Report on

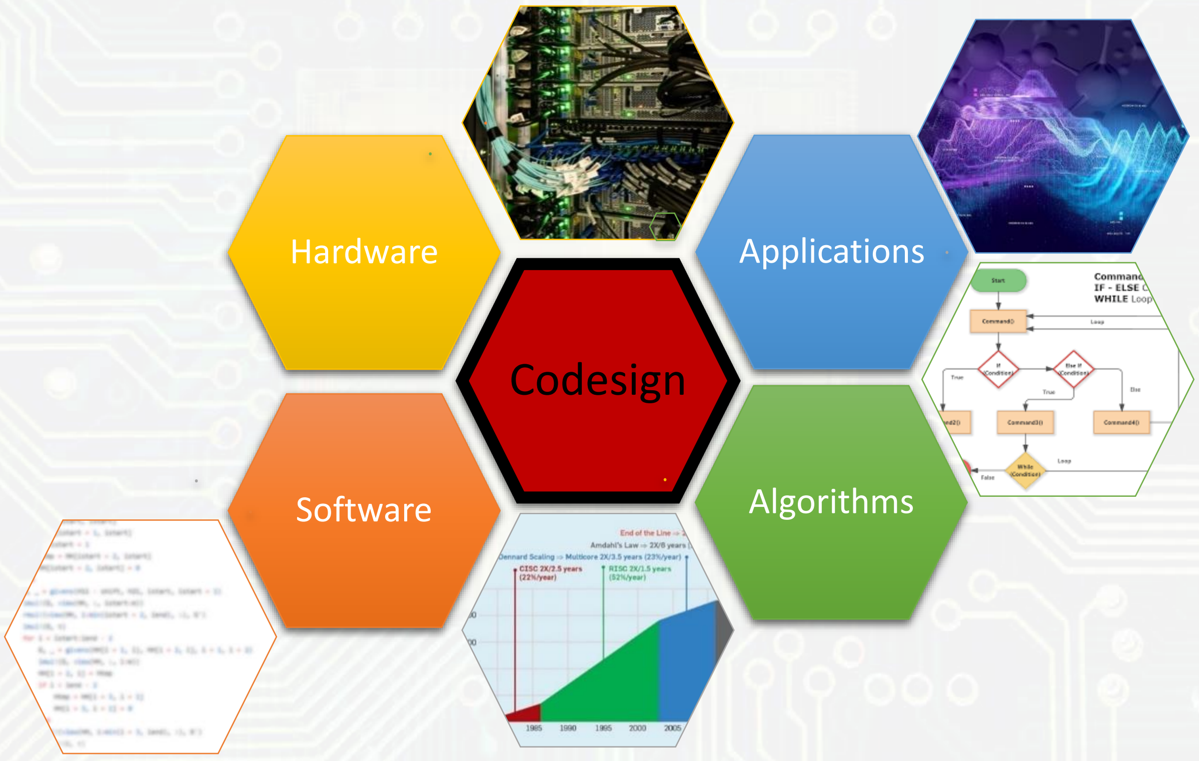
Basic Research Needs for

Reimagining Codesign for Advanced Scientific Computing

Unlocking Transformational Opportunities for Future Computing Systems for Science

16-18 March 2021

https://doi.org/10.2172/1822199



U.S. DEPARTMENT OF ENERGY | Office of Science

RECODE: WORKSHOP REPORT ON REIMAGINING CODESIGN

James A. Ang
Pacific Northwest National Laboratory

Andrew A. Chien
Argonne National Laboratory
and University of Chicago

Simon Hammond
Sandia National Laboratories

Adolfy Hoisie
Brookhaven National Laboratory

Ian Karlin
Lawrence Livermore National Laboratory

Scott Pakin
Los Alamos National Laboratory

John Shalf
Lawrence Berkeley National Laboratory

Jeffrey S. Vetter
Oak Ridge National Laboratory

April 29, 2022

ABSTRACT

In March 2021, the U.S. Department of Energy's Advanced Scientific Computing Research program convened the *Workshop on Reimagining Codesign*. The workshop, also known as ReCoDe, was organized around discussions on eight topic areas: (1) codesign for traditional high-performance computing workloads; (2) codesign of memory/storage systems; (3) codesign of machine learning, neuromorphic, quantum, and other non-von Neumann accelerators; (4) codesign for edge computing and processing at experimental instruments; (5) codesign for security and privacy; (6) hardware design tools and open-source hardware for high-productivity codesign; (7) tools, software stack, and programming languages for high-productivity codesign; and (8) quantitative tools and data collection for modeling and simulation for codesign. The panels identified four Priority Research Directions from these deliberations: (1) breakthrough computing capabilities with targeted heterogeneity and rapid design; (2) software and applications that embrace radical architecture diversity; (3) engineered security and integrity, from transistors to applications; and (4) design with data-rich processes.

Contents

Executive Summary

In March 2021, the U.S. Department of Energy (DOE)'s Advanced Scientific Computing Research program convened the Workshop on Reimagining Codesign. The workshop was organized around discussions on eight topic areas. From these, the workshop panelists identified four Priority Research Directions.

Motivation. Silicon-based transistors are nearing the limits of miniaturization, and the slowing pace of performance gains from smaller transistors is driving large-scale disruption of the entire computing ecosystem. As a result, the high-performance computing (HPC) environment has transitioned from being dominated by a few relatively similar general-purpose central processing unit (CPU) architectures to being led by some mostly comparable and increasingly general-purpose graphics processing unit (GPU) accelerator architectures. In alignment with the computing industry, it seems probable that these CPU/GPU architectures will be rapidly augmented by a diverse set of systems that comprise a broad portfolio of commodity plus customized modular components that include CPUs, GPUs, and artificial intelligence (AI) accelerators where specialization provides benefits (as illustrated in Fig. 1). This new ecosystem offers opportunities to significantly improve the performance, energy efficiency, productivity, reliability, and security of scientific applications. Yet, exploiting these opportunities requires the development of innovative techniques and tools to codesign future software and hardware rapidly using verified, data-driven methodologies. Five key factors that distinguish the present or near future from the past, as shown in Fig. 2, are enabling transformative research in this new technological environment.

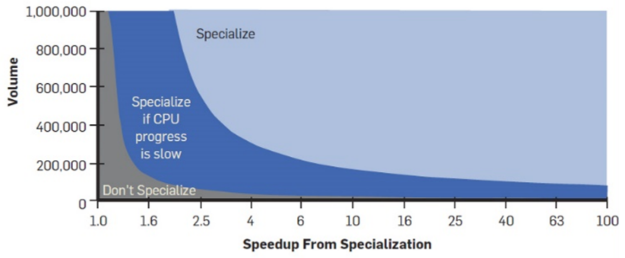
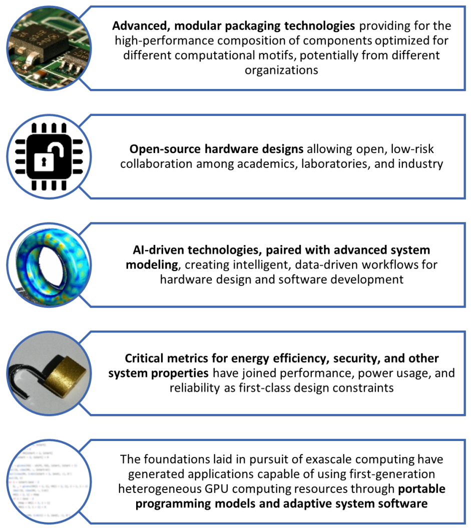


Figure 1: Benefits of Specialization. (Courtesy N.C. Thompson)

Reimagining Codesign. Although DOE has successfully employed the codesign methodology to improve software and hardware in several advanced HPC systems (see Fig. 3), codesign has been a distinct process, starting with workload analysis and ending with deployment and operation. Workshop attendees concluded that a *reimagined* codesign process (illustrated in Fig. 4) that is continuous, agile, and secure would better reflect the new reality of rapidly changing workloads and architectures. That is, future computing architectures will require substantially expanded scope to account for a broadened spectrum of applications that include AI, machine learning (ML), and graph methods for data-driven science; end-to-end processing for experimental instruments that aggregate and analyze real-time experimental data; and traditional numerically intensive HPC workloads. Each will require a start-to-finish codesign approach that meets these new, more diverse mission requirements. In addition, these computing architecture designs must account for new deployment scenarios at the edge and in the cloud alongside traditional HPC data centers.



Advanced, modular packaging technologies providing for the high-performance composition of components optimized for different computational motifs, potentially from different organizations

Open-source hardware designs allowing open, low-risk collaboration among academics, laboratories, and industry

AI-driven technologies, paired with advanced system modeling, creating intelligent, data-driven workflows for hardware design and software development

Critical metrics for energy efficiency, security, and other system properties have joined performance, power usage, and reliability as first-class design constraints

The foundations laid in pursuit of exascale computing have generated applications capable of using first-generation heterogeneous GPU computing resources through **portable programming models and adaptive system software**

Figure 2: Enabling Key Technology Factors.

Findings and Priority Research Directions. Codesign in HPC and AI has been critical to the design and implementation of contemporary computer architectures. As HPC applications evolve to include features for AI, connections to experimental facilities, and potentially mobile devices, the architectures and software also must adapt much more quickly to serve these new emerging workloads efficiently. In this regard, the codesign process has to be reimagined to be continuous, agile, and secure to reflect the new reality of rapid change in both workloads and architectures. The four Priority Research Directions outlined herein provide a sound foundation for a cohesive, long-term research and development strategy in reimaging codesign for advanced scientific computing. Over the last decade, DOE has invested

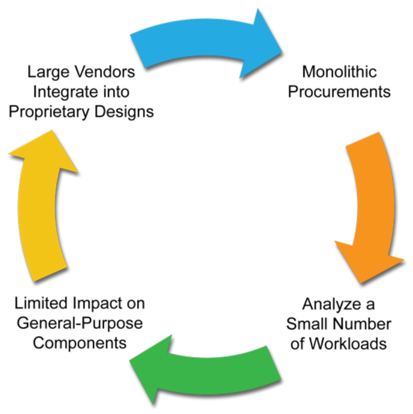

Large Vendors Integrate into Proprietary Designs
Monolithic Procurements
Analyze a Small Number of Workloads
Limited Impact on General-Purpose Components

Figure 3: Past Codesign Activities.

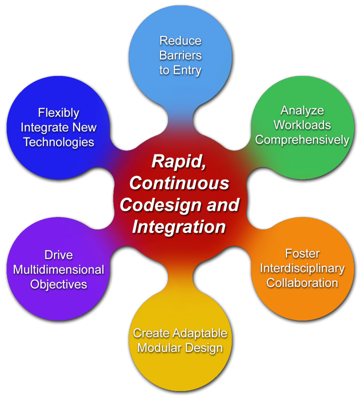

Reduce Barriers to Entry
Analyze Workloads Comprehensively
Foster Interdisciplinary Collaboration
Create Adaptable Modular Design
Drive Multidimensional Objectives
Flexibly Integrate New Technologies
Rapid, Continuous Codesign and Integration

Figure 4: Reimagined Codesign: Rapid, Continuous Codesign and Integration.

heavily in codesign through the Exascale Computing Project. This effort has created a baseline for codesign activities that will underpin key advances in these four Priority Research Directions. Such advances will build on this prior work from leading researchers in computer architecture, programming systems, simulation tools, workflow management. Too, new research areas will emerge from the pursuit of next-generation computing systems.

Priority Research Directions (PRD)

PRD 1: Drive Breakthrough Computing Capabilities with Targeted Heterogeneity and Rapid Design

Key Questions: What new methods and technologies are required to rapidly create breakthrough hardware designs? How can we ensure they align to support increasingly diverse and demanding computing requirements?

In today's rapidly changing technology landscape, designing radically heterogeneous systems requires new codesign methodologies that afford an accurate understanding of workloads and use it to target and drive the creation of complementary sets of accelerators and heterogeneous structures that combine to produce breakthrough systems. The flexible accelerated integration of such heterogeneous customized elements depends on advanced new physical integration technologies (e.g., chiplets and other leading-edge packaging), architectural integration (e.g., new memory interfaces, communication links, or open standards/protocols), and accelerated hardware development (e.g., open-source designs or technology libraries and open-source tools). This research will yield innovative approaches to system design that require only incremental enhancements to underlying microelectronics technologies while simultaneously complementing longer-term microelectronics research.

PRD 2: Software and Applications that Embrace Radical Architecture Diversity

Key Question: What novel approaches to software design and implementation can be developed to provide performance portability for applications across radically diverse computing architectures?

Programming models, compilers, libraries, and runtime systems must work with many types of compute engines and even new compute paradigms that differ dramatically from the traditional von Neumann architecture abstraction. Novel developments are needed in software abstractions to increase application portability; in dynamic runtime systems to discover, schedule, monitor, and control highly varied resources; in tools for analyzing and predicting performance in the context of radical architectural diversity; and in metrics and benchmarks for quantifying progress beyond mere performance.

PRD 3: Engineered Security and Integrity from Transistors to Applications

Key Questions: How does codesign consider needs for end-to-end scientific computing and scientific data security, provenance, integrity, and privacy? What computer security innovations from the commercial computing ecosystem *(e.g., trusted execution environments) can be codesigned to provide security for DOE scientific discovery? How do we validate components with increasingly diverse supply chains and development sources?*

Revolutionary advancements are needed to extend roots of trust and other security capabilities to support DOE's scientific discovery continuum that leverages research networks, such as the Energy Sciences Network, to integrate supercomputing facilities with experimental user facilities. This may extend to advanced wireless-networking -enabled Internet of Things sensors. A distributed scientific ecosystem provides increased exposure to threats, but it also offers the opportunity to leverage ubiquitous logic capabilities to enhance computer and data security. As another area of concern, quantitative metrics assess how security codesign trade-offs can be made in conjunction with traditional metrics (e.g., power, performance, and reliability). Moreover, given the proliferation of complex, modular components and community-developed open-source technology, the ability to validate new technology to protect against defects, including any intentional ones, must grow substantially.

PRD 4: Design with Data-rich Processes

Key Questions: What are the quantitative tools that are practical, accurate, and applicable to codesigning various layers of the hardware/software stack and data-driven, dynamic, irregular workflows, such as those occurring in experimental science or AI/ML workloads?

To be successful, quantitative tools, such as simulators, emulators, or profilers, must be applicable to design ahead (i.e., in advance of implementation) and follow through to assist optimizations during the execution of complex workflows on the target systems. These modeling and simulation capabilities must: (1) be sufficiently fast for repeated and potentially online use; (2) consider the triad of performance, power, and reliability in an integrated fashion; (3) be accurate over a broad range of hardware and software architectures; and (4) be scalable as the system complexity and size increase. Dominant workflows are steadily more data-driven, dynamic, and irregular, requiring new methods and codesign tools that are dynamic and runtime-oriented.

1 Introduction

Silicon-based transistors are nearing the limits of miniaturization, and the slowing pace of performance gains from smaller transistors is driving large-scale disruption of the entire computing ecosystem. As a result, the high-performance computing (HPC) ecosystem has transitioned from being dominated by a few, relatively similar general-purpose central processing unit (CPU) architectures to being led by a handful of analogous and increasingly general-purpose graphics processing unit (GPU) accelerator architectures. In alignment with the computing industry as a whole, it is likely these CPU/GPU architectures will be supplanted by a diverse set of systems that comprise a broad portfolio of commodity and customized modular components that include CPUs, GPUs, and artificial intelligence (AI) accelerators. This new ecosystem offers opportunities to significantly improve the performance, energy efficiency, productivity, reliability, and security of scientific applications. However, exploiting these opportunities requires the development of innovative techniques and tools to codesign future software and hardware using verified and data-driven methodologies.

Although the U.S. Department of Energy (DOE) has successfully employed codesign as a methodology to improve software and hardware for advanced HPC systems, future computing needs require an expansion of codesign scope far beyond contemporary techniques. Codesign engagements must be broadened to include edge computing and data processing for experimental sciences. This reimagining requires codesign as a fundamental ingredient for all areas of scientific computing, including:

- Traditional, numerically intensive HPC workloads
- Data-intensive computing, graph analytics, and AI for data-driven science
- End-to-end and federated processing for experimental systems that span the continuum, from computing embedded into sensors (i.e., edge computing) to HPC centers that aggregate, analyze, and store experimental data.

Each focus will require a tailored codesign approach to meet the diversity of its respective mission requirements.

Emerging transformational technologies provide the opportunity for powerful new approaches in hardware and software codesign. These technologies will allow for specialization in future computing environments by addressing the needs of emerging DOE applications in HPC, edge computing, and data processing for experimental sciences. In response to the slowing performance gains from scaling transistors, specialized accelerators are being designed, which is fueling a trajectory toward extreme heterogeneity. Application trends—such as the increasing complexity of software, incorporation of AI, and use of system-specific constructs that increase the effort needed to target new hardware technologies—are compounding the challenges of programming specialized accelerators. Fortunately, advances in fabrication, packaging, portable programming models, open-source hardware, open-source hardware simulation tools, and other areas offer new opportunities for codesigning hardware, software, and applications in a variety of domains.

1.1 Defining Codesign

The DOE has successfully employed codesign as a methodology to improve software and hardware in advanced HPC systems for a number of leadership-class deployments [1, 2]. The most recent examples are the soon-to-be-deployed Aurora, Frontier, and El Capitan exascale systems. By working with system vendors in a codesign mode, the mission-relevant capabilities of these computing systems have been enhanced. Reimagining codesign contemplates the expansion of codesign in scope and ambition to reflect the clear and increasing importance of computing as a foundation for science. Furthermore, a reimagined perspective of leadership-class computing will broaden codesign within the DOE to include future edge-computing and advanced data-processing components for experimental sciences.

Transformative opportunities for future performance growth will require tighter integration of domain scientists, mathematicians, analysts, and computer scientists within the codesign process, as well as the tools and multidisciplinary vocabulary to enable productive collaborations with software and hardware designers. The rapid change of contemporary hardware and computing ecosystems affords new opportunities for codesign. For example, combining new hardware and algorithms to be transformative, not just evolutionary. In an era where creating specialized hardware to match an application is the primary means of substantial performance improvement, the economic model for designing future systems will need to change dramatically to decrease hardware design and verification costs. An enabling technology for this vision is the emergence of agile microelectronics packaging methods, such as the use of chiplets [3, 4]. Rather than have a single large piece of silicon that integrates all of the diverse accelerators included in the customized hardware, chiplets break each piece of functionality into small blocks. Each chiplet then can be integrated by bonding to a common silicon substrate that provides fast inter-chiplet data transfers. This approach enables manufacturers to rapidly piece together a mosaic of chiplets to serve diverse specialized applications at a lower cost and with a faster time to production. However, this approach fails if the desired functionality does not exist in the available chiplets. Then, the community must consider approaches that can augment available designs with open-source solutions from DOE and academia. In the future, "algorithm-driven hardware design" combined with the chiplet approach will bring forth a new economic model [5] that can enable productive architecture specialization for small markets.

1.2 Reimagining Codesign

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 [6]. Advances in fabrication, packaging, and other areas continue to create opportunities for codesigning hardware and software in a variety of different domains. While large-scale scientific computing continues to drive the leading edge in HPC, codesigning the next generation of systems for HPC, edge computing, and all scales in between—along with the development of scientific software—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-source hardware simulation environments, open instruction-set architectures, and open-source software toolchain components has 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 components of the codesign equation. The workshop participants explored opportunities to reimagine future codesign methodologies for hardware and software relevant to scientific discovery.

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

Overall, there is solid consensus that the tapering of Moore's Law [7] will lead to a broader range of accelerators or specialization technologies than has been seen in the past three decades. Examples of this trend exist in smartphone technologies, which contain dozens of specialized accelerators colocated 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 machine learning (ML) tasks [8]; in field-programmable gate arrays (FPGAs) in the Microsoft Cloud used for Bing searches, in AI/ML and other applications; and a vast array of other deep learning accelerators. The industry is moving forward with production implementations of diverse acceleration in the AI/ML markets (e.g., Google TPU, Nervana's AI architecture, and Facebook's Big Sur) and other forms of compute-in-network acceleration for mega data centers (e.g., Microsoft's FPGA Configurable Cloud, Project Catapult for FPGA-accelerated search). Even before the explosive growth in the AI/ML market, system-on-a-chip (SoC) vendors for embedded, Internet of things (IoT), and smartphone applications were already pursuing specialization to notable effect. Shao et al. [9], from Harvard University, surveyed the 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 sixth-generation iPhone SoC to well over 40 discrete accelerators in their 11th-generation chip—not to mention Apple's recent move to the Arm-based M1 chip that offers seamless use of many heterogeneous accelerator cores and outperforms the leading-edge Intel x86 designs that it replaced.

Success also has been demonstrated in creating science-targeted accelerators, such as D.E. Shaw's Anton [10], which accelerates molecular dynamics (MD) simulations by more than 180 times that of contemporary HPC systems, and the GRAPE series of specialized accelerators for cosmology and MD. At the 2016 International Symposium on Computer Architecture workshop on the future of computing research beyond 2030 (Arch2030) [11] concluded that heterogeneity and architecture diversity are nearly inevitable given current architecture trends. The trend toward co-packaging of diverse "extremely heterogeneous" accelerators is already well under way. Although many conflate codesign with whole-system customization (e.g., Anton), one likely outcome of codesign is a number of efficient and specialized components (chiplets or chips) and general-purpose integrants that can be flexibly combined in a system design matched to a particular application domain or workload.

On the application side, proxy applications 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 and often mix HPC applications, data analytics functions, and ML. In addition, AI and edge computing are growing in importance, and ensuring these applications are represented will be key to optimizing hardware and software trade-offs at the core of codesign engagements. The methods used to program portable applications, such as using C++ frameworks including Kokkos [12] and RAJA [13], along with advances in programming languages and tools, are changing how even large-scale applications can evolve. These changes are also vital for understanding future codesign opportunities.

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

Licensable IP for Server-Class Processors. The dominant cost for chip production is the design and verification of the logic, which traditionally was amortized over the sales of many chips (i.e., in a large market where the chip is the commodity). However, in the embedded market, intellectual property (IP) is the commodity, i.e., the IP is the verified design, which is the most expensive activity, 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 IPs (e.g., Arm V8 and Scalable Vector Extension (SVE) instruction sets) is offering capable double-precision floating point and 64-bit address capability, as well as options for high-performance input/output (I/O) and memory interfaces. The growing server-class licensable IP ecosystem could enable a new path to affordable flexibility for custom hardware for government processing needs.

Chiplets. Conventional SoCs integrate heterogeneous circuits onto a single die, which is expensive as die sizes grow. A chiplet integration strategy breaks the components into pieces that can be fabricated on much smaller dies then integrated by a common interposer using 2.5D integration methods. The emerging chiplet approach offers a faster, less expensive way to assemble various types of third-party chips, such as I/Os and memory and processor cores, in a single package (a silicon motherboard). The combination of chiplets can be flexibly altered for each customer, which allows for deployment-specific hardware solutions. This approach can dramatically reduce the costs of specialized components assembled from a library of premade building blocks. Furthermore, and the emerging open chiplets marketplace based on open die-to-die interconnect standards such as ODSA and UCIe [5, 14–16]. Combined with licensable IP (such as Arm) and open-source hardware (RISC-V), this emerging open chiplets marketplace encourages opportunities for creating a new economic model that could transform HPC as we know it.

Standardized Accelerator Interfaces. The emergence of CCIX, Coherent PCIe, and CXL [17] as industry standards for co-integration of diverse heterogeneous accelerators provides an opportunity for multi-vendor heterogeneous integration to become mainstream. This affords HPC integrators the ability to tailor the delivery of heterogeneous accelerators at the system-integration level rather than producing their own custom silicon, thereby reducing the cost, time, and risk associated with delivering specialized hardware.

Advanced Packaging Technologies that Enable Heterogeneous Integration. The Heterogeneous Integration Roadmap (HIR) is an industry-led initiative for delivering performance improvements [18] by integrating separately manufactured components (e.g., system in a package [SiP]). This idea follows the conceptual vision to build large, complex systems out of smaller, separately packaged functions (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 five performance doublings. Immersion cooling represents a complementary breakthrough technology for both performance density and heterogeneous integration. The ability of two-phase immersion cooling to transport heat can be increased by 10–100 fold, which would enable large increases in performance density [19, 20]. Just as important, immersion cooling decouples the cooling problem from functional system design, increasing flexibility in design and late-binding system integration choices.

Open-source Hardware and Open-silicon Compilers. Most hardware design is proprietary, and the commodity is the chip. Licensable hardware IP (e.g., Arm and Cadence) has enabled third parties to create their own custom designs, but licensed IP still can be expensive. Even with low-cost academic licensing, the results cannot be openly distributed to a broad research community. The emergence of open-source hardware/IP (e.g., RISC-V) and open-source silicon compilers (e.g., OpenRoads) offers a path to reducing licensing costs, which may accelerate hardware development and democratize hardware design. With open-source hardware, noncommercial entities (e.g., laboratories and academia) can actively participate in hardware design and development.

Advanced Hardware Description Languages and Hardware Generators. Some of the most significant costs for hardware development come at the design and verification stages. Emerging advanced hardware description languages (e.g., Chisel [21], PyRTL [22], and PyMTL [23]) have incorporated modern programming language techniques, including inheritance, polymorphism, and strong-type systems. Frameworks such as Aladdin use automation to enable more targeted accelerator design. By bringing these modern techniques to hardware design, languages can dramatically lower the cost of specialized hardware and architectural designs that target science applications. More importantly, these new expressive languages can enable hardware designers to introduce a broader set of users (e.g., domain scientists or mathematicians) into the design loop, which is essential for guiding targeted specializations.

Open-source Hardware Simulation and Modeling Frameworks. Historically, hardware projects have developed highly specific, overly focused models to evaluate potential design trade-offs. For the most part, these models have been proprietary and customized, leading to a frustrating situation where comprehensive modeling capabilities that explore full-node or full-system behavior cannot be implemented or analyzed. Developments toward community-based, high-performance, and open-source tools [24] offer an alternative in which individual component models can be integrated to improve analysis capabilities. As the industry moves toward a more customized and component-driven hardware future, developing these types of capabilities will give research teams true comprehensive modeling tools that will allow for critical trade-offs within codesign to be more concretely explored.

Coarse-grained Reconfigurable Arrays. Examples of the reemergence of static-dataflow and reconfigurable computing 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 of thinking about algorithm design [25] (e.g., superpipelining).

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. Photonic-based networks enable runtime specialization and 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, Advanced Research Projects Agency–Energy's (ARPAe's) ENLITENED, and the Defense Advanced Research Projects Agency (DARPA)'s PIPES. Continued evolution of these technologies could enable disaggregation efficient enough for state-of-the-art HPC systems.

Programming Abstractions and Languages. Modern application development increasingly relies on techniques made possible by modern programming languages. Many DOE applications now use C++ frameworks, such as Kokkos [12] and RAJA [13], to enable portability across different accelerator technologies. Increasingly, application teams are investigating new paradigms and languages (e.g., Legion [26]; Julia [27]), and some use 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.

Open-source and Extensible Compiler Frameworks. The emergence of open-source compiler frameworks with intermediate representations (IRs) that can be more easily extended (e.g., LLVM with MLIR [28]) could offer incentives for vendors to more easily accept complex HPC programs and transform them into optimized binaries for their own heterogeneous hardware. Extensions such as MLIR further enable higher-level optimizations that can target accelerators while working with existing languages that currently dominate DOE's HPC code portfolio (C, C++, and Fortran).

AI-integrated Applications. Science applications are increasingly integrating AI/ML technologies as part of "outer loop" workflows and "inner loop" core algorithms. The merging of data-driven modeling into core aspects of HPC applications, fed by experimental and simulation results, enables new hardware-acceleration paradigms, such as those using reduced-precision arithmetic. Moreover, this integration opens up new possibilities for multiscale modeling, adaptive simulation, and *in situ* analysis.

Although DOE is not in the business of fabricating HPC components, numerous opportunities exist within DOE data centers to exploit codesign techniques. At the procurement stage, supercomputers can be selected based on how well they match applications' needs by considering the ability for applications to adapt to the hardware. At the configuration stage, firmware and system software—and sometimes hardware decisions 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 toward the future, a number of transformative opportunities exist for DOE to collaborate much closer with hardware vendors, such as using processors based on chiplets [29, 30], where DOE could outright design or assist in designing a small unit of DOE-centric logic for incorporation into a base processor.

1.3 Workshop

In March 2021, DOE's Advanced Scientific Computing Research (ASCR) program convened the *Workshop on Reimagining Codesign* (also known as ReCoDe). The workshop was organized around breakout groups focused on eight topics:

(1) Codesign for Traditional HPC Workloads
(2) Codesign of Memory/Storage Systems
(3) Codesign of Machine Learning, Neuromorphic, Quantum, and Other Non-von Neumann Accelerators
(4) Codesign for Edge Computing and Processing at Experimental Instruments
(5) Codesign for Security and Virtualization
(6) Hardware Design Tools and Open-Source Hardware for High-Productivity Codesign
(7) Tools, Software Stack, and Programming Languages for High-Productivity Codesign
(8) Quantitative Tools and Data Collection for Modeling and Simulation For Codesign.

Each breakout group generated a number of findings from their respective discussions, which were distilled into four Priority Research Directions.

In addition, the workshop advertised an open call for contributed white papers on this topic. These contributed white papers are archived at OSTI.gov (https://doi.org/10.2172/1843574).

2 Workshop Findings

2.1 Codesign for Traditional HPC Workloads

2.1.1 Rationale

For many decades, DOE has been at the forefront of traditional high-performance supercomputing, leading the world into the giga-, tera-, and peta-flop computing eras. Around 2008, DOE pushed the boundaries of computing to new limits with the deployments of Titan at Oak Ridge National Laboratory (ORNL) and Roadrunner at Los Alamos National Laboratory as both systems employed heterogeneity in their compute nodes to accelerate specific application kernels. Acceleration that leverages specialized compute hardware can be dramatic with multiple orders of magnitude increases in performance for algorithms with sympathetic mapping to the specialized hardware design. This trend has continued with subsequent deployments of Summit, Sierra, and Perlmutter. However, with all of these deployments, the basic approach to designing systems—via a single replicated compute node design interconnected by a high-speed network—has remained commonplace. Moreover, using the system with a batch-operated queue of jobs allocated to a fixed set of resources for the life of the computation has remained unchanged since the 1970s. Although using a singular node design across an entire supercomputer has some benefits, including for extremely large-scale runs where node uniformity allows them all to be used for computation, the result often can be a complete lack of flexibility in supporting more modern, dynamic, and varied workloads or application requirements.

As DOE's scientists more readily adopt AI/ML techniques and become increasingly reliant on large-scale data science, there is a growing need for HPC resources to exhibit additional flexibility in executing and accelerating a broader mixture of traditional and novel software runtimes, programming models, and algorithms. Put simply, user requirements in DOE facilities are becoming more varied, which implies that a continuation of the "one-size-fits-all" compute node and batch-queued allocation model will no longer enable the rapid scientific delivery that program sponsors and the community require.

To address these concerns, there is a burgeoning expectation that future HPC systems and leadership facilities must be designed differently. Rather than a relentless focus on *uniformity* of compute node designs and software environments to create the largest possible scale for applications, the scientific community will be able to gain greater benefit from *differentiation* in compute nodes, compute partitions, or even with variation in system partitions. The advantage of such an approach arises from acknowledging that *not all* algorithms or workloads benefit from a specific, single hardware design or software environment. Some algorithms work well when ported to accelerators, like GPUs, whereas others obtain their highest performance on a CPU. For certain scenarios, per-thread performance of recent systems has largely plateaued challenging application design. Still, other algorithms might produce their greatest performance on alternative approaches, such as dataflow computing, processing near the network, or on novel devices such as neuromorphic or quantum processing elements. Furthermore, different user environments provide variation in functionality placing software as a key component of the codesign landscape. Combined with this observation is an expectation that the approach to simulating physical phenomena in the future will also change. Utilizing a single application written in C/C++ or Fortran with the Message Passing Interface (MPI) is more likely to be viewed as a component in a much larger and more complex workflow that blends ML training and inference with data conditioning or feature analysis and sends data to online analysis and visualization resources. Each component in such a complex workflow is expected to have differing hardware and software requirements, which implies heterogeneity as a viable method to provide the highest overall throughput.

Several options for HPC system designs are described herein. These options should not be regarded as distinct paths, and composition or overlap of these approaches is possible. Given the diversity of the system design options, community researchers must develop credible evaluation methods for these designs and consider planning and research for the associated impacts on system software, libraries, tools, programming models, and applications.

Workload-optimized HPC Systems. In this approach, supercomputing resources are designed for specific workloads. By focusing on narrowly defined classes of workloads, high levels of customization are possible in the entire stack, from the underlying hardware to the applications, including specific, fixed-function circuit designs. The negatives of such a narrowly focused approach include leaving large parts of the HPC community without acceleration and the potentially inflexible solutions that cannot adapt to changing science drivers over short periods of time.

Heterogeneity of Nodes within a Single HPC Machine. This approach uses multiple partitions within the same machine, and each partition includes workload- or application-optimized components. This method replicates designs such as those used in the National Energy Research Scientific Computing Center (NERSC)'s Cori and Perlmutter systems, where parts of these systems contain CPU-only nodes and other parts feature CPU/GPU-based nodes. In the future, increasing heterogeneity could boost the number of partitions within HPC systems and see more narrowly defined/optimized partitions for specific sections of the user base. The advantages to such a design include sharing resources between parts of full application workflows (in which individual parts may run best on individual partitions) and the potential for multiphysics applications to use different partitions for each physics component. This type of application design would clearly impact system software, runtimes, libraries, and algorithms. Some downsides in using this design include correctly determining the size and constituent components of the partitions and ensuring there is sufficient dynamic utilization of the resources to improve aggregate scientific output.

Heterogeneity within Compute Nodes. This approach extends the state of the art to include multiple accelerators within a single compute node. Although CPU/GPU nodes are becoming commonplace today, a future node might include additional fixed-function or general-purpose accelerators for the application and libraries, as well as accelerators in the network, storage, and memory components. This approach adds complexity to system software, library, and algorithm designs because kernels and functions must be offloaded and shepherded to the appropriate hardware either statically or dynamically. Deciding which components to provide and how to assemble them into a high-performance, single-node design is not yet a fully understood problem. Moreover, on the downside, this approach requires provisioning many silicon components that may not all be efficiently used at any single point in time, which could lead to potentially higher power and capital cost.

Heterogeneity within the HPC System Package or Chip. Modern processors typically include a range of small functional units to handle dedicated tasks. These range from simple units that interface with USB devices or disk/storage controllers to more complex SoC devices that may integrate graphics processing acceleration or advanced memory subsystems. In the future, HPC-optimized SoCs or system packages could integrate a variety of accelerators to offload processing from the main cores and may include compute, ML, and network acceleration units. Codesign projects should work to ensure that the units selected provide the greatest acceleration to HPC workloads running on the SoC/package.

Heterogeneity in End-to-End Systems. The last approach is heterogeneity in system design across all aspects of the data center, which may comprise multiple HPC resources, networking components, storage devices/methods, and analysis resources. This approach could include orchestration of HPC resources to support scientific instrument processing and steering. A full data center and instrument-wide approach to system design and optimization would have significant impact across the HPC space and may influence where HPC systems are geographically located when connected to large-scale scientific instrument facilities.

Proxy applications and benchmarks. Proxy applications, also called *mini-apps*, have a considerable influence on the codesign process for DOE's exascale-class deployments [31]. However, the community has recognized that several lessons can be learned from these experiences for future codesign activities. One of the most pressing involves acknowledging that proxy applications have been mostly focused on specific sections of an application or even on certain algorithms. As a trade-off, the associated complexity of the implementation must be kept low and tractable for rapid modification in iterations of codesign. Representations of end-to-end workflows for future systems are becoming increasingly important, particularly in the context of heterogeneous compute resources for technologies that include exotic accelerators, as well as processing in the network fabric, memory, and storage. Future codesign activities will need to further broaden the definitions of proxy applications and take a more workflow-centric approach.

In addition, contemporary mini-app libraries have only limited coverage of AI and ML capabilities. This is caused partly by the rapidly evolving nature of AI/ML research within the community, which has yet to settle on a consensus for the methods, data types, libraries, and algorithms that should be employed. Similar concerns exist for exemplar data sets and models. While agreement on a unique approach is unlikely, representation of a few leading methods in future proxy application portfolios will be essential to enable integration in future HPC system designs.

Quantitative analysis of HPC systems. One outcome of past exascale codesign activities has been that hardware, system software, algorithm, and application design steps require a rich and accurate set of quantitative performance, power, energy, and cost data to drive decision making. In the past, often only a limited amount of information in these areas was available to many codesign teams, which may have prevented some areas of exploration. Future codesign activities should consider developing tools and processes to adequately capture relevant, accurate, and timely information to support crosscutting codesign. In particular, this impacts a wide variety of areas in modern systems, including hardware counters, software profilers, emulators, simulators, compiler-based instrumentation, and full system-wide data capture. Given the immense amount of data that may be obtained by instrumentation analysis of large-scale HPC systems, there is an opportunity to develop ML and data analytics capabilities that could lead to more meaningful and deeper data insights. One outcome of this area also should include the development of community-consensus metrics that can appropriately guide codesign and show the inherent value of specific approaches (or designs) in the future rather than repetitive use of LINPACK. LINPACK as a metric does not accurately reflect the performance requirements of complex applications on next-generation HPC systems.

Success stories in HPC codesign. Although not an HPC system, one of the most recent and often cited examples for the potential of codesigned systems is the use of *Apple Silicon* [32], an SoC and software ecosystem designed to produce exceptional battery life and system performance for laptop and tablet form factors. Significant improvements in the SoC come from using a broad range of accelerators, including functional units dedicated to video, image, sound, and ML processing, which offload specific compute tasks from the general-purpose processor cores. Although the Apple system is commonly referenced in the community, numerous examples of processor cores bound to accelerators have been developed, including AMD's Accelerated Processing Unit [33], Intel's processor-integrated Iris Graphics and AVX vector units, NVIDIA/Mellanox BlueField SmartNICs, numerous Arm IP-blocks, and IBM's Matrix Math Accelerator (MMA) found in its Power10 server processor [34].

Using optimized subsystems or components to improve supercomputing performance has been an established part of the field since its inception. Throughout its history with supercomputing deployments, DOE has provided research funding to hardware vendors and experts at its laboratories to jointly explore, develop, and optimize the critical components needed to increase the performance and scalability of its systems. Most recently, this was demonstrated in the DOE PathForward hardware research and development projects [35] that funded six industry partners and DOE staff to explore improvements in networking, processors, accelerators, and memory subsystems collaboratively. PathForward builds on a legacy of earlier smaller projects, including FastForward, DesignForward, and an earlier version of PathForward for the original Advanced Simulation and Computing Program at DOE's National Nuclear Security Administration.

However, selecting and designing optimized components differ from a truly integrated and codesigned system, where end users, software (e.g., applications, system software, runtimes), and hardware subject matter experts collectively develop and optimize full system designs. In this approach, more drastic and impactful changes can be developed owing to a narrower focus. Contemporary examples of this in action are the MDGRAPE family of machines [36–38], which were designed to perform N-body simulations at exceptional levels of performance and efficiency, and the Anton-series of systems [10, 39, 40] developed by D.E. Shaw Research to accelerate MD workloads. In both cases, the codesign team was able to utilize deep knowledge of the workloads, including specific data types, data formats, and algorithms, to much more precisely optimize the associated processors and interconnect fabric. By also having control of the software environment, workload-specific optimizations could be integrated directly into the control and runtime systems. The results of this integration are dramatic with several problems executing at more than 20× the performance of comparable general-purpose architectures of the day. These increases have a clear impact on scientific delivery from the system and make intractable science problems possible.

2.1.2 Gaps and Challenges

Complex application and workflow models/descriptions. The complexity of contemporary scientific simulation workflows is already changing substantially. Although this is partially driven by the inclusion of online data analytics and ML capabilities, many users also must identify scientific artifacts or important data items while executing simulation code to handle the vast increase in dataset sizes that exascale computing platforms provide. Writing out large swaths of application data for each run during simulation is entirely infeasible and motivates many users to perform online analysis and store the critical data during the workflow's execution. The outcome of these new approaches is that many users are including additional libraries and frameworks into their scientific computational pipelines using nontraditional languages and products, such as Python, Jupyter notebooks [41], PyTorch [42], TensorFlow [8], and others. If the codesign of future HPC systems is to become a common approach, representative, full, end-to-end workflow definitions, including mini-apps and benchmarks that use nontraditional software packages and the linkages between them, will be necessary. These mini-apps will differ significantly from the current state of the art, which typically concentrates on the computational aspects of a single application, thereby ignoring data load/store operations and links to analysis or visualization packages. In particular, future mini-apps and benchmarks must include special-purpose frameworks to expand the codesign capabilities from being focused on the typical languages of C, C++, and Fortran using MPI to interpreted languages, complex adaptive runtimes, and multi-node communication frameworks that can connect heterogeneous node types, system partitions, or even multiple machines within a data center.

By extension, the codesign community, which to date has focused mostly on the optimization of single applications, will need to adopt these more complex workflow definitions to ensure performance analysis and quantitative system studies are performed with all of the disparate software components in mind. The implication is that traditional profiling methods used for HPC, which typically are single-application focused and link to running applications using MPI and accelerator programming hooks, must be greatly broadened to include a growing list of HPC and workflow-critical software packages. Without robust and accurate quantitative analysis, the performance baselines and success metrics will be poorly defined, reducing the impact of codesign as a tool to enhance scientific output.

HPC system architecture design baselines and exemplars. For HPC, one of the fundamental successes of the last two decades has been a largely similar machine model for subsequent system designs. The predominant model, described crudely, has been a domain-decomposed problem executing on a cooperating group of homogeneous processors with explicit message passing to share partially computed results. Heterogeneity in node design has altered this model somewhat, but the fundamental approach has remained similar. If heterogeneity is more aggressively used and extended to support increasingly complex scientific workflows, then application developers must consider alternative system architecture baselines and exemplars. Such exemplars will be agnostic to low-level hardware specifics at first, but they will need to provide a method for reasoning about algorithm design and associated software behaviors. As the community explores the exemplars through codesign, more refined and detailed hardware simulation and modeling can demonstrate the viability and associated opportunities of each candidate to the application developer and software teams.

Standards-based integration and composition of heterogeneous or multi-vendor components. The vision of HPC systems utilizing much greater degrees of heterogeneity implies substantial complexity as these disparate components are interconnected. Whether the heterogeneous components are connected within the die, package, node, system, or across the data center, high-performance and efficient protocols will be required to connect them. In the event that components are supplied by a variety of vendors, which would be ideal for cost flexibility and reduction in design risk, open standards-based protocols with broad industry support will be markedly preferred. Examples of this approach include PCIe, CCIX, AXI, TileLink [43], CXL [17], and Ethernet. It is important for future HPC systems to ensure that evolution of these or other future novel protocols can support a diversity of hardware implementations and use cases while offering the highest levels of performance, energy efficiency, and implementation flexibility. In particular, research to extend these protocols across the data center or the full-scale HPC system remains a considerable challenge that is unique to the HPC community.

Application porting for codesigned heterogeneous HPC systems. To target the most recent system deployments within the DOE, application developers have spent recent years porting their code to performance-portable programming models, such as Kokkos [12], RAJA [13], OpenMP [44], OpenACC [45], OpenCL, and DPC++ [46], to name only a few. These programming models provide the ability to write a single body of code and have it execute correctly on a variety of computer hardware. Often, a very small amount of code (typically less than 5%) must be optimized for a specific piece of hardware. Future codesign systems could utilize this existing body of code development if the delivered hardware provides a sufficient general-purpose programming option for which these programming models can produce a good semantic match. Still, for some workloads, porting code functions at a higher level to use additionally optimized hardware units may be desirable and offer better performance. For example, if a given unit is optimized specifically to accelerate fast Fourier transforms (FFTs), then the application may gain the greatest benefit by porting its FFT calls to utilize the hardware unit rather than by attempting to employ its lower-level, programmer-developed code path. This approach is appealing because a significant number of linear algebra or scientific functions is ported to use interfaces such as the Basic Linear Algebra Subroutines (BLAS) library [47] or other packages, e.g., FFTW [48]. If accelerators are developed to map to these operations, then minimal application porting may be required beyond that already in development today. Where hardware functions differ from these interfaces, higher-level, domain-specific, and HPC-oriented languages may offer another alternative option. Such languages are not commonplace, so the HPC community must ensure they are developed to be broadly applicable and usable.

2.1.3 Opportunities

Enhanced system performance, cost, energy efficiency, and reliability. By definition, commodity processors and accelerators are designed to appeal to a broad range of systems and use cases. By amortizing the associated design costs over the large sales volume, lower component prices typically can be achieved. This has been evident for almost two decades in the HPC community as large-scale, massively parallel machines have utilized commodity processors to optimize aggregate system performance and cost.

However, the downsides associated with commodity-based designs stem from the cross-market nature of their implementation. To provide cross-market, multiple-user relevance, the commodity HPC designs make many trade-offs to provide performance on a selection of workloads or metrics. In short, these designs rarely provide maximum performance for any one specific workload. Instead, they aim to supply higher average performance across multiple workloads. This is not merely a performance factor as similar trade-offs must also be made for energy/power consumption, reliability, size, and weight.

The implication of these trade-offs is clear: if true codesigned HPC systems can be proposed, they can be more aggressively optimized to specific use cases in a user community. This allows hardware designers to walk back on the generality of the optimization to focus on more narrowly defined use cases. As such, the designs can focus on providing higher performance, increased reliability, or greater energy/power efficiency. At the extreme, designs may completely remove unused components, stripping the final hardware to a minimal set of functionality needed to run a given workload. The result can be smaller, lower-power designs or extra silicon real estate to enhance/increase the design's performance for a specific workload. Put more concisely, codesigned HPC systems allow the community to drastically increase the efficiency of the silicon by more tightly specifying the desired functionality.

Codesigned HPC compute nodes and systems have the potential for higher initial design cost because multiple multidisciplinary teams will be required to work collaboratively to shape and optimize the potential design earlier in its life cycle. However, the expected gain in aggregate system performance and capability is likely to exceed that commonly offered by commodity approaches. Therefore, much greater performance per hardware unit/dollar is possible with codesigned HPC systems versus commodity systems. For DOE leadership-class facilities and large HPC installations, funding profiles and project plans must adapt to have more consistent expenditure loading for multidisciplinary teams as opposed to drastic spikes in expenditures for deployments, which occurs in existing procurements.

Supply-chain management and reliability. HPC systems are critical components of DOE's scientific mission and make significant contributions to national security and economic competitiveness. Therefore, ensuring a flow of high-performance components for use in large-scale deployments is a critical aspect for managing the hardware supply chain that supports nationally important activities. Codesigned components or systems have the potential to help secure the supply chain by providing DOE teams with the ability to specify and design the most important performance components in these systems. If a trusted fabrication facility and set of packaging processes can be adopted, the security of the chips can be further enhanced.

At a minimum, having greater control over the component supply chain is an important aspect for providing continued availability of high-performance processors and accelerators. With additional research, these components could be codesigned to reduce the defects and security weaknesses used to gain unrestricted access to important scientific data, results, or application codes.

Support for novel and next-generation HPC workloads. Future codesigned HPC systems have the potential to enable entirely new classes of scientific workflows. This can be achieved in part through hardware and application optimizations to improve performance and scalability to such a level that previously intractable operations become possible. Examples of this approach may include significant increases in dataset sizes for AI/ML functions; tighter integration of processing and communication/storage to ensure that expensive data transformations, searches, and filters can be applied; drastically improved dataset access and queries; and vast improvements in online analytics or visualization that can be coupled to traditional HPC simulation workloads. In many cases, heterogeneous and optimized codesigned systems can enable these approaches by supplying step-function-like improvements in performance over existing solutions, which turns weeks of computing into operations that take days or hours. Such improvements are the difference in HPC being useful to scientists or too expensive to even contemplate. These are a few examples of how contemporary systems lack the tight integration that a truly codesigned solution could support.

Additional extensions of this opportunity may include changes to workload and workflow designs to require fully dynamic scheduling, provisioning, and mapping of resources. Existing approaches rely on a mostly static mapping of a single HPC job to a set of compute capabilities. In the future, a system-wide monitoring and resource management capability coupled to disparate compute resources could effectively map compute operations to the best available hardware by adapting the resource scheduling to the machine's current application and workload demands. Importantly, the main opportunity here would be large increases in resource utilization, which would lead to greater cost efficiency and higher aggregate system performance. Such changes likely will require disruptive changes to HPC application codes, which implies novel runtimes, algorithms, and system software components to support truly dynamic execution.

2.2 Codesign for Edge Computing and Processing at Experimental Instruments

2.2.1 Value Proposition

DOE operates a number of experimental facilities, including accelerators and light sources. Measurements taken at these facilities increasingly require substantial computational power for filtering and analysis as their data generation rates continue to grow at a double-exponential pace. Edge technologies (e.g., smart sensors) are becoming more widely used and eventually may work their way into core DOE usage models in distributed environmental sensor arrays. This development would mean avenues for codesign that have not yet been thoroughly explored involving experimental facilities, supercomputers, data storage, and scientific applications, which represent the complete end-to-end set of resources, including the edge. The resource-limited and distributed nature of edge computing implies this is another form of hardware that may interest DOE and can benefit from codesign of all system components—edge devices, back-end supercomputers, applications, and (where possible) networking. Here, the focus is not edge computing per se rather the entire continuum (to include the edge), which covers computing near the experiment, in-transit processing, and the HPC center.

Experimental and observational data are often collected at sites located a considerable distance from an HPC center, and the volume and velocity of experimental data threaten to overwhelm both the wide area network and the HPC center's ingest rates—even with upgraded network capacity. Just as each experiment operating today frequently benefits from custom sensors, specialized hardware, and advanced algorithms to gather the data, future experiments will gain from specialized solutions to reduce, analyze, and respond to collected experimental data in real time. Realizing this vision will require hardware deployed throughout the experimental pipeline to increase scientific throughput, automate control of experiments (no human in the loop), and reduce the burden on networks and HPC data centers. At the same time, these specialized analytics accelerators will support increases in the quality (i.e., resolution, sampling frequency, signal-to-noise ratio) of the data collected. Deploying commodity clusters or servers adjacent to experiments is useful for some workflows, but this is neither a complete nor scalable solution as commercial platforms may exceed power, space, and/or portability constraints. In addition, deploying a full custom solution for each experiment is undesirable because the cost to design, deploy, and support such "solutions" quickly becomes unwieldy. Field-deployable sensors optimized for science have different requirements than commercial devices in terms of reliability, resilience, and accuracy and often contain highly specialized functionality. Finally, as sensor data rates continue to increase, it can become impossible to get all measured data off of the physical sensor due to the package's physical constraints, namely I/O pin limitations. In this case, data-reduction processing must be integrated directly on the sensor die, which requires a robust hardware generation toolchain.

To address the unique needs presented by the diversity of experiments, a generalized framework must be developed to allow rapid design, prototyping, and deployment of workflows from edge computing devices all the way to the cloud and HPC center. As envisioned, ASCR researchers would work together with science teams to deploy sensor networks, so DOE-designed edge computing devices can be integrated directly onto sensors, in the field, in the network, and inside the data center. These devices can range from tiny distributed sensors deployed in the environment (e.g., Smart Dust for wide-area edge computing) to powerful FPGA or application-specific integrated circuit (ASIC) accelerators performing *in situ* data reduction and analysis for large-scale experiments with high-performance edge computing requirements. Using edge computing devices will help DOE realize the vision of ubiquitous computing seamlessly

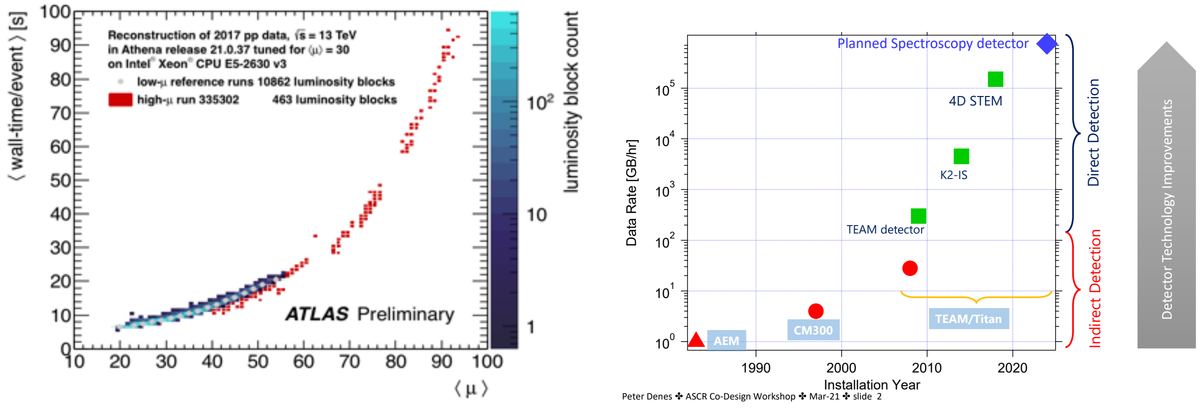


Figure 5: Compute requirements for the ATLAS detector at CERN (left) grow exponentially (y-axis) with increasing beam luminosity (x-axis). Likewise, the history of electron microscopy detectors and their planned successors (right) are expanding in performance at double-exponential rates.

integrated throughout complex scientific workflows. The creation of new algorithmic approaches and custom computing devices will be combined with commodity hardware and integrated into complex workflows as transparently as any software-based library, enabling non-experts to design, deploy, and utilize custom edge computing devices.

Answering these challenges will require new technologies for edge and in-network processing. These technologies will make it possible to preprocess and filter data online prior to sending it to an HPC center or the cloud for deeper analysis. Furthermore, on-site processing power will enable immediate feedback about the data quality of an experiment. Although the overall filtering approach (e.g., using *triggers*) is not new, it usually relies on highly customized and expensive one-off solutions. In contrast, DOE should consider a full range of data analysis pipelines, identify common analysis operations, and develop a highly adaptable hardware/software framework to move computing into the network of experimental and observational data facilities using FPGAs, GPUs, other emerging accelerators, and specialized computing technologies. The resulting modular/reconfigurable data reduction and analysis platform would eliminate the need for costly singular solutions and simplify access to the HPC center or cloud for data, enabling it to support a more diverse science area portfolio and increase its value to DOE. Furthermore, it will facilitate real-time data analysis pipelines during experiments, which would enable more efficient use of costly facilities. For example, at the National Center for Electron Microscopy, a return on a multimillion dollar investment in microscopes and a new camera has been hampered because data analysis and transfer capabilities lag behind the instruments' data collection rates.

Questions. The workshop participants were asked to address the following questions about the future of specialized computing capabilities for experimental facilities and their requirements:

- What are the largest challenges and research opportunities for codesign for experimental instruments?
- What gaps in our technology ecosystem must we fill to realize these opportunities?
- What will be the impact of these new capabilities if we are successful (i.e., what will we be able to do that is inaccessible today)?

2.2.2 Gaps and Challenges

Experimental data processing rapidly exceeding capabilities. Experimental facilities are experiencing double-exponential increases in data production rates from emerging detectors (Figure 5). For example, a single detector from an electron microscopy (EM) experiment can saturate the available bandwidth of the pipeline to a DOE HPC resource, and the number of EM experiments will likely grow in the near future (Figure 7). Similarly, in high energy physics, the computing requirements grow exponentially compared to the planned increases in luminosity for the next generation of the Large Hadron Collider (Figure 5). This would not be a problem if Moore's Law were still delivering exponential improvements to processing performance (as for the past 50 years). However, with the tapering of Moore's Law (Figure 6), improvements in data processing efficiency are commensurately lower.

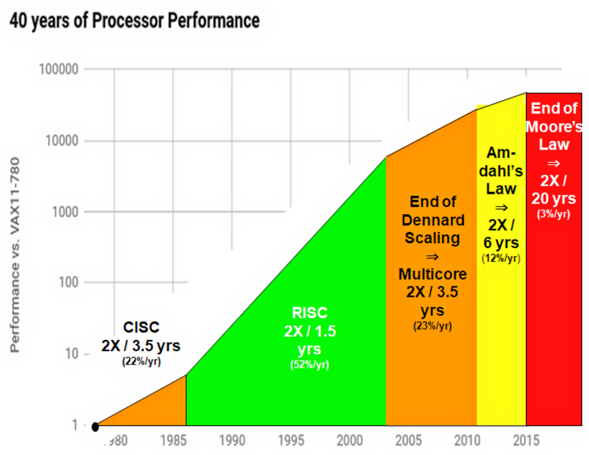


Figure 6: Over the past 50 years, Moore's Law has delivered exponential improvements in computing performance, yet this exponential growth rate now is beginning to taper off.

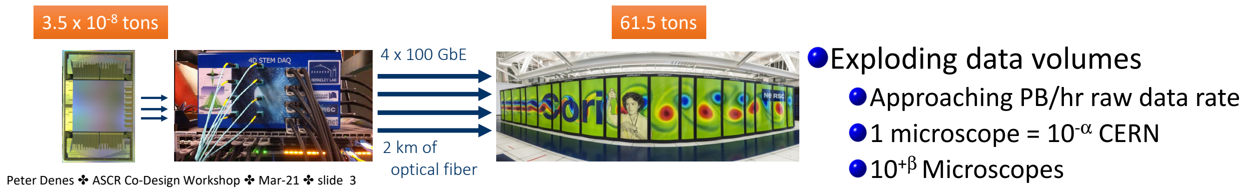


Figure 7: Current HPC centers are having difficulty keeping up with increasing data rates. A more streamlined solution for experimental sciences can be crafted using contemporary codesign processes.

Need for real-time analysis and feedback. For the past several decades, the experimental community (both the domain scientists and engineering teams that support them) has done the simplest tasks on the frontend and moved analysis far away from the experiment. However, this approach no longer scales and is insufficient. FPGAs and even custom hardware are increasingly employed to localize decision making and reduce the latency for making crucial decisions by moving the algorithms and math closer to the experiment itself. These algorithms have multiple goals, including data triage to determine what data are worth keeping and quick analysis and steering of experiments (e.g., CAMERA's gpCAM) using ML, fast computer vision, and analysis algorithms to extract key information from data as it is acquired. Denoising, sample placement, focus/calibration, and quick analysis/steering of the experiment are essential for efficient operation. For example, analyzing and adjusting the operating conditions (temperature and pressure) of a detector to compensate for environmental factors within the surrounding experiment is a common use case for real-time, *in situ* computing and analysis.

Sometimes, the issue is not raw bandwidth but, rather, reducing the latency for receiving the feedback needed to plan the next step in an experiment. In some cases, it is crucial for experiments to make decision within a few microseconds. For example, at ORNL's Spallation Neutron Source, having the detectors in beams on the operational side of the neutron source could damage the equipment. Deploying relatively simple logic by using an FPGA *in situ* with the experiment could solve this problem, and a push toward integrating more sophisticated ML techniques would be ideal. However, the experimental community lacks a framework or architecture for making these components and capabilities shareable and reusable across experiments.

Furthermore, when considering autonomous facilities as envisioned in the AI Town Hall report [49], the need for real-time feedback becomes even more evident. Traditional HPC centers—which are not often colocated with the instruments that generate the data—and their queue/allocation models are not equipped for real-time access during the experiment. The data must be analyzed immediately (i.e., in real time) to inform the next step in a given experiment, especially if AI is automating the selection of the next experiment. Data reduction to a region of interest and general data triage are key uses of near-detector processing. In the future, tight integration of models with real-time imaging/analysis will become even more important as the math moves closer to the instrument.

No one-size-fits-all solution. Support for a variety of experimental use cases and workflows is important at light sources and similar facilities. Edge computing performs initial data processing *in situ* or close to where the data are generated. This near-data computing provides data reduction, filtering, or even data-to-knowledge conversion to reduce backend processing requirements. However, the edge does not stand alone. The edge is part of a continuum that includes in-network processing, in-cloud processing, and reprocessing at the data center. All data processing pipeline elements must be coordinated and managed as a continuum (Figure 8). Even within the course of performing a single experiment on one beamline at a solitary light source, the workflows and required computational resources can vary immensely. Light source facilities already have put considerable thought into data acquisition to capture the intent of the experiment (collecting data from all data sources), but each pipeline still tends to be standalone in practice. Having seamless access to resources at the correct level, including moving data and workflows, would make the construction of these pipelines more efficient.

Reusable building blocks for hardware and software to support various use cases. DOE operates many different experimental apparatus, and, as stated, there is no one-size-fits-all solution. Nonetheless, there are opportunities to make the building blocks for these experimental pipelines modular and highly configurable to provide a greater opportunity to reuse them in multiple experiments. There also is a desire to make a diverse array of experiments programmable and maintainable on the experimental side in the context of a multiuser data and compute infrastructure that includes data analysis and simulation capabilities. Although this capability does not currently exist, available resources should be composable and expandable, so they can be redeployed in many experimental contexts.

Rapid prototyping of algorithms and hardware. DOE has unique capabilities at its disposal with experimental facilities and laboratories featuring their own ASIC design teams that can build custom CMOS or CCD sensors, as well as the means to support data processing electronics that meet the needs of its unique experimental requirements. This presents an opportunity to include specialized data processing capabilities within the sensor (i.e., active sensors) or near the sensor in cases of extreme environments (e.g., high radiation or cryogenic). This computing workflow is a continuum that spans from the edge where the sensor is producing data to the intermediate in-network computing and all the way to the cloud and/or HPC facility that archives and reprocesses the data. Exploiting emerging Python analysis and silicon compiler frameworks could enable prototyping algorithms in Python or C++ then porting key elements of that algorithm to custom hardware circuits in FPGAs. These accelerated circuits could be built onto custom silicon that is tightly integrated into the experiment. Deploying such a capability for experimental processing pipelines would be revolutionary, but no program exists to make this jump from proof-of-concept to impactful production and utilization.

Orchestration system across diverse hardware and software pipelines. The primary gap identified here is the integration of these different technologies into a coherent application. Many experimental pipelines analyze data concurrently from the same source. For example, for the National Synchrotron Light Source II (NSLS-II) at Brookhaven National Laboratory, the biggest issue is the wide variety of workflows. There can be up to 60 concurrent experiments within the same end station. Some will need modest computing, while others need HPC. However, the kind of computing depends profoundly on the pipeline's immediate needs and the underlying properties of the processing algorithms. In terms of experimental time frames, the processing pipelines may be up for a few hours or several days.

The pipelines must be orchestrated with diverse algorithms and hardware already used in practice, including FPGAs, GPUs, CPUs, and other emerging hardware such as coarse-grained reconfigurable arrays (CGRAs). Current one-off solutions do not offer a means to seamlessly switch between these different computing resources. The current state of the art for enabling this kind of configurable pipeline is to bounce data off of the disk (the experimental community is moving away from fast triggers). Ultimately, there is no one-size-fits-all processing solution, so there must be a modular and flexible orchestration framework to configure processing pipelines rapidly for experiments across diverse libraries of algorithms *and* their diverse hardware targets.

Control systems. A control plane that can be easily and efficiently updated to meet emerging experimental requirements is urgently needed. When looking at the software stack and what could be done to integrate it better with hardware, many facilities use EPICS for control/acquisition, coordinating I/O, and data management. However, EPICS is a control/acquisition layer for instrumentation, and, consequently, it does not provide the end-to-end, closed-loop control needed and is not designed for high-performance data processing. It would be hugely impactful to have higher-level tools and standardized (EPICS-like) interfaces that enable integration so anyone can use it. A hardware/software

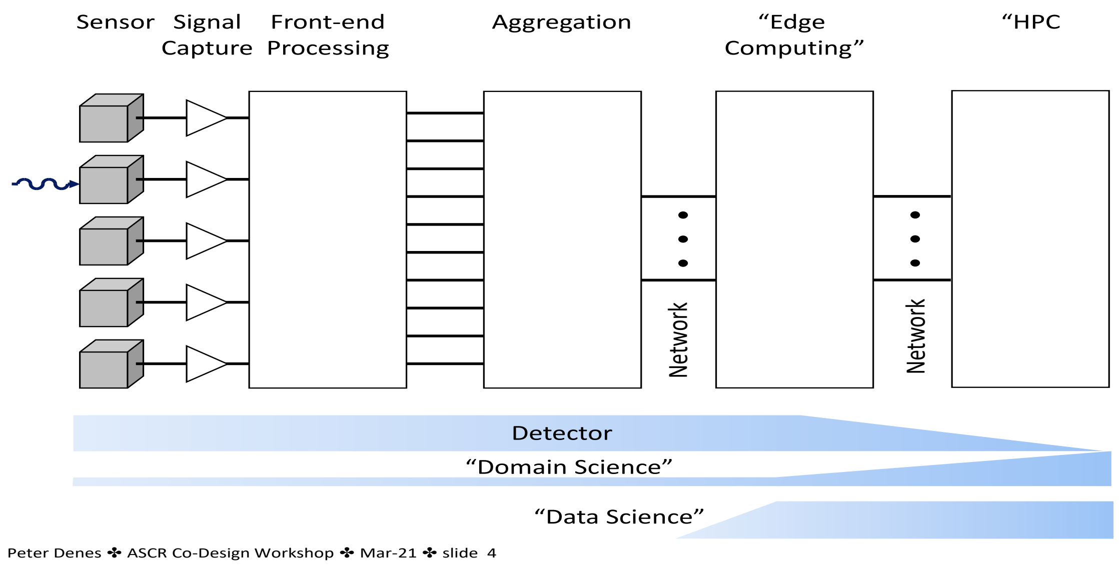


Figure 8: The edge is part of a continuum of computing workflows that span from the experiment to the HPC center. There is an urgent need to create frameworks that enable codesign across experimental pipelines using composable hardware/software elements.

wrapper that can expose these capabilities in a software-accessible solution is highly desirable. Furthermore, this kind of generalized control system and software wrapper for hardware services would benefit many facilities, experiments, and scientists.

Wide-area sensor arrays for smart grids. In many cases, hundreds or thousands of sensors reside at different buildings with controllers that are pushed toward the edge. These sensor arrays must analyze data in a distributed manner before aggregating it to an HPC facility because of limited communication resources in the field. For example, a sensor array could identify correlations between data streams to be more efficient in how power grids deliver electricity. Alternatively, an array of environmental sensors could *learn* the observed data, so they can deliver just the model to the HPC system rather than all of the raw data. The challenges are similar for distributed environmental sensors, where multitenancy in these smart sensors creates deep challenges for data provenance and security.

2.2.3 Opportunities

Composition framework for streaming pipelines. Softwarization, or software-defined everything *X* (SDX), is a paradigm from software-defined radio in which software wrappers for diverse instruments and computing systems are used to expose hardware capabilities and other resources in software (a service-oriented architecture [SOA]). This enables software environments to compose and orchestrate different pipelines made up of diverse hardware resources to monitor and control them via software, run analytics, and provide uniform/customized user interfaces. For example, this kind of SDX framework could be used to support a continuum of computations from the edge, cloud, or core as needed by the specific experimental pipeline by pulling the required resources from a broker (e.g., GPU) and assembling them into a coherent data processing pipeline.

The Priority Research Direction would be to develop a framework and set of software tools that enable experimental application developers to access codesign and implement applications to process streaming data—particularly embedding non-CPU resources into the experiment's data processing workflows. This is aligned with being able to mix and match software and mixed/diverse accelerator resources for processing pipelines of all scales. Tools and standardized interfaces enable flexible, composable, and interoperable workflows to meet the requirements for a particular experiment and allow for deploying these tools at the appropriate resource (i.e., from detector silicon to the HPC center and the cloud to everything in between).

Open-source hardware and software codevelopment frameworks and ecosystems. Open-source hardware programming ecosystems include high-level open hardware design tools and reusable hardware libraries (compression,

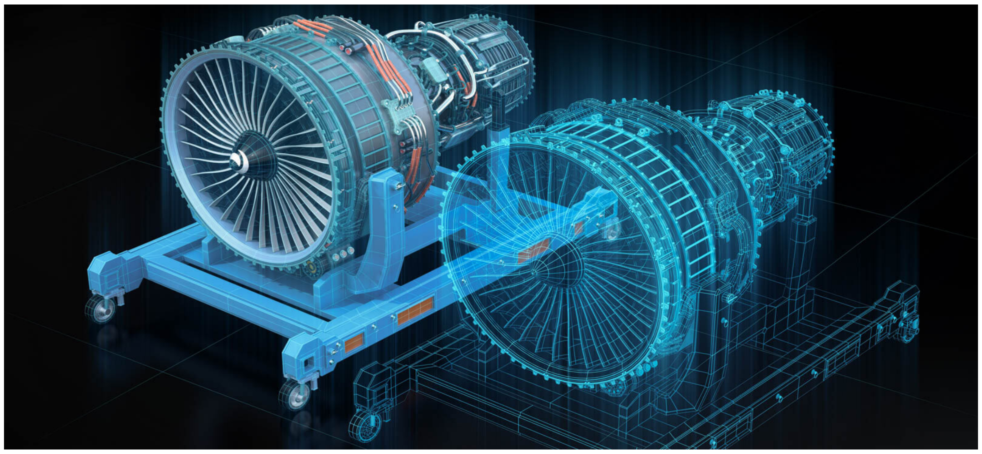


Figure 9: Digital twins combine real-time experimental data acquisition with additional information derived from models of the experimental system to gain added insights into the observations and underlying causes of observed behaviors.

encryption, filtering). For streaming, dataflow computing would greatly accelerate the development of a specialized data processing apparatus for future experimental pipelines. Algorithms expressed in Python must migrate efficiently to optimized compiled code, FPGAs, and even custom silicon. This fluid movement between a software specification and the data pipelines between specialized hardware components could be adopted productively by DOE's high-performance sensors community.

As the math/processing moves closer to the sensors, this new codesign process would require more engagement from domain scientists and applied mathematicians. The rigorous definition of interfaces can enable common compute acceleration for different data sources, which is a necessary feature for composability when targeting diverse experimental requirements as this segregates the customization from the modular interfaces. Codesign of a dataflow architecture for edge devices, software-hardware interfaces, and compilers would ensure greater reuse of development across multiple experiments. For example, open-source neural processing chips for AI model inference could be shared in all DOE experimental facilities close to the detectors/sensors. A better programming environment is also needed for making such dataflow processing paradigms more productive, which will be covered in more detail in Section 2.6 and Section 2.7 of this document.

Digital doppelgangers: Tight integration of models with streaming experimental data. Performing real-time numerical simulations in concert with receiving real-time data analysis from a live experiment is referred to as having *digital twins* or *digital doppelgangers* (Figure 9). Employing this method would enable a scientist to view a simulation of a model of a sample while also processing sensor data in real time to gain better insight into both the observations and the model that attempts to explain the physical mechanisms underlying what is being is observed. Another example might be real-time feedback from the model coupled with experimental observation to predict sample degradation.

Understanding how dynamic codesign of hardware and software advances and how computing accelerators can support this kind of model/experiment with real-time interactions are two significant research opportunities. Computing hardware and applications will need to reconfigure workflows dynamically for optimal performance in response to changing experimental conditions. This kind of dynamic codesign uses a model-based approach to coordinate among experimental devices and enable real-time reconfiguration decisions. Accurate performance models and monitoring are required for dynamic control. Frameworks that afford real-time performance analysis and modeling of edge/experimental workflows and algorithms via novel tools and proxy applications are needed to feed codesign efforts and match workloads to resources. The enabling technologies required for this kind of tight integration between experiment and model are a largely unexplored research space, offering enormous opportunities to revolutionize scientific understanding.

2.2.4 Crosscutting Issues

Fostering and funding multidisciplinary teams. For this vision to be successful, integrated teams must be formed, so software, hardware, applied mathematics, and domain scientists can develop a common vocabulary for communicating requirements. The domain-specific aspects will be managed by other program sponsors, but the interfaces need to be established. From a project management perspective, these other sponsor activities will be externalities, which the software-defined radio community has termed *software-designed laboratory* or *software-defined experiment*. The scale of the solution predominantly depends on the problem. As such, this could be evaluated by carving up the problem by scale and finding commonalities. It may be a question regarding the scale of the problem that is being addressed. From edge to cloud, in-network computing for distributed, real-time data processing requires a distributed computing infrastructure to tie together the distributed components—more hardware-like toward the edge and software-like toward the cloud. DOE used to have a robust distributed computing research program that could be reinvigorated with a distinct focus on integration with stakeholders at the experimental facilities contributing the requirements of their respective experiments.

Security crosscut with codesign. To support emerging edge computing use cases, methods for data provenance, attestation of any multitenant or third-party processing systems, and efficient end-to-end encryption of data throughout the pipeline must be addressed. Third-party hardware and multitenant isolation is common in IoT and data centers, but DOE environments differ somewhat. Integrating these services into an end-to-end execution environment (e.g., MPI, Kubernetes, or global address space) has not yet developed for these use cases. According to some observations, the combinatorial data explosion strongly correlates with data digitization at multiple levels. Security becomes a more concerning issue when using third-party components. Moreover, security must be thought of as spanning from the edge to the cloud/HPC center and everything in between, and it must include integration/mitigation strategies for untrustworthy entities within the ecosystem, owing to increased use of IoT devices in experimental environments.

Security issues on the edge, particularly in combination with third-party sensors and computation, is a growing topic in the public sector (see Confidential Computing Consortia [50]). Yet, mechanisms for data protection, multitenant isolation, and attestation of hardware and software will come at a cost—either in terms of software overhead or hardware real estate. Managing the communication bandwidth and compute throughput (speeds and feeds) for detectors *and* maintaining security are huge challenges. Secondarily but certainly related, end-to-end security absolutely should be a multiparty hierarchy (i.e., sensor $\rightarrow$ aggregator $\rightarrow$ gateway $\rightarrow$ ... $\rightarrow$ supercomputer) to assure that *trusted computing environments* can be carved out of workflow components that come from potentially untrustworthy third parties. These sort of pipelines differ from traditional HPC MPI or OpenShmem environments, and a good model is needed for those workflows, as well as streaming computation across them, that can then hook back into end-to-end security and multitenancy. This crosscutting issue is further addressed in the discussion of requirements and Priority Research Directions in Section 2.5 and is mentioned here because this crosscut is *essential* to achieve success of the edge computing vision.

2.3 Codesign of Memory/Storage Systems

2.3.1 Memory

Value proposition. Memory always has been critical to computing performance. Memory serves as the integral repository of application data (i.e., the current application state) for computation, as well as the staging for conveyance to other computing engines, such as parallel machine nodes or accelerators (i.e., GPUs). As computing engines have grown in speed and heterogeneity, memory increasingly has been the limiting factor and a key determinant of application scale and performance on HPC platforms.

In the past decade, new memory capabilities arising from novel bit-cell technologies (e.g., ReRAM) and the latest interconnections (e.g., stacked DRAM) have made serious bids to transform high-end and distributed computing capabilities. For example, ReRAM memories such as XPOINT [51] increase the available memory density (i.e., capacity in a physical footprint) and reduce the cost of memory (i.e., cost per bit; bits per watt). To date, these technologies have not reached widespread adoption, largely because of limitations in bandwidth, lifetime, and write energy. However, these persistent memory technologies have the promise to permit new flexible abstractions for computing over petabytes (or even exabytes) of data and blur memory storage distinctions.

Perhaps the biggest change in memory has been a revolution brought about not by a new bit storage technology, but via a novel interconnection (through-silicon vias) and packaging (die-stacking with solder bump arrays) to dramatically lower the energy required to transmit bits from memory to computing chips. The energy reduction along with thousands of I/O bumps and wires in 2.5D integration has provided a radical (10–100×) increase in memory bandwidth, e.g., HMC, McDRAM, and now HBM (high-bandwidth memory) and HBM2e [52–55], which has revolutionized high-end computing. Moreover, it is having a growing impact on AI/ML accelerators and will soon be in widespread use on CPUs [56, 57]. The surge in memory bandwidth has triggered innovation that produced a superlinear increase in computing performance, i.e., 50–100× (refer to Section 2.3.1). Notably, these benefits come at a cost: a radical reduction in capacity or increase in cost per unit capacity. *Novel memory technologies combined with new approaches for integration and packaging are essential to continue rapid expansion in computing capabilities across a wide range of applications.*

However, the tight coupling of stacked DRAM with individual compute chips has led to an increased fracturing of node memory (e.g., CPU, GPU, and other accelerator memory) into domains of nonuniform performance and addressability. Both the software abstractions (consistency, coherence, addressability) and interconnect performance between these memories have failed to keep pace. As a result, applications face difficult challenges in managing complex memory performance on DOE's leading computing platforms. Some examples include partitioned address spaces within a node (e.g., CPU versus GPU versus other accelerators), severe performance heterogeneity (producing performance cliffs with 10× drops in bandwidth), and low capacity-to-bandwidth ratios (1 GB/teraflop). These difficulties aggravate programming complexity and limit an application's ability to realize potential computing performance. With new variations of high-performance memory emerging, their critical contribution to HPC application performance makes codesign vital to continue advances in application performance.

A new and growing opportunity exists in open compositional memory frameworks, which federate memory subsystems horizontally (e.g., OpenCAPI, GenZ, and CXL [58–60]) to provide both convenient data access and rich support for data movement and orchestration beyond traditional coherence. The creation and widespread adoption of such open compositional frameworks cannot only ease programming, but it may provide performance opportunities by allowing systems to incorporate the best possible technology and avoid vendor lock-in through proprietary interfaces. This flexibility can significantly improve the system cost-performance ratio or capability at configuration time or through subsequent upgrades/extensions. As such, future productive memory systems will be ensembles of different memory technologies and packaging that are federated using rich, new open interfaces that support flexible composition of the best technologies available at system build time. *Memory composition frameworks for horizontal federation can increase system design flexibility by unlocking late-stage configuration choices that then unlock new system capabilities.*

With efficient data movement at the forefront, minimizing the number of bits and their movement has become the primary concern for designing power-efficient computations. Traditional load-store memory interfaces are limiting, and there are many rich opportunities for new memory functions (e.g., fixed-sets, configurable, and programmable computing in memory) that can yield 10×, or even 100×, increases in performance. These benefits not only stem from access to higher bandwidth, but also reduction in the energy required to perform a given computation. Categories of these functions include data movement (vertically within memory hierarchies and horizontally), data transformation (compression; encoding), and new explorations with ample programmability [61–67]. The performance opportunities are significant, but major challenges exist in effective control and high-level programmability. Early commercial examples (i.e., startups; limited features) are emerging, but codesign represents a path to develop these opportunities for large application benefits. *Novel in- or near-memory functions can deliver increases in performance by unlocking much higher memory bandwidth. However, substantial experience shows that codesign is essential for creating programmable and flexible capability through function in memory (FIM).*

Success stories. Research into memory and processing near memory (PNM) has started crossing into high-volume products and industrial adoption. Examples are listed as follows:

Stacked DRAM memories have been established as the mainstream memory paradigm in HPC, in the form of HBMx [55, 56, 68]. With lower energy per bit and ample interconnection density provided by through-silicon vias, arrays of solder bumps, and fine-wire pitch substrates, the novel packaging and integration of DRAM memories have enabled radical increases in computational performance. Recent breakthroughs include adoption as CPU [53, 56, 57] and node memory, large-scale success of systems that employ HBMx as a primary memory system [68, 69], and powering the world's fastest computer, Fugaku [56]. The 10×–20× bandwidth increases delivered by stacked DRAM

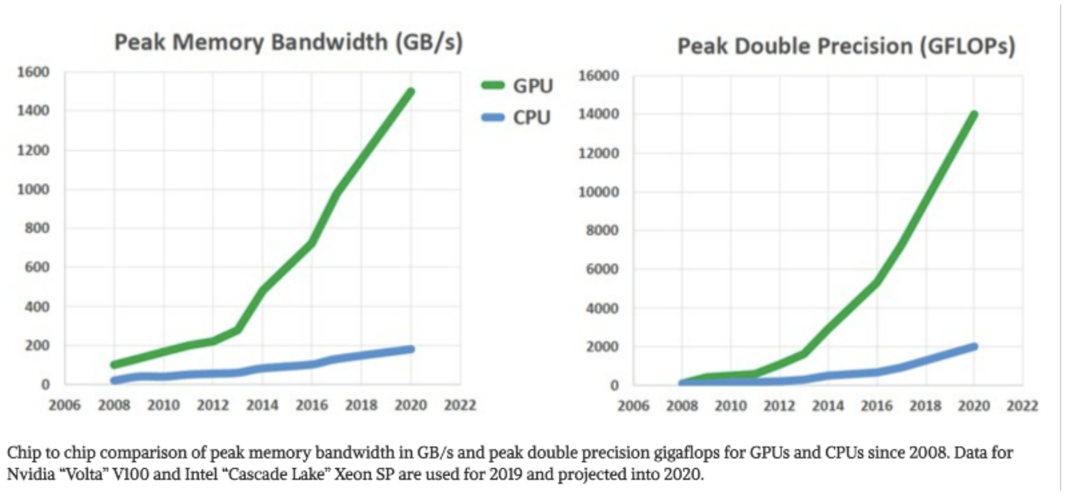


Figure 10: Memory bandwidth from GDDR and stacked DRAM has enabled GPUs to outpace CPUs in terms of peak teraflops for a single package. The NVIDIA A100 GPU achieves 19.5 teraflops in double precision on 2,039 GB/s of memory bandwidth [68].

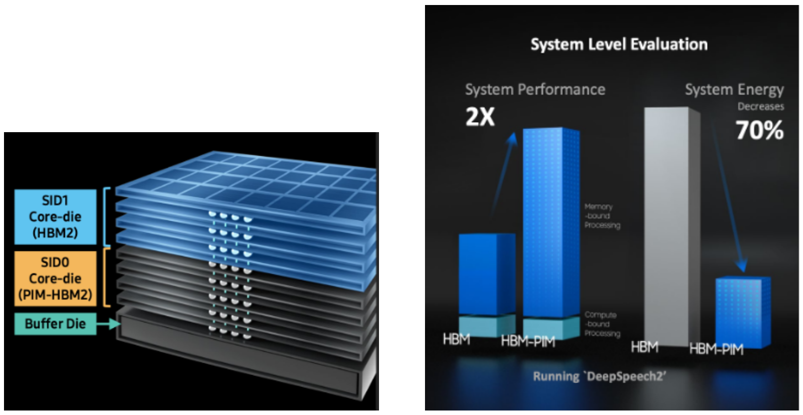


Figure 11: Samsung PIM in a stacked HBM (left) and the resulting performance improvement.

have enabled new computing models (e.g., streaming and task models) and architectures (e.g., GPUs) to achieve higher levels of computing performance and density.

Novel FIM and Processing in Memory (PIM) are emerging paradigms in the mainstream that include data movement and transformation support [68, 70] and build on a growing body of research. Driven by AI/ML requirements, a promising set of early products and research prototypes has added software-defined functions to memory. For example, Samsung's PIM integrates functions into the aforementioned stacked DRAM (HBM), leveraging the 8× higher memory bandwidth within the memory array for elevated performance. The Samsung approach places computation behind a standard DRAM interface [71, 72].

Facebook has integrated computation into RecPNM DIMMs to speed up the sparse-length sums (SLS), which is a critical sparse computation for inference. The computation, also placed behind a standard memory interface, enables up to a 5× throughput increase per node with the same stringent latency bound [73, 74].

Opportunities.

(1) Breakthrough performance is possible using **codesign of new FIM or near-memory features**, which exposes greater memory bandwidth and associated programming abstractions that resolve challenging problems in addressability, security, and locality (e.g., direction of intelligent memory hierarchies; near/in-memory processing).

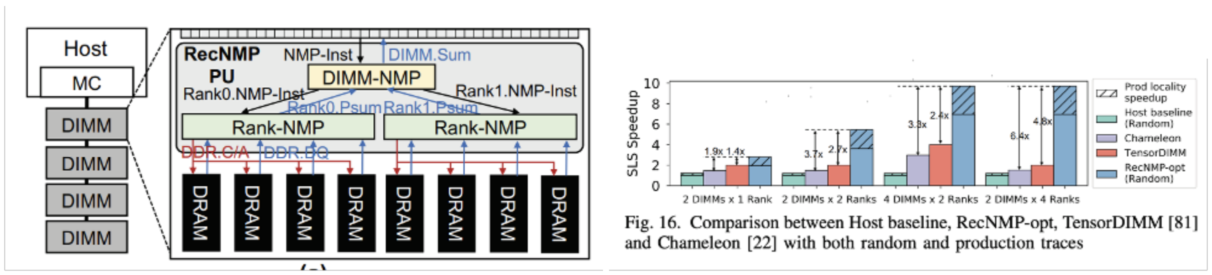


Fig. 16. Comparison between Host baseline, RecNMP-opt, TensorDIMM [81] and Chameleon [22] with both random and production traces

Figure 12: Facebook's SLS accelerator based on AxDIMM (PNM) [73, 74].

(2) **Novel packaging and heterogeneous memory technologies can dramatically increase key memory capabilities** (e.g., increase bandwidth, reduce latency, expand capacity). In particular, broad acceptance of addressable memory abstractions, such as CXL, GenZ, and OpenCAPI [58–60], are key for system designs that can be customized to meet novel HPC and DOE mission needs by unlocking the best system elements for the applications.

(3) **Shaping the features, runtimes, and programming systems to address high-performance access** would enable applications to more easily access full memory performance across the growing nonuniform and fragmented memory systems (e.g., direction of HSA and unified addressing) [75].

Current gaps.

(1) **Research in FIM and function-near-memory (FNM)** can unleash opportunities in memory bandwidth and data movement that are critical to breakthroughs and continued improvement in computing energy efficiency. Open questions include: *What computations can most effectively exploit memory bandwidth? How does shifting specific computations to be near memory affect desirable algorithms, application architectures, and data layout? What are the opportunities for codesigned algorithms and FIM/FNM, and what are the net benefits?* A marked focus on applications from HPC, edge, and scientific applications using ML is essential to create codesigned systems with crucial capabilities applicable to DOE missions.

(2) **Aggressive study, prototyping, and exploration of new packaging, interconnects, and memory technologies** can unlock new capabilities in memory capacity, cost, and bandwidth. Memory technologies with novel system integration can address critical system performance bottlenecks across all domains and have the potential for the largest, robust, system-level performance increases in the coming decade. Open questions include: *What are the new technology and packaging opportunities to increase bandwidth and capacity? Can they enable new capabilities? How does their integration affect application algorithms, computing approaches, and computational models?*

(3) **Exploiting the emerging flexible memory composition interfaces and interconnects** may enable flexible composition of memory in systems, which, in turn, would allow large-scale designs to employ late customization and best-in-class memory elements to optimize system performance and cost. Open questions include: *What composition interfaces are critical to enabling systems built from best-in-breed components? How can application software navigate, manage, and exploit composition efficiently? How can the composed system achieve larger-scale science?*

(4) **Additional research is needed regarding effective hardware and software codesign methodologies for memory across HPC, edge, and scientific applications using ML**. Creating appropriate benchmark workloads and methodologies is critical for shaping the increasingly customized architectures and components needed to meet challenging future DOE application requirements. Open questions include: *What are appropriate models and summaries of applications for these broader, system-wide studies? What scale (e.g., thousands of nodes) and duration (minutes, hours, weeks) are essential? How can efficient summaries be used? How can higher-level models be used to increase the scope of codesign exploration?*

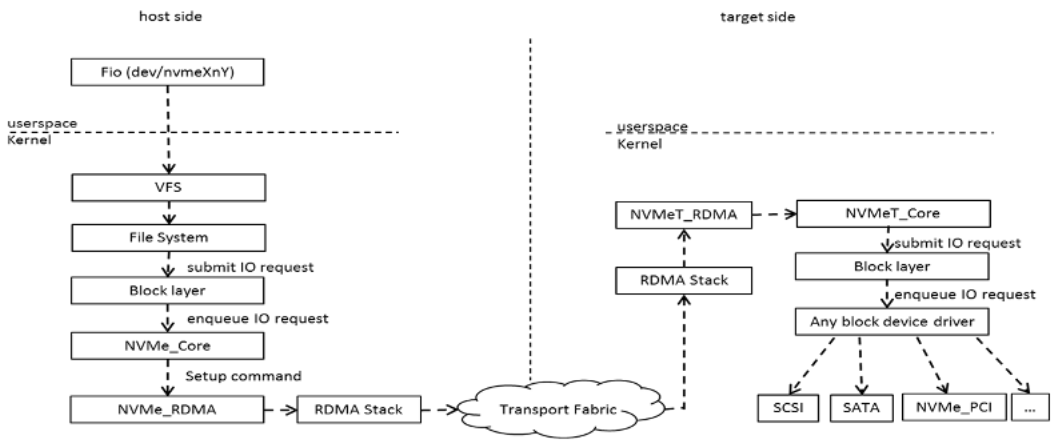


Figure 13: The NVMe-over-Fabrics stack [76] is one method to incorporate network-attached devices into distributed storage services.

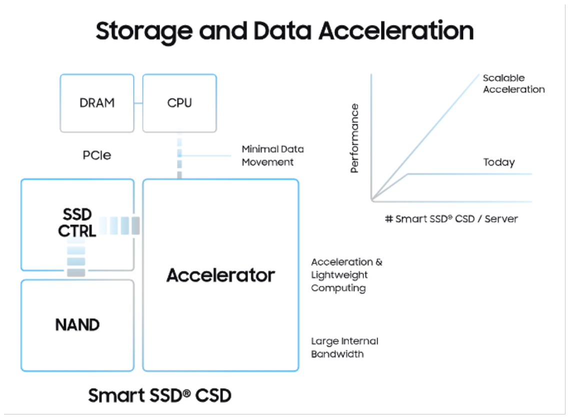


Figure 14: Samsung SmartSSD incorporates Xilinx computational capabilities into the device, which enables computing in storage and other forms of customization of device behavior and interfaces.

2.3.2 Storage

Value proposition. Storing data outside of compute memory is a key capability in DOE computing platforms for retaining valuable data over the long term, temporary data storage in long-running workflows and campaigns, and as a reliable backup location for partial results to mitigate the impact of faults during application execution.

The landscape of storage hardware and related networking technologies is rapidly evolving. Solid-state storage is widely used [77], hybrid devices mix solid-state technologies with DRAM memory [78] to create even more storage options, and new hard drive technologies are pushing capacity. All of these developments have created a range of storage device technologies that vary by orders of magnitude in bandwidth, latency, addressability, and cost. Similarly, the exploration of network-attached devices [76] has continued, resulting in devices that can participate in networked services without a traditional storage software stack to manage the device locally. This is a specific instance of a more general trend to incorporate greater compute capabilities within storage devices and expose them to serve in what traditionally have been operating system or storage software roles.

In HPC, adoption of the newest technologies often trails commercial cloud computing. DOE compute facilities have explored using dedicated solid-state storage file systems for bursts of data, while the latest-generation platforms incorporate solid-state storage in a more central role [79] (Table 1). The potential for hardware acceleration in the storage subsystem, however, remains largely untapped.

HPC systems traditionally have focused on the performance of compute, memory, and networking components. Generally, storage is not regarded as an area for innovation, or one where risks should be taken by the HPC facilities.

Table 1: Storage characteristics of upcoming leadership-class computing resources.

	ALCF Aurora				**NERSC Perlmutter**				**OLCF Frontier**			
	Archive	**Center**	**Platform**	**Embedded**	**Archive**	**Center**	**Platform**	**Embedded**	**Archive**	**Center**	**Platform**	**Embedded**
SW	HPSS	Lustre	DAOS	*N/A*	HPSS	GPFS	Lustre	*N/A*	*TBD*	Lustre	*N/A*	XFS
HW	LTO8 Tape/ HDD	SSD/ HDD	PM/ NVME		3592 Tape/ HDD	HDD	NVME			SSD/ HDD		*TBD*
Capacity	305 PB	200 PB	220 PB		230 PB	200 PB	35 PB			700 PB		*TBD*
BW	90 GB/s (cache)	650 GB/s	25+ TB/s		100 GB/s (disk)	500 GB/s	5 TB/s			10 TB/s		*TBD*
Usability	Medium	High (POSIX)	? (non-POSIX)		Medium (hsi, ftp, globus)	High (POSIX)	High (POSIX)			High (POSIX)		Low (per-node POSIX)

Note: Some of these resources have not yet been deployed. As such, configurations may change.

Especially for compute-intensive codes, the typical commercial parallel file system seems to be “good enough” and is familiar to system operators. Similarly, application teams on the computational science side often work to minimize time to solution in situations where storage performance is not the primary limiting factor. Thus, even in situations when a facility (e.g., the Argonne Leadership Computing Facility) is willing to bet on a new technology, e.g., Distributed Asynchronous Object Storage (DAOS), much effort goes into confirming it can be presented to users via interfaces that make it look like a commercial parallel file system. This is a stark contrast to the cloud, where the term *data center* reflects the importance of storage, data services, and rapid innovation to increase ease of use, capability, and efficiency.

Nonetheless, there is a growing emphasis on data-intensive and learning workflows in HPC that must change the balance and drive innovation in storage and data services, particularly as performance is often more dependent on I/O operations (e.g., search, random access) and less on the ability to hit some high-aggregate throughput (e.g., checkpoint, restart, or large array access). Cloud systems have been experiencing I/O-intensive workloads for longer as a larger fraction of their overall workload, which likely motivates exploration and adoption of new technologies.

Concurrently, an explosion in differentiated data services in commercial cloud computing has occurred (e.g., file systems, object stores, document stores, SQL databases, and streaming services) [80], including those customized to specific domains. These services also allow applications to mix and match and be customized to meet specific needs and provide higher performance to the workflows they serve. Perhaps more importantly, their customization to the domain affords higher productivity to workflow developers.

These changes in commercial cloud computing also have disrupted traditional notions of storage (i.e., hard drives and file systems) in HPC. Initial demonstrations have shown the feasibility of this approach for specialization on DOE systems [81–87]. Because data services are essential to productive data-centric science, flexibility should be allotted to emulate this broad approach by recognizing that DOE’s data services needs are distinguished in many ways—extremely large data volumes; high data rate; real-time, extreme bursts; and deep domain semantics, to name a few [88]. *Codesign of customized data services that focus on specific DOE mission needs and leverage the latest and most advantageous hardware technologies will be key to data storage keeping pace with developments in accelerated computing.*

Success stories.

(1) DataWarp burst buffer uses solid-state storage accelerators on the Cori platform. This employs a traditional file system model, but it is globally accessible and has informed plans for a solid-state lustre volume in Perlmutter [89].

(2) The DAOS object store, IO500, serves as primary application storage for Aurora and is pushing facilities to solve challenges in data translation between object and file modalities [90].

(3) MarFS archive provides a new storage tier, *campaign store* [91], which affords dramatically higher accessibility than the standard model, where data otherwise reside in a tape system.

(4) Mochi framework is used to develop numerous distributed data services and demonstrates use of common infrastructure to build customized services for DOE applications [92].

(5) Unify and similar tailored file systems accelerate specific workloads, including checkpoint, shared library loading, and AI training input [87, 93].

Opportunities.

(1) **Harnessing the exploding diversity and rapidly increasing capabilities of storage hardware technologies could create breakthrough flexible, customized data services to power data-centric science.** Codesign should be engaged to combine varied storage technologies in ensembles that power new efficient, scalable, and customized data services to meet the varied needs of current and future DOE science missions (HPC, edge, data-centric). This codesign must include prototyping and demonstration at scale with production application workflows and deployments. To do so will require engagement across communities and a full-scale research testbed exploited by researchers to prototype, harden, and support custom HPC and edge-driven data services.

(2) **Creating a strategic roadmap of data service requirements and key supporting hardware technologies explicitly for data-centric science would focus strategic investment.** Codesign should be used to create a roadmap of the data service requirements for future applications. That study could create storage motifs that expand the requirements for data services to support complex workflows, edge-driven applications, distributed data services, and more.

Current gaps.

(1) **Researching effective hardware and software codesign methodologies and interfaces is essential.** The block interface traditionally is the interface/abstraction presented to storage consumers. New technologies merit revisiting what these abstractions should expose and could facilitate codesign, as well as reveal other technical barriers not yet known or fully understood.

(2) **Storage function offload should be leveraged.** Along with exploring potential interfaces, new research is needed to facilitate migration of storage functionality that typically has lived in software (i.e., the operating system). This challenge is similar to offloading computation to an accelerator or FPGA. Understanding what and how to offload will be necessary for the successful development of codesigned systems. Additionally, new challenges, such as how to ensure access control in these environments, must be addressed.

(3) **Supporting a broader user community is essential.** Although some storage codesign examples have been explored for DOE applications on HPC platforms, deeper investigation of use cases at experimental facilities, the edge, and for data science use cases would open up the possibility for more widespread adoption of codesigned storage solutions within the DOE complex.

(4) **Hybrid systems are key.** As in the DAOS example described previously, realistic deployment of codesigned storage solutions will involve fitting them into existing environments. Understanding how to impedance match between fast, new services and other layers is a notable challenge to overcome.

2.3.3 Blurring the Boundary between Memory and Storage

Value proposition. Recent trends are changing how computing systems use and store program data, and performance distinctions between memory and storage are shrinking. With the emergence of heterogeneous memory architectures that incorporate high-capacity persistent memories, the volatile DRAM tier on many recent platforms is only about $2\times$–$3\times$ faster than the persistent memory tier [94]. Furthermore, their persistence represents new opportunities to exploit them for storage functions. Unfortunately, applications often lack the controls necessary to take full advantage of the features and capabilities of new technologies. For example, modern systems include configurations to access persistent memory through interfaces built for older technologies, that is, either as a byte-addressable memory device with volatile storage or a block device with durable storage. Such interfaces typically lack options for applications to control how different parts of their address space use the various features of the underlying hardware.

Simultaneously, applications that use supercomputing platforms have become more dependent on memory and storage devices to meet performance goals. In particular, the increasing use of AI and ML in HPC has driven demand for fast and large-scale data processing to new heights. Such data-intensive workloads have only exacerbated the challenges and importance of using data processing and storage resources efficiently. New memory and storage interfaces present a significant opportunity to increase performance across the execution stack, which would unlock new applications

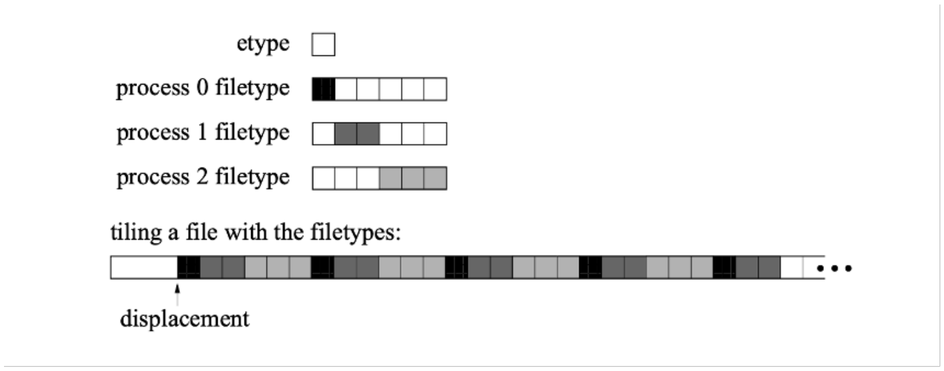


Figure 15: Partitioning a file among multiple parallel processes in MPI-IO avoids the shared file pointer limitations of POSIX IO. *Source: MPI standard.*

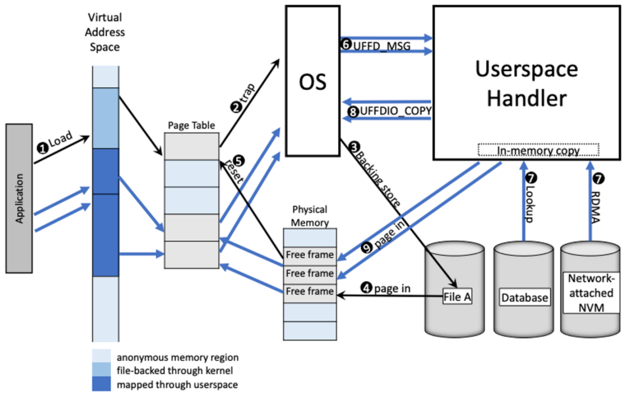


Figure 16: A user space memory map handler can exploit persistent memory features to access file system objects.

on ever-larger datasets that could power new scientific discoveries and technology breakthroughs. Legacy interfaces applied to new and emerging data technologies are impeding progress. If replaced, the new interfaces could unlock opportunities to increase performance, capacity, and power.

Most importantly, research for new interfaces should reexamine the role of various technologies and algorithms used in data management. Investigating application context can enable better data placement decisions to increase efficiency on complex memory and storage hierarchies. Another avenue is to revisit how software interacts with new and emerging data technologies and evaluate how different interface choices affect outcomes, *including performance, efficiency, flexibility, and compatibility*. Of course, participation across the stack is essential for success, and ASCR ReCoDe poses a unique opportunity in this regard.

Success stories. Preliminary and exploratory storage hierarchy-based topics have a renowned history within HPC research [95–97]. One of the advances provided by MPI-IO addressed a fundamental shortcoming with POSIX IO, namely shared writing to a file. By introducing a mapping of memory objects to file objects, MPI-IO avoids false-sharing performance penalties (Figure 15).

From a memory perspective, low-latency persistent memories play a dual role by enabling large datasets to persist beyond process lifetimes in file systems and allowing memory-mapped processing and analysis. As shown in Figure 16, user space memory mapping handlers offer flexibility and performance optimization opportunities to exploit novel persistent memory characteristics, such as customized block size and read/write asymmetries.

The current state of the art includes a diverse field of related topics, including unified interfaces (e.g., UNITY [98], MPI-IO [99, 100], Proc Topology Interface [101], UMap [102]), portable layer interfaces (e.g., hwloc [103], SICM [104, 105], disaggregated memory [102]), power awareness (e.g., tile-based network-on-a-chip [NoC] [106], automated power saving approaches, data placement and data movement (e.g., place trees [107], locality abstractions [108], hierarchical

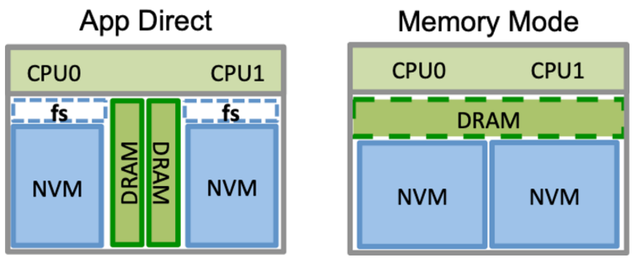


Figure 17: Two views of the Intel Optane device illustrate the blurring of memory and storage. On the left, NVM coexists with main memory as a separate memory or storage resource. On the right, the same NVM extends memory capacity transparently to the software stack *Source*: [51].

views [109], chunking approach [110], algorithmic distributions [111], deep memory run times [112]), very large datasets (e.g., non-volatile opportunities [113]), and increased heterogeneity (e.g., hardware approaches to extend the lifetime of flash storage and persistent memory [114, 115]).

Opportunities. New interfaces that span memory, persistent memory, and storage can unlock new performance opportunities by enabling applications to share context and intent, providing complex access and usage patterns and temporal structures, and enabling logistical data movement across memory and storage (e.g., workflows). These benefits all require codesign across applications, hardware, and system software.

If these new interfaces blur the interface between storage and memory, using a uniform abstraction can unlock new hardware and system software opportunities. Examples include automated, intelligent data movement; intelligent allocations that select the appropriate physical memory/storage type and location; and application portability across different platforms.

The largest benefits likely will result from a complete, bare metal-to-application replacement of the memory and storage interfaces. Key research questions include: *What are the missed opportunities incurred from adding another middleware layer instead of redesigning the stack? What gains can be realized by delegating data placement and data movement to lower levels while empowered with knowledge of application context? Given the blurring of performance tiers and usage semantics, what are the pros and cons of one unified interface for data disposition that covers the traditionally separate realms of memory and storage?*

Current gaps.

(1) Research should explore new interfaces that span memory and storage, as well as capture complex application behavior and intent and the implementations that exploit it. These interfaces may include complex parameters or even language-level constructs or *pragmas*.

(2) Exploring new codesign strategies that connect interfaces and implementations may create insights into what forms of behavior are exploitable and how to exploit them.

(3) Exploring customization for complex workflows (e.g., distributed, edge, or instrument) could add more context to the memory/storage/system perspective and help drive innovation.

(4) Exploring new methodologies beyond program traces, access patterns, and usage patterns could provide a wealth of information to application data structures (compilers) and storage metadata that may then be used to manage data across complex hierarchies.

(5) Exploring application-level dynamic allocators that understand memory latency and bandwidth could enable directed allocations in specific types of memory.

(6) Resources should be used to explore alternative physical memory management designs. For example, rather than a single memory manager, the operating system could provide a selection of memory management systems in the same way it provides a selection of file systems.

(7) Novel hardware (e.g., inspector monitors) could be explored to detect access patterns and, likely, memory regions to prefetch, migrate, or transform.

2.4 Codesign of ML, Neuromorphic, Quantum, and Other Non-von Neumann Accelerators

As discussed in Section 1.2, the industry is trending toward increased variety in computational accelerators. Early CPUs supported limited parallelism. Later CPUs increased parallelism through wider vector units, deeper pipelines, more threads per core, and more cores per socket but still largely optimizing execution for latency-sensitive operations. GPUs represent an early counterpoint to CPUs by optimizing execution for throughput-sensitive operations with far more numerous but individually much weaker execution units. While GPUs initially were intended for graphics processing, new accelerators specialized for different tasks are emerging, including ML accelerators, neuromorphic architectures, quantum computers, and other such devices.

Machine learning accelerators. Recognizing the computational challenge presented by training deep learning models, a number of specialized hardware platforms recently have emerged specifically for this computational task. The most notable example is the Tensor Processing Unit (TPU) developed by Google [116] to accelerate the training of TensorFlow models. There also has been a profusion of different accelerator startups, such as Cerebras [117] (wafer-scale), Mythic [118] (analog), Groq [119], and Graphcore [120], which have emerged to provide novel hardware solutions for training deep learning models. Sidestepping the issue of hardware implementations, cloud computing services, like Amazon Web Services's SageMaker [121] and SambaNova's Dataflow-as-a-Service [122], have started to provide all-in-one deep learning training and deployment solutions where users can be blissfully ignorant of the underlying computational requirements.

Neuromorphic architectures. In the late 1980s, Professor Carver Mead pioneered neuromorphic engineering using silicon devices to mimic biology. These were analog circuits that utilized subthreshold dynamics of CMOS transistors to emulate biological systems. Today, neuromorphic systems encompass both digital and mixed-signal approaches. Developments in large-scale digital neuromorphic chips have shown the promise of these systems at scale. The University of Manchester's SpiNNaker chip (130 nm CMOS) represents a more configurable approach with programmable ARM cores and an interconnect fabric optimized for spiking communication [123]. This platform is flexible in that it supports different neuron and synapse models. IBM's TrueNorth chip was the first neuromorphic chip with a million neurons [124]. Intel's Loihi is fabricated using 14 nm FinFET technology with 128 neuromorphic cores and an integrated, on-chip learning engine [125]. The SpiNNaker and Loihi architectures lend themselves well to scaling and are front-runners in the race to provide a billion neurons. The million-ARM-core SpiNNaker system aims to simulate a billion neurons, and Intel recently announced the Poihiki Springs system with 100 million neurons [126]. Plans to build the next generation of SpiNNaker2 chips using 22 nm FDX CMOS are currently underway [127]. Both systems support on-chip learning, are configurable, and have a dedicated software stack to program the hardware. These systems also support research communities, which is key to the adoption of such emerging technologies. Other approaches in analog and mixed-signal CMOS chips include large-scale, mixed-signal integrated circuits (ICs), such as DYNAP-SEL [128]; Neurogrid [129]; and analog CMOS floating-gate-based reconfigurable approaches, like GT's learning-enabled neuron IC [130] and field programmable analog arrays [131]. For a detailed overview of analog approaches, refer to Thakur et al. [132]. For a comprehensive survey of neuromorphic approaches, see Schuman et al. [133].

Quantum computers. Quantum computers employ quantum effects to perform certain computations asymptotically faster than would be possible using *any* classical mechanism. Instead of classical bits, quantum computing is based on quantum bits (qubits) that take advantage of superpositioning, entanglement, and quantum parallelism plus amplitude cancellation to achieve a performance advantage over classical computing.

Other novel computational accelerators. Many other innovative technologies currently are being explored [134]. These technologies include digital annealers [135–137], memristive circuits [138, 139], dataflow engines [118, 122], polariton and photon condensates [140, 141], injection-locked and coupled laser systems [142, 143], coherent Ising machines [144–146], and biological computers [147, 148]. These technologies can be categorized broadly into two groups: (1) digital computations built on CMOS architectures that implement specific, well-known algorithms in hardware and (2) analog computing systems where the bulk of the computation is conducted by a physical system and digital computing is used only for hardware programming and reading results. These technologies span a range of technology readiness levels, and many require significant research and development before becoming practical. At this time, digital annealers, coherent Ising machines, and memristive circuits are among those with higher technology readiness levels and are benefiting from significant industry investments.

2.4.1 Crosscutting Issues for Novel Computing

Codesign of hardware, software, and algorithms is especially important for novel architectures. Hooker has coined the term *hardware lottery* to reference the notion that the algorithms and methods which gain traction are the ones best suited to currently available hardware and software [149]. Conversely, many good algorithms and methods are disregarded by the community because they run poorly on existing systems—even if they potentially could exhibit superior performance over the alternatives on an architecture codesigned with the algorithm/method. Hooker highlights that hardware is expensive to design and fabricate, which necessitates an extremely conservative approach to architecture. Thus, hardware is designed to accelerate *only popular* software patterns and tools *at the expense* of novel ones. Codesign offers the opportunity for applications to benefit from more radical software and architectural changes. Instead of developing algorithms/methods and hardware in isolation, devising innovative algorithms and methods along with the hardware that optimally runs them implies that a larger number of innovative ideas will catch on as they do not have to risk losing the hardware lottery.

Key needs for non-von Neumann architectures include simulators, codesign benchmark problems that are hardware-agnostic, and more generic benchmarks that do not target a specific set of applications. The science of workload characterization is important. Without it, it will be difficult to discern if progress is being made. Capturing the architecture workloads also will aid codesign for assessing performance, functionality, and accuracy. Often, these metrics are not correlated, and all of them are needed for codesign. Programming languages pose another limitation for novel computing paradigms as they tend to focus on the device and architectural challenges. Still, the biggest barriers to technology adoption are compiler and system software. To integrate a diverse set of devices in a chip (or node), the system software is critical.

As non-von Neumann architectures continue to mature, compilers must change dramatically to leverage the strengths of these technologies. Figure 18 presents a simplified version of existing mainstream compiler technologies with one shared IR and two target instruction sets, one for the CPU and the other for the GPU. To take advantage of non-von Neumann architectures, new hardware-agnostic instructions (e.g., linear system solving, sampling, optimization) and IRs must be introduced to enable application code portability across a diversity of hardware platforms, including quantum processing units (QPUs), neuromorphic processing units (NPUs), and accelerators (illustrated in Figure 19). The precise specification of these abstraction layers is an area of active research and likely will require many years to develop a solid synergy with the emerging hardware platforms. The emergence of multi-level intermediate representations (MLIR) [150] for AI workloads highlights that this compiler transformation is already underway.

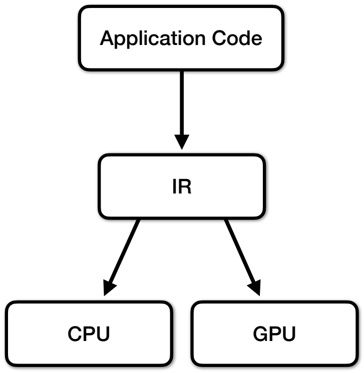


Figure 18: A simplistic illustration of established compiler information flow.

Another important question to ask is: *How can scientific discovery be accelerated*? Executing scientific calculations in near real time (instead of hours or days) will completely change how scientific discovery is done. The process may be overconstrained using traditional von Neumann architectures. Algorithms may be overlooked because they do not fit the von Neumann machine but might be practical with novel computing paradigms. Codesign is a substitute for the roadmap to keep innovation moving at a faster pace. Every application cannot be built as an ASIC, and successes from one application must be translated to another seamlessly. AI-enhanced codesign tools are another means to accelerate codesign for scientific discovery. Although still a relatively new area, recent work [151] has successfully applied deep reinforcement learning for chip floor planning to design Google's next-generation AI accelerators, and this effort shows

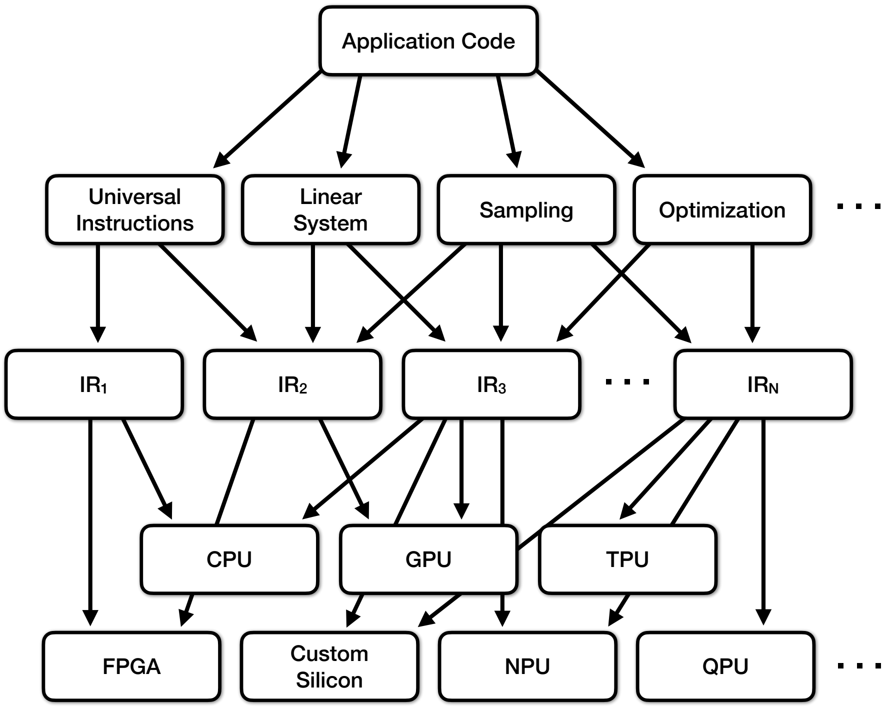


Figure 19: The anticipated increase of compiler complexity as non-von Neumann hardware continues to mature.

real promise. Memory (storage) and data communication are primary architectural bottlenecks that negatively affect the efficient implementation of algorithms. To overcome these constraints, novel memory technologies are being developed. However, they are not compatible with current foundries.

To surmount these challenges, the ability to design and demonstrate chips must be more accessible. Reusable IP and structures that can be ported to different applications and domains will be important. The DARPA 3D SoC program currently supports monolithic 3D integration with opportunities for shuttles. Neuromorphic computing, for example, does not separate memory stores from compute (floating gates and memristors). Perhaps, 3D architectures will develop further and change the landscape. Still, it will require re-envisioning and codesign of the entire design stack. The right proportion of non-von Neumann and von Neumann architectures is difficult to determine. Multiscale problems are well suited to these approaches—with ML at one level and HPC simulation at other levels. This requires tools that are easily extensible to different domains and compiler infrastructure to ensure compatibility. Successful projects will require a highly interdisciplinary effort along with experts who are familiar with the entire design stack.

2.4.2 Value Proposition

The central value proposition concerning all of the technologies discussed in this section is *they have the potential to solve certain computationally time-consuming tasks faster than general-purpose hardware platforms*—in the case of quantum computing, asymptotically faster. In most cases, they require less energy to perform the same task, which implies they can be deployed in very-low-power contexts, such as off-grid and edge computing use cases [135].

2.4.3 Success stories

To date, there are relatively few success stories involving *codesign* or conventional hardware or software with novel computational accelerators. The closest examples are in the ML accelerator space. Software abstraction layers such as TensorFlow [8] and PyTorch [42] provide user-friendly, domain-specific modeling layers for deep learning. Recognizing the computational challenge presented by training deep learning models, a number of specialized hardware platforms have emerged specifically for this computational task, often directly targeting TensorFlow and PyTorch. Again, the most notable example is the TPU developed by Google [116]. In short, the existence of widely accepted software abstraction layers has facilitated hardware innovation in the deep learning space. By designing hardware against a known abstraction model, hardware vendors are all but guaranteed a market for their products if they provide a superior implementation of that model. In the HPC codesign space, hybrid computing models pairing Cerebras's CS-1 with Lawrence Livermore National Laboratory (LLNL)'s Lassen supercomputer [152] provide a real-world testbed for how an AI-accelerated supercomputer could be designed.

A codesign success story involving quantum computing is Intel's design of the Tangle Lake quantum processor, specifically with a surface code [153] in mind. Surface codes are a well-known quantum error-correction (QEC) algorithm. By architecting Tangle Lake around a specific QEC algorithm, Intel's intention is to implement error-corrected quantum algorithms more efficiently and scalably [154, 155].

Discrete optimization, particularly quadratic unconstrained binary optimization problems [156], is a widely applicable and well-studied computational model. Fujitsu's Digital Annealer [157], coherent Ising machines [158], and D-Wave's quantum annealer [159] all natively support this computational model but with different trade-offs in performance, scalability, connectivity, maintainability, and cost. Many programs have been adapted specifically to D-Wave's hardware, which provides sparse inter-qubit connectivity and limited qubit count while still offering the ability to sample a large number of solutions per unit time.

2.4.4 Opportunities

Because all of the technologies discussed in this section are new and their capabilities and applicability to scientific computations of interest to the DOE's Office of Science are, for the most part, poorly understood, there is much opportunity for codesign to help them deliver on their promise of faster and/or more energy-efficient computation. As these novel technologies mature, they most often can be scaled to especially compact packages [146] that can be colocated near traditional CPUs, reducing latency and presenting opportunities to offload highly specialized operations to these unique computing models. However, because the computing models proposed by these hardware technologies generally are not well studied, redesigning software to take advantage of their distinctive operations will present a significant challenge. Some broad opportunities for codesign that span multiple types of novel architectures include (as follows):

How can problems of interest to DOE be partitioned both in terms of data—most computational accelerators provide substantially less memory than a conventional CPU—and function—what computations would be beneficial to offload to what accelerators? Codesign opportunities exist here in balancing the expression of a scientific problem against the set of high-level operations on which various accelerators provide a quantitative advantage.

What is an appropriate level of abstraction for a computational scientist to target in order to exploit novel computational accelerators? If the level of abstraction is too low, DOE applications sacrifice portability. If the abstraction level is too high, it may not be possible to exploit the advantages offered by different accelerators. These abstractions will need to be reified in software. Then, this software will best be codesigned with application software and, if possible, the underlying hardware.

How can DOE applications articulate which codesign trade-offs they are willing to make with the underlying hardware? These presumably would fall into categories of "more" versus either "faster" or "more richly interconnected" for the tensors, neurons, qubits, dataflow blocks, spins, or other units of computation appropriate to a particular accelerator. A similar "more" versus "faster" trade-off involves on-chip memory capacity. An even trickier trade-off for applications to express is "faster" versus "more correct." Many novel accelerators provide low-precision or only integer numerics, while others provide results nondeterministically and with no guarantee of correctness/optimality. For the former, an application needs to specify the precision it requires or prefers, possibly increasing iteration counts to compensate for lower precision. For the latter, an application must specify the number of samples to take or, in an annealing model, the rate at which execution progresses. The ability for applications to declare their computational requirements would be an important step in hardware/software codesign.

Opportunities also are available that are more specific to certain technologies:

Machine Learning. The emergence of novel and specialized hardware for ML workloads has greatly expanded codesign opportunities. Cluster designers have the choice of integrating specialized ML accelerators into individual cluster nodes or centralizing the ML workloads in separate computing platforms. The two-phase nature of ML workloads (i.e., training and deployment) presents distinct options for leveraging entirely different platforms for training (e.g., via cloud computing) and deployment (e.g., on premises). Across a large computing platform, the ideal mixture of ML accelerators, GPUs, and CPUs is unclear and will vary based on the types of workloads executed. Consequently, additional research is required to develop the novel ML-enhanced proxy applications and performance modeling to guide the codesign process.

Neuromorphic. Next-generation neuromorphic circuits and systems will need to balance the trade-off between scalability and biological complexity. Novel approaches in fabrication, such as 3D architectures and wafer-scale technology, as well as in-memory computing devices, could further mitigate current communication and connectivity bottlenecks. This will require synergistic collaboration across devices, architectures, software, and algorithms.

A modular approach to codesign tools also is important, especially tools that enable integration of new architectures and device characteristics. Digital accelerator tools have democratized the ability to test and validate different dataflow architectures. The neuromorphic field needs these open-access codesign tools to be available to the larger community that supports varied backends. This could involve analytical tools, cycle-accurate tools, and tools that enable the integration of neuromorphic accelerators with conventional processors.

Quantum. Because quantum computing is a nascent technology, it offers numerous opportunities for codesign. As industry works on scaling up the number of qubits, e.g., PsiQuantum's quantum computers based on photonics [160]; scaling down the physical size, such as Quantum Brilliance's quantum computers based on nitrogen vacancies in diamonds [161]; and improving reliability, like Microsoft Research's quantum-computing approach based on non-Abelian anyons [162], these systems will become more amenable to integration with traditional HPC systems. However, discrete HPC applications or components of a single application may favor different quantum technologies. Codesign among application developers, system integrators, and hardware developers can inform the choice among a larger number of qubits shared across the entire HPC system, a smaller number of qubits available per node, or a modest number of more reliable qubits—along with how applications and workflows should adapt to any of those choices.

2.4.5 Current Gaps

The technology readiness level of these diverse technologies remains the principal challenge to codesign. The hardware and low-level software often are in such a nascent state that it may prove difficult to evaluate the potential improvement the technology might deliver even if it was codesigned with application software and HPC middleware. Hence, much time and labor may need to be invested in codesigning with a novel-computing platform only to discover that the technology offers less benefit to the scientific computing than initially perceived.

Another codesign gap involves hardware uniqueness. While codesign is easily applicable to optimizing the number or size of some hardware component, new codesign approaches must be developed to handle fundamentally different computational models. There is a qualitative difference between codesigning an application with a conventional processor, i.e., one may be able to customize, for example, the number of floating-point units or hardware threads or cache entries, versus an unconventional processor, where the units of customization typically differ from one technology to the next and devices are optimized for some specific function that may be a force-fit to HPC.

Software tools are key to codesign. However, tool development requires deep understanding of the underlying platform and involves substantial work per platform. The novel technologies discussed in this section tend to be one-offs that differ radically from each other, and even their basic architecture may not be publicly disclosed. This impedes the creation of tools that facilitate codesign activities.

Benchmarks and metrics also are necessities for codesign. Due to the varied computational paradigms involved in novel computing, it is difficult to readily define appropriate benchmarks and metrics, for example: *Would a given application favor more spikes per second in a neuromorphic processor or more pulses per second in a superconducting quantum processor*? Research is needed to identify even the most basic terms to describe application performance of radically different underlying hardware.

In addition to the preceding gaps that are general to novel computing technologies, the following technology-specific gaps are also worth highlighting:

Machine Learning. ML accelerators are arguably the most mature accelerators discussed in this section. Still, the optimal design for clusters that feature both ML accelerators and traditional processors remains an open question. Specifically, depending on the ML application under consideration [163], the optimal hardware composition can vary greatly. Fortunately, the abundance of ML workflows built on industry-standard software provides a copious collection of applications for testing novel hardware designs. Exploration of accelerator designs has ushered in a new "golden age in computer architecture" [164]. A spectrum of computer architecture design tools has emerged to facilitate research into these new architectural options, which range from analytical tools to cycle-accurate simulations. Analytical tools provide estimates of hardware performance by computing energy and latency-related metrics based on data movement across the device coupled with known hardware behaviors. Example analytical approaches include Modeling Accelerator Efficiency via Spatio-Temporal Resource Occupancy (MAESTRO) and Eyeriss [165, 166]. Other analytical tools focus on assessing properties of a hardware architecture, such as resource utilization and identifying an optimal dataflow strategy for the architecture. An example is the Timeloop tool [167] developed by the Massachusetts Institute of Technology and NVIDIA. Cycle-accurate tools offer increased fidelity and sometimes couple with executable hardware-description-level simulations. Examples include the Systolic CNN AcceLErator Simulator (SCALE Sim) and the NVIDIA Deep Learning Accelerator (NVDLA) [168, 169]. These tools largely focus on ML accelerator approaches such as systolic arrays and convolutional neural network accelerators.

Neuromorphic. Although significant progress has been made in codesign methodologies and their adoption, a major gap exists in the integration of novel computing paradigms, such as neuromorphic architectures in heterogeneous computing. In part, this is due to the lack of a cohesive codesign toolset, as well as the diverse nature of neuromorphic backends, which range from digital, analog, and mixed-signal to beyond-CMOS approaches. Furthermore, novel algorithms are required to leverage the unique properties of neuromorphic systems.

Quantum. Codesign involving quantum computing is uncharted territory. Different quantum computers are based on specific gate sets (analogous to a classical instruction set), provide varied numbers of qubits, arrange their qubits in distinct topologies (which leads to different cost trade-offs), offer assorted robustness to noise, exhibit diverse numerical precision, and run at extremely divergent speeds (differing by orders of magnitude). Codesign involving one type of quantum computer conceivably would be largely inapplicable to another. Finding meaningful ways to generalize and describe performance and quality trade-offs in a quantum context is an impediment to including quantum computing in a codesign framework.

2.5 Codesign for Security and Privacy

2.5.1 Value Proposition

DOE's Office of Science invests in a world-class scientific infrastructure, including computational science capabilities, leadership computing capabilities, and unique experimental user facilities integrated with the Energy Sciences Network. Given the tremendous challenges the nation faces from cyberattacks on infrastructure, data breaches, and IP theft, ensuring that DOE's enabling infrastructure is secure and reliable and associated scientific datasets and personal information are protected is critical. ASCR supports DOE's science mission by stewarding the research and development of capabilities in computational analysis, algorithms, mathematics, computer science, and advanced architectures to support world-class research capabilities. The next-generation infrastructure to enable such capabilities most likely will be driven by complex interdependent edge computing and advanced wireless networking along with the aim to make scientific instruments more autonomous, which unfortunately can increase their overall vulnerability. These activities and the advanced modeling and simulation (ModSim) tools built using their data rely on scientific integrity and reproducibility, as well as the integrity of instruments—all of which presume the security and integrity of the underlying infrastructure.

2.5.2 Success Stories

Confidential computing and isolation mechanisms. Despite the hardware security flaws discovered in recent years, including Spectre, Meltdown, and variants of those attacks, there also have been significant advances in hardware-based security solutions, including isolation mechanisms within the hardware. These developments have come about in, at least, three distinct ways. The first is memory protection keys (MPKs), which are a hardware approach in recent CPUs that combines with virtual machine functions and extended page table switching to provide lightweight kernel isolation [170].

The second development is the advancement of capability-based systems, such as the Capability Hardware Enhanced RISC Instructions (CHERI) project [171], which is funded by DARPA and Google and developed by a team from the University of Cambridge and SRI International. Here, capabilities are defined as *unforgeable tokens of authority* that can be used to securely implement specific functionality. CHERI uses capabilities to implement pointers in C and C++ to enhance memory protection. CHERI has been implemented in multiple CPUs and a full software stack, including compilers and operating systems. Additionally, CHERI has been implemented in QEMU, on FPGAs, and in the ARMv8-A architecture used in ARM Morello processors, SoCs, and boards.

Confidential computing is the third area that has garnered attention in recent years, particularly with the use of trusted execution environments (TEEs). TEEs have no standard definition but are generally portions of certain microprocessors that provide a stronger degree of isolation between processes than has existed previously by using some combination of hardware mechanisms and software application programming interfaces (APIs). Examples of commercial TEEs include Intel's SGX (Software Guard Extensions), AMD's SEV (Secure Encrypted Virtualization), and—debatably—ARM's TrustZone. They also include the recently announced (but not yet available) ARM Confidential Computing Architecture and Intel Total Memory Encryption along with RISC-V-based approaches, such as MultiZone, Keystone, and Penglai.

Each of these TEEs has important distinct traits in terms of trust model, usage model, performance, and features. For example, Intel SGX enables computing within an enclave in a way that does not require trusting the operating system, other programs on the system, or even the system administrator. It encrypts the memory in the enclave and protects against many physical attacks targeting the computing system. On the other hand, it limits enclave memory to 64 MB, thereby precluding its utility for most HPC tasks. ARM's TrustZone requires a trusted operating system and does not leverage encrypted memory, but it does enforce strong separation of processes between the secure and the insecure worlds. Both TrustZone and SGX rely on a model that requires modifying the source code of programs executed within their environments. AMD's SEV enables entire virtual machines within the enclave without having to trust the host operating system, the hypervisor, or the system administrator. It gives the enclave access to the full encrypted memory and does not suffer the performance penalty that SGX does for most scientific applications. However, SEV assumes a weaker threat model for integrity attacks, particularly physical attacks, and requires trusting the entire guest operating system, which is a very large trusted computing base (TCB).

The major cloud providers have adopted isolation approaches. For example, Google's confidential computing offers access to both SGX and SEV systems, Azure's confidential computing provides access to SGX systems, and Packet.com features bare-metal access to SEV systems. Amazon also has developed its own alternative approach to hardware-enforced separation for Amazon Web Services in the form of Nitro Enclaves. Nitro Enclaves are distinct from TEEs in their functionality and use model but provide similar properties. The exception is it is necessary to include Amazon itself in the trust model, which is at odds with the TEE ethos of not having to trust the system administrator or the operator of the computing environment.

These new isolation mechanisms, including MPKs and TEEs, have clear benefits [172]. Yet, despite the important advances in all of these technologies in cloud environments, it seems clear that at least the current instantiations of these approaches all have weaknesses for both end users and developers in terms of their performance and usability for HPC. For example, all of the systems currently require either programming modifications or virtualization, which means a trade-off between developer usability and performance. None explicitly support multimode communication—except for SSL over TCP, which is very slow. Nothing supports secure remote direct memory access, and all hardware security support is CPU only: GPUs, FPGAs, and domain-specific accelerators are not supported. That said, mismatches between the threat and trust models in commercial and HPC applications could exist, so if TEEs are retooled, it may enable better optimization of security, performance, and usability for HPC. [173]

Attribution and data integrity. Although there has always been a need to track the provenance of scientific artifacts (e.g., source code for mathematical models and simulations), the increasing move from ModSim to data-driven science enhances this demand significantly, especially when data are collected from disparate sources (e.g., telescopes or global sensor networks) rather than a single, central source (e.g., the Hubble Space Telescope). New requirements assuredly will arise that will necessitate a confluence of solutions crossing cultural, secure hardware/software (as discussed earlier regarding process isolation), and algorithmic (e.g., distributed and fault-tolerant data) issues.

2.5.3 Opportunities

Isolation mechanisms. The presence of new hardware isolation mechanisms in modern CPUs provides significant advances in kernel protection. The adoption of CHERI's capability system in ARM Morello processors demonstrates significant progress toward real-world usable capability-based systems. Finally, given the presence of numerous commercial TEEs and their adoption by cloud providers, it is evident that the confidence and momentum exists for leveraging TEEs to improve security. Moreover, given the availability of ARM and RISC-V, particularly, the existence of open-source TEEs (e.g., Keystone), opportunities exist to research methods for addressing the weaknesses of current commercial TEEs using custom architectures—including heterogeneous ones—independent of Intel and AMD. All of these advances reflect new isolation mechanisms in one form or another. Yet, open questions remain about their applicability to HPC. Ample options exist to adapt these techniques, or others like them, in a way that is optimized for HPC.

Test hardware. Not long ago, experimenting with processor design was limited to high-end microprocessor companies, such as AMD and Intel. However, a variety of new methods have opened experimental opportunities to the masses. This encompasses ARM Morello; the open-source hardware movement; the license-free RISC-V instruction set architecture; various boards that support experimentation with RISC-V (e.g., HiFive Boards from SiFive [174]); and the common availability of FPGA hardware, including cloud-based systems such as FireSim [175]. All of these resources offer new chances for codesign.

Simulation tools. The gem5 simulator [176] is "a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture." It is a cycle-level simulator that enables rapid and accurate analysis of a diversity of architectural elements, including numerous traditional CPU architectures, GPUs, and RISC-V—including the RISC-V based Keystone TEE [177]. Historically, gem5 has been used for performance analysis, but it also has been engaged to analyze a variety of security vulnerabilities and attacks, including Spectre and Meltdown [178]. The success of using gem5 to examine these vulnerabilities combined with its increasing capability to include even more HPC system elements make it an invaluable tool for exploring design space trade-offs.

In addition to software-based simulators, FPGA-based emulation platforms also play a role to enable scale-up of simulations through FPGA-emulation. Hardware emulators such as FireSim [179] can boot full-fledged operating system environments and even simulate scaled-up clusters at near real-time speeds. Although there are trade-offs between using test hardware (e.g., FPGAs) and simulators (e.g., gem5), it does seem apparent that experimentation with hardware can be substantially faster. Gem5 can make experiments simpler than having to begin with the register-transfer language (RTL) or Verilog to test some simple new ideas, but scale-up using hardware emulation can also play a role. As such, additional development and experimentation of secure HPC architectures leveraging gem5 and FPGA-accelerated hardware emulation should be pursued.

Computing beyond traditional processors. Traditionally, processing happened in CPUs. Today, processing occurs in many more places, including in smart NICs (network interface cards) or smart storage. This poses both a challenge and an opportunity. On one hand, this increase in computing expands the attack surface. On the other hand, the increase also could create new options for codesigning protections now that intelligence and programmability are present when previously they were not.

Moving from malicious activity detection to adaptive control. Historically, much effort has been spent detecting system anomalies, including those in applications or application behavior, data movement through a network, or user behaviors. In general, the goal of anomaly detection is to alert human users (possibly through some hierarchy that also involves automated systems at low levels), but this response time does not result in robust operations in the face of threat activity. Research and development is needed to detect and identify signatures of malicious activity [180].

Future systems presumably will shorten the response loop by incorporating adaptive control that makes decisions based on the perceived system state. In contrast to existing approaches to dynamism that operate blindly (e.g., address space layout randomization), this approach to adaptive control would be intelligent and realized through a combination of system observations; ML or other automated reasoning to optimize response; and actuation that may include reconfiguration, changes in isolation, or other agile responses. This degree of dynamism marks a significant departure from today's security models, which often rely on protecting data or systems. In the adaptive control view, the processes themselves will be protected, and the data or physical systems are built using an ephemeral construct. Clouds, virtualized containers, and other execution environments are already obfuscating the meaning of a system in favor of an unambiguous understanding of the processes running on those systems. The codesign requirements presented by security realized in such a fluid environment differ greatly from those driven by static environments.

2.5.4 Current Gaps

Skilled workforce: National laboratories have cybersecurity expertise not traditionally engaged in HPC. National laboratories have long had both operational cybersecurity and HPC expertise. However, in the research and development space, with limited exceptions, the two have not traditionally worked together on codesign enhancements to secure HPC. This gap has existed since, at least, 2007 when a workshop [181] on the subject was convened, and a subsequent report was released by scientists about approaches to transform cybersecurity research and development within DOE [182]. This effort continued between 2008 and 2010 with the DOE Grassroots Cybersecurity Initiative [183], which resulted in several key reports [184, 185]. It renewed in 2015 when two ASCR workshops were held on Cybersecurity for Scientific Computing [186, 187]. Although these past efforts did not result in creating sustained programs to enhance the workforce in this space, one of the many important changes since 2015 has been the revolution in computer architecture, including GPUs, FPGAs, ARM-based architectures, domain-specific accelerators, and open-source RISC-V. A considerable opportunity exists to tap new graduates from the burgeoning systems and architecture communities to realize the potential of these new technologies.

Outside of DOE, the National Science Foundation (NSF)'s Office of Advanced Cyberinfrastructure stood up Trusted CI, the NSF Cybersecurity Center of Excellence. The organization has since developed security design patterns for research, including the Trusted CI Framework [188], a tool to help science and research organizations establish and refine their cybersecurity programs, and the Open Science Cyber Risk Profile [189], which is listed in a variety of NSF solicitations as a requirement for principal investigators to address in successful proposals. This type of concrete documentation can serve as a valuable resource along with encouragement and mandates from sponsors to ensure this guidance is put into practice. Trusted CI also has a successful fellows program [190] designed to expand the cybersecurity-capable workforce through a guided, hands-on training process. Given the increasing collaboration with the academic community, a partnership with Trusted CI could be invaluable toward expanding DOE's posture and workforce.

Application performance versus cybersecurity. Historically, there has been an unyielding *MIPS or bust* cultural ideology in HPC. This ideology may be hindering development and adoption of improvements in cybersecurity that would reduce computational performance (even slightly). Understanding the degree to which cybersecurity performance can affect computational performance is an important and open question that must be answered. At the same time, any conclusion other than *no effect* (which is unlikely) is likely to result in, at least, *some* need to overcome the cultural challenge of application versus cybersecurity performance. How to approach this ideally also is an open question.

Trusted computing base. *What is the trust model, threat model, or TCB for HPC, and does it differ from cloud computing*? Certainly, the usage model is different. Clouds rely on shared, virtualized resources, whereas users usually have dedicated access to the bare metal of the entire compute node in HPC. *So, how does this change what potential solutions look like*? DARPA's CRASH (Clean-Slate Design of Resilient, Adaptive, Secure Hosts) program explored clean-slate design of computer systems to enable adaptive maneuvering, execution of code under partial degradation, and adaptive auto-configuration of systems that learn from prior threat activities. Recently, DARPA CASE (Cyber Assured Systems Engineering) focused on technologies for establishing trust for embedded systems exposed to threat activities. This program also explored adaptivity and auto-reconfiguration to create more nimble, agile systems.

Different levels of trust defined at various stages of the design process. In the context of high-level synthesis or design automation, what trust capabilities can be described at the hardware or Verilog specification level? Are there layers in security that will provide different levels of trust at different stages of the design process? Foundational research may be undertaken to formally describe the scope of trust that can be imparted at different layers. For instance, at design time, a system's operating state may be specifiable so that the system maintains a particular trust posture. However, at design time, one does not have full awareness of the system's context. Hence, there are limits to what is knowable from a trust perspective. At the other end of the spectrum, at runtime, a great deal of additional context is available to a system (e.g., its user's trust level, other applications running on it, and the network environment in which it is operating). It may be possible to leverage this additional information into a runtime trust posture that relies on more abstract design-time trust posture features of the underlying system.

2.6 Democratizing Codesign with Open-source Hardware

2.6.1 Value Proposition

Since the adoption of commercial-off-the-shelf (COTS) cluster technology as a cost reduction measure in HPC, the commodity has traditionally been the chip, which usually is proprietary and monolithic. HPC-specific innovation typically happens at the board or system level. The availability of proprietary but licensable hardware IP (e.g., products from Arm and Cadence) has enabled third parties to create their own customized SoC designs from these discrete building blocks. Although cost barriers have diminished by sharing expensive development and verification costs across larger markets, licensed IP still can be too expensive to be viable in the low volumes needed for experimentation. Even with low-cost academic licensing, results cannot be openly shared among a broad research community owing to their proprietary nature. The emergence of open-source hardware/IP (e.g., RISC-V) and silicon compilers (e.g., OpenROAD) offers a path to democratizing hardware design by reducing licensing costs for design and accelerating hardware development. With open-source hardware, noncommercial entities, such as laboratories and members of academia, can participate in hardware design and development and publish results that are more easily verifiable and reproducible. This approach has the potential to revolutionize the manner in which the research side interacts with HPC vendors.

The potential role that open-source hardware can play in reimagining codesign touches on several key areas:

Enabling closer interaction with HPC suppliers through the exchange of shareable hardware artifacts that are unencumbered by IP issues, International Traffic in Arms Regulations, or Export Administration Regulations. Open source could play a role in communication by providing models backed by actual taped-out hardware artifacts that validate the models. The models provide the theory, but open-source hardware also allows DOE to verify the results experimentally.

Lowering barriers to using FPGA emulation to evaluate larger/more complex applications. Industry already uses this practice for the same purpose. Open hardware IP blocks (e.g., RISC-V cores, open NoCs, open designs for network interfaces, and memory controllers) are an enabling technology for these kinds of high-performance, cycle-accurate emulation technologies, which are capable of modeling full applications at system scale, for example, FireSim for system-scale simulation using synthesizable RTL.

Filling gaps in the licensable hardware portfolio, especially features or IP blocks that have a disproportionate impact on HPC. Licensable IP has lowered the cost of SoC development significantly by promoting IP reuse, but not every IP will garner interest for an entity, like Arm, to license. Just as DOE has done with past investments in HPC, following an 80:20 rule is recommended, where 80% is commodity driven by a larger market and HPC can focus its attention on the 20% that makes the biggest difference for DOE mission applications. Rather than targeting the motherboard with these innovations, the emergence of in-package (chiplet) [4] integration and democratization of design through mixed licensable and open-source hardware enable innovation to be integrated at the package-level (the new *silicon motherboard*). Furthermore, emerging die-to-die interface standards (Open Compute Project's Open Domain Specific Architecture [5] and the UCIexpress standard [16]) to support an open marketplace that allows chiplets from multiple vendors into a common package offer a new potential economic model for delivering HPC-relevant specializations.

Engaging experimental systems with high data rates. Such systems need more specialized hardware in the detectors and equipment than DOE is designing, and modular open-source hardware elements can be reused in many different experimental pipelines.

To tighten the codesign loop for evaluating realistic design, an open-source approach can enable more effective guidance to vendors via an exchange of hardware artifacts that have been synthesized and demonstrated/verified experimentally. Open-source hardware can facilitate rapid commercial hardware development by allowing companies to use existing open-source IP to accelerate the path from research to implementation and provide for mixing and matching designs that can be shared. DOE can accelerate this work by developing open-source implementations of existing open standards and interfaces for mixing open-source and proprietary IP. This kind of agile development environment may be further enhanced by supporting the creation of open-source software that can generate fundamental communication blocks, such as Serializer/Deserializers (SerDes); analog devices, like Phase Locked Loops (PLLs); Analog-to-Digital Converters (ADCs); and Digital Signal Processors (DSPs), to allow reference implementations of base standards. In the same way that open-source standards such as TCP/IP, HTTP, and TLS allowed applications to build value quickly by combining existing software over standard communication channels, developing simple, flexible standards for hardware IP interoperability will allow open-source hardware IP to develop paths into existing commercial products. In turn, it will allow commercial IP to provide new hardware capabilities, accelerate its time to market, and reduce the overall cost to DOE.

Currently, DOE shares proxy applications with primary HPC vendors and funds them to do codesign with DOE input in the form of advice and the proxy applications. The problem with this approach is the applications and hardware are fundamentally changing due to emerging extreme heterogeneous integration. There is a need for tighter/faster/more-integrated interactions with DOE's HPC suppliers to better codesign systems for its mission-critical applications. Application proxies can miss crucial interactions/features, and there are too many of them to mount a practical effort for full coverage. Because of the dangers of information leakage between vendor interactions, DOE ends up funding redundant codesign with each supplier, which spreads resources very thin (i.e., for six vendors, up to 6× more effort is required).

Open-source hardware offers a new way to interact with suppliers "in the open," which is crucial for allowing full engagement from DOE's scientists and subject matter experts. DOE can operate a tight internal codesign loop and share results *and* the artifacts in the form of open simulators backed by open-source hardware/RTL prototypes (e.g., FPGAs or taped-out hardware). Vendors can recreate these experiments using their internal tools—potentially mixing open models with closed internal models and RTL. Additionally, the DOE can share concrete artifacts that run actual code (e.g., emulation platforms or taped-out prototypes) to provide evaluation of more realistic applications rather than simpler proxies.

Lastly, DOE does design its own hardware for advanced sensor apparatus using a mix of licensable IP and custom circuit designs. There would be a huge benefit to DOE experimental detector designers if they had access to a library of open-source components curated by the DOE that could be reused across projects. For example, rad-hard (radiation-hardened) variants of RISC-V microcontroller cores or other electronic subsystems would be extremely valuable for numerous DOE experiments operating in high-radiation environments. Specialized applications also could be preserved for future generations of related use cases. History has shown that specialized engineering designs kept closed are often lost to time. By developing them using an open approach, these designs can more easily evolve to meet changing needs in the future and be reused across multiple generations of hardware, maximizing their overall value.

Questions. Workshop breakout participants were asked to consider many questions during the session, and this document tries to capture the input from the session participants.

- What is the potential role for the emerging open-source hardware ecosystem? Is open-source hardware a viable alternative to current IP, or does it only serve as a sandbox to try out ideas to prove to mainline vendors?
- How do open-source simulators keep up with the exploding accelerator design space and stay relevant?
- What is the role of open source and open evaluation of hardware ideas? To try out new ideas? Act as a second vote? Something else?
- What amount of validation of open source is needed to be useful? What should be done about validation?
- How do we make open hardware and FPGA hardware emulation more accessible so codesign is more productive and accessible to the masses?

2.6.2 Opportunities

Tighter codesign with HPC suppliers with fewer legal encumbrances. Proxy applications alone limit the ability to engage in productive codesign conversations with HPC system suppliers. A tighter codesign loop within DOE is needed to independently perform design-space explorations then openly communicate the results with vendors. Open-source hardware and IP components are required to allow DOE researchers to share more detailed artifacts without getting into legal entanglements associated with proprietary designs. Using this approach, DOE researchers can share a single open-source artifact with the vendors rather than conducting N separate experiments with the N proprietary IP environments that each vendor entails. With this, DOE could tighten the codesign loop for evaluating realistic design, which would enable more effective guidance to vendors. Open source also could be made interoperable with commercially licensable IP by creating open-source versions of existing open interface standards (e.g., CXL; AXI)

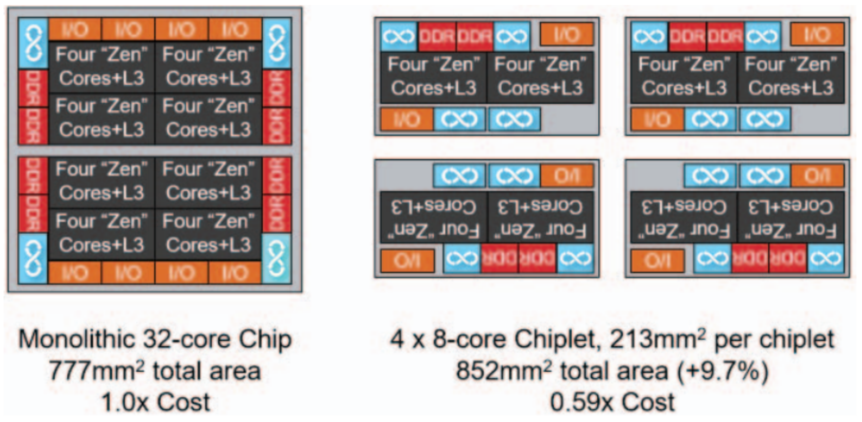


Figure 20: AMD has moved to chiplets to cointegrate different capabilities in the same package. Economic and technological justification of this approach can be found in AMD's article in ISCA2021 [29].

to afford mixing and matching of commercial and open-source designs. This would provide for broader sharing of IP components along with a path to commercial adoption for different use cases.

Another area where open-source IP could make a big difference for codesign is in accelerating the adoption of FPGA emulators by the academic research community. FPGA emulation is commonly used by industry to simulate full-chip designs (and even system designs) at high speed before tape-out, so the software and applications can be evaluated at scale prior to tape-out. Without a full application-level evaluation, the true impact of a proposed hardware/architectural innovation cannot be appreciated. However, FPGA emulation is difficult to use because of availability gaps for the hardware blocks, which are required for a complete design. This library of reusable open-source hardware blocks also will be beneficial for edge computing designs because they are doing real hardware design and using FPGAs for deployment, but the incomplete inventory of hardware components hinders productivity. There are complementary roles for simulators and open-source hardware artifacts to exchange crucial data with suppliers. Investment in open-source hardware will allow for productive use of emulators for edge computing *and* support more productive vendor-laboratory interactions for codesigning future systems.

New path to specialization for HPC with chiplet/co-packaging of open IP with commercial licensable IP—a new 80/20 rule. Since supercomputing's transformation from full custom machines to COTS-based clusters, HPC has continued to innovate on targeted subsystems (e.g., successive generations of Cray interconnects). This is a form of the so-called "80/20 rule," where 80% of the design is COTS and 20% is customized. This was done mostly at the board level, and there was no path into the processor itself because the cost of design and fabrication of full-custom ASICs is prohibitive, especially for leading-edge technology nodes. The embedded computing market developed a more modular approach to license the IP of an ASIC design's reusable parts (e.g., microprocessor cores and memory controllers). The commodity IP market (e.g., Arm) has addressed one of the largest parts of ASIC costs, but not every IP block is profitable enough to be prioritized for commercialization—even as licensable IP. Open-source IP blocks could fill in the gaps for HPC-relevant functionality, for which there is limited commercial interest provided, if their interfaces were made fully compatible and interoperable with licensable IP. Given that Arm and the rest of the licensable IP market have many existing open-interface standards (e.g., AXI, CXL, and AIB), there is abundant opportunity to mix commercially supported IP (the commodity for 80%) with open-source IP blocks for very targeted HPC-relevant functionality for which there is limited commercial interest.

Even with licensable and open-source IP to lower the prohibitive design and verification costs for new hardware IP, the mask and tape-out costs for a monolithic integrated design on a leading-edge technology node are also quite high. A recently emerging alternative is to cointegrate components into the same package using much smaller chips, called *chiplets*. At ISCA 2021 [29], AMD published its analysis of the costs and benefits of moving to chiplet-based heterogeneous-integration (Figure 20), and much of the industry is shifting rapidly toward this approach. Not only does it offer a much lower cost barrier to cointegrating IP, chiplets also allow IP components that use different technology nodes to be cointegrated. As such, some components could, for example, be fabricated using an older and much less costly technology node. This also has been observed by the broader industry and ASME-led HIR effort [3] as a potential approach to continue performance gains through new packaging technologies that lower the cost of cointegration. It also presents opportunities for public-private partnerships with the U.S. government's HPC suppliers to offer a path to specialization through integration of open-source chiplets from the government into proprietary platforms at a finer granularity and much higher bandwidth. Furthermore, emerging open standards for die-to-die interfaces between chiplets [3, 14–16] provide for packaging of licensable IP together with DOE-supplied IP. This emerging Open Chiplets Marketplace promises new opportunities for cost-effective specialization for HPC [5]

Intelligent sensors for DOE instruments and the edge. DOE already designs specialized hardware for detectors for its leading-edge experiments because the volume of its experiments is too low, and the requirements are too diverse for commercial vendors to support. Currently, these tailored experimental processing pipelines require long development cycles, and the hardware design complexity makes this prospect challenging for DOE's detector design team. However, at the hardware component level (i.e., IP blocks as opposed to chips), there are multiple options for reuse across experimental pipelines. Codesign of DOE detector designs could be transformed by creating a curated portfolio of modular, reusable, and open-source hardware components that can be arranged quickly for different experiments and will enable the Office of Science to realize its vision of ubiquitous computing that is seamlessly integrated throughout complex scientific workflows. This could reduce development cycle times and provide opportunities for more in-depth codesign of the overall hardware-software environment to support the experiment through improved component reuse. Such a portfolio would be beneficial for hardware descriptions that go into FPGAs, which are commonly used in DOE experimental workflows, or hardware that is integrated directly into the detectors. A curated repository of open-source components also affords the chance to incorporate novel computing devices, including neuromorphic or other non-von Neumann processors, to maximize energy-efficient data reduction and analysis. Creation of new algorithmic approaches and custom computing devices will be combined with commodity hardware and integrated into complex workflows as transparently as any software-based library, allowing non-experts to design, deploy, and utilize custom edge computing devices.

2.6.3 Current Gaps

Design verification and correctness testing. By far, the most expensive part of hardware design (whether open or proprietary) is actually the verification of correct operation. Traditional design verification techniques involve formal methods that are neither scalable to larger designs nor practical to run long enough to ensure full coverage of potential device defects. The challenge here is to develop systematic techniques and tools to design, verify, and optimize hardware components (e.g., chiplets) and their respective vertically integrated software components in a modular fashion for various objectives (e.g., power or performance). For example, emerging techniques such as QED (quick error detection) and SQED (symbolic quick error detection) show promise for pruning the test vector space, which is starting to make systematic verification more practical. Furthermore, there is an opportunity for crossover of these techniques to cover both verification of software *and* hardware correctness.

Curated library of reusable open-source components. The current portfolio of open-source hardware IP is incomplete and has varied levels of curation. Open-source components are scattered across the web with little structure. During the workshop, Tim Ansell, from Google, observed: "We need more memory controllers, coherency, [and] networks being open-source. We need them to be working on FPGAs and ASICS, and it would be great if DOE invested more in that."

Creating a curated repository for hardware design (akin to what Netlib did for numerical libraries) will serve the DOE experimental sensors design community, ensure the ability to share research artifacts when codesigning with vendors, and democratize the use of FPGA-based emulation systems and hardware prototyping for design-space exploration.

Enable agile hardware design through modular interfaces. Although multiple interface standards exist in the licensable IP market (e.g., AXI, CXL, and AIB), the licensing requirements of many implementations prevent customization for DOE needs and limit reproducing the resulting designs. Fully open-source implementations of these interface standards are needed to create a common ground for commercial vendors to support the DOE 80/20 approach for codesign at the ASIC level. This complements and simplifies the software abstractions if the underlying hardware is made more modular through promotion of open and interoperable interface standards across proprietary licensed and open-source IP. There is substantial existing work that can be built upon, including CXL, AIB, and Arm AXI. Too, there are many other areas (e.g., CCD sensor interfaces) that could benefit from extensions to those existing interfaces.

Programming abstractions. Programming abstractions are needed for accelerators, approximate/stochastic computing, non-von Neumann architectures, ML, and variable-precision methods. Although covered in more detail in Section 2.7, it is an important crosscutting issue that ties into all topics and open questions discussed throughout this report.

New mathematical abstractions that provide algorithmic descriptions, from applications to hardware. Techniques and tools must be developed for formal knowledge/semantics management within the same HPC layer and across different layers to enable accumulating, sharing, and reusing diverse and heterogeneous knowledge about scientific domains, software, and hardware. This may be an offshoot of the semantic-lifting and machine-programming effort already undertaken by ASCR in FY21, which could be significantly expanded if semantic-lifting and machine-programming were inclusive in terms of modifying the hardware with the software.

2.7 Tools, Software Stack, and Programming Languages for High-productivity Codesign

2.7.1 Value Proposition

Programming systems, libraries, runtime systems, and operating systems play critical roles in the successful deployment of effective and efficient applications on any computing resource. As such, this crosscut is broadly focused on programming languages, runtime and operating systems, libraries, performance and debugging tools, autotuning, and other elements of the software stack. In HPC, where performance and scalability are essential, the system software must provide a delicate balance of abstraction and efficiency. For each platform, the software must make provision for the design, implementation, evaluation, and optimization of mission-critical applications and the supporting layers of the software stack (e.g., data analytics, ML, math libraries, tools, runtime systems, workflow systems) on each architecture. Clearly, as computer architectures grow more diverse, so does the requisite software capabilities and expertise required to make use of them [191]. Performance portability will be a fundamental challenge. Current trends indicate a future that limits the accessibility of these platforms to only those who can afford large teams working over years to port and optimize applications for each new architecture. With respect to codesign, programming systems influence both how scientists write their applications and how the architectures are designed. For example, MPI applications and software have influenced the development of interconnects and Smart NICs, POSIX I/O has driven/constrained HPC I/O architectures, and CUDA has impacted development of GPU architectures for HPC.

Computational scientists and software developers already struggle to keep their codes running smoothly on heterogeneous platforms while maintaining the flexibility to move to other machines. Improving programming techniques to retarget codes efficiently with good performance across an increasingly diverse set of architectures is an emerging and significant challenge. Compilers and libraries usually lag hardware in capability, oftentimes taking months to years to provide optimized access to new hardware features. Portability layers like SYCL and Kokkos struggle with long compile times and rely on programming techniques that require expertise to implement critical kernels. Although large teams can afford experts, smaller teams often cannot. 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 of performance-portability strategies. Those strategies may 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.

Meanwhile, how programming environment and system software capabilities are being shaped by the growing diversity of application domains and execution scenarios must be recognized. In addition to traditional workloads, data-centric applications from experimental and observational science activities, as well as hybrid applications that include traditional applications augmented with aspects of AI and ML, present further diversification, complexity, and new challenges for the programming environment and system software.

Questions. Workshop breakout participants were asked to consider the following questions during the session:

- Do specific programming models, paradigms, or frameworks make codesign easier? Why?
- What information can the software stack capture to enable codesign or profiling?
- How does one normalize/transform architectural factors of application workloads across diverse architectures? What are the gaps?

- How do programming systems influence application development and vice versa? Gaps?
- How do programming systems influence architectures and vice versa? What are the gaps?
- How should codesign enable performance portability and productivity? What are the gaps?

In the framework of codesign, an important gedankenexperiment for the software stack is:
Would the same application written in different programming models/systems and executed on the exact same hardware yield the same architectural characteristics?

2.7.2 Success Stories

Unsurprisingly, HPC has accrued a number of software success stories over the decades. These successes include MPI; a variety of scientific and I/O libraries; and various languages, including Fortran. MPI [192, 193] has long been the de facto choice for nearly all scalable applications on contemporary systems. The implementation of MPI as an interface is built on a library and runtime system that allows it to be designed and deployed on many architectures. Furthermore, researchers have codesigned optimizations and hardware acceleration of important MPI features. MPI continues to be a practical choice for today's exascale applications because most of the new architectural complexity is confined to the node architecture.

Another success story is the development of rich scientific and I/O libraries. Over the decades, these libraries have grown to include a long list of recognizable names: BLAS, LAPACK, FFT, GMRES, METIS, HDF5, and many more. The traditional structure of these software libraries has enabled easy integration and porting to various architectures. However, for heterogeneous platforms, where data movement can easily dominate performance, it can be challenging for libraries to provide satisfactory performance unless great care is taken in optimizing their composition.

In terms of languages, Fortran has been a stalwart in HPC for over five decades. Fortran's support for multidimensional data structures, complex numbers, and special intrinsic functions makes it a natural fit for many scientific computing algorithms. Internally, some design choices in Fortran can make it easier for the compiler to optimize the source code. That said, nearly all programming systems (e.g., CUDA, OpenCL, HIP, and SYCL) for heterogeneous architectures are based on C, C++, or some library approach. This trajectory is beginning to reduce Fortran use across scientific applications as teams target these new architectures.

Recently, the community has been witnessing a fragmentation of programming models and implementations to deal with architectural heterogeneity, but a number of success stories are already appearing.

First, the LLVM compiler ecosystem has become the compiler infrastructure of choice for nearly all heterogeneous architectures. Its modular design facilitates composition of components that can support multiple frontends (e.g., C++, Fortran, and Julia) that target an array of architectures (e.g., multicore CPUs, GPUs, and SoCs). In fact, owing to its permissive licensing model, LLVM has gained considerable traction in the vendor software community and is the core of many existing heterogeneous compiler systems from NVIDIA, AMD, Intel, ARM, IBM, and others. Over the past two years, several vendors, including IBM and Intel, have abandoned their proprietary compiler software in favor of LLVM.

Second, the community is developing a number of programming frameworks to provide portability across heterogeneous architectures (e.g., SYCL, Kokkos, and Alpaka). These frameworks typically exploit C++'s metaprogramming capabilities to map their constructs onto lower-level programming systems, such as OpenCL. Although these frameworks are relatively new, they are providing an initial capability for addressing performance portability.

Finally, new programming languages and systems are emerging that may quickly change the ecosystem. In areas such as ML, TensorFlow [8] and PyTorch [42] are preferred because they typically hide much of the architectural complexity from developers. In the scientific computing community, languages such as Julia and Halide are growing in popularity for specific domains.

2.7.3 Opportunities

Investigate techniques and tools for building flexible, modular, and composable high-performance software across diverse architectures. As previously reviewed in several venues [191], future software stacks must be redesigned to meet the challenges of diverse architectures. As mentioned in Section 2.7.2, several software systems have been successful in these initial stages of architectural transition. Taking lessons from these early success stories, several factors, including flexibility, modularity, and composability, can contribute to tools and techniques that enable these solutions, and they must be balanced against end-to-end performance and reliability. For example, the LLVM compiler ecosystem offers a layered design that separates functionality of the source language parsing and optimization from the backend code generation and optimization. Meanwhile, LLVM offers an IR that provides a modular introduction of new analysis, transformation, and optimization across many frontend parsers and backend code generators.

In this regard, the ReCoDe participants recognized a need for new tools and techniques to construct software for these diverse architectures. It is no longer practical for each architecture to have its own monolithic software stack maintained solely by the hardware vendor. Rather, new software strategies must facilitate composability, interoperability, and flexibility while simultaneously providing high performance and reliability. In these relatively early stages of heterogeneous computing, technologies such as just-in-time compilation, autotuning, and containers are emerging as potential solutions to respective software problems. Another example is a multilevel intermediate representation (e.g., LLVM's MLIR) that can work at a higher level of abstraction than what compilers traditionally use. It can also provide a standard dialect that the compiler can use early in the compilation process to map high-level computation to new hardware.

Promote features throughout the software stack that enable codesign. Profiling interfaces and hardware counters traditionally have enabled codesign by allowing experts to analyze performance statistics in software and correlate them to hardware characteristics. However, future codesign efforts will require more advanced mechanisms given the plethora of architectures and execution models that heterogeneous platforms will support. Notably, in current systems, it already is challenging to interpret the outcomes of hardware counters. There is a need for features that allow programmers to reason about performance defects, as well as be able to isolate and address them in a more automatic manner. Composability of hardware counters and profiling interfaces will be essential to allow codesign for diverse platforms and application workloads. The OMPT interface, which recently was added to the OpenMP standard, is one example of a successful composable profiling interface. Embracing comprehensive and composable future sets will provide for the application of data mining and ML, which could assist several codesign tasks.

Create techniques and tools for decoupling algorithmic functionality from dynamic scheduling and mapping of computation onto heterogeneous resources. Manually attempting to coordinate, reason about, and schedule data placement, which types of processors to select for certain calculations, and when computations will occur is becoming an intractable problem as the complexity, diversity, and scale of HPC systems grows alongside increasingly challenging scientific and data-centric applications. The ReCoDe participants recognized the need for more intelligent, dynamic runtime systems that can schedule and map computation to appropriate resources in an intricate heterogeneous system with complex memory hierarchies. New schedulers must be able to capture characteristics of kernels and map them to specific devices meant to accelerate them. Additionally, because some of these devices will have their own direct-attached memory, reducing latency by orchestrating data movement simultaneously with scheduling and computation will be critical. In these systems, optimized resource management decisions must be made at a pace, scale, and level of complexity that exceeds human ability. Furthermore, today's operating and runtime systems are not designed to manage rapid changes in resource scheduling and workloads that cutting-edge science and extremely heterogeneous systems will demand. Along with algorithmic scheduling techniques, ReCoDe participants expect these new scheduling systems can be infused with AI, especially ML, to improve and automate system use, thereby increasing overall productivity and accelerating scientific discovery and innovation. Conversely, these intelligent runtime systems can capture and report on workload execution, which will provide architects and software developers with *real* performance data for codesign.

Develop techniques and tools for managing formal knowledge/semantics within and across different layers of software abstractions. Having formal semantics and well-defined behavior of software across different layers will enable sophisticated techniques to prove correctness and identify performance issues. First, efforts must be made to formalize the semantics for crucial components of the software stack, including runtime systems and code-generation frameworks. Next, tools and techniques must be developed to manage these formal semantics in a portable manner across different platforms. The formalization should have different aspects of software development, including but not limited to numerical aspects (floating-point), concurrency, and interoperability of software modules. This formalization also will encourage the creation of correctness tools that could verify ported codes or increase the level of confidence in existing ones.

2.7.4 Current Gaps

Methods and tools for designing a vertical software stack and interfaces. As mentioned in Section 2.7.3, the workshop participants felt strongly about the need for new tools and methods to design the software stack and its associated interfaces. In reality, the process of software construction today largely is the same as it was three decades ago. That is, developers combine languages with libraries and runtime systems using archaic software practices (e.g., build management, packaging, and source code control). Furthermore, a number of different programming techniques and systems from the AI/ML community have been injected into the portfolio.

Structured ways to bypass abstraction to engender competitive performance. Although there are distinct productivity advantages to programmer-defined abstractions, they also introduce complexity and obfuscation in terms of performance portability; performance analysis; and enabling compilers to analyze, optimize, and generate code effectively and efficiently. In many ways, this limitation forces application developers to address the complexity directly within their code versus having a toolchain designed to assist/support the specificity of the abstractions versus forcing developers to eventually cast their algorithms in terms of generalities. This intersection between the applications, software toolchain, and underlying system architecture provides numerous codesign opportunities.

Efficient, always-on hierarchical profiling of real workloads on contemporary architectures. Always-on production profiling allows for profiling live code and making more automated diagnoses of application slowdowns or problems. Daily profiling at the scale of hundreds of applications on thousands of nodes in an HPC center presents a significant challenge—overhead arises from intercepting the application state (usually via sampling) and managing large amounts of traces. Although the cost of always-on profiling can be consequential in HPC systems, there have been successful cases of data-center-scale tools (e.g., at Google) that can produce accurate profiles with negligible overhead [194]. With the trend of cloud and HPC convergence, such tools could play a role in codesigning always-on profiling. There is a gap, however, in designing composable interfaces for different architectures that can be stacked together.

2.8 Quantitative Tools and Data Collection for Modeling and Simulation for Codesign

2.8.1 Introduction

There is general consensus that the development of critical hardware/software technologies requires quantitative tools of codesign (ModSim) along multiple dimensions. To be successful, these tools must be applicable to design-ahead (in advance of implementation) and assist optimizations during execution of complex workflows on the target systems. ModSim capabilities must be practical (i.e., the time to solution for full system simulation under a realistic application workload should be reasonably low), accurate over a broad range of hardware/software architectures, and scalable as system complexity and size increase. Successful codesign tools must consider the performance/power/reliability triad in an integrated fashion and capture many of the boundaries up and down the hardware and software stacks. Moving forward, the nature of increasingly data-driven, dynamic, and irregular workflows will require continued emphasis on dynamic modeling methods and tools. To capture the behavior of such workflows, models could be assembled on the fly from data collection and prior training and have an increased spectrum of uses, such as optimization at runtime, introspection for runtime systems, workflow control, monitoring, and optimization. Data collected from production systems should be used to augment and inform codesign studies. However, real system data are, by nature, backward-looking, and this limitation must be addressed.

2.8.2 Performance, Energy, and Resilience Modeling

Value proposition. As CMOS technology scaling enters its twilight, architectural and algorithmic specialization has become essential for delivering continued exponential performance scaling. To realize a proper codesign cycle, architectures and algorithms must be evaluated along the metrics of performance, energy, and resilience in the context of real applications. As the cost of simulating each design is high, the resultant design-space explosion must be mitigated with low-cost modeling in a design-space triad, wherein potentially superior designs undergo additional detailed simulation while inferior designs are discarded. To accelerate this process, actionable feedback regarding why an architecture or algorithm is unprofitable must be passed back to the relevant architects and applied mathematicians. Performance models have been used as an intermediate language that is readily understandable by applied mathematicians, computer scientists, and computer architects. This ensures computer scientists can convey insights about how applications and algorithms stress architecture bottlenecks.

With the end of both Dennard and CMOS scaling on the horizon, performance alone is insufficient when assessing architectures. Increasingly, modern systems are not just power and/or energy limited, but the sheer scale of DOE systems makes them sensitive to Failure in Time (FIT) rates as well. As such, along with power or energy models, resilience models also must be incorporated into the codesign design-space triad.

Concurrent with the aspirations of a fully integrated codesign process is the reality of DOE computational center procurements that must grasp and guide opaque vendor codesign efforts. In these scenarios, procurement teams face two challenges. First, they must offer tractably small but workload-representative sets of unclassified benchmarks to drive vendor codesign efforts. Second, they must evaluate vendor offerings that often are presented as little more than a *speeds-and-feeds* table. While performance models can provide intuition about the impact of architectural changes on application performance, they also can be used to quantify the similarity and orthogonality of benchmarks, applications, and workloads.

Success stories. Most simulators and emulators operate in the time domain, stepping through execution cycle by cycle, whereas the majority of models operate in a time-integrated fashion by combining application and architectural parameters into a function that relates execution time (performance) to inputs and parameters.

Historically, the most successful and widely used model has been computational complexity with its associated big O, big Theta, and big Omega variants. As long as the cost of an operation could be parameterized and the operations dominated runtime, one could reasonably bound the execution time. Unfortunately, over the decades, data movement has dominated compute time, and performance has become more nuanced.

Significant success in analytical modeling most notably relates to application-centric performance models. These models and the quantitative codesign they enable have been successfully engaged for system design [195, 196] using applications that are representative of the anticipated production workloads [197].

To more accurately bound performance, a number of models have tried explicitly or implicitly to integrate a concept of data movement complexity [198, 199]. However, incorporating load imbalance, hotspots, or data dependencies has proven difficult because they lack a sense of computational depth or intuition regarding when a processor is starved for or gorging on parallelism.

Although a set of procurement benchmarks may be selected through intuition or via an attempt to replicate a set of canonical performance metrics, formal approaches embrace regimented mathematical underpinnings and unbiased data collection [200, 201].

Opportunities. *Public-Private Codesign Partnerships.* Although vendors are particularly adept at designing and engineering processor architectures, the way in which substantial information about DOE applications and algorithms is conveyed to them poses a primary dilemma. DOE provides benchmarks and proxy applications rather than information about the overall workload. Vendors use such benchmarks to evaluate architectural choices. Vendors will be guided in the right direction only when DOE benchmarks and proxy applications (in conjunction with industry benchmarks) perfectly match the performance characteristics of a center's workload. Because simulating a workload for all possible designs is computationally intractable, it is imperative that DOE provides representative benchmarks and also works with vendors to model architectures and characterize HPC center workloads and applications. Concurrently, any modeling effort must allow for changes in implementations or algorithms that mitigate identified performance, energy, or resilience bottlenecks. For all codesign endeavors to succeed, workload characterization and dissemination are critically important.

Big Tent and Specialized Procurements. Recently, DOE procurement processes coupled with vendor scale and expertise have led to a steep reduction in the number of companies that can deliver integrated systems at extreme scales. Smaller, more risk-adverse vendors that produce specialized designs lack the scale of engineering teams or computational resources to simulate a large design space. Codesign in collaboration with smaller vendors may have the beneficial result of a larger number of relevant responses to DOE procurement requests. Moreover, such initiatives could lead to a broader range of specialized designs, potentially offering superior performance and energy efficiencies at acceptable FIT rates.

Current gaps. Today, properly architecting the bandwidth tapering in a system is a well-understood (but not often realized) process. Codesign for these processes relies on cache analytical modeling or simulation to understand the two paths for mitigating cache/memory bottlenecks (i.e., less data, higher bandwidth). Analytical models provide a higher level of abstraction and are focused on a subset of the design parameters. Simulation is cycle-accurate and could, in principle, capture the full complexity of the design space. In a practical sense, analytical modeling and simulation trade-off accuracy for time to solution and are complementary tools in the codesign process. To date, analytical models have assumed processes behave in a bulk-synchronous and uniform manner. At low concurrencies, these assumptions usually hold. However, the last decade has witnessed an explosion in on-chip parallelism that now rivals the parallelism found in applications. Simultaneously, algorithms and implementations have diverged from the bulk-synchronous model and embraced a more dynamic, point-to-point synchronized approach. Today, few performance models fully incorporate parallelism, data dependencies, or computational depth. Moreover, such models ignore the temporal (e.g., alternating between compute- or memory-intensive phases) or spatial (e.g., cores, accelerators, network-on-chip [NoCs], or network links) burstiness of computation (i.e., hot spots).

Although performance models can be especially useful for providing performance intuition for the evolution of existing vendor offerings, they are challenged by novel non-von Neumann architectures. Some of the architectural and application metrics developed to date still apply (e.g., machine balance and arithmetic intensity), but models for applications running on spatial architectures, such as CGRAs and FPGAs, must develop architectural and application metrics as the compute graph is mapped in space among functional units rather than multiplexed across a fixed number of them.

Modeling how much energy a current architecture will consume for a given application can be accomplished using empirical methodology, such as parametrical models calibrated by measurement. However, predicting the energy (and energy efficiency) of future architectures can be difficult and particularly dependent on CMOS processes and technology. Nevertheless, realizing successful codesign in an energy-limited world (Joules-to-solution) will require the ability to predict energy as a function of application and architecture to an accuracy on par with performance. Failure to do so will place a much higher onus on simulation to accurately replicate energy. Modeling tools must expand their scope to include accurate energy models.

System architecture requires a careful balance of performance, energy efficiency, and FIT rates. Overemphasis on performance and energy efficiency at the expense of FIT rate will produce systems usable at only the smallest scales. In addition, the engineering effort required to reduce a FIT rate may be beyond the scope of smaller vendors. Open-source tools for modeling and simulating an architecture's sensitivity and FIT rate will be essential if DOE wants to expand the number of vendors that can aptly respond to exascale procurements.

2.8.3 Simulation and Simulators

Value proposition. Designing future HPC systems will require capable simulators that can handle greatly increased size and complexity, as well as provide accurate and timely feedback to the design process. Emerging dynamic and irregular applications and workflows present new challenges to existing simulators and will require brand-new simulation methods. The complexity of these applications may be such that existing static analytical models are insufficient, and simulation will be required to capture their dynamic behavior.

The hardware architecture space is increasingly diverse. The growth of chiplets, increasing popularity of custom ASIC and FPGA designs, new design languages, and a growing acceptance of fixed- or limited-function accelerators provide a host of novel architectural possibilities. Radically new architectures, from analog to neuro-inspired computing, also are expanding the design space. These specialized compute units may be easier to simulate individually than a conventional general-purpose CPU, but the sheer variety of new devices will present simulation challenges.

To handle this growing complexity and diversity, simulation must play an even greater role in the design process. Simulators must evaluate proposed hardware designs, explore new architectures, and be used to design programming models/languages and runtime systems.

Success stories. Preliminary and exploratory simulation of advanced computer architectures and applications has been a central aspect of DOE computer research and procurement [202]. The ability to estimate future performance, evaluate architectural or algorithmic alternatives, prototype system software, and even diagnose performance shortfalls [203] relies heavily on simulation. Although traditionally focused on estimating performance, simulators also have been used to estimate power and reliability, which are increasingly important constraints on future systems.

The current state of the art includes an array of simulators that run the gamut from single core to million-node system simulators. Some popular simulators used for HPC include the following:

- Processor: gem5 [204], GPGPUSim [205], Stake/Spike [206], Scarab [207], and Accel-Sim [208]
- Memory: DRAMSim [209], HMCSim [210], and Cramsim [211]
- Network: CODES/ROSS [212] and booksim [213]
- Lower-level simulation languages: Chisel/ESSENT [214], SystemC, and SystemVerilog
- Multicomponent frameworks: SST [215] and LSE [216].

DOE has a long history of using and developing simulators with industry partners. The Exascale Computing Project (ECP) has made considerable use of SST, Accel-Sim, and CODES. ECP has also funded enhancements to Cray's SST simulation models used to evaluate the future Slingshot network. By developing and extending these simulators, DOE can ensure future designs are evaluated with relevant workloads.

Opportunities. *Automate and Accelerate the Codesign Cycle.* Reducing the time of a codesign cycle—design, simulate, diagnose, optimize, and repeat—will become increasingly important as systems scale and complexity grows. Existing simulators can be made faster, but fundamentally new approaches to automate the cycle must be explored. ML approaches should be sought to examine wide design spaces more efficiently and develop faster ML-based models from conventional simulations. Basic research is needed to understand which parts of simulation are amenable to ML and how to collect and process data into trusted, actionable results. This research should include close interaction with vendors to understand the broader community's needs and provide timely guidance to industry partners.

Interdisciplinary Accessibility. The system design process will become increasingly interdisciplinary, and simulators must serve a diverse set of domain experts. Future simulation tools must support circuit designers, computer architects, application developers, and applied mathematicians. Providing useful feedback to all of these groups will require a level of agility and accessibility not currently available. Simulators must be able to mix high-detail and fast, low-detail models to allow trade-offs between accuracy and time to solution. They will need to operate at multiple fidelities and scales. For example, simulating all CPUs in a full system at a high level of detail may be impractical, but it could be possible to simulate a handful of CPUs at full detail and the rest more abstractly. Additionally, simulators must provide quantitative guidance on how much accuracy is being lost or how much speed is being gained.

Interoperability. Having a variety of simulators provides the architecture community with a range of options and trade-offs. However, the lack of interoperability makes it difficult to take advantage of these options fully. The community needs standard interfaces between simulators to enable composable simulation, i.e., the ability to mix and match the best simulators to meet user needs. Such composability would improve research at the national laboratories and could enhance industry and academic research. By lowering the barrier to entry, more groups would be able to perform better simulations. Other requirements for simulation interoperability include the following:

- **Long-term support for integration**. Without this, integrated simulators will inevitably diverge. Industry and the national laboratories must play a role in sustaining these toolchains because academia is not well suited for long-term software maintenance.
- **Standard interfaces**. Simulation interfaces must allow integration between high- and low-level models to allow for accuracy and performance trade-offs. Interfaces to profiling tools and high-level analytical models also must be provided.
- **Near-seamless integration**. Integration should be as seamless as possible. Presenting a common user interface will improve analysis workflows and reduce the learning curve.

Current gaps. *Uncertainty Quantification for ModSim.* Architectural simulators exist on a spectrum between higher-level analytical models and emulators on one side and lower-level circuit design tools and hardware prototypes on the other. Each simulator presents a set of trade-offs and excels at some tasks yet may be ill suited for others. Simulators must balance performance and accuracy. Higher-level simulators can be orders of magnitude faster than low-level simulators, but they may also be less accurate. It is important to understand a simulator's weaknesses (e.g., what does the simulator capture, or what does it neglect?).

Perhaps, the biggest gap in current simulation is a rigorous, quantifiable meta-methodology to decide what level of fidelity is needed for a given experiment, which could guide the choice of simulator. Currently, the simulator choice usually is governed by which tool a research group is most familiar with using. A robust methodology could provide much more effective guidance on the appropriate simulator to use and what the error bounds may be for it. Developing such a method can rely upon the vast expertise in uncertainty quantification and the determination of provable error bounds.

Emerging Architectures. Most current simulation tools focus on traditional architectures (i.e., CMOS CPUs/GPUs connected to memory and a network). Future architectures are expected to be much more diverse and include a range of new accelerators, devices, and programming models. Better support for emerging neuromorphic, non-von Neumann, and edge computing devices will be critical. Low-level models for non-CMOS and analog devices also will be required.

In addition to hardware models, a flexible software stack of easily retargeted prototype runtimes, compilers, and algorithms is needed to support fast design-space exploration. For example, tools to automatically map algorithms for CGRA and neuromorphic architectures are needed to perform hardware exploration. Irregular workloads pose additional problems because their dynamic nature must be taken into account. Static approaches (e.g., static instruction traces) will be insufficient. Future simulators will need a software stack that can support workflow simulation, particularly multiuser interactions and contention.

Interdisciplinary Skill Set. Resolving these gaps and challenges will require an interdisciplinary approach. Siloed teams of architects or application developers will not have the range of expertise to support true codesign. Future balanced teams will require knowledge of architecture, runtimes, applications, devices, and reliability effects. Traditionally, power and reliability modeling are complicated because the required design details are highly proprietary. There also will be a need for expertise in the economics of custom architectures. Currently, the cost of new architectures often is neglected in research or is represented only by inaccurate proxies, such as chip area or network ports. Because economics usually poses the fundamental limit on an architecture, it will be necessary to embrace quantitatively driven and financially constrained codesign.

2.8.4 Data Collection and Other System-related Topics

Value proposition. Notable benefits are realized when decisions are based on data-driven rather than heuristics-driven strategies. The problem with a subjective set of rules is that it merely offers an approximation of the optimal solution for any given batch of criteria. For a significant spectrum of problems that are nonlinear in nature, the goal must be to achieve close-to-full utilization of system resources. For such workloads, the accuracy of runtime models that map system resources to workload characteristics is of paramount importance.

Detailed knowledge of application demands could enable architects to make more informed decisions about how to select and organize computing hardware. Detailed understanding within data-driven system software could enable improved hardware resource management (e.g., better energy and workflow management). Moreover, integrating high-level profiling and analysis with low-level resource management routines could enable these systems to implement new policies that respond flexibly to changes in application demands that may expose powerful new efficiencies on platforms with heterogeneous hardware.

Success stories. HPC centers already collect and analyze statistical information to guide procurement and allocation decisions. The Collaboration of Oak Ridge, Argonne, and Livermore (CORAL) procurement that resulted in ORNL's Summit and LLNL's Sequoia machines made extensive use of such data [217]. These procurements were possible owing to advances made by monitoring frameworks and studies in global operating systems (e.g., the Lightweight Distributed Metric Service [LDMS] [218], Data Center Data Base [DCDB] [219], disaggregated memory, and the Hobbes research operating system [220]).

Monitoring systems, such as Ganglia [221], LDMS [218], DCDB [219], and others, have paved the way for numerous applications, including improved health and diagnostics, resource management, cybersecurity, and networking. Matching application demands to the appropriate resources is essential for achieving efficient performance on platforms with diverse hardware capabilities. Researchers have been developing tools and strategies to address this problem with some promise [222–225].

Opportunities. *Increased Data Collection Scope and Detail.* Multiple drivers are promoting increased data collection in terms of scope (the universe of data sources) and detail. Enabling supercomputer integration with empirical methods must accommodate scientific instruments that incorporate more sensors than ever, leading to a data deluge [226]. In addition, big data processing and analysis techniques pioneered in cloud computing are being adopted and refined for use in HPC environments, particularly for analyses based on ML [227]. Together, these trends result in not only more data, but data that are also much more detailed and may require more associated metadata for their potential to be realized.

Reducing Latencies. Human-in-the-loop approaches have limited ability to address global or architecturally complex situations or respond on timescales of interest, e.g., Kramer et al. [228]. These offline approaches preclude many desirable time-sensitive use cases, and challenges that currently limit analytical architectures to a largely offline mode must be overcome. Dynamic understanding must be founded on architectures that can provide up-to-date system state information and be backed by a runtime analysis engine. The runtime analysis engine will be improved by higher data ingestion rates, increased flexibility to evolve execution and system state models, and faster feedback to system software.

Interoperability. As supercomputing evolves from traditional use cases to much more expansive roles (e.g., integrated experiments), seamless operation of data access and discovery offers novel opportunities for interoperability. The community needs standard interfaces between data sources to lower the barrier to entry. The challenge posed by more use cases (e.g., system software, application software, or ML-based systems for various purposes) resides with the competing interests of applicability and specificity: **the optimal interface will support discovery and access interfaces that are general for broad application yet specific enough to select the appropriate data from a sea of available information.**

Current gaps. *Interfaces for Data Discovery and Access.* To fully realize the tremendous potential of data-driven techniques, an abstraction layer is needed. The abstraction layer must support accessing HPC center data/information repositories for generalized use. It also must provide for straightforward access and support use cases at various levels of operation in a way that still impedes unauthorized entry. Ideally, the framework will support diverse use cases, from low-level system software kernels to meta-tools working in expanded contexts, in which the supercomputer is only one component of a larger interlocking system.

Dynamic Data-driven System Software. Traditionally, global resource management has been a largely static process with limited information about the system and subsystem states and minimal knowledge of the application demands. Typically, scheduling decisions are based on priority assessments (e.g., fair share), and allocation decisions are merely a sequential allotment of identical compute resources. As HPC systems become more heterogeneous and application codes increase in complexity with respect to optimization of computational kernels for architecture-specific features (e.g., vendor-specific CPU/GPU architectures, memory technologies and hierarchies, network bandwidth, and latency characteristics), more intelligent and dynamic resource management will be required.

Many cloud and edge computing technologies that ease deployments across heterogeneous environments, such as containerized applications and services and elastic compute and storage resource provisioning, can also be engaged to enhance HPC applications and system software, as well as improve adaptability for complex dynamic workflows. For example, containers present an obvious opportunity for enhanced insight into workflow behaviors through the encapsulation of integrated monitoring and provenance tracking.

Finally, these data-driven techniques must not introduce unacceptable levels of overhead.

Data Disposition Policies and Data Sharing. Data-driven approaches frequently give rise to complicated right-to-privacy and confidentiality issues. The actual dissemination of data and information, raw or derived, must be governed by rules set forth by the appropriate data owners. Moreover, data can be made appropriate to a larger audience through proper anonymization techniques. Today, data disposition is mostly managed by small silos of information maintained by many different subgroups within the HPC universe. To realize the highly desirable open exchange of available data, each data item must be labeled with access rights, and those rights must be enforced throughout the data life span.

3 Priority Research Directions

3.1 Drive Breakthrough Computing Capabilities with Targeted Heterogeneity and Rapid Design

Key questions.

- *What new methods and technologies are required to rapidly create breakthrough hardware designs?*
- *How can we ensure that they align to support increasingly diverse and demanding computing requirements?*

In today's rapidly changing technology landscape, designing radically heterogeneous systems requires new codesign methodologies that create an accurate understanding of workloads and use this understanding to target and drive the creation of complementary sets of accelerators and heterogeneous structures that combine to create breakthrough systems. Accelerators must balance capability with breadth of utility. The flexible accelerated integration of such heterogeneous customized elements depends on new advanced physical integration technologies (e.g., chiplets; advanced packaging), architectural integration (e.g., new memory interfaces, communication links, and open standards/protocols), and accelerated hardware development (e.g., open-source designs, technology libraries, and open-source tools).

Comprehensive design to support end-to-end scientific workflows. For the past few decades, the HPC community has focused on increasing ModSim performance. This focus has driven faster compute capabilities and increasing node counts that promoted gains through weak scaling. Although improvements have been made in file systems, visualization, and analysis functionality, a traditionally macroscopic and cohesive view of scientific workflows and full machine workloads was rarely employed. The nature of the science being performed on the largest HPC resources is changing, which was clearly demonstrated during the COVID-19 pandemic with a substantial allocation of resources being allocated to help support drug and vaccine discovery. Strengths in characterizing individual applications (proxy applications) were a marked contrast to limited characterization of workflows and workloads, e.g., mesh/input generation, data conditioning, large-scale pre-/post-simulation data analysis, ongoing uncertainty quantification, and continued model verification studies. Development of advanced workload and workflow descriptions, which are analogous to mini-apps, are critical to shaping and characterizing future heterogeneous system-design activities.

Remaining research questions.

- *What are the best methods to capture user and hardware requirements?*
- *How are such operations supported at scale?*
- *What is the best way to summarize these observations in a digestible format that is usable across multiple DOE facilities and user communities?*
- *How should this information be represented to support codesign activities in a similar way to how mini-apps have been used during exascale development?*

In the background, important questions also remain regarding how to validate these behaviors and ensure that models are representative over the course of application and/or system lifetimes. Such questions also raise challenges for ensuring system software, programming models, and the supporting HPC environment (e.g., user policies) are all reflected in representations selected by DOE to promote exploration and research and development activities.

Composable and interoperable HPC systems for the future. Codesign of future HPC systems does not imply the creation of narrowly applicable hardware, software, or applications. Rather, increased composability and interoperation of highly capable system components can enable integrated systems to include best-in-breed solutions and flexible IP blocks and be augmented with key novel or newly developed components. Specific device/component designs may come from industry, academia, or even be developed at a national laboratory. Prior experience proves that allowing one industry player to specify the base protocols can disadvantage others, so broader consortia are preferred. How best to design interoperable and flexible open standards that support rapid integration of hardware components and IP blocks into a high-performance, energy-efficient, and cost-effective system is an open challenge. The protocols must be extensible into the future as newer devices or technologies are developed. Industry has moved to adopt several such technologies in the past—PCIe and CXL to name two. However, fundamental research into applying these approaches to integrated IP blocks with SoC designs or fully heterogeneous systems is needed to realize the unique challenges of HPC system-scale and node-level performance requirements. Successful outcomes from this approach will be optimized protocols based on open standards that can be easily adapted for disparate system configurations and deployments across DOE facilities and the world.

Employ an accurate understanding of applications/workflows to codesign novel memory structures (FIM, PNM, novel packaging) that unlock breakthrough increases in memory performance. Memory performance bandwidth, capacity, and latency are critical to continued performance increases for applications across the spectrum, from traditional HPC to distributed workflows coupled to scientific instruments. Unlocking the most promising opportunities requires new approaches. Prior codesign efforts mostly have focused on small computational kernels and studied limited processor/GPU examples and bounded memory subsystems. This is insufficient for novel FIM and PNM. Moreover, codesign must include a wider scope in all dimensions, such as: (1) workloads that span from traditional model-driven simulation to AI-based modeling—even AI-based data analytics—that provide rich, complex demands for data movement and computational structures with a single package, node, or whole system; (2) simulation and emulation models that span traditional compute elements (CPUs and GPUs), interconnects, and memory structures; add FIM or PNM devices; and capture performance, power, and even device wear/lifetime; and (3) significant software toolchain exploration to create, evaluate, and refine efficient programmability and robust performance.

Remaining research questions.

- *What operations are most suitable for memory, how tightly should they be integrated, and how are they exposed in the larger system architecture?*
- *How are they exposed to software and shared among applications in a multitasking or service scenario?*
- *How will these capabilities play out in a computing industry historically divided at the processing-memory interface?*

Employ entire application life cycles, workflows, full system models, and emulation to codesign novel computational storage that unlocks dramatic new data science capabilities. Exponential increases in storage bandwidth provide compelling new opportunities in data-driven science and analytics if equally capable computational storage accelerators can emerge to leverage this potential. Again, a new type of codesign that spans the hardware elements (i.e., CPU/GPU, memory, interconnect, and storage) and the software stack (i.e., application, operating system, and more) is required. For effective computational storage, codesign must include: (1) workloads involving computation and large-scale, interleaved access to storage, which necessitates modeling the application, operating system, and storage system layers (e.g., object stores, key value, and direct access); (2) simulation and emulation models that span from compute microarchitecture to system fabric to storage device (and computational acceleration) that capture performance and power at scale; and (3) significant software stack prototyping and deployment at scale to create, evaluate, and refine efficient programmability and robust performance.

Remaining research questions.

- *What operations are most suitable for computational storage, and how might they be defined and exposed in the larger system architecture?*
- *If there is programmability, how is this securely exposed?*
- *How are the operations are exposed to software and shared amongst applications in a multitasking or service scenario?*

3.2 Software and Applications that Embrace Radical Architecture Diversity

Key question. *What novel approaches to software design and implementation can be developed to provide performance portability for applications across radically diverse computing architectures?*

Current trends in computer architecture pose the challenge of codesigning a complete system architecture that includes traditional CPUs combined with a vast array of potential accelerators: GPUs, FPGAs, CGRAs, DSPs, TPUs, dataflow accelerators, AI accelerators, chiplets containing application- or domain-specific circuitry, analog processors, neuromorphic processors, quantum computers, and classical or quantum annealers. These technologies may be integrated into a single package within a node, across nodes within a single parallel computer (with different nodes containing different accelerators), or across multiple parallel computers (with one computer providing a single accelerator architecture and some other computer providing another).

Given this range of architectural diversity, the community has to rethink the design of software systems and applications, which must adapt to each target architecture to provide performance, efficiency, and robust results while maintaining some level of portability across all of these options. In contrast to traditional hard-coded applications that rely on static compilation and mapping computation to accelerators, programming systems (i.e., compilers, runtime systems, operating systems, and libraries) that are modular, configurable, introspective, and easily extensible to accommodate the expected architectural diversity are needed. In fact, findings from the ReCoDe workshop emphasize that codesigning novel hardware is stymied by a lack of standards and widely accepted quality metrics for benchmarks and proxy codes. Each accelerator favors its own programming system, which leads to a proliferation of those often-competing models. Similarly, codesign tools (including simulators), if they exist, tend to be accelerator-specific, so codesign investigations are often not portable to other computational paradigms.

Applications must be written in programming models that can be broadly applied to many different computational paradigms. From the application source code, the compilers must generate optimized kernels for all of the targeted accelerators in the target architecture. Here, LLVM's MLIR is an effort to provide a higher level of abstraction than traditional compiler IRs that provides a standard abstraction (or MLIR dialect) which can be used by both the software and hardware. Then, the runtime system must be able to discover available accelerators, map kernels to accelerators, schedule kernels across accelerators to provide good performance, and orchestrate data movement so it does not impede application execution. Most contemporary runtime systems provide only basic mechanisms to applications, forcing users to rebuild their application for each accelerator architecture to account for these challenges. When possible, libraries should provide domain-specific abstractions to applications that hide this architectural complexity. Finally, the operating systems must be redesigned so they can provide system services for execution, memory management, communication, and security uniformly for all accelerators.

Ultimately, the ReCoDe participants advocated for creating a new strategy to design software stacks. If accelerators were created and added to architectures using a configurable and modular technology, like chiplets, the software stack could be extended similarly in a modular and configurable way. This modularity would promote reuse, interoperation, and rapid deployment.

Because massively heterogeneous systems remain an emerging technology, there are numerous research opportunities for improving the use of codesign in this space. For example, the aforementioned LLVM MLIR can work at a higher level of abstraction than typical compilers and offers a standard dialect for both software and hardware use. As there may be many ways to represent a given problem, even on a single piece of hardware, problem encoding is a new aspect of codesign to consider. Additionally, in some cases, such as approximate and probabilistic computing, codesign must be used to design and optimize new algorithms to take full advantage of specialized architectures. Algorithms have to understand how less precision, accuracy, and/or repeatability than conventional CPUs can provide better solutions. Programming models, tools, benchmarks, and proxy applications all will need rethinking for an era of massive heterogeneity in which codesign offers numerous benefits.

3.3 Engineered Security from Transistors to Applications

Key questions.

- *How does codesign consider needs for end-to-end scientific computing and scientific data security, provenance, integrity, and privacy?*
- *What computer security innovations from the commercial computing ecosystem (e.g., trusted execution environments; hardware separation approaches) can be codesigned to provide security for DOE scientific discovery?*

Revolutionary advances are needed to extend roots of trust and other security capabilities to support DOE's scientific discovery continuum, which leverages research networks (e.g., Energy Sciences Network) to integrate supercomputing facilities with experimental user facilities, and this may extend to advanced IoT sensors. This distributed scientific ecosystem means increased exposure to threats, but it also offers the opportunity to engage ubiquitous logic capabilities to enhance computer and data security. As a new area of concern, quantitative metrics assess how security codesign trade-offs can be made in conjunction with traditional metrics (e.g., power, performance, reliability).

ReCoDe Security and Privacy Breakout Group Priority Research Direction Recommendations. The ReCoDe cybersecurity breakout team identified three candidate recommendations for inclusion in ReCoDe Priority Research Directions:

(1) **The need to develop community-wide definitions of security in scientific computing, such as those created for data trustworthiness and data confidentiality by the Trusted CI Trustworthy Data Working Group [229, 230].** Although cybersecurity performance benchmarks primarily are an elusive idea in the near term, it is important to adopt a community-wide understanding of the appropriate threat model for HPC, definitions of security for HPC, and a common set of use cases that can be considered with respect to security issues. This will help ensure that security enhancements are compatible with the scientific computing workflows and also may guide how security impacts on performance can be examined. In addition, it helps to frame potential classes of threats and their impact on scientific computing. Similarly, workflows could be divided into categories, such as batch versus interactive jobs, or sensor-accumulation-analysis-storage, control-device-process, and data-orchestration HPC models, which would probe differences in the impacts of threat classes on workflows that prioritize networks, devices, and systems, respectively. This would require ways to characterize the impact of various categories of attacks on scientific computing workflows, including potential information leakage if confidentiality is violated, disruption of workflows if availability is attacked, or alteration of scientific results should integrity be corrupted.

(2) **Codesign tools to support exploration of trade-offs in the design space between cybersecurity, performance, and other key properties.** Current codesign simulation tools that analyze computational performance, energy efficiency, and resource utilization ideally would be augmented so security properties are explicitly part of the trade-off analysis performed. Any progress made in characterizing cybersecurity performance, as defined in the first Priority Research Direction, would drive the design of codesign simulators so this additional fitness function could be included in simulation assessments. Likewise, example scientific privacy workflow use cases and threat classes would be incorporated into such a framework. This would allow architectural designers to dial in their desired balance between security, computational performance, and energy efficiency for their applications.

(3) **Models for enhancing isolation mechanisms, extending roots of trust, and other security properties appropriate for HPC.** For enhanced security properties, processing is increasingly available in the network itself (i.e., the interconnect fabric), memory, and storage devices, which enables computing to be performed ubiquitously throughout computational workflow environments. A major goal of this Priority Research Direction would be to leverage codesign in these environments to explore how security properties and trust roots flow through such a dynamic and distributed environment, which also would include HPC platforms and/or data centers, cloud environments, control and experimental facilities, and other elements of scientific workflows. As instruments and devices themselves become *smarter* in their ecosystem of sensor, analysis, network, control, and storage devices, this will motivate exploration of new mechanisms for establishing trust in distributed dynamic systems.

3.4 Design with Data-rich Processes

Key question. *What are the quantitative tools that are practical, accurate, and applicable to codesigning various layers of the hardware and software stacks and data-driven, dynamic, irregular workflows, such as those occurring in experimental science or AI/ML workloads?*

To be successful, quantitative tools (e.g., simulators, emulators, and profilers) must be applicable to design-ahead approaches (in advance of implementation) and assist optimizations during execution of complex workflows on the target system. These ModSim capabilities must be sufficiently fast for repeated and potentially online use; consider the triad of performance, power, and reliability in an integrated fashion; be accurate over a broad range of hardware and software architectures; and be scalable as the system complexity and size increases. Dominant workflows are increasingly data driven, dynamic, and irregular, thereby requiring new, dynamic, and runtime-oriented methods and tools for codesign.

4 Contributors

The following people contributed to writing this report:

- Ang, James, *Pacific Northwest National Laboratory*
- Ansel, Tim, *Google Inc.*
- Cardwell, Suma, *Sandia National Laboratories*
- Chien, Andrew A., *Argonne National Laboratory* and *University of Chicago*
- Coffrin, Carleton, *Los Alamos National Laboratory*
- Crawford, Sam (technical editor), *Oak Ridge National Laboratory*
- Cunningham, Aaron, *Google Inc.*
- Denes, Peter, *Lawrence Berkeley National Laboratory*
- Finkel, Hal, *U.S. Department of Energy, Office of Science*
- Hammond, Simon, *Sandia National Laboratories*
- Hoisie, Adolfy, *Brookhaven National Laboratory*
- Jantz, Michael, *University of Tennesse - Knoxville*
- Jones, Terry, *Oak Ridge National Laboratory*
- Karlin, Ian, *Lawrence Livermore National Laboratory*
- Laguna, Ignacio, *Lawrence Livermore National Laboratory*
- McCormick, Pat, *Los Alamos National Laboratory*
- Oehmen, Chris, *Pacific Northwest National Laboratory*
- Pakin, Scott, *Los Alamos National Laboratory*
- Peisert, Sean, *Lawrence Berkeley National Laboratory*
- Plata, Charity (editor), *Brookhaven National Laboratory*
- Rodrigues, Arun, *Sandia National Laboratories*
- Ross, Robert, *Argonne National Laboratory*
- Schuman, Catherine, *Oak Ridge National Laboratory*
- Shalf, John, *Lawrence Berkeley National Laboratory*
- Vetter, Jeffrey S., *Oak Ridge National Laboratory*
- Wiedlea, Andrew, *ESNet*
- Williams, Samuel, *Lawrence Berkeley National Laboratory*

References

[1] Vladimir Getov, Adolfy Hoisie, and Harvey J. Wasserman. Codesign for systems and applications: Charting the path to exascale computing. *Computer*, 44(11):19–21, 2011. DOI: 10.1109/MC.2011.334.

[2] J. A. Ang, R. F. Barrett, R. E. Benner, D. Burke, C. Chan, J. Cook, D. Donofrio, S. D. Hammond, K. S. Hemmert, S. M. Kelly, H. Le, V. J. Leung, D. R. Resnick, A. F. Rodrigues, J. Shalf, D. Stark, D. Unat, and N. J. Wright. Abstract machine models and proxy architectures for exascale computing. In *2014 Hardware-Software Co-Design for High Performance Computing*, pages 25–32, 2014. DOI: 10.1109/Co-HPC.2014.4.

[3] William Chen and Bill Bottoms. Heterogeneous integration roadmap: Driving force and enabling technology for systems of the future. In *2019 Symposium on VLSI Technology*, pages T50–T51, 2019. DOI: 10.23919/VLSIT.2019.8776484.

[4] Tao Li, Jie Hou, Jinli Yan, Rulin Liu, Hui Yang, and Zhigang Sun. Chiplet heterogeneous integration technology—status and challenges. *Electronics*, 2020.

[5] Kevin Drucker, Dharmesh Jani, Ishwar Agarwal, Gary L. Miller, Millind Mittal, Robert Wang, and Bapiraju Vinnakota. The open domain-specific architecture. *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 25–32, 2020.

[6] J. S. Vetter, R. Brightwell, M. Gokhale, P. McCormick, R. Ross, J. Shalf, K. Antypas, D. Donofrio, A. Dubey, T. Humble, C. Schuman, B. Van Essen, S. Yoo, A. Aiken, D. Bernholdt, S. Byna, K. Cameron, F. Cappello, B. Chapman, A. Chien, M. Hall, R. Hartman-Baker, Z. Lan, M. Lang, J. Leidel, S. Li, R. Lucas, J. Mellor-Crummey, Jr. P. Peltz, T. Peterka, M. Strout, and J. Wilke. Extreme heterogeneity 2018: DOE ASCR basic research needs workshop on extreme heterogeneity, 2018. DOI: 10.2172/1473756.

[7] Gordon E. Moore. Cramming more components onto integrated circuits. In *Electronics*, volume 38, pages 114–117, April 19, 1965.

[8] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, Georgia, USA, November 2016. USENIX Association. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[9] Yakun Sophia Shao, Sam Likun Xi, Vijayalakshmi Srinivasan, Gu-Yeon Wei, and David M. Brooks. Co-designing accelerators and soc interfaces using gem5-aladdin. In *49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2016, Taipei, Taiwan, October 15-19, 2016*, pages 48:1–48:12. IEEE Computer Society, 2016. DOI: 10.1109/MICRO.2016.7783751.

[10] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Ierardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. Anton, a special-purpose machine for molecular dynamics simulation. *SIGARCH Computer Architecture News*, 35(2):1–12, June 2007. DOI: 10.1145/1273440.1250664.

[11] Luis Ceze, Mark D. Hill, and Thomas F. Wenisch. Arch2030: A vision of computer architecture research over the next 15 years. *CoRR*, abs/1612.03182, 2016. URL: http://arxiv.org/abs/1612.03182, `arXiv:1612.03182`.

[12] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74(12):3202–3216, 2014.

[13] David A. Beckingsale, Jason Burmark, Rich Hornung, Holger Jones, William Killian, Adam J. Kunen, Olga Pearce, Peter Robinson, Brian S. Ryujin, and Thomas R. W. Scogland. RAJA: Portable performance for large-scale scientific applications. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 71–81, 2019. DOI: 10.1109/P3HPC49587.2019.00012.

[14] Shahab Ardalan, Ramin Farjadrad, Mark Kuemerle, Ken Poulton, Suresh Subramaniam, and Bapiraju Vinnakota. An open inter-chiplet communication link: Bunch of wires (BoW). *IEEE Micro*, 41(1):54–60, 2021. DOI: 10.1109/MM.2020.3040410.

[15] Bapiraju Vinnakota. The Open Domain-Specific Architecture: Next steps to production. In *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, NANOCOM '21, New York, New York, USA, 2021. Association for Computing Machinery. DOI: 10.1145/3477206.3477462.

[16] UCIexpress Open Chiplets Die to Die Interconnect Standard. URL: https://www.uciexpress.org.

[17] Compute Express Link (CXL) Specification Revision 2.0, October 2020. URL: https://www.computeexpresslink.org/spec-landing.

[18] Heterogeneous Integration Roadmap 2019 Edition, 2019. URL: https://eps.ieee.org/technology/heterogeneous-integration-roadmap/2019-edition.html.

[19] LiquidStack. Immersion cooling: A revolution in data center cooling. URL: https://liquidstack.com/.

[20] Majid Jalili, Ioannis Manousakis, Inigo Goiri, Pulkit Misra, Ashish Raniwala, Husam Alissa, Bharath Ramakrishnan, Phillip Tuma, Christian Belady, Marcus Fontoura, and Ricardo Bianchini. Cost-efficient overclocking in immersion-cooled datacenters. In *Proceedings of the International Symposium on Computer Architecture (ISCA'21)*, June 2021. URL: https://www.microsoft.com/en-us/research/publication/cost-efficient-overclocking-in-immersion-cooled-datacenters/.

[21] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. Chisel: Constructing Hardware in a Scala Embedded Language. In *DAC Design Automation Conference 2012*, pages 1212–1221. IEEE, 2012.

[22] John Clow, Georgios Tzimpragos, Deeksha Dangwal, Sammy Guo, Joseph McMahan, and Timothy Sherwood. A Pythonic approach for rapid hardware prototyping and instrumentation. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7. IEEE, 2017. DOI: 10.23919/FPL.2017.8056860.

[23] Derek Lockhart, Gary Zibrat, and Christopher Batten. PyMTL: A Unified Framework for Vertically Integrated Computer Architecture Research. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 280–292. IEEE, 2014.

[24] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. Cooper-Balis, and B. Jacob. The Structural Simulation Toolkit. *SIGMETRICS Performance Evaluation Revivew*, 38(4):37–42, March 2011. DOI: 10.1145/1964218.1964225.

[25] Doru-Thom Popovici, Andrew Canning, Zhengji Zhao, Lin-Wang Wang, and John Shalf. A systematic approach to improving data locality across Fourier transforms and linear algebra operations. In Huiyang Zhou, Jose Moreira, Frank Mueller, and Yoav Etsion, editors, *ICS '21: 2021 International Conference on Supercomputing, Virtual Event, USA, June 14-17, 2021*, pages 329–341. ACM, 2021. DOI: 10.1145/3447818.3460354.

[26] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken. Legion: Expressing locality and independence with logical regions. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, 2012. DOI: 10.1109/SC.2012.71.

[27] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv:1209.5145*, 2012.

[28] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. MLIR: Scaling compiler infrastructure for domain specific computation. In *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 2–14. IEEE, 2021.

[29] Samuel Naffziger, Noah Beck, Thomas Burd, Kevin Lepak, Gabriel H. Loh, Mahesh Subramony, and Sean White. Pioneering chiplet technology and design for the AMD EPYC and Ryzen processor families: Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 57–70, 2021. DOI: 10.1109/ISCA52012.2021.00014.

[30] Intel. Embedded multi-die interconnection bridge (EMIB), 2020. URL: https://www.intel.com/content/www/us/en/silicon-innovations/6-pillars/emib.html.

[31] David F. Richards, Omar Aziz, Jeanine Cook, Hal Finkel, Brian Homerding, Tanner Judeman, Peter McCorquodale, Tiffany Mintz, and Shirley Moore. Quantitative performance assessment of proxy apps and parents. Technical Report LLNL-TR-750182, Exascale Computing Project, US Department of Energy, April 25, 2018. URL: https://www.osti.gov/biblio/1438753-quantitative-performance-assessment-proxy-apps-parents, DOI: 10.2172/1438753.

[32] Apple announces Mac transition to Apple Silicon, June 2020. URL: https://www.apple.com/newsroom/2020/06/apple-announces-mac-transition-to-apple-silicon/.

[33] Alexander Branover, Denis Foley, and Maurice Steinman. AMD Fusion APU: Llano. *IEEE Micro*, 32(2):28–37, 2012. DOI: 10.1109/MM.2012.2.

[34] William J. Starke, Brian W. Thompto, Jeff A. Stuecheli, and José E. Moreira. IBM's POWER10 processor. *IEEE Micro*, 41(2):7–14, 2021. DOI: 10.1109/MM.2021.3058632.

[35] PathForward Hardware Research and Development Projects (DOE Exascale Computing Project), June 2017. URL: https://www.exascaleproject.org/research-group/pathforward/.

[36] Daiichiro Sugimoto, Yoshihiro Chikada, Junichiro Makino, Tomoyoshi Ito, Toshikazu Ebisuzaki, and Masayuki Umemura. A Special-Purpose Computer for Gravitational Many-body Problems. *Nature*, 345(6270):33–35, 1990.

[37] Tetsu Narumi, Yousuke Ohno, Noriaki Okimoto, Atsushi Suenaga, Ryoko Yanai, and Makoto Taiji. A High-Speed Special-Purpose Computer for Molecular Dynamics Simulations: MDGRAPE-3. In *NIC Workshop*, volume 34, pages 29–36. Citeseer, 2006.

[38] Yousuke Ohno, Eiji Nishibori, Tetsu Narumi, Takahiro Koishi, Tahir H. Tahirov, Hideo Ago, Masashi Miyano, Ryutaro Himeno, Toshikazu Ebisuzaki, Makoto Sakata, and Makoto Taiji. A 281 Tflops calculation for X-ray protein structure analysis with special-purpose computers MDGRAPE-3. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, SC '07, New York, New York, USA, 2007. Association for Computing Machinery. DOI: 10.1145/1362622.1362698.

[39] David E. Shaw, J.P. Grossman, Joseph A. Bank, Brannon Batson, J. Adam Butts, Jack C. Chao, Martin M. Deneroff, Ron O. Dror, Amos Even, Christopher H. Fenton, Anthony Forte, Joseph Gagliardo, Gennette Gill, Brian Greskamp, C. Richard Ho, Douglas J. Ierardi, Lev Iserovich, Jeffrey S. Kuskin, Richard H. Larson, Timothy Layman, Li-Siang Lee, Adam K. Lerer, Chester Li, Daniel Killebrew, Kenneth M. Mackenzie, Shark Yeuk-Hai Mok, Mark A. Moraes, Rolf Mueller, Lawrence J. Nociolo, Jon L. Peticolas, Terry Quan, Daniel Ramot, John K. Salmon, Daniele P. Scarpazza, U. Ben Schafer, Naseer Siddique, Christopher W. Snyder, Jochen Spengler, Ping Tak Peter Tang, Michael Theobald, Horia Toma, Brian Towles, Benjamin Vitale, Stanley C. Wang, and Cliff Young. Anton 2: Raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 41–53, 2014. DOI: 10.1109/SC.2014.9.

[40] David E. Shaw, Ron O. Dror, John K. Salmon, J. P. Grossman, Kenneth M. Mackenzie, Joseph A. Bank, Cliff Young, Martin M. Deneroff, Brannon Batson, Kevin J. Bowers, Edmond Chow, Michael P. Eastwood, Douglas J. Ierardi, John L. Klepeis, Jeffrey S. Kuskin, Richard H. Larson, Kresten Lindorff-Larsen, Paul Maragakis, Mark A. Moraes, Stefano Piana, Yibing Shan, and Brian Towles. Millisecond-scale molecular dynamics simulations on Anton. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, New York, New York, USA, 2009. Association for Computing Machinery. DOI: 10.1145/1654059.1654099.

[41] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, Amsterdam, The Netherlands, 2016. DOI: 10.3233/978-1-61499-649-1-87.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library, 2019. `arXiv:arXiv:1912.01703`.

[43] SiFive TileLink Specification Version 1.8.0, August 2019. URL: https://sifive.cdn.prismic.io/sifive%2Fcab05224-2df1-4af8-adee-8d9cba3378cd_tilelink-spec-1.8.0.pdf.

[44] OpenMP Architecture Review Board. OpenMP application programming interface version 5.1.

[45] The OpenACCR application programming interface version 3.0, November 2019. URL: https://www.openacc.org/sites/default/files/inline-images/Specification/OpenACC.3.0.pdf.

[46] James Reinders, Ben Ashbaugh, James Brodman, Michael Kinsner, John Pennycook, and Xinmin Tian. *Data Parallel C++: Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL*. Springer Nature, 2021.

[47] Shirley Browne, Jack Dongarra, Eric Grosse, and Tom Rowan. The Netlib mathematical software repository. *D-lib Magazine*, 1(9), 1995.

[48] Matteo Frigo and Steven G Johnson. FFTW: Fastest Fourier transform in the west. *Astrophysics Source Code Library*, pages ascl–1201, 2012.

[49] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. AI for science: Report on the Department of Energy (DOE) town halls on artificial intelligence (AI) for science. February 2020. URL: https://www.osti.gov/biblio/1604756, DOI: 10.2172/1604756.

[50] Linux Foundation: Confidential Computing Consortium. URL: https://confidentialcomputing.io.

[51] Ivy B. Peng, Maya B. Gokhale, and Eric W. Green. System evaluation of the Intel Optane byte-addressable NVM. In ACM, editor, *International Symposium on Memory Systems MEMSYS19*, 2019.

[52] Paul Dempsey. Mounting Fiji: How AMD realized the first volume interposer. *Tech Design Forum*, December 2015. URL: https://www.techdesignforums.com/practice/technique/mounting-fiji-how-amd-realized-the-first-volume-interposer/.

[53] Endong Wang, Qing Zhang, Bo Shen, Guangyong Zhang, Xiaowei Lu, Qing Wu, and Yajuan Wang. *High-Performance Computing on the Intel Xeon Phi: How to Fully Exploit MIC Architectures*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[54] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, jun 2017. DOI: 10.1145/3140659.3080246.

[55] Brian Bailey. HBM takes on a much bigger role. *Semiconductor Engineering*, May 2021. URL: https://semiengineering.com/hbm-takes-on-a-much-bigger-role/.

[56] Mitsuhisa Sato, Yutaka Ishikawa, Hirofumi Tomita, Yuetsu Kodama, Tetsuya Odajima, Miwako Tsuji, Hisashi Yashiro, Masaki Aoki, Naoyuki Shida, Ikuo Miyoshi, Kouichi Hirai, Atsushi Furuya, Akira Asato, Kuniki Morita, and Toshiyuki Shimizu. Co-design for A64FX manycore processor and "Fugaku". In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2020. DOI: 10.1109/SC41405.2020.00051.

[57] Arijit Biswas and Sailesh Kottapalli. Next-gen Intel Xeon CPU—Sapphire Rapids. In *Proceedings of Hot Chips 33*, August 2021.

[58] OpenCAPI Consortium. OpenCAPI specifications. URL: https://opencapi.org/technical/specifications/.

[59] GenZ Consortium. GenZ specifications. URL: https://genzconsortium.org/.

[60] CXL Consortium. Compute Express Link (CXL) specifications. URL: https://www.computeexpresslink.org/.

[61] Yuanwei Fang, Chen Zou, Aaron J. Elmore, and Andrew A. Chien. UDP: A programmable accelerator for extract-transform-load workloads and more. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, page 55–68, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3123939.3123983.

[62] Scott Lloyd and Maya Gokhale. In-memory data rearrangement for irregular, data-intensive computing. *Computer*, 48(8):18–25, August 2015. DOI: 10.1109/MC.2015.230.

[63] Yuanwei Fang, Tung T. Hoang, Michela Becchi, and Andrew A. Chien. Fast support for unstructured data processing: The Unified Automata Processor. In *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, page 533–545, New York, New York, USA, 2015. Association for Computing Machinery. DOI: 10.1145/2830772.2830809.

[64] Scott Lloyd and Maya Gokhale. Near memory key/value lookup acceleration. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '17, page 26–33, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3132402.3132434.

[65] Jian Weng, Sihao Liu, Vidushi Dadu, Zhengrong Wang, Preyas Shah, and Tony Nowatzki. *DSAGEN: Synthesizing Programmable Spatial Accelerators*, page 268–281. IEEE Press, 2020. URL: https://doi.org/10.1109/ISCA45697.2020.00032.

[66] Vidushi Dadu, Jian Weng, Sihao Liu, and Tony Nowatzki. Towards general purpose acceleration by exploiting common data-dependence forms. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 924–939, New York, New York, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3352460.3358276.

[67] Yuanwei Fang, Chen Zou, and Andrew A. Chien. Accelerating raw data analysis with the ACCORDA software and hardware architecture. *Proceedings of the VLDB Endowment*, 12(11):1568–1582, July 2019. DOI: 10.14778/3342263.3342634.

[68] NVIDIA. NVIDIA A100 data sheet. Technical report, June 2021. URL: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf.

[69] NVIDIA. NVIDIA V100 tensor core GPU data sheet. Technical report, January 2020. URL: https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf.

[70] Arjit Biswas. Sapphire Rapids (Xeon CPU). In *Hot Chips 33*, August 2021.

[71] Young-Cheon Kwon, Suk Han Lee, Jaehoon Lee, Sang-Hyuk Kwon, Je Min Ryu, Jong-Pil Son, O Seongil, Hak-Soo Yu, Haesuk Lee, Soo Young Kim, Youngmin Cho, Jin Guk Kim, Jongyoon Choi, Hyun-Sung Shin, Jin Kim, BengSeng Phuah, HyoungMin Kim, Myeong Jun Song, Ahn Choi, Daeho Kim, SooYoung Kim, Eun-Bong Kim, David Wang, Shinhaeng Kang, Yuhwan Ro, Seungwoo Seo, JoonHo Song, Jaeyoun Youn, Kyomin Sohn, and Nam Sung Kim. 25.4 a 20nm 6GB function-in-memory DRAM, based on HBM2 with a 1.2TFLOPS programmable computing unit using bank-level parallelism, for machine learning applications. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 350–352, 2021. DOI: 10.1109/ISSCC42613.2021.9365862.

[72] Sukhan Lee, Shin-haeng Kang, Jaehoon Lee, Hyeonsu Kim, Eojin Lee, Seungwoo Seo, Hosang Yoon, Seungwon Lee, Kyounghwan Lim, Hyunsung Shin, Jinhyun Kim, O Seongil, Anand Iyer, David Wang, Kyomin Sohn, and Nam Sung Kim. Hardware architecture and software stack for PIM based on commercial DRAM technology: Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 43–56, 2021. DOI: 10.1109/ISCA52012.2021.00013.

[73] Liu Ke, Udit Gupta, Benjamin Youngjae Cho, David Brooks, Vikas Chandra, Utku Diril, Amin Firoozshahian, Kim Hazelwood, Bill Jia, Hsien-Hsin S. Lee, Meng Li, Bert Maher, Dheevatsa Mudigere, Maxim Naumov, Martin Schatz, Mikhail Smelyanskiy, Xiaodong Wang, Brandon Reagen, Carole-Jean Wu, Mark Hempstead, and Xuan Zhang. *RecNMP: Accelerating Personalized Recommendation with Near-Memory Processing*, page 790–803. IEEE Press, 2020. DOI: 10.1109/ISCA45697.2020.00070.

[74] Liu Ke, Xuan Zhang, Jinin So, Jong-Geon Lee, Shin-Haeng Kang, Sukhan Lee, Songyi Han, Yeongon Cho, Jin Hyun Kim, Yongsuk Kwon, Kyungsoo Kim, Jin Jung, Ilkwon Yun, Sung Joo Park, Hyunsun Park, Joonho

Song, Jeonghyeon Cho, Kyomin Sohn, Nam Sung Kim, and Hsien-Hsin Sean Lee. Near-memory processing in action: Accelerating personalized recommendation with AxDIMM. *IEEE Micro*, page 1, 2021. DOI: 10.1109/MM.2021.3097700.

[75] Heterogeneous System Architecture Foundation. URL: http://hsafoundation.com/.

[76] Zvika Guz, Harry Li, Anahita Shayesteh, and Vijay Balakrishnan. NVMe-over-fabrics performance characterization and the path to low-overhead flash disaggregation. In *Proceedings of the 10th ACM International Systems and Storage Conference*, pages 1–9, 2017.

[77] S. Venkatesan and M. Aoulaiche. Overview of 3D NAND technologies and outlook (invited paper). In *2018 Non-Volatile Memory Technology Symposium (NVMTS)*, pages 1–5, October 2018. DOI: 10.1109/NVMTS.2018.8603104.

[78] F. T. Hady, A. Foong, B. Veal, and D. Williams. Platform storage performance with 3D XPoint technology. *Proceedings of the IEEE*, 105(9):1822–1833, September 2017. DOI: 10.1109/JPROC.2017.2731776.

[79] Glenn K. Lockwood, Damien Hazen, Quincey Koziol, R. Shane Canon, Katie Antypas, Jan Balewski, Nicholas Balthaser, Wahid Bhimji, James Botts, Jeff Broughton, Tina L. Butler, Gregory F. Butler, Ravi Cheema, Christopher Daley, Tina Declerck, Lisa Gerhardt, Wayne E. Hurlbert, Kristy A. Kallback-Rose, Stephen Leak, Jason Lee, Rei Lee, Jianlin Liu, Kirill Lozinskiy, David Paul, Prabhat, Cory Snavely, Jay Srinivasan, Tavia Stone Gibbins, and Nicholas J. Wright. Storage 2020: A vision for the future of HPC storage. Technical Report LBNL-2001072, National Energy Research Scientific Computing Center, November 2017. DOI: 10.2172/1632124.

[80] Geoffrey Fox, Judy Qiu, and Gregor von Laszewski. Introduction to cloud computing and data engineering, Intelligent Systems Engineering course E222, 2018. URL: http://dsc.soic.indiana.edu/presentations/E222-2018-CloudIntroUpdatedDec2018.pptx.

[81] Matthieu Dorier, Philip Carns, Kevin Harms, Robert Latham, Robert Ross, Shane Snyder, Justin Wozniak, Samuel K. Gutiérrez, Bob Robey, Brad Settlemyer, Galen Shipman, Jerome Soumagne, James Kowalkowski, Marc Paterno, and Saba Sehrish. Methodology for the rapid development of scalable HPC data services. In *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage Data Intensive Scalable Computing Systems (PDSW-DISCS)*, pages 76–87, November 2018. DOI: 10.1109/PDSW-DISCS.2018.00013.

[82] Michael A Sevilla, Noah Watkins, Ivo Jimenez, Peter Alvaro, Shel Finkelstein, Jeff LeFevre, and Carlos Maltzahn. Malacology: A programmable storage system. In *Proceedings of the Twelfth European Conference on Computer Systems*, pages 175–190. ACM, 2017.

[83] Michael A. Sevilla, Carlos Maltzahn, Peter Alvaro, Reza Nasirigerdeh, Bradley W. Settlemyer, Danny Perez, David Rich, and Galen M. Shipman. Programmable caches with a data management language and policy engine. In *Proceedings of the International Symposium on Cluster, Cloud and Grid Computing (CCGrid'18)*, 2018. DOI: 10.1109/CCGRID.2018.00035.

[84] Anthony Kougkas, Hariharan Devarajan, Jay Lofstead, and Xian-He Sun. LABIOS: A distributed label-based I/O system. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 13–24. ACM, 2019.

[85] Craig Ulmer, Shyamali Mukherjee, Gary Templet, Scott Levy, Jay Lofstead, Patrick Widener, Todd Kordenbrock, and Margaret Lawson. Faodel: Data management for next-generation application workflows. In *Proceedings of the 9th Workshop on Scientific Cloud Computing*, page 8. ACM, 2018.

[86] Jungwon Kim, Seyong Lee, and Jeffrey S. Vetter. PapyrusKV: A high-performance parallel key-value store for distributed NVM architectures. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3126908.3126943.

[87] André Brinkmann, Kathryn Mohror, Weikuan Yu, Philip Carns, Toni Cortes, Scott A Klasky, Alberto Miranda, Franz-Josef Pfreundt, Robert B Ross, Marc-André Vef, et al. Ad hoc file systems for high-performance computing. *Journal of Computer Science and Technology*, 35(1):4–26, 2020.

[88] Robert Ross, Lee Ward, Philip Carns, Gary Grider, Scott Klasky, Quincey Kozio, Glenn K. Lockwood, Kathryn Mohror, Bradley Settlemyer, and Matthew Wolf. Storage systems and I/O: Organizing, storing, and accessing data for scientific discovery. Technical Report 1491994, US DOE Office of Science, Advanced Scientific Computing Research, May 2019. DOI: 10.2172/1491994.

[89] Glenn K. Lockwood, Kirill Lozinskiy, Lisa Gerhardt, Ravi Cheema, Damian Hazen, and Nicholas J. Wright. Designing an all-flash Lustre file system for the 2020 NERSC Perlmutter system. In *Proceedings of the 2019 Cray User Group (CUG)*, May 6–9, 2019. URL: https://cug.org/proceedings/cug2019_proceedings/includes/files/pap131s2-file1.pdf.

[90] Zhen Liang, Johann Lombardi, Mohamad Chaarawi, and Michael Hennecke. DAOS: A scale-out high performance storage stack for storage class memory. In Dhabaleswar K. Panda, editor, *Supercomputing Frontiers*, pages 40–54. Springer, 2020. DOI: 0.1007/978-3-030-48842-0_3.

[91] Peter Braam and Dave Bonnie. Campaign storage. In *33rd International Conference on Massive Storage Systems and Technology (MSST 2017)*, pages 1–8, May 15–19, 2017. URL: https://storageconference.us/2017/Papers/CampaignStorage.pdf.

[92] Robert B. Ross, George Amvrosiadis, Philip Carns, Charles D. Cranor, Matthieu Dorier, Kevin Harms, Greg Ganger, Garth Gibson, Samuel K. Gutierrez, Robert Latham, Bob Robey, Dana Robinson, Bradley Settlemyer, Galen Shipman, Shane Snyder, Jerome Soumagne, and Qing Zheng. Mochi: Composing data services for high-performance computing environments. *Journal of Computer Science and Technology*, 35(1):121–144, January 2020. DOI: 10.1007/s11390-020-9802-0.

[93] Teng Wang, Kathryn Mohror, Adam Moody, Kento Sato, and Weikuan Yu. An ephemeral burst-buffer file system for scientific applications. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 807–818, 2016. DOI: 10.1109/SC.2016.68.

[94] Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, and Steven Swanson. Basic performance measurements of the Intel Optane DC persistent memory module, 2019. URL: https://arxiv.org/abs/1903.05714, DOI: 10.48550/ARXIV.1903.05714.

[95] Robert Ross, Lee Ward, Philip Carns, Gary Grider, Scott Klasky, Quincey Koziol, Glenn K. Lockwood, Kathryn Mohror, Bradley Settlemyer, and Matthew Wolf. Storage systems and I/O: Organizing, storing, and accessing data for scientific discovery. In *Report for the DOE ASCR Workshop on Storage Systems and I/O*, 2019.

[96] Tom Peterka, Deborah Bard, Janine Bennett, E. Wes Bethel, Ron Oldfield, Line Pouchard, Christine Sweeney, and Matthew Wolf. ASCR workshop on in situ data management: Enabling scientific discovery from diverse data sources. Technical Report 1493245, US DOE Office of Science (SC), February 4, 2019. DOI: 10.2172/1493245.

[97] Jeffrey S. Vetter, Ron Brightwell, Maya Gokhale, Pat McCormick, Rob Ross, John Shalf, Katie Antypas, David Donofrio, Anshu Dubey, Travis Humble, Catherine Schuman, Brian Van Essen, Shinjae Yoo, Alex Aiken, David Bernholdt, Suren Byna, Kirk Cameron, Frank Cappello, Barbara Chapman, Andrew Chien, Mary Hall, Rebecca Hartman-Baker, Zhiling Lan, Michael Lang, John Leidel, Sherry Li, Robert Lucas, John Mellor-Crummey, Paul Peltz Jr., Thomas Peterka, Michelle Strout, and Jeremiah Wilke. Extreme Heterogeneity 2018: Productive computational science in the era of extreme heterogeneity. Report for DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity. Technical Report 1494112, January 23–25 2018. URL: https://www.osti.gov/servlets/purl/1494112, DOI: 10.2172/1473756.

[98] Terry Jones, Michael J. Brim, Geoffroy Vallee, Benjamin Mayer, Aaron Welch, Tonglin Li, Michael Lang, Latchesar Ionkov, Douglas Otstott, Ada Gavrilovska, Greg Eisenhauer, Thaleia Doudali, and Pradeep Fernando. UNITY: Unified memory and file space. In *Proceedings of the 7th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS 2017)*, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3095770.3095776.

[99] Peter Corbett, Dror Feitelson, Sam Fineberg, Yarsun Hsu, Bill Nitzberg, Jean-Pierre Prost, Marc Snirt, Bernard Traversat, and Parkson Wong. *Overview of the MPI-IO Parallel I/O Interface*, pages 127–146. Springer, Boston, Massachusetts, USA, 1996. DOI: 10.1007/978-1-4613-1401-1_5.

[100] Rajeev Thakur, William Gropp, and Ewing Lusk. On implementing MPI-IO portably and with high performance. In *Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems*, IOPADS '99, pages 23–32, New York, New York, USA, 1999. Association for Computing Machinery. DOI: 10.1145/301816.301826.

[101] Torsten Hoefler, Rolf Rabenseifner, Hubert Ritzdorf, Bronis R. de Supinski, Rajeev Thakur, and Jesper Larsson Träff. The scalable process topology interface of MPI 2.2. *Concurrency and Computation: Practice and Experience*, 23(4):293–310, 2011.

[102] Ivy Peng, Roger Pearce, and Maya Gokhale. On the memory underutilization: Exploring disaggregated memory on HPC systems. In *2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 183–190. IEEE, 2020.

[103] François Broquedis, Jérôme Clet-Ortega, Stéphanie Moreaud, Nathalie Furmento, Brice Goglin, Guillaume Mercier, Samuel Thibault, and Raymond Namyst. hwloc: A generic framework for managing hardware affinities in HPC applications. In *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 180–186. IEEE, 2010.

[104] Michael Kenneth Lang. Simplified interface to complex memory (SICM) FY19 project review. Technical Report LA-UR-19-30136, Los Alamos National Laboratory, October 7, 2019. URL: https://www.osti.gov/biblio/1569724-simplified-interface-complex-memory-sicm-fy19-project-review, DOI: 10.2172/1569724.

[105] M. Ben Olson, Brandon Kammerdiener, Michael R. Jantz, Kshitij A. Doshi, and Terry Jones. Portable application guidance for complex memory systems. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '19, pages 156–166, New York, New York, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3357526.3357575.

[106] Jingcao Hu and Radu Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, pages 233–239, 2003.

[107] Yonghong Yan, Jisheng Zhao, Yi Guo, and Vivek Sarkar. Hierarchical place trees: A portable abstraction for task parallelism and data movement. In *International Workshop on Languages and Compilers for Parallel Computing*, pages 172–187. Springer, 2009.

[108] Didem Unat, Anshu Dubey, Torsten Hoefler, John Shalf, Mark Abraham, Mauro Bianco, Bradford L. Chamberlain, Romain Cledat, H. Carter Edwards, Hal Finkel, Karl Fuerlinger, Frank Hannig, Emmanuel Jeannot, Amir Kamil, Jeff Keasler, Paul H. J. Kelly, Vitus Leung, Hatem Ltaief, Naoya Maruyama, Chris J. Newburn, and Miquel Pericás. Trends in data locality abstractions for HPC systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):3007–3020, 2017. DOI: 10.1109/TPDS.2017.2703149.

[109] Millad Ghane, Sunita Chandrasekaran, and Margaret S. Cheung. Towards a portable hierarchical view of distributed shared memory systems: Challenges and solutions. In *Proceedings of the Eleventh International Workshop on Programming Models and Applications for Multicores and Manycores*, pages 1–10, 2020.

[110] Mike Folk, Albert Cheng, and Kim Yates. HDF5: A file format and I/O library for high performance computing applications. In *Proceedings of Supercomputing*, pages 5–33, 1999.

[111] Peter Braam. The Lustre storage architecture, 2019. URL: https://arxiv.org/abs/1903.01955, DOI: 10.48550/ARXIV.1903.01955.

[112] Tong Jin, Fan Zhang, Qian Sun, Hoang Bui, Melissa Romanus, Norbert Podhorszki, Scott Klasky, Hemanth Kolla, Jacqueline Chen, Robert Hager, Choong-Seock Chang, and Manish Parashar. Exploring data staging across deep memory hierarchies for coupled data intensive simulation workflows. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 1033–1042, 2015. DOI: 10.1109/IPDPS.2015.50.

[113] Jeffrey S Vetter and Sparsh Mittal. Opportunities for nonvolatile memory systems in extreme-scale high-performance computing. *Computing in Science & Engineering*, 17(2):73–82, 2015.

[114] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. A durable and energy efficient main memory using phase change memory technology. *ACM SIGARCH computer architecture news*, 37(3):14–23, 2009.

[115] Moinuddin K. Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 14–23, 2009. DOI: 10.1145/1669112.1669117.

[116] Norman Jouppi, Cliff Young, Nishant Patil, and David Patterson. Motivation for and evaluation of the first tensor processing unit. *IEEE Micro*, 38(3):10–19, 2018. DOI: 10.1109/MM.2018.032271057.

[117] Cerebras, Inc. Homepage—Cerebras. Accessed 11-Feb-2022. URL: https://cerebras.net/.

[118] Mythic, Inc. Mythic—accelerating AI. Accessed 11-Feb-2022. URL: https://www.mythic-ai.com/.

[119] Groq, Inc. Home—Groq. Accessed 11-Feb-2022. URL: https://groq.com/.

[120] Graphcore, Ltd. Graphcore: Accelerating machine learning for a world of intelligent machines. Accessed 11-Feb-2022. URL: https://www.graphcore.ai/.

[121] Ameet V. Joshi. Amazon's machine learning toolkit: Sagemaker. In *Machine Learning and Artificial Intelligence*, pages 233–243. Springer, 2020.

[122] SambaNova Systems, Inc. Sambanova Systems | creating the AI enabled enterprise. Accessed 14-Feb-2022. URL: https://sambanova.ai/.

[123] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The SpiNNaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[124] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. URL: https://www.science.org/doi/abs/10.1126/science.1254642, DOI: 10.1126/science.1254642.

[125] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[126] Intel Newsroom. *Intel Scales Neuromorphic Research System to 100 Million Neurons*, March 18, 2020. Accessed June 13th, 2020. URL: https://newsroom.intel.com/news/intel-scales-neuromorphic-research-system-100-million-neurons/#gs.7xo39i.

[127] Sebastian Höppner and Christian Mayr. SpiNNaker2–towards extremely efficient digital neuromorphics and multi-scale brain emulation. In *Proc. NICE*, 2018.

[128] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Transactions on Biomedical Circuits and Systems*, 12(1):106–122, 2017.

[129] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[130] Stephen Brink, Stephen Nease, Paul Hasler, Shubha Ramakrishnan, Richard Wunderlich, Arindam Basu, and Brian Degnan. A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Transactions on Biomedical Circuits and Systems*, 7(1):71–81, 2012.

[131] Suma George, Sihwan Kim, Sahil Shah, Jennifer Hasler, Michelle Collins, Farhan Adil, Richard Wunderlich, Stephen Nease, and Shubha Ramakrishnan. A programmable and configurable mixed-mode FPAA SoC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(6):2253–2261, 2016.

[132] Chetan Singh Thakur, Jamal Lottier Molin, Gert Cauwenberghs, Giacomo Indiveri, Kundan Kumar, Ning Qiao, Johannes Schemmel, Runchun Wang, Elisabetta Chicca, Jennifer Olson Hasler, Jae-sun Seo, Shimeng Yu, Yu Cao, André van Schaik, and Ralph Etienne-Cummings. Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Frontiers in Neuroscience*, 12, 2018. URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00891, DOI: 10.3389/fnins.2018.00891.

[133] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware. *arXiv:1705.06963*, 2017.

[134] John M. Shalf and Robert Leland. Computing beyond Moore's Law. *Computer*, 48(12):14–23, 2015. DOI: 10.1109/MC.2015.374.

[135] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno. 20k-spin Ising chip for combinational optimization problem with CMOS annealing. In *2015 IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*, pages 1–3, February 2015. DOI: 10.1109/ISSCC.2015.7063111.

[136] C. Yoshimura, M. Yamaoka, H. Aoki, and H. Mizuno. Spatial computing architecture using randomness of memory cell stability under voltage control. In *2013 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4, September 2013. DOI: 10.1109/ECCTD.2013.6662276.

[137] Fujitsu. Digital Annealer, May 2018. Accessed: 02/26/2019. URL: http://www.fujitsu.com/global/digitalannealer/.

[138] Francesco Caravelli. Asymptotic behavior of memristive circuits and combinatorial optimization, 2017. URL: https://arxiv.org/abs/1712.07046, arXiv:arXiv:1712.07046.

[139] Massimiliano Di Ventra Fabio L. Traversa. MemComputing integer linear programming, 2018. URL: https://arxiv.org/abs/1808.09999, arXiv:arXiv:1808.09999.

[140] Natalia G. Berloff, Matteo Silva, Kirill Kalinin, Alexis Askitopoulos, Julian D. Töpfer, Pasquale Cilibrizzi, Wolfgang Langbein, and Pavlos G. Lagoudakis. Realizing the classical XY Hamiltonian in polariton simulators. *Nature Materials*, 16(11):1120–1126, 2017. DOI: 10.1038/nmat4971.

[141] David Dung, Christian Kurtscheid, Tobias Damm, Julian Schmitt, Frank Vewinger, Martin Weitz, and Jan Klaers. Variable potentials for thermalized light and coupled condensates. *Nature Photonics*, 11(9):565, 2017.

[142] Micha Nixon, Eitan Ronen, Asher A. Friesem, and Nir Davidson. Observing geometric frustration with thousands of coupled lasers. *Physical Review Letters*, 110(18):184102, 2013.

[143] Shoko Utsunomiya, Kenta Takata, and Yoshihisa Yamamoto. Mapping of Ising models onto injection-locked laser systems. *Optics Express*, 19(19):18091–18108, 2011.

[144] Peter L. McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, Robert L. Byer, M. M. Fejer, Hideo Mabuchi, and Yoshihisa Yamamoto. A fully programmable 100-spin coherent Ising machine with all-to-all connections. *Science*, 354(6312):614–617, 2016. URL: https://www.science.org/doi/abs/10.1126/science.aah5178, DOI: 10.1126/science.aah5178.

[145] Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L. McMahon, Takeshi Umeki, Koji Enbutsu, Osamu Tadanaga, Hirokazu Takenouchi, Kazuyuki Aihara, Ken-ichi Kawarabayashi, Kyo Inoue, Shoko Utsunomiya, and Hiroki Takesue. A coherent Ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016. URL: http://science.sciencemag.org/content/354/6312/603, DOI: 10.1126/science.aah4243.

[146] D. Kielpinski, R. Bose, J. Pelc, T. Van Vaerenbergh, G. Mendoza, N. Tezak, and R. G. Beausoleil. Information processing with large-scale optical integrated circuits. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–4, Oct 2016. DOI: 10.1109/ICRC.2016.7738704.

[147] Naren Ramakrishnan, Upinder S. Bhalla, and John J. Tyson. Computing with proteins. *Computer*, 42(1):47–56, 2009.

[148] Alireza Goudarzi, Matthew R. Lakin, and Darko Stefanovic. DNA reservoir computing: A novel molecular computing approach. In David Soloveichik and Bernard Yurke, editors, *DNA Computing and Molecular Programming*, pages 76–89, Cham, 2013. Springer International Publishing.

[149] Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, November 2021. DOI: 10.1145/3467017.

[150] Jacques Pienaar. MLIR in TensorFlow ecosystem. In *2nd Compilers For Machine Learning (C4ML) workshop, International Symposium on Code Generation and Optimization*, San Diego, California, USA, February 23, 2020. Presentation available from https://research.google/pubs/pub48996/.

[151] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A graph placement methodology for fast chip design. 594(7862):207–212, 2021. DOI: 10.1038/s41586-021-03544-w.

[152] Jeremy Thomas. LLNL pairs world's largest computer chip from Cerebras with Lassen to advance machine learning, AI research. Accessed: 2021-06-16. URL: https://www.llnl.gov/news/llnl-pairs-worlds-largest-computer-chip-cerebras-lassen-advance-machine-learning-ai-research.

[153] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86:032324:1–48, September 2012. DOI: 10.1103/PhysRevA.86.032324.

[154] Jeremy Hsu. CES 2018: Intel's 49-qubit chip shoots for quantum supremacy. *IEEE Spectrum*, January 8, 2018. Accessed 18-Mar-2022. URL: https://spectrum.ieee.org/intels-49qubit-chip-aims-for-quantum-supremacy.

[155] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo. Scalable quantum circuit and control for a superconducting surface code. *Physical Review Applied*, 8:034021:1–7, September 2017. DOI: 10.1103/PhysRevApplied.8.034021.

[156] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: A survey. *Journal of Combinatorial Optimization*, 28(1):58–81, July 2014. DOI: 10.1007/s10878-014-9734-0.

[157] Oylum Şeker, Neda Tanoumand, and Merve Bodur. Digital Annealer for quadratic unconstrained binary optimization: A comparative performance analysis, 2020. arXiv:arXiv:2012.12264.

[158] Ryan Hamerly, Takahiro Inagaki, Peter L. McMahon, Davide Venturelli, Alireza Marandi, Tatsuhiro Onodera, Edwin Ng, Carsten Langrock, Kensuke Inaba, Toshimori Honjo, Koji Enbutsu, Takeshi Umeki, Ryoichi Kasahara, Shoko Utsunomiya, Satoshi Kako, Ken ichi Kawarabayashi, Robert L. Byer, Martin M. Fejer, Hideo Mabuchi, Dirk Englund, Eleanor Rieffel, Hiroki Takesue, and Yoshihisa Yamamoto. Experimental investigation of performance differences between coherent Ising machines and a quantum annealer. *Science Advances*, 5(5):eaau0823:1–10, May 24, 2019. DOI: 10.1126/sciadv.aau0823.

[159] Kelly Boothby, Colin Enderud, Trevor Lanting, Reza Molavi, Nicholas Tsai, Mark H. Volkmann, Fabio Altomare, Mohammad H. Amin, Michael Babcock, Andrew J. Berkley, Catia Baron Aznar, Martin Boschnak, Holly Christiani, Sara Ejtemaee, Bram Evert, Matthew Gullen, Markus Hager, Richard Harris, Emile Hoskinson, Jeremy P. Hilton, Kais Jooya, Ann Huang, Mark W. Johnson, Andrew D. King, Eric Ladizinsky, Ryan Li, Allison MacDonald, Teresa Medina Fernandez, Richard Neufeld, Mana Norouzpour, Travis Oh, Isil Ozfidan, Paul Paddon, Ilya Perminov, Gabriel Poulin-Lamarre, Thomas Prescott, Jack Raymond, Mauricio Reis, Chris Rich, Aidan Roy, Hossein Sadeghi Esfahani, Yuki Sato, Ben Sheldan, Anatoly Smirnov, Loren J. Swenson, Jed Whittaker, Jason Yao, Alexander Yarovoy, and Paul I. Bunyk. Architectural considerations in the design of a third-generation superconducting quantum annealing processor, August 5, 2021. arXiv:2108.02322v1 [quant-ph]. DOI: 10.48550/arXiv.2108.02322.

[160] PsiQuantum, Inc. PsiQuantum | building the world's first useful quantum computer. Accessed 14-Feb-2022. URL: https://psiquantum.com/.

[161] Quantum Brilliance, Ltd. Quantum Brilliance—room temperature diamond quantum accelerators. Accessed 14-Feb-2022. URL: https://quantumbrilliance.com/.

[162] Elizabeth Gibney. Inside Microsoft's quest for a topological quantum computer. *Nature*, October 21, 2016. DOI: 10.1038/nature.2016.20774.

[163] Yu Wang, Gu-Yeon Wei, and David Brooks. A systematic methodology for analysis of deep learning hardware and software platforms. In Inderjit Dhillon, Dimitris Papailiopoulos, and Vivienne Sze, editors, *Proceedings of Machine Learning and Systems 2*, pages 30–43, 2020.

[164] Jeff Dean, David Patterson, and Cliff Young. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro*, 38(2):21–29, 2018.

[165] Hyoukjun Kwon, Michael Pellauer, and Tushar Krishna. Maestro: an open-source infrastructure for modeling dataflows within deep learning accelerators. *arXiv preprint arXiv:1805.02566*, 2018.

[166] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[167] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer. Timeloop: A systematic approach to DNN accelerator evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 304–315. IEEE, 2019.

[168] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. SCALE-Sim: Systolic CNN accelerator simulator. *arXiv preprint arXiv:1811.02883*, 2018.

[169] NVDLA, 2020. URL: http://nvdla.org/index.html.

[170] Vikram Narayanan, Yongzhe Huang, Gang Tan, Trent Jaeger, and Anton Burtsev. Lightweight kernel isolation with virtualization and VM functions. In *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 157–171, Lausanne, Switzerland, March 2020. DOI: 10.1145/3381052.3381328.

[171] Robert N. M. Watson, Jonathan Woodruff, Peter G. Neumann, Simon W. Moore, Jonathan Anderson, David Chisnall, Nirav Dave, Brooks Davis, Khilan Gudka, Ben Laurie, Steven J. Murdoch, Robert Norton, Michael Roe, Stacey Son, and Munraj Vadera. CHERI: A hybrid capability-system architecture for scalable software compartmentalization. In *2015 IEEE Symposium on Security and Privacy*, pages 20–37, 2015. DOI: 10.1109/SP.2015.9.

[172] Sean Peisert. Trustworthy Scientific Computing. *Communications of the ACM (CACM)*, 64(5), May 2021.

[173] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. Performance Analysis of Scientific Computing Workloads on General Purpose TEEs. In *Proceedings of the 35th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May 2021.

[174] SiFive boards. URL: https://www.sifive.com/boards.

[175] FireSim. URL: https://fires.im.

[176] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach, Nils Asmussen, Brad Beckmann, Srikant Bharadwaj, Gabe Black, Gedare Bloom, Bobby R. Bruce, Daniel Rodrigues Carvalho, Jeronimo Castrillon, Lizhong Chen, Nicolas Derumigny, Stephan Diestelhorst, Wendy Elsasser, Carlos Escuin, Marjan Fariborz, Amin Farmahini-Farahani, Pouya Fotouhi, Ryan Gambord, Jayneel Gandhi, Dibakar Gope, Thomas Grass, Anthony Gutierrez, Bagus Hanindhito, Andreas Hansson, Swapnil Haria, Austin Harris, Timothy Hayes, Adrian Herrera, Matthew Horsnell, Syed Ali Raza Jafri, Radhika Jagtap, Hanhwi Jang, Reiley Jeyapaul, Timothy M. Jones, Matthias Jung, Subash Kannoth, Hamidreza Khaleghzadeh, Yuetsu Kodama, Tushar Krishna, Tommaso Marinelli, Christian Menard, Andrea Mondelli, Miquel Moreto, Tiago Mück, Omar Naji, Krishnendra Nathella, Hoa Nguyen, Nikos Nikoleris, Lena E. Olson, Marc Orr, Binh Pham, Pablo Prieto, Trivikram Reddy, Alec Roelke, Mahyar Samani, Andreas Sandberg, Javier Setoain, Boris Shingarov, Matthew D. Sinclair, Tuan Ta, Rahul Thakur, Giacomo Travaglini, Michael Upton, Nilay Vaish, Ilias Vougioukas, William Wang, Zhengrong Wang, Norbert Wehn, Christian Weis, David A. Wood, Hongil Yoon, and Éder F. Zulian. The gem5 simulator: Version 20.0+, 2020. URL: https://arxiv.org/abs/2007.03152, DOI: 10.48550/ARXIV.2007.03152.

[177] Ayaz Akram, Venkatesh Akella, Sean Peisert, and Jason Lowe-Power. Enabling Design Space Exploration for RISC-V Secure Compute Environments. In *Proceedings of the Fifth Workshop on Computer Architecture Research with RISC-V (CARRV), (co-located with ISCA 2021),*, June 17, 2021.

[178] Jason Lowe-Power, Venkatesh Akella, Matthew K. Farrens, Samuel T. King, and Christopher J. Nitta. Position paper: A case for exposing extra-architectural state in the ISA. In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '18, New York, New York, USA, 2018. Association for Computing Machinery. DOI: 10.1145/3214292.3214300.

[179] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, Qijing Huang, Kyle Kovacs, Borivoje Nikolic, Randy Katz, Jonathan Bachrach, and Krste Asanović. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, pages 29–42, Piscataway, NJ, USA, 2018. IEEE Press. DOI: 10.1109/ISCA.2018.00014.

[180] Sean Peisert. Security in High-Performance Computing Environments. *Communications of the ACM (CACM)*, 60(9):72–80, September 2017.

[181] D. A. Agarwal, W. Dykas, and M. Robertson. DOE Cybersecurity R&D Challenges for Open Science: Developing a Roadmap and Vision, Jan. 2007. URL: https://web.archive.org/web/20091006081342/http://acs.lbl.gov/Workshops/CyberWorkshop/.

[182] D. A. Frincke, C. Catlett, F. Siebenlist, R. Strelitz, E. Talbot, and B. Worley. Transforming CyberSecurity R&D within the Department of Energy: Getting ahead of the threat, 2008. URL: https://www.osti.gov/biblio/923678-wbxW3Y/.

[183] Transforming Cyber Security Through Science (Grass Roots Community). URL: https://wiki.cac.washington.edu/display/doe/Home.

[184] Charlie Catlett, Mine Altunay, Robert Armstrong, Kirk Bailey, David Brown, Robert R. Burleson, Matt Crawford, John Daly, Don Dixon, Barbara Endicott-Popovsky, Ian Foster, Deborah Frincke, Irwin Gaines, Josh Goldfarb, Christopher Griffin, Yu Jiao, Tammy Kolda, Ron Minnich, Carmen Pancerella, Don Petravick, J. Christopher Ramming, Chad Scherrer, Anne Schur, Frank Siebenlist, Dane Skow, Adam Stone, Chris Strasburg, Richard Strelitz, Denise Sumikawa, Craig Swietlik, Edward Talbot, Troy Thompson, Keith Vanderveen, Von Welch, Joanne R. Wendelberger, Paul Whitney, Louis Wilder, and Brian Worley. A scientific research and development approach to cyber security. Technical report, December 2008. URL: https://science.osti.gov/~/media/ascr/pdf/program-documents/docs/Cyber_security_science_dec_2008.pdf.

[185] James M. Brase and David L. Brown. Modeling, simulation and analysis of complex networked systems. Technical Report LLNL-TR-412733, May 2009. URL: https://science.osti.gov/-/media/ascr/pdf/program-documents/docs/Complex_networked_systems_program_final.pdf.

[186] Sean Peisert, George Cybenko, Sushil Jajodia, David L. Brown, Christopher L. DeMarco, Paul Hovland, Sven Leyffer, Celeste Matarazzo, Stacy Prowell, Brian Tierney, and Von Welch. ASCR cybersecurity for scientific computing integrity. Technical Report LBNL-6953E, U.S. Department of Energy Office of Science, February 27, 2015. URL: https://escholarship.org/uc/item/5zp1t2j7, DOI: 10.2172/1223021.

[187] Sean Peisert, Thomas E. Potok, and Todd Jones. ASCR cybersecurity for scientific computing integrity—research pathways and ideas workshop. Technical Report LBNL-191105, U.S. Department of Energy Office of Science, Gaithersburg, Maryland, USA, September 18, 2015. URL: https://escholarship.org/uc/item/5j00n7h2, DOI: 10.2172/1236181.

[188] Trusted CI framework, 2001. URL: https://www.trustedci.org/framework.

[189] Sean Peisert et al. Open Science Cyber Risk Profile (OSCRP). URL: http://trustedci.github.io/OSCRP/.

[190] Trusted CI Fellows Program. URL: https://www.trustedci.org/fellows/.

[191] J. S. Vetter, R. Brightwell, M. Gokhale, P. McCormick, R. Ross, J. Shalf, K. Antypas, D. Donofrio, T. Humble, C. Schuman, B. Van Essen, S. Yoo, A. Aiken, D. Bernholdt, S. Byna, K. Cameron, F. Cappello, B. Chapman, A. Chien, M. Hall, R. Hartman-Baker, Z. Lan, M. Lang, J. Leidel, S. Li, R. Lucas, J. Mellor-Crummey, P. Peltz Jr., T. Peterka, M. Strout, and J. Wilke. Extreme heterogeneity 2018—productive computational science in the era of extreme heterogeneity: Report for DOE ASCR workshop on extreme heterogeneity. Technical report, USDOE Office of Science (SC) (United States), 2018. DOI: 10.2172/1473756.

[192] Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra. *MPI: The Complete Reference*, volume 1—The MPI Core. MIT press, 1998.

[193] William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir, and Mark Snir. *MPI: The Complete Reference*, volume 2—The MPI Extensions. MIT press, 1998.

[194] Gang Ren, Eric Tune, Tipp Moseley, Yixin Shi, Silvius Rus, and Robert Hundt. Google-wide profiling: A continuous profiling infrastructure for data centers. *IEEE Micro*, 30(4):65–79, 2010. DOI: 10.1109/MM.2010.68.

[195] Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q. In *SC '03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, pages 55–55, 2003. DOI: 10.1145/1048935.1050204.

[196] Kevin J. Barker, Kei Davis, Adolfy Hoisie, Darren J. Kerbyson, Mike Lang, Scott Pakin, and Jose C. Sancho. Entering the petaflop era: The architecture and performance of Roadrunner. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, pages 1–11, 2008. DOI: 10.1109/SC.2008.5217926.

[197] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings. Predictive performance and scalability modeling of a large-scale application. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*, SC '01, page 37, New York, New York, USA, 2001. Association for Computing Machinery. DOI: 10.1145/582034.582071.

[198] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, April 2009. DOI: 10.1145/1498765.1498785.

[199] Muhammad Shoaib Bin Altaf and David A. Wood. LogCA: A high-level performance model for hardware accelerators. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 375–388, 2017. DOI: 10.1145/3079856.3080216.

[200] Cosine similarity, 2021. URL: https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=1049342690.

[201] Omar Aaziz, Courtenay Vaughan, Jonathan Cook, Jeanine Cook, Jeffery Kuehn, and David Richards. Fine-grained analysis of communication similarity between real and proxy applications. In *2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pages 93–102, 2019. DOI: 10.1109/PMBS49563.2019.00016.

[202] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. Chisel: Constructing hardware in a Scala embedded language. In *DAC Design Automation Conference 2012*, pages 1212–1221, 2012. DOI: 10.1145/2228360.2228584.

[203] Adolfy Hoisie, Laura Carrington, Jon Hiller, Darren Kerbyson, Martin Schulz, Dolores Shaffer, Ankur Srivastava, Jeffrey Vetter, Bill Ward, Noel Wheeler, and Sudhakar Yalamanchili. Report of: Workshop on modeling and simulation of systems and applications. Technical Report PNNL-23916, Seattle, Washington, USA, August 13–14 2014. URL: https://www.osti.gov/biblio/1471088, DOI: 10.2172/1471088.

[204] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The Gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011. DOI: 10.1145/2024716.2024718.

[205] Ali Bakhoda, George L. Yuan, Wilson W. L. Fung, Henry Wong, and Tor M. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 163–174, 2009. DOI: 10.1109/ISPASS.2009.4919648.

[206] SST "Stake" generator, 2018. URL: https://github.com/tactcomplabs/SSTStake.

[207] Scarab, 2021. URL: https://github.com/hpsresearchgroup/scarab.

[208] Mahmoud Khairy, Zhesheng Shen, Tor M. Aamodt, and Timothy G. Rogers. Accel-Sim: An extensible simulation framework for validated GPU modeling. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 473–486, 2020. DOI: 10.1109/ISCA45697.2020.00047.

[209] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. DRAMSim2: A cycle accurate memory system simulator. *IEEE Computer Architecture Letters*, 10(1):16–19, 2011. DOI: 10.1109/L-CA.2011.4.

[210] John D. Leidel and Yong Chen. HMC-Sim: A simulation framework for Hybrid Memory Cube devices. In *2014 IEEE International Parallel Distributed Processing Symposium Workshops*, pages 1465–1474, 2014. DOI: 10.1109/IPDPSW.2014.164.

[211] Michael B. Healy and Seokin Hong. Cramsim: Controller and memory simulator. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '17, page 83–85, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3132402.3132408.

[212] Xin Wang, Misbah Mubarak, Xu Yang, Robert B. Ross, and Zhiling Lan. Trade-off study of localizing communication and balancing network traffic on a dragonfly system. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1113–1122, 2018. DOI: 10.1109/IPDPS.2018.00120.

[213] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, D. E. Shaw, John Kim, and William J. Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 86–96, 2013. DOI: 10.1109/ISPASS.2013.6557149.

[214] Scott Beamer and David Donofrio. Efficiently exploiting low activity factors to accelerate RTL simulation. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020. DOI: 10.1109/DAC18072.2020.9218632.

[215] The Structural Simulation Toolkit, 2021. URL: http://sst-simulator.org/.

[216] Manish Vachharajani, Neil Vachharajani, David A. Penry, Jason A. Blome, Sharad Malik, and David I. August. The Liberty simulation environment: A deliberate approach to high-level system modeling. *ACM Trans. Comput. Syst.*, 24(3):211–249, August 2006. DOI: 10.1145/1151690.1151691.

[217] Fact Sheet: Collaboration of Oak Ridge, Argonne, and Livermore (CORAL), 2014. URL: https://www.energy.gov/downloads/fact-sheet-collaboration-oak-ridge-argonne-and-livermore-coral.

[218] Anthony Agelastos, Benjamin Allan, Jim Brandt, Paul Cassella, Jeremy Enos, Joshi Fullop, Ann Gentile, Steve Monk, Nichamon Naksinehaboon, Jeff Ogden, Mahesh Rajan, Michael Showerman, Joel Stevenson, Narate Taerat, and Tom Tucker. The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, page 154–165. IEEE Press, 2014. DOI: 10.1109/SC.2014.18.

[219] Alessio Netti, Micha Müller, Axel Auweter, Carla Guillen, Michael Ott, Daniele Tafani, and Martin Schulz. From facility to application sensor data: Modular, continuous and holistic monitoring with DCDB. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, New York, New York, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3295500.3356191.

[220] Ron Brightwell, Ron Oldfield, Arthur B. Maccabe, and David E. Bernholdt. Hobbes: Composition and virtualization as the foundations of an extreme-scale OS/R. In *Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '13, New York, New York, USA, 2013. Association for Computing Machinery. DOI: 10.1145/2491661.2481427.

[221] Matthew L. Massie, Brent N. Chun, and David E. Culler. The Ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004. URL: https://www.sciencedirect.com/science/article/pii/S0167819104000535, DOI: https://doi.org/10.1016/j.parco.2004.04.001.

[222] Neha Agarwal and Thomas F. Wenisch. Thermostat: Application-transparent page management for two-tiered main memory. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, page 631–644, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3037697.3037706.

[223] M. Ben Olson, Brandon Kammerdiener, Michael R. Jantz, Kshitij A. Doshi, and Terry Jones. Portable application guidance for complex memory systems. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '19, page 156–166, New York, New York, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3357526.3357575.

[224] Ron Brightwell, Kurt B. Ferreira, Ryan E. Grant, Scott Levy, Jay Lofstead, Stephen L. Olivier, Kevin T. Pedretti, Andrew J. Younge, Ann Gentile, and Jim Brandt. Alamo: Autonomous lightweight allocation, management, and optimization. In Jeffrey Nichols, Becky Verastegui, Arthur 'Barney' Maccabe, Oscar Hernandez, Suzanne Parete-Koon, and Theresa Ahearn, editors, *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, pages 408–422, Cham, 2020. Springer International Publishing.

[225] Huang Ying. autonuma: Optimize memory placement for memory tiering system, August 25, 2020. URL: https://lkml.org/lkml/2020/8/24/1910.

[226] Tony Hey and Anne Trefethen. *The Data Deluge: An e-Science Perspective*, chapter 36, pages 809–824. John Wiley & Sons, Ltd, 2003. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470867167.ch36, DOI: 10.1002/0470867167.ch36.

[227] Steven R. Young, Derek C. Rose, Travis Johnston, William T. Heller, Thomas P. Karnowski, Thomas E. Potok, Robert M. Patton, Gabriel Perdue, and Jonathan Miller. Evolving deep networks using HPC. In *Proceedings of the Machine Learning on HPC Environments*, MLHPC'17, New York, New York, USA, 2017. Association for Computing Machinery. DOI: 10.1145/3146347.3146355.

[228] Bill Kramer, Greg Bauer, Brett Bode, Mike Showerman, Jeremy Enos, Aaron Saxton, Saurabh Jha, Zbigniew Kalbarczyk, Ravi Iyer, James M. Brandt, and Ann C. Gentile. Holistic measurement driven system assessment. Technical Report SAND2019-0329D, Houston, Texas, USA, January 14–17, 2019. URL: https://www.osti.gov/biblio/1592279.

[229] Trusted CI: 2020 Trustworthy Data Working Group, 2020. URL: https://www.trustedci.org/2020-trustworthy-data.

[230] Sean Peisert. An examination and survey of data confidentiality issues and solutions in academic research computing. Trusted CI Report, September 2020. URL: https://escholarship.org/uc/item/7cz7m1ws.

U.S. DEPARTMENT OF ENERGY | Office of Science