewer and few hardware processing units at each layer in a network. Coupled with mixed precision arithmetic, in situ analyses can be performed alongside expensive exascale simulations. Compression for simulation checkpoint also improves simulation resiliency and benefits *dP* schemes[3]. Furthermore, these reduced representations enable scientific perceptual losses, computed through latent space data representations, that can further inform mixed precision analysis as well as dynamic sampling and visualization. Finally, dynamic surrogates developed on these reduced latent spaces can allow exascale insights to inform decision-making outside of an HPC environment or even on edge devices.

Visualization

Visualization should be integrated into the AI learning process to increase the understanding and trust of the ML surrogate and *dP* design gradients. We can enable critical insights into the model design by visualization of the states, sensitivities, and uncertainty, creating more explainable and interpretable models. In situ visualization has been at the core of exascale research to make data analysis and capture tractable from simulations that produce more data than can be moved or stored. Looking forward, we need to consider how we can integrate visualization techniques into ML-specific hardware, enabling us to visualize the reconstructions from the ML surrogates and leverage visualization to expose the underlying AI processes. As ML takes a more central role in these exascale simulations, it will be critical embed visualization that can characterize principal reasons for specific decisions, and uncertainty in crucial model behaviors.

Timeliness or Maturity

Scientific machine learning (SciML) and differentiable programming are rapidly maturing to the point that they can be unified into a coherent computational science framework that splits computing, learning, and visualizing across different hardware elements in specialized and heterogeneous architectures. Ready access to gradients of all terms in a model (both traditional PDE and modern data-driven terms) allows for an integrated simulation, training, and compression paradigm. Increasing the space of outputs from an exascale run to include trained ML models, gradients, and compressed representations can actually result in reduced computational demands to achieve the same scientific insight, while better utilizing increasingly heterogeneous and specialized hardware.

References

1. Innes, Mike, et al. "A differentiable programming system to bridge machine learning and scientific computing." *CoRR, abs/1907.07587* (2019).
2. Rackauckas, Christopher, et al. "Generalized Physics-Informed Learning through Language-Wide Differentiable Programming." *AAAI Spring Symposium: MLPS*. (2020).
3. Glaws, Andrew, et al. "Deep learning for in situ data compression of large turbulent flow simulations." Physical Review Fluids (2020).

Title: Neuromorphic Co-Design Towards Linking Emerging Materials with Algorithm Development

Authors: Shruti R. Kulkarni, Oak Ridge National Laboratory; Catherine D. Schuman, Oak Ridge National Laboratory; Maryam Parsa, Oak Ridge National Laboratory; J. Parker Mitchell, Oak Ridge National Laboratory; Prasanna Date, Oak Ridge National Laboratory.

Topic: architectures, modeling and simulation, emerging technologies

Challenge: Neuromorphic computing is one of the fastest growing beyond Moore computing paradigms that has shown its potential in implementing efficient systems for a diverse set of applications [1]. However, the main challenge faced by this research community is that there is no generic hardware platform or a stable software framework that can be used to design spike-based event-triggered systems. There have been several demonstrations of spike-based systems using the industry built neuromorphic platforms such as Loihi, TrueNorth, etc. [2], which are based on the digital CMOS technology. However, going forward, the technologies post Moore's era will focus more on platforms other than CMOS, such as resistive non-volatile memories (NVM), optoelectronics, bio-molecular membrane, etc., that have fundamentally different characteristics compared to the conventional implementations. Some of the characteristics of these emerging devices have been shown to be suitable for realizing neuromorphic algorithms [3-6]. However, with the existing programming paradigm, there is no way to leverage those characteristics to build a complete programmable neuromorphic system.

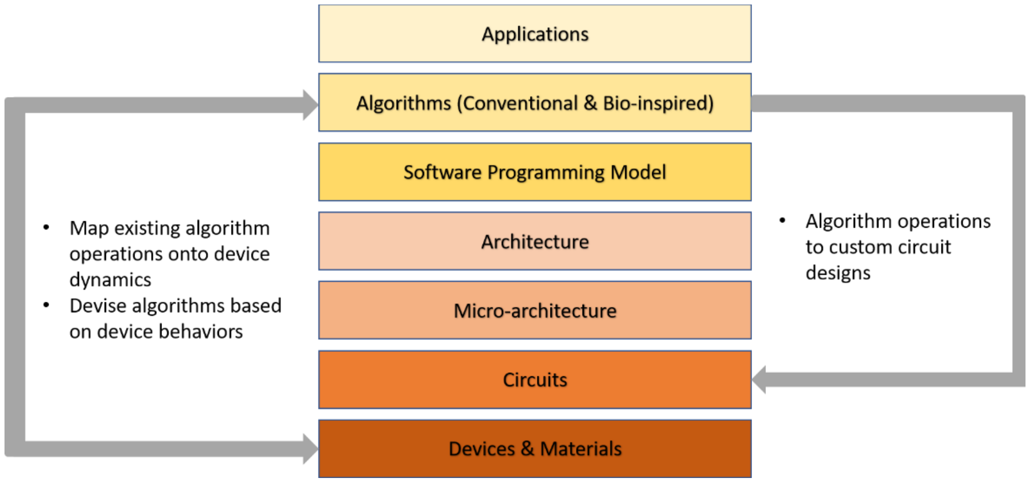

Applications
Algorithms (Conventional & Bio-inspired)
Software Programming Model
Architecture
Micro-architecture
Circuits
Devices & Materials
• Map existing algorithm operations onto device dynamics
• Devise algorithms based on device behaviors
• Algorithm operations to custom circuit designs

Figure 1. Opportunities to bridge different levels in the neuromorphic computing stack

Opportunities: As shown in Figure 1, while most co-design efforts follow a top down or bottom up approach along the computing stack, there are also a lot of unexplored areas between the non-adjacent levels in the stack. The question to be pursued is, can we build algorithms that take advantage of device and material physics, and can we map the operations in the existing algorithms onto the existing device physics? The main inspiration for this pursuit is the multiple demonstrations of the bio-inspired STDP phenomenon emulated in memristive devices and using that to build trainable neuromorphic systems [4,5]. Some spin-based devices have also been shown to emulate the dynamics of biological neurons [7]. Similarly, there are opportunities to engineer devices and circuits that mimic several signal processing and learning behaviors observed in the brain that can in-turn influence the process solving a given computation task [4]. In this regards, relatively less explored domain is the computing domain itself, where efforts like temporal computing machines, oscillator-based computing, etc. [8,9], that have strong dependence on the device behaviors, can offer improvements in performance and efficiency. Hence, there is an opportunity to develop programming models that are aware of the underlying device physics and circuit-level dynamics.

Timeliness or maturity: Most of the computation workloads of interest to the Department of Energy (DOE) are complex and require heterogeneous mix of accelerators in the HPC framework [10]. Neuromorphic computing has shown its potential in various HPC tasks such as scientific data processing, large-scale data modeling, etc. [1], which makes neuromorphic hardware a suitable platform for future heterogeneous HPC systems. To build a generic neuromorphic computer that takes advantage of broader range of characteristics of these emerging technologies, we need a robust software framework that allows visibility at different levels of the stack. This framework would need to be able to provide the algorithm developer the device/circuit level details and at the same time be scalable and flexible to accelerate the development process. The TENNLab neuromorphic framework is mature to a certain extent that it allows the algorithm developer several different back-end hardware to map and optimize algorithms [11,12]. However, it is not aware of the underlying device physics or circuit level features that can be used at higher levels. The efforts towards designing neuromorphic hardware are also less mature to date, with major focus being on designing digital CMOS based architecture. Hence, the co-design effort, towards building a scalable software programming model that is aware of the device and circuit dynamics is crucial. This could also allow a designer to parameterize these low-level design components for designing an optimal system.

References:

[1] Schuman, Catherine D., et al. "A survey of neuromorphic computing and neural networks in hardware." *arXiv preprint arXiv:1705.06963* (2017).

[2] Frady, E. Paxon, et al. "Neuromorphic Nearest Neighbor Search Using Intel's Pohoiki Springs." *Proceedings of the Neuro-inspired Computational Elements Workshop*. 2020.

[3] M. S. Hasan *et al*., "Biomimetic, Soft-Material Synapse for Neuromorphic Computing: from Device to Network," *2018 IEEE 13th Dallas Circuits and Systems Conference (DCAS)*, Dallas, TX, 2018, pp. 1-6, doi: 10.1109/DCAS.2018.8620187.

[4] Ielmini, Daniele. "Brain-inspired computing with resistive switching memory (RRAM): Devices, synapses and neural networks." *Microelectronic Engineering* 190 (2018): 44-53.

[5] Serrano-Gotarredona, Teresa, et al. "STDP and STDP variations with memristors for spiking neuromorphic learning systems." *Frontiers in neuroscience* 7 (2013): 2.

[6] **Kulkarni, Shruti R**., et al. "An on-chip learning accelerator for spiking neural networks using STT-RAM crossbar arrays." *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020.

[7] Sengupta, Abhronil, et al. "Spin orbit torque based electronic neuron." *Applied Physics Letters* 106.14 (2015): 143701.

[8] Csaba, Gyorgy, and Wolfgang Porod. "Coupled oscillators for computing: A review and perspective." *Applied Physics Reviews* 7.1 (2020): 011302.

[9] Madhavan, Advait, Matthew Daniels, and Mark Stiles. "Temporal State Machines: Using temporal memory to stitch time-based graph computations." *arXiv preprint arXiv:2009.14243* (2020).

[10] Vetter, Jeffery S., et al., "Extreme Heterogeneity 2018: Productive Computational Science in the Era of Extreme Heterogeneity", Report for DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity, Jan 2018.

[11] Plank, James S., et al. "The TENNLab exploratory neuromorphic computing framework." *IEEE Letters of the Computer Society* 1.2 (2018): 17-20.

[12] Schuman, Catherine D., et al. "A programming framework for neuromorphic systems with emerging technologies." *Proceedings of the 4th ACM International Conference on Nanoscale Computing and Communication*. 2017.

Challenges and Opportunities to Co-design Approximate Computing Scientific Applications[*]

Ignacio Laguna, Harshitha Menon, James Diffenderfer, Giorgis Georgakoudis
Daniel Osei-Kuffuor, Konstantinos Parasyris, Markus Schordan
Lawrence Livermore National Laboratory
{ilaguna, gopalakrishn1, diffenderfer2, georgakoudis1, oseikuffuor1, parasyris1, schordan1}@llnl.gov
Topics: hardware, applications

Introduction

As researchers in the computer architecture community have designed approximate hardware realizations for CPU cores, ALUs, memories, and accelerators, approximate computing (AC) has become an important emerging research area [1]. AC relies on the applications' ability to tolerate some loss of accuracy in the output results to trade-off for better performance and energy efficiency. Some approximate software techniques, such as code perforation via loop iteration skipping in image processing applications, have reported $3\times$ performance gains with less than 10% accuracy loss [2]. While AC methods have been demonstrated mostly outside of scientific applications, we believe that co-designing approximate hardware along with system software and applications could provide significant benefits to DOE scientific workloads.

Challenges

Challenge 1: Advanced Numerical Hardware Support

Low-Precision Arithmetic. Scientific applications rely heavily on floating-point arithmetic to perform numerical calculations, and as a result, co-designing floating-point hardware is of significant importance. Within the AC domain, lower-precision floating-point arithmetic is a promising technique to improve performance in exchange for increased error [3, 4]. As accelerators promote the use of lower-precision numerical units, such as half-precision units, the ability of applications to use these units efficiently is critical.
Low-Cost Co-Design of Efficient Type Conversion. Due to the use of lower-precision arithmetic, applications require frequent type conversions of variables and data structures. Hardware support for efficient (possibly low cost) type conversions (possibly in-memory) can significantly enable broader use of mixed-precision approaches. Also, help from the hardware and system software to represent data layouts efficiently would be handy.
Alternative Numerical Representations. There has been a recent development of several number representations [5] as an alternative to IEEE-754 floating-point. They particularly make efficient use of bits (e.g., wider dynamic range than IEEE with a similar number of bits), have less potential to incur overflow/underflow with fundamental operations (e.g., $+$, $*$), and enable the use of 32-bit representations in place of IEEE `double` precision without altering application's performance [6]. There is a challenge designing and or demonstrating the efficiency of alternate representations in hardware, such as Posits.

Challenge 2: Hardware-Level Compression

HPC applications use 64-bit (double) precision by default to prevent finite-precision round-off errors. However, not all the bits store valid information as many of these 64 bits represent error coming from rounding. As memory bandwidth and data transfer are becoming the dominating factors for an application's performance, researchers are exploring ways to reduce this bit's over-allocation. Compression is a suitable technique to address this challenge, which will reduce the amount of data and time for data transfer. A consorted effort between the programming model and the compression hardware unit can enable selective use of compression hardware for achieving performance while using the high-precision computing hardware when accuracy needs to be maintained. A significant challenge is the hardware implementation of floating-point compression/decompression algorithms that can significantly reduce the overhead by leveraging dedicated hardware units.

[*]This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-TR-819497).

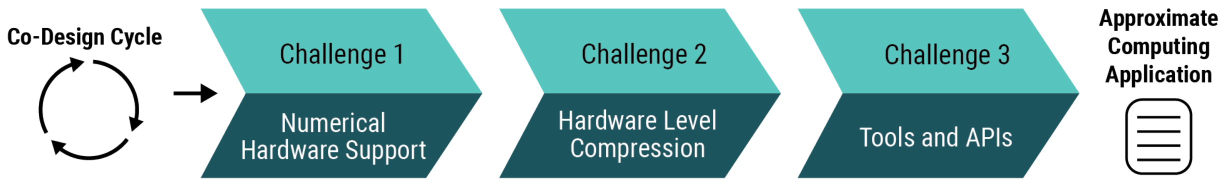

Co-Design Cycle
Challenge 1
Numerical Hardware Support
Challenge 2
Hardware Level Compression
Challenge 3
Tools and APIs
Approximate Computing Application

Figure 1: **Overview of the challenges to co-design AC applications. The challenges include (1) hardware support for multiple levels of floating-point precision and new numerical representations, (2) hardware-level support for numerical compression, and (3) Improved tools and compilers to apply multiple AC methods automatically.**

Challenge 3: Tools and APIs Co-design

Tools to Analyze Error Propagation. The ability to track down round-off errors and algorithmic-level errors is critical for adopting AC methods. A significant challenge is co-designing tools to enable error propagation at different granularity levels (e.g., function level versus code line level).
Co-Design of Compiler Support for AC. The hardware can support approximations using different techniques such as approximate adders and multipliers or aggressive voltage scaling. However, leaving to the programmer the task of developing software to specific AC hardware is not an effective solution. A crucial challenge is designing compiler support for automatic translation of certain computations to specific AC hardware.

Opportunity

Several new computer science trends suggest that addressing the above challenges now becomes feasible. These include:

1. The proliferation of open-source compilers and standardized intermediate representations (e.g., LLVM) that allow sophisticated static analysis of software and exploration of numerical variability in several architectures.

2. Co-designing AC applications and hardware is now more feasible than before as there is a trend on domain-specific accelerator technologies that can be modeled and simulated using DOE workloads more quickly.

3. Advances in the machine learning and AI fields that could be applied to speed up automated modeling, characterization, and search for optimal solutions across the large search space of floating-point and numerical requirements of DOE workloads.

Timeliness

Addressing the above challenges would have more impact on DOE scientific codes than before. There are many opportunities for return on investment in co-designing AC techniques for scientific applications. These returns fall into short- and long-term time scales. As short-term investments, we propose supporting the extension of existing research software tools for production use and integrating AC ideas from the non-HPC world into HPC applications. As long-term investments, we propose a research agenda to understand better using practical AC methods into production applications along with co-design activities with vendors to develop the required hardware support.

References

[1] S. Mittal, “A survey of techniques for approximate computing,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–33, 2016.

[2] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard, “Using code perforation to improve performance, reduce energy consumption, and respond to failures,” 2009.

[3] H. Menon, M. O. Lam, D. Osei-Kuffuor, M. Schordan, S. Lloyd, K. Mohror, and J. Hittinger, “ADAPT: algorithmic differentiation applied to floating-point precision tuning,” in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 614–626, IEEE, 2018.

[4] I. Laguna, P. C. Wood, R. Singh, and S. Bagchi, “GPUMixer: Performance-driven Floating-point Tuning for GPU Scientific Spplications,” in *International Conference on High Performance Computing*, pp. 227–246, Springer, 2019.

[5] P. Lindstrom, “Variable-radix coding of the reals,” in *2020 IEEE 27th Symposium on Computer Arithmetic (ARITH)*, pp. 111–116, 2020.

[6] J. Gustafson and I. Yonemoto, “Beating floating point at its own game: Posit arithmetic,” *Supercomputing Frontiers and Innovations*, vol. 4, pp. 71–86, 01 2017.

Hardware Partitioning Support for Discrete Resources

Jack Lange[1], Andrew Younge[2], Kevin Pedretti[2], and Ron Brightwell[2]

[1]University of Pittsburgh
[2]Sandia National Laboratories

1 Introduction

Future HPC reosurces are predicted to exhibit large degrees of resource heterogeneity in order to maintain performance improvement despite the loss of performance scaling in general purpose processors. While the form factor of these system architectures is uncertain, it is likely that they will exhibit a significant number of discrete components managed using traditional mechanisms based on a combination of kernel level device drivers and OS-bypass I/O channels. The viability of these approaches has been demonstrated on previous system generations and have a proven track record of providing performant access to both network and accelerator based devices. While conceptually this approach is amenable to a wide range of use cases, our past experiences with OS/R research has exposed a number of limitations. At the heart of the issue is that fact that current hardware and device driver architectures do not support sufficient flexibility in their usage models due to their reliance on complex and centralized control interfaces. While a significant amount of work has been made in the area of OS-Bypass I/O, unfortunately this has almost entirely focused on the data plane and not the control plane. The end result is that while it is possible to create OS-Bypass I/O channels in an application context, the interface to do so requires that the application adheres to a set of assumptions in the driver that limits how applications can be deployed on a system.

The limitations imposed by the driver based interfaces has implications for novel application architectures as well as the viability for application deployment techniques. For instance, these driver requirements make it very difficult to employ alternative OS/R environments such as virtual machines [1], co-kernels [2], user level driver frameworks, and containerization. In our experience the three most common issues are the following: centralized address space management, non-partitionable control interfaces, and firmware/driver version dependencies. Each of these problems arises from the selection of a monolithic architectural approach that centralizes device management into a single complex driver architecture. This allows the internalization of many fundamental environmental concepts exported by the driver's hosting OS kernel, which in turn propagates those assumptions up to any application that wishes to access the device.

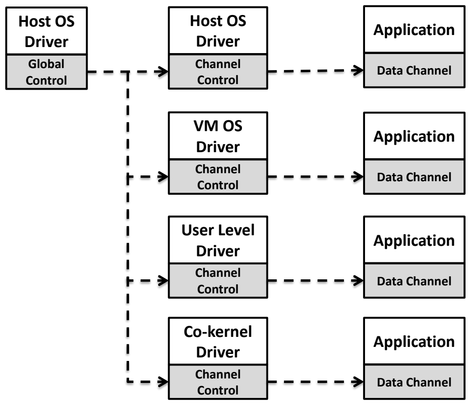


Figure 1: A partitionable I/O interface

Our position is that in order to allow future HPC systems to adapt to novel workloads and changing deployment mechanisms, the system must support flexible resource management at the hardware and driver layer. Furthermore, this support needs to be developed using a co-design based approach so it can incorporate feedback from the OS/R community to specify the interface requirements necessary to support a wide range of software architectures. We contend that future discrete hardware resources should support resource partitioning in terms of both functionality and performance, or, in simpler terms, that existing OS-Bypass mechanisms should be extended to the control plane. At a high level breakdown of the architecture is shown in Figure 1. What has been lacking is the development of portable hardware and driver interfaces that *enable* resource partitioning in a flexible and portable way that does not require large development effort and system support due to driver complexity. Ideally, device partitioning would enable flexible OS/R environments without extensive driver support or complex subsystems needed by specialized driver components.

2 Challenges

We claim that a co-design effort is necessary in order to ensure that future discrete hardware resources support a wide range of potential OS/R configurations and environments. In order to achieve this it will be necessary to support dynamic resource partitioning such that multiple independent driver instances can simultaneously manage partitions of a given hardware resource. These driver instances would then be deployed in multiple potential configurations including virtual machines, co-kernels, and user level I/O subsystems. To achieve these requirements, a number of challenges must be addressed including distributed memory mapping and address space management mechanisms, partitionable control channels, and driver/firmware version compatibility.

Memory mapping is the single largest obstacle to resource partitioning seen in the last generation of HPC I/O devices. In particular this problem has presented itself in the high performance network devices that are the cornerstone for large scale high-end supercomputers. Fortunately the solution to this problem is already seen in the form of IOMMUs, that allow IO address space virtualization. IOMMUs are readily available on commodity systems and have been adapted to serve many roles in modern device driver frameworks. While simply integrating IOMMUs would go a long way to solving the partitioning issue, there are a number of challenges that remain. For instance, what granularity of resource partition should the IOMMU function at? Modern IOMMUs virtualize the address space at the PCI end-point granularity using SRIOV, which places a dependency on PCI based hardware and might be too course grained for future devices.

The second challenge that needs to be addressed is the design of the control interfaces to support partitioned operation. This challenge presents a significant opportunity for a co-design effort, since it needs to balance exposing enough functionality to allow efficient operation while not being so complex as to require large and extensive driver architectures. Ideally, this interface would allow small and lightweight drivers that can be easily developed and embedded into a variety of OS/R environments. To fully address this challenge will require insight into both hardware and system software capabilities, requirements, and goals.

The third challenge concerns version compatibility across software device drivers and hardware firmware. During the operational lifecycle of HPC resources, it is common for the system to undergo a number of software and firmware updates to improve performance and address bugs. Oftentimes these updates consist of simultaneous changes to the firmware and driver stack which need to be applied in unison, otherwise the hardware will often fail to function correctly. The result of this places considerable difficulty in supporting OS/R flexibility because it requires ensuring that each potential OS/R contains an updated driver version. It is also not uncommon for these version changes to propagate upwards to higher level libraries, creating problems for portable software packaging and container based systems. Ensuring cross version compatibility will be a requirement for any system hoping to deploy a distributed driver architecture. In addition the architecture will need to ensure that an OS/R with an outdated driver is able to operate concurrently with another version of the driver in a separate OS/R. Addressing this challenge will require the development of interfaces and driver architectures that minimize the overhead of tracking changes to firmware versions as well as ensuring compatibility when multiple driver versions are active in the system.

3 Opportunity

As the move towards heterogeneity increases in the coming years, many new hardware architectures will be developed and integrated into existing systems. These efforts present an opportunity to rethink OS/R integration of discrete resources and build in mechanisms to enable OS/R flexibility from the beginning of the design phase. In addition, many new advances in hardware prototyping are coming online with widely available FPGA architectures capable of supporting prototype hardware designs (i.e. RISC-V). This presents an opportunity to rapidly distribute design prototypes to OS/R researchers without the need for complex simulation environments or expensive hardware fabrication.

References

[1] LANGE, J., PEDRETTI, K., HUDSON, T., DINDA, P., CUI, Z., XIA, L., BRIDGES, P., GOCKE, A., JACONETTE, S., LEVENHAGEN, M., AND BRIGHTWELL, R. Palacios and kitten: New high performance operating systems for scalable virtualized and native supercomputing. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium* (April 2010).

[2] OUYANG, J., KOCOLOSKI, B., LANGE, J., AND PEDRETTI, K. Achieving Performance Isolation with Lightweight Co-Kernels. In *Proc. 24th International ACM Symposium on High Performance Distributed Computing (HPDC)* (2015).

Codesign for the Masses

Cannada Lewis (canlewi@sandia.gov)[*], Simon Hammond, and Jeremiah Wilke

February 16, 2021

1 Topic

(**Architecture & Codesign Methologies**) In this position paper we will address challenges and opportunities relating to the design and codesign of application specific circuits. Given our background as computational scientists, our perspective is from the viewpoint of a highly motivated application developer as opposed to career computer architects.

2 Challenge

It is no secret that major breakthroughs in computational sciences are often reexaminations of previously proposed ideas, in the context of either new algorithmic techniques or new developments in computational capability. In the last 15 years adoption of GPUs and their programming models [9] has given new life to old ideas in a plethora of disciplines from quantum chemistry [4, 13] to—most famously—deep learning [12]. These are examples of approaches that have won *the hardware lottery*[6], where technological developments have favored certain application and algorithmic choices at the expense of others; often irrespective of their theoretical merits. What if we could partially solve the issue of *the hardware lottery*? Because it is not possible to fully decouple modern computational sciences from the specifics of hardware design, what if we could give the application developers better tools to design and understand hardware, instead of forcing them to pick between a few mostly general purpose designs? This is the very essence of codesign.

The current paradigm forces applications to optimize for either a specific architecture or a narrow band of similar architectures, none of which were specifically tailored to the exact problems the application faces. As extreme heterogeneity becomes the norm, between CPUs, GPUs, FPGAs, reconfigurable hardware and combinations of all of them, ideally application developers would have an easy way to design and iterate on their own accelerator designs. This goal presents great challenges though; while modern approaches provide significant quality of life improvements for hardware designers, creating an application specific accelerator is still very much the domain of hardware experts. For example, it is a great achievement that open source CPU cores require only thousands of lines of code[15], albeit in unfamiliar (to application developers) hardware description languages (HDL), such as Verilog, VHDL, pyRTL, or Chisel. But, their design at the HDL level is still out of reach for all but the most dedicated application programmers.

The main alternative to HDL is high level synthesis (HLS) of programming languages like C and C++. HLS of C++ can require less effort than a similar HDL design [11], but it is not a panacea; much like high-performance linear algebra routines, the code required for performant HLS rarely resembles the initial high level implementation and demands the developer have a deep understanding of how the code will be synthesized. All is not lost though, reference [11] suggests that there are quality of life and time to solution improvements available via HLS.

Finally, there is the issue of hardware simulation, the efficient and accurate modelling of general purpose accelerators is a non-trivial task. Many tools exist in this space, but they can be vendor dependent or difficult to use for novices. Work must be done to simplify the design, simulate, and iterate cycle if we want to empower application developers to embrace accelerator design. Other position papers will most likely address this topic in more depth.

The challenges to bring hardware design to the masses (application developers) are great. Computational scientists do not have HDL experience, hardware simulation and the use of FPGAs is unfamiliar, and they may be sceptical of the expected value their applications would ultimately receive. But, there also exists a great opportunity to achieve success that could only come via codesign of algorithms and hardware. Ultimately to truly enable codesign, we require new approaches via software engineering, programing models, and compilers to bridge the gap between software and hardware engineering.

3 Opportunity

The end of Dennard Scaling and the inevitable demise of Moore's law dictate that future performance gains for scientific applications are likely to come from a combination of software engineering, algorithmic improvements, and hardware specialization [8]. In many cases these are multiplicative, which itself

[*]Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

speaks to the value of codesign. The oppportunity for DOE is to lead the creation of a suite of tools that bring these kind of improvements to the broadest possible user base—not just the determined few which has arguably been the situation for some large-scale HPC architectures. The result could be vast increases in scientific delivery and broad families of codesigned accelerators or SoCs for entire computing communities. The piece that is missing to enable this future is the tooling to give developers the ability to explore the space of interactions between algorithms and hardware. One need look no farther than deep learning to see the transformative effects that having access to domain specific hardware can achieve.

By providing the tools required to enable application developers to steer the process of codesign themselves, DOE will be able to initiate a new era of computational science that mirrors the explosion of successes in deep learning. Performance improvements of more than 30X are not out of reach [7], but achieveing this will require deeply coupled codesign of the hardware and software. No one knows better than the domain scientists where the opportunities for codesign lie, but to date, they have not had the technology to make hardware exploitation feasible. As we progress into a post Moore's era this needs to change.

4 Timeliness

We have precedent that application developers are willing to explore the interplay of algorithms and hardware, as evidenced by the rise of parallel programming models such as CUDA [9], SYCL [1] and Kokkos [5]. Adopting these models for parallel programing requires significant investment from applications, but in the end the performance gains are often worth it. Before these libraries and languages existed, it was simply too difficult for the vast majority of application developers to attempt to target GPUs, limiting GPU adoption. Currently, software hardware codesign is in a similar state, but the tools are beginning to become available to allow scientist to explore the hardware landscape. Similarly, to how CUDA helped to bring GPU programming to the masses new compiler tools and domain specific languages for HLS from both academia and industry [2, 3, 14, 10] will help empower application developers to explore domain specific architectures for their problems. But this work is still in its infancy and DOE should invest now, both to help develop the tools and abstractions needed to make high level codesign a reality and also to connect application developers with these tools. In some sense we have the opportunity to codesign the future generation of codesign tools.

References

[1] https://www.khronos.org/sycl/.

[2] https://github.com/google/xls/.

[3] https://github.com/llvm/circt.

[4] M. Dupuis, J. Rys, and H. F. King, *Evaluation of molecular integrals over gaussian basis functions*, The Journal of Chemical Physics, 65 (1976), pp. 111–116.

[5] H. C. Edwards, C. R. Trott, and D. Sunderland, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202 – 3216. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[6] S. Hooker, *The hardware lottery*, 2020.

[7] N. P. Jouppi et al., *In-datacenter performance analysis of a tensor processing unit*, 2017.

[8] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, *There's plenty of room at the top: What will drive computer performance after moore's law?*, Science, 368 (2020).

[9] J. Nickolls, I. Buck, M. Garland, and K. Skadron, *Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for?*, Queue, 6 (2008), p. 40–53.

[10] R. Nigam et al., *Predictable accelerator design with time-sensitive affine types*, in Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2020, New York, NY, USA, 2020, Association for Computing Machinery, p. 393–407.

[11] J. Noguera, S. Neuendorffer, K. Vissers, and C. Dick, *Wireless mimo sphere detector implemented in fpga*, Xcell Journal, 74 (2011), pp. 38–45.

[12] R. Raina, A. Madhavan, and A. Y. Ng, *Large-scale deep unsupervised learning using graphics processors*, in Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, 2009, Association for Computing Machinery, p. 873–880.

[13] I. S. Ufimtsev and T. J. Martínez, *Quantum chemistry on graphical processing units. 1. strategies for two-electron integral evaluation*, Journal of Chemical Theory and Computation, 4 (2008), pp. 222–231. PMID: 26620654.

[14] J. Wang, L. Guo, and J. Cong, *Autosa: A polyhedral compiler for high-performance systolic arrays on fpga*, in Proceedings of the 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2021.

[15] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, *Sonicboom: The 3rd generation berkeley out-of-order machine*, (2020).

A Co-designed Data Layout Transformation Infrastructure for Scientific Computing

Lingda Li
Computational Science Initiative, Brookhaven National Laboratory, lli@bnl.gov

Topic: architectures, programming systems, modeling and simulation.

I. Challenge

The spectrum of high-performance computing (HPC) applications that are memory bound rather than compute bound include not only a broad swath of traditional scientific simulation applications, but also emerging machine learning (ML) and data analytics applications. To optimize their performance, data layout organization optimization is a key ingredient. Optimized data layout alone can significantly improve data locality and reduce unnecessary data transfer between memory hierarchies. It also opens the space of algorithmic optimizations.

However, optimizing data layout is a complex endeavor due to the interplay between heterogeneous hardware and application development. Architectural trends suggest the increment of both accelerator number and diversity in future HPC systems. Different accelerators have specific memory designs. Therefore, they prefer varied data access patterns and layouts. For instance, general CPUs require data layout to be optimized for the cache hierarchy (i.e., locality). GPUs prefer coalesced accesses across contiguous threads to leverage its memory bandwidth while mostly ignoring the cache performance due to limited cache capacity per thread. Static random access memory (SRAM)-based accelerators (e.g., Cerebras) prefer compact data layouts due to limited memory capacity, and processors with persistent memory will benefit from data layouts that reduce the number of fine-grained stores. There also is memory heterogeneity existing in a single accelerator. For example, a GPU's main memory prefers coalesced accesses, while its scratchpad memory (i.e., shared memory in NVIDIA GPUs) does not require it but is sensitive to bank conflicts.

There are also great challenges for application developers to make use of data layout transformation. Because there are limited data layout supports in available HPC programming models and they are tightly coupled with computation, most existing HPC applications feature hard-coded data layouts. Hard-coded layout contributes a huge portion to the performance portability hurdle as computation is often organized around the data layout. To facilitate the utilization of data layout transformation in HPC applications, corresponding innovations of programming models and software tools are required. If there is a common programming model to specify the data layouts and required transformations, performance portability can be greatly improved and programming efforts can be saved.

In summary, the data layout implication of future HPC includes: 1) a general data representation and transformation framework will be required because manual data layout optimization will not be a viable option and impedes performance portability, 2) such a framework should support a broader range of transformations than what are presently available, 3) more frequent data layout transformation will be required as data flow from one memory to another (either inter- or intra-accelerator), and 4) more performant data layout transformation will be required because of the increasing demand.

II. Opportunity

To address these challenges, a universal data layout transformation infrastructure (DLTI) would be both valuable and useful. Furthermore, the resultant infrastructure should have the following properties:

- Flexible and User Friendly Programming Interface. The desired infrastructure should support a wide range of data layout transformations required by scientific applications and be flexible enough to support future extensions. It should also provide friendly interfaces and be easily integrated into existing codebases.
- Portability across Processors and Memory Devices. Because different processors and memory devices have distinct data layout and transfer preferences, the desired system should consider these diverse requirements to ensure its portability across various HPC systems.
- Performance and Efficiency. Performant and efficient data layout transformation is crucial because it must be performed frequently and dynamically.
- Modeling and Tuning Support. The system should provide interfaces for users/tools to reason about the benefits and overheads of particular transformations.

A thoughtful co-design approach is key to the success of DLTI. Particularly, DLTI includes three components: a hardware transformation accelerator, a data layout programming model, and an optimized compiler. Achieving the flexibility, portability, performance, and modeling goals requires to carefully co-design all three.

Transformation Primitive. Scientific applications often organize data in the form of multidimensional arrays. Across dimensions, transposition, packing, splitting, and tiling are commonly needed transformations. Transposition changes the order of dimensions. Packing refers to combining multiple dimensions, while splitting is the opposite operation to split one dimension into multiple ones. Tiling blocks the array across multiple dimensions to improve locality. Data shuffling within a dimension is also required in some scenarios. A set of primitives should be defined to support all these transformations as the hardware-software interface.

Computer Architecture. The desired accelerator should support all transformation primitives. Design-wise, such accelerators should locate near the source and/or destination memory to be efficient, as well as prefer to have low area and power footprints. Complexity of the architecture, such as the memory hierarchy, should be managed transparently in the hardware to reduce the burden on programmers.

Programming Model. Some recent HPC programming models have introduced data abstractions, and it is more economical to embed the data transformation functionality into them to leverage existing work. For instance, Kokkos [3] introduces `View` as an abstract representation of multidimensional array and supports to specify its memory layout statically. However, it does not allow dynamic layout transformation and is limited to C++ applications. Another example is OpenMP that enables data mapping between different accelerators [1]. The OpenMP offloading model is a proper interface for data layout transformation, and existing data mapping features, such as OpenMP `map` clauses and user-defined data mapper, can be extended to support transformation.

Compiler. LLVM compiler infrastructure provides an excellent framework to bridge the gap between the data layout transformation accelerator and programming model [4]. The recently introduced multi-level intermediate representation (MLIR) [5] can help optimize these operations in a high level to generate efficient hardware transformation instructions.

The design of DLTI is inspired by the ML ecosystem, which is a great co-design example. Programming model-wise, domain-specific languages, such as TensorFlow and PyTorch, are defined for users to construct ML models easily. At the hardware level, various accelerators have been designed, such as TPU and GraphCore. To bridge the gap, compiler technology is widely applied on performance optimization, e.g., by combining computation operations and optimizing data flow. As a result, ML applications can often achieve sustainable performance close to the hardware peak without much effort from end users.

III. Timeliness or Maturity

Timeliness. Considerable research has been done for data layout transformations in the software space [2], [7], [9]. They tend to have relatively low performance as software-only solutions and are not adequate for handling challenges introduced by increasing heterogeneity. Recently, several hardware engines have been proposed that can perform some transformations, such as scatter/gather [8], [6]. They cannot be easily utilized by applications and also need to support more general data layout transformations.

In recent years, the cost of specialized hardware has decreased significantly, while architectural diversity has increased manifold thanks to open source hardware and standard interfaces. In addition, the availability of high-performance open source compilers, such as LLVM, makes it easier to capture the boundary between software and hardware for optimal co-design. The confluence and availability of these requisite links points to the timeliness of a co-designed DLTI.

Feasibility. Although it involves considerable effort to co-design a system for the scientific computing community, which is much smaller and has more diverse requirements, it should be feasible to do so for data layout transformation. Initially, it is a more constrained scope and will require much less effort. Secondly, most scientific applications need data layout transformations. Thus, it is worthy to co-design such a system that could benefit the broader scientific community.

Potential Impact. The resulting data transformation system will have significant impacts on the development and optimization of scientific applications. First, thanks to the hardware-accelerated transformation engine, many data-layout-related algorithmic optimizations can be explored and applied, which are not beneficial presently with expensive software-based data layout transformations. Second, application developers will be able to specify and transform between various data layouts with ease, using the proposed programming model. Third, such a system will facilitate the performance portability across diverse HPC systems that have distinct data layout preferences.

References

[1] "OpenMP 4.5 specifications," 2015.

[2] C. Ding and K. Kennedy, "Improving cache performance in dynamic applications through data and computation reorganization at run time," in *Proceedings of the ACM SIGPLAN 1999 Conference on Programming Language Design and Implementation*, ser. PLDI '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 229–241.

[3] H. C. Edwards, C. R. Trott, and D. Sunderland, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," *Journal of Parallel and Distributed Computing*, vol. 74, no. 12, pp. 3202–3216, 2014.

[4] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization*, ser. CGO '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 75–.

[5] C. Lattner, J. Pienaar, M. Amini, U. Bondhugula, R. Riddle, A. Cohen, T. Shpeisman, A. Davis, N. Vasilache, and O. Zinenko, "Mlir: A compiler infrastructure for the end of moore's law," *arXiv preprint arXiv:2002.11054*, 2020.

[6] S. Lloyd and M. Gokhale, "In-memory data rearrangement for irregular, data-intensive computing," *Computer*, vol. 48, no. 8, pp. 18–25, 2015.

[7] I. Sung, J. A. Stratton, and W. W. Hwu, "Data layout transformation exploiting memory-level parallelism in structured grid many-core applications," in *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2010, pp. 513–522.

[8] T. Thanh-Hoang, A. Shambayati, and A. A. Chien, "A data layout transformation (dlt) accelerator: Architectural support for data movement optimization in accelerated-centric heterogeneous systems," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 1489–1492.

[9] T. Zhao, P. Basu, S. Williams, M. Hall, and H. Johansen, "Exploiting reuse and vectorization in blocked stencil computations on cpus and gpus," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019.

A Common Machine-Readable Vocabulary and Knowledge Base Supporting Multiple Programming Models

Chunhua Liao
Lawrence Livermore National Laboratory, liao6@llnl.gov

Topic: programming systems, emerging technologies, codesign methodologies

Challenge: Designing and implementing HPC programming systems requires a wide range of diverse knowledge about multiple layers of software/hardware stack, including science disciplines, application domains, compilers, runtime systems and hardware. For example, understanding and communicating high-level domain and application semantics related to data dependence can enable a wide range of compiler and runtime optimizations aimed for improving parallelism and reducing unnecessary data movement. However, such knowledge has been either implicit or expressed in an ad-hoc manner in each domain. People from difficult domains may not even talk in the same vocabulary for semantically identical concepts. Different programming systems and compilers have to develop different interfaces or annotations to redundantly encode the same information. The ad-hoc management of heterogeneous information from multiple layers of HPC stack causes unnecessary burden for both developers and users of the programming systems. It also makes co-design of HPC systems difficult in general.

Opportunity: What if as a community, we can collectively and systematically accumulate, share, and reuse formally defined, machine-readable, and human-friendly knowledge across multiple layers of software/hardware stack? If such a vision is realized, it will enable stakeholders from different backgrounds to easily collaborate. Different software and hardware components in the HPC stack will also be interoperable. A variety of programming systems can be designed and tested on top of a reusable knowledge base of different science disciplines, application domains, software packages, and hardware platforms.

Advances in the knowledge representation community may already generate sufficient techniques and tools to help HPC researchers create a formal and machine-readable knowledge base across different layers of software/hardware stack. For example, a starting point for any group of people to collaborate is to have a common vocabulary, taxonomy and properties to describe one or more domains. Ontology techniques define a systematic approach to capture and represent concepts, instances and their relations for a domain. The Resource Description Framework (RDF) data model[6] encodes knowledge in the form of subject-predicate-object expressions. These techniques are very relevant for building HPC programming systems since a key driven factor for HPC optimization is the extracted software and hardware properties.

On the other hand, this vision of course has its unique challenges. For one, no single person can understand all the domains of HPC. It requires organized efforts to start from a smaller domain then incrementally aggregate smaller vocabularies and knowledge bases into more comprehensive ones. Another challenge is knowledge engineering, including domain knowledge gathering and verification, is still a labor intensive process.

Timeliness or maturity:

Knowledge representations and knowledge bases have been studied for decades. They have increasingly been used in many domains as the related techniques such as Web Ontology Language[4], Resource Description Framework[6], and JSON-LD[5] mature. For example, Schema.org[2] is a collaborative vocabulary started by Google, Microsoft, Yahoo etc. to annotate the Internet with structured data. Wikidata[3] is another collaboratively edited multilingual knowledge graph managed by the Wikimedia Foundation (who runs Wikipedia). More recently, Yago 4[1] builds one of the most comprehensive knowledge bases on top of crowd-sourced wikipedia articles. Linked Open Vocabularies (LOV)[7] gathers more than 700 vocabularies from different domains and provides popularity statistics and a searchable interface for users to find the right choices.

With all the aforementioned efforts going on, It is a good time for the HPC community to investigate these techniques and build our own common vocabulary and shared knowledge base to facilitate co-design of programming systems and even the entire HPC systems.

References

1. Tanon, Thomas Pellissier, Gerhard Weikum, and Fabian Suchanek. "Yago 4: A reasonable knowledge base." In European Semantic Web Conference, pp. 583-596. Springer, Cham, 2020.
2. Guha, Ramanathan V., Dan Brickley, and Steve Macbeth. "Schema. org: evolution of structured data on the web." Communications of the ACM 59, no. 2 (2016): 44-51.
3. Vrandečić, Denny, and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase." Communications of the ACM 57, no. 10 (2014): 78-85.
4. Motik, Boris, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. "OWL 2 web ontology language profiles." W3C recommendation 27 (2009): 61.
5. Sporny, Manu, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. "JSON-LD 1.0." W3C recommendation 16 (2014): 41.
6. Pan, Jeff Z. "Resource Description Framework." In Handbook on ontologies, pp. 71-90. Springer, Berlin, Heidelberg, 2009.
7. Vandenbussche, Pierre-Yves, Ghislain A. Atemezing, María Poveda-Villalón, and Bernard Vatant. "Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web." Semantic Web 8, no. 3 (2017): 437-452.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. The work was also supported by the U.S. DOE Advanced Scientific Computing Research. LLNL-ABS-810981

Leveraging open source simulators for HPC codesign

Bobby R. Bruce[1], Jason Lowe-Power[1], and Matthew D. Sinclair[2]

[1] *University of California, Davis, {bbruce, jlowepower}@ucdavis.edu*
[2] *University of Wisconsin-Madison, sinclair@cs.wisc.edu*

Topics architecture, modeling and simulation, codesign methodologies

Challenge *We need agile iteration of software and hardware changes in order to achieve hardware-software codesign. As such, we require accurate, modular, and high speed simulators to enable this iteration.* Current simulation techniques are either too high level, too inflexible, or too slow to enable codesign for exascale and post-exascale HPC architectures.

As Moore's Law continues to fade, future performance improvements are going to need to rely more on codesign between multiple layers of the hardware-software stack [3]. Codesign, by definition, involves modifying many levels of the hardware-software stack, such as the architecture, the application runtime, the operating system, compilers, computer networks, and the batch scheduler. Traditionally, many of these layers have relied on proprietary, closed source components, which makes research on next-generation solutions difficult, if not impossible. There is a new opportunity for researching codesign with modern open source platforms such as RISC-V, LLVM, and Linux. These platforms make it possible for researchers to modify *every level* of this stack. However, when making these modifications, researchers need a way to predict their performance impact before undergoing the long and costly process of engineering real-world hardware. We therefore need simulation and modeling which can capture these *full system* effects as the benefits of codesign can only come from understanding the underlying interactions between different parts of the hardware-software stack.

Unfortunately, current modeling and simulation techniques suffer from poor performance, insufficient detail to capture cross-stack optimizations, and are difficult to modify and extend. Full-system cycle-level simulators like gem5 [4] can capture the hardware-software interactions since they support all levels of the stack including the OS, runtime, and unmodified software, but cycle-level simulators are slow (at least 10,000× slowdown) and incapable of simulating large-scale applications in a reasonable time frame. Other systems such as SST [5] scale to modeling full HPC centers, but do so at the cost of modeling detail, often falling back on trace-based or analytic models which do not have enough detail to model important codesign components like the application runtime. Finally, fast FPGA-accelerated simulators like FireSim [2] are capable of running full-systems and mitigate some of the performance issues with software-only simulation systems, but are inflexible, often requiring fully implemented RTL designs. These FPGA-based simulation systems can be useful for tweaking current designs, but are not flexible enough to explore entirely novel architectural designs (e.g., new accelerators, which HPC systems are widely adopting).

Opportunity We need to develop new modeling and simulation techniques to understand the performance impact of novel codesigns at an HPC-center scale and that provide accurate performance predictions in tolerable timeframes. Since no single methodology can provide the required accuracy, detail, or scale, we believe the path forward is through "multi-fidelity simulation." We aim to leverage multi-fidelity simulation in two axes: over time and over space. We envision a unified simulation framework which integrates the best in class models—the fastest with good high-level behavioral accuracy and the most detailed with cycle-level accuracy—in the same simulation instance in both time and space.

Leveraging different fidelity simulation *over time* has been a popular technique to improve the performance and energy of detailed simulation techniques. Examples include SimPoints, SMARTS, and the BarrierPoint sampling methodologies. Recently, researchers have leveraged analytical models to improve cache warmup time which dominates most of the sampled simulation techniques, and used hardware virtualization platforms to fast-forward simulators at native speed (i.e., with no slowdown) to the region of interest or the sampling location. Although many of these techniques have been published, few have been integrated into any widely-used simulation system in a straightforwardly useable fashion. Thus, there is an opportunity to provide codesigners with easy-to-use simulation frameworks which natively support multi-fidelity simulation over time.

While multi-fidelity simulation over time has been previously explored, combining different fidelity models *at the same time* in a single simulation is a new research direction—which we define as multi-fidelity over space. For instance, to simulate an entire compute rack running one application, it is likely unnecessary to simulate all systems at a high fidelity (e.g., cycle level). Instead, we can simulate one or two systems at a cycle level and use higher level models (e.g., trace based or analytical models) for the other systems in the rack. This technique can also be applied within a single node. For instance, we can simulate the processors with a high-level "simple" out-of-order model which instead of providing detailed representation of every structure, provides simple configuration options for components like the instruction window and commit width, while at the same time uses a cycle-level memory system design to investigate the impact of software-cache coherence codesign.

To enable multi-fidelity simulation, we must have a modular, composable, and standardized simulation framework. This framework will allow codesigners to combine different fidelity models in both time and space. There is an opportunity to develop a simulator backplane which will allow modular simulators to share "best in class" models which may operate at different fidelities. We believe that building off of today's modular simulation infrastructures such as gem5 or SST is a viable direction to create this simulator backplane. There is evidence that these simulator systems can integrate other models such as gem5-gpu which integrates GPGPU-Sim with gem5, Emerald which integrates graphics simulation with gem5, gem5-Aladdin and gem5-SALAM which integrate auto-generated accelerators with gem5, and many others. Similarly, SST Elements provides a broad set of simulator integrations for SST. However, these integrated models are not readily available in a centralized easy to use and access location. Additionally, we need to improve the usability and composability of current simulation platforms and expand the availability of multi-fidelity models.

Timeliness or Maturity It is now assumed that future gains in performance are going to come above the level of silicon [3] and performance improvements will increasingly come from hardware-software codesign. Now is the time to build the vital infrastructure to make this possible. This infrastructure will, in large part, consist of simulators which can deliver reliable simulations in acceptable time scales. Without such an infrastructure, it will be extremely difficult for future generations of system designers to develop their ideas, without resorting to time consuming RTL designs or relatively expensive chip tapeouts. Thus, developing this next generation simulation infrastructure will enable system developers to rapidly prototype, test, and iterate on their ideas while being confident that the infrastructure is representative of modern HPC systems.

Work has already began in this area. In 2020 gem5 released it's first stable version as part of an ongoing effort to improve the stability of the project. Documented APIs have been added with guarantees of stability between versions, thereby making a start on the engineering effort necessary for a common architecture backend which other tools may integrate with. This stability will help foster and support a common, community-based infrastructure. Getting value out of this backend will require forming collaborations between open source architecture projects, and ensuring common standards and processes are agreed upon. Moreover, developing such an infrastructure with common standards and processes will enable additional projects, such as those that enable accelerated simulation of RTL [1], to easily integrate into the infrastructure. While difficult to achieve, such an effort would be of immense benefit to researchers investigating codesign and beyond.

References

[1] S. Beamer. A Case for Accelerating Software RTL Simulation. *IEEE Micro*, 40(4):112–119, 2020.

[2] S. Karandikar et al. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 29–42. IEEE, 2018.

[3] C. E. Leiserson et al. Theres plenty of room at the Top: What will drive computer performance after Moores law? *Science*, 368(6495), 2020.

[4] J. Lowe-Power et al. The gem5 simulator: Version 20.0+. *arXiv preprint arXiv:2007.03152*, 2020.

[5] A. F. Rodrigues et al. The structural simulation toolkit. *SIGMETRICS Perform. Eval. Rev.*, 38(4):3742, Mar. 2011.

APPLICATION-DRIVEN CODESIGN FOR ONLINE AND CONTINUAL LEARNING USING NEUROMORPHIC ARCHITECTURES

SANDEEP MADIREDDY*, ANGEL YANGUAS-GIL, PRASANNA BALAPRAKASH

Argonne National Laboratory

Topics: Architectures, Applications, Emerging technologies, Codesign methodologies

Challenge: Machine learning, in particular, deep learning methods are increasingly being used in scientific applications [1] in addition to the more ubiquitous non-scientific data. These models are traditionally trained on large computing clusters and deployed elsewhere for inference. Once trained, these inference models are primarily used for prediction-only tasks and often they cannot adapt to the changing environment. This limits the scope and usability of the model since the new data might differ significantly from the statistical properties of the training data. To this end, online continual learning addresses the scenario where a system has to learn and process data that are streamed continuously and mitigate catastrophic forgetting. Many real-life scientific applications such as adaptive data collection at the edge, temporal sequence mining for anomaly detection in manufacturing, control and continual reinforcement learning for chemical synthesis or autonomous steering of manufacturing processes can benefit from online continual learning. However, in some cases, such as on-chip AI at the edge, additional considerations such as resource limitations in the hardware, data privacy, or data security, limit the ability to implement these algorithms at scale. The challenges in the online continual learning scenario are therefore two-fold: first, resource constraints impose limitations to algorithms, the extreme case being learning just from streaming data with no control of when and how this data is presented. Second, systems need to learn new tasks from data that may have not been seen before deployment.

Specialized neuromorphic hardware that take inspiration from principles of brain computation, have grown in popularity as they promise distributed and power efficient computation on silicon chips in addition to application-specific customization and continual learning. Numerous neuromorphic architectures either based on analog, digital or mixed-signal approaches have been successfully proposed and demonstrated emulation of both supervised and reinforcement learning environments, hence provides a promising ground to implement application-specific customization and online learning. However, since the neuromorphic architectures are not as solidly grounded in theory as the Von-Neumann architectures, achieving optimal application-specific configuration that lead to design decisions for fabrication is more empirical and discovery-driven. Hence, a software-hardware codesign of algorithmic learning and neurmorphic hardware design is crucial to achieve efficient and custom hardware for online and continual learning.

Opportunity: Design and fabrication of application-specific custom hardware that maximize the efficiency and power/performance profile can be achieved by a careful codesign of the learning algorithm and neuromorphic hardware design. One approach to achieve this is through the design of biologically-inspired network architecture [3] that incorporates heterogeneous and local synaptic plasticity that can emulate the adaptation and consolidation in biological neural networks and mitigate catastrophic forgetting. Formally, given a stream of inputs $\mathbf{u}(t)$ and potentially sparse modulatory signals $\mathbf{x}_m(t)$, the learning can be defined by:

E-mail address: `smadireddy@anl.gov`.

$$\begin{aligned}\mathbf{x}_0(t) &= F\left(\mathbf{u}(t), \mathbf{x}_m(t); S(t)\right)\\ S(t+1) &= F_s\left(S(t), \mathbf{u}(t), \mathbf{x}_m(t), \mathbf{x}_0(t)\right)\end{aligned}$$

where $S(t)$ represents the internal state of the system. To make this amenable for software-hardware codesign, we can decompose this into four coupled systems:

$$\begin{aligned}\mathbf{x}_e &= F_e(\mathbf{u}, W_e)\\ \mathbf{x}_o &= F_l(\mathbf{x}_e, W_l)\\ W(t+1) &= f\left(\mathbf{x}_e, \mathbf{x}_o, \mathbf{x}_m; W(t), \beta_k(t)\right)\\ \beta(t+1) &= f_\beta\left(\mathbf{x}_e, \mathbf{x}_o, \mathbf{x}_m; W(t), \beta_k(t)\right)\end{aligned}$$

subject to the initial conditions $W(0)$, $\beta(0)$. The first two equations represent the feedforward inference system, broken into a feature extraction and a learning modules, and the last two equations represent the synaptic plasticity mechanism and the evolution of the state of the network, represented by the evolution of hyperparameters β. In contrast to conventional machine learning approaches, there is no separation between architecture and learning algorithm: the ability to learn is defined by the choice of f, f_β, and their hyperparameters.

Design choices to fabricate an application-specific neuromorphic hardware can be obtained through a careful optimization of the f, f_β, and their hyperparameters. The design space defined by the learning rules f and f_β are extremely large, can be chosen from several possible biological learning mechanisms and theory, and can have a tangible impact on the performance of the learning algorithm.

This equivalency between the learning algorithm and design choices for a neuromorphic chip fabrication, modularity, and well defined interfaces between components provides a unique opportunity to applying mixed-integer optimization tools to general architecture design in a way that is driven by the final application or performance. Scalable optimization frameworks [2] designed to utilize state-of-the-art leadership class high performance computing systems can be adopted to our advantage in this co-design approach. A key advantage of abstracting architecture design as a mathematical problem is that the same codesign methodology can be applied both to software and hardware implementations. Each hardware implementation will introduce different constraints that will impact functionality, performance, and observables such as speed or power consumption. Consequently, beyond optimizing a specific architecture, our approach can provide a way of exploring the impact that the physical layer has on the specific optimal configuration for each application. Successful demonstration of the neuromorphic hardware design with these principles has also been shown recently [4].

Timeliness: Scientific Machine Learning is a core component of artificial intelligence which has been identified in a recent DOE report [1] as a key technology to enable tranformative science and energy research. On the other hand, research and advancement on materials and processes for neuromorphic hardware design has brought us to the crossroads where the time is ripe to develop the scientific application driven custom hardware/software co-design. This has the potential to enable science at the scale and pace that was not possible earlier.

References

[1] N. Baker, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. USDOE Office of Science (SC), Washington, DC, 2019.

[2] P. Balaprakash, et al. DeepHyper: Asynchronous Hyperparameter Search for Deep Neural Networks. In 25th IEEE International Conference on High Performance Computing, Data, and Analytics (2018).

[3] S. Madireddy, A. Yanguas-Gil, P. Balaprakash, Multilayer Neuromodulated Architectures for Memory-Constrained Online Continual Learning, arXiv:2007.08159 (2020).

[4] A. Daram, D. Kudithipudi, and A. Yanguas-Gil. "Task-based neuromodulation architecture for lifelong learning." 20th ISQED. IEEE, 2019.

Towards Hybridizing AutoML and Codesign for Next-Generation Scientific Applications

Challenges and Opportunities

Addi Malviya-Thakur
Software Engineering Group, Advanced Computer Systems Research Section,
Computer Science and Mathematics Division, Oak Ridge National Laboratory,
Oak Ridge, USA. malviyaa@ornl.gov, ORCID: 0000-0002-2681-9992

Topic— architectures, emerging technologies, codesign methods

I. Challenge

Exascale Computing and Artificial Intelligence will bring transformative change to the scientific community and foster novel discoveries. This change will largely be fueled by the scientific applications that would operate at the intersection of problem and solution space across varied domains, including nuclear, materials, neutrons, and energy. Although never pursued together, both Codesign and Artificial Intelligence (AI) has separately been an active interest among the scientific community and DOE for more than a decade now. While successes have been achieved by applying AI, albeit, owing to its complexity and technological difficulty, AI is still limited to its domain experts and core practitioners. With the arrival of Automated machine learning (AutoML), there is now an unparallel opportunity to democratize the use of AI across all scientific domains and give its power in the hands of non-practitioners too. AutoML is the process of automating the process of applying machine learning to real-world problems[1], [2]. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model. Scientific Applications powered by AutoML will be an ideal scenario for deriving new insights and pursue data-driven discoveries reliably faster. Investing in Codesign efforts of scientific application with a focus on bringing AutoML into the decision-making process[3]. To that end, author proposes hybridizing Codesign and AutoML for the next generation of Scientific software to maximize the overall performance, efficiency, and other desirable qualities of the system as a whole. However, there are certain challenges before the community can realize its full potential. Some of these challenges are listed below:

A. *AutoML focused on automating and improving ML pipeline:*

The current state of AutoML is narrowly focused on improving model building that includes feature selection and engineering, model selection, hyperparameter optimization, and stacking. While important, the state of art AutoML still struggles with complex ML problems involving high-dimensional data and imbalanced classes. Besides, current focus is on supervised learning problems and not on unsupervised or reinforcement learning type of ML problems. Scientific software development that combines Codesign will envision both supervised and unsupervised class of research problems to generate optimal solution.

B. *AutoML is limited to non-scientific applications:*

Current trends and need for AutoML are driven by enterprise application requirements. Scientific research involving mathematical tractability and solutions requires additional considerations that are currently unavailable.

C. *AutoML is data-intensive, while Codesign is compute-intenstive:*

AutoML best works when the applications have very large quantities of raw data. It is purposefully made to deliver data-driven and data-intensive solutions. Most Codesign is focused on optimal utilization of compute-intensive resources. This dichotomy has to resolved before realizing the potential of AutoML in scientific applications.

D. *AutoML is implemented in high-level programming languages:*

The implementation of AutoML is done in contemporary high-level languages such as Python or Java. The use of high-level programmatic constructs is driven by the need for productivity and not performance. Besides, interpreted and dynamic languages are difficult to optimize. Also, Python has an inbuilt global interpreter lock making it difficult to parallelize the code. Finally, these language designs and implementation are made without considering the realities of HPC.

E. *AutoML is hardware agnostic*

Since AutoML is implemented in high language constructs they assume unlimited raw power and rarely focus on optimization and efficient use of hardware resources. For scientific applications, it is critical to have bounds on execution. Also, Codesign are implemented in-silico for optimization considerations. It is important that AutoML and underlying hardware understand each other, while model selection processes, bring underlying hardware for performance.

II. Opportunity

There are several opportunities and advantages that make it feasible to envision AutoML driven scientific software Codesign. Majority of the innovation is occurring in open-source community, providing access to source code and open license agreement for further development and customization.

Exascale computing project has already recognized the need for high-level constructs and therefore welcome and like to support Codesign that support Python development in HPC environment via a combination of tools and libraries such as Cython[4], mpi4py[5], and PyFr[6]. AutoML can be developed on the top of these packages.

A wide range of feature selection and hyperparameter optimization approaches exists making it uniquely positioned to benefit HPC community. Besides, the Codesign community is currently under the process to develop frameworks and workflow which is apt for the current development stage of AutoML, thereby getting involved at an early stage.

Despite current approaches are agnostic, the DOE should consider this a need for immediate attention to developing programmatic urgencies surrounding the development of *in-silico* approaches and programs.

Research and opportunity in AI Systems Hardware/Software Co-Design is being felt across industry and academia, creating powerful support in the effort to develop distributed training, energy-efficient architectures, scalable communication, high-performance and fault tolerant communication, among others. Some custom frameworks for scientific applications are also under an active development[7], [8].

III. Timeliness or Maturity

There is no better timing than now to establish research around hybridization of AutoML and Codesign for several reasons.

- Data-intensive computing is continuing to grow, and, in any research, AI will be cornerstone for novel discoveries. Making AI easier through recent advances in AutoML for non-practitioners could be a game changer.
- It makes sense for future scientific applications to be AI-enabled. Thus, for effective and efficiency it is pertinent to explore R&D opportunities that combines AutoML and Codesign for optimal utilization of resources. This has not been done yet.
- Both these areas are currently under active development across industry and academia separately. The DOE office should consider this an important priority and develop funding programs surrounding this combined scientific challenge.
- A recent DOE AIML working group has noted lack of a system in place to share AI models, algorithms, and lessons learned across the DOE. Investment in AutoML and Codesign can also enable a guidance towards this.

The computing effort surrounding reimagining Codesign initiative has successfully pursued foundation scientific software that can run on Exascale hardware, the next logical step would be to develop joint research portfolio for Codesign that enables the use of new hardware-acceleration paradigms and AI. The author hopes this integration will open novel possibilities for artificial intelligence integrated scientific applications.

Acknowledgment

This manuscript has been authored by UT-Battelle, LLC under Contract No. DEAC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan http://energy.gov/downloads/doe-public-access-plan.

References

[1] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2013, vol. Part F128815, pp. 847–855, doi: 10.1145/2487575.2487629.

[2] “AutoML workshop @ ICML’14,” 2014. https://sites.google.com/site/automlwsicml14/

[3] B. Tine, F. Elsabbagh, L. Seyong, J. Vetter, and H. Kim, “Cash: A Single-Source Hardware-Software Codesign Framework for Rapid Prototyping,” in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020, p. 321, doi: 10.1145/3373087.3375340.

[4] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, “Cython: The best of both worlds,” *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 31–39, Mar. 2011, doi: 10.1109/MCSE.2010.118.

[5] R. Smith, “Performance of MPI codes written in python with NumPy and mpi4py,” in *Proceedings of PyHPC 2016: 6th Workshop on Python for High-Performance and Scientific Computing - Held in conjunction with SC16: The International Conference for High Performance Computing, Networking, Storage and Analysis*, Jan. 2017, pp. 45–51, doi: 10.1109/PyHPC.2016.010.

[6] F. D. Witherden, A. M. Farrington, and P. E. Vincent, “PyFR: An open source framework for solving advection-diffusion type problems on streaming architectures using the flux reconstruction approach,” *Comput. Phys. Commun.*, vol. 185, no. 11, pp. 3028–3040, Nov. 2014, doi: 10.1016/j.cpc.2014.07.011.

[7] E. Barreiro, C. R. Munteanu, M. Cruz-Monteagudo, A. Pazos, and H. González-Díaz, “Net-Net Auto Machine Learning (AutoML) Prediction of Complex Ecosystems,” *Sci. Rep.*, vol. 8, no. 1, p. 12340, Dec. 2018, doi: 10.1038/s41598-018-30637-w.

[8] M. S. Abdelfattah, Ł. Dudziak, T. Chau, R. Lee, H. Kim, and N. D. Lane, “Best of both worlds: Automl codesign of a CNN and its hardware accelerator,” *arXiv*. arXiv, Feb. 11, 2020, Accessed: Feb. 14, 2021. [Online]. Available: https://arxiv.org/abs/2002.05022v2.

Rethinking Codesign Tools for Data-movement

Andres Marquez (Andres.Marquez@pnnl.gov), Pacific Northwest National Lab
Nathan Tallent, Pacific Northwest National Lab

February 16, 2021

Topic. Modeling and simulation. Codesign methodologies.

Challenge. Scientific exploration and hypothesis generation is increasingly dependent on the convergence of scientific modeling, data analytics, and AI/ML. The result is *workflows*, composed of multiple stages of compute and communication between distributed and rapidly increasing heterogeneous and distributed computing resources. *Data movement between devices, memories, and storage is the most significant bottleneck.* Many analytics and AI/ML methods fundamentally rely on large amounts of data, which is costly to move; they employ algorithms whose data structures and access patterns are detrimental to performance. Workflow modules are usually written in software packages using multiple computing paradigms and assumptions, yet performance requires careful orchestration of tasks and data flows on hardware and networks with diverse characteristics.

The problem will likely be exacerbated by trends in hardware and networking. Hardware is becoming increasingly fragmented, with new CPU architectures and domain accelerators abounding, targeting specialized kernels. Memory systems are far more complex and commonly include multiple distinct memory pools. Networking latency growth is flat, meaning the mismatch between data movement and compute will remain large for the foreseeable future.

Vision. These problems are not new, but a perfect storm of new computing paradigms in the HPC space, paired with increasing heterogeneity –driven by limits in Moore and Dennard scaling – as well as integrated, federated workflows – including edge computing, in the loop computing – nudge us to learn insights gleaned elsewhere.

DOE's exascale codesign efforts with industry – Fast, Path Forward – developed some capability within this important technology segment but it has become clear that further performance and power efficiency gains will require less hands-off codesign outsourcing, with the labs picking up a larger share of the codesign mantle.

The embedded world has dealt with specialization and extreme heterogeneity for much of its existence. The result has been a flurry of HW/SW codesign tools to accelerate the product development cycle. Although it might be attractive to adopt the processes developed in the embedded space, there are some caveats for the HPC community:

- First, most of the existing codesign tools are HW centric, as they were developed bottom-up by HW designers. The programming models are limited, lacking the programmability sophistication HPC practitioners have grown accustomed.
- Second, due to their HW centricity, the tools operate on design cycles that are reasonable for HW designers but are almost unworkable for SW coders, accustomed to incremental or RAD design cycles.
- Third, governing metrics that drive optimization in the embedded space are not necessarily aligned with the HPC space – e.g., performance, power, memory/storage capacity.
- Fourth, embedded codesign tools tend to include memory considerations in a phased approach, whereas the execution engine is designed a-priori and memory is dropped in a-posteriori. Such an approach in the HPC domain would negate much of the promise codesign would hold.

Position statement. We posit that codesign in the HPC space has to take these shortcomings of the embedded space into consideration and propose that a new crop of codesign tools need to come to the forefront to be applicable to the HPC community. These codesign tools need to satisfy the needs of SW and HW developers in equal measure, by emphasizing SW developer needs towards programmability, design cycles and metrics, and most importantly, emphasizing memory shortcomings detrimental to HPC performance. These tools should interface with HW and SW design tools in order to be integrated into the corresponding design flows of each domain. They should serve as seamless platforms over which HW and SW developers exchange metrics and constrains.

Tackling the fourth problem will require reassessing existing tools in existence. Today's codesign methods utilize performance models that are useful only for simple data access patterns or throughput performance [14, 13]. They fail for data-intensive analytics with sparse structures, irregular accesses, etc [4]. The HW approach of simulation provides high accuracy, but requires enormous resources, even for less-than realistic inputs. Analytical modeling methods either require significant human labor; or use automated methods that focus on limited domains such as throughput. ML modeling performs well for easier problems such as throughput performance, but performs poorly on noisy data with heavy tails, which is characteristic of latency-sensitive bottlenecks and resource congestion.

Opportunities. The opportunities to develop these next-gen codesign tools within the HPC community derive from the confluence of its expertise with high performing workflows [3, 4] running on cutting edge technology. This body of knowledge should drive data-centric measurement and analysis tools that characterize how tasks affect and use data. These tools should highlight toxic data layouts and access patterns that cause performance penalties across storage [2, 1], memory [5, 8, 10, 9], and network [12, 15].

Steeped in the HPC ModSim tradition, model-generation and evaluation techniques should provide for concise and elastic data-centric models that enable reasoning about data representations, layouts [7], task compositions, and run-time data policies.

Finally, these tools should provide execution guidance and runtime support for runtime data partitioning, task placement, data layouts, and polices for data-movement in order to coordinate data object interchange and movement [6], enabled by introspection, control and adaptation based on guidance from template predicates and models [11].

Timeliness. As to the timeliness of such endeavor, several aspects come into play. As mentioned above, the HPC community and industry at large are at a juncture to provide performance and power efficiency gains beyond technology node scaling.

Furthermore, over several decades of pursuing the holy grail of High Level Synthesis (HLS), the Electronic Design Automation (EDA) industry has come to an understanding of what is practically feasible.

Fortuitously, some ML driven novel architectures realized as static- and or dynamic- dataflow lend themselves to EDA, easing the HLS burden by restricting generality. In the same spirit, the HPC community should heed the cue of the EDA industry and constrain its problem formulations accordingly.

[1] O. Bel, K. Chang, N. R. Tallent, D. Duellmann, E. L. Miller, F. Nawab, and D. D. E. Long. Geomancy: Automated performance enhancement through data layout optimization. In *36th Intl. Conf. on Massive Storage Systems and Technology*, October 2020.

[2] R. D. Friese, B. O. Mutlu, N. R. Tallent, J. Suetterlein, and J. Strube. Effectively using remote I/O for work composition in distributed workflows. In *Proc. of the 2020 IEEE Intl. Conf. on Big Data*. IEEE Computer Society, December 2020.

[3] R. D. Friese, N. R. Tallent, M. Schram, M. Halappanavar, and K. J. Barker. Optimizing distributed data-intensive workflows. In *Proc. of the 2018 IEEE Conf. on Cluster Computing*, pages 279–289. IEEE, September 2018.

[4] R. D. Friese, N. R. Tallent, A. Vishnu, D. J. Kerbyson, and A. Hoisie. Generating performance models for irregular applications. In *Proc. of the 31st IEEE Intl. Parallel and Distributed Processing Symp.*, pages 317–326, Los Alamitos, CA, USA, May 2017. IEEE Computer Society.

[5] O. O. Kilic, N. R. Tallent, and R. D. Friese. Rapid memory footprint access diagnostics. In *Proc. of the 2020 IEEE Intl. Symp. on Performance Analysis of Systems and Software*. IEEE Computer Society, Oct. 2020.

[6] J. Landwehr, J. Suetterlein, J. Manzano, A. Marquez, K. J. Barker, and G. R. Gao. Designing scalable distributed memory models: A case study. In *Proceedings of the Computing Frontiers Conference*, CF'17, page 174–182, New York, NY, USA, 2017. Association for Computing Machinery.

[7] A. Marquez, J. Manzano, S. L. Song, B. Meister, S. Shrestha, T. St. John, and G. Gao. Acdt: Architected composite data types trading-in unfettered data access for improved execution. In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 289–297, 2014.

[8] S. Shrestha, G. R. Gao, J. Manzano, A. Marquez, and J. Feo. Locality aware concurrent start for stencil applications. In *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 157–166, 2015.

[9] S. Shrestha, J. Manzano, A. Marquez, J. Feo, and G. R. Gao. Jagged tiling for intra-tile parallelism and fine-grain multithreading. In J. Brodman and P. Tu, editors, *Languages and Compilers for Parallel Computing*, pages 161–175, Cham, 2015. Springer International Publishing.

[10] S. Shrestha, J. Manzano, A. Marquez, S. Zuckerman, S. Song, and G. R. Gao. Gregarious data re-structuring in a many core architecture. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 712–720, 2015.

[11] J. Suetterlein, J. Landwehr, A. Márquez, J. B. Manzano, and G. R. Gao. Asynchronous runtimes in action: An introspective framework for a next gen runtime. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1744–1751, 2016.

[12] N. R. Tallent, K. J. Barker, D. Chavarría-Miranda, A. Tumeo, M. Halappanavar, A. Márquez, D. J. Kerbyson, and A. Hoisie. Modeling the impact of silicon photonics on graph analytics. In *Proc. of the 11th IEEE Intl. Conf. on Networking, Architecture, and Storage*, pages 1–11. IEEE Computer Society, Aug 2016.

[13] N. R. Tallent and A. Hoisie. Palm: Easing the burden of analytical performance modeling. In *Proc. of the 28th ACM Intl. Conf. on Supercomputing*, pages 221–230, New York, NY, USA, 2014. ACM.

[14] N. R. Tallent, D. J. Kerbyson, and A. Hoisie. Representative paths analysis. In *Proc. of the Intl. Conf. for High Performance Computing, Networking, Storage and Analysis (SuperComputing)*, pages 34:1–34:12, New York, NY, USA, November 2017. ACM.

[15] N. R. Tallent, A. Vishnu, H. V. Dam, J. Daily, D. Kerbyson, and A. Hoisie. Diagnosing the causes and severity of one-sided message contention. In *Proc. of the 20th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, New York, NY, USA, 2015. ACM.

Acknowledgment. This work was partially funded by the PNNL Laboratory-Directed Research and Development (LDRD) funded Data-Model Convergence Initiative and the DOE Office of Science funded Center for Advanced Technology Evaluation (CENATE) project. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract No. DE-AC05-76RL01830.

Compiler, OS, and Scheduler Support for Future Disaggregated HPC Systems

George Michelogiannakis, Tan Nguyen, and John Shalf
Lawrence Berkeley National Laboratory, {mihelog, tannguyen, jshalf}@lbl.gov

Topic: Architectures, codesign methodologies

Challenge

Specialization is slowly coming into focus for future HPC systems as a strategy to preserve performance scaling for key applications. The diversity of important HPC applications creates the concern that future non-specialized systems will contain tens or hundreds of different types of compute units from general-purpose such as traditional CPUs, compute units that are efficient for specific types of computation such as GPUs, fixed-function accelerators such as for FFT, or even reconfigurable compute units such as FPGAs. Likewise, future systems may also have a variety of memory resources to better fit specific data access patterns or provide non-volatile storage.

Even without specialization, today's HPC systems overprovision resources in order to satisfy a few important applications. For example, previous studies have shown that at the majority of the time, nodes use less than 25% of their memory [1,2]. Because applications reserve resources in units of nodes and the configuration of nodes is uniform and rigid, in today's systems applications that do not have high memory requirements have no choice but to reserve enough nodes to satisfy their compute requirements and subsequently idle most of node memory, leading to overall resource underutilization. On the other hand, if we reduce available memory in nodes, this will penalize few but important applications.

This has led to a push towards resource disaggregation. Resource disaggregation is the ability of the system to allocate resources to applications in a fine-grained manner. This way, an application can reserve only the resources it requires, without being bound by pre-decided node configurations. In some sense, resource disaggregation allows the system to pool and compose resources according to each application's requirements (figure 1). There are numerous approaches on how to provide this capability from the hardware such as using optics or placing resources of the same type at different parts of the system instead of distributing them [3].

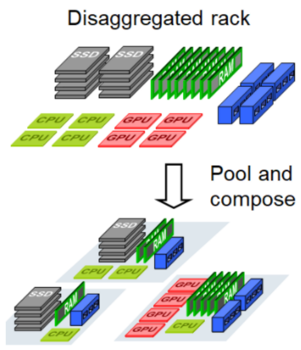


Figure 1

However, even if we assume a perfectly disaggregated hardware, currently we have no strategy on how to best extract each application's resource requirements, especially without any assumptions on prior knowledge from the application. This cannot simply rely on the programmer to express such requirements in the application code because more resource types combined with different missions of different systems means that future HPC systems are likely to differ significantly in terms of what compute and memory resources they have available. Therefore, if we devise a solution that is based on the application code making use of system-specific API, the application's code would have to be modified to run efficiently in another system. This significantly hinders code portability. Finally, even if we have perfect knowledge of each application's resource requirements, the next question is how to communicate those requirements to the job scheduler and how scheduling policies should adapt, taking into account resource utilization as well as application performance.

Efficient use of future hardware requires eliminating resource underutilization while assigning the most suitable resources to each application and avoiding re-writing application code when porting to a different system.

Opportunity

Addressing this challenge warrants a multi-disciplinary and collaborative approach. The first challenge is how to accurately extract an application's resource requirements without having to rely on the programmer to explicitly specify them. For this, there are several approaches such as: (i) task the compiler with extracting information such as a dataflow graph and then match the computation and data access patterns to available hardware resources (for instance, graphs that resemble FFT will be paired to an FFT accelerator), (ii) create an abstract API that HPC systems share, and (iii) allow applications to execute on the system but monitor their compute and memory usage in order to match them with available resources using methods such as reinforcement learning or graph matching. Another relevant question is which parts of the application are best to specialize.

This exploration has to consider cost versus accuracy tradeoffs. That is, observing an application is less invasive to the system, but at the same time will produce an optimal matching to specialized resources less frequently. This leads towards a cost model that includes the penalty to an application's execution time from suboptimal assignments. In addition, assigning an application to a specialized resource has more costs such as the delay and energy to transfer data to and from it, that should be taken into account to offset the benefit of using a specialized resource.

In tandem with extracting an application's requirements, the complexity of job scheduling will grow. This means that job schedulers now should strive to increase the utilization of all the different types of resources available in the system, but at the same time balance this goal with "traditional" goals such as preserving high application performance and low energy usage. In addition, the job scheduler should be more aware of the hardware architecture of a particular system such as to estimate the various costs and potential contention scenarios between applications for different application placement options it is considering. Ideally, this should all be happening abstractly from the user who may not understand or do a good job at manually picking specialized resources for his applications.

This effort produces more opportunities with synergistic technologies. For instance, FPGAs or other reconfigurable fabrics could be adapted to this trust by being reconfigured by the job scheduler as part of the step of finding specialized resources for each application. In other words, while setting up a job to initiate in the system, the job scheduler, knowing the computation or data access characteristics of the application, can program an FPGA, all abstractly from the user.

Timeliness or Maturity

This research trust is made timely by the constantly-increasing research focus on hardware to enable resource disaggregation, the projected slow down of traditional technology performance scaling, and the constant desire for more performance. Already industry is investigating resource disaggregation but not necessarily for applications important to DOE. Still, by doing so, industry is setting up an ecosystem friendly to resource disaggregation.

Since we are slowly gaining more understanding on how hardware in future HPC systems will look like, it is becoming easier to co-design software to go along. In addition, synergistic technologies such as photonics, reconfigurable networks, different types, and a variety of specialized compute units are maturing, giving this research trust a clear benefit not just to the system but to each application as well.

If successful, this research area will not only improve application performance and reduce resource underutilization and thus cost of future HPC systems, but it will also make application code portability easier, which is otherwise sure to only become harder with future heterogeneous hardware.

References

[1] J. Shalf, G. et al, "Photonic Memory Disaggregation in Datacenters," in OSA Advanced Photonics Congress (AP) 2020, OSA Technical Digest (Optical Society of America, 2020).
[2] I. Peng, R. Pearce and M. Gokhale, "On the Memory Underutilization: Exploring Disaggregated Memory on HPC Systems," 2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD).
[3] M. Glick et al., "PINE: Photonic Integrated Networked Energy efficient datacenters (ENLITENED Program)," in IEEE/OSA Journal of Optical Communications and Networking.

Title: Application Specific Machine Learning Accelerators Using Digital and Mixed-Signal Reconfigurable Computing
Authors: J. Parker Mitchell, Oak Ridge National Laboratory (ORNL), mitchelljp1@ornl.gov; Catherine D. Schuman, ORNL; Maryam Parsa, ORNL; Shruti R. Kulkarni, ORNL; Prasanna Date, ORNL
Topic: architectures, programming systems, emerging technologies

Challenge:

The rate of new data is outpacing advancements in performance and energy efficiency, and machine learning techniques continue to proliferate throughout Department of Energy (DOE) research domains. In order to support this growing computational need, it is necessary to consider new approaches beyond today's CPU and GPU dominated systems. Because DOE applications often have unique requirements, custom architectures deployed on reconfigurable computing platforms offer one possible solution. Today, some techniques, such as HLS4ML [Duarte], exist to help map deep learning networks to Field Programmable Gate Arrays (FPGAs). However, there is a challenge in considering emerging or nontraditional technologies such as Field Programmable Analog Arrays (FPAAs). FPAAs may offer compelling efficiency advantages as shown by [Suma], but there is not currently sufficient tooling for researching and deploying application specific machine learning accelerators.

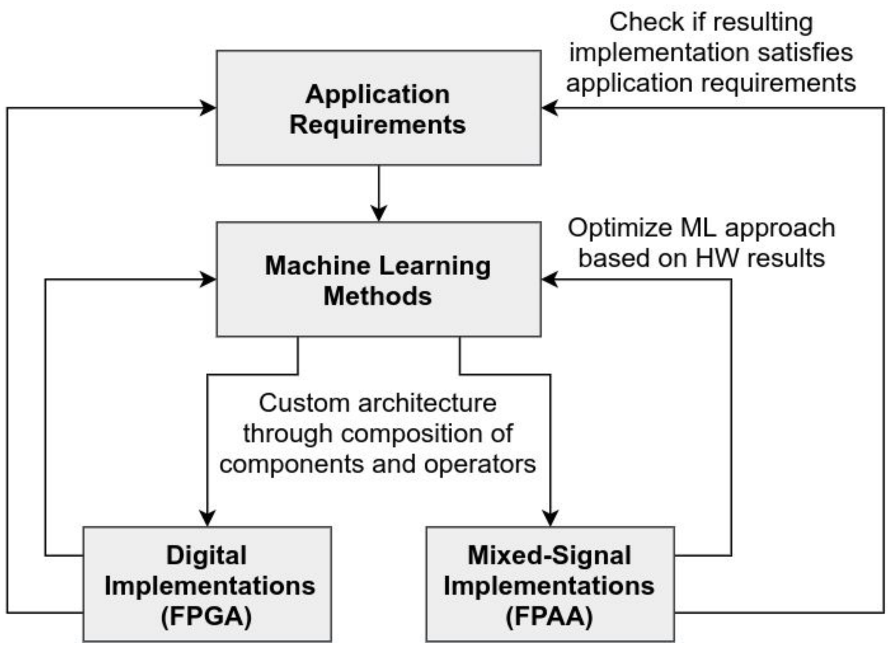

Check if resulting implementation satisfies application requirements
Application Requirements
Optimize ML approach based on HW results
Machine Learning Methods
Custom architecture through composition of components and operators
Digital Implementations (FPGA)
Mixed-Signal Implementations (FPAA)

Figure 1: High level diagram of application-specific accelerator framework

Opportunity:

Machine learning offers a set of abstractions and constructs which allow for various optimized implementations. Techniques like deep learning can be formulated into a static dataflow graph of math operations like matrix multiplication which may be efficiently implemented in both digital and analog domains. Other emerging technologies such as neuromorphic computing may also be readily implemented as digital or analog. By developing a digital and mixed-signal codesign framework for methods like deep learning and neuromorphic computing, researchers can examine the tradeoffs of customized implementations for FPGAs and FPAAs. These tradeoffs include metrics such as mathematical precision, prediction accuracy, energy efficiency, and inference latency. This framework should ideally allow the user to explore a range of algorithms and algorithm parameters in conjunction with hardware implementations. Accelerator architectures can be built by composing and parameterizing logical components like matrix multiplication or spiking neuron models. By leveraging reconfigurable platforms, design decisions can be tailored to a specific application's needs. A high level view of this methodology is shown in Figure 1.

Timeliness or maturity:

FPGA hardware is very mature and is currently used at many DOE user facilities. However, FPAA hardware exists primarily in academic environments. Recent works provide optimism that FPAAs can scale to newer process nodes and achieve the necessary density and efficiency for practical applications [Hasler]. Utilizing insights from a codesign framework, new FPAAs could be developed assuming sufficient funding for fabrication. Many machine learning methods such as convolutional neural networks, decision tree ensembles, and support vector machines are very mature and currently being utilized. Other methods such as spiking neural networks are a hot topic of research with a wide range of digital and mixed-signal implementations [Schuman] which naturally lends itself to reconfigurable computing to allow research flexibility.

References:

Duarte, Javier, et al. "Fast inference of deep neural networks in FPGAs for particle physics." *Journal of Instrumentation* 13.07 (2018): P07027.

George, Suma, et al. "A programmable and configurable mixed-mode FPAA SoC." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.6 (2016): 2253-2261.

Hasler, Jennifer. "Large-scale field-programmable analog arrays." *Proceedings of the IEEE* 108.8 (2019): 1283-1302.

Schuman, Catherine D., et al. "A survey of neuromorphic computing and neural networks in hardware." *arXiv preprint arXiv:1705.06963* (2017).

Confident, Adaptable, and Robust Machine Learning to Augment Traditional Modeling and Simulation

Matthew R. Norman and Muralikrishnan Gopalakrishnan Meena

Topics: Applications, modeling and simulation, emerging technologies

Introduction

Surrogate models can greatly simplify otherwise complex components of multi-scale and multi-physics scientific models. While there is a growing volume of research in Machine Learning (ML) surrogate models that take advantage of current (and likely future) HPC architectures, many scientific domains and applications are slow to apply these advances. To alleviate scientific hesitations in using ML surrogates, we need *novel AI architectures and workflows* that:

1. Improve domain science confidence and interpretability for practical production-level use;
2. Adapt to changing data in the middle of an ongoing simulation campaign as data diverges from the training set; and
3. Robustly handle noisy and often discontinuous input-output mappings in domain science data.

Further, improving domain scientists' willingness to integrate ML surrogates into production-level environments requires identifying candidate simulation codes and working closely with them to better understand what issues need to be addressed.

Scientific Confidence

Current Limitations: The possibility of ML surrogates producing unrealistic results often leads domain scientists to distrust using them in production-level environments. The inability to interpret most of the internal logic of a ML prediction also gives scientists pause in using them.

Opportunities: An approachable way to improve scientific confidence in ML surrogates is to project Neural Network (NN) architectures onto already known and vetted models with parameters that have known bounds for realism and/or stability. As an example, suppose a surrogate model for unresolved sub-grid-scale (SGS) effects from a high-resolution fluid model is desired. One could directly predict those fluid tendencies from high-resolution model data. However, there is no guarantee that when used in practice, that the tendencies will be bounded to realistic and stable values. If, however, one projected these tendencies onto a known model such as dynamic Smagorinsky scheme [2] or a novel combination of SGS mixing and transport, then one can bound the learned parameters to known stability limits during both training and inference stages. Further, this injects a layer of easy interpretation to the ML prediction, although the prediction of the used constants may still elude easy interpretation. Significantly more research needs to be performed into how to project existing ML models onto more interpretable models where resulting parameters can be monitored and bounded during production use.

NNs are also differentiable, giving easy access to adjoints for error estimation and uncertainty quantification in the learned models. We need more research into how to perform these processes to give scientists greater understanding of the behavior of ML surrogates.

Further, there is growing research in understanding Neural Network behavior more robustly, but often, this research is not well-understood by domain scientists, limiting their inclusion in production codes.

Adaptability

Current Limitations: Another hurdle in real-world application of ML models is their inability to smoothly adapt to data that diverges from the training regime. As an example, climate simulations that project changes as they evolve in time due to the influence of anthropogenic forcings and feedbacks will inevitably create novel flow regimes that haven't been seen before. In fact, scientific simulations are *expected* to produce never-before-seen data. ML surrogates must find ways to adapt to this inevitability.

Opportunities: One mitigation is to apply surrogates at much smaller (less monolithic) levels in simulations. Research needs to be performed into how seemingly irreducible operators might be broken down into more manageable pieces that are easier to train in a robust way.

Another approach is to construct realistic workflows that can manage unexpected surrogate model inputs. This requires the ability to determine if a given input is "sufficiently" similar to the training set, with a contextual definition for what "sufficiently" means. For instance, the "detector" of a General Adversarial Network (GAN) [3] could be used to determine how similar an input is to the training set. Furthermore, a fluid in-situ workflow is needed for aggregating and exporting inputs for further training as well as training and deploying new surrogates – all during the scientific simulation. Novel transfer learning techniques and system-level software are likely needed to make this both possible and convenient.

Robustness

Current Limitations: A significant limitation for traditional NN architectures is that largely *linear* mathematical basis functions are used. Somewhat large dense linear algebraic operators are wrapped in relatively infrequent non-linear "activation" functions, which are the sole means of fitting NNs to arbitrary non-linear input-output mappings. While mathematically, NNs are *able* to be trained to emulate any non-linear operator with a finite number of discontinuities [1], in practice, when the input-output mappings are noisy or discontinuous, NNs tend to give poor accuracy, e.g., increasing Reynold's numbers in Computational Fluid Dynamics [4]. While Random Forests often perform well in noisy or discontinuous regimes, they are based on graph traversals, which do not perform particularly well on accelerators.

Opportunities: There is significant need for more research into novel ML basis functions that inject discontinuities and non-linearity at a much more *fine-grained* level. This way, the ML model is more likely to be able to adapt to significant discontinuities in the internal state space. Moreover, we need to merge this research with the constraint that the models perform well on accelerator devices that increasingly reward floating point operations and penalize reliance on data movement and large amounts of cache. This is a relatively under-investigated aspect of ML, even though it has the potential to fundamentally change the level of accuracy we should expect from a surrogate model. New mathematical basis functions for ML may also influence the specialized hardware created by vendors for AI workflows.

Timeliness

The recent confluence of advances in ML effectiveness and new AI-specific hardware motivate an increased push to ensure domain science applications take full advantage of these maturing technologies. For many domains, there still remains a disconnect in using ML surrogates in production environments, often related to uncertainty in behavior when encountering new inputs. This provides an opportunity to bridge that disconnect with advances in ML intepretability, HPC workflows, I/O capabilities, and ML mathematical basis functions.

References

[1] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[2] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 1991.

[3] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[4] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Co-Design for Radiation Transport in Astrophysics
Topic: Modeling and Simulation

Josh Dolence (LANL:jdolence@lanl.gov)   Chris Fryer(LANL:fryer@lanl.gov)
Daniel Kasen (LBNL:kasen@berkeley.edu)   Peter Nugent(LBNL:nugent@lbl.gov)

February 2021

1 Challenge

While past co-design efforts were focused on a single solution that was broadly optimized to tackle the vast majority of high-performance computing relevant to DOE, we argue here that such a path to solution provides for only modest gains for the majority of users at significant cost. To be truly transformational, we should not think of one, uber, co-design effort, but of multiple co-design programs grouped by similar underlying algorithmic and science challenges which will result in several different, highly-optimized, compute capabilities. The time is right for such an effort as the both the size of the next generation exascale machines and the emerging market for the creation of optimized hardware is such that one could envision smaller, focused programs driving this market in several new directions rather than settling for a single machine driven by the largest next-generation commodity computing. For this effort, we are focusing on the modeling of the emission from astrophysical transients. From shock breakout to late-time emission, modeling radiation arising from astrophysical explosions requires a broad suite of multi-physics, multi-regime and multi-scale modeling. This challenge is similar to those faced by several other simulation efforts in the DOE portfolio and is driven by a desire to balance needs in both memory and speed.

1.1 Radiation Transport across Different Simulation Regions

In astrophysical transients, conditions vary across time and space in such a way that the required fidelity of the radiation modeling can also vary significantly. At one extreme, local thermodynamic equilibrium between the radiation and matter can be safely assumed and the radiation field is adequately described by a single equilibrium temperature. At the other extreme, the full six-dimensional phase space must be evolved, requiring an enormous number of degrees of freedom. These conditions, and the full spectrum of intermediate conditions, often occur in the same problem, leading to inappropriate approximations that sacrifice fidelity for the sake of computational expedience or a lack of progress because solving the problem at full fidelity is orders of magnitude more costly than feasible. Meanwhile, emerging facilities will yield observational data that demand numerical models well beyond the current state-of-the-art if they are to be appropriately interpreted. A particularly timely and challenging problem concerns shock breakout in supernovae and subsequent interactions with complex environments. Early-time observations of supernovae have the potential to place strong constraints on properties of the supernova progenitors including the companions of thermonuclear supernovae and the immediate surroundings of core-collapse supernovae set by the explosive activity of its progenitor just before collapse. However, understanding and learning from these observations requires codes that can capture the radiation hydrodynamics across all regimes and resolve the physics on a wide range of scales. With the appropriate capabilities, we can both better understand existing data and support the case for proposed missions focused on observing these early-time transient signals.

1.2 3-D, Time-dependent, NLTE Radiation Transport

Local thermodynamic equilibrium refers to a system where all of the particles and (e.g. electrons, ions, radiation, atomic level states) are in equilibrium where they can be described by a single temperature. Non-

LTE refers to any situation where there is a break in this equilibrium. The simplest situations, electron/ion decoupling (but still represented by individual temperatures) or radiation decoupling (and even radiation deviating from a simple temperature blackbody) have been worked on for decades. But modeling electron or ion energy distributions, or solving a series of rate equations to determine the level populations for a selected set of ionic species, remains in its infancy. For our application, calculating the electron energy distribution and its effect on the level populations is important. Given a finite piece of the atmosphere, these effects include both non-thermal ionization due to the local γ-ray deposition and the effects of the impinging radiation field. One solves for the level populations of every important, ionic species. This includes solving both radiative and collisional rates, which are a function of the impinging radiation field and electron energy distribution. The rate equations are non-linear with respect to the population densities and the system of equations is closed by the conservation equations for the nuclei and the charge conservation equation. Typically these are solved via an iterative, operator splitting method.

As both the understanding of thermonuclear supernovae progenitor systems and their explosion mechanisms have been pushed by new observational data, the ability to have theory confront these observations is the only way to rule out certain scenarios and even offers a path to reduce systematic uncertainties in cosmology. A major issue for 3D NLTE calculations are the memory requirements for storing the relevant data. For each voxel we need to store (at least) the number of ions, line profiles, the radiative rates (up/down), the rate operators (one up/down pair per considered interaction), the general equation of state data (partial pressures for all species), and the data needed for the solution of the 3D radiative transfer equation at every wavelength point. For very modest supernova atmosphere calculations (with $\sim$ 300k voxels, 5k NLTE levels, 80k transitions, and 165k explicit coupled rate transitions), this amounts to over 1.2 TB. Thus effective domain decomposition methods must be employed. Time-dependence just exacerbates this issue. Here calculations of realistic supernova atmospheres require $\sim$70M cpu-hours for one atmosphere. In addition to optical spectra of supernovae, this work also impacts our understanding of radio/x-rays observations, relativistic jets, SN remnants, and the thermalization of radioactivity in kilonovae - the result of the recently observed neutron-star neutron-star mergers.

2 Opportunity

Simultaneous breakthroughs in both numerical methods and computational hardware, purpose-built in tandem, are required to tackle these and related problems. A multi-scale, multi-fidelity method for radiation transport and material coupling, that dynamically adapts the representation of the radiation field in space and time and seamlessly links disparate regimes of radiating flows, may appreciably reduce the overall cost of adequately modeling these systems. However, advances in hardware that accelerate the operations required for radiation transport, but are tolerant of the variable workload associated with an adaptive method and support dynamic load balancing, will likely prove as vital. Together, these developments could qualitatively redefine the state-of-the-art and enable unprecedented predictive modeling and new 3-D modeling workflows. For nuclear reaction networks, the challenge is in solving ODE's where the bottlenecks for GPU-based architectures are all memory-based. Issues with thread-level parallelism, load balancing and fast access to a shared memory pool are key hurdles to overcome.

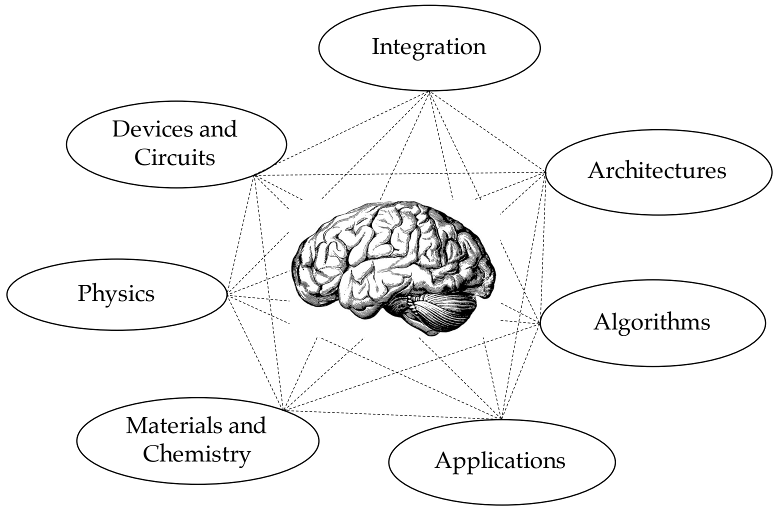
3 Timeliness

This work is particularly timely because of the wealth of transient telescopes being developed in the astrophysical community spanning an energy range from radio to gamma-ray wavelengths. The current hot topic of multi-messenger astrophysics is primarily focused on astrophysical transients and the electromagnetic radiation is crucial for much of the science done in this field. Given the impending launch of both new satellite missions (JWST, Roman) and ground-based telescopes (Rubin, ZTF, LS4), as well as proposed new missions (AMEGO, LOX, Athena, Altena, ULTRASAT, SIBEX), there is an urgent need for complementary advances in modeling capabilities.

Title: Bayesian Artificial Intelligence Governing Software-Hardware Codesign
Authors: Maryam Parsa, Oak Ridge National Laboratory (ORNL); Catherine Schuman, ORNL; Shruti Kulkarni, ORNL; Prasanna Date, ORNL; J. Parker Mitchell, ORNL
Topic: codesign methodologies, architectures, applications, programming systems, emerging technologies

Challenge:

Current CMOS-based microelectronics technologies are designed, developed, and manufactured from integrated contributions of bottom-up disciplines (layers) such as materials, physics, devices and circuits, architectures, algorithms, and finally applications. Each layer is modeled and built largely independent of the next layer [1]. However, in the Beyond Moore's era, it is necessary to rethink the sequential stacks and instead develop and codesign each layer connected to all other layers with multi-directional information flow.



There are, however, several **key challenges** involved in the **fully connected multi-directional layers** codesign:

1. What type of information should be transferred from one stack to another?
2. What is the performance metric of interest that affects the design of one layer based on the feedback received from another layer?
3. How frequent should the information be transferred between different layers?
4. How should we lead the codesign to the optimum performance so that all layers are co-optimized?

In addition, the diversity of the extremely heterogeneous architectures and various performance definitions among the layers add to the complexity of the software-hardware codesign in the fully connected approach discussed above.

Opportunity:

One of the leading theoretical frameworks in the area of neuroscience is "Bayesian Brain hypothesis". This hypothesis is based on believing that the brain implements statistically optimal algorithms by combining ***prior*** knowledge with new ***observations*** and evaluating quantities of interest with respect to ***posterior*** distribution [2]. Prior distribution, likelihood model (observations), and posterior distribution are the taxonomy of Bayesian optimization. A key opportunity exists in the area of software-hardware codesign to mimic the human brain and use Bayesian optimization to govern the codesign by learning from current performance of software and hardware (***prior***), observe the performance change from various interactions between materials and chemistry, physics, device and circuits, integration, architecture, and algorithm (***likelihood***), update the performance of the co-design (***posterior***), and finally decide on the best co-optimized design(s) leading the system to the optimal performance. In the context of hardware-software codesign, examples of the performance metric are software productivity (accuracy, speed), hardware productivity (energy consumption, and area requirements), execution-time productivity (such as efficiency, time, and cost for running scientific workloads), and workflow and analysis productivity (effort, time, and cost for the overall cycle of simulation and analysis) [3]. There is the opportunity to leverage **Multi-Objective Bayesian-based Optimization** that governs this **fully connected multi-directional layers of software-hardware codesign**, where objectives are the several aforementioned performance metrics.

Timeliness or maturity:

To achieve higher performance and avoid human driven optimization, significant efforts have been placed on automating model selection. A powerful method for obtaining the best performance neural architecture designs is Neural Architecture Search (NAS) [4]. The objective of these techniques is to automate architectural engineering to discover a network design which provides maximum performance. Hardware-aware NAS [5, 6], and Bayesian-based multi-objective hyperparameter optimization [7, 8], are among the available techniques for tuning hyperparameters of a neural network where accuracy is not the only performance metric. It is already shown in [7, 8, 9] the sensitivity of performance (in terms of accuracy, energy, size, and latency) in neuromorphic computing to the application and underlying hardware. Therefore, the relative maturity of multi-objective Bayesian optimization techniques in software-hardware codesign in neuromorphic computing today, provide the opportunity to establish a Bayesian AI that governs software-hardware codesign beyond neuromorphic computing.

References

[1] Murray, Cherry, et al. "Basic research needs for microelectronics": Report of the office of science workshop on basic research needs for microelectronics, October 23 – 25, 2018

[2] Lange, Richard D, et al., "Bayesian encoding and decoding as distinct perspectives on neural coding", bioRxiv, 2020

[3] Vetter, Jeffrey S, et al. "Extreme heterogeneity 2018 - productive computational science in the era of extreme heterogeneity", Report for doe ascr workshop on extreme heterogeneity, 2018

[4] Zogh, Barret, et al., "Learning transferable architectures for scalable image recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8697–8710, 2018

[5] Wu Bichen, et al., "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 10734–10742, 2019

[6] Srinivas Sai, et al., "Hardware aware neural network architectures using fbnet". arXiv preprint arXiv:1906.07214, 2019

[7] **Parsa, Maryam**, et al. "Pabo: Pseudo agent-based multi-objective Bayesian hyperparameter optimization for efficient neural accelerator design". In ICCAD, International Conference on Computer-Aided Design, pages 1–8, 2019

[8] **Parsa, Maryam**, et al., "Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design", Frontiers in Neuroscience, 14:667, 2020

[9] **Schuman, Catherine D.**, et al., "Non-traditional input encoding schemes for spiking neuromorphic systems," in IJCNN: The International Joint Conference on Neural Networks, Budapest, 2019

ASCR Workshop on Co-Design White Paper: Hardware/Software Co-Design to Enable Trustworthy Data Domains for HPC

Sean Peisert, Berkeley Lab and UC Davis, `sppeisert@lbl.gov`
Jason Lowe-Power, UC Davis, `jlowepower@ucdavis.edu`
Venkatesh Akella. UC Davis, `akella@ucdavis.edu`

Topics: architectures, emerging technologies

Challenge Our main research question is *how to co-design hardware and software security mechanisms that will allow HPC centers to implement security policies such that data providers can give usable access to their sensitive data sets to data scientists.* Data useful to science is not shared as much as it should or could be, particularly when that data contains sensitive information. Examples of this include protected health information (PHI); IP addresses or data representing the locations or movements of individuals, containing personally identifiable information (PII); the properties of chemicals or materials, and more. Two drivers for this reluctance to share are concerns of data owners about the risks of sharing sensitive data, and concerns of providers of computing systems, including HPC centers, about the risks of hosting such data. And yet, as emphasized widely in scientific communities, finding ways to make sensitive data available is vital for advancing scientific discovery and public policy.

At the largest scale, scientific computing is characterized by massive datasets, distributed, international collaborations, and HPC centers such as those sponsored by DOE SC/ASCR. And, for HPC centers, when sensitive data is used, secure computing options available are limited in scale and access [4]. The problem can be particularly acute for HPC centers because such centers host and process data at the largest scale and therefore assume commensurate risk [6, 7, 3]. Today, where remote access to data is permitted at all, significant technical and procedural constraints may be put in place. However, even with these security protections, traditional enclaves still require implicitly trusting system administrators, and anyone with physical access to the system containing the sensitive data, thereby increasing the risk to and liability of an institution for accepting responsibility for hosting data. This security limitation can significantly weaken the trust relationships involved in sharing data, particularly when groups are large and distributed.

Opportunity We must consider mechanisms for providing security guarantees as the next generation of leading-edge DOE facilities' hardware and software are designed. There is nascent support for hardware *trusted execution environments (TEEs)*, but further co-design will allow compute providers to significantly change approaches to trust relationships involved in secure, scientific data management. TEEs can be used to maintain or even increase security over traditional enclaves, at minimal cost to performance in comparison to computing over plaintext. TEEs can isolate computation, preventing even system administrators of the machine in which the computation is running from observing the computation or data being used or generated in the computation, including even from certain "physical attacks" against the computing system. They can implement similar functionality as software-based homomorphic and mutiparty computation approaches, but without the usability issues and with dramatically smaller performance penalties.

Common commercial TEEs today include ARM's TrustZone, Intel's SGX, and AMD's Secure Encrypted Virtualization (SEV). In addition, there now exist RISC-V-based open-source hardware efforts such as Keystone [2], which give DOE Labs the opportunity to co-design and demonstrate alternative TEE concepts that overcome the limits of current practice to meet DOE/HPC needs. Keystone, and related efforts, carry both the promise of broadening the scope of processors that contain TEEs, while also being open source and possible to formally verify. However, RISC-V based TEEs have not yet been developed that target scientific computing. Most likely, an entirely new TEE architecture tailored for HPC will be needed, which is our aim.

We aim [5] to develop new approaches to addressing shortcomings of existing processors that make the use of TEEs in HPC an obvious and natural extension to the way that data is secured in HPC environments, and leverages a hardware/software co-design effort to accomplish, because solutions will clearly require modifications from architectures, operating systems, runtimes, and communication libraries.

Limits of Current Practice The Linux Foundation's Confidential Computing Consortium [8], Microsoft Azure's Confidential Computing, AWS's Nitro Enclaves, and Google's recent "Move to Secure the Cloud From Itself" demonstrate the interest in TEEs. However, current TEEs are not yet available that are appropriate for the performance requirements and vendor and protocol-specific hardware and software stacks used in HPC. We have been empirically evaluating commercial TEEs for their performance under typical HPC workloads, including both traditional MPI benchmarks, ML/AI benchmarks, and real-world ML/AI applications. Our results [1] show that Intel's SGX has significant performance limitations, but AMD's SEV imposes minimal performance degradation on single-node operation. However, low-latency communication between SEV nodes is currently impossible, making most HPC also impossible. Further, SEV requires using virtual machines which can impose high virtualization overhead.

Current hardware TEEs and TEE software environments are designed for either client and IoT devices or cloud systems; whereas HPC systems have different programming environments and system constraints which should be exploited to co-design a higher-performance and easier-to-use secure environment. Moreover, current TEEs are restricted to just a single CPUs, whereas HPC infrastrastructures are increasingly *heterogeneous* with CPUs, GPUs, FPGA, and domain-specific accelerators.

Our Approach We argue that the dichotomy between the usage model (software view) and the implementation on today's hardware architectures (hardware view) is the fundamental obstacle to designing secure HPC systems that needs to be overcome. Software has a single (unified) view of data with an understanding of what is sensitive, whereas a hardware implementation of an application results in data distributed across multiple, fine-grained "silos" in the form of cores, memory, and communication and I/O subsystems. Enforcing *isolation*, the core functionality of a TEE, involves restricting the ability to share with a combination of hardware/software mechanisms such as physical memory protection registers, security monitors, different "modes" of operation, etc.. This bottom-up approach is problematic when an application runs across multiple nodes (especially accelerators) and third-party network and I/O subsystems. The semantics of sharing in the OS on a core cannot be extended to a heterogenous computing infrastructure, which means we need to replicate parts of the OS at different places. This becomes cumbersome with third-party hardware/firmware.

Our insight is that HPC applications do not benefit by fine-grain resource sharing via time-multiplexing that is offered by today's hardware and OSes. We envision a *data-centric* approach to secure HPC that is based on co-designing the hardware, software, key exchange and attestation protocols around *encrypted data domains*, that delineate data sharing boundaries in memory. Capability-based approaches, such as CHERI, try to isolate different pieces of software. Our key idea is to replace fine-grain software compartmentalization with an alignment of architectures around a data-centric view, or how data moves through the system and enforces checks. Unlike CHERI, this naturally extends to heterogeneous computing.

Timeliness or Maturity HPC centers have made it clear that there is a need for enabling processing of sensitive data with the appropriate security protections. Current technical and procedural approaches are functional but leave large gaps both in security and usability. TEEs represent a valuable solution for enabling computation, including computation previously relegated to slow cryptographic operations, such as multi-party computation, without trusting system administrators. Commercial TEEs exist and are used in the cloud but have significant performance limitations for HPC use. Open-source hardware, such as the RISC-V-based Keystone represents an opportunity to design and build new solutions specific to HPC needs.

References

[1] A. Akram, A. Giannakou, V. Akella, J. Lowe-Power, and S. Peisert. Performance Analysis of Scientific Computing Workloads on Trusted Execution Environments. In *IEEE Int'l Parallel & Distributed Processing Symp.*, 2021.

[2] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song. Keystone: An Open Framework for Architecting Trusted Execution Environments. In *Proc. European Conference on Computer Systems (EuroSys)*, 2020.

[3] S. Peisert. Security in High-Performance Computing Environments. *Comm. ACM (CACM)*, 60(9), Sept. 2017.

[4] S. Peisert. An Examination and Survey of Data Confidentiality Issues and Solutions in Academic Research Computing. Trusted CI Report — `https://escholarship.org/uc/item/7cz7m1ws`, September 2020.

[5] S. Peisert. Trustworthy Scientific Computing. *Communications of the ACM (CACM)*, 64(5), May 2021.

[6] S. Peisert *et al.* ASCR Cybersecurity for Scientific Computing Integrity. Technical Report LBNL-6953E, U.S. Department of Energy Office of Science report, February 2015.

[7] S. Peisert *et al.* ASCR Cybersecurity for Scientific Computing Integrity — Research Pathways and Ideas Workshop. Technical Report LBNL-191105, U.S. Department of Energy Office of Science report, September 2015.

[8] F. Y. Rashid. The Rise of Confidential Computing. *IEEE Spectrum*, 57(6):8–9, 2020.

Rethinking Performance Monitoring for Co-Design Computer Systems with Heterogeneous Nodes

Ivy B. Peng
Lawrence Livermore National Laboratory
peng8@llnl.gov

Maya B. Gokhale
Lawrence Livermore National Laboratory
gokhale2@llnl.gov

Topics: codesign methodologies, architectures, emerging technologies

New accelerators and chiplets are increasing the possible design space of next-generation supercomputers. Future HPC systems are likely to leverage specialized hardware to accelerate specific algorithms in scientific applications. Specialized accelerators can be more cost- and power-efficient for accelerating suitable algorithms, while scaling up the clock frequency of general processors are becoming prohibitively expensive. Today, GPUs have proven effective in boosting dense linear algebra. However, a substantial portion of applications critical to the DOE is inherently sparse, data-dependent, and irregular, which are known to perform inferior on GPUs. They require different types of accelerators. For instance, accelerators with logics in-memory or logics packed near memory, target to accelerate many memory-intensive algorithms. Besides, neuromorphic chiplets could speedup brain-inspired neural networks that feature low-power, resilient, and unsupervised local optimization. Reconfigurable hardware like field-programmable gate array (FPGA), supports flexible deployment of accelerators that can adapt at runtime and coarse-grained reconfigurable architecture (CGRA) could provide energy-efficient acceleration of data-flow graphs.

Architectural options of incorporating various types of accelerators on a computer system can be explosive. Future systems are likely to employ heterogeneous nodes, in contrast to the homogeneous nodes on existing supercomputers. Today, even on GPU-accelerated supercomputers, nodes are still homogeneous – each consisting of a fixed configuration of resources, e.g., two CPUs, six GPUs, and 256 GB RAM. Imagine if now each node is equipped with multiple types of accelerators. The majority of jobs are unlikely to utilize all these provisioned resources in each node concurrently. Hence, overprovision and under-utilization become a significant problem if future computer systems continue with homogeneous nodes. Two disruptive directions may address this challenge: allowing multiple jobs to share a node or employing disaggregated architecture. Both require tremendous efforts for revamping programming systems. Instead, a moderate architectural option is to have heterogeneous nodes – each node may be equipped with a different type of accelerators and resources, e.g., large memory or small memory capacity, slow or fast network switches – and supporting reconfiguration of resources based on jobs. Such architectral options have already been adopted in the Cloud [2], which provides experience and lessons for codesign HPC systems. Figure 1 illustrate one exemplar scenario of reconfiguring resources for two jobs.

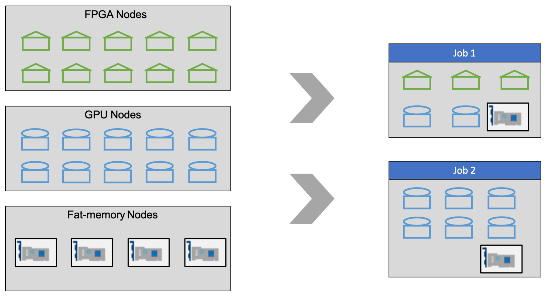


Fig. 1. An example of resource reconfiguration for two jobs

Reconfiguration of resources requires rethinking performance monitoring when codesign between hardware and software stack. On current systems, resources are bound to the job once a job is launched. Thus, no special design considerations on performance monitoring are needed for resource reconfiguration. However, a computer system that supports reconfigurable resources will need a new set of performance counters and metrics to guide resource reconfiguration. For instance, if system monitoring detects that a job has low utilization of memory resources, the unused memory resources can be reconfigured for other memory-intensive jobs. Also, system-wide monitoring is needed to identify unbalanced configurations of different resources, e.g., if system monitoring discovers that computation on accelerators is throttled by a low influx rate of data packets on the network, the job may be reconfigured with nodes with faster network switches. Moreover, codesign would produce an efficient software stack that speeds up the path from performance counters to the reconfiguration procedure.

I. Research Opportunities and Challenges

1) There is a need to use realistic system-wide usage data to guide design space exploration for accurate prediction. Today, many HPC facilities have already been equipped with system monitoring capabilities and capture system-wide usage data. For instance, LDMS [4] provides low-overhead monitoring capability and the LLNL Sonar project provides a centralized database of these realistic datasets. Realistic usage data on current supercomputers can provide a picture of how applications utilize different resources, including accelerators, CPUs, memory, I/O, and network. However, the challenge is that the collected data reflects the current architectural choices. Thus, performing informed extrapolation for new architectural options at the system-level becomes necessary. One direction is to identify metrics that are needed for reconfiguring the resources of a job. Another direction is to identify the appropriate scope for collecting these metrics, which trades off the data volume and coverage for reconfiguration decisions. Therefore, new methodologies must be developed to extract such metrics from the raw usage data and extrapolate them onto new architecture choices for design space exploration.
2) There is a need to codesign performance monitoring and software stack to handle performance monitoring data. Tremendous datasets could be generated from additional counters that come with new accelerators and devices on heterogeneous nodes and also from new metrics added for resource reconfiguration. The codesign should target two performance objectives – first, the massive amount of data needs to be processed in time to assist the reconfiguration timely, second, processing the data should not interfere with other tasks on the computer system. One possible research direction is to have streamlined software stacks that 'fast-fuse' the data path from performance counter to reconfiguration procedure. For instance, expose low-overhead interfaces to OS and software stack. Another direction is to explore in-line compression to reduce the data size or employ an out-of-band network to avoid interference on the shared network. As the number of counters and metrics grows, new opportunities like including dedicated hardware for in-transit analysis/inline compression of performance metrics to accelerate software decisions are emerging.
3) New methodologies for leveraging performance monitoring data for resource configurations need to be developed. The codesign needs to determine the number of different node types and the specific resources configured in each node to meet the performance target of most applications on a system. For instance, if the target workloads have more memory-intensive applications than compute-intensive applications, more nodes should be configured with near-memory accelerators and large memory capacity. The realistic usage data would guide the selection of such configurations so that the selected architectural options can satisfy the performance requirement of a targeted portion of works, e.g., 95% of jobs would be configured with their preferred resources while all jobs should be able to make progress on available resources. New methodologies could also be derived to balance the benefit and overhead of resource reconfigurations. Such use cases are common on cloud and data center, e.g., re-routing in fat-tree networks depends on network traffic statistics that are collected at the device level [1] and reconfigurable fabrics reconfigure routing when faults are detected [3].

References

[1] Li, Y., Pan, D. OpenFlow based load balancing for fat-tree networks with multipath support. In Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary (pp. 1-5).

[2] Lee, Gunho, and Randy H. Katz. "Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud." HotCloud 11 (2011): 4-8.

[3] Putnam, Andrew, et al. "A reconfigurable fabric for accelerating large-scale datacenter services." 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). IEEE.

[4] Agelastos, Anthony, et al. "The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications." SC'14. IEEE.

Automated Codesign of Domain-Specific Hardware Accelerators and Compilers

Priyanka Raina, Fredrik Kjolstad, Mark Horowitz, Pat Hanrahan, Clark Barrett, Kayvon Fatahalian
Stanford University, praina@stanford.edu

Topics: Architectures, applications, programming systems and codesign methodologies.

I. Challenge

With the slowdown in technology scaling, domain-specific hardware accelerators will play a major role in improving the performance and energy-efficiency of computing systems. In their Turing Lecture, John Hennessy and David Patterson make a strong case for this trend, and predict that it will lead to a new golden age of computer architecture [6]. However, because the applications that run on these systems, such as image classification, speech recognition, language modeling, recommendation systems and scientific computing, are evolving rapidly with advances in machine learning, the accelerators must be programmable, to avoid quickly becoming obsolete. Such accelerators require a complete compiler system in order to be useful, and this compiler must get updated as the accelerator hardware evolves. The methodology for evolving accelerators, and more importantly their compilers, is more or less a completely manual process today, where large engineering teams study the accelerator architecture in detail and make the necessary modifications to the compiler and the low-level libraries to leverage the accelerator. Because of the large overhead of maintaining the entire software stack, real-world usage of an accelerator lags far behind its design. A key challenge, therefore, is to automate the co-design of programmable accelerators and the compilers that map applications to them, for fast-changing application domains.

II. Opportunity

We propose to tackle this challenge with CGRA accelerators and compilers that adapt as the hardware evolves.

A. *CGRA as an Accelerator Template*

Our approach to solving this problem has been by using coarse-grained reconfigurable arrays (CGRAs). A CGRA is similar to an FPGA but with larger compute and memory units, and word-level interconnect as shown in Fig. 1. By tuning the amount of configurability in these units and the interconnect, we can create more specialized (closer to ASICs) or more general-purpose accelerators (closer to FPGAs). Thus, a CGRA provides a standard accelerator template for a compiler to target. For example, a CGRA specialized for neural networks would look similar to a hand-designed neural network accelerator like TPU [7] with compute units implementing multiply-accumulate operations, and the interconnect supporting systolic connections between them. To map applications to CGRAs, we have created a compiler shown in Fig. 2, which takes applications written using a high-level library such as Halide [10], lowers it to a dataflow graph based intermediate representation (IR) called CoreIR [3], and then maps, places and routes the graph onto the CGRA. Using this compiler, we can accelerate a wide range of dense linear algebra applications, such as those in image processing and machine learning, on our CGRA and achieve 7 to 25× lower energy than an FPGA.

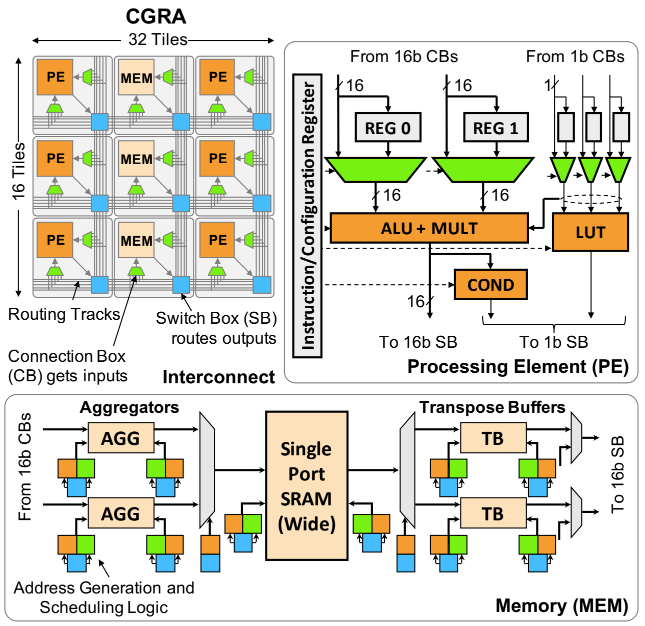


Fig. 1: A baseline CGRA architecture with processing element (PE) tiles, memory (MEM) tiles and a statically-configured interconnect.

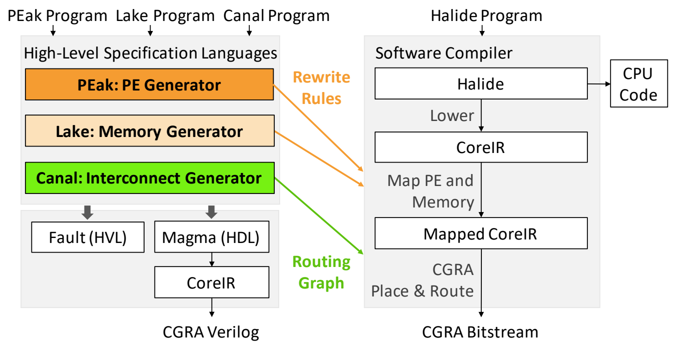


Fig. 2: End-to-end hardware generation and software compilation flow, starting with programs written in PEak, Lake, Canal, and Halide.

B. Accelerator-Compiler Codesign

The key insight to our approach is that, unlike previous work, our compiler *automatically updates* as the CGRA hardware evolves. We achieved this by creating mini specification languages—PEak for processing elements, Lake for memories, and Canal for interconnects—for formally specifying the hardware units, and then from those specifications automatically deriving both the hardware implementation and the collateral needed by the compiler as shown in Fig. 2 [1].

For example, the compiler for PEak, which is our specification language for processing elements (PEs), generates RTL Verilog, a functional model, and the rewrite rules the application compiler needs to map applications, all from a single PE specification. One of the primary enablers of PEak is its ability to leverage advanced SMT (Satisfiability Modulo Theories) solvers [2]. The PEak compiler synthesizes a collection of rewrite rules from a compiler IR (like CoreIR) to a PE ISA by finding an instruction in the ISA that is formally equivalent to one or more instructions in the IR. Similarly, from Lake specifications we generate memory hardware with programmable addressing logic, and the collateral needed by the compiler to map access patterns from applications to these memories. We perform this mapping from access patterns to configuration of address generators also using SMT solvers. Finally, Canal takes a set of (potentially heterogeneous) PE and memory cores and a specification of the interconnect. It generates the hardware, the routing graph that place-and-route tools need to map the dataflow graph onto the generated hardware, and the configuration bitstream that implements the routing result on the hardware, from the specification. As a result, a change in the design of any component automatically propagates through the flow to affect dependent components without manual intervention, and the compiler continually updates with the hardware.

C. Large-Scale Automated Design Space Exploration

Architects often explore many alternatives when designing an accelerator to achieve the best performance, power and area trade-offs. They analyze application kernels to find common sequences of operations that they can make faster or more energy-efficient. This is often done incrementally by proposing a design change, implementing it, then reevaluating the efficiency. A major impediment to design space exploration is implementing the software changes needed to compile the application to the new accelerator. The techniques we describe make it easy to modify an accelerator using PEak, Lake and Canal, and automatically derive a code generator so that the application can be compiled. This enables quick iterative design. Using such hardware-compiler codesign approaches, there is an opportunity to automate large-scale design space exploration (DSE) of accelerator architectures.

As a step in this direction, we are creating a PE DSE framework, that analyzes application graphs using subgraph mining [4] and maximal independent set analysis [5] and generates an ordered list of frequent subgraphs. It then uses subgraph merging [9] to merge several frequent subgraphs to generate a candidate PE graph, which it automatically converts into a PEak specification. From this, the PEak compiler generates both PE hardware and the rewrite rules required by the application mapper as described before. Finally, it synthesizes the CGRA with these specialized PEs and evaluates it using the mapping produced by the compiler. Using this method, we show that optimizing the PE for image processing reduces area by 29.6% to 32.5% and energy by 44.5% to 65.25%. Building on this initial work, given a set of applications in a domain, we hope to automatically produce an accelerator specialized for that domain. Finally, reinforcement learning techniques like [8], in conjunction with such a system, are very promising for performing fast DSE.

III. Timeliness

With the slowdown of Moore's law, hardware specialization is the most promising technique for continued improvement of scientific computing systems. The lack of a structured approach for evolving the software stack, as the underlying hardware becomes more specialized, has been one of the biggest impediments to its adoption. Our approach provides a systematic way of thinking about accelerators as specialized CGRAs and employs a combination of new programming languages and formal methods to automatically generate the accelerator hardware and its compiler from a single source of truth. Furthermore, it enables the creation of DSE frameworks that automatically generate accelerator architectures that approach the efficiencies of hand-design ones, with a significantly lower design effort. This has the potential to massively improve productivity of hardware-software engineering teams and enable quicker customization and deployment of complex accelerator-rich computing systems.

References

[1] R. Bahr, C. Barrett, N. Bhagdikar, A. Carsello, R. Daly, C. Donovick, D. Durst, K. Fatahalian, K. Feng, P. Hanrahan, T. Hofstee, M. Horowitz, D. Huff, F. Kjolstad, T. Kong, Q. Liu, M. Mann, J. Melchert, A. Nayak, A. Niemetz, G. Nyengele, P. Raina, S. Richardson, R. Setaluri, J. Setter, K. Sreedhar, M. Strange, J. Thomas, C. Torng, L. Truong, N. Tsiskaridze, and K. Zhang. Creating an agile hardware design flow. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[2] C. Barrett, P. Fontaine, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2016.

[3] R. Daly and L. Truong. Invoking and linking generators from multiple hardware languages using coreir. In *WOSET*, 2018.

[4] M. Elseidy et al. Grami: Frequent subgraph and pattern mining in a single large graph. *Proc. VLDB Endow.*, March 2014.

[5] H. Cheng et al. *Mining Graph Patterns*. Springer US, 2010.

[6] J. Hennessy and D. Patterson. A new golden age for computer architecture. *Commun. ACM*, 62(2):48–60, January 2019.

[7] N. P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, page 1–12, New York, NY, USA, 2017. Association for Computing Machinery.

[8] A. Mirhoseini et al. Chip placement with deep reinforcement learning, 2020.

[9] N. Moreano et al. Efficient datapath merging for partially reconfigurable architectures. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005.

[10] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *SIGPLAN Not.*, 48(6):519–530, June 2013.

Vision for Co-designing a Unified-Memory Centric Heterogeneous Node Architecture

Sivasankaran Rajamanickam,[1] Tushar Krishna,[2] and Simon Hammond[1]
[1]*Center for Computing Research, Sandia National Laboratories*[a]*, Albuquerque, NM 87185, USA*
[2]*Georgia Institute of Technology, Atlanta, GA, USA*

I. INTRODUCTION

Current high performance computing applications are reliant on GPU accelerators with the host CPU orchestrating the communication. With the exception of Fugaku, the top supercomputers are all accelerator-based systems that support application needs by exposing massive amounts of concurrency and some mechanism to reduce memory access overheads (e.g. coalesced memory access). The emergence of societal uses for machine learning (e.g. recommender systems, natural language processing) and recent successes in the scientific applications (e.g. protein folding) is resulting in the rise of special-purpose hardware that can accelerate such applications. It should be noted that not all applications are able to exploit current accelerators equally well. The disruptive changes in the architecture space provide an opportunity for such applications to benefit by taking advantage of the new spatial or data flow hardware emerging for machine learning use cases, or by using an FPGA or CGRA style accelerators. As a result, we envision a heterogeneous node architecture where several of these compute units are present on a single compute node (as IPs on a chip or as separate chiplets on a package). Such a heterogeneous node will benefit current accelerator-based applications, applications that do not do well on current accelerators, and emerging applications that combine traditional science use cases with machine learning. The rest of this short paper describes our vision for such an heterogeneous node, potential problems we foresee from a co-design perspective and one idea for a future heterogeneous node. This white paper will touch on several topics that are appropriate for the workshop.
Topics : *Architectures, programming systems, and emerging technologies.*

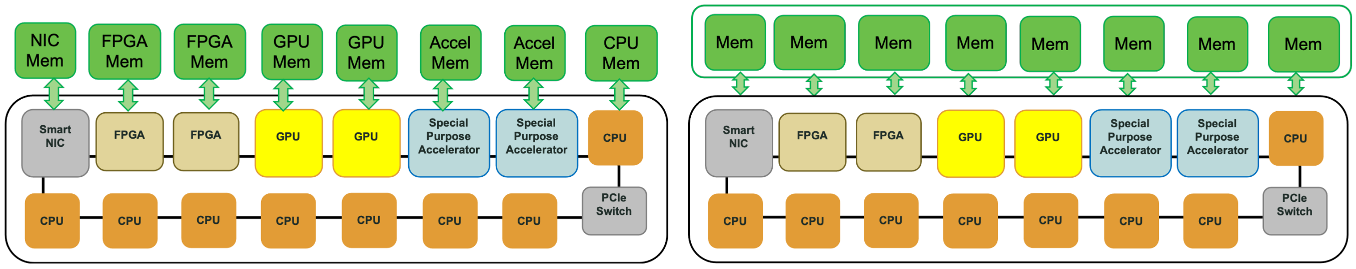
II. CHALLENGE DESCRIPTION

Office of Science's Advanced Scientific Computing and Research funded projects such as ARIAA are focused on developing and modeling novel data flow accelerators, specialized algorithms for such accelerators, and programming models for them. There are hundreds of millions of dollars of venture investments that are also focused on developing machine learning accelerators. We begin with the premise that such efforts are going to be successful in developing such accelerators. Our focus is to look beyond the challenge of designing special-purpose accelerators and instead how to design a future heterogeneous node that has more than one such accelerator in it.

If we extrapolate the current developments in hardware specialization for performance/energy-efficiency, and chipletization for reducing manufacturing cost, we might end up with a heterogeneous node as shown in Figure 1a. This shows several components developed from different vendors assembled together to arrive at a heterogeneous node. This is not an extreme use case as we already have CPU/GPU systems, special purpose accelerators and NICs with their own memory. While such a node design will still offer tremendous benefits to current CPU/GPU systems, such a design will have several disadvantages as well.

- The number of available memory spaces and the requirement to move data/instructions back and forth between them could dominate the performance and could seriously limit the benefits from using special-purpose accelerators. For example, the latency cost of launching several thousand kernels, reconfiguring the accelerators for them and/or returning to host communicate through a NIC is not a scalable approach.
- Efficient use of all the available compute resources *concurrently* will require significant changes to programming models and applications. In addition, maintaining an application that can *explicitly* manage data in all different memory types will be challenging. For example, applications currently explicitly use four different memory types in the CPU/GPU accelerator systems (CPU memory, GPU memory, Unified Virtual memory, and host pinned memory).
- The asymmetric view of memory and network where some compute units are "closer" to certain memory or network creates scaling bottlenecks.
- Coherency issues arising out of having multiple memory spaces have to be managed explicitly in hardware or software.

[a] Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.



(a) Disparate Memory View (b) Unified Memory View

FIG. 1: ASCR Co-design vision for a heterogeneous HPC node built using CPU/GPU/FPGA/Accelerator/NIC chiplets: where we don't want to be (left) and where we want to be (right).

- Network communications become even more expensive when there are several levels of data movement from an accelerator to a host CPU and then to a smart-NIC before actually touching the wire.

III. OPPORTUNITIES FOR CODESIGNING AN UNIFIED MEMORY-CENTRIC NODE ARCHITECTURE

An important research direction is to focus on the co-design of a node architecture that will be centered around an unified memory, where all compute units and the NIC will have a "symmetric" view of the memory. One idealistic view of such a node is shown in Figure 1b. The ideal characteristics of such a node will be:

- Low latency costs in launching a kernel on an accelerator with no copies of data/instruction needed from one memory to another
- To be able to setup and communicate from any compute unit without the requirement to go to a host and do it with minimal latency
- Ability to avoid multiple coherence domains
- Ability to program all the compute units on the node seamlessly without explicitly managing multiple memory spaces

We see significant challenges and opportunities in co-designing such a heterogeneous node. The primary problem is that the right accelerators to mount on such a platform might vary significantly based on the target applications. The rise of open-source hardware designs provide a significant opportunity for Department of Energy (DOE) to participate in efforts to influence design of accelerators that serve the needs of DOE applications and not just rely on industry developed solutions. A second opportunity that naturally arises out of such a co-design effort is to arrive at a node architecture with the right mix of CPUs, GPUs, FPGAs and special-purpose accelerators that is targeted towards high performance computing use cases. Leveraging interfaces such as TileLink being developed by RISC-V or Arm's AXI might play a role in building scalable chiplet-based platforms (similar to IP-based SoCs today). This is not entirely orthogonal to industry interests in developing customized compute nodes for cloud services. The node design that arises out of such an effort can also influence other co-design activities in microelectronics and other use cases such as embedded system design for constrained applications. Such a system could be made easier to program with a task runtime that can schedule tasks on all the heterogeneous compute units. This will result in an opportunity to lead next-generation node programming systems. These efforts could benefit from DOE's tools in simulations, expertise in co-design of micro-architecture, programming models, compilers, algorithms and HPC applications. Another challenge relates to scaling out to larger number of HPC nodes. Here, emergence of technologies like photonics offers promise in potentially extending the shared memory paradigm across several nodes.
Timeliness: This idea is timely for co-design work to start right away as hardware technologies such as special purpose accelerators, programming models, simulation tools and applications are getting ready for a heterogeneous node. Such a co-design effort could directly benefit the science workloads of next decade in two aspects. They will benefit from the improved performance of such a node and the ease of programming a node with unified view of memory.

Domain Specific Co-Design beyond Exascale

Sanjay Rajopadhye (CSU) & Nirmal Prajapati (LANL)

HPC suffered a huge disruption 30 years ago when supercomputer manufacturers moved away from custom designed chips to commercial, off-the-shelf (COTS) microprocessors. It was as if Formula 1 race cars no longer had the most advanced custom designed engines, but were "merely assembled," albeit in very clever ways, from the same components that went into mass produced cars. Although many supercomputer vendors went out of business, the HPC *user community* was not significantly affected. The surviving supercomputer vendors switched to COTS, and focused on advanced engineeding for building supercomputers from them (primarily the interconnection networks). Twenty years ago the end of Denard scaling caused another disruption, starting the trend towards increasing on-chip parallelism, and to programmable GPUs. HPC users were again shielded, and simply rode the rising tide of improved performance.

The previous decade has seen another disruption, but with a few new twists. HPC no longer dominates large scale computing as it used to. The global cloud computing market crossed $10B in 2020, surpassing the HPC market. Its size was valued at USD 266.0 billion in 2019 and is expected to expand at a Compound Annual Growth Rate (CAGR) of 14.9% from 2020 to 2027. The AI/ML deep learning (DL) revolution, enabled by the extrapolation of increasing on-chip parallelism (GPUs and accelerators) triggered this exponential growth, and this segment stands today at over $1400B. Moreover, typical HPC workloads are very different from cloud computing, although AI technology is being increasingly integrated into HPC scientific algorithms and applications.

HPC is also unique in another sense. The end users do not pay for their resource usage. Rather, because of high stakes of "national importance" in most indistrialized countries, it is the taxpayers that pay, regarless of where they live. Although users always seek efficiency in order to best utilize public resources, they do not have as much "skin in the game" as say programmers writing applications on commercial clouds, data-centers and/or users of private clusters in industry, where performance that is significantly poorer than that of a competitor may lead to bankruptcy.

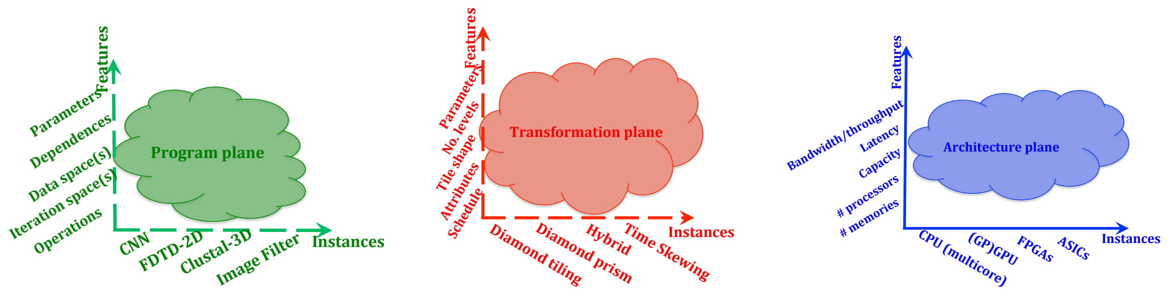
In light of the above context, the question is whether a relatively small market, albeit one with deep pockets, can drive the dominant marketplace to a specialization of the entire computing stack, geared specifically towards its particular needs. Our position is that this is possible if HPC carves out enough of a niche in the ecosystem. We outline the key elements of one possible scenario.

- ***First***, the emergence of chiplets indicates that it is possible to envision specialized accelerators specifically for HPC (e.g., those for full-precision convolutions).

- ***Second***, and at the other end of the spectrum, it is essential to involve algorithm designers and applied mathematicians into the loop, and this can only be done by raising the level of abstraction to more mathematical formalisms (e.g., equations, tensor notation, etc.) as complete, *declarative* specification of what is to be computed.

- ***Third***, bridging this gap requires a complete tool chain involving better programming language abstractions, extensible compiler frameworks, and standardized accelerator interfaces. Much of this was envisaged and accomplished in previous DoE codesign efforts, but the above two elements render the challenge more difficult. This necessitates *extreme* domain specificity.

Because this is such a deep stack, the challenge is how to fit the choices at all the levels seamlessly (note that raising the level of abstraction requires a very sophisticated and fully automatic toolchain). Architectures are constantly evolving, leading to a corresponding evolution of performance optimization strategies. The target platform affects not only the parallelization methods but also the tool-chains used. Even within a narrow domain such as dense stencils there is a huge design space (see Figure 1): algorithms/applications may have a wide range of parameters, making them either intensely bandwidth bound or heavily compute bound, limited only by the hardware's operational throughput. This significantly affects the algorithms used to obtain the best solution.



(a) Programs (b) Transformations (c) Architectures

Figure 1: The Domain-Specific Design Landscape for the "stencil" motif. Programs (a) are described by a small number of features. Similarly the mappings/transformations (b) thet we apply are drawn form a small set, each one parameterized by a set of features. Architectures (c) are also drawn from a similarly small set, also parameterized by their features. In each plane, the features may be hierarchical.

ReCoDe faces too many design choices in too many dimensions. We posit that the way out of this very tall stack is by making it narrow, i.e., through ***domain specificity***, namely the use of techniques suitable for specific domains of computational patterns, or what we call "sub-motifs." One benefit of extreme domain specificity is that it would be possible to develop accurate analytical cost (Time/Power/Area/Energy) models for performance prediction and optimization.

Co-Design for Edge-Core Stretched Computing with Integrated Instruments

Nageswara S. Rao, Neena Imam, Seth Hitefield
Oak Ridge National Laboratory
{raons,imamn,hitefieldsd}@ornl.gov

Topics: Emerging technologies and codesign methodologies

I. The Challenge

Science workflows are executed by composing and automating complex scientific computations and experiments to support collaborations among researchers. They enable scientists to remotely execute codes, and collect measurements and steer experiments [1]. Department of Energy's (DOE) science workflows often require large computations executed as batch jobs at remote supercomputer facilities, such as Argonne Leadership Computing Facility (ALCF) and Oak Ridge Leadership Computing Facility (OLCF), and experiments conducted at science facilities such as the Advanced Photon Source (APS) and Spallation Neutron Source (SNS). For example, a tomographic workflow consists of multiple cycles, each requiring measurements and code execution to generate parameters for the experiment in next cycle.

The productivity of current workflows, however, continues to be impeded by the "stop-and-go" steps needed to utilize these geographically separated resources via disparate mechanisms. There is a current trend to mitigate these effects by enhancing the computing capabilities at the instrument edge to (i) locally execute smaller to jobs that do not need remote supercomputers, and (ii) close the latency gap between computations and instruments in collecting measurements and steering experiments. These systems range from specialized NVIDIA EGX-AI systems, to multi-core, hybrid memory systems, to generic micro Data Centers (mDC) at the edge. This development signals a paradigm shift from "flops provided at a distance" to "edge-core stretched computing", but also increases the diversity and heterogeneity of computing platforms. Furthermore, DOE facilities are spread within and across the sites, which adds to the challenge as wide-area networks are now an integral part of both computing and experimentation. Indeed, these computations that span heterogeneous, distributed computing platforms and experimental facilities require an effective ecosystem to make them transparently available to science users that use them and facility providers that federate them.

We propose a co-design framework for integrated computations and instrument operations over an ecosystem of edge-core computing systems, wide-area networking and experimental facilities, for science environments. Under this framework, a science user will be able to develop and execute codes that (a) transparently stretch across computing systems at the edge and remote High Performance Computing (HPC) systems, and (b) access instruments for collecting measurements and steering experiments at science facilities (experiments-in-the-loop). This capability enables an effective use of DOE science facilities by providing a transparent, performance-integrated programming and runtime environments and tools to science users.

II. Enabling Co-Design Components of Ecosystem

Developments in multiple areas contribute to the proposed ecosystem, including: increasing proliferation of computing systems at instrument sites, expanded reach of HPC core capabilities such as MPI and file systems, and increasingly sophisticated instrument control software.

A. Expanding Computing at Instrument Edge

Increased deployment of edge systems, including specialized DGX/EGX machines and generic mDCs, facilitate computations that range from smaller jobs at the edge, e.g. tomographic reconstruction, to large computations at remote HPC systems, e.g. radiation transport simulations for a 3D printed reactor pressure vessel. They also support access to instruments to collect measurements and steer experiments, e.g. neutron scattering images by positioning targets at end stations.

A mDC is a small, modular data center designed to be deployed on location to support compute workloads that do not require the full capabilities of traditional HPC or cloud computing facilities. These vary in size and configuration to match the computing needs: they can be scaled from one to tens of servers and can also integrate heterogeneous accelerator cards such as FPGAs, GPUs, and SmartNICs. A custom mDC may consist of (i) a cluster of dedicated compute nodes with virtualization and container support, (ii) dedicated server with clients for instrument access at the edge facilities, and (iii) dedicated Data Transfer Node (DTN) servers with wide-area networking capabilities, multiple 10GigE connections, and GridFtp services.

Within a computing system, core specialization techniques provide effective executions of complex computations by keeping the computing near the edge to the extent possible and mitigating the latency effects due to remote systems via task-specific core assignment techniques [2]. They are increasingly viable at the edge with the advent of many low-powered processors with multiple cores and increasing core counts.

B. MPI and Lustre at Round the Earth Distances

Current HPC facilities are typically deployed within a site and are supported by core technologies with limited reach, such as MPI and file systems over IB connections. They do not scale well to computations stretched between facilities distributed across the country, e.g. BNL and NERSC separated by thousands of miles; in particular, 2.5 ms IB latency limit restricts its reach to tens of miles. Recently, Lustre file system has been extended to round the earth distances using LNet routers[3], and MPI codes have been shown to scale similar distances [4]. These are indicators that programming elements and file systems can be scaled to support stretched computations, and similar solutions may be co-designed by taking into account the entire ecosystem.

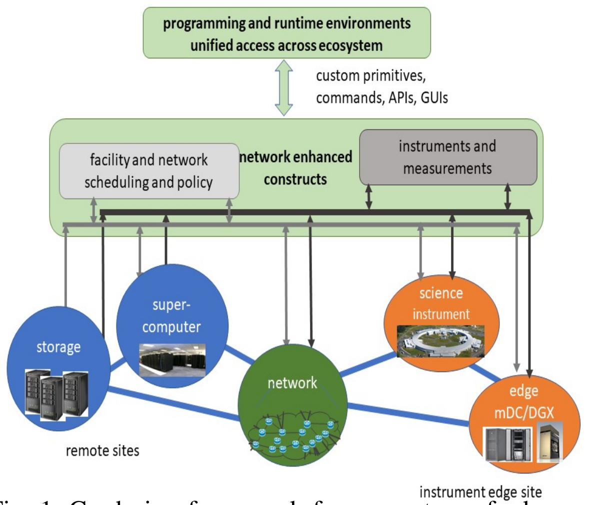

programming and runtime environments
unified access across ecosystem
custom primitives, commands, APIs, GUIs
facility and network scheduling and policy
network enhanced constructs
instruments and measurements
storage
super-computer
network
science instrument
edge mDC/DGX
remote sites
instrument edge site

Fig. 1. Co-design framework for ecosystem of edge-core computing and instruments.

C. *Target Control Using EPICS*

Experimental facilities, such as accelerators and telescopes, run the Experimental Physics and Industrial Control System (EPICS) [5] software, which is an open-source control toolkit for distributed control systems of scientific instruments. It supports application interfaces and networking protocols for hardware components of instruments, such as sensors, motors and detectors. Command line and graphical interfaces provide controlled access for science users to collect measurements and supply configuration parameters for targets used in science workflows.

III. Co-Design Framework and Approach

We propose a co-design framework to support efficient code development and execution environment for stretched computations and instrument operations, as illustrated in Fig. 1, over a time-frame of next few years.

A. *Edge-Core Computing Architecture*

The architecture will be designed with customized edge computing systems deployed at instrument sites with custom local connections to the instruments and connections to core computing facilities over both local- and wide-area networks. These systems will be specially designed to meet the requirements of computations pushed to the edge and provide instrument control. Large data transfers will be supported by edge DTNs with dedicated network connections for the transfer. Within a computing system, this architecture supports task-specific core assignment by exploiting the locality of information and users by focusing on three main components: computation, memory and processor orchestration, and networking [2].

B. *Programming and Runtime Environments*

Transparent software environments will allow uniform access to resources over heterogeneous compute, storage, and instrument systems, by extending basic computing operations to ensure workflow performance over the edge-core architecture. Extended programming primitives enable the development of application and workflow codes in a similar manner to current HPC systems by incorporating complementary co-design of individual computing systems. The run-time environment will support unified execution of application and system codes across the distributed resources.

C. *Integrated Instrument Access using EPICS*

Programming and runtime constructs will be augmented with EPICS command line and graphical interfaces custom designed for the site facility instruments and access policies. Specifically, access to EPICS process variables of target instruments will be provided for measurement collection and target parameter specification by science users.

D. *Networking and Analytics*

The wide-area networking performance will be explicitly incorporated into the co-design to support large and agile data transfers needed for computations and instrument control. Measurement and analytics modules will be developed using machine learning methods [3], [4] to estimate throughput and latency variations to support the optimization and customization of network-enabled primitives and services. Resulting analytics help inform compute and data transfer scheduling decisions to optimize the overall science workflow.

E. *Virtual Design and Test Environment*

Assessing the design options of the ecosystem and the development and testing of its software would require allocations of significant resources and coordination among the sites. It is impractical and ineffective, particularly during early development, to carry out these tasks over production facilities due to the expense and potential for disruptions. We propose a digital twin that replicates the ecosystem by emulating its hosts, networks and instrument systems to assess the ecosystem designs and software stack. It will be based on enhanced Virtual Federated Science Instrument Environment (VFSIE)[6] that emulates a federation using a combination of containers and virtual hosts connected over an emulated network. For computing systems, we will employ simulators (e.g. gem5) and micro-benchmarks to study combinations of efficient memory and networking cores that target communications tasks and more performant cores that target computations.

References

[1] "Software defined networking for extreme-scale science: Data, compute, and instrument facilities," DOE ASCR Workshop, Bethesda, MD, Aug. 2014.
[2] N. Imam, F. Aderholdt, and N. Rao, "Compute, memory, and orchestration: Specialized architecture in edge computing," in *proceeedings of Hotedge '20*, 2020.
[3] N. S. V. Rao, S. Sen, Z. Liu, R. Kettimuthu, and I. Foster, "Learning concave-convex profiles of data transport over dedicated connections," in *Machine Learning for Networking* (E. Renault, P. Muhlethaler, and S. Boumerdassi, eds.), Springer, 2019.
[4] N. S. V. Rao, N. Imam, Z. Liu, R. Kettimuthu, and I. Foster, "Machine learning methods for connection rtt and loss rate estimation using mpi measurements under random losses," in *Machine Learning for Networking* (E. Renault and S. Boumerdassi, eds.), Springer, 2020.
[5] "Experimental physics and industrial control system." epics.anl.gov.
[6] A. Al-Najjar, N. S. V. Rao, and et al, "VFSIE - development and testing framework for federated science instruments." Available from: http://arxiv-export-lb.library.cornell.edu/abs/2101.02184, 2020.

Codesign And Computational Methods

Robert Robey, Nathaniel Morgan, and Jeffery Kuehn

1 Introduction

New hardware can be designed to quickly calculate e^x, $tanh(x)$, and other highly non-linear functions. Such hardware can accelerate existing methods such as machine learning and benefit new methods. Given this, we seek to develop novel hydrodynamic methods that can leverage these new hardware designs to yield robust and more accurate solutions to shock dynamics problems.

2 History and Background

At the start of the computer era, there were both digital and analog computers. Analog computers are an old technology, which consisted of basic electrical components laid out on a large board, programmed by manually adding plug-wiring to connect the components into a circuit designed to have the same governing equations as the physical system the programmer was solving. Because the circuit shared the governing equations of the physical system, it captured the full complexity (functions and derivatives). However, analog computers were limited by: (1) scaling (component count), (2) complexity (designing/wiring the circuit), and (3) component quality vs ideal. Digital technology quickly overtook analog technology with advancements in scale and program-ability. While digital computers are flexible and fast for basic mathematical operations, complex functions and operators must be algorithmically constructed (often with series approximation) from basic operations, constraining both performance accuracy. In the context of a hyperbolic first-order partial differential equation, numerical methods use a linear polynomial across a cell to yield up to second-order accuracy or use a quadratic polynomial to give up to third-order accuracy. However, linear and quadratic polynomials may not accurately approximate the underlying physics, generally driving piecewise fits with discontinuities in one or more derivatives. Improving this requires an increase in the number of polynomial terms at the risk of becoming more prone to overfit. We know that most physics arise from and give rise to differentiable functions. Looking closer, we are generally interested in two shapes. The most common is the "S" or sigmoid shape shown in Figure 1. This sigmoid gives a smooth, differentiable function that is well behaved and avoids the numerical problems that occur with a sharp if-then-else conditions (i.e., 1 or 0). We are also interested in accurately handling the abrupt transition and expansion wave that occurs with shock waves. This physics waveform can be described with a class of functions similar to the Friedlander equation shown in Figure 2.

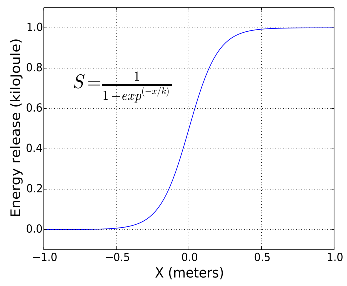


Figure 1: Sigmoid waveforms mimic physical processes such as energy release from combustion

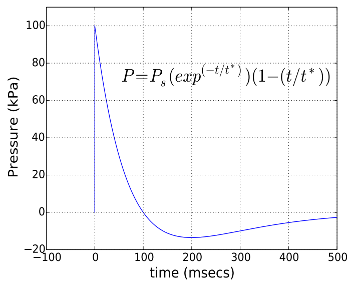


Figure 2: Friedlander equation for a shock wave is based on an exponential form

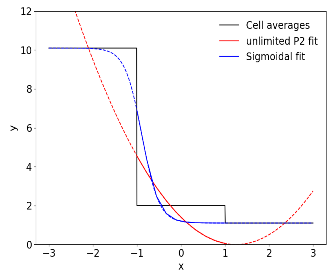


Figure 3: Fitting Sigmoid and Quadratic (P2) functions to a shock wave.

What if digital computers also provided faster transcendental functions such as exp, log, tanh, or even a derivative? Similar to other mathematical operations, multiple functional units for each of these could allow the "retiring" of the operators at a higher rate for work on an array. What could we do with them? We have to go back to the derivation of numerical methods. Mathematically, methods are based on a Taylor series expansion into a polynomial function.

3 Impact on Numerical Methods

A diverse range of models and numerical methods rely on transcendental functions; likewise, new numerical methods can be developed that use these functions. As such, we propose to create a new finite volume (FV) cell-centered hydrodynamic (CCH) method based on sigmoidal reconstructions that can leverage the new hardware for timely simulations with improved accuracy.

The FV CCH method evolves cell-average fields $\bar{U}$ by discretizing the integral form of the governing physics equations. In these methods, surface fluxes $\mathbf{f}^*$ are calculated using a Riemann solver that takes as inputs the discontinuous polynomial fields at the cell surface. Higher-order accuracy is achieved by reconstructing the fields using higher-order polynomials that are built by fitting the neighboring cell-averages. A Taylor-series polynomial is,

$$P(x) = \bar{U} + (x - x_c)a_1 + (y - y_c)a_2 + (z - z_c)a_3... \tag{1}$$

Here, the coefficients $a_1, a_2, a_3, ...$ are found by least-squares fitting neighboring cell averages. The subscript c denotes the centroid of the cell. To ensure robust solutions, the higher-order terms in the polynomial are limited toward zero to ensure that the polynomial remains within the bounds defined by the local and adjacent cell averages. The limiting process delivers robust solutions but can substantially degrade the accuracy. As an alternate approach, we propose to research and develop a revolutionary new FV CCH method that builds bounded reconstructions using a Sigmoidal function given by

$$S(x) = \bar{U}_{min} + \left(\bar{U}_{max} - \bar{U}_{min}\right)\left(\frac{1}{1 + e^{-a_1(x-\widetilde{x})}}\right)\left(\frac{1}{1 + e^{-a_2(y-\widetilde{y})}}\right)\left(\frac{1}{1 + e^{-a_3(z-\widetilde{z})}}\right) \tag{2}$$

The min and max cell average values are U_{min} and U_{max}. The coefficients a_1, a_2, and a_3 are found by non-linear least squares fitting of the neighboring cells averages, and $\widetilde{x}$, $\widetilde{y}$, and $\widetilde{z}$ is found so that $\frac{1}{V_h}\int\limits_{V_h} S(x)dV = \bar{U}$; in other words, the Sigmoidal reconstruction is conservative. The merit of a Sigmoidal reconstruction lies in the improved representation of the spatial variation of a physical shock. As an example, Fig. 3 compares a quadratic polynomial (P2) to a Sigmoidal function that are fit to the cell-average values over [-1,1]. In addition to remaining in-bounds, the Sigmoidal reconstruction is significantly more accurate at representing the spatial field variations whereas the the quadratic (P2) polynomial performs poorly near the sharp transition. The challenges that we seek to solve are to (1) create technologies that quickly build a Sigmoidal reconstruction, and (2) create new FV CCH schemes that are robust and accurate using these reconstructions. This work fits within co-design effort by creating new numerical methods (based on Sigmoidal reconstructions) that can leverage novel chip designs for timely calculations.

Beyond the new hydrodynamic schemes, fast transcendental functions also benefit machine learning (ML) which makes heavy use of exponential-based activation functions and suffers under non-differentiable approximations. A recent paper by Google identified significant improvements to training performance and classification accuracy by replacing the ReLU activation with a continuously differentiable sigmoid-based "drop-in" replacement function: swish(x)=x*sigmod(x). Swish fixes defects in both sigmoid and ReLU while preserving the strengths of each: it is more robust against the vanishing derivative problem which can plague sigmoid in deeper networks and unlike ReLU, it is continuously differentiable leading to faster and more stable training.

4 Path Forward

We have already seen some improvement in the performance of transcendental performance. Vectorized versions of these functions have recently become available in the standard system library. But hardware vendors have not optimized the functions extensively because they are not used. And they are not used in calculations because they are not optimized. If we could get some of these functions down to 10-20 cycles, we would find the exponential form to be superior to the traditional Taylor series polynomial expansion. We propose developing some numerical methods for testing for simulation fidelity in comparison to existing methods and projecting performance on potential hardware designs.

Unified, Open, Multi-Fidelity, Scalable Architectural Simulation

Simon Hammond, sdhammo@sandia.gov
K. Scott Hemmert, kshemme@sandia.gov
Clayton Hughes, chughes@sandia.gov

Arun Rodrigues, afrodri@sandia.gov
Gwendolyn Voskuilen, grvosku@sandia.gov

Topics: architectures, modeling and simulation, emerging technologies, codesign methodologies

Challenge

The architectural design space for HPC systems is becoming increasingly complicated. Existing processors such as CPUs and GPUs will be augmented by custom accelerators, FPGAs, and coarse grain reconfigurable processors. These will be integrated with increasingly sophisticated on-chip networks, off-chip optical and electrical networks, and a dizzying array of 2D, 2.5D, and 3D packaging options. This growing complexity in turn requires increasingly complex design space exploration and architectural simulation tools to enable predictive performance analysis. However, while simulators exist for some of these options individually, they do not work together and thus do not enable full system design space exploration. ***Currently, we lack unified simulation options for many of the hardware architectures and software stacks we need to simulate.***

These complicated systems will grow in scale. They will be comprised of more nodes, each with more sockets, more cores, and more threads than ever before. With disaggregated memory, processing-in/near-memory, and advanced packaging, the traditional barriers between node, socket, and core that enable isolated development will break down. Embarrassingly, many simulators for these devices are still serial [1]**.** ***Currently, simulation and modeling tools lack the speed and flexibility to address these systems at scale.***

Designing these complex heterogeneous systems will require exploration of large design spaces using models at large scale and multiple levels of fidelity. Cycle-level pipeline simulation of a single processor core can incur a slowdown of 1,000-10,000x [2]. Lower-level RTL simulation, necessary for detailed design, can be another one to two orders of magnitude slower. Device-level simulations, which may be required for emerging technologies, will be even slower. ***Currently, detailed simulation is both necessary and impossible to perform at scale.***

Codesign requires open tools and interfaces for both hardware and simulation which cross industry, lab, and academic boundaries. Codesigning these systems will necessarily involve close collaboration between DOE, vendors, and other agencies. Without a common framework for exploration, meaningful collaboration on hardware and software will be curtailed. ***Currently, the architectural modeling and simulation landscape is a hodgepodge of incompatible tools which do not support close collaboration between multi-lab and multi-vendor partnerships*** [3]***.***

Opportunity

The DOE is uniquely poised to address these challenges. Already, the DOE National Labs have an acknowledged multi-decade leadership in parallel multi-fidelity physical simulations across a wide range of domains. We have, both in-house and through existing external collaborations, access to a wide range of hardware simulators spanning system- and node-level to circuit- and device-level.

With this foundation, it is possible to build a simulation framework which is **unified, open, multi-fidelity, and scalable**. The labs are well suited to building this framework. We have existing expertise in developing parallel simulations to address scale and have developed simulators across the necessary levels of fidelity. We also have a long history of creating complex open software.

The primary new technique which needs to be developed is how to seamlessly transition between multiple fidelities of simulation and how to better understand and quantify the tradeoffs between the different levels. Though there are several simulators that can switch between simulation granularities and simulators which can combine low-level and high-level models, the community lacks a rigorous quantitative understanding of the performance/accuracy tradeoff.

Critical to this success is the network of inter- and extra-lab collaborators which DOE has built from decades of architectural leadership. Gaining support from vendors and academics is required to make this infrastructure useful and sustainable. Luckily, there has been substantial interest across the community for these sorts of tools. Many vendors use open simulators such as gem5 and Verilator, but are recognizing the scalability limits they impose.

Timeliness:

There are three major changes that now make a unified framework possible and achievable:

1. A **critical mass** of open simulators and simulation frameworks which have a growing acceptance within industry, government, and academic settings. Tools such as gem5, SST [4], Chisel, Verilator, ESSENT [5], and BigSim will provide the building blocks for a unified simulation framework.
2. Addressing **scalability** is a large enough problem that it cannot be ignored and cannot be tackled through existing serial simulation models. The need to simulate at scale means a unified parallel framework is not just *preferable*, but *required*.
3. A **spate of new** tools for hardware design (HLS languages, HDL languages like Chisel), new packaging approaches (e.g., chiplets), and new economic models (AMD semi-custom) make custom hardware design more accessible than ever before. The DOE has an unprecedented opportunity to take advantage of these trends to increase compute efficiency and capability, but only if proper tools for simulation and modeling are available to guide us.

If successful, this Unified, Open, Multi-Fidelity, Scalable Architectural Simulation infrastructure will provide a large increase in capability for the community. It will allow models to be exchanged between different groups, providing a common language for hardware development and promoting interoperability. It will provide a platform for software development on and in tandem with advanced and emerging architectures. This will create new opportunities for true codesign of hardware and software and allow the next generation of HPC systems to be more efficient and capable.

References

[1] X. Guo and R. Mullins, "Accelerate Cycle-Level Full-System Simulation of Multi-Core RISC-V Systems with Binary Translation. arXiv:2005.11357v1 [cs.AR].," 2020.

[2] A. Saidi and A. Sandberg, "gem5 Virtual Machine Acceleration," December 2012. [Online]. Available: http://www.m5sim.org/wiki/images/c/c3/2012_12_gem5_workshop_kvm.pdf.

[3] A. Akram and L. Sawalha, "A Survey of Computer Architecture Simulation Techniques and Tools," *IEEE Access,* vol. 7, pp. 78120-78145, 2019.

[4] S. Hammond, K. S. Hemmert, S. Kelly, A. Rodrigues, S. Yalmanchili and J. Wang, "Towards A Standard Architectural Simulation Framework," in *Workshop on Modeling & Simulation of Exascale Systems & Applications*, September 2013.

[5] S. Beamer and D. Donofrio, "Efficiently Exploiting Low Activity Factors to Accelerate RTL Simulation," in *Design Automation Conference (DAC)*, San Francisco, CA, July 2020.

Raising the Level of Accelerator Abstraction

Tim Rogers, Purdue University (*timrogers@purdue.edu*)
Milind Kulkarni, Purdue University (*milind@purdue.edu*)

Research topics: Programming Systems and Architectures

1 Challenge

The end of Dennard scaling a decade ago and now the slowing and eventual end of Moore's law has demonstrated that CMOS device technology can no longer provide the gains it once did. As a result, computer systems designers must innovate across higher-levels of the computing stack to improve energy-efficiency, where co-design is increasingly important outside of the traditional embedded space.

Accelerators have emerged as an attractive energy-efficient alternative to CPUs. These accelerators take various forms, but the most universally applicable accelerator is the general-purpose GPU. GPGPUs maintain the same Turing completeness as CPUs and are an order of magnitude more energy-efficient, but have generally maintained a C-like interface and really only work well for dense codes. Enabling the efficient execution of emerging, often *irregular*, algorithms on general-purpose accelerators is a key challenge. Scalable solutions require architects and programming system designers to find innovative ways to work together.

Programmability almost always comes at a cost to efficiency and vice versa. So when hardware designers work with programming language designers, the question becomes, what will each side give up to handle more complex software in efficient ways? Several recent real-world examples of this can be found in contemporary GPUs: Tensor cores for efficient machine-learning processing [1] and ray tracing cores [2] for emerging, irregular graphics workloads.

Ray tracing hardware is particularly interesting, as it efficiently handles a highly-irregular task: ray tracing based scene rendering. Unfortunately, ray tracing hardware is essentially impossible to program for general-purpose or scientific applications, despite its potential uses in irregular high-performance workloads like collision detection [4], nearest-neighbor classification [5], or n-body simulation [3]. The interface to current ray tracing hardware is a definition of a *bounding volume hierarchy* [7], around which the hardware does ray-object intersection. While useful in the context of the narrow ray tracing space, the brittle interface is not suited for general-purpose scientific computing.

Inspired by these real-world examples of high-performance co-design, we take the position that architects and programming systems designers can do better.

2 Opportunities

While companies have done a reasonable job at providing efficient hardware for very specific workloads that drive product sales (i.e. machine learning and graphics rendering), there has been less thought given to more general abstractions that can be applied to irregular, scientific applications. There is an opportunity for architects and system designers from both academia and industry to collaborate on abstractions that are more *domain-specific* than *application specific*, while still allowing for efficient hardware. Hearkening back to original lessons on instruction set architecture design [8] the irregular applications of the future need efficient primitives in hardware, not solutions in hardware.

Specific to our ray tracing example, the interface to the ray tracing cores could be a general tree accelerator, not a ray accelerator. As Gray and Moore observed in 2000, a large number of important kernels, including ray tracing and collision detection, but also important scientific computing applications like nearest neighbor, gravitational force computations, and 2-point correlation, can be thought of as *generalized* *n-body problems* [5]. Rather than thinking about bounding volume hierarchies and ray-object intersection, the hardware interface could be built around general tree traversals, with alternate spatial tree implementations and layouts, application-specific object/subject interaction rules, and generalized truncation conditions. From the hardware's perspective, such a mechanism would have an efficiency similar to the of the ray tracing specific solution — ray tracing with bounding volume hierarchies is a straightforward instantiation of those primitives — while enabling more general computations.

In early work, we showed that this kind of abstraction could lead to efficient, but general, distributed tree traversals [6]; we believe a hardware implementation could provide similar benefits. On the hardware side, we have demonstrated the cost of raising the abstraction level of accelerator code can be high [9]. By designing cross-layer solutions that expose the nature of the abstraction to the lower-levels of the system, minor hardware modifications can be made to make much better use of the highly-efficient accelerator. More generally, we think that it is time to revisit the hardware abstraction layer for accelerators. Rather than focus on complete ISAs (as in GPGPUs), or low-level, limited hardware (as in TPUs), or extremely general but hard-to-program accelerator fabrics (as in FPGAs), a careful selection of generalizable domain-specific primitives could yield efficient hardware for irregular applications.

3 Timeliness

Moore's law is dead. The most viable solution to increase computing capability in the next decade is the use of accelerators. The previous decade saw the birth of GPGPUs as the catch-all accelerator of choice and the last few years of industrial innovation has been largely centered around application-specific hardware components for economically important workloads. However, it is clear that innovation in both the programming system and architecture space is needed to continue this trend, adding generality to the interfaces while maintaining efficiency. The impact of success will be accelerators that that meet the abstraction needs tomorrow, are intuitive to program and demonstrate a level of efficiency on challenging, irregular problems that is impossible to achieve today.

References

[1] NVIDIA Tesla V100 GPU Architecure Whitepaper. `https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf`, 2017.

[2] NVIDIA Turing GPU Architecure Whitepaper. `https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf`, 2019.

[3] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[4] Christer Ericson. *Real-Time Collision Detection*. CRC Press, Inc., USA, 2004.

[5] Alexander G. Gray and Andrew W. Moore. 'n-body' problems in statistical learning. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 500506, Cambridge, MA, USA, 2000. MIT Press.

[6] Nikhil Hegde, Jianqiao Liu, and Milind Kulkarni. Spirit: A framework for creating distributed recursive tree applications. In *Proceedings of the International Conference on Supercomputing*, ICS '17, pages 3:1–3:11, New York, NY, USA, 2017. ACM.

[7] I. Wald. On fast construction of sah-based bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing*, pages 33–40, 2007.

[8] William A Wulf. Compilers and computer architecture. *Computer*, 14(7):41–47, 1981.

[9] Mengchi Zhang, Alawneh Ahmad, and Timothy G. Rogers. Judging a Type by its Pointer: Optimizing GPU Virtual Functions. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operation Systems (ASPLOS)*, 2021.

Composition as a tool for HPC service development

Robert Ross (corresponding author),
Philip Carns, Matthieu Dorier, Kevin Harms,
Robert Latham, Pierre Matri, and Shane Snyder
Argonne National Laboratory
rross@mcs.anl.gov, carns@mcs.anl.gov, mdorier@anl.gov,
harms@alcf.anl.gov, robl@mcs.anl.gov, pmatri@anl.gov,
ssnyder@mcs.anl.gov

Bob Robey, Bradley Settlemyer, and
Galen Shipman
Los Alamos National Laboratory
brobey@lanl.gov, bws@lanl.gov, gshipman@lanl.gov

George Amvrosiadis, Ankush Jain,
Charles Cranor, Greg Ganger, and Qing Zheng
Carnegie Mellon University
gamvrosi@cmu.edu, ankushj@andrew.cmu.edu
chuck@ece.cmu.edu, ganger@andrew.cmu.edu,
qingzhen@andrew.cmu.edu

Neelam Bagha, Dana Robinson,
Jerome Soumagne, and David Young
The HDF Group
nbagha@hdfgroup.org, derobins@hdfgroup.org,
jsoumagne@hdfgroup.org, dyoung@hdfgroup.org

Topics: codesign methodologies, programming systems

Challenge

For decades high-performance computing (HPC) systems have been technology leaders in utilization of massive concurrency, in no small part because HPC applications are composed of software components that provide only the communication, concurrency, and synchronization needed for the task at hand. This approach has enabled the modern DOE portfolio that consists of a rich ecosystem of simulation, data analytics, and learning applications, each with distinct data management and analysis needs.

At the same time, these successful principles of modular reuse, composition, and specialization have not become commonplace in the remainder of the software stack, and in particular as they relate to storage services such as parallel systems. As a result, most HPC platforms provide a “one size fits all” solution to data storage --- a parallel file system --- which has been recognized as a key performance bottleneck for many codes.

Without the benefits of modular reuse, composition, and specialization, custom solutions are simply too expensive to develop and maintain, in time and resources, to justify an ecosystem of services. Additionally, the traditional monolithic approach to these services inhibits adaptation of these services to new platforms. Codesign is dramatically more expensive to engage in when modularity and composition have not been embraced in the software design.

Opportunity

Modular reuse, composition, and specialization have enabled the rapid development of a new generation of data services [1,2], building on technologies such as lightweight threads [3] and remote procedure call abstractions [4]. This model promises to lower the cost of codesign by compartmentalizing changes in reusable modules to adapt to new applications and/or new hardware. Further, this architecture allows low-level components (e.g., mapping data to nonvolatile memory devices) to be codesigned with hardware vendors while application facing aspects (e.g., application interfaces) can be codesigned with application development teams.

Codesign with hardware vendors is critically important to exploit the potential of emerging technologies. Advanced network technologies suitable for traditional messaging as well as low-latency remote procedure call models could provide major advances in communication bound micro-services. Hardware features for lightweight threading and network driven scheduling of threads is another area that should be examined. Smart storage technologies enabled by processing in storage architectures could enable concurrent execution of otherwise high-cost data services concurrent with traditional simulation workloads with minimal noise to the application. System-on-a-chip offers further opportunities to reimagine the composition of data-services enabled by special purpose accelerator technologies that can be efficiently partitioned to different micro-services on-node.

Additionally, the formalization of deployment and configuration facilitated by breaking a service into components and defining dependencies leads to more rigorous and programmatic descriptions of configuration that in turn aid in the application of learning approaches to tuning of these services for specific platforms and to the adaptation of these services in dynamic runtime environments.

Timeliness

This need for new architectural approaches in storage services was noted in a recent ASCR workshop [5], and the growing diversity of applications in the DOE portfolio has pushed the "one size fits all" model to the brink of failure. A number of new data services adopting this methodology and model have been recently developed, showing the promise of the approach as relates to application specialization.

At the same time, disruptive innovation is occurring at the hardware layer. Increasingly, the limiting factor in deploying these innovations is the time it takes to provide a robust software service that makes good use of the innovative capabilities. A nimble, modular, service layer is critical to developing the partnership and connections between storage and application.

Successful research and development of rich compositional methodologies and programming systems in system software will result in a more nimble software stack that is more amenable to codesign for both hardware and applications, more robust through increased reuse, and higher performance through appropriate specialization where it matters.

References

[1] Robert B. Ross et al. Mochi: Composing Data Services for High-Performance Computing Environments. Journal of Computer Science and Technology. 35, 121–144 (2020).

[2] The Mochi Project. https://www.mcs.anl.gov/research/projects/mochi/

[3] Sangmin Seo et al. Argobots: A Lightweight Low-Level Threading and Tasking Framework, IEEE Transactions on Parallel and Distributed Systems (TPDS), Oct. 2017.

[4] Jerome Soumagne, Philip Carns and Robert B. Ross. Advancing RPC for Data Services at Exascale. IEEE Data Engineering Bulletin. 43, 23-34 (2020).

[5] Robert Ross et al. Storage Systems and Input/Output: Organizing, Storing, and Accessing Data for Scientific Discovery. Report for the DOE ASCR Workshop on Storage Systems and I/O. United States. 2018.

A Full-Stack Architecture for Efficient Sparse Computation

Saman Amarasinghe, Joel Emer, Jonathan Ragan-Kelley, Daniel Sanchez
Massachusetts Institute of Technology
{saman,emer,jrk,sanchez}@csail.mit.edu

Fredrik Kjolstad
Stanford University
Department of Computer Science
kjolstad@cs.stanford.edu

Milind Kulkarni
Purdue University
School of Electrical and Computer Engineering
milind@purdue.edu

Topic

Programming Systems and Architectures

1 Challenges

Computer systems have long been designed and optimized for dense computations, i.e., those that operate on dense and regularly structured data. Since the first version of Fortran, with its regular loops and dense arrays, systems have evolved a full stack of techniques for dense computations, spanning languages, compilers (e.g., autovectorization, the polyhedral model), processor architectures (e.g., SIMD units and GPUs, prefetchers), and system and network designs (e.g., interconnects optimized for bulk transfers). Now that systems are becoming specialized to particular domains to improve performance, we are again seeing the same narrow focus on dense computations, like dense deep neural networks. This has given rise to domain-specific languages (e.g., Spiral [12], Halide [13], and TVM [3]), compilers (e.g., TensorFlow [1] and PyTorch [11]), and architectures (e.g., TPUs [8] and NVIDIA Tensor Cores [10]) to scale dense workloads to cloud settings.

We need a similar development for sparse computations. These computations are common because many relations and interactions are sparse. For example, most people are not friends and most neurons in a sparse neural network are not connected. Sparse data structures (graphs, sparse matrices, and trees) and codes leverage this sparsity by encoding and processing only meaningful relations. Sparse applications are ubiquitous in high-performance computing, where they include unstructured grids, sparse linear or tensor algebra, and simulation. They are also becoming crucial in industry, where graph analytics, sparse neural networks, and learning on sparse data (like graphs or point clouds) are increasingly important.

Current systems are ill-suited to sparse computations. This mismatch wastes many billions of dollars yearly. For example, whereas supercomputers achieve 50–80% of their theoretical peak performance on dense linear algebra [14], they achieve only 1–3% of their peak on HPCG [7], a sparse linear algebra solver, and are well below 1% on graph algorithms [6]. As systems become more specialized, the lack of support for sparsity becomes more limiting: it stymies algorithm progress by forcing the use of inefficient dense computations, and eventually renders specialized architectures obsolete.

Sparse computations pose *challenges that demand a full-stack approach.* These challenges stem from two key properties. First, sparse data structures are *complex*: whereas dense data structures use negligible metadata (e.g., the length of each dimension in a dense matrix) and support arbitrary iteration orders, sparse structures use substantial metadata (e.g., coordinates associated with each value in a sparse matrix), and only some iteration orders are efficient (e.g., by rows, but not by columns, for a matrix in CSR format). Second, stemming from the characteristics of sparse data, sparse computation is *irregular*: tasks, data dependences, and operand sizes are not known in advance, but depend on runtime values (e.g., when multiplying two sparse matrices, the amount of work and output matrix size heavily depend on the coordinates of their nonzeros). Stemming from these properties, we identify the following key challenges:

Algorithmic choice: Sparse computations have a rich space of choices in algorithm, data representation, and schedule, which current languages and compilers cannot capture and optimize. Instead, programmers manually encode a concrete algorithm and data representation, and a suboptimal choice may lead to asymptotically worse performance.

Composition: Composing sparse computations is challenging because their combination may alter the right choices of algorithm and data representation, requiring global optimization. For example, computing the distances among vertices in a graph can involve a matrix multiplication followed by an element-wise multiplication with a sparse matrix (SDDMM). These operations must be fused to avoid asymptotically worse performance.

Adaptation and scheduling: In sparse computations, the right choice of algorithm and data representation are often unknown in advance and may change at run-time, thwarting the rigid division between current compilers and schedulers. Moreover, substantial runtime support is needed to find parallelism and locality, and to schedule and load-balance tasks and their data in a parallel system.

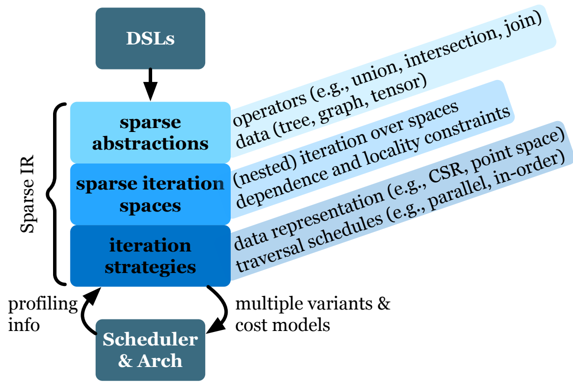
Hardware efficiency: Processor and network architectures include techniques tailored to dense computations, like vector units, prefetchers, and rigid memory systems and networks optimized for bulk transfers and infrequent synchronization. These techniques are inefficient on sparse computations, demanding new hardware techniques and software mechanisms to exploit them.

2 Opportunities

2.1 A Unified, Sparse IR

Modern applications often not only feature sparse computations but *combinations* of sparse computations that make efficient execution challenging. For example, many types of simulation compose different computational models and data structures to model different aspects of the overall system (e.g., weather simulations which combine separate models of the sea, atmosphere, and ocean surface). Different components may use different spare computation styles. For example, a a finite element model time integrator might use sparse tensors, incorporating contact force at the points determined by a collision detection algorithm based around spatial tree codes.

The variety of data structures and access patterns in sparse programs results in a corresponding variety of iteration constructs over *sparse iteration spaces.* A program over sparse data might feature loops over indirection arrays that result in irregular memory accesses (as in sparse tensor codes), a recursive traversal (as in tree-traversing simulation codes), or any combination of these constructs (e.g., dense loops with recursive traversals in n-body codes, or sparse loops with dense loops in semi-sparse tensor codes). These



DSLs
sparse abstractions
sparse iteration spaces
iteration strategies
Sparse IR
operators (e.g., union, intersection, join)
data (tree, graph, tensor)
(nested) iteration over spaces
dependence and locality constraints
data representation (e.g., CSR, point space)
traversal schedules (e.g., parallel, in-order)
profiling info
multiple variants & cost models
Scheduler & Arch

Figure 1. Overview of proposed intermediate representation.

iteration constructs arise from higher-level kernels that express algorithmic concepts, like an element-wise join of two trees [5] or an intersection of two sparse tensor spaces in a tensor contraction. The kernels operate over any number of *data* representations—pointer-based trees, sparse tensors stored as coordinates, graphs stored in CSR data structures. Realizing efficient iteration strategies to implement these complex programs using complex data structures on complex hardware requires two components: *(i)* defining the right *interfaces* for programmers to communicate computation and data properties to the compiler, and for the compiler to convey parallelism and locality information to the scheduler and architecture; and *(ii)* transformations of the sparse traversals to best match an application's computation and data to the available hardware.

We believe there is an opportunity for a unified *intermediate representation* (IR) for representing, optimizing, and generating code for sparse applications. Figure 1 shows the three levels of abstraction we envision: abstractions of sparse operations; representation of traversal over sparse iteration spaces; and concrete realization of specific traversal strategies that match data layouts and target architectures. The IR would be a target for domain-specific languages (DSLs) like TACO [9] or GraphIt [15]. Transformations of the IR would generate different *variant* implementations that the scheduler chooses from to target specific architectures. The IR thus serves two purposes. First, it is the glue that connects the high-level representation of computations and data expressed in DSLs to the low-level hardware operations. Second, it serves as a unified representation for expressing transformations on sparse computations to enhance locality and parallelism and generate efficient code.

2.2 A Data-Centric Sparse Accelerator

Existing accelerators mainly target dense computations. We believe there is an opportunity for a new kind of accelerator that is flexible enough to accelerate a broad range of sparse computations. The key challenge is that these applications feature *irregular* control flow and memory accesses, as they manipulate complex data structures (e.g., sparse tensors or graphs) and need complex coordination. Supporting this is infeasible with existing specialization approaches, which are limited to dense computations like deep learning [2, 4].

Our key insight is adopting *data-centric specialization.* Most specialized architectures *focus on the processing cores* rather than on the memory system. This is ill-suited to the complex access patterns and coordination required by sparse computations. Moreover, because data movement costs dwarf those of computation, scalable accelerators must be designed to *minimize data movement first.*

To tackle this challenge, we observe that the complex and irregular access patterns in these applications arise from the complexity and size of the data representation, e.g., of large sparse multi-dimensional tensors, and not intrinsically from the data values in the tensor. Thus, providing hardware support to manipulate these data structures both *enables effective acceleration* and results in a *simple hardware/software interface* that is well-matched to the capabilities of our proposed sparse IR.

To this end, we propose an architecture with three distinguishing features. First, hardware implements *data orchestration units* specialized for *sparse data structures.* Each of these units would perform a primitive function (e.g., pattern generation, dereference, intersection, union/merge, and accumulation), so the system will support the rich variety of storage formats and traversal structures needed by different applications. Data orchestration units will fetch complex data structures, marshal them to a spatial fabric of efficient processing elements, and combine their results to produce sparse outputs efficiently. Second, on-chip memory will be organized as a distributed collection of banks that can be reconfigured as a cache or a local memory. This hierarchy will support the mix of implicit and explicit decoupled data movement required by these applications. Third, hardware will provide adaptive partitioning, load-balancing, and pipelining techniques to achieve high utilization of compute and data orchestration units while minimizing data movement.

3 Timeliness

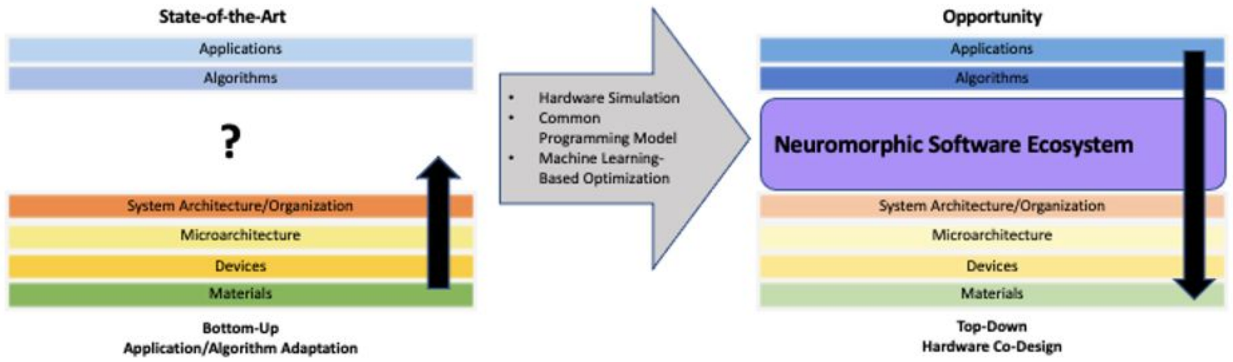
Now is the right time to develop a full-stack approach to handle sparsity. The waning of Moore's Law means that we cannot rely on silicon fabrication technology to improve over time and hide the massive inefficiencies of existing systems. And the push for specialization demands that we cater towards this broad and emerging domain; otherwise, we risk stifling algorithmic innovation and eventually making accelerators obsolete. Our proposed approach will especially benefit sparse computations that are key in emerging domains, such as data analytics, machine learning, simulation, and in-memory databases. This work leverages our recent work on languages, compilers, schedulers, and architectures for sparse computations. By innovating across the full software and hardware stack, these techniques can achieve performance, scalability, and efficiency gains that single-layer approaches cannot provide.

References

[1] M. Abadi et al. Tensorflow: A system for large-scale machine learning. In *Proc. OSDI*, 2016.
[2] T. Chen et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proc. ASPLOS*, 2014.
[3] T. Chen et al. TVM: An automated end-to-end optimizing compiler for deep learning. In *Proc. SOSP*, 2018.
[4] Y.-H. Chen, J. Emer, and V. Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proc. ISCA*, 2016.
[5] R. R. Curtin et al. Tree-independent dual-tree algorithms. In *Proc. ICML*, 2013.
[6] Graph500 supercomputer list. https://graph500.org/, 2020.
[7] HPCG performance for Top500 supercomputers. https://www.top500.org/lists/hpcg/hpcg-november-2020/, 2020.
[8] N. P. Jouppi et al. In-datacenter performance analysis of a Tensor Processing Unit. In *Proc. ISCA*, 2017.
[9] F. Kjolstad et al. The tensor algebra compiler. In *Proc. OOPSLA*, 2017.
[10] NVIDIA. Tensor cores, 2021. URL https://developer.nvidia.com/tensor-cores.
[11] A. Paszke et al. PyTorch: An imperative style, high-performance deep learning library. In *Proc. NeurIPS*, 2019.
[12] M. Püschel et al. SPIRAL: Code generation for DSP transforms. *Proc. IEEE*, 93(2), 2005.
[13] J. Ragan-Kelley et al. Decoupling algorithms from schedules for easy optimization of image processing pipelines. *ACM Trans. Graph.*, 31(4), 2012.
[14] Top500 supercomputer list. https://www.top500.org/lists/top500/2020/11/, 2020.
[15] Y. Zhang et al. GraphIt: A high-performance graph DSL. In *Proc. OOPSLA*, 2018.

Title: Top-Down Neuromorphic Hardware Co-Design via Machine Learning and Simulation
Authors: Catherine Schuman, Oak Ridge National Laboratory (ORNL); J. Parker Mitchell, ORNL; Maryam Parsa, ORNL; Shruti Kulkarni, ORNL; Prasanna Date, ORNL; James S. Plank, University of Tennessee
Topic: architectures, modeling and simulation, programming systems, emerging technologies

Challenge: Neuromorphic systems have tremendous potential in a variety of diverse settings, including for edge computing applications and as co-processors for future HPC systems. **A key challenge with neuromorphic computing systems today is the disconnect between ongoing research efforts in architecture, devices, and materials development and the needs of the algorithms and applications at the top of the compute stack, leading to a "bottom-up" approach to neuromorphic hardware and software**. This lack of a common software ecosystem to support hardware development has led neuromorphic architectures and devices researchers to follow a "hardware first, applications/algorithms later" approach. This approach can result in hardware that is suboptimal for some or all of the applications of interest to the Department of Energy (DOE). Different neuromorphic hardware implementations can have radically different performance implications; for example, a neuromorphic hardware system implemented with superconducting optoelectronics may be very fast but sacrifice energy efficiency, whereas a mixed analog-digital approach may be very energy efficient but may operate much slower. Because of the diversity of neuromorphic hardware implementations and because of the vastly different needs of application use cases ranging from edge computing to HPC, it is vital that application-hardware co-design play a key role in the development of neuromorphic hardware, specifically so that application and algorithm designers can help to shape the hardware itself.


State-of-the-Art
Applications
Algorithms
?
System Architecture/Organization
Microarchitecture
Devices
Materials
Bottom-Up Application/Algorithm Adaptation
Hardware Simulation
Common Programming Model
Machine Learning-Based Optimization
Opportunity
Applications
Algorithms
Neuromorphic Software Ecosystem
System Architecture/Organization
Microarchitecture
Devices
Materials
Top-Down Hardware Co-Design

Opportunity: There is the opportunity to establish a software ecosystem that connects the algorithms and applications of interest to DOE to the design and development of architectures, devices, and materials for neuromorphic computing, to enable application/algorithm-driven co-design of neuromorphic systems. Two key components of this software ecosystem, **large-scale hardware simulation on existing HPC** to inform hardware design and **a common programming model for neuromorphic applications** to enable rapid application and algorithm development, can be combined with a third component, **machine learning-based hyperparameter optimization**, to tailor hardware designs in simulation to meet application and algorithm needs. This approach can be used to define hardware requirements for future neuromorphic architectures, devices, and materials.

Large-scale hardware simulation is a critical component of the complete co-design workflow. There is a vast diversity of ways to implement neuromorphic systems, ranging from fully digital CMOS and mixed analog-digital CMOS to beyond CMOS implementations with memristors, magnetic tunneling junctions, spintronics, superconducting optoelectronics, and even biomimetic materials. Each of these different implementations has vastly different performance characteristics in terms of processing speed, power efficiency, and space efficiency, and thus are not necessarily well-suited for every application of interest to neuromorphic. The ability to perform hardware simulation at scale to understand the performance characteristics on real-world applications is critical for top-down hardware co-design. Such a hardware simulator needs to be parameterizable to allow for the evaluation of radically different device types. Equally important to hardware simulation is a **common programming model** for neuromorphic computers to enable rapid application development and evaluation with hardware simulation. Finally, because the design exploration space for potential hardware systems is vast, **machine learning** approaches will be required to automatically optimize hardware characteristics to achieve desired performance characteristics (i.e., speed, precision, energy efficiency, etc.) for a particular application or suite of applications.

Timeliness or maturity: Neuromorphic systems now exist in a variety stages of maturity and are under active development by large industry players (e.g., Intel with Loihi [1]), start-ups (e.g., Rain Neuromorphics [2]), as well as large-scale academic efforts such as EFRCs with a neuromorphic focus [3]. All of these efforts are under active development and can be influenced by compelling demonstrations of applications of interest with their associated deployment requirements. At the same time, there have been a variety of real-world applications demonstrated on neuromorphic hardware or in simulations of neuromorphic hardware, ranging from scientific data classification for large-scale scientific instruments [4] to edge computing applications [5] to non-machine learning tasks such as modeling epidemic spread [6]. As such, there are already a set of applications that can be used to seed the development of a neuromorphic software ecosystem to enable application-hardware co-design. The relative maturity of neuromorphic systems today, along with sets of applications that are ready to go provide the opportunity to establish a template for future tightly coupled application-hardware co-design.

References

[1] Davies, Mike, et al. "Loihi: A neuromorphic manycore processor with on-chip learning." *Ieee Micro* 38.1 (2018): 82-99.
[2] Pantone, Ross, Jack Kendall, and Juan Nino. "Building smarter, scalable hardware for artificial intelligence." URL: https://rain.ai/blog/posts/building-smarter-scalable-hardware-for-artificial-intelligence
[3] Ivan Schuller, et al. "Quantum Materials for Energy Efficient Neuromorphic Computing." URL: https://qmeenc.ucsd.edu/index.php
[4] Schuman, Catherine D., et al. "Neuromorphic computing for temporal scientific data classification." Proceedings of the Neuromorphic Computing Symposium. 2017.
[5] Schuman, Catherine D., et al. "Low Size, Weight, and Power Neuromorphic Computing to Improve Combustion Engine Efficiency." 2020 11th International Green and Sustainable Computing Workshops (IGSC). IEEE, 2020.
[6] Hamilton, Kathleen, et al. "Modeling epidemic spread with spike-based models." International Conference on Neuromorphic Systems 2020. 2020.

Co-designing from Atoms to Architectures

A. Aiken (Stanford, aaiken@stanford.edu), J. Baniecki (SLAC, banieck@slac.stanford.edu), J. Bokor (Berkeley, jbokor@berkeley.edu), V. Churavy (MIT, v.churavy@gmail.com), R. Coffee (SLAC, coffee@slac.stanford.edu), A. Dragone (SLAC, dragone@slac.stanford.edu), A. Edelman (MIT, edelman@math.mit.edu), D. Ernst (Cray, dje@cray.com), S. Hearne (ORNL, hearnesj@ornl.gov), J. Hennessy (Stanford, hennessy@stanford.edu), H. Mabuchi (Stanford, hmabuchi@stanford.edu), P. McIntyre (Stanford & SLAC, pcm1@slac.stanford.edu), N.A. Melosh (Stanford, nmelosh@stanford.edu), Sh. Misra (Sandia, smisra@sandia.gov), S. Mitra (Stanford, subh@stanford.edu), J. Neaton (Berkeley & LBNL, jbneaton@lbl.gov), T. Ogitsu (LLNL, ogitsu1@llnl.gov), O. Ovchinnikova (ORNL, ovchinnikovo@ornl.gov), J. Rabaey (Berkeley, jan_rabaey@berkeley.edu), A. Salleo (Stanford, asalleo@stanford.edu), E. Schwegler, (LLNL, schwegler1@llnl.gov), M. Shulaker (MIT, shulaker@mit.edu), R. Stevens (U of Chicago & Argonne, stevens@anl.gov), T.P. Straatsma, (ORNL, straatsmatp@ornl.gov), J.B. Thayer (SLAC, jana@slac.stanford.edu), N. Wichmann (Cray), H. P. Wong (Stanford, hspwong@ee.stanford.edu), R. N. Zare (Stanford, rnz@stanford.edu), and S. Shankar[1](Harvard, sshankar@seas.harvard.edu)

1. Topic: The following document is a Position Paper on Co-design as a methodology for integration of application-specific computing to solve real challenges. This methodology is based on concurrent design and optimization of the computing architecture, software, algorithms, materials, devices, information processing, communication, etc. Previous analysis of specific examples of Co-design (e.g., Anton for protein folding[2]) indicates great promise for acceleration of computing for applications on top of Moore's law. However, these examples are still a selective few with dedicated resources and budgets, and no current framework exists to more broadly integrate the various components of computing for most applications. We believe that a radical rethinking of innovation across components of computing would be useful to address both the multi-disciplinary complexity of constructing new application-specific computing systems and the continuing offshoring of the US innovation ecosystem[3].

2. Co-design Scientific and Technological Challenges: Development of new design abstractions as an integral part of the codesign process remains a fundamental methodological challenge. On a more focused level, low-level aspects of codesign could be greatly accelerated by the development of cyber-physical simulation toolkits and workflows that facilitate simple construction of models in which dynamical systems with real-time control algorithms or other rule-based adaptive supervision. High-level aspects of codesign will likely demand new theoretical tools related to algorithm design for complex heterogeneous hardware systems that may exhibit behavioral uncertainties that cannot effectively be captured by low-dimensional parameterization. We see three types of challenges in a general-purpose computing solution for all applications.

Efficiency: Aspects of efficiency include energy per bit, computational complexity of an application, and manufacturability. Of these, energy or power minimization is a universal macro-constraint for on-chip architectures. The computer industry is actively dealing with trade-offs between performance and energy efficiency. Detection and real-time mitigation of manufacturing abnormalities in the lower levels (materials/devices/circuits) of the system stack is crucial for maintaining system performance, yield, and reducing waste and energy consumption in the manufacturing process. In addition, trade-offs between large volume manufacturing of a few systems versus custom-manufacturing for multiple systems need to be systematically evaluated. Although there are many efforts to analyze energy efficiency and complexity, no dedicated efforts exist that integrate efficiency for researchers and scientists to design and develop tools and methodologies for developing the building blocks for an optimal design.

Prototyping and Manufacturing: There are many challenges including the time lag from design to prototyping, ability for integration of novel materials across multiple technologies, co-optimization between packaging and silicon, ability to ramp up production with well-understood cadences, manufacturing within the US for security and resilience. Current approaches to manufacturing are based on existing designs which are optimized for a given process technology. The process from design to product is still largely sequential with many iterations that are becoming increasingly necessary for efficient optimization across performance, cost, and adaptability. Understanding the opportunities and limitations presented by new ideas in materials, devices and circuits has been limited by the practical limitation of testing those ideas at the system level, at least, as a prototype.

Security and Resiliency: Even if the above building blocks are available for a wider audience, security of the components and the integrated systems are key to resilient computing systems. This aspect is critical for both the components of co-design and the tools used to design, prototype and manufacture. The need for trust in microelectronics design along with implementation of software packages is a critical need for all sectors of computing and system control. The problem is further compounded when sensors designed and manufactured by

[1] *Contact for correspondence*

[2] Shaw DE, Deneroff MM, Dror RO, Kuskin JS, Larson RH, Salmon JK, Young C, Batson B, Bowers KJ, Chao JC, et al. (2008); "Anton, a special-purpose machine for molecular dynamics simulation". Commun ACM 51:91–97

[3] Co-design Meetings, Harvard (2017); Stanford/SLAC (2019)

third party suppliers are integrated into large-scale computational systems. The security and resiliency at the system and component levels are necessary for mission critical applications.

3. Guiding Principles for Co-design: A new proposed framework should enable a modular, "Legos-type" approach where researchers could simulate and put together the building blocks to design and prototype the computing advances as they become available. The components themselves should be based on all aspects of efficiency. We propose a few guiding principles below related to global co-design for optimizing efficiency, use of smart hardware-software labs for ability to emulate and prototype, open workflows and tools to design for resilience and testing, and a hub for a multi-disciplinary engagement to engage in cross-cutting research.

Use of Global Co-design from Atoms to Architectures (Bottom-up Design): As proposed, Co-design must reach all the way down to the atomic level to integrate materials engineering and advanced characterization early in the process. The rise of multifunctional atomically engineered materials opens up unprecedented opportunities for bottom-up engineering of building blocks for computing, starting from single atoms or even single electrons. Due to its engineered functionality, the material or the molecule thus becomes the device, which is to be reproduced and integrated in a circuit and, later, a system. This design approach in turn will help integrate information processing between different classes or materials to systems such as analogue, classical, and quantum systems.

Design of Intelligent Hardware-Software Labs: Integration of Artificial Intelligence/Machine Learning (AI/ML) with Hardware/Lab Equipment can enable smart design, development, testing, and prototyping.[4] The current proliferation of opportunistic applications of AI/ML tools in scientific workflows are short-circuiting the detailed physics and chemistry-based analysis of critical complex phenomena. Developing powerful yet human-interpretable AI/ML is still a grand challenge for the AI/ML field in general, but this challenge of overlaying principled statistical theory with practical "big data" can be enabled by Co-design. The concepts and methods from cybernetics and robust control theory and from the emerging field of control-over-channels, could prove useful if properly translated to the codesign context.

Develop Open Workflows and Tools: The exponential increase in sensors both in high-energy physics detectors and in Internet of Things, especially in large-scale scientific facilities and in intelligent consumer systems, pre-supposes a commensurate revolution in data analysis workflow. Realizing this revolution requires multiple stages to transparently handle the acquisition, processing, transfer, analysis, and visualization. Optimization of the workflow to extract information that distills data into actionable information on the timescales required of experiments requires co-design of detectors, edge computing layers, and data analytics which may run on a variety of architectures, from local compute clusters to Department of Energy's leadership computing facilities.

Create Application-Enabled Innovation Hubs: Development of an "innovation hub" will enable the community to use the building blocks for developing physical computing prototypes for visualizing and testing new designs. We propose a two-dimensional approach: 1) Scientific and Engineering Research linking Applications, Architectures, System and Devices, Novel materials along with their synthesis and processing, Information Abstractions; 2) Prototyping for exploring various computing options by developing following components as needed: Design methodologies, Validation strategies, Tool sets for design, Fabrication, Integration, and Packaging. The sensing may be precise in particle accelerators or noisy in wireless networks. Working across the stack in applications, systems software and hardware creates opportunity for innovative Co-design and enabling a disciplined process to bridge/apply shared knowledge across the spectrum.

4. Timeliness of Co-design: With the increasing difficulty of sustaining the cadence of Moore's Law, there is a time critical need to rethink how can we build the next generation of computing while lowering the increasing costs of design and manufacturing. Our vision will enable a new era in personalized and application-centric computing ("Cambrian" era[5]) that bridges information theory, computing and communication abstractions with materials, devices, hardware, systems, architecture, algorithms and software for enabling new applications. As early examples have shown the promise of this approach, our integration would enable a new systematic approach for designing and building efficient computing systems from atoms to materials to devices to systems, which can be rapidly prototyped within an innovation hub.

[4] Dally, W. et al. Hardware-enabled artificial intelligence. In Proceedings of the Symposia on VLSI Technology and Circuits (Honolulu, HI, June 18–22). IEEE Press, 2018, 3–6.
[5]Hennessy, J. and Patterson, D. A New Golden Age for Computer Architecture. Communications of the ACM, Vol. 62, No. 2, February 2019.

Supplementary Materials: Co-design from Atoms to Architectures

A. Aiken (Stanford), J. Baniecki (SLAC), J. Bokor (Berkeley), V. Churavy (MIT), R. Coffee (SLAC), A. Dragone (SLAC), A. Edelman (MIT), D. Ernst (Cray), S. Hearne (ORNL), J. Hennessy (Stanford), H. Mabuchi (Stanford), P. McIntyre (Stanford & SLAC), N.A. Melosh (Stanford), Sh. Misra (Sandia), S. Mitra (Stanford), J. Neaton (Berkeley & LBNL), T. Ogitsu (LLNL), O. Ovchinnikova (ORNL), J. Rabaey (Berkeley), A. Salleo (Stanford), E. Schwegler, (LLNL), M. Shulaker (MIT), R. Stevens (U of Chicago & Argonne), T.P. Straatsma, (ORNL), J.B. Thayer (SLAC), N. Wichmann (Cray), H. P. Wong (Stanford), R. N. Zare (Stanford), and S. Shankar (Harvard)

APPENDIX 1: *Comments on Areas of Emphasis*

1. **Key aspects of codesign across the entire hardware/software stack to include applications, algorithms, system software, and system architecture;**
   - Novel approach to co-design is needed, that bridges from applications all the way to materials. Traditional co-design has focused on tradeoffs between algorithms, (software) and hardware implementations of functions. Permitting trades in power, performance, flexibility, cost, etc. In the future we need to consider deep trades due to additional alternatives such as surrogate models of applications, broader options in architecture and potentially novel materials that permit improvements in power.

2. **Insights into codesign for workflows arising from scientific experiments, on supercomputers, or to support large-scale scientific instrument;**
   - Pushing past traditional single applications to entire end to end workflows implies that you might have different systems components carrying out different stages of computation and I/O. This opens up the idea of hierarchical or multi-component optimization as part of the co-design with different trade offs being exploited for different elements of the workflows.

3. **Methods and tools for quantitative codesign, including both those that inform high-level decision-making and those impacting low-level aspects of the codesign process;**
   - Design synthesis (either traditional or via new AI driven methods) will be an important future driver of co-design. The idea of extending design synthesis to the full stack should be considered as it may result in deeper optimizations. Iterative design is another approach that revisits design choices multiple times as different trajectories are explored.
   - Principled approaches to the development of new design abstractions as an integral part of the codesign process remains a fundamental methodological challenge. On a more focused level, low-level aspects of codesign could be greatly accelerated by the development of cyber-physical simulation toolkits that facilitate simple construction of models in which dynamical systems (specified in terms of ordinary or partial differential equations) interact in real time with control algorithms or other rule-based adaptive supervision. High-level aspects of codesign will likely demand new theoretical tools related to algorithm design for complex heterogeneous hardware systems that may exhibit behavioral uncertainties that cannot effectively be captured by low-dimensional parameterization. It seems likely that concepts and methods from robust control theory (and from the emerging field of control-over-channels, e.g., Bode-Shannon theory) could prove useful in this context if properly translated to the codesign context.

4. **New codesign challenges anticipated over the next decade**
   - The dream of single computational systems where analog and digital computation is mixed in to a hybrid computation system has been simmering in the background of computing for decades [A. Hausner, Analog and analog/hybrid computer programming, Prentice-Hall, 1971]. The recent advances in quantum computing and integrated non-linear photonic systems [N Singh, et. al. Photonics (2020) https://doi.org/10.1364/PRJ.400057] have greatly expanded the potential capabilities and applications of these systems. However, the complexity of building, optimizing and implementing such computational systems has stymied their applications to anything more than a few niche applications. This is an opportunity for co-design of materials, devices, and system architecture provide a unique opportunity to greatly expand the impact of hybrid computing. Challenges include understanding dissimilar interfaces, and joining of analog and digital components together, as well electrically / optically coupling the devices.
   - From an applications perspective, the next decade will see dramatic new challenges in public health that can only be met by revolutionary codesign methodologies. Tracking the emergence and mutation

of novel viruses will require widespread environmental sampling and genomic/proteomic characterization on unprecedented scale; sheer numbers will require innovative codesign of hardware systems and search algorithms in order to enable surveillance at scale within achievable supply rates of biochemical reagents, dedicated laboratory time, and compute cycles. The next decade will likewise see a dramatic increase in the number of autonomous/auto-piloted vehicles on roadways and in commercial airspace; robust, secure, verifiable and adaptable/updatable collision avoidance systems will likely require codesign of sensors, transponders and algorithms/protocols at their core. From a methodological perspective, the generalization of nascent codesign principles beyond the relatively familiar substrate of semiconductor electronics to biological and quantum systems will demand accelerated development of core theory regarding codesign as a new science in its own right.

- System on a Chip and flexible hardware design to accommodate variety of the needs in industry and scientific community. It is apparent that there are only a handful of companies who are capable of mass producing SOC with sufficient size of on chip high-bandwidth memory in economical way. Most likely, HPC project will also rely on their capability. Designing capability of a prototype is important but supplying sufficient number of HPC chips is critical.

APPENDIX 2: *Answers to a few Notational Questions*

1. **How to balance breadth of applications versus customization benefit? Does this vary by system type?**
    - This is a hard question to answer in general for the breadth of applications. For it to be viable, the breadth of applications includes enough users to put the project above the threshold to recover the fixed costs of designing and building hardware. And to the extent that we can make codesign more efficient we help reduce those fixed costs and the risk associated with any new codesign project. It most certainly will vary by system type and use case, but by focusing on those use cases with higher potential gains (that could be cumulative) will keep things moving. With some scenarios that are exploiting surrogatges we can see factors of thousands in improvement coming from alternative algorithms with changes to hardware needed to support those new methods.
    - System on a Chip and flexible hardware design to accommodate variety of the needs in industry and scientific community. Designing capability of a prototype is important and needs to be connected to scaled-up manufacturing.
    - The resent revolution in additive manufacturing has been fired by the ability to use a library of materials to produce nearly an infinite number of new structural that solve specific problem. Co-design enables this same revolution to occur in microelectronics manufacturing.

2. **What are the tools and techniques that enable successful codesign interactions and where are the gaps?**
    - We need next generation of vertically integrated design and simulation tools and high speed emulators to run them on.

3. **With artificial intelligence and machine learning becoming more widely used in scientific workflows and applications, what new challenges and opportunities does this present?**
    - The current proliferation of opportunistic applications of "black box" artificial intelligence and machine learning (AI/ML) tools in scientific workflows has the unfortunate consequence of short-circuiting the traditional emphasis on painstaking careful development of new concepts and reduced models for critical complex phenomena. As a result, hand-waving interpretations of otherwise inscrutable designs discovered by application of AI/ML to limited datasets launch dubious ideas into complex systems engineering, which may in fact reflect only improper generalization of specific results to broader context, or even unwarranted rationalizations of overfitting by AI/ML routines. Of course, developing powerful yet human-interpretable AI/ML is a grand challenge for the AI/ML field in general, but this challenge of overlaying principled statistical theory with practical "big data" seems to echo the broader paradigm of codesign. There could be opportunities to help guide the development of AI/ML practice within the mainstream scientific community by establishing "governed" cloud computing services that incorporate not only data storage and computation but rigorous meta-analysis of results and curated guidance regarding best practices for interpretation of specific types of results.
    - The primary one is that the low level architectural features needed to support AI centric workloads are different than traditional architectures and in many ways are simpler and need to be part of the design space considerations.

4. **New accelerator technologies and chiplets are increasing the possible design space. What is the potential of these new technologies and how can codesign be used to take maximum advantage of them?**
    - Co-design is to provide accurate information regarding application side requirement as well as constraint coming from options in fabrication capabilities thereby most effective platform for developing future HPC chipset that can be the most (cost) effective for both scientific and industry uses. If one admits SOC is the way to go, understanding possible design space will

likely to have constraints coming from the fabrication capability is must. Understanding could be achieved if good communication is established.

- Essentially these open the space of solutions a bit by making it easier to integrate technologies from different nodes etc. A major breakthrough would be possible if we could use these new packaging technologies to integrate components fabricated with different materials. Also one could consider that we now have the ability to map IP to chiplets and do fine grain integration.

5. **How can we further the state of the art in efficient and flexible open-source hardware, modeling, and simulation tools that can underpin hardware codesign activities?**
    - Co-design of materials, devices, and architecture dramatically increases the parameter space for engineering the next generation of computational systems. It will only be possible to take full advantage of these advances if there are models that can rapidly explore the parameter space in real time during fabrication.
    - Encourage open source design stack with well defined APIs that encourage groups to develop components. Require that for government procurements that in addition to the hardware that the vendors ship the full simulation stack so it can be used as baselines for further co-design work.

6. **Is there a performance benefit to codesigning scientific applications and the computer systems (hardware and software) they run on, or will the additional time and cost outweigh the benefits observed relative to more-or-less portable applications running on stock supercomputers?**
    - A significant benefit to the additional complexity of co-designed hybrid systems is the ability to improve the digital / materials fingerprint for increased security. The need for trust in microelectronics design along with implementation of software packages is a critical need for all sectors of computing and system control. Co-design greatly increases the ways in which unique digital fingerprints can be related to enable trusted systems.
    - With the characteristics of next generation scientific facilities in National Laboratories and elsewhere, experiments requiring massive-scale data analytics are on the horizon. Co-design of smart sensors, customized computing systems and real-time algorithms, not only will increase performance but would constitute an enabling factor. Co-design will enable a new level of customization and optimization focused on information extraction and data reduction throughout the entire data acquisition chains.

7. **How do scientific applications and supercomputer codesign differ fundamentally from the codesign employed regularly for embedded systems (such as in automobiles and home appliances)? Can either area learn from the other?**

- Fundamentally the ability to simulate end-to-end is much greater in embedded systems and this enables co-design. Addressing the simulation stack is one way to improve this.

Observing and Optimizing Performance of Coupled MPI Applications

Sameer Shende — Univ. of Oregon — sameer@cs.uoregon.edu
D.K. Panda, Hari Subramoni — The Ohio State Univ. — panda@cse.ohio-state.edu
Jon Rood, Michael Sprague, Shreyas Ananthan — The National Renewable Energy Laboratory (NREL) — {Jon.Rood,Michael.A.Sprague,Shreyas.Ananthan}@nrel.gov

CoDesign Position Paper Topic: Programming Systems, Codesign methodologies
Keywords: MVAPICH2, TAU, LLVM, Kokkos, AMReX, Nalu-Wind, AMR-Wind, ExaWind, GPU

Introduction

Scalable, parallel computing architectures are the foundation of modern high-end computing (HEC) systems, with the potential to deliver vast amounts of processing power required to solve grand challenges in scientific and engineering high performance computing (HPC) applications as well as artificial intelligence/machine learning (AI/ML) applications. It is their ability to scale along two fundamental dimensions – node-level parallelism with an increasing number of cores and graphics processing units (GPUs) and system-level parallelism through clustering and high-performance networks – that has propelled advances in computational power through massive parallelism. As compilers and runtime systems for intra-node parallelism evolve, it is becoming harder to comprehend and interpret performance down to the source level. This often leads to suboptimal performance and under utilization of HEC and lost programmer productivity.

Challenges

Computational scientists want to create next-generation scalable applications that are performant and portable, without having to re-engineer and optimize the code for every new platform. Optimizing the performance of coupled applications is even more challenging at extreme-scales. This is due to the following reasons:

1. Coupled multi-physics solvers operating at different rates need to synchronize, coordinate, and exchange data. The ExaWind [5] project features Nalu-Wind and AMR-Wind applications with different MPI communicators within an MPMD application. Achieving optimal MPI performance for coupled codes at extreme-scale is challenging.
2. GPUs are now widely recognized as the key component in extreme-scale HPC systems, but obtaining and sustaining good performance on GPUs remains a challenge.
3. Advanced optimizing compilers based on LLVM are now able to leverage capabilities of GPUs including support for OpenACC and OpenMP target offload directives. These compilers feature support for expressing loop level transformations/optimizations using pragmas, and efficient performance instrumentation using a TAU LLVM plugin, but these improvements are scattered in different repositories and cannot be used to compile an application yet. With the recent work for the creation of a DOE LLVM fork from ECP[6], there is an opportunity to leverage these features in a single C++ compiler framework.
4. Node level programming models have evolved and it is now possible to express parallelism for multiple GPU architectures from vendors such as NVIDIA, AMD, and Intel using C++ lambda functions. Kokkos, AMReX, and RAJA provide GPU optimized backends, but higher-level libraries such as Trilinos still can't substitute one Kokkos implementation for another leading to limitations in portability to new GPUs due to embedded software components. Nalu-Wind uses Trilinos but doesn't support new GPU architectures from Intel or AMD GPUs yet, although Kokkos has been ported to each.

5. Solver libraries such as Trilinos [2] are not the only users of the GPU. Optimized MVAPICH2-GDR libraries use GPUs for efficient inter-GPU communication using GPU Direct Async protocols and use the GPU cores in large scale computations for asynchronous collective operations. As the number of nodes increases, the need for efficient global collective operations increases and poses a challenge.
6. The role of network cards has evolved. Collective operations can now be offloaded to processors in specialized network cards including Mellanox Infiniband cards that support BlueField-2 adapters with ARM64 cores. SHaRP protocols used by MVAPICH2 also offload collective operations to the network card to improve the efficiency of collectives.
7. It is getting harder to observe and optimize the performance of HPC applications running on multiple GPU architectures, using advanced MPI optimizations involving GPUs, using multiple communicators in coupled simulation codes.

Opportunity

We envision a codesign project that will optimize the coupled AMR-Wind and Nalu-Wind Computational Fluid Dynamics (CFD) codes, written in C++, that aim to advance fundamental understanding of flow physics governing wind plant performance, including wake formation, complex-terrain impacts, and turbine-turbine effects. The technologies that we will target include MVAPICH2 [4] for inter-node parallelism and runtime optimization of coupled MPI applications, TAU [1] for performance evaluation and optimization, LLVM-DOE compilers [6] for code optimization, Kokkos [3] and AMReX libraries for node-level parallelism, and GPU runtimes including HIP, CUDA, and OneAPI DPC++/SYCL.

Timeliness and Maturity

Recent advances in MVAPICH2 to support the MPI Tools Interface (MPI_T) in tools such as TAU, GPU based communication and collective operation optimization, and support for multi-communicator MPMD codes fits in well for runtime optimization of coupled executions of Nalu-Wind and AMR-Wind. Advances in the Kokkos profiling API including support for tracking deep copies and GPU-CPU interactions can help improve GPU profiling features of TAU. It allows for multi-level instrumentation from the compiler-level to the runtime using fine-grained instrumentation and mappings to higher-level source code in the form of Lambda functions. Runtime optimization of coupled CFD solvers using MVAPICH2, LLVM, Kokkos, AMReX, TAU seems within reach as we evaluate and attempt to co-design using emerging new technologies.

References

[1] S. S. Shende and A. D. Malony, “The TAU Parallel Performance System,” International Journal of High Performance Computing Applications, vol. 20, no. 2, pp. 287–311, 2006.
[2] Trilinos Project. https://trilinos.github.io/, 2021.
[3] Kokkos: Core Libraries. https://github.com/kokkos/kokkos, 2021.
[4] Network-Based Computing Laboratory, “MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE.” http://mvapich.cse.ohio-state.edu/, 2021.
[5] Sprague, M. A., Ananthan, S., Vijayakumar, G., Robinson, M., “ExaWind: A multifidelity modeling and simulation environment for wind energy”, NAWEA/WindTech 2019 Conference.
[6] The LLVM Compiler Infrastructure, https://github.com/llvm-doe-org/llvm-project.git, 2021.

Co-designing Power Management with Job Scheduling for Efficient Exascale Computing

Matthew D. Sinclair and Shivaram Venkataraman
University of Wisconsin-Madison
{sinclair,shivaram}@cs.wisc.edu

Topics: Architectures, Applications, Co-Design Methodologies

1 Motivation

In recent years, supercomputers have increasingly targeted petascale and exascale levels of computing. To reach this lofty goal, many of these systems have turned to using large numbers of compute accelerators, which offer greater power efficiency and thus enable exascale computing for emerging AI and HPC workloads within a constrained power budget. For example, nearly all of the top 10 supercomputers in the Top-500 today utilize GPUs and the top ranked supercomputer, Oak Ridge National Labs' Summit, has approximately 24000 NVIDIA Volta V100 GPUs. Moreover, the recently announced Aurora, El Capitan, and Frontier supercomputers (as part of the DOE's CORAL-2 program) are expected to have even more GPUs. Further, with the development of new accelerators and customized chips (e.g., TPUs, GraphCore), future supercomputers will likely be comprised of a large variety of compute devices. However, using accelerators increases heterogeneity at multiple levels, including the architecture, resource allocation, competing user needs, and manufacturing variability. Accordingly, exascale-class and beyond systems need to efficiently handle many simultaneous jobs while balancing PM and multiple levels of heterogeneity.

Recent studies on supercomputers have shown that PM can impact application performance by up to 20% on CPUs in supercomputing systems, even when the CPUs have the same architecture and vendor SKU [1, 2, 4, 6]. This variation occurs due to the manufacturing process and the chip's power constraints [2]. However, despite their increasingly widespread use in modern HPC systems and supercomputers, little work exists that examines how PM and manufacturing variability in other accelerators (e.g., GPUs) affect application performance. Further, since there is no standardized way for accelerators to expose PM information, system management software (e.g., operating systems or job schedulers) struggle to control performance and predictability. Given their importance in driving modern HPC systems and supercomputers, it is imperative to understand how PM affects application performance. We next present some initial results on how PM can impact performance, which highlights the need for **co-designing power-management policies alongside job scheduling**.

2 Challenges

To observe the impact of PM on GPU performance, we chose GPU applications that stress the steady state power consumption of the GPU. Specifically, we use a SGEMM kernel from NVIDIA's cuBLAS library, which stresses the compute, and STREAM, which stresses the memory subsystem. These applications are representative of key kernels from next-generation workloads including machine learning. We use them to study the behavior of GPUs across a number of clusters ranging from Cloudlab[1], a small GPU cluster with 12 NVIDIA Volta V100 GPUs, to Oak Ridge (Summit supercomputer), Sandia (Vortex), and TACC (Frontera and Longhorn) which contain between 216 and 24000 V100 GPUs, and which use a variety of cooling methods (air, mineral oil, and water).

To study the effects on performance, we size the benchmarks to fully occupy all of the GPU's streaming multiprocessors (SMs). Additionally, as the GPU's PM controller relies on dynamic voltage and frequency scaling (DVFS) to maintain the per-GPU power limit for safe operation [5], it is important that the kernels run long enough for the DVFS controller to reach a stable state (i.e., a constant SM frequency) [3]. We define 1 run of our experiment as 100 repetitions of each benchmark's kernel. The repetitions help avoid statistical bias and any other additional transient effects. In all our experiments we use V100 GPUs in the SXM2 configuration (max frequency: 1530 MHz, TDP: 300W). Moreover, to avoid transient effects, we collected data for multiple runs on the same machine over multiple weeks.

In total, we ran over 100000 experiments and our preliminary study has identified several key insights.[2] Regardless of cooling approach or cluster size, we observed 5%-10% performance variability caused by the lack of global PM (similar to prior work for CPUs [2]). Some outliers are severe: for example, in TACC, PM throttled 300W TDP GPUs to as low as 250W, causing a 20% slowdown. However, while performance variations are directly correlated with frequency changes, temperature and power consumption are in general not directly correlated and are poor indicators of GPU performance. Finally, we also observed that PM can be influenced by spatial (i.e., neighboring GPUs) and temporal (i.e., kernels running previously on same GPU) effects. Although mineral oil and water cooling reduce the relative temperature variation, we still observed performance and power vari-

[1]https://cloudlab.us
[2]Due to space constraints we do not show figures of these results.

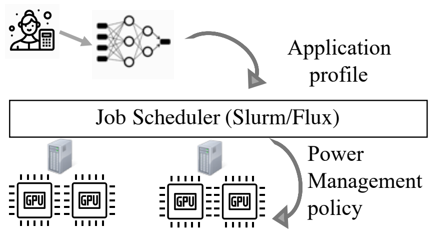


Figure 1: Proposed co-design of application profiling, job scheduling and accelerator power management policies.

ability.

As multi-GPU experiments become increasingly common, having such a variation across GPUs due to PM can significantly affect applications that coordinate across GPUs. Further we find that the performance variation also depends on the application (STREAM vs. SGEMM). We also discussed our preliminary findings with researchers at AMD and NVIDIA to validate that this behavior was beyond that expected from process variation.

Unfortunately, it is difficult to address these challenges directly, because modern GPUs make only local PM decisions where each node merely allows its GPUs (and other processors) to optimize for its power consumption. This means that we cannot perform application-aware global PM, where scheduling frameworks or administrative tools can effectively place workloads to avoid slowdowns. This motivates our proposed co-design that we describe next.

3 Opportunity

To overcome the observed performance variability in modern systems, we propose working with system vendors, administrators, and integrators, as well as end users to create a new ecosystem that transforms the efficiency of PM in existing systems and creates best-in-class methodologies that can be adopted to improve both current and future systems. Figure 1 shows our overall approach.

Thus far we focused on GEMM and STREAM, which are representative of workloads such as machine learning on modern GPUs. However, further experiments with additional applications that stress different system components are needed to **design application-specific PM policies**. We will further enhance these application-specific PM policies by **making job schedulers variability-aware**. Variability-aware schedulers will utilize software-runtime co-design to identify and harness the performance variation across GPUs in existing systems. As a result, schedulers can optimize for each application's power needs. Moreover, grouping GPUs with similar performance variability together and scheduling jobs across those GPUs, we can reduce time spent waiting for stragglers and ensure consistent behavior. Finally, in addition to identifying performance variation amongst accelerators, we will extend our experiments into a benchmark suite (and corresponding user-facing interface) that is run periodically to provide better **tools for system administrators** to identify slow accelerators and mark them for further investigation, reboot, or potential replacement.

However, scheduling jobs more efficiently at the software and runtime layers is limited in its ability to quickly, dynamically change policies as cluster conditions evolve. A major limiter to further improving efficiency is the lack of standards for exposing power information in modern accelerators. Thus, for future systems we will build on the insights generated by our optimizations for current systems, and apply co-design that makes the hardware, software, and runtime layers aware of the variance in the systems. To do this, we will design a standard for accelerators to expose PM information from the hardware to the software and runtime. Using this information, instead of performing PM locally, we plan to develop a **global power management** scheme to enable optimal PM decisions across accelerators and further reduce performance variability.

4 Timeliness

As modern HPC systems are designed to use increasing number of accelerators such as GPUs, it is imperative to understand how power management affects the behavior of applications. Our preliminary results show that significant performance variation already exists on modern systems. Thus, given the US government's planned investment in additional exascale-class and beyond machines with even more accelerators (which will further increase both heterogeneity and variability), our proposed work is both timely and important to improving the efficacy and utility of these machines. By making power management a first-class citizen in these systems, our co-design will ensure more consistent performance between runs, help administrators better spot and investigate bad GPUs, and reduce thermal throttling, thereby improving energy efficiency.

References

[1] D. Chasapis et al. Runtime-Guided Mitigation of Manufacturing Variability in Power-Constrained Multi-Socket NUMA Nodes. In *ICS*, 2016.

[2] D. Chasapis et al. Power Efficient Job Scheduling by Predicting the Impact of Processor Manufacturing Variability. In *ICS*, page 296–307, 2019.

[3] J. Coplin and M. Burtscher. Energy, Power, and Performance Characterization of GPGPU Benchmark Programs. In *IPDPSW*, pages 1190–1199, May 2016.

[4] A. Das et al. Mitigating the Effects of Process Variations: Architectural Approaches for Improving Batch Performance. In *ASGI*, 2007.

[5] R. Ge et al. Effects of Dynamic Voltage and Frequency Scaling on a K20 GPU. In *ICPP*, pages 826–833, 2013.

[6] Y. Inadomi et al. Analyzing and Mitigating the Impact of Manufacturing Variability in Power-Constrained Supercomputing. In *SC*, 2015.

Automatic derivation of performance models for programs with data dependent behavior

Matthew Sottile, Lawrence Livermore National Laboratory (sottile2@llnl.gov) [1]

TOPIC: Programming systems

CHALLENGE:

Co-design methods often require system designers and application developers to communicate their requirements via models and specifications. System designers may provide simulators that can mimic systems that do not yet exist. Application developers will provide reduced programs or mathematical models that capture the behavior of their application without the complexity of the full application. A key challenge in creation of these application models is the mapping process from code to model, in particular the problem of characterizing dynamic properties of the programs. Prior work in derivation of program skeletons [1] [2] demonstrated that complex programs can be reduced to skeleton applications via static analysis techniques with user guidance. These skeletons are limited in that the removal of code as part of the skeletonization process removes computations that define values that data takes on at runtime. This data very likely impacts control flow properties that are critical to capturing realistic properties of program behavior. A key challenge is to characterize this state within a skeleton without retaining the full code that performs the calculation. We believe that this is similar to the challenge in software assurance contexts of generating test inputs and reasoning about data flow state in programs via formal program analysis.

OPPORTUNITY:

The development of cost models for programs relative to current or future systems allows performance to be predicted without explicitly executing the code. This is extremely important at stages during system development when performant instances of a system do not exist yet and designers rely on simulators to predict the impact of design decisions. Execution of programs that require large scale compute hardware in a simulation environment is simply not possible. As demonstrated in the context of existing simulators [3], automatically derived skeletons can be used to make performance predictions. We believe that extending the analysis used to derive skeletons to include formal analysis and characterization of data state will make automatically generated skeletons and related performance models more useful when considering programs with runtime behavior that is highly dependent on data.

Recent advances in test generation and data flow analysis for software correctness and testing purposes can be applied in this performance modeling context, although research remains to apply those techniques to large programs and ensure good coverage of possible program behaviors [4] [5]. In particular these techniques focus on reasoning about the plausible values

[1] This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

that one can expect data to take on at runtime. When augmented with human provided annotations that express constraints not inferable directly from source code, constraint solvers and model checkers can be used. In addition to classic program skeletons, where a skeleton is represented as a program in the same language as the application from which it was derived, we believe there are additional opportunities to build models that borrow from these formal analysis methods as well. For example, abstracting a program from a specific language to a higher-level state transition system that can be used to analyze concurrency and control flow properties.

TIMELINESS:

Advances in formal analysis tools in the last decade, particularly constraint satisfaction and model checking methods, has made it possible to reason about program execution beyond basic static analysis. In the last decade a number of efforts have combined SMT solvers with compiler and program transformation frameworks to perform data flow analysis and symbolic execution. These tools have been developed largely in testing and correctness contexts but can be repurposed to build richer performance models of programs by augmenting structural features with approximate runtime properties. We envision applications of such models beyond their traditional use to drive simulators for systems during the design process. For example, significant interest is arising in program synthesis methods in which one or more candidate programs may be automatically assembled to execute on a target system. In the case of HPC applications the evaluation of these candidate programs to pick the best cannot rely on direct execution: a predictive model of their expected performance is necessary to rapidly explore the space of potential programs that can be synthesized.

Bibliography

[1] M. Sottile, J. Dagit, D. Zhang, G. Hendry and D. Dechev, "Static Analysis Techniques for Semiautomatic Synthesis of Message Passing Software Skeletons," *ACM Transactions on Modeling and Computer Simulation,* 2015.

[2] J. J. Wilke, J. P. Kenny, S. Knight and S. Rumley, "Compiler-Assisted Source-to-Source Skeletonization of Application Models for System Simulation," in *Proceedings of the 2018 International Conference on High Performance Computing*, 2018.

[3] G. Hendry and A. Rodrigues, "SST: A simulator for Exascale Co-design," in *ASCR/ASC Exascale Research Conference*, 2012.

[4] J. Rushby, "Automated Test Generation And Verified Software," in *Position paper for VSTTE*, Zurich, 2005.

[5] A. Kolchin, S. Potiyenko and T. Weigert, "Challenges for Automated, Model-Based Test Scenario Generation," in *Proceedings of the International Conference on Information and Software Technologies (ICIST)*, 2019.

Title: Codesign for Experimental Workflows at User Facilities
Author: Christine Sweeney, Los Alamos National Laboratory, cahrens@lanl.gov
Topic: codesign methodologies, applications
Challenge:
DOE experimental user facilities such as light sources and other accelerators have moved into the arena of data-intensive computing and even compute-intensive applications as facilities are upgraded. Data volumes and rates are increasing, new capabilities are enabled by the upgrades, and analysis workflows [1-4] and facility control workflows are increasingly challenged by a number of requirements:

- Sophisticated data science and compression techniques: Experimental scientists increasingly want faster and better analytics that include multi-modal data and can be done in real-time to help steer experiments. The workflow is often limited by I/O, reading in large data volumes, and reducing them to the format needed for analysis. In some cases, it is an analysis that can be done by streaming through an FGPA (detecting "hits" versus "misses" in crystallography data) and in other cases, such as when reconstructing images of samples based on long-running scans, it requires checking data quality, compressing, and possibly reformatting before reconstruction.
- Compute-intensive analytics and simulations: As light sources become brighter, for example, more imaging can be done and the analytics for these can be computationally-intensive, in some cases requiring inter-facility workflows to process the data at supercomputer facilities. Experimentalists want comparison with simulations even during experiments or between shifts, for example, to help with prediction and to supplement analytical solutions when they are not unique.
- Distributed sensor monitoring, data collection and coordination: New sensors and edge devices are just coming online to meet challenges of upgraded facilities, so have not been integrated into control and/or experimental workflows previously and have new performance characteristics. For example, mechanical sample delivery for automating dynamic compression experiments will transform the workflow.
- Widely different experimental workflows for experiments: Diverse applications make it hard to optimize and codesign solutions that will work for many. For example, long-running experimental workflows such as nanocrystallography produce orders of magnitude more data than do dynamic compression workflows, however each has their own real-time challenges. Human interaction and decision making aspects also vary widely.

In order to continue to forge new scientific discoveries and more fully utilize these upgraded science facilities, the computational workflows associated with experimental data and facility control need hardware and software codesign as well as codesign with a number of communities such as applied mathematics, statistics, mechanical and electrical engineering, computational science, and computer science.
Opportunity:
Key to being able to codesign these experimental workflows is a better understanding of them now and as they evolve. Light sources such as LCLS have done detailed studies on data rates and volumes for a number of experiments in order to better understand how to tackle data compression and data transfer needs going forward. However, widespread, cross-cutting

understanding of the performance of these diverse workflows is needed. The following opportunities emerge as ones that will enable codesign in this domain:

- Experimental workflow performance monitoring systems for user facility sensor, network, storage, memory, compute and user interaction: The opportunity here is to make an independent, portable framework that measures performance of resources at experimental user facilities. It is important that this tool work independently of workflow management systems(WMS) so that measurements can be made across those systems, however provide an interface so that the WMS can do this monitoring also.
- Performance modeling tools that correspond to resources monitored above in order to create and predict performance of a diverse set of complex workflows: This may require coordination between computing systems and beamline control systems. While some performance monitoring systems are being made for larger supercomputing and interfacility workflows, the edge, local network, local storage and compute workflows at experimental facilities are less characterized. We need to better understand how to overlap, distribute, and parallelize computation and make way for creative hardware solutions and resource utilization unique to experimental facilities.
- Proxy applications: Proxy applications have not typically been used for computations for experimental workflows, but have focused on large-scale scientific computing applications. There is an opportunity to select subsets of experimental workflows as proxy applications to start providing a view of these to hardware vendors. Examples of proxy applications would be those that highlight common operations on experimental data, such as compression methods, data calibration or fast analytics for accept/reject decisions based on control and beamline sensor data. Proxy applications can also cover automated workflows that employ control methods to change beamline or user experiment parameters.

Timeliness:
It is quite timely to work toward the discovery, modeling and proxy applications for experimental workflow resource usage, since facilities are being upgraded now and beamtimes are scarce and expensive. Edge computing is burgeoning for experimental science and many are currently interested in better harnessing it in their workflows now. The impact of success would be that experimentalists can better understand resource usage and optimize it, real-time workflows can be enabled more often to make better use of beam time, and better science will result, as intended by the facility upgrades.

[1] Sweeney, C., et al., Data Science and Computation for Rapid and Dynamic Compression Experiment Workflows at Experimental Facilities. Workshop Report. 2021. LA-UR-20-24310. https://www.lanl.gov/conferences/compress/
[2] Sweeney, C., et al., Gap Analysis: Materials Discovery through Data Science at Advanced User Light Sources. Workshop Report. 2019. LA-UR-19-21342. https://www.lanl.gov/2018gapanalysis
[3] Bethel, E. W et al. Report of the DOE Workshop on Management, Analysis, and Visualization of Experimental and Observational data – The Convergence of Data and Computing. 2016.
[4] Peterka, Tom, et al. ASCR Workshop on In Situ Data Management: Enabling Scientific Discovery from Diverse Data Sources. 2019.

Resilience-Aware Codesign: A Full-Stack Approach

Keita Teranishi, Jeremiah Wilke, Michael P. Frank, Kurt B. Ferreira, and Jackson R. Mayo
Sandia National Laboratories, Livermore, CA, and Albuquerque, NM, USA
knteran@sandia.gov, jjwilke@sandia.gov, mpfrank@sandia.gov, kbferre@sandia.gov, jmayo@sandia.gov

Topics

Architectures, Codesign Methodologies, Applications, Programming Systems

Challenge

Dennard scaling has been over more than a decade and Moore's law is near its end. Therefore, the conventional CMOS architectures that comprise the Top500 supercomputers have only a few generations left, with rapidly diminishing returns. Improvements are being sought in numerous directions, ranging from similar architectures based on new materials and physics [1], specialization and customization within the CMOS world, new paradigms like non-von Neumann and data flow, or dramatically new ways of computing including reversible, neuromorphic, and quantum computing.

Exascale will be achieved largely through GPGPU accelerators, fighting off dire predictions that dramatic hardware changes would be needed. Similarly, once gloomy predictions of failures for exascale have not wholly materialized. Checkpoint/restart remains the dominant model for fail-stop faults, albeit including I/O optimizations like in-memory checkpointing, burst buffers, and acceptable mean time between failures (MTBF). Error correction in hardware is trusted to correct soft errors (though still actively studied [5, 12]). Programming models and other redundant computing strategies have advanced, yet not seen wide adoption. As a result, hardware and system approaches attempt to isolate faults and errors in an effort to avoid mitigation at the application level. Though applications themselves often do not address resilience (beyond checkpointing), significant costs for fault mitigation are already a fact of life in HPC. Even if current standard design approaches (*e.g.*, built-in hardware error correction, software checkpoint/restart) remain feasible, they are not necessarily optimal. Therefore, pushing beyond exascale will require significant innovation and paradigm shifts. Moreover, extreme heterogeneity will challenge established strategies like checkpoint/restart. Finally, the power-performance tradeoffs for resilience deemed acceptable today may be dramatically altered. This motivates codesign for resilience.

Research should seek potential benefits in overall cost, performance, reliability and usability, via new tradeoffs in hardware and software design that affect the origin and mitigation of faults. For example, current hardware and system reliability (and hence power) requirements could be relaxed if more faults can be tolerated at the application level. The value of such work is increased by the trend to increasingly customized and heterogeneous hardware in HPC. Alternatively, one may consider reversible computing, which represents a non-traditional paradigm for digital computing that offers an unconventional path for improving the power-efficiency of systems while maintaining high reliability, via *recovering* logic signal energies rather than reducing them to the point where gate-level reliability is impaired. However, reversible computing requires significant architectural changes (*e.g.*, adoption of reversible hardware algorithms) which incur their own overheads. A proper codesign analysis would also be needed to compare a reversible computing approach to other methods for resiliently scaling the power-efficiency of systems. Both perspectives will be critical to addressing resilience. We should not neglect addressing resilience in the application if unreliable devices provide power/performance benefits, nor should unreliable or low-voltage devices be accepted as inevitable.

Opportunities

Overall, our idea of codesign still addresses the same problem of optimizing performance, power, productivity and reliability. The major questions are what type of errors and faults need to be mitigated or isolated at the hardware or middleware levels, how to manage the complex tradeoffs of performance, power, and reliability while keeping the computing system usable, and what type of error or fault information needs to be reflected up and across the system stack. We call out the need for five critical advances in future codesign to address resilience:

- Hardware experts delivering fault-tolerance models accessible to software developers
- Benchmarking on test beds and development of a centralized repository of observed faults on actual systems to inform and validate models
- Application developers engaged in developing kernel-specific resilience methods beyond generic approaches like triple modular redundancy
- Programming models providing abstraction of unreliable execution environment to enable a variety of fault tolerance techniques in a usable manner
- System simulation and modeling tools capable of evaluating both built-in hardware fault tolerance and ABFT strategies.

The computer architecture and fault-tolerant algorithms space already offers many opportunities to pursue such codesign, which we address below.

Timeliness and Maturity

Numerous advances in architectures, ABFT, and programming models provide an excellent foundation for rethinking how we approach resilience in co-design. The prime example of fault-tolerance codesign directly in the algorithm is quantum computing [4]. Beyond quantum, several architecture proposals pose new fault-tolerance challenges. Low-voltage, "unreliable" devices promise dramatic power reductions. Reliably computing on these devices, however, may require rethinking the instructions sets as seen in proposals, e.g., based on the residue number system [6]. Similarly, ABFT techniques such as those found in [16] and [3] may tolerate unreliable, low-voltage devices. Coarse-grained reconfigurable architectures may dramatically limit data movement relative to von Neumann architectures. While von Neumann models lend themselves to obvious error-correcting code (ECC) strategies on memory or registers, it is less clear how to optimize error correction for data-flow architectures. If data-flow accelerators provide abundant parallelism, it may be possible to incorporate some redundant compute in kernels with limited overhead. Analog computing, particularly for individual kernels like matrix-multiplication, could provide dramatic performance and power gains, but reliability of such devices remains to be studied [7]. Even if such devices regularly experience faults, it may be possible to make them resilient through ABFT techniques, e.g., [16] for matrix-multiplication. Neuromorphic architectures can provide power-performance improvements, most obviously for neural networks and machine learning. Neuromorphic algorithms for other domains like graph analytics or random walks have been proposed [2]. Certain algorithm utilizing spiking neuromorphic devices embed information temporally [11], not spatially, which clearly requires a dramatic shift in thinking from simple ECC strategies to address fault-tolerance.

The ABFT community has been exploring a new opportunity for codesign and better integration with resilience techniques accommodated by systems and programming models [9]. Additionally, there has been increased attention to randomized methods [10] for numerical and combinatorial algorithms, which significantly reduce computation through random sampling techniques. These algorithms, delivering similar accuracy to conventional deterministic algorithms, are potentially tolerant to occasional soft errors because of the way they handle randomness in the computation.

Fault tolerance support in parallel programming models have been explored for the past several years. The MPI standardization committee have been discussing a number of fault tolerance extensions (ReInit [8] and User Level Fault Mitigations [13]) as alternatives to global checkpoint/restart. Recently, a resilient extension of Kokkos [14] has been proposed to leverage performance portable abstractions and enable resilient program execution and data objects for unreliable computing systems. Additionally, there have been emerging task parallel models resilience extensions to enable localized program recovery [15].

Acknowledgments

This work is supported by by Sandia National Laboratories, a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

[1] International roadmap for devices and systems 2020 edition. https://irds.ieee.org/editions/2020.

[2] J. B. Aimone et al. Composing neural algorithms with fugu. In *International Conference on Neuromorphic Systems*, ICONS '19. Association for Computing Machinery, 2019.

[3] J. Chen, S. Li, and Z. Chen. GPU-ABFT: Optimizing algorithm-based fault tolerance for heterogeneous systems with GPUs. In *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 1–2, 2016.

[4] E. Conover. To live up to the hype, quantum computers must repair their error problems. https://www.sciencenews.org/article/quantum-computers-hype-supremacy-error-correction-problems.

[5] T. J. Dell. A white paper on the benefits of chipkill-correct ECC for PC server main memory. IBM Microelectronics Division, Nov. 1997.

[6] B. Deng et al. Extending moore's law via computationally error-tolerant computing. *ACM Trans. Archit. Code Optim.*, 15(1), Mar. 2018.

[7] E. Fuller et al. Li-ion synaptic transistor for low power analog computing. *Advanced Materials*, 29, 11 2016.

[8] G. Georgakoudis et al. Reinit^{++}: Evaluating the performance of global-restart recovery methods for MPI fault tolerance. In *ISC 2020*, volume 12151 of *Lecture Notes in Computer Science*, pages 536–554. Springer, 2020.

[9] L. Giraud, U. Rüde, et al. Resiliency in Numerical Algorithm Design for Extreme Scale Simulations. *Dagstuhl Reports*, 10(3):1–57, 2020.

[10] N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53:217–288, 2011.

[11] A. J. Hill et al. A spike-timing neuromorphic architecture. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, 2017.

[12] L. Jaulmes et al. Runtime-guided ECC protection using online estimation of memory vulnerability. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

[13] N. Losada et al. Fault tolerance of MPI applications in exascale systems: The ULFM solution. *Future Gener. Comput. Syst.*, 106:467–481, 2020.

[14] J. S. Miles et al. Software resilience using Kokkos ecosystem. Technical Report SAND2019-3616, Sandia National Laboratories, 2019.

[15] S. R. Paul et al. Enabling resilience in asynchronous many-task programming models. In R. Yahyapour, editor, *Euro-Par 2019*, pages 346–360, 2019.

[16] P. Wu et al. Fault tolerant matrix-matrix multiplication: Correcting soft errors on-line. In *Proceedings of the Second Workshop on Scalable Algorithms for Large-Scale Systems*, ScalA '11, page 25–28. Association for Computing Machinery, 2011.

Reimagining Hardware/Software Codesign Tools for Domain Specialization

Antonino Tumeo, Marco Minutoli, Vito Giovanni Castellana, Vinay Amatya, Nicolas Bohm Agostini,
Joseph Manzano, Shihao Song, Cheng Tan
Pacific Northwest National Laboratory
Richland, WA, USA
{antonino.tumeo, marco.minutoli, vitoGiovanni.castellana, vinay.amatya, nicolas.agostini,
joseph.manzano, shihao.song, cheng.tan}@pnnl.gov

Topic: Codesign methodologies, architectures

I. Introduction

As we approach the end of Moore's Law, computing systems at all scale, from edge devices to supercomputers, will have to resort to the use of domain-specific processing elements to keep providing performance improvements in tightly constrained power envelopes. At the extremes, we can expect the appearance of computing systems integrating modular processors with seas of accelerators [1], where the accelerators themselves may present different levels of configurability (from simple parameters, to array of coarse grained functional units, to bit-level configurability such as the one provided by Field Programmable Gate Arrays).

While the programmability challenges posed by heterogeneous systems are not new, the need to exploit extreme specialization, coupled with the appearance of modular compiler technologies, open source hardware and licensable Intellectual Property (IP) Cores, and new chip integration methods, is creating new opportunities for reimagining tools able to quickly transition from algorithmic specification to their hardware description and implementation.

II. Challenges and Current Approaches

We are moving towards a new generation of applications where algorithms with significantly different behaviors are composed in complex workflows that alternatively are compute-intensive (e.g., scientific simulation), memory-intensive (e.g., data analytics, data structure preprocessing through graph algorithms), or both (e.g., machine learning methods that operate on large dense or sparse tensors).

A domain specialized accelerator for one phase may not adapt to the subsequent phase, requiring the composition of a complex heterogeneous system. At the system level, thus, the challenge is making sure that the adequate accelerators are select, and/or generated, considering the algorithmic computational patterns, without directly requiring the intervention of an expert hardware designer. Hand-designing accelerators is, arguably, time consuming, and does not even consider the added system programmability challenges. Even in the case of Application Specific Integrated Circuits (ASICs), the generation methodology needs to be aware of specific features (analysis) of the patterns to accelerate, the metrics, and the constraints, to allow exploration along a variety of old (area, performance, energy) and new (security, cooling, size) metrics.

Some of these accelerator generation challenges have somewhat been previously discussed in the area of fine-grained (re)configurable devices (FPGAs). While these devices have been around for a long time, they have mainly been used to implement *mission critical* custom architectures for designs where the low volume could not justify ASICs production. However, the possibility to instantiate domain-specific accelerators after deployment adapts well to novel emerging application areas such as machine learning and data analytics, where the algorithms (and, thus, the computational pattern) keep quickly evolving and fixed domain accelerators may support only subsets of progressively less relevant methods (or only the most general computational patterns). Because High-Level Synthesis (i.e., the process that allows generating hardware description instantiable on the FPGAs starting from higher-level languages) historically focused on approaches mostly considering digital signal processing (DSP) worklaods, research and industry have renewed their commitments to develop new synthesis tools (or new "libraries" of components) for various types of parallel computational patterns, with different degrees of success - but typically focusing only on very specific domains.

The performance/flexibility/adaptability tradeoffs between bit-level configurable devices and fixed accelerators also led research to restart investigating *coarser grained* reconfigurable designs, which promise to provide higher performance through specialized functional units while maintaining adaptability through quick runtime reconfiguration of the computational substrate. This led to the appearance of a plethora of new Coarse Grained Reconfigurable Array (CGRA) like designs and dataflow architectures (e.g., SambaNova, GraphCore, Cerebras, NextSilicon, etc.). However, the variety of architectures highlights how there is not yet clarity on how such architectures needs to be actually organized, and how the key challenges are in the software toolchains, typically limited by the fact that each and every platform adopts its own abstractions and related infrastructures.

III. Opportunity

Retargetable opensource compiler frameworks, which have been a staple point for the realization of commercial and research High-Level Synthesis tools [2], have today reached a level of maturity that can really enable the development of modular and reusable, custom hardware generators. Retargetability allows, for example, to instantiate appropriate frontends for different input abstractions. The variety of domain specific languages (with different levels of appreciation depending on the domain science and scientist) and/or general purpose languages (with large amount of existing complex applications that can still benefit from domain specialization), requires a decoupling of the frontend from the middle end. At the same time, there is a need for these infrastructures to initiate architecture independent optimization and design space exploration at the front-end level (as early as possible), to maximize the benefit of user-provided information. Approaches such as the *MultiLevel Intermediate Representation* (MLIR) [3] which allows building reusable and extensible compiler infrastructures enables defining such decoupling and the "High-Level IRs" to apply optimizations at the right abstraction level. Furthermore, they provide opportunities to define "attributes" and extensions (e.g., to existing languages) to drive hardware design space exploration and optimizations along novel metrics (e.g., security).

Middle ends (such as the actual LLVM compiler) can today leverage a richness of algorithmic solutions to generate highly optimized machine code, and can be relatively quickly retargeted to new instruction sets (or instruction set extensions). Many of the compiler algorithms already available are basilar for hardware synthesis, especially in the case of Finite State Machines with Datapath (FSMD), which leverage instruction level parallelism and static scheduling. Interfacing with novel HLIRs, with their natural support for hierarchy and (task level, coarse grained) parallelism, opens further opportunities in generating and composing hierarchical hardware systems.

The hardware synthesis process can greatly benefit from the availability of opensource or licensable hardware IPs, which can become part of the resource library for such compiler-based toolchains, enabling algorithmic and hierarchical system-level design. This not only accounts for opensource instructions sets (such as RISC-V) but also templated accelerators [4] or even functional units. Compiler-based generators enable exploring the design space and setting parameters for these components (e.g., precision). Hence, they directly tie to configurability of templated components. Additionally, they provide a path to supporting reconfigurability, for example leveraging just-in-time compilation, where the IR (bit code) can be lowered to slighlty different machine code depending on the overall system status. On the line of modularity, employing circuit-level IRs (e.g., FIRRTL, CIRCT) [5] before actually generating the designs in hardware languages provides yet another level of decoupling, enabling composability of hardware modules and the possibility of optimizing the final design depending on the actual device technology (different types of devices, technology nodes, or logic cells).

Employing compiler based toolchains also provide unique opportunities to implement profile driven hardware synthesis. Akin to profile driven compilation, beside leveraging typical static analysis, hardware synthesis can also benefit from dynamic analysis, especially for data dependent workloads. Modern compiler toolchain can easily interface with instrumentation and profiling tools (e.g., binary instrumentation), and even provide such functionalities through appropriate compiler passes. The resulting information can be fed back to drive the synthesis process. This is especially crucial with novel memory intensive workloads, which dramatically change the typical hardware generation approaches focused on compute first. Compiler retargetability even allows profiling on a given host architecture, and reuse the information for the synthesis. Dynamic analysis becomes even more critical if targeting configurable, or reconfigurable components, which can modify behaviors and parameters as program execution proceeds. Obviously, these mechanisms also require a close collaboration with the hardware component designers, to provide necessary hardware hooks, and runtime layers, to support monitoring.

Furthermore, design space exploration calls for integration with modeling and simulation methods. Hardware synthesis algorithms require metrics to drive the optimization process. Such techniques need to provide quick estimation of the metrics, along the different dimensions, possibly without performing actual simulation and logic synthesis of the identified design point. This obviously need the development effective models for performance, area, and energy. Modular compiler based toolchains can streghten such integration, enabling association of metrics to the operations represented in the IRs, and opening opportunities for automating model building.

IV. Related frameworks

OpenCGRA [6]: OpenCGRA is a parameterizable opensource CGRA (Coarse-Grained Reconfigurable Arrays) generator based on user-specified configurations (e.g., size, type of the computing units in each tile, communication connection, etc.). Implemented by leveraging PyMTL, OpenCGRA uses a modular design and standardized interfaces between modules to generate synthesizable Verilog of the designs.

SODA Synthesizer [7]: the Software Defined Architectures (SODA) is a new modular opensource synthesis infrastructure. Leveraging a variety of community effort, SODA supports a variety of domain specific and general purpose languages that interface with the Multi-Level Intermediate Representation (MLIR) compiler infrastructure, implements a synthesis backend fully integrated within the LLVM framework, and is able to generate a circuit representation in FIRRTL. The backend can interface with OpenCGRA, or generate RTL for FPGAs and ASICs, supporting both opensource (e.g., OpenROAD) [8] and commercial logic synthesis tools. The whole design flow will be able to perform optimization and design space exploration at the front-, middle, and back-end levels.

V. Timeliness

Novel converged workloads, integrating scientific simulation, data analytics, and machine learning, coupled with waning benefits of technology scaling, requires complex systems integrating many domain-specific accelerators. At the same time, new methods of integration (e.g., chiplets), the appearance novel configurable/reconfigurable devices, the availability of opensource or licensable IPs, and new community efforts in the area of compilers, are creating a unique opportunity to develop novel, modular, and extensible hardware generators. These generators have the potential to bridge not only some of the programmability gaps, but also bridge the design gap from algorithmic specification to hardware implementation.

References

[1] A. A. Chien, T. T. Hoang, D. P. Vasudevan, Y. Fang, and A. Shambayati, "10x10: A case study in highly-programmable and energy-efficient heterogeneous federated architecture," *SIGARCH Computer Architecture News*, vol. 43, no. 2, pp. 2–9, 2015.

[2] "Bambu: A Free Framework for the High-Level Synthesis of Complex Applications, available at: https://panda.dei.polimi.it," 2021.

[3] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko, "MLIR: A Compiler Infrastructure for the End of Moore's Law," 2020.

[4] M. Minutoli, V. G. Castellana, N. Saporetti, S. Devecchi, M. Lattuada, P. Fezzardi, A. Tumeo, and F. Ferrandi, "Svelto: High-Level Synthesis of Multi-Threaded Accelerators for Graph Analytics," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[5] A. Izraelevitz, J. Koenig, P. Li, R. Lin, A. Wang, A. Magyar, D. Kim, C. Schmidt, C. Markley, J. Lawson, and J. Bachrach, "Reusability is FIRRTL ground: Hardware construction languages, compiler frameworks, and transformations," in *ICCAD 2017*, 2017, pp. 209–216.

[6] C. Tan, C. Xie, A. Li, K. J. Barker, and A. Tumeo, "opencgra: An open-source unified framework for modeling, testing, and evaluating cgras."

[7] M. Minutoli, V. G. Castellana, C. Tan, J. B. Manzano, V. Amatya, A. Tumeo, D. Brooks, and G. Wei, "SODA: a New Synthesis Infrastructure for Agile Hardware Design of Machine Learning Accelerators," in *ICCAD 2020*.

[8] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu, "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project," in *DAC '19*.

Unifying the HPC and Edge Programming Models

John Paul Walters, USC Information Sciences Institute
jwalters@isi.edu

Topic: Programming Systems

Challenge:
Heterogeneity and specialization have led to a new generation of high performance architectures. These architectures span a variety of power and performance profiles, from resource-constrained (though increasingly powerful) edge devices to leadership class supercomputers.

Today's approach to software development for both HPC and edge computing requires either computing experts (HPC or edge) to become domain experts, or vice versa. Either case comes at great cost, taking time away from the research problem at hand and slowing momentum. Previous efforts to port highly scalable and performant code to the Titan supercomputer at Oak Ridge National Lab required 2 FTE years per codebase [1]. Domain specific approaches are common, but traditionally cannot bridge the performance portability gap and have not yet been shown capable of targeting systems of vastly differing scales, with unique performance, power, and reliability constraints. Instead, software is developed, verified, and debugged on every system of interest.

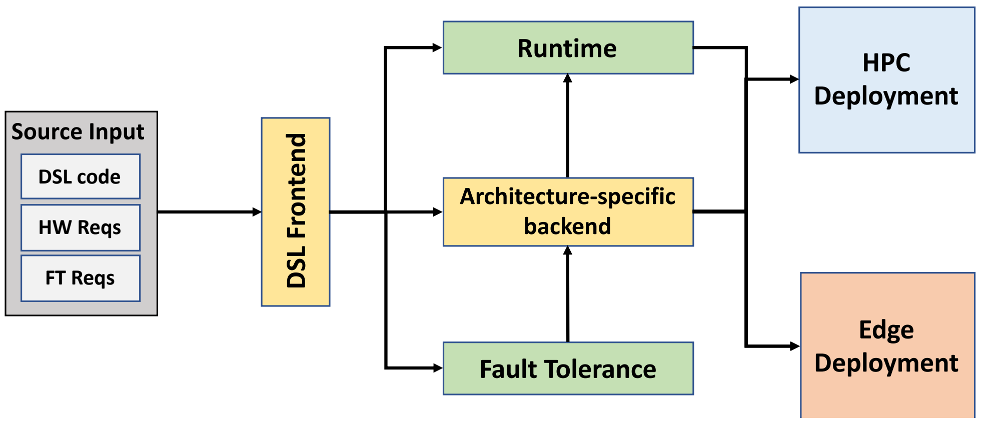


Figure 1: Notional DSL compiler with support for heterogeneous deployment scenarios

Opportunity:
A new generation of domain-specific programming models provides an opportunity to unify edge computing, HPC, and intermediate systems and scales of interest. Domain-specific programming models have been shown to offer substantial performance portability, particularly within the supercomputing space, enabling domain experts to rapidly express scientific simulations from computational fluid dynamics to radar applications [2]. These domain-specific approaches, in combination with next-generation compilers and intermediate representations such as MLIR, offer opportunities for rapid adaptation to multiple accelerator technologies and heterogeneous

resources. This may represent a major step for high performance computing, but on its own is insufficient to unify the HPC to edge computing spectrum and their programming models.

Instead, the next generation of domain-specific programming models must be capable of optimizing across multiple, radically different, architectures as shown in Figure 1. A domain-specific language should be capable of producing an optimized implementation for both today's leadership class supercomputers and resource-constrained edge devices, including support for fault tolerance, reduced precision, and more. This will require advances in system modeling, performance prediction, domain-specific compilers, and runtimes.

The architecture envisioned in Figure 1 accepts a standard domain-specific application along with fault tolerance requirements, deployment/architecture requirements. The backend emits architecture-specific object code including support for fault tolerance according to the deployment and reliability characteristics requested by the programmer. Likewise, an application-specific and architecture-specific runtime is emitted which is responsible for monitoring the execution of the application via introspection, heartbeats, etc. and adapting the execution for improved performance, power efficiency, or fault tolerance. The runtime may, for example, rebalance computation across a supercomputer due to load imbalance that cannot be detected at compile time. On an edge deployment, the runtime may observe increasing memory failures and disable a memory rank to improve reliability.

Timeliness:
Now is the time for investment in the next generation of domain-specific programming models. The research community has seen a proliferation of DSLs with increasing capabilities, from Halide for computer vision [3] to Spiral [4] for signal processing and numerical kernels and CASPER [2] for high performance radar and CFD applications. The DARPA PAPPA program has advanced domain-specific programming models specifically targeting heterogeneous targeting high performance computing resources. As a result, the community is well-positioned to extend this work to the edge computing space.

References:

[1] W. Joubert *et al.*, “Accelerated Application Development,” *Comput Electr Eng*, vol. 46, no. C, pp. 123–138, Aug. 2015, doi: 10.1016/j.compeleceng.2015.04.008.
[2] C. Imes, A. Colin, N. Zhang, A. Srivastava, V. Prasanna, and J. P. Walters, “Compiler Abstractions and Runtime for Extreme-scale SAR and CFD Workloads,” presented at the IEEE/ACM 5th International Workshop on Extreme Scale Programming Models and Middleware (ESPM2), 2020, doi: 10.1109/ESPM251964.2020.00010.
[3] J. Ragan-Kelley *et al.*, “Halide: Decoupling Algorithms from Schedules for High-Performance Image Processing,” *Commun ACM*, vol. 61, no. 1, pp. 106–115, Dec. 2017, doi: 10.1145/3150211.
[4] F. Franchetti *et al.*, “SPIRAL: Extreme Performance Portability,” *Proc. IEEE Spec. Issue “From High Level Specif. High Perform. Code,”* vol. 106, no. 11, 2018.

Co-Designing Scalable Hardware and Algorithms with Application-Driven Benchmarks

Jean-Sylvain Camier (LLNL), **Paul Fischer** (UIUC), **Tzanio Kolev** (LLNL), **Misun Min** (ANL), **Stanimire Tomov** (UTK), **Tim Warburto**n (Virginia Tech)

The Center for Efficient Exascale Discretizations (CEED) is a focused team effort within the U.S. Department of Energy (DOE) Exascale Computing Project (ECP) that is developing the next-generation discretization software and algorithms to enable a wide range of finite element applications to run efficiently on exascale hardware [1]. In this whitepaper we present lessons learned and propose future directions based on the co-design research and development activities in the CEED project.

High-Order Applications. Efficient exploitation of modern architectures requires rethinking of the numerical algorithms for solving partial differential equations (PDEs) on general unstructured grids. Many of these new architectures, such as general purpose graphics processing units (GPUs) favor algorithms that expose ultra fine-grain parallelism and maximize the ratio of floating point operations to energy intensive data movement. In large-scale PDE-based applications that employ unstructured finite element discretizations, practical efficiency is measured by the accuracy achieved per unit computational time. One of the few viable approaches to achieve high performance in this case is to use matrix-free high-order finite element methods, since these methods can both increase the accuracy and/or lower the computational time due to reduced data motion. To achieve this efficiency, high-order methods use mesh elements that are mapped from canonical reference elements and exploit, where possible, the tensor-product structure of the canonical mesh elements and finite element spaces. Through matrix-free partial assembly, the use of reference elements enables substantial cache efficiency and minimizes extraneous data movement in comparison to traditional low-order approaches.

As a co-design center positioned between applications and hardware vendors the CEED efforts have been focused on two major goals: 1) Help applications leverage architectures by providing them with state-of-the-art high-order discretization algorithms that better exploit the hardware and deliver significant performance gain over conventional low-order methods; and 2) Collaborate with hardware vendors to utilize and impact hardware design and its software stack through proxies and miniapps. In achieving these goals, we have found the following factors to be important and we believe that they will continue to be important for future co-design activities:

Scalable hardware that not only achieves high on-node peak performance but requires small amounts of data on the node to achieve e.g. 80% of that peak, see the key $n_{0.8}$ parameter in [2]. This is critical for strong scaling and reducing the time to solution. Low system noise and reproducibility are also important for algorithmic development.

Benchmarks that are not synthetic but motivated by applications, incorporating key local and global kernels in model problem settings. We have found that such application-relevant performance testing and analysis (as opposed to e.g. using Top500 benchmarks) is critical to effective HPC software deployment and achieving our first goal of real application impact. We have also found it useful to develop a hierarchy of benchmarks that are inter-connected but serve different purposes:

- ***Streaming Benchmarks.*** These represent basic memory-bound linear algebra operations that give us bounds on expected performance of the more complex benchmarks, and are directly relevant to performance modelling discussed below.

- ***Bake-off Problems (BPs).*** These represent the simplest PDE-motivated application kernel, e.g. conjugate gradient iterations with a mass matrix, but combine local dense linear algebra with MPI communication and global scatter/gather and reduction operations. The BPs were designed to establish best practices for performant implementations of high-order methods across a variety of platforms. This has allowed us to pool the community efforts of multiple high-order development groups to identify effective code optimization strategies for candidate architectures. In addition to driving algorithm design and providing meaningful goals for vendor optimizations (e.g. size of on package memory, internode latency, kernel launch overheads, hardware collectives), the BPs are also useful for comparing different HPC systems and identifying the hardware bottlenecks to application performance.

- ***Miniapps.*** One step above the BPs are the miniapps, which are simple yet capture application-relevant physics to work with vendors, be used in system procurement, collaborate software technologies projects, and provide test and demonstration cases for application scientists. One of their uses is to highlight performance critical paths with the goal to impact the design of exascale architectures, and system and application software, for improved portability and performance of the high-order algorithms. A good miniapp should cover at least 80% of key kernels in a distinct part of the application, and so combining with the $n_{0.8}$ parameter above we aim for "80:80:80", i.e. 80% MPI scalability at 80% of device throughput for 80% of representative app capabilities.

Performance and Energy Efficiency Models that are important to understand the benchmarks and directly inform hardware design choices and trade-offs. In addition to performance benchmarks and models, we have previously investigated the inclusion of benchmark capabilities to evaluate energy efficiency. This would allow us to incorporate power-efficient computations in the CEED software and produce feedback of value to hardware designers. For example, we have established important cases where capping power consumption leads to improved power efficiency of up to 20% without loss of performance [3].

[1] ECP co-design Center for Efficient Exascale Discretizations, https://ceed.exascaleproject.org.
[2] P. Fischer, M. Min, T. Rathnayake, S. Dutta, T. Kolev, V. Dobrev, J.-S. Camier, M. Kronbichler, T. Warburton, K. Swirydowicz, and J. Brown, Scalability of high-performance PDE solvers, *The International Journal of High Performance Computing Applications*, 34(5): 562–586, 2020.
[3] A. Haidar, H. Jagode, P. Vaccaro, A. YarKhan, S. Tomov, and Jack Dongarra, Investigating Power Capping toward Energy-Efficient Scientific Applications, *Journal of CCPE*, 31 (6), 2019.

Systemwide Codesign with Community Software Stack

Johannes Doerfert*[1], Salman Habib†[2], Christopher Knight‡[3], Scott Parker§[4], Venkatram Vishwanath¶[5], and Timothy Williams‖[6]

[1]Mathematics and Computer Science Division, Argonne National Laboratory
[2,3,6]Computational Science Division, Argonne National Laboratory
[4,5]Argonne Leadership Computing Facility, Argonne National Laboratory

February 2021

1 Topic

architectures, applications, programming systems, codesign methodologies

2 Challenge

Current codesign activities in conjunction with next-generation supercomputer procurements are often too focused on (a) achieving performance at a single-node or fine-grained hardware level and (b) using vendor-specific software to assess a limited class of performance metrics (such as designing a new language around a new accelerator, creating a development environment which is by definition not portable). In addition, the widening of the mission space of leadership-class facilities to support complex modeling and simulation tasks and data-intensive and AI-based application components is a rapidly evolving trend that should be addressed through co-design efforts. The appearance of these application classes as peers to more traditional high performance computing (HPC) is highly significant. Without due recognition of this fact, the use of conventional codesign practices will lead to an increased gap between the real-world demands posed on the systems and the design and performance metrics by which they are being designed and formally evaluated. The existence of this gap must therefore be properly recognized and taken into account in any holistic codesign process.

The relatively narrow focus at the node level can lead to missing out on important opportunities at system-scale codesign. These opportunities, when properly exploited, can significantly increase the scientific return by supporting new classes of applications as well as by optimizing the way science campaigns are actually carried out on the machines. Examples of these opportunities include data-intensive computing, large-scale dynamic workflows, and smart scheduling. Considering the scale and costs of next-generation leadership class systems, this level of codesign should have the force of an architectural design imperative.

Issues of urgent relevance for production application use such as how to layer concurrency across MPI, GPUs, GPU subunits, etc. can become afterthoughts, with getting codes to build or run with evolving vendor tools taking up all the time. Eventually, the influence on hardware is in any case very limited—in part because vendors are aiming to deploy a new software stack and new hardware concurrently. The scope of codesign in procurements is also extremely limited: One option is picked from a few vendors, by which time most design parameters are already fixed by the vendor to fit their hardware roadmap targeting mainstream customers. Few if any parameter/feature changes can be made in the 2-3 year timeframe from award to

*jdoerfert@anl.gov
†habib@anl.gov
‡knightc@anl.gov
§sparker@anl.gov
¶venkat@anl.gov
‖tjwilliams@anl.gov

delivery that fit within the vendor's target market, and the vendor is ultimately the one making the final decisions.

3 Opportunity

The challenges described above could be addressed by defining a community/open-source development environment (compilers, libraries, and data science frameworks) which will be evolved to deliver optimal performance on the new hardware. This environment should exist, fully-functional on existing hardware platforms today. For post-exascale system design, this could include products of the Exascale Computing Project [1]. A tool chain should allow customization, corrections, and interoperability in a reasonable way based on documented interfaces.

A modest set of complex-workflow applications targeting high performance at scale will be used to codesign the system. Compilers, libraries, and frameworks in that "stack" will be iteratively refined as hardware design evolves. Full-system application test cases, with all levels of concurrency considered from the outset, will lead to a more holistic design rather than hotspot and microbenchmark optimization. Asserting this common development context will help:

- lead to increased diversity in hardware vendors with strong responses to RFPs, and more opportunity to pick best-fit systems for the needs of compute facility users
- address non-traditional scientific use cases such as processing experimental/observational data on demand through dynamic and preemptable scheduling, re-training or revising embedded AI models confronted with new data patterns, time-sharing of accelerator resources by CPUs, and dynamically growing/shrinking resources for an application

Vendors working with research and compute facility staff toward a next-generation system would commit to deep engagement on the set of complex-workflow applications, to develop system software supporting the development environment, and the at-scale runtime environment. Important considerations such as power constraints would manifest in hardware at the integration level—numbers of nodes and node architecture to achieve applications performance within those constraints. At the node architecture level, there might be minor hardware design changes to support, for example, timesharing of accelerators.

System procurement is not the place for traditional microscale-focused codesign projects. These should be more open-ended efforts begun long before product roadmaps solidify. When hardware is designed we need to think early on about programmability and exposure into a non-vendor-specific software stack. Here again, in the context of microscale-focused codesign, we argue that community-based development tools are the way to go for something to be used by DOE researchers. The hardware design process could also take advantage of using production or near-future-production DOE scientific computing applications as targets.

4 Timeliness

This would be well-timed with respect to procurement of the first post-exascale systems at DOE facilities. Additionally, the current explosion of specialized accelerators and node architectures (neuromorphic, wafer-scale, etc.) has led to a rapid growth of trying out modest new systems on scientific computing workloads. A common, community-based development environment is more important than ever for performance portability across this emerging new ecosystem. Smaller vendors could leverage DOE development of system-scale design to build large systems with lower software investment needs. New microscale hardware design, such as novel accelerators with potential for DOE computing, could benefit early in the process by focus on programmability using a community development environment and by optimizing around a well-chosen set of modern, forward-looking DOE computing workflows.

References

[1] Kothe, D., Lee, S., and Qualters, I. Exascale computing in the united states. *Computing in Science Engineering 21*, 1 (2019), 17–29.

Title: Toward Machine-Actionable and Reusable Codesign Decisions

Authors: Matthew Wolf (wolfmd@ornl.gov), Jong Choi, Kevin Huck, Greg Eisenhauer, Jeremy Logan, Kshitij Mehta

Topic: Codesign Methodologies, Programming Systems

Challenge: The explosion of the architectural design space that comes from the many newly available technologies (chiplets, photonics, etc.) opens up more options for the scientific computing community, and existing efforts have shown that applying deep application knowledge to drive choices can lead to important new design points. However, approaching codesign as just the interaction between an end application and the heterogeneous hardware leaves out the crucial systems and middleware components from the HPC and scientific computing communities. Adding these components, libraries, and workflows into the codesign process is key if we are to get beyond individual, small-community, hero-effort codesign efforts. Further, the emergence of edge computing and experimental pipelines that span codesign workflows across a variety of systems and architectures has added even more heterogeneity to the parametric exploration of the codesign space. The scientific HPC community brings together a wide array of people and specialties, and the codesign of the tools and runtimes that bind all of that together needs to be a prime consideration.

Opportunity: The key opportunity in redefining how codesign works with the scientific HPC community is in reorienting from a one-time activity to an ecosystem that supports persistent, long-term investments in measuring, modeling, and portably applying fresh insights. The practice of codesign coming from our previous experiences [1,2] has led us to the observation that the output of a codesign study should not be a preferred solution for a current problem. Instead, we contend that the point of the codesign study is to determine a decision tree or, more generally, a **decision space** with prioritized directions for optimization when solving a problem, once the particular hardware, power, and/or performance constraints have been determined for a particular effort. Adopting this change in approach offers a number of benefits: (1) it highlights the need for tools that can generate and manage the decision space and the data upon which it is built; (2) it gives an abstract representation of the performance trade-off choices that separates out the different time scales on which performance decisions might need to be taken, and (3) it offers a chance for constructing software systems where hardware/software codesign can merge smoothly into models for auto-tuning of portable and reusable codes.

The tools for constructing and conducting the codesign experiments need to have traits of workflow and performance monitoring systems while delving further into supporting the codesign experimental platform. To make the analysis, data, and constituent components of the codesign reusable, a common abstraction that supports an easy and structured approach is important. Codesign-focused workflow and monitoring software can help users experiment across different layers in the hardware and software stack such as heterogeneous compute endpoints (CPUs, GPUs, FPGAs), tiered storage hierarchies (SSDs, NVMs, PFS), communication and I/O middleware, as well as application patterns. The ability to extract and manage live and rich performance information dynamically will enable evaluating multi-step science workflows as opposed to a single set of targeted benchmarks. As an example, consider the ECP Whole Device Model Application (WDMApp) project where multiple concurrent, heterogeneous simulations (some run only on CPUs, some run on GPUs/FPGAs, some need tightly connected networks, etc.) create a complex decision space to be explored [2]. The codesign of code-coupling requires clear experimentation, prediction, evaluation, and refinement stages for each exascale context. As a start to supporting this sort of codesign within ECP's Center for Online Data Analysis and Reduction [1], we developed the Cheetah and Savanna suite of tools that demonstrated to us the power of using a regularized experimental environment to guide codesign hypothesis testing and decision exploration. Thus a standardized method of performing codesign studies through new workflow systems will play an important role in accomplishing the end goals for codesign studies.

This new perspective on codesign enables but also requires dynamic configuration of both simulation and infrastructure resources, including libraries and dependent technology. Telemetry data from HPC systems is broadly available and can be dynamically captured, aggregated, modeled, and fed as input into feedback & control frameworks that use traditional or modern machine learning methods to guide algorithmic choices. Consider the challenge of tuning workflows that consist of multiple applications connected via streaming data links, such as federated instruments or multi-application code coupling. For example, [3] found that RDMA-based communication of simulation output data to an analysis application had an outsized negative impact on simulation performance because it interfered with the simulation's use of MPI. However, utilizing application-level monitoring to discern the simulation's "communication phases" vs. "compute phases" and scheduling output RDMA use appropriately eliminated this interference and maximized the performance of the linked applications. Detailed performance understanding is necessary both to diagnose such interactions and also to provide the infrastructure upon which automated amelioration solutions could be constructed.

Traditional codesign provides a human-actionable plan for adapting and tuning software to run on targeted hardware platforms. The target of a codesign decision space should be adjusted to be machine-actionable and dynamically reusable throughout the lifespan of the application. Such a reimagined codesign drives a shared decision space where architecture decisions, application performance, and auto-tuning middleware can all benefit. Treating it as a process to generate a static roadmap for systems or architectural engineers leaves out too many possibilities. As we have described, the monitoring and capture of runtime performance provenance information from a system is essential for developing rich models. In this case, a machine-actionable codesign provides an easy-to-deploy workflow that can be executed on a different target platform to not only measure the performance characteristics of that platform, but also to interpret the results and provide adaptation of codesigned software to run efficiently. Codesigning the interaction between heterogeneous hardware, allocation systems, workflow management, simulations, infrastructure libraries and the runtime analysis is key to enabling the type of reconfigurable computing that will fully utilize costly HPC resources.

Timeliness: Recent innovations due to exascale efforts, along with the overall degree of maturity in HPC software development, leave us ready to translate these gains to a broader set of interests. Individual hero efforts have been foundational, but we need scalable approaches that encourage reuse and sharing.

References:
1. I. Foster et al. "Online data analysis and reduction: An important co-design motif for extreme-scale computers". *International Journal of High-Performance Computing Applications*, in press, 2021
2. M. Wolf et al., "Scalable Performance Awareness for In Situ Scientific Applications," *2019 15th International Conference on eScience (eScience),* San Diego, CA, USA, 2019, pp. 266-276, doi: 10.1109/eScience.2019.00037.
3. H. Abbasi et al. "DataStager: scalable data staging services for petascale applications". In *Proceedings of the 18th ACM international symposium on High performance distributed computing* (*HPDC '09*). 2009. doi:10.1145/1551609.1551618

Co-design for Whole System Architectures

Nicholas J. Wright, Glenn K. Lockwood
NERSC
Lawrence Berkeley National Laboratory
{njwright,glock}@lbl.gov

Scott Atchley, Christopher Zimmer
OLCF
Oak Ridge National Laboratory
{atchleyes,zimmercj}@ornl.gov

Matthew L. Leininger
Livermore Computing
Lawrence Livermore National Laboratory
leininger4@llnl.gov

Topic Architectures, modeling and simulation, codesign methodologies

Challenge To-date co-design approaches within the DOE have focused primarily upon the processing element (e.g., CPU and GPU) using mini-apps and proxy applications. There has been some focus on network co-design using traces, modelling, SST and similar methods [1]. To fulfill its science and national security missions, the DOE needs to procure HPC systems and continued increases in effectiveness will require the exploration of co-design at the system, rather than only the component, level. Previous approaches have not adequately explored this aspect which will become more important as we enter the era of extreme heterogeneity and a slowing of Moore's Law. There are several challenges with current approaches:

- **Under-represented Application Space** - During various *Forward programs, selected vendor partners used on the order of ten proxy or mini apps, but ASCR facilities host hundreds of applications each year. Vendors limit the number of applications because they are focused on very low-level architectural details (e.g., cache replacement policies, memory access optimizations) that require very expensive cycle-accurate simulations. A related issue for the ASCR facilities is application turnover. While the NNSA facilities have more fixed and well-defined application and user base, the ASCR labs use annual public allocation programs that can see up to 30% turnover in applications from year to year. Even if the facility profiles applications in a given year, that information is perishable and declines in value over time.
- **Poor Representation of Full Applications** - Proxy apps and mini apps are not full applications. Proxies and mini apps tend to focus on a specific area of interest for the developer but, by design, leave out many aspects of full applications (e.g., serial sections, data movement between CPU and accelerator, non-representative data, I/O). It is challenging to understand the limitation of proxy and mini apps. Often a vendor partner may not understand or factor in these limitations, which can lead to hardware architectural choices based on invalid, unrepresentative, or incomplete code representations.
- **Applications in Isolation** - Most *Forward co-design efforts have focused on a single application on a single processor (or perhaps a single CPU and accelerator pair). Few jobs, even at the Leadership Computing Facilities, run in isolation. Assessing the impacts of co-scheduled jobs on current and proposed architectures is a challenge. DesignForward-2 had the goal to encourage vendors to consider whole system design and simulation although its budget was limited and outcomes from this program were varied. Also, the vendor-developed tools remained with the vendors and are not generally available for DOE use.
- **Lack of I/O** - While I/O can be included in the previous item, we call it out explicitly because most proxy and mini apps do not perform any I/O except perhaps to read input files. One job's I/O has been shown to negatively impact another job's performance [2]. Vendors may try to mitigate this by provisioning a separate, dedicated I/O fabric which is an expensive alternative or by providing Quality-of-Service (QoS) implemented using traffic classes. Being able to assess a proposed design's ability to adequately handle I/O is a challenge, and exploitation of co-design opportunities involving I/O is currently not possible.
- **Limited Market Scope** - A challenge in the co-design of HPC resources for DOE Supercomputers is matching the wider market need. Vendors are hesitant to work on options at the sole benefit to HPC. Identifying robust architectural solutions that benefit both HPC and the wider technology market is necessary for continued success.

In combination, these challenges outline the key differences as compared to co-design for embedded systems - the constantly evolving workloads and the multiple simultaneous users of an HPC system.

Opportunity The era of extreme heterogeneity and an increased focus upon considering application workflows provide opportunities for new directions of research that both address the aforementioned challenges as well as incorporate new trends. The target should be optimal performance of a whole HPC system; delivering a robust set of performance increases that will be perceived by end users. This whole system, holistic approach to co-design may also benefit a broader base of applications by offering optimizations that target common data paths between heterogeneous components such as NIC-to-accelerator or accelerator-to-storage. Such considerations will also provide a set of design goals for system software research, allowing identification of needed new capabilities in the future.

- **System Resource Usage Data** - A necessary step in to enabling this vision is to research how to collect resource usage for existing systems, and how to interpret it. At a minimum, facilities can use the data to assess whether the measured resources are over- or under-provisioned as well as for hotspot determination to help guide future procurements. Existing telemetry tools, such as LDMS [3] and Lustre Jobstats [4], are very useful but provide a limited view of the full application and full system behavior. Combining multiple streams of telemetry from inside (e.g., scheduler logs, resource usage) as well as outside (e.g., system power/cooling, outside ambient temperature) the system, researchers should be able to gain insights into the behavior of the system as a whole as it executes various workloads. Increased collection of telemetry will enable using AI/ML to discover new correlations (e.g., power/cooling trends for various projects/applications, optimized job scheduling in order to minimize inter-job interference or to smooth power usage, improve job throughput).

- **More Informative Application Proxies** - Although many mini-apps have been developed, there has been no systematic study of their deficiencies and how these may be addressed. Can application developers, in partnership with computer science researchers, provide more information to vendors about the valid uses of the proxy or mini-app and, more importantly, what uses are invalid (e.g., looking at data movement between a processor and accelerator when the proxy is only meant to focus on a specific kernel on an accelerator, whether the data is representative or not)? Relatedly, can application proxies be developed to addresses the gaps in coverage identified? Can the needs of DOE and HPC applications that are unique as compared to the overall IT industry be documented and quantified?
- **Continuous, Lightweight Application Characterization** - To complement the facility data collection above and to go beyond what mini-apps and proxies can provide, facilities need the capability to continuously characterize full applications running on their systems. These tools need to be lightweight to avoid slowing the applications down. Darshan [5] and AutoPerf [6] provide good examples of the desired performance but are of limited scope, I/O and MPI. Traditional profiling is useful for understanding application behavior on existing hardware, but lacks the ability to project performance on new hardware. The tools need to be able to capture the compute phases as well as data movement between components within a process and between processes. To minimize storage space for profiles, an ideal tool would recursively extract patterns that represent the application's use of resources (e.g., compute, memory, data movement).
- **Standardized System Model Format** - To assist facilities as they evaluate proposed architectures, the need exists for a standardized, machine-readable format to describe a proposed system's architecture including compute elements (e.g., chiplets within a processor, processors within a node), memory, interconnects, and storage with their associated performance characteristics. It should be have the granularity to describe individual processors, how processors are configured within a node, and how nodes are interconnected at the system level, and how systems are interconnected (e.g., compute and storage). It should be flexible enough to describe current systems and also future designs that could include dis-aggregated resources connected by photonics, compute in memory, and compute in the network.
- **Full System Modeling** - With the above full application representations and standardized system models, the final piece of the puzzle are tools to *play* the workload on the system model to determine a facility-defined Figure of Merit (FOM) (e.g., 50x faster, Volume of work, shortest time to complete the workload). Early work towards this has been promising if labor-intensive due to the nonstandard approaches to system profiling and data exchange between facilities, researchers, and vendors. Such tools might employ AI/ML to find the optimal balance for a given facility tailored to its specific workload. Much remains to be done in order to be able to determine the optimal balance of resources for a given facility's workload.

Timeliness or Maturity Given the rising costs of pre-exascale and exascale systems and the large step functions for provisioning key resources (e.g., memory, compute, interconnect, and storage), understanding the actual needs of applications rather than relying on received rules-of-thumb (e.g., 2 GBs of memory per core, checkpointing 50% of memory) will allow DOE facilities to provision the optimal combination of resources to provide the most effective system for their users. To this end, we need to co-design the system, not just the individual components. The research topics described above provide a pathway towards that goal. Gathering as much telemetry as possible about pre-exascale and exascale systems as they are deployed will allow computer scientists to make insights into resource usage quickly. More and improved proxy and mini-apps will allow vendors to have more insights into DOE's workload. Further out, tools to capture application motifs, standardized machine representations, and the ability to combine the two could enable DOE facilities to continue to improve the quality and quantity of science in an era of extreme heterogeneity.

References

[1] M. Mubarak, P. Carns, J. Jenkins, J. K. Li, N. Jain, S. Snyder, and R. Ross, "Quantifying I / O and Communication Traffic Interference on Dragonfly Networks Equipped with Burst Buffers," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, 2017, pp. 204–215.

[2] G. K. Lockwood, S. Snyder, T. Wang, S. Byna, P. Carns, and N. J. Wright, "A Year in the Life of a Parallel File System," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE, nov 2018, pp. 931–943. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291755 https://ieeexplore.ieee.org/document/8665806/

[3] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, nov 2014, pp. 154–165. [Online]. Available: http://ieeexplore.ieee.org/document/7013000/

[4] "Lustre Jobstat," https://doc.lustre.org/lustre_manual.xhtml#dbdoclet.jobstats, [Online; accessed 13-February-2021].

[5] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, "Modular HPC I/O characterization with Darshan," *Proceedings of the 5th Workshop on Extreme-Scale Programming Tools (ESPT '16)*, pp. 9–17, 2016.

[6] S. Chunduri, S. Parker, P. Balaji, K. Harms, and K. Kumaran, "Characterization of mpi usage on a production supercomputer," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2018, pp. 386–400.

End-to-End PowerStack Codesign for Energy Efficient HPC

Xingfu Wu
Argonne National Laboratory
The University of Chicago, USA
Email: xingfu.wu@anl.gov

Aniruddha Marathe
Lawrence Livermore National Laboratory
Livermore, CA, USA
Email: marathe1@llnl.gov

Siddhartha Jana
Intel Corporation
Hillsboro, OR, USA
Email: siddhartha.jana@intel.com

I. Topic: Software Codesign Methodologies

Efficiently utilizing procured power and optimizing performance of scientific applications under power and energy constraints are important challenges in HPC. The HPC PowerStack - a global consortium of laboratories, vendors, and universities - has highlighted a design shift towards standardization of the HPC power-management software stack. This enables seamless integration of software codesign solutions that enable management of energy/power consumption of large scale HPC systems. This position paper presents the findings of a working group focused on the end-to-end tuning of this power management codesign stack. We identify the research opportunities and challenges for collective auto-tuning of two or more management layers (or domains) in the PowerStack. This paper lays the foundation for this initiative by identifying and aggregating the important R&D challenges in streamlining optimization efforts across the multiple layers, contributors and stakeholders of the PowerStack for aiding in energy efficient HPC codesign.

II. Challenges

As we enter the Exascale computing era, power and energy management are key design points and constraints for any next generation of supercomputers [1]. Efficiently utilizing procured power and optimizing the performance of scientific applications under power and energy constraints are challenging for several reasons including dynamic phase behavior, manufacturing variation, and increasing system-level heterogeneity. While several individual techniques have been proposed for the automatic and efficient management of power and energy, the majority of these techniques have been devised to meet the needs of a specific high-performance computing (HPC) center or specific optimization goals. Furthermore, each technique tends to improve the management of power and energy for a different subset of the site or system hardware and at different (and often conflicting) granularities. Unfortunately, the existing techniques have not been designed to coexist simultaneously on one site and cooperatively manage resources in a streamlined fashion.

To address these gaps, the HPC community needs a holistic stack for power and energy management. The HPC PowerStack Initiative [1], [2] started in May 2018 as a working group to gather the experience of active developers in industry, computing centers, and academia for building software interfaces and solutions for handling and optimizing the power and energy consumption in production HPC systems. Based on the state of the art of the components available in the community for power and energy management a hierarchical strawman PowerStack design [1] was proposed to manage power and energy at three levels of granularity: the system level, the job level, and the node level. This implies the need to put in place the following incrementally:

- Define policies that govern site-level requirements, a power-aware system Resource Manager (RM) / job scheduler, a power-aware job-level manager, and a power-aware node manager.
- Define the interfaces between these layers to translate objectives at each layer into actionable items at the adjacent lower layer.
- Drive end-to-end optimizations across different layers of the PowerStack.

III. Research Opportunity

To address these requirements above, we formed a PowerStack End-to-End Auto-tuning Working Group in 2019. A plethora of literature on power-aware tuning exists, including notable works by the members of this working group. A primary limitation of most—if not all—of these efforts is that the tuning research has been solely limited to the individual layers of the PowerStack.

Our recent work [3] (a) surveyed the high-level objectives of the existing layer-specific tuning approaches at the different layers: system (i.e., cluster), job / application, and node, (b) defined the tunable parameters at each layer, and (c) proposed and discussed how to autotune the combination of different parameters at the distinct layers (parameter space) for an optimal solution (the smallest runtime, the lowest power, or the lowest energy) under a system power cap as shown in Figure 1. This diagram shows the interactions among four layers: system-level, job-level, node-level, and application-level. However, to the best of our knowledge, it still lacks an end-to-end autotuning component to target all four layers for the optimal solution. Specifically, for DoE HPC platforms and simulations, each system node consists of not only CPUs and GPUs but also FPGAs and AI accelerators; each simulation involves not only computation and communication but also

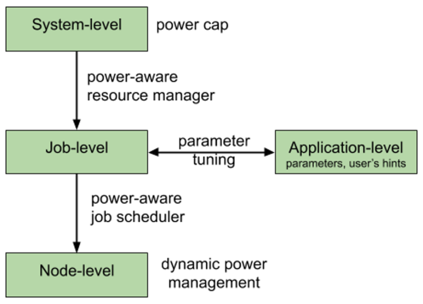


Fig. 1. The PowerStack System Autotuning Framework

surrogate model training, evaluation, and prediction. Thus, achieving optimal system-level target metric while satisfying component-specific goals becomes challenging. In the rest of this paper, we identify the important unsolved challenges in collective autotuning of two or more management layers (or domains) in the PowerStack.

Research Question: How to explore a holistic performance, power and energy management software codesign stack that is capable of optimizing the target power- or energy-efficiency application-aware metric so that it can trade-off power, energy, and time to solution in order to optimize the end-to-end efficiency of HPC codesigns?

IV. Timeliness

As the complexity of heterogeneous HPC ecosystems (hardware stacks, software stacks, applications) continues to rise, achieving optimal performance becomes a challenge. The number of tunable parameters the user of each layer can configure has increased, resulting in the overall parameter space growing significantly. Exhaustively evaluating all parameter combinations becomes very time-consuming, and consequently, impractical. Therefore, automatic exploration of the parameter space is desirable. In our recent work [4], we developed an autotuning framework that leverages Bayesian optimization with four supervised machine learning methods to explore the parameter space search and used the autotuning framework to optimize the loop pragma parameters to improve the application performance.

Based on our experience of autotuning [3], [4] performance, power, and energy of applications and systems, we propose an End-to-end PowerStack Autotuning Framework for DoE HPC platforms, supplemented by a prototype of the framework with dynamic power management, and demonstrate at-scale impact on controlling trade-offs between system performance and power trade-offs. For the proposed end-to-end PowerStack autotuning framework, we will integrate the existing power-aware resource scheduler SLURM at system level, job constraint-aware power/energy optimizer GEOPM [5] at job and node levels, and the application autotuning framework at application level to develop a prototype to tune all four layers of PowerStack so that we can have better understanding of the tunable parameters at each layer and interaction interfaces between layers and potential new requirements in order to achieve energy efficient codesign goal. The process of co-tuning in the layers (a) typically targets performance or power efficiency as the primary metric, (b) complies with the operating power constraint imposed on the layer, and (c) attempts to improve the management and orchestration of the available control parameters that affect the application and/or hardware performance.

The end goal of this tuning software codesign space is to enable HPC sites under power constraints to leverage feedback-driven interoperability between system resource managers, runtime systems, and applications to maximize system performance and energy efficiency. Interfaces are required to facilitate the interaction across three layers. The interfaces defined must leverage codesign principles as follows:

1) They must enable translation of system-level targets from the resource manager into job-level targets at the granularity of runtime systems and the applications.
2) They must enable end-to-end flow of telemetry data at multiple levels of granularity.
3) They must enable interacting system components to validate and verify the reliability and robustness of the flow of signals and controls across the stack

A classic beneficiary of the above efforts will be that of a site driven by system throughput and power constraints. A solution adopted by this site will require resource managers to translate throughput-based metrics into optimization policies which, in turn, would translate to 'job-level power budgets' or 'total allocated runtime'. These metrics in turn would be leveraged by application-aware runtimes to boost calculations per time-step per watt. Such end-to-end solutions are extremely critical for resource-constrained sites. The time is right to invest in such opportunities to boost gains in power efficiency from tapping into interoperability between multiple layers of the HPC PowerStack.

References

[1] C. Cantalupo, J. Eastep, S. Jana, M. Kondo, M. Maiterth, A. Marathe, T. Patki, B. Rountree, R. Sakamoto, M. Schulz, and C. Trinitis, "A strawman for an hpc powerstack," 8 2018. [Online]. Available: https://www.osti.gov/biblio/1466153

[2] A. Bartolini, S. Brink, D. Cesarini, D. Ellsworth, R. Grant, S. Jana, M. Kondo, E. K. Lee, M. Maiterth, A. Marathe, T. Patki, S. Perarnau, V. Reis, M. Schulz, O. Vysocky, T. Wilde, and X. Wu, "White paper on powerstack." [Online]. Available: https://hpcpowerstack.github.io/raitenhaslach20.html

[3] X. Wu, A. Marathe, S. Jana, O. Vysocky, J. John, A. Bartolini, L. Riha, M. Gerndt, V. Taylor, and S. Bhalachandra, "Toward an end-to-end auto-tuning framework in HPC PowerStack," in *Proceedings of Energy Efficient HPC State of Practice 2020 (EE HPC SOP 20)*. Washington, DC, USA: IEEE Computer Society, 2020.

[4] X. Wu, M. Kruse, P. Balaprakash, H. Finkel, P. Hovland, V. Taylor, and M. Hall, "Autotuning PolyBench Benchmarks with LLVM Clang/Polly loop optimization pragmas using Bayesian optimization," in *Proceedings of SC20 Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, ser. PMBS'20. Washington, DC, USA: IEEE Computer Society, 2020.

[5] J. Eastep, S. Sylvester, C. Cantalupo, B. Geltz, F. Ardanaz, A. Al-Rawi, K. Livingston, F. Keceli, M. Maiterth, and S. Jana, "Global extensible open power manager: a vehicle for HPC community collaboration on co-designed energy management solutions," in *in International Supercomputing Conference: High Performance Computing (ISC 2017)*, ser. LNCS, vol. 10266, 2017.

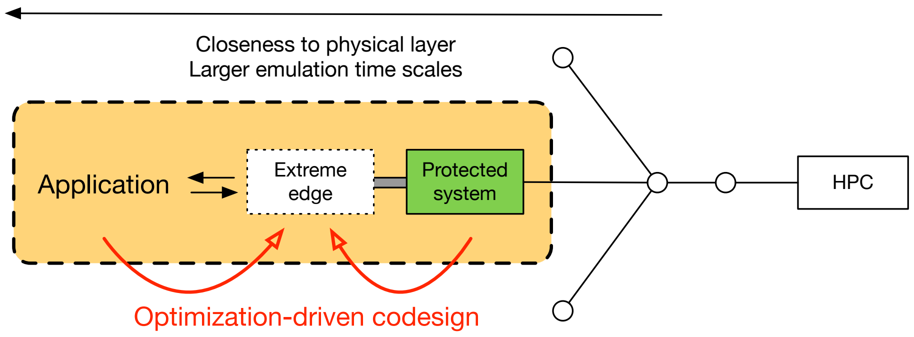
CODESIGN APPROACHES TO ENABLE AND OPTIMIZE COMPUTING AT THE EXTREME EDGE

ANGEL YANGUAS-GIL*, SANDEEP MADIREDDY, PRASANNA BALAPRAKASH

Argonne National Laboratory

Topics: Applications, modeling and simulation, emerging technologies, codesign methodologies

Challenge: Applications at the extreme edge are subject to power, bandwidth, and environmental constraints, in some cases requiring the design from scratch of a hardware layer capable of operating in high temperature and high radiation environments. These conditions are crucial for the DOE mission, both in terms of enhancing our ability to carry out new science and to enable monitoring and sensing over the energy lifecycle.


Closeness to physical layer
Larger emulation time scales
Application
Extreme edge
Protected system
HPC
Optimization-driven codesign

The design and optimization of these architectures often requires integrating multiple levels of abstraction, spanning from high performance computing for data processing, wireless communication, and system level operation, down to HDL and circuit level emulation for edge architectures. It also requires integrating vastly different timescales: for instance, in order to properly characterize power consumption and bandwidth we need to model edge architectures over timescales and tasks relevant for the target applications, which may involve many clock cycles.

This leads to the following **two challenges**:

(1) How can we accelerate the optimization of novel architectures in a way that integrates all the steps from application down to novel devices and incorporates system-level constraints?
(2) Can we emulate architectures at the HDL and circuit levels as they interact with complex, scientifically relevant environments at speeds faster than real time?

Opportunity: Overcoming existing gaps in hardware design tools and workflows to enable codesign of novel architectures could be accomplished by focusing on three different thrusts:

(1) The development of rich, interacting environments (digital twins) that can run natively in leadership computing facilities.
(2) The exploration of novel ways of accelerating the emulation of circuits to benefit from massive parallelism beyond the capabilities of existing tools.

E-mail address: `ayg@anl.gov`.

(3) The integration of these novel tools with optimization frameworks that can focus on architecture exploration, from swapping modules and components down to the optimization of specific devices in the case of analog architectures.

We have started to explore such codesign approaches to develop architectures capable of computing at temperatures exceeding 300 deg C, well above the limit of existing hardware, as proof of concept for an LDRD-funded research project on hardware optimization. These temperatures are beyond the limits of current Si and SiGe SOI CMOS technology, and would require integrating JFET-based nmos logic on wide bandgap semiconductors with novel refractory materials.[1, 2]

Our approach focuses on two key ideas: first, we leverage existing machine learning frameworks to emulate the behavior of digital and analog circuits, using stochastic gradient descent methods to optimize our architectures to carry out inference and reinforcement learning tasks. This is an approach that we have already demonstrated in the context of spiking neural networks.[3] Second, we integrate our architecture with black-box optimization tools to efficiently search over the design space. In particular, we use DeepHyper [4], a scalable mixed-integer nonlinear optimization package that is built to take advantage of leadership class computing systems through parallel algorithms and efficient workflow management systems. The design space includes parameters intrinsic to the architecture itself, parameters that relate to underlying devices, and parameters, such as the use of synchronous or asynchronous implementations, that relate to how the architecture interacts with the rest of the system. Since the model is grounded in the physical layer, we can also extract information on power consumption and the impact of interfaces in the overall performance.

This approach to codesign can be generalized well beyond this particular example: by designing modular architectures with components that can be swapped while preserving the overall functionality, we can explore complex design spaces spanning from application to basic building blocks in a massively parallel, asynchronous way. This ability is limited solely by our ability to bring efficient hardware emulation into leadership computing machines. This approach complements well existing digital design tools, enabling the exploration of potentially thousands of architectures and downselecting the most promising candidates for their implementation and verification using existing design workflows.

Timeliness: Our approach to codesign is enabled by recent advanced in the area of machine learning, including both artificial neural networks and optimization. The implementation of efficient emulation tools in leadership computing facilities is also well aligned with DOE's traditional emphasis on scientific computing, and could broaden the scope of architectures beyond AI accelerators and neuromorphic computing approaches to other critical areas such as advanced wireless and quantum computing. The ability to bridge from applications back to novel materials would also benefit other DOE initiatives such as microelectronics, providing a top-down approach that complements research that is largely focused on emergent materials.

Acknowledgments: This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.

References

[1] P. G. Neudeck et al, Prolonged silicon carbide integrated circuit operation in Venus surface atmospheric conditions, AIP Advances 6, 125119 (2016).

[2] S. Babar et al, W:Al2O3 Nanocomposite Thin Films with Tunable Optical Properties Prepared by Atomic Layer Deposition, J. Phys. Chem. C 120, 14681 (2016).

[3] A. Yanguas-Gil, Coarse scale representation of spiking neural networks: backpropagation through spikes and application to neuromorphic hardware, International Conference on Neuromorphic Systems 2020, https://doi.org/10.1145/3407197.3407221

[4] P. Balaprakash, et al. DeepHyper: Asynchronous Hyperparameter Search for Deep Neural Networks. In 25th IEEE International Conference on High Performance Computing, Data, and Analytics (2018).

[5] S. Madireddy, A. Yanguas-Gil, P. Balaprakash, Multilayer Neuromodulated Architectures for Memory-Constrained Online Continual Learning, arXiv:2007.08159 (2020).

Co-design Streaming Processing Hardware using Chisel Hardware Construction Language

Kazutomo Yoshii
Mathematics and Computer Science
Argonne National Laboratory
email: kazutomo@mcs.anl.gov

I. Introduction

Integrating seamlessly between large scientific instrumentation such as X-ray light source and HPC systems is a crucial next step to accelerate scientific discoveries towards future scientific laboratories. We are at an early stage of such integration. As both temporal and spatial resolution keep increasing, a massive amount of data will be generated. As an example, the frame rate of pixel array detector chips for X-ray light sources will approach MHz soon, which will eventually generate a terabit of raw data per chip. Yet large percentage of data can be outside of region of interests. Connecting between the edge of scientific facility (e.g., X-ray detector chip) and HPC systems with high-end networks could only solve the integration problem partially and is likely to be impractical in terms of the economical stand point of view. The size of data needs to be reduced by data compression, filtering or ultimately AI-based feature detection in a streaming manner, taking into account application or experiment requirements. We discuss a true hardware/software co-design process for streaming processing hardware, leveraging Chisel, a Scala-based modern hardware construction language.

Topics: architectures, simulation, codesign methodologies

II. Challenges

True co-design requires both software and hardware development. However, genuine hardware experts are minority in DOE community. Even with more hardware experts, developing hardware using hardware description language (HDL) such as Verilog, VHDL is still a daunting task due to lack of flexible parameter systems and modern software constructs. In HDL, recursive structure needs to be unrolled and attributes are hard-coded (e.g., names and parameters for unrolled instances), which significantly affects reusability. On the other hands, emerging high-level synthesis (HLS) tools [5], which allows developers to express hardware in a familiar language (e.g., a subset of C/C++ language), can potentially improve productivity and reusability. It still requires hardware expertise to optimize HLS codes. Unfortunately we have little control over generated circuits such as timing, resource usage in order to meet requirements.

III. Opportunities

As the transistor scaling is coming to halt, custom hardware development is a new trend in large companies such as Apple, Google, Microsoft, even previous known as a software company. This new custom hardware trend is becoming a good tailwind to hardware ecosystem, including open-source instruction-set such as RISC-V [2], open-source hardware implementations [4, 7] and open-source hardware tools [11, 3] In terms of hardware platform, field-programmable gate array (FPGA) is readily available these days. HLS further lower the hurdle to FPGA. Even the barrier to entering ASIC development is lowering. Google announced that an open-source foundry PDK recently [1]. The advent of innovative semiconductor companies like efabless.com help a small group to tape-out a chip with significantly lower cost, benefiting from a fully open-source end-to-end ASIC design flow [6].

What caught our attention was a new class of hardware description language called hardware construction language (HCL), which addresses productivity challenges while offering control of generated digital circuits. Chisel [3] is one of the emerging HCLs and offers higher expressivity that dramatically improves the productivity of the circuit design process. In fact, Chisel is used for many real-world tapeout designs (e.g., RISC-V processors, Google's Edge TPU) and many open-source academic hardware projects. Technically Chisel is a class library written in the Scala functional programming language [9], instead of a standalone programming language. In Chisel, developers write a hardware circuit generator in Scala using hardware construction primitives provided by Chisel class library so that they can leverage the power of modern programming language. Synthesizable Verilog codes are generated by executing Chisel codes.

One of the interesting aspect of Chisel is that it encourages test-driven development and offers fully integrated testing harnesses so that users can write test-bench codes in Scala. Running simulators to test user designs is straightforward in Chisel and requires no high-end machines. Since Scala is a powerful modern programming language, software/hardware co-design can be done through test-bench development. Other important point is that the quality of circuit design is proportional to the number of iterations in the design loop (coding, building, evaluating) in many cases.

At Argonne, we have been using Chisel for hardware design exploration on data compressor circuits for X-ray detector ASIC [8]. With a thousand input signals and large reduction components, the number of lines in Verilog codes of the compressor design can be an order of ten thousands for the compressor circuit. In early design exploration stages, design parameters often change, which made maintaining Verilog implementations impractical. Chisel not only allowed us to explore various compressor designs, but also allowed us to perform RTL-simulation with actual X-ray input datasets without additional effort.

Near-detector real-time AI inference capability is expected to be a core technology for X-ray data analysis in future. We are currently seeking an opportunity that we can apply Chisel to generate light-weight AI inference FPGA firmware that can perform classification at FPGAs near the detector. We have previously studied FPGA-based AI acceleration [10] and demonstrated its real-time performance. New challenges are to optimize firmware in AI training level (e.g., lower precision) and generate FPGA firmware from AI model format such as ONNX, which can lead to automatic data-driven co-design.

IV. Conclusion

Since recent custom hardware development trend is becoming a good tailwind for hardware ecosystem, it is a good timing to ride on the trend. We have successfully evaluated Chisel for design exploration of X-ray detector ASIC's compressor circuits. From our experiences, we believe that Chisel or other HCLs can be a great research vehicle for software/hardware co-design for scientific instruments/HPC integration and other stream computing applications (bump-in-wire network, storage). Additionally, with Chisel's RISC-V root, our expertise can be smoothly translated into co-designing RISC-V accelerators.

References

[1] Tim Ansell and Mehdi Saligane. The missing pieces of open design enablement: A recent history of google efforts: lnvited paper. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2020.

[2] Krste Asanović and David A Patterson. Instruction sets should be free: The case for risc-v. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.

[3] J Bachrach, H Vo, B Richards, and Y Lee DAC an d 2012 Design. Chisel: constructing hardware in a scala embedded language. *DAC Design Automation Conference*, pages 1212–1221, 2012.

[4] Christopher Celio, Pi-Feng Chiu, Borivoje Nikolic, David A Patterson, and Krste Asanovic. Boomv2: an open-source out-of-order risc-v core. In *First Workshop on Computer Architecture Research with RISC-V (CARRV)*, 2017.

[5] Philippe Coussy and Adam Morawiec. *High-level synthesis*, volume 1. Springer, 2010.

[6] R Timothy Edwards, Mohamed Shalan, and Mohamed Kassem. Real silicon using open source eda. *IEEE Design & Test*, 2021.

[7] Farzad Fatollahi-Fard, David Donofrio, George Michelogiannakis, and John Shalf. Opensoc fabric: On-chip network generator: Using chisel to generate a parameterizable on-chip interconnect fabric. In *Proceedings of the 2014 International Workshop on Network on Chip Architectures*, pages 45–50, 2014.

[8] Mike Hammer, Kazutomo Yoshii, and Antonino Miceli. Strategies for on-chip digital data compression for x-ray pixel detectors. *Journal of Instrumentation*, 16(01):P01025–P01025, jan 2021.

[9] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in scala*. Artima Inc, 2008.

[10] Ahmed Sanaullah, Chen Yang, Yuri Alexeev, Kazutomo Yoshii, and Martin C Herbordt. Real-time data analysis for medical diagnosis using fpga-accelerated neural networks. *BMC bioinformatics*, 19(18):19–31, 2018.

[11] Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys-a free verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, 2013.

Acknowledgments

This work was partially funded by the DOE grant DE-SC0012531 and the NSF grant CCF-1422489. This material was based upon work supported by the U.S. Department of Energy Office of Science, under contract DE-AC02-06CH11357.

Emerging Heterogeneous Systems Provide Great Opportunities for Codesign

Aaron R. Young, Jeffrey S. Vetter, Frank Liu,
Narasinga Rao Miniskar, Sarat Sreepathi, and Anthony M. Cabrera
Oak Ridge National Laboratory, Oak Ridge, TN, USA
{youngar,vetter,liufy,miniskarnr,sarat,cabreraam}@ornl.gov

Topic: Architectures, Applications, Modeling and Simulation, Emerging Technologies

1 Challenge

As Moore's Law and Dennard scaling are coming to an end, simple technology scaling cannot be relied on for performance gain, and new technologies and computing paradigms must be developed to continue improving application performance. To this end, hardware accelerators like GPUs, TPUs, and FPGAs are being employed as co-processors to traditional systems to accelerate computation. Additionally, new ways of computing, for example, neuromorphic and quantum computing, are also showing promise and could be incorporated into large-scale HPC systems. Because different scientific domains and applications will benefit from different configurations of accelerator types and computing paradigms, we predict that HPC systems will become "extremely" heterogeneous [1]. Though such systems could significantly accelerate application performance, there are many resulting application, system, and hardware development challenges, including: 1) how to write software to target extremely heterogeneous systems, 2) how to port legacy software to new systems, 3) how the programming environment and runtime system should map the software to utilize the hardware accelerators efficiently, and 4) how the heterogeneous systems should be designed (i.e., which accelerators should be included and how they should be configured).

We believe that by codesigning the applications, runtime environment, and accelerators, it is possible to achieve high-performance without losing performance portability in future, extremely heterogeneous systems.

2 Opportunity

Interconnect The challenge of how to design these heterogeneous systems leads to many research opportunities. One such opportunity is architecture exploration of the interconnect between compute and memory components. CXL, an emerging and open industry standard processor interconnect, is designed to enable low-latency memory access and create coherent memory space between CPUs, accelerators, and memory pools. CXL is a promising interconnect to enable multiple accelerators and new emerging memory technologies to be incorporated within an extremely heterogeneous node.

Architecture Modeling The interconnect, applications, runtime, and accelerators all need to be designed together to meet performance and compatibility requirements. Architecture level models built using tools like GEM5 and SST will enable the joint exploration of all these components using simulation. Architecture level simulations and machine learning techniques can be used to perform design space exploration to determine the best architecture configuration [2].

Hardware Design Accelerators are more specialized than general compute cores. Some devices like FPGAs are re-configurable, while others perform more limited operations. In all cases, a joint design effort by hardware and software designers is required to develop software that can leverage the more specialized hardware and design hardware that is well suited to accelerate important software tasks.

Application Characterization Workload characterization of key target application(s) is pivotal to a successful co-design effort in informing design space exploration. An in-depth analysis of applications and associated proxy apps [3] would entail both static and dynamic analyses as well as gathering architecture-aware and agnostic metrics. The proliferation of high-bandwidth memory and complex memory hierarchies necessitate research into trade-offs for memory-bound applications.

Heterogeneous Programming How to best write maintainable code that can target a range of different accelerators is an ongoing challenge. However, current work is looking at the portability of popular accelerator languages such as OpenCL and C. Additionally, new frameworks like OneAPI, SYCL, and OpenARC [4] show promise for enabling heterogeneous compute; however, additional work is needed on both the language and hardware sides to ensure the performance portability of software written in these frameworks.

Runtime Another considerable challenge and opportunity is in designing runtime frameworks to map the work expressed by the application onto hardware for execution. MPI and OpenMP are widely used solutions for expressing concurrency for distributed applications, but these methods will fall short when targeting large heterogeneous systems. More complex heterogeneous workflows could be expressed using a task-based or dataflow-based programming model. Work expressed in these models could then be automatically parallelized and distributed to the appropriate compute nodes and accelerators by a runtime system. This system could leverage multiple tuned application kernels that were either provided or generated from a higher-level language by a compiler. The runtime could also use architecture-independent workload characteristics [5] and workload-independent hardware characteristics, along with an AI-assisted scheduling policy to assign tasks and map the work across the heterogeneous HPC system.

3 Timeliness

The end of simple compute scaling, the rise of open hardware, and the increase in new accelerators are leading to increasingly heterogeneous architectures. New developments in heterogeneous languages and runtimes are enabling the software support to leverage this heterogeneous hardware. By codesigning the applications, programming languages, runtimes, system architectures, and accelerator hardware, future heterogeneous HPC systems are designed to accelerate workloads beyond what is currently possible. If we work to codesign these systems now, we can overcome these challenges and create maintainable, performance portable code.

References

[1] Jeffrey S Vetter, Ron Brightwell, Maya Gokhale, et al. *Extreme heterogeneity 2018-productive computational science in the era of extreme heterogeneity: Report for DOE ASCR workshop on extreme heterogeneity.* Tech. rep. USDOE Office of Science (SC), Washington, DC (United States), 2018.

[2] Frank Liu, Narasinga Rao Miniskar, Dwaipayan Chakraborty, et al. "Deffe: A Data-Efficient Framework for Performance Characterization in Domain-Specific Computing". In: *Proceedings of the 17th ACM International Conference on Computing Frontiers.* CF '20. Catania, Sicily, Italy: Association for Computing Machinery, 2020, pp. 182–191. ISBN: 9781450379564. DOI: 10.1145/3387902.3392633.

[3] Sarat Sreepathi, M. L. Grodowitz, Robert Lim, et al. "Application Characterization Using Oxbow Toolkit and PADS Infrastructure". In: *Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing.* Co-HPC '14. New Orleans, Louisiana: IEEE Press, 2014, pp. 55–63. ISBN: 9781479975648. DOI: 10.1109/Co-HPC.2014.11. URL: https://doi.org/10.1109/Co-HPC.2014.11.

[4] S. Lee, D. Li, and J. S. Vetter. "Interactive Program Debugging and Optimization for Directive-Based, Efficient GPU Computing". In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium.* May 2014, pp. 481–490. DOI: 10.1109/IPDPS.2014.57.

[5] B. Johnston and J. Milthorpe. "AIWC: OpenCL-Based Architecture-Independent Workload Characterization". In: *2018 IEEE/ACM 5th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC).* Nov. 2018, pp. 81–91. DOI: 10.1109/LLVM-HPC.2018.8639381.

Containerization Should Influence New Hardware-assisted Mechanisms for Resource Multiplexing and Isolation

Andrew J. Younge[*1], Kevin Pedretti[*], Jack Lange[^], Ron Brightwell[*]
[*]Center for Computing Research, Sandia National Laboratories
[^]Computer Science Department, University of Pittsburgh

Topic: Programming Systems, Co-design Methodologies

Challenge: Container deployment models have taken the computing industry by storm, fundamentally changing how computational workloads are deployed on distributed systems and clouds. More recently, container-based scientific computing has also gained traction [1] within the HPC community, with several new runtimes and processes specially derived for supercomputers [2]. If this trend in usage continues and containers are integrated into modern DevOps solutions for scientific computing, then new opportunities arise for better integration between containerized workloads, the OS-level virtualization mechanisms, and managing hardware resources for any given supercomputing system. This need will be further driven by the increase in system and node level heterogeneity expected in the next decade, whereby no single workload will be able to concurrently use all available resources of a given system. As such, increasing supercomputing efficiency for extreme heterogeneous systems is likely to require on-node state sharing and hardware partitioning of containerized workloads, and hardware co-design is needed.

Predating the advent of containers, virtual machines and hypervisors handled the task of abstracting, emulating, and isolating the underlying hardware for deploying one or more operating systems. While this capability ushered in novel cloud paradigms with Infrastructure-as-a-Service, hypervisors also drove co-design in hardware where the goal was to simplify or offload hypervisor tasks directly to hardware whenever possible. Several examples include second-level address translation (often known by implementations such as Extended or Nested Page Tables from Intel and AMD, respectively), where CPU instructions were added to manage shadow page table entries directly in hardware. Other examples include device multiplexing through hardware capabilities such as SR-IOV, which allowed for PCI-Express devices to be safely shared or multiplexed in virtual environments. All of these advances made in hardware resulted in hypervisor performance improvements that allowed VMs to approach near-native performance, for both the hyperscaler market and production HPC workloads [3, 4].

However, containers currently do not utilize any hardware-assisted mechanisms beyond traditional memory isolation mechanisms for process separation & security. Instead, the OS is responsible for isolating the container process, and then acting as a loose arbiter of underlying resources through OS primitives provided by cgroups [5]. While this has proven successful for basic resource sharing, time-slicing mechanisms such as those found in the CPU cgroup interfaces are fundamentally not well suited for resource partitioning. As an anecdotal example, HPC users running containerized workloads constrained by cgroups have found up to 10x performance degradation due to ill-informed limits and over-scheduling of OpenMP tasks. While workloads can be manually adapted to perform better in cgroups-constrained environments, it nevertheless demonstrates that OS-level time-slicing mechanisms do not meet the needs of HPC.

[1] Corresponding Author

Opportunity: Instead, the HPC system software community requires a new effort to co-design hardware assistance for improved resource isolation, partitioning, and quality of service. This effort can be divided into three sub co-design areas: CPU partitioning, Memory isolation, and I/O multiplexing. For CPU partitioning, cores could be cordoned off from the rest of the OS (and other containers) to execute specific CPU-sensitive container processes. This method could build from CPU off-lining techniques pioneered by multi-kernels [6], with added features in hardware to speed transitions and reroute IRQ requests, for example. L2 & L3 cache partitioning strategies and the assignment of NUMA-aware memory regions could help container workloads increase memory bandwidth when needed. Furthermore, container runtimes could even negotiate NUMA affinity to directly map memory regions to accelerators and GPUS and to improve offload latencies, effectively bypassing the OS entirely. Containers will also benefit from hardware-based I/O multiplexing solutions like SR-IOV to assign networking interfaces and SmartNICs to specific containers. Such hardware-based I/O assistance could provide isolation, quality of service, and OS bypass solutions in hardware to insure optimal performance for networking and I/O tasks. Designs could offer more ubiquitous quality of service on every type of hardware resource, perhaps by hardware tagging networking data & memory regions, or offloaded kernels, coupled with a unified set of OS interfaces exposed to container runtimes. If hardware-assisted resource isolation and partitioning for containers can be coupled with process-based state-sharing properties in shared memory and IPC, then HPC workloads can be fundamentally cast as coordinated ensembles of containerized units, rather than just as another queue of batched jobs. Effectively, this will enable the DOE to increase efficiency by matching available hardware to workload ensembles rather than jobs, so extremely heterogenous architectures could achieve 2-4x utilization improvements compared to current course-grained node allocation strategies.

Timeliness & Impact: As containers are increasing in popularity and existing work in hardware co-design for VMs has reached relative maturity, now is the ideal time to invest in the co-design of hardware-accelerated containerization mechanisms. With more heterogenous hardware designs on the horizon, hardware-assisted partitioning and isolation mechanisms are needed more than ever to help increase system utilization across an entire HPC resource. Such capabilities could have a significant impact on how scientific computing workloads are developed, deployed, and orchestrated on the next generation of supercomputing systems.

References:

[1] Jacobsen, D. M., & Canon, R. S. Contain this, unleashing docker for hpc. *Proceedings of the Cray User Group*, 33-49, 2015.
[2] Pedretti, K., Younge, A. J., Hammond, S. D., Laros, J. H., Curry, M. L., Aguilar, M. J., ... & Brightwell, R., Chronicles of Astra: challenges and lessons from the first petascale arm supercomputer. In *SC20: Proc. Int. Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2020.
[3] Lange, J., Pedretti, K., Hudson, T., Dinda, P., Cui, Z., Xia, L., ... & Brightwell, R., Palacios and Kitten: New high performance operating systems for scalable virtualized and native supercomputing. In *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)* (pp. 1-12). IEEE, Apr 2010.
[4] Younge, A. J., Walters, J. P., Crago, S. P., & Fox, G. C., Supporting high performance molecular dynamics in virtualized clusters using iommu, sr-iov, and gpudirect. *ACM VEE, 50*(7), 31-38, 2015.
[5] Zhuang, Z., Tran, C., Weng, J., Ramachandra, H., & Sridharan, B., Taming memory related performance pitfalls in linux Cgroups. In *2017 ICNC,* (pp. 531-535). IEEE, Jan 2017.
[6] Wisniewski, R. W., Inglett, T., Keppel, P., Murty, R., & Riesen, R., mOS: An architecture for extreme-scale operating systems. In Proceedings of the 4th International Workshop on Runtime and Operating Systems for Supercomputers (pp. 1-8), Jun 2014.

Title:

CoDesign 2030 Agenda

Author: Victor Zhirnov, Chief Scientist, Semiconductor Research Corporation (SRC)4819 Emperor Blvd, Ste 300, Durham NC 27703, (919) 941-9454, Victor.Zhirnov@src.org

Topic: architectures, applications, emerging technologies, codesign methodologies

Challenge:

The current hardware-software (HW-SW) paradigm in information and communication technologies (ICT) is reaching its limits and must change. As the first step Semiconductor Research Corporation (SRC), partnering with Semiconductor Industry Association (SIA), has launched a new industry-wide road-mapping initiative called the 2030 Decadal Plan for Semiconductors. The use of the information and communication technologies continues to grow without bounds dominated by the exponential creation of data that must be moved, stored, computed, communicated, secured and converted to end user information. Future ICT systems will require a *true codesign optimization* across all layers from materials to applications (Fig. 1). A number of emerging *codesign challenges* anticipated over the next decade are outlined in the Decadal Plan, just three examples:

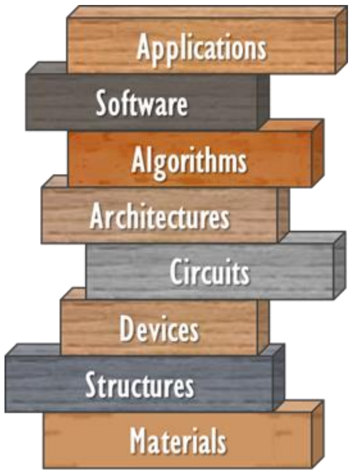


Fig. 1. Codesign optimization of ICT systems: Balancing over layers (source: Decadal Plan for Semiconductors [1])

1) Machine Learning/Artificial Intelligence: The recent explosion of artificial intelligence (AI) applications is a clear example, and we have only begun to scratch the surface. Having computing systems move into domains with true cognition, i. e., acquiring understanding through experience, reasoning and perception is a new regime. This regime is unachievable with the state-of-the-art semiconductor technologies and design practices. As a result, the current design paradigm must change to address an information and intelligence-based value proposition with semiconductor technologies as the driver. Industry has progressed beyond the production of general-purpose processors, advanced GPUs, and AI accelerator chips to now give way to approximate computing hardware. Approximate computing refers to the tradeoff in effort expended with computation quality. It has become pervasive in newer CPUs, GPUs, FPGAs, and memory. A key issue to be addressed is the memory access energy, which is about three orders of magnitude in excess of compute energy. This constraint underscores the inconvenience of moving data to the CPU for computation and makes near-memory computing (NMC) imperative. That said, the need for precision-scaling neutralizes the gains from current accelerators' near-memory compute. Research is underway to design memory arrays that exploit parallelism to lower data-movement cost. Critical evaluations of the performance of binary, ternary, and super-ternary/analog in-memory computing, e.g., with resistive switching, are underway. This in-memory computing requires better codesign of programmable architectures and peripheral circuits. It can be realized by storing data in RAM and processing it in parallel across a cluster of computers. This development in hardware should also be complemented with greater development in AI programming frameworks like graph networks and Causal/Explainable AI to yield codesign development that could achieve greater workload capabilities.

2) AI for Science: In 2019, the DOE and National Labs conducted four 'AI for Science' townhalls with a final objective of obtaining community consensus to guide the strategic planning for scientific AI for the 10 years [2]. Since algorithms like deep learning, as data-fitted functions between inputs and outputs, may have reached a stagnation point in their potential, it is necessary for the National Labs to drive greater collaboration with industry and academia to co-design heterogeneous computing solutions that integrate AI, data analysis, and scientific computing hardware designs. Algorithms and computer architectures for AI are evolving quickly and growing more diverse (e.g., neuromorphic, quantum, brain-inspiring computing). Effort at the intersection of these domains is also required.

3) Edge applications: A paradigm shift in how sensed signal or "information" is processed is required in order to provide an output (analog or few bytes) of detected "actionable information" from the sensed signal. High understanding of the key action objective is needed, as well as the signal and the associated "detection entropy"—and thus certainty or robustness. In classical information theory, Shannon defines the "information entropy" metric as the absolute minimum amount of storage and transmission needed for succinctly capturing any information (as opposed to raw data). Here this concept is being extended to the minimum actionable output required to take action, which is detected from sensing. This output could be data bits (even a single bit) or an analog output signal controlling driving actuation. To produce an actionable output, system knowledge will be required, as will consideration of added intelligence to all system components, from the sensor itself to analog signal processing, and possibly neural processing in analog and digital domains. Therefore, overall co-design will be required for the most

robust, compact, energy-efficient, and cost-eff ective solution, as also highlighted in the "Basic Research Needs for Microelectronics" report published by the Department of Energy Office of Science Workshop in 2018 [3]. This holistic codesign approach is recommended will require:

- Intelligent sensors and sensor-fusion research — multisensor distributed intelligence
- Applications and system knowledge research
- Hierarchical and distributed exploration/optimization
- Collaborative multi-expertise research projects — Moon Shot demonstrator platform
- System approach to optimization that crosses boundaries — sensor, analog processing, digital processing, ML/detection, etc.

The codesign aspects of the future Edge system will also impact DOE-specific tasks such as integration of processing with scientific instruments and diagnostics for experimental facilities. Also, edge computing capabilities are crucial for secure, resilient, operation and control of the future smart power grid. To prevent cascading failure modes, inferencing and training needs to be distributed throughout the power grid.

Perhaps the *biggest challenge* for the future ICT systems is the absence of unified codesign framework, with most current codesign effort being ad hoc, task-specific, and often labor intensive. In fact, different organizations have different definitions of what 'codesign' is. For example, for semiconductor companies the codesign occurs mainly on device-to-circuits level for emerging products, while the IT companies usually consider HW/SW codesign using of-the-shelf hardware offerings.

Opportunity:
The proposed strategy is to develop, based on our Decadal Plan, a Codesign 2030 Agenda, with a primary goal of creating a world-class Research Center focused on the development and dissemination of tools and methodologies for codesign of efficient electronic systems that address the topics above. *This will be a new, previously unexplored model of collaboration between DOE and industry via SRC*, a non-profit specializing in management of industry-relevant fundamental research.

In our experience public-private partnership is the best avenue to gain rapid advances in the codesign space. A public-private partnership in the form of a non-profit and neutral third-party consortium that connects academia, industry, and government provides the benefits of reproducibility, replicability, and quality of research. The main mechanism ensuring the high standards of research is technology transfer from the university lab into companies, which reproduce it within their research and development (R&D) facilities, tailoring it to their needs for integration into commercial products. The experience gained by industry can then be returned into the National Laboratories and academic setting in a virtuous cycle of learning and innovation that motivates benefits all participants in the consortium.

SRC (Semiconductor Research Corp.) is a consortium of semiconductor and IT companies that funds and manages university research on industry-relevant topics and that explore new technologies important for SRC members. SRC has 24 industry and government partners and has invested over $2.5 billion in university research over its 40-year history. Through collaborative research, SRC has fueled the technology engine defining the semiconductor and IT industries. The consortium has worked successfully with different government agencies and different business models.

SRC-led workshops on five seismic shifts in ICT, culminated in the 2030 Decadal Plan for Semiconductors. Along with our refined approach to collaborative research, it can be activated to address this research imperative and build a foundation for transforming codesign effectiveness. SRC is well positioned to accomplish this transformation by catalyzing the intersection between different 'layers' of the technology stack shown in Fig. 1. As one example of where SRC is currently driving codesign, IBM, Arm and Harvard's CHIPKIT framework provides a reusable SoC subsystem with basic IO, an on-chip programmable host, off-chip hosting, memory, and peripherals. New IP blocks can be added to generate custom test chips. Central to CHIPKIT, is an agile RTL development flow, including VGEN, a simple Python-based code generation tool [4].

Timeliness:
It is paramount to restore U.S. leadership in microelectronic technologies and innovation. With the 2030 Decadal Plan for Semiconductors released in January 2021, now is the crucial time to drive the conversion of the high-level Grand Goals of the Decadal Plan into a detailed Semiconductor Agenda toward 2030. Future ICT systems will require a true codesign optimization across all layers from materials to applications, and thus the Codesign Research Center and 2030 Agenda are most timely.

References:

1. SIA/SRC Decadal Plan for Semiconductors (SRC 2021) https://www.src.org/about/decadal-plan/.
2. Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown, "AI for Science" (DOE 2020), https://doi.org/10.2172/1604756
3. Basic Research Needs for Microelectronics (DOE 2018), https://doi.org/10.2172/1616249
4. Marco Donato, Glenn G. Ko, David Brooks, Gu-Yeon Wei, Paul N. Whatmough, Sae Kyu Lee, https://mrc-donato.github.io/CHIPKIT-Tutorial/