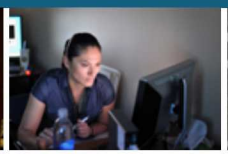
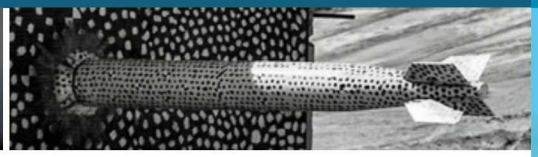
SAND2019-11168C

Sandia National Laboratories

CCR Center for Computing Research

Thoughts on Autonomous Resource Management for HPC





Department 1423 - Scalable System Software

Ron Brightwell, R&D Manager

U.S. DEPARTMENT OF ENERGY NNSA

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

Most Recent Example of Resource Management Complexity (1/3)

MVAPICH

Performance Evaluation of MPI Libraries on GPU-enabled OpenPOWER Architectures: Early Experiences

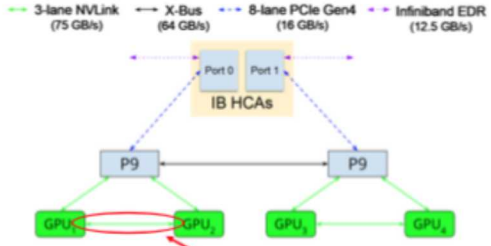
Kawthar Shafie Khorassani, Ching-Hsiang Chu, **Hari Subramoni**, and Dhabaleswar K (DK) Panda
{shafiekhorassani.1, chu.368}@osu.edu, {subramon, panda}@cse.ohio-state.edu
Network-based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University

Native Performance of Interconnects

- NVLink between CPU and GPU
  - ***BandwidthTest*** from NVIDIA CUDA sample: Multiple cudaMemcpyAsync back-to-back between system & GPU memory
  - https://github.com/NVIDIA/cuda-samples
- GPU HBM2 and NVLink between GPUs
  - ***simpleIPC*** test from NVIDIA CUDA sample: CPU processes transfer data within a GPU and between GPUs
- X-Bus
  - **STREAM** benchmark
  - https://www.cs.virginia.edu/stream/
- InfiniBand
  - InfiniBand verbs performance test (**ib_read_bw**)

POWER9 Topology (Summit)

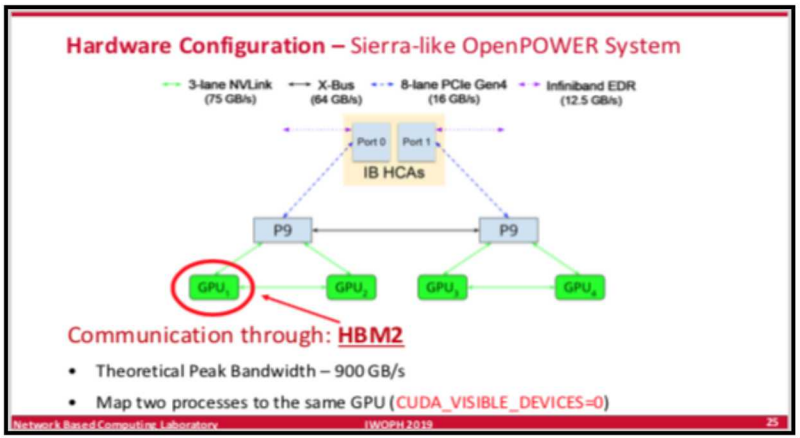
Hardware Configuration – Sierra-like OpenPOWER System

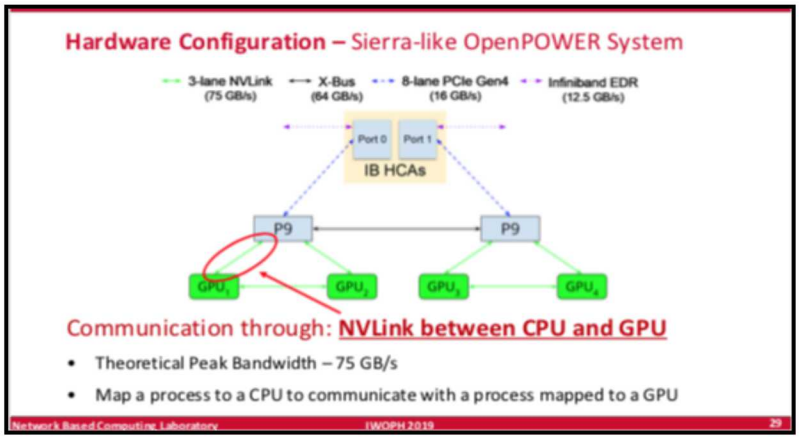


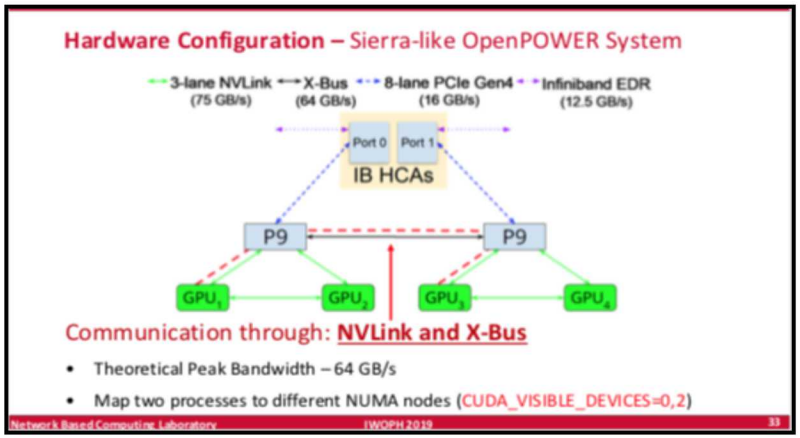
Communication through: **NVLink between GPUs**

- Theoretical Peak Bandwidth of 3-lane NVLink – 75 GB/s
- Map MPI processes to two GPUs w/ NVLink connection (CUDA_VISIBLE_DEVICES=0,1)

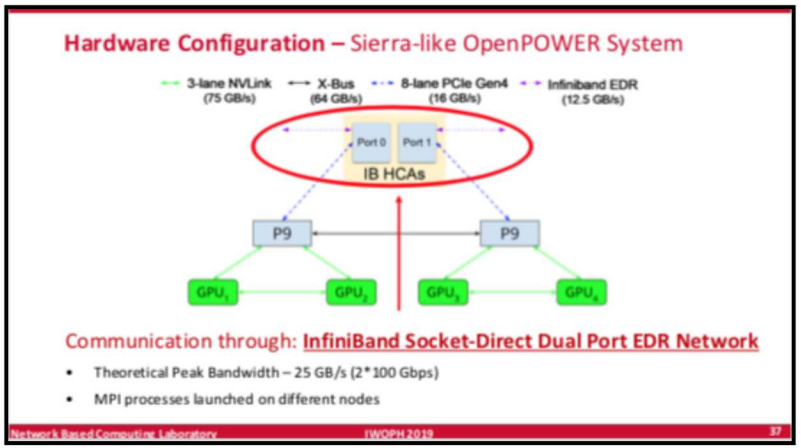
Most Recent Example of Resource Management Complexity (2/3)

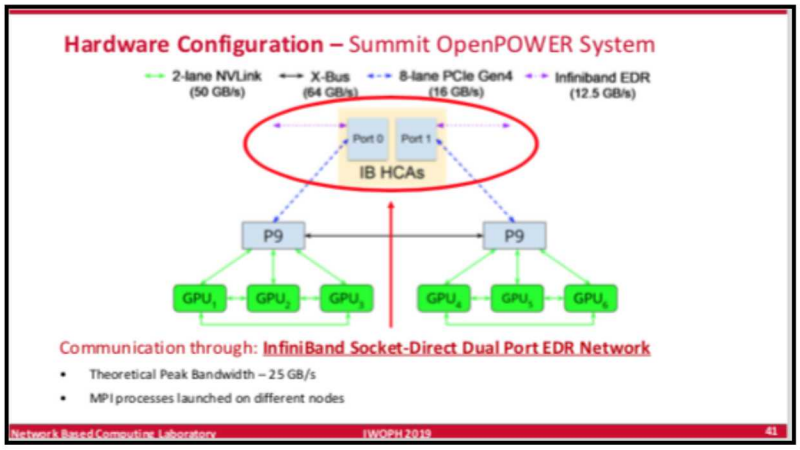






Most Recent Example of Resource Management Complexity (3/3)





Summary - Performance of Interconnects

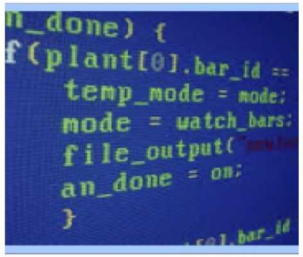
Achievable peak bandwidth of MPI libraries and fraction of peak over Interconnects on a Sierra-like GPU-enabled OpenPOWER System

- MPI Libraries perform similarly for **NVLink GPU-GPU** and **X-Bus** communications
- MVAPICH2-GDR achieves highest performance through **HBM2** and **IB** communications
- Spectrum-MPI achieves highest performance for **CPU-GPU** communication

	GPU HBM2	3-lane NVLink2 CPU-GPU	3-lane NVLink2 GPU-GPU	X-Bus	InfiniBand EDR x 2
Spectrum MPI	329 GB/s	17.52 GB/s	69.12 GB/s	40.32 GB/s	13.67 GB/s
	36.55%	23.36%	92.16%	63%	54.68%
OpenMPI	0.457 GB/s	0.74 GB/s	69.78 GB/s	32.64 GB/s	11.815 GB/s
	0.05%	0.98%	93.04%	51%	47.26%
MVAPICH2-GDR	390.88 GB/s	13.18 GB/s	69.17 GB/s	40.27 GB/s	22.47 GB/s
	43.43%	17.57%	92.22%	62.92%	89.88%

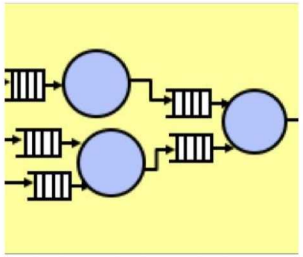
Extreme Heterogeneity Priority Research Directions (PRDs)

Maintaining and improving programmer productivity



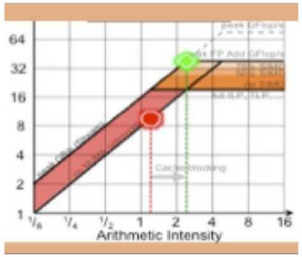
- Flexible, expressive, programming models and languages
- Intelligent, domain-aware compilers and tools
- Composition of disparate software components

Managing resources intelligently



- Automated methods using introspection and machine learning
- Optimize for performance, energy efficiency, and availability

Modeling & predicting performance



- Evaluate impact of potential system designs and application mappings
- Model-automated optimization of applications

Enabling reproducible science despite non-determinism & asynchrony

- Methods for validation on non-deterministic architectures
- Detection and mitigation of pervasive faults and errors

Facilitating execution of science workflows

- Mapping of science workflows to heterogeneous hardware & software services
- Adapting workflows & services to meet facility-level objectives through learning approaches

Autonomous Resource Allocation and Management

- HPC systems have become too complex to manage every critical resource
- Mapping the application to the machine is inefficient and tedious
- Locality management is done by hand
  - Processes are placed on specific nodes and cores by the job launcher
  - Processes are pinned to cores and never move to minimize memory traffic
  - Few (no?) mechanisms for measuring data locality
- Significant effort required to optimize just for a single machine
  - Situation becomes untenable for future heterogeneous platforms
  - Continuously increasing complexity of systems, applications and workflows
- Software stack needs to enable autonomous resource allocation and management
- Mapping the machine to the application requires dynamic discovery and adaptation
  - Determine goals of the application workflow
  - Understand how well resources are being used
  - Make informed optimization decisions and track behavior in response to decisions
  - Evolve with constantly changing cost models and temporal non-determinism
- Ideal opportunity to leverage and apply AI/ML technologies for HPC

Existing Approach

- Generalized abstractions and machine models that allow algorithm designers and application developers to create code that works reasonably well on a broad spectrum of systems
- Compilers, libraries, RTS, and OS work within the constraints of these abstractions to map the application to the underlying hardware as efficiently as possible
- Performance tools identify shortcomings in the mapping
- Refine the mapping on a per-platform basis
- Adjust the abstractions and models in response to evolving hardware
- Leverage RTS adaptivity within bounded set of resources and relatively fixed cost models

Vision for Post-Exascale Runtime Systems

- Responsible for mapping the machine to the application
- Require dynamic discovery
  - Determine the goals of the application
  - Develop knowledge on how well resources are being used
  - Make informed optimization decisions
  - Understand behavior in response to decisions
  - Consider constantly changing cost models
- Respond to elastic system and application resources
- Richer abstractions and models at the system level
- Improve the productivity of application and library developer as well as the scalability and efficiency of the system

US DARPA: Microsystems Exploration - Performant Automation of Parallel Program Assembly (PAPPA)

Solicitation No. DARPA-PA-19-04-02
Microsystems Exploration Topic (μE)
Performant Automation of Parallel Program Assembly (PAPPA)

I. Topic Description

The Defense Advanced Research Projects Agency (Exploration topic (μE) inviting submissions of innovative technical domain of massively parallel heterogeneous cor self-modifying compilers, program synthesis, and predic under the Program Announcement for Microsystems proposals in response to the technical area described here 04 and if selected, will result in an award of an Other Tra to exceed $1,000,000.

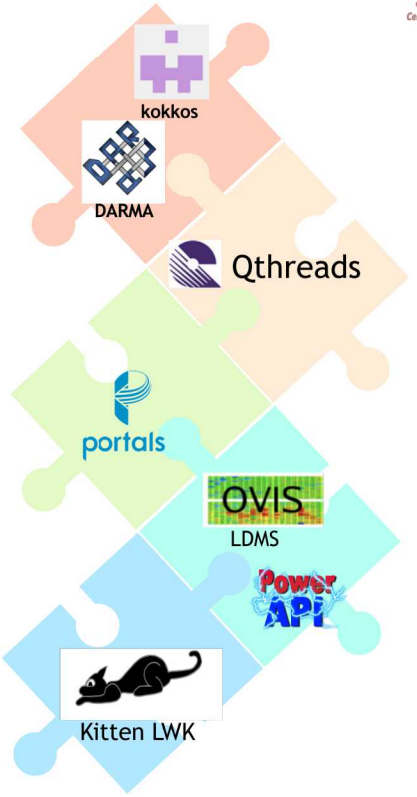
B. Objective/Scope

The PAPPA topic will explore tradeoffs between programming productivity, solution generality, and scalability to enable scientists and domain experts with no understanding of parallel programming and hardware architectures to create highly efficient performance portable programs. The goal of PAPPA is to explore the creation of compiler technology that improves the programming productivity of massively parallel and heterogeneous processing systems with 1,000,000 way parallelism by 10,000X. If successful, such performance portable compilers will significantly lower the barrier to deploying new algorithms and science on future state of the art programmable systems.

To enable productive high performance programming going forward, it is imperative that all low level tasks including task partitioning, aggregation, mapping, scheduling, resource allocation, memory management, inter process communication, synchronization, and reliability be removed as burdens from the programmer. To enable scalable and performance portable generation of executables from a highly productive programming framework, it seems likely that a completely new approach is needed. PAPPA hypothesizes that many of the challenges associated with the high performance programing of massively parallel heterogeneous systems can be overcome by accurately modeling and predicting performance of all key components within the system and applying global cost optimization to successively lower and optimize appropriately constrained domain specific programs to generate highly efficient executable code.

Post-Exascale Software Stack Components

- Builds on existing Sandia capabilities and expertise in
  - Asynchronous many-task programming systems
  - Dynamic event-driven lightweight threading runtime systems
  - High-performance networking interfaces and protocols
  - Lightweight monitoring, instrumentation, and control
  - Lightweight HPC-oriented operating systems
- Strengthens Sandia's capabilities in system software and provides opportunities for research in
  - AI/ML techniques for resource management
  - Hardware/software co-design to support introspection
  - Resilient computing infrastructure
- Creates opportunities for partnerships in
  - Continued exploration of AMT programming systems
  - System-wide resource management and workflows
  - Managing I/O and data-intensive programming systems



Issues/Concerns

Managing the memory hierarchy
- Lots of evidence that the RTS/OS are not good at this for HPC

Increasing complexity and responsibility of the RTS
- Pushing complexity to the RTS with less info

Resource requirements of the RTS
- Potentially significant overhead

Compelling application evaluation
- Applications need to exercise the advanced RTS functionality
- Implementation bias

Application performance portability
- From laptop to exascale

Transparency is in the eye of the application developer
- Need to support both experts and ambivalent

Cost of modularity
- Not all RTS services should be componentized

Ability to constrain the problem
- Too many hardware and application "knobs"

Performance portability of the RTS
- Not any easier to solve than application performance portability

Dependence on hardware advancements
- Inability to demonstrate compelling results on current systems

Lack of standard low-level network API
- Fundamental issue for RTS communication

HPC market pressure
- Influence of non-HPC "solutions"

Amount of asynchrony
- Ability of algorithms to reduce global operations

Jitter
- Will lack of balance absorb noise effects

Mechanisms to support event-driven capability
- More efficient ways to enable adaptivity

Walking before running
- Make progress at small scale while working towards large scale