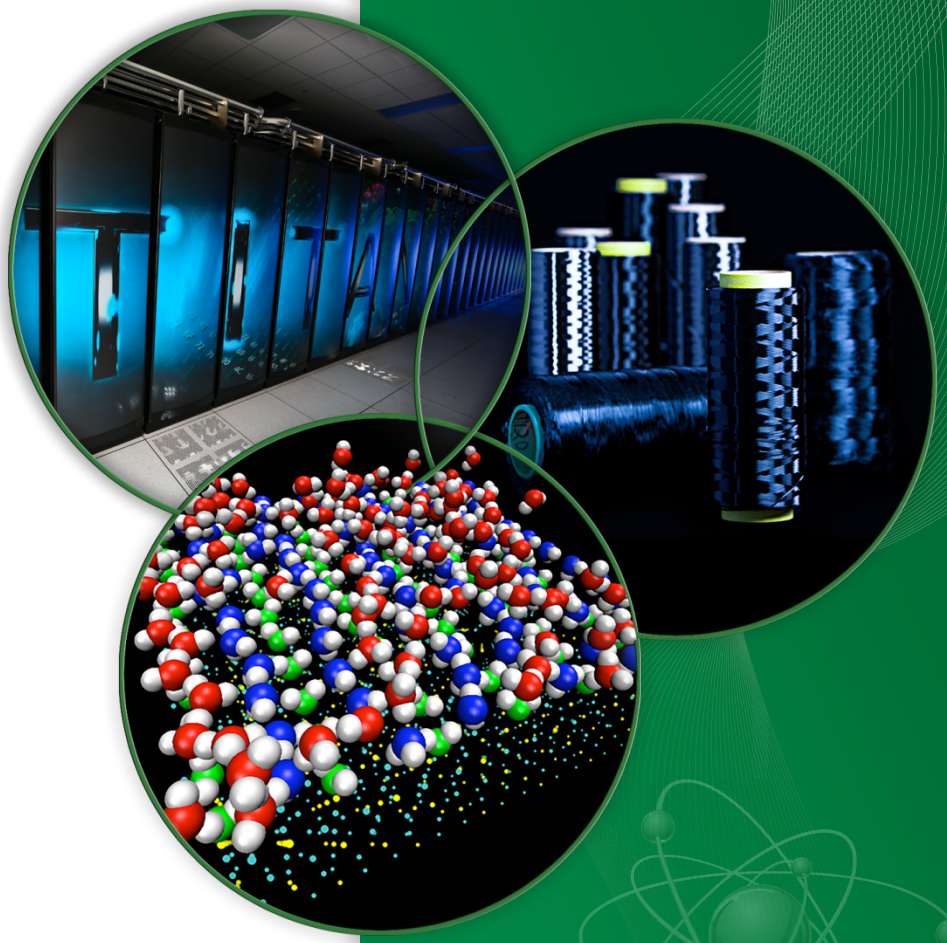
Interconnect Related Research at Oak Ridge National Laboratory



Barney Maccabe

Director, Computer Science and Mathematics Division

July 16, 2015
Frankfurt, Germany

ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE National Laboratory

The story of this talk

Underlying thesis

- RAM is a bad abstraction for memory systems
  - It's going to get much worse
- Latency versus bandwidth
  - Well understood for IPC and I/O
  - Exploit bandwidth capabilities when possible
- We need to start supporting other abstractions
  - Ifetch loop requires RAM so we can't get rid of it entirely

Outline of content

- Interconnect performance is not tracking for Exascale
- Relevant work at ORNL
  - Understanding applications
    - Tools for measurement and prediction
  - Programming interfaces
    - Application and library APIs
  - Exploiting in-transit processing
  - Understanding node structure
- Back to the thesis!

The urgency for Interconnect R&D

Contributions from:

Al Geist

Short story:

We're on track for almost everything needed for an Exascale platform.

We're off the tracks in Interconnects, inter-node is really bad and I/O is bad. Worse than that, we didn't really consider on-node interconnects.

OAK RIDGE National Laboratory

2017 OLCF Leadership System

Hybrid CPU/GPU architecture (like Titan)

Vendor: IBM (Prime) / NVIDIA™ / Mellanox®

At least 5X Titan's Application Performance

Approximately 3,400 nodes, each with:

- Multiple IBM POWER9™ CPUs and multiple NVIDIA Volta® GPUs.
- CPUs and GPUs completely connected with high speed **NVLink**
- Large coherent memory: over 512 GB (**HBM** + DDR4)
  - all directly addressable from the CPUs and GPUs
- An additional 800 GB of NVRAM, which can be configured as either a **burst buffer** or as extended memory or both
- over 40 TF peak performance

Dual-rail Mellanox® EDR-IB full, non-blocking fat-tree interconnect

IBM Elastic Storage (GPFS™) - 1TB/s I/O and 120 PB disk capacity.

OLCF-5 What's exascale look like?

Date	2009	2012	2017	OLCF-5 2022
System	**Jaguar**	**Titan**	**Summit**	**Exascale**
System peak	2.3 Peta	27 Peta	150+ Peta	**1-2 Exa**
System memory	0.3 PB	0.7 PB	2-5 PB	**10-20 PB**
NVM per node	none	none	800 GB	**~2 TB**
Storage	15 PB	32 PB	120 PB	**~300 PB**
MTTI	days	days	days	**O(1 day)**
Power	7 MW	9 MW	10 MW	**~20 MW**
Node architecture	CPU 12 core	CPU + GPU	X CPU + Y GPU	**X loc + Y toc**
System size (nodes)	18,700	18,700	3,400	**How fat?**
Node performance	125 GF	1.5 TF	40 TF	**depends (X,Y)**
Node memory BW	25 GB/s	25 - 200 GB/s	100 – 1000 GB/s	**10x fast vs slow**
Interconnect BW	1.5 GB/s	6.4 GB/s	25 GB/s	**4x each gen**
IO Bandwidth	0.2 TB/s	1 TB/s	1 TB/s	**flat**

Exascale architecture targets circa 2009
2009 Exascale Challenges Workshop in San Diego

Attendees envisioned two possible architectural swim lanes:

1. Homogeneous many-core thin-node system
2. Heterogeneous (accelerator + CPU) fat-node system

System attributes	2009	"Pre-Exascale"		"Exascale"	
System peak	2 PF	100-200 PF/s		1 Exaflop/s	
Power	6 MW	15 MW		20 MW	
System memory	0.3 PB	5 PB		32–64 PB	
Storage	15 PB	150 PB		500 PB	
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW	25 GB/s	0.1 TB/s	1 TB/s	0.4 TB/s	4 TB/s
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	18,700	500,000	50,000	1,000,000	100,000
Node interconnect BW	1.5 GB/s	150 GB/s	1 TB/s	250 GB/s	2 TB/s
IO Bandwidth	0.2 TB/s	10 TB/s		30-60 TB/s	
MTTI	day	O(1 day)		O(0.1 day)	

Exascale architecture targets
defined at 2009 Exascale Challenges Workshop in San Diego

Where we are going "off the tracks" is
data movement between nodes and from node to storage
Summit: Interconnect BW= 25 GB/s, I/O BW= 1 TB/s

System attributes	2009	"Pre-Exascale"		"Exascale"	
System peak	2 PF	100-200 PF/s		1 Exaflop/s	
Power	6 MW	15 MW		20 MW	
System memory	0.3 PB	5 PB		32–64 PB	
Storage	15 PB	150 PB		500 PB	
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW	25 GB/s	0.1 TB/s	1 TB/s	0.4 TB/s	4 TB/s
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	18,700	500,000	50,000	1,000,000	100,000
Node interconnect BW	1.5 GB/s	**150 GB/s**	**1 TB/s**	250 GB/s	2 TB/s
IO Bandwidth	0.2 TB/s	**10 TB/s**		30-60 TB/s	
MTTI	day	O(1 day)		O(0.1 day)	

Understanding Applications

Contributions from:

Jeffery Vetter,
Phil Roth,
Jeremy Meredith,
Seyong Lee, and
Christian Engelmann

Short story:

We are developing tools to understand applications. Including both measurement and prediction.

While not specifically developed for interconnect research, these tools can be used to study issues related to interconnects

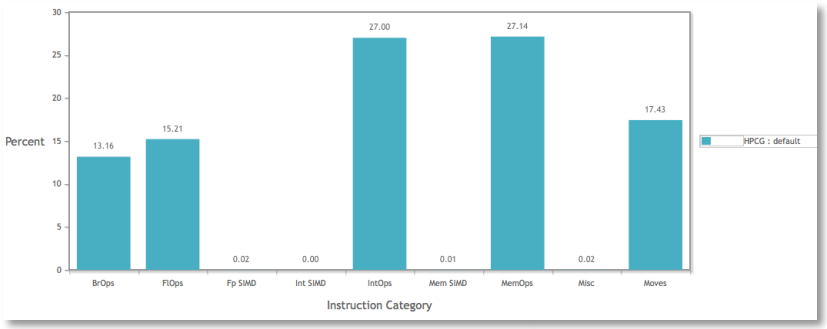
- Oxbow
- Aspen/Compass
- xSim

Oxbow: Understanding application needs

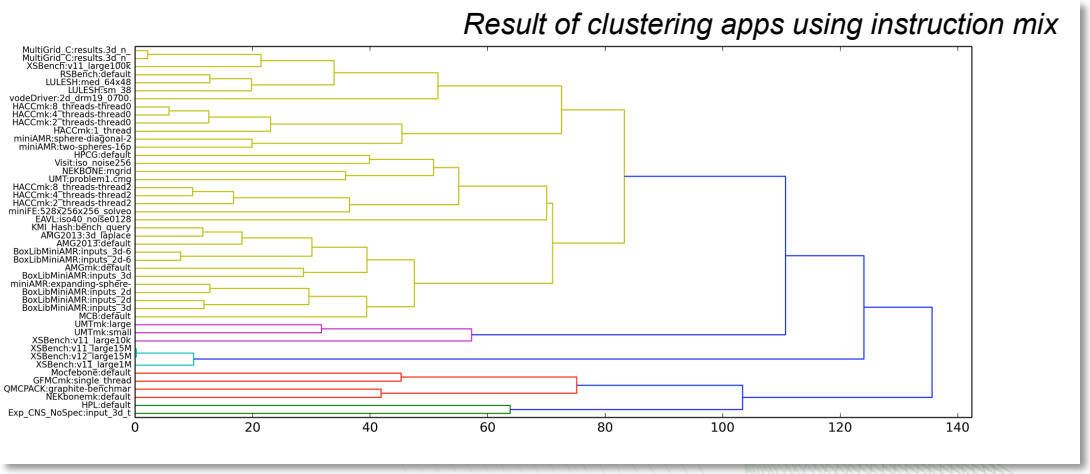
http://oxbow.ornl.gov

- Characterization on several axes:
  - Communication (topology, volume)
  - Computation (instruction mix)
  - Memory access (reuse distance)
- Impact
  - Representativeness of proxy apps
  - System design
- Online database for results with web portal including analytics support

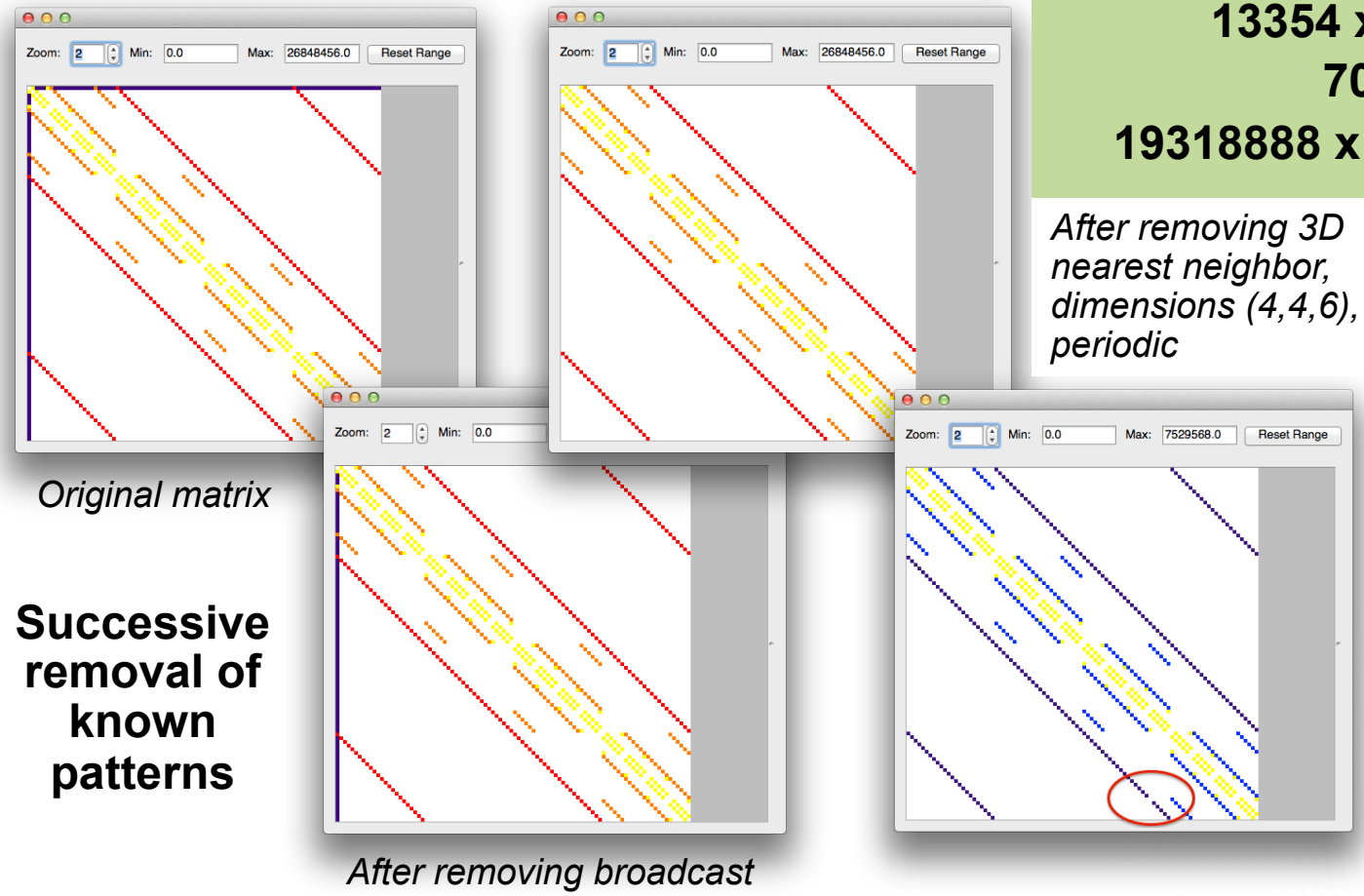
Instruction Mix, HPCG, 64 processes



Result of clustering apps using instruction mix



Example: Understanding communication patterns



Original matrix

After removing reduce

After removing broadcast

Successive removal of known patterns

13354 x Broadcast(root:0) +
700 x Reduce(root:0) +
19318888 x 3DNearestNeighbor(dims:(4,4,6), periodic:True)

After removing 3D nearest neighbor, dimensions (4,4,6), periodic

LAMMPS

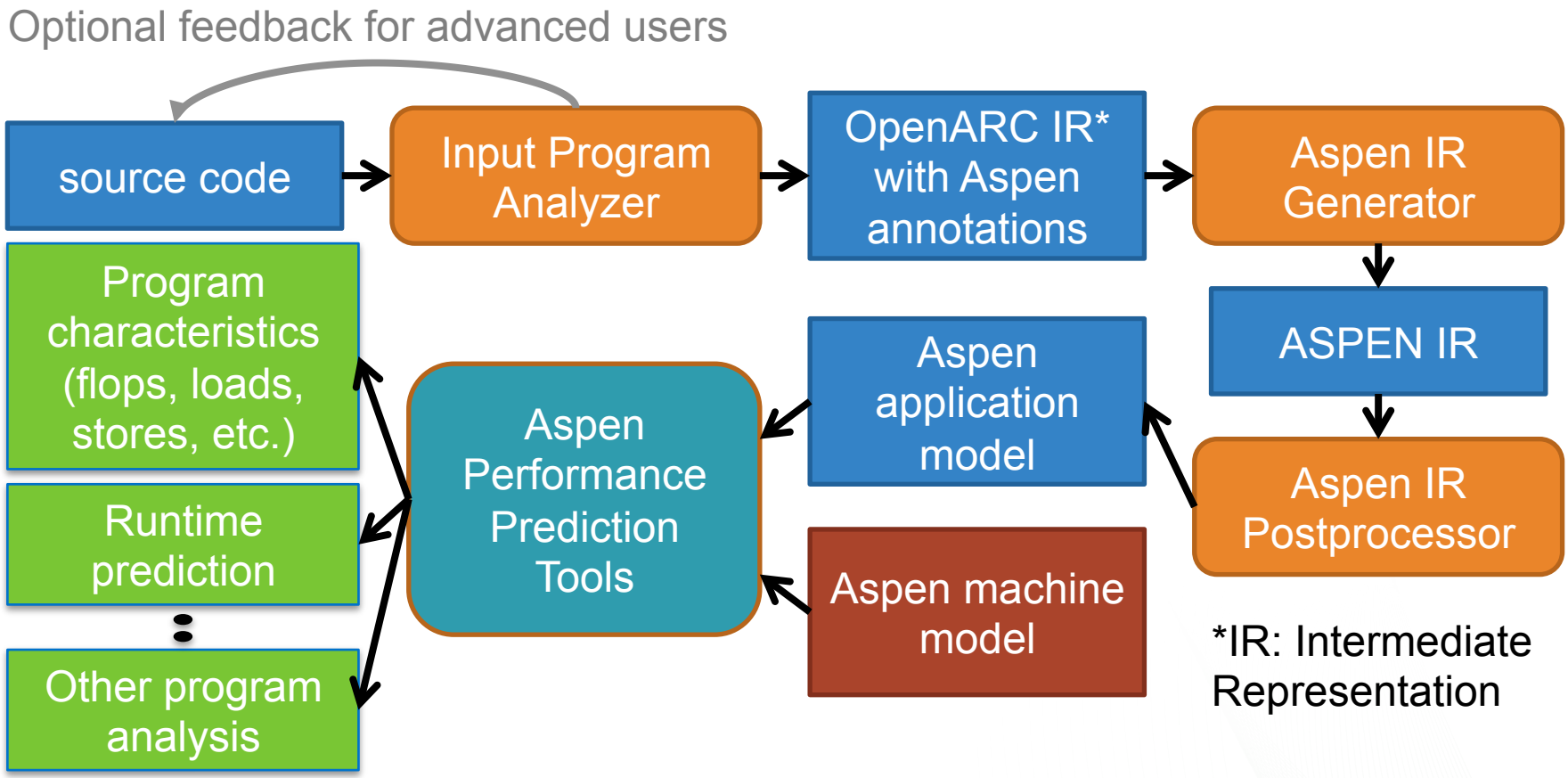
"Automated Characterization of Parallel Application Communication Patterns," Roth, Meredith, and Vetter, **HPDC 2015**

- Communication matrix collected using mpiP from 96 process run on Keeneland
- Imperfect 3D Nearest Neighbor pattern (note red circle in last figure)

SUPER INSTITUTE FOR SUSTAINED PERFORMANCE, ENERGY, AND RESILIENCE

OAK RIDGE National Laboratory

COMPASS System Overview



COMPASS is:

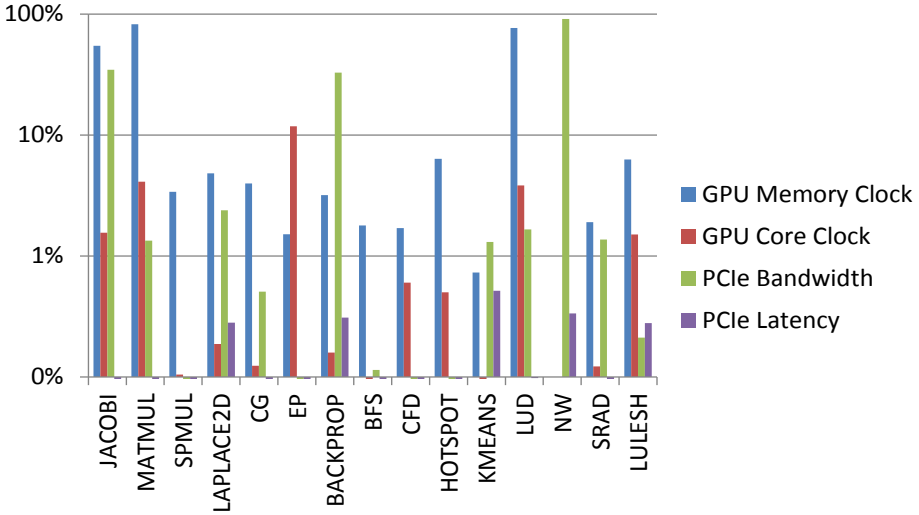
- Implemented using OpenARC (Open Accelerator Research Compiler) http://ft.ornl.gov/research/openarc, and
- Uses Aspen for performance analysis http://ft.ornl.gov/research/performance

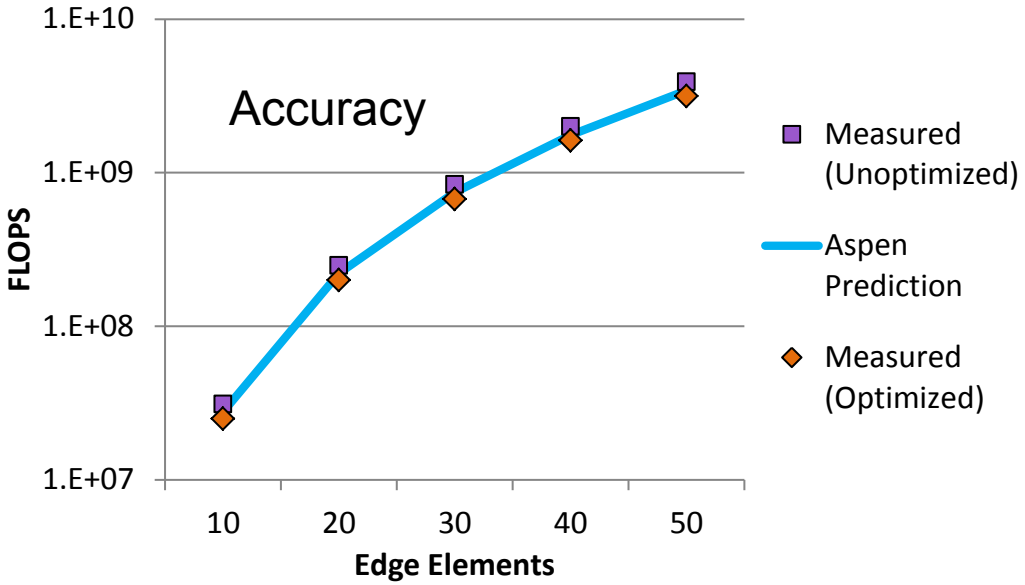
Example: LULESH

Automatic generation of an Aspen application model enables highly complex models that exhibit a very high degree of accuracy

- Input LULESH program:
  3700 lines of C codes
- Output Aspen model:
  2300 lines of Aspen specifications

Sensitivity to variation in hardware parameters



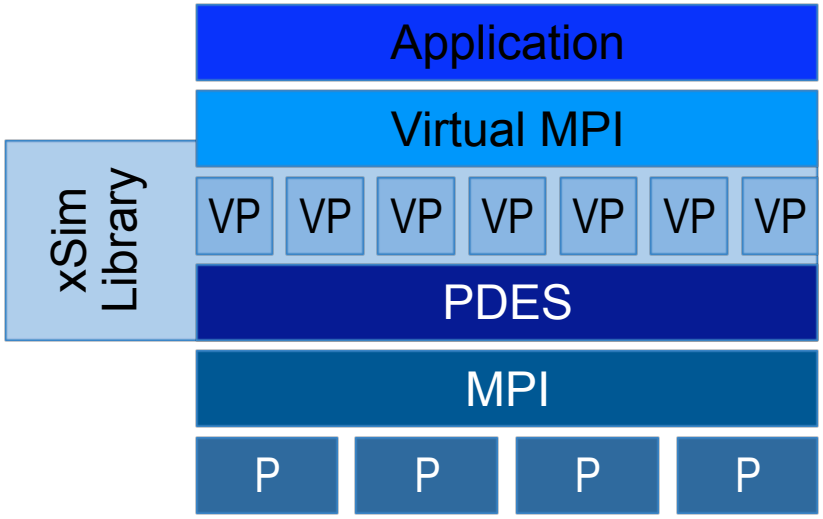


$$\text{Sensitivity} = \frac{\text{Application improvement}}{\text{Hardware improvement}}$$

S. Lee, J.S. Meredith, and J.S. Vetter, "COMPASS: A Framework for Automated Performance Modeling and Prediction," in *Proceedings of the **29th ACM on International Conference on Supercomputing***. Newport Beach, California, USA: ACM, 2015.

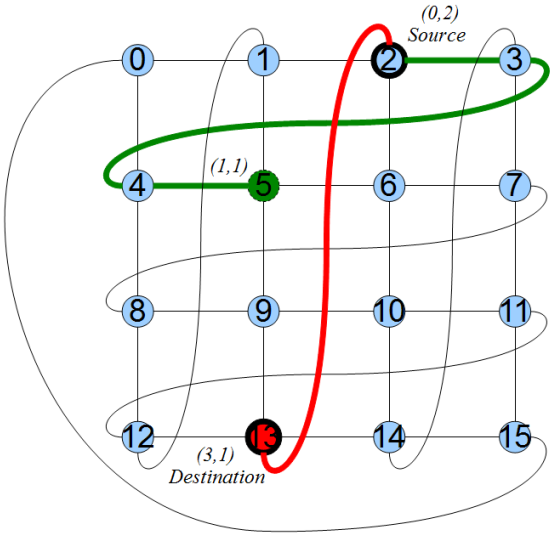
xSim: the Extreme Scale Simulator

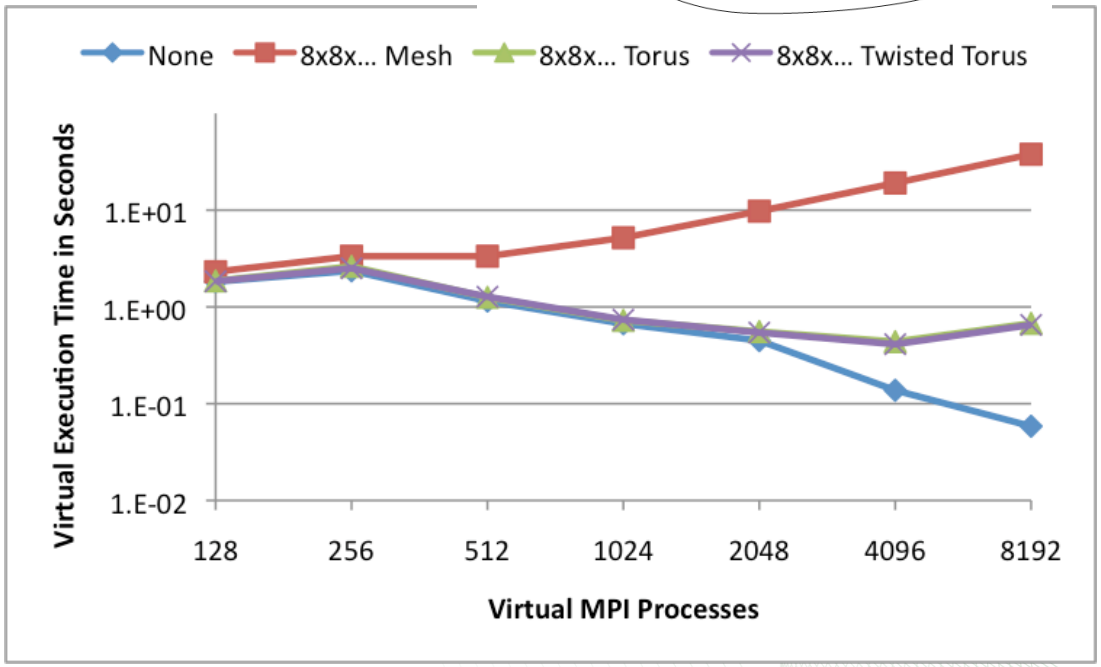
- Use a moderate scale system to simulate extreme scale systems
- Implementation
  - Applications run within the context of virtual processors
    - Applications are unmodified and only need to link to the xSim library
  - The virtual processors expose a virtual MPI for applications
    - Utilizes PMPI to intercept MPI calls and to hide the PDES
  - Parallel Discrete Event Simulation (PDES)
    - Uses the native MPI to simulate virtual processors
    - Implements a time-accurate network with tunable performance (e.g., bandwidth, latency, topology)



Example: Network Topology

- Configurable network model
  - Star, ring, mesh, torus, twisted torus, and tree
  - Link latency & bandwidth
  - Contention and routing
  - Hierarchical combinations, e.g.,on-chip, on-node, & off-node
  - Simulated rendezvous protocol
- Example: NAS MG in a dual-core 3D mesh or twisted torus





C. Engelmann. xSim: The Extreme-scale Simulator. Munich, 2015.

Programming Interfaces

Contributions from:

Oscar Hernandez,
Pavel Shamis, and
Manjunath Gorentla Venkata

Short story:

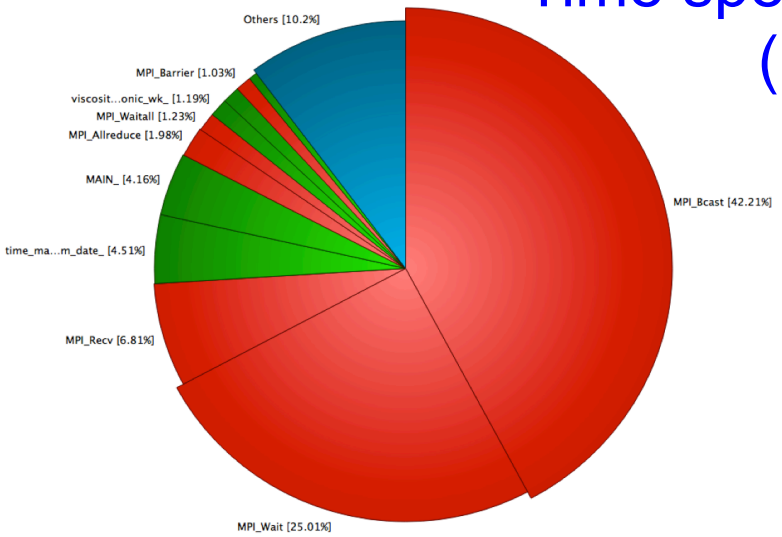
We are also engaged in the development of APIs related to interconnects.

- Open MPI,
- OpenACC
- OpenSHMEM
- UCX

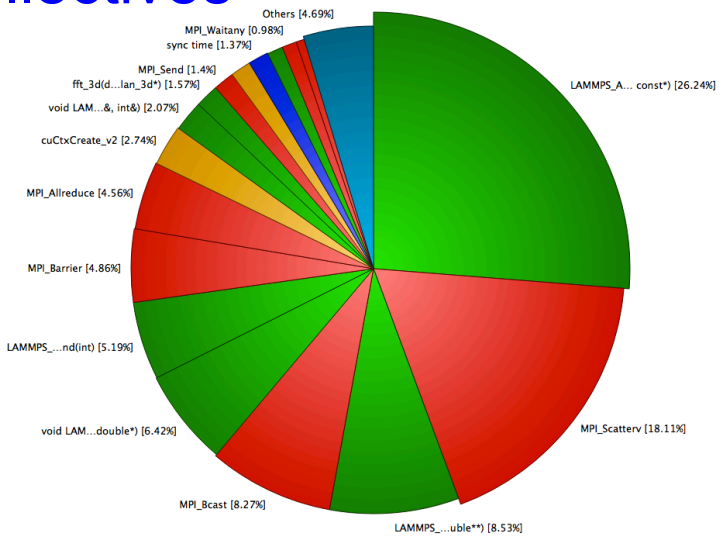
Improving the Performance of Collective Operations in Open MPI

- Collective operations are global communication and synchronization operations in a parallel job
- Important component of a parallel system software stack
  - Simulations are sensitive to collectives performance characteristics
  - Simulations spend significant amount of time in collectives

Time spent in collectives (in Red)



CESM > 50%



LAMMPS > 35%

Cheetah: A Framework for High-performing Collectives

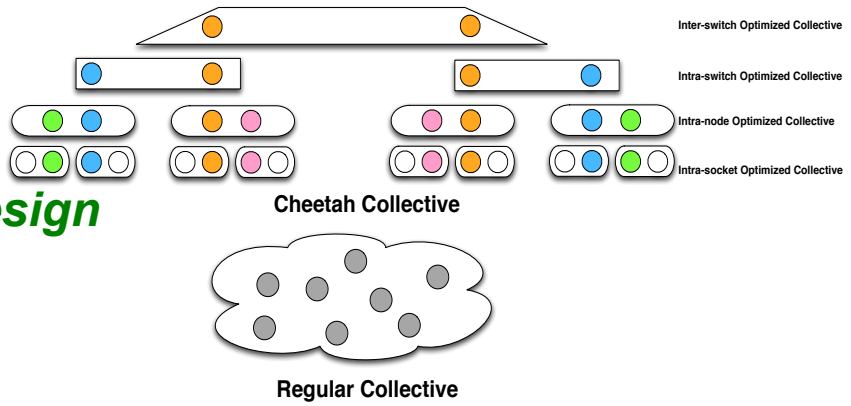
Objectives

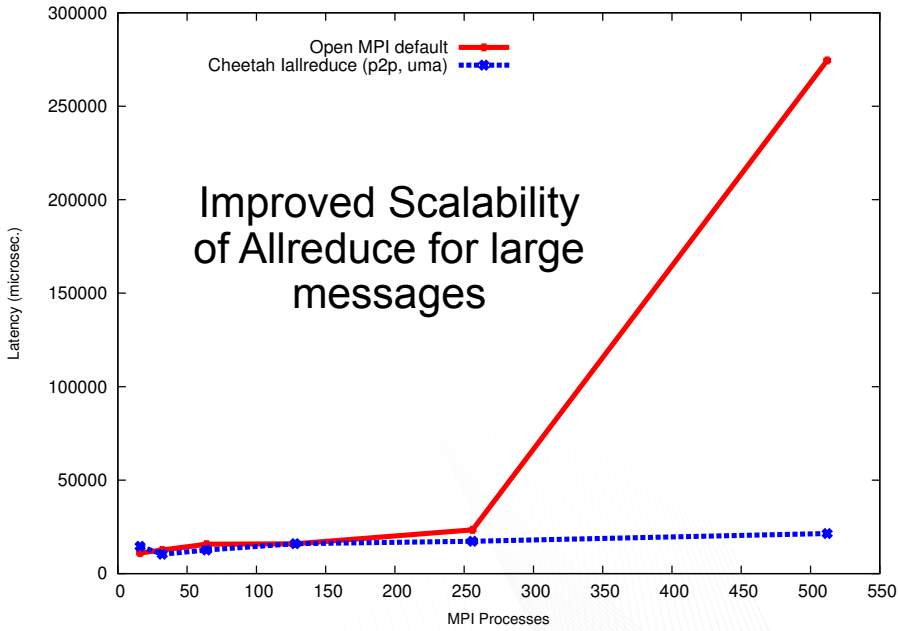
- Develop a high-performing and highly scalable collective operations library
- Develop collective offload mechanism for InfiniBand HCA
- Designed to support blocking and nonblocking semantics, and achieve high scalability and performance on modern multicore systems

Approach

- Hierarchical design driven implementation to achieve performance and scalability
- Offload collective processing to the network hardware, reducing OS noise effects and enhancing CPU availability to the computations

Design





Outcomes

- Influenced capabilities and functionality of Mellanox's CORE-Direct technologyAvailable in Open MPI 1.8

http://www.csm.ornl.gov/cheetah

Support

- FAST-OS (PI: Richard Graham) and
- Oak Ridge Leadership Facility (OLCF)

OAK RIDGE National Laboratory

Reminder: OpenACC in a Nutshell

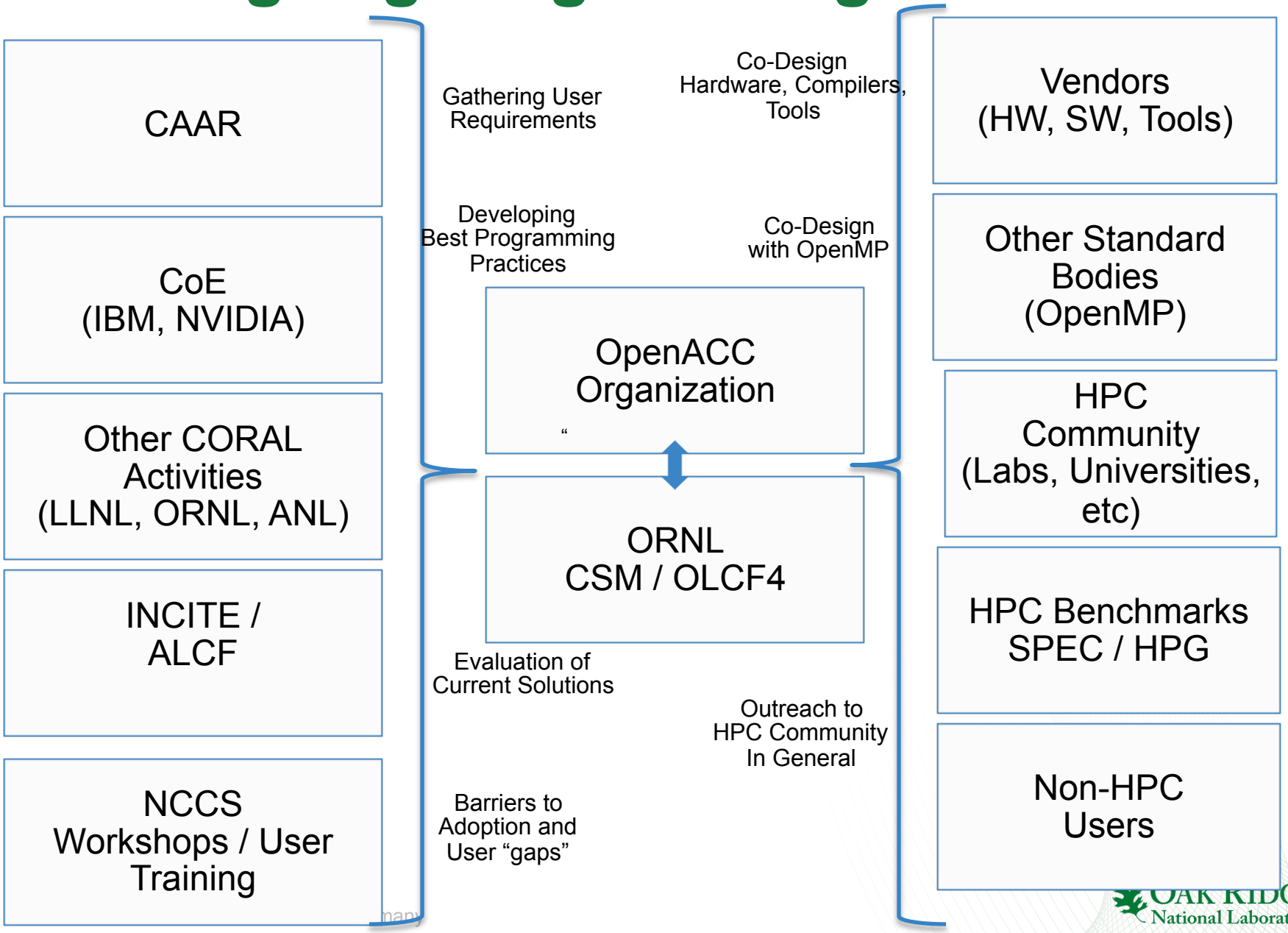
- Directive-based API
- Bindings for C, C++, Fortran
- **Single code base** for both accelerator and non-accelerator implementations
- **Strategically important to ORNL/CORAL** as a standard, portable solution to programming accelerators
  - Implementations available for NVIDIA GPUs, AMD GPUs and APUs, Intel Xeon Phi
- **OpenACC is a "research vehicle" for OpenMP accelerator support**
  - Use OpenACC to figure it out (rapid release cycle)
  - Put "the right answer" into OpenMP (slow release cycle)

OpenACC
DIRECTIVES FOR ACCELERATORS

```
#pragma acc kernels loop
for( i = 0; i < nrows; ++I ){
   double val = 0.0;
   int nstart = rowindex[i];
   int nend = rowindex[i+1];
   #pragma acc loop vector
      reduction(+:val)
   for( n = nstart; n < nend; ++n )
      val += m[n] * v[colndx[n]];
   r[i] = val;
}
```

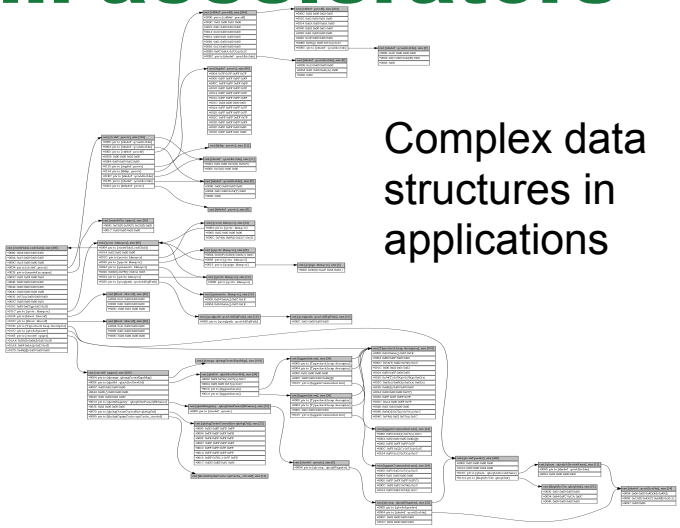
Sparse (CSR) matrix-vector multiplication in C + OpenACC

Co-Designing Programming Environment



Current Challenges
Moving complex data to/from accelerators

- Hardware solutions: NVLINK, etc
  - First touch policy, OS support
- Programming model solutions
  - "Deep copy" is a blocking issue for OpenACC adoption
  - Deep copy solution also addresses key C++ issues: STLs, etc



Complex data structures in applications

OpenACC "deep copy" directives

```
template <class T>
class Matrix {
    size_type nRow, nCol, lDim, physicalSize;
    bool owner;
    T* data;
    #pragma acc policy(managed) inherit(data[0:lDim*nCol])
    #pragma acc policy(aliased) present(data[@])
…}
…
Matrix<double> a(M*N,M*N);
Matrix<double> b, c;
b.owner = false;
c.owner = false;
#pragma acc data enter copyin(managed:a, aliased:b)
{
   #pragma acc parallel for private( c )
      for( i=0; i<M*N; ++i )
…
```

OpenACC Complex Data Management Tech Report
http://www.openacc.org/sites/default/files/TR-14-1.pdf

OpenSHMEM Activities at ORNL

OpenSHMEM Reference Implementation

- Open source reference implementation
  - Supports InfiniBand and Gemini network interfaces
  - Leverages UCCS (UCX) and GASNet for network functionality
  - Available on GitHub: https://github.com/openshmem-org/
  - Partners : UH
- UCX: High-performing network layer for OpenSHMEM libraries

OpenSHMEM Research

- OpenSHMEM with GPU-Initiated communication
  - Co-design with NVIDIA
- OpenSHMEM Analyzer
  - Analyzes semantic compliancy of OpenSHMEM programs
  - Partners: UH
- Fault-tolerant OpenSHMEM
  - Partners: University of Tennessee, Knoxville

Partners

NVIDIA

Mellanox TECHNOLOGIES

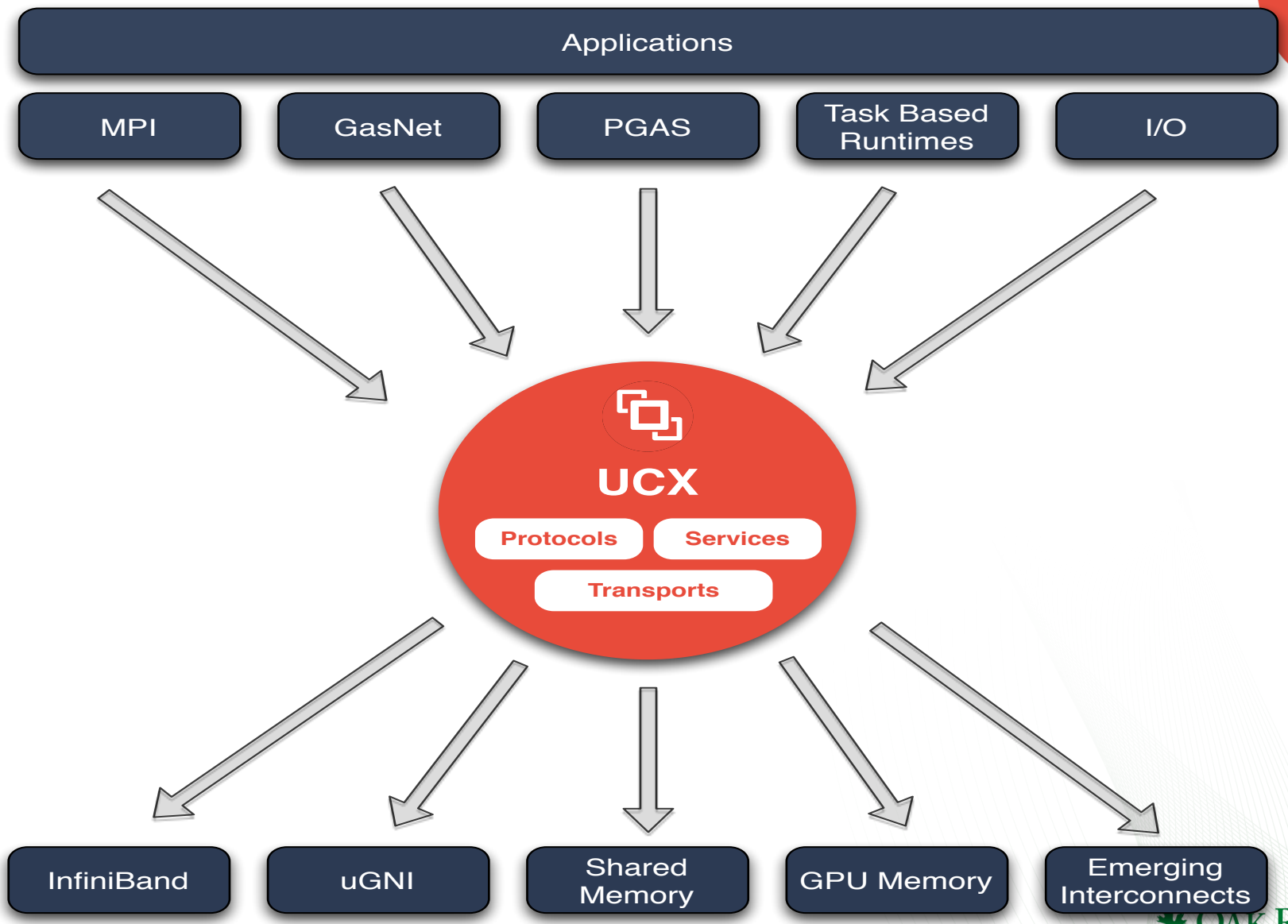
ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

Open SHMEM

http://openshmem.org

OAK RIDGE National Laboratory

The Vision



Collaboration

- Mellanox co-designs network interface and contributes MXM technology
  - Infrastructure, UD, RC, DCT, shared memory, protocols, integration with OpenMPI/SHMEM, MPICH
- ORNL co-designs network interface and contributes UCCS project
  - IB optimizations, support Crays devices, shared memory
- NVIDIA co-designs high-quality support for GPU devices
  - GPU-Direct, GDR copy, etc.
- IBM co-designs network interface and contributes ideas and concepts from PAMI
- UH/UTK focus on integration with their research platforms

UCX: Background

MXM

- Developed by Mellanox Technologies
- HPC communication library for InfiniBand devices and shared memory
- Primary focus: MPI, PGAS

UCCS

- Developed by ORNL, UH, UTK
- Originally based on Open MPI BTL and OPAL layers
- HPC communication library for InfiniBand, Cray Gemini/Aries, and shared memory
- Primary focus: OpenSHMEM, PGAS
- Also supports: MPI

PAMI

- Developed by IBM on BG/Q, PERCS, IB VERBS
- Network devices and shared memory
- MPI, OpenSHMEM, PGAS, CHARM ++, X10
- C++ components
- Aggressive multi-threading with contexts
- Active Messages
- Non-blocking collectives with hw accleration support

Portals?

- Sandia, UNM, ORNL, etc
- Network Interface Architecture
- Define essential capabilities

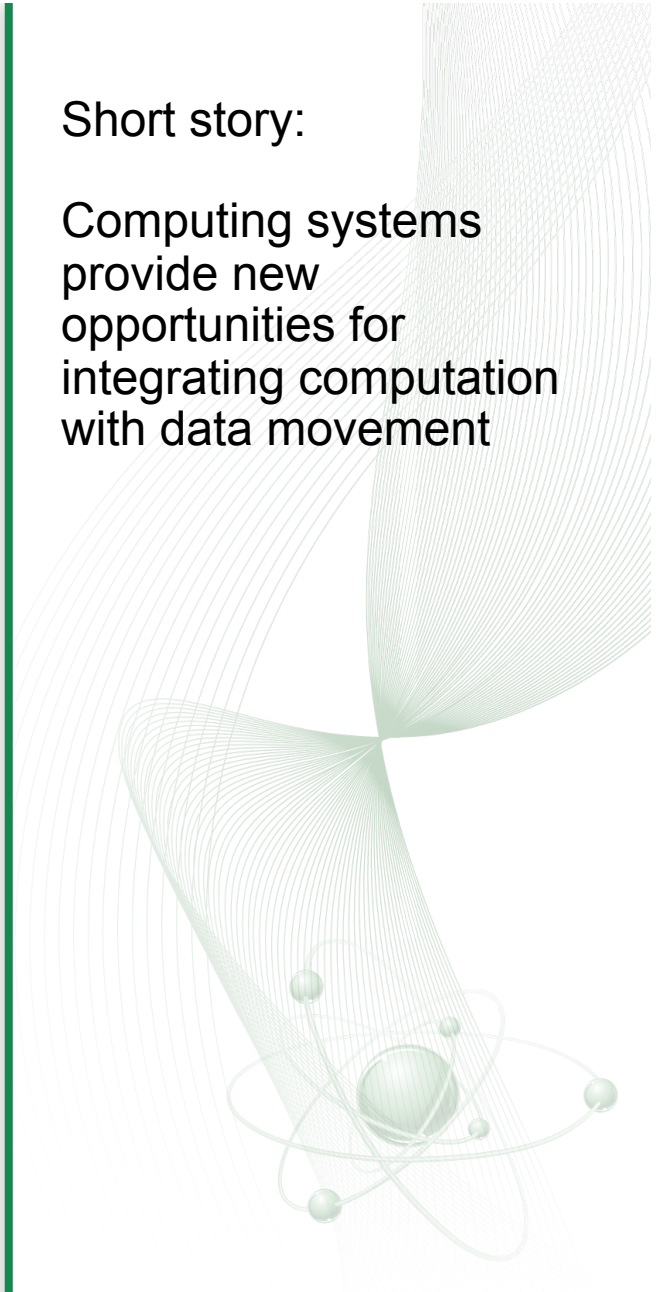
Interactions Between Data Processing and Data Movement

Contributions from:

Scott Klasky

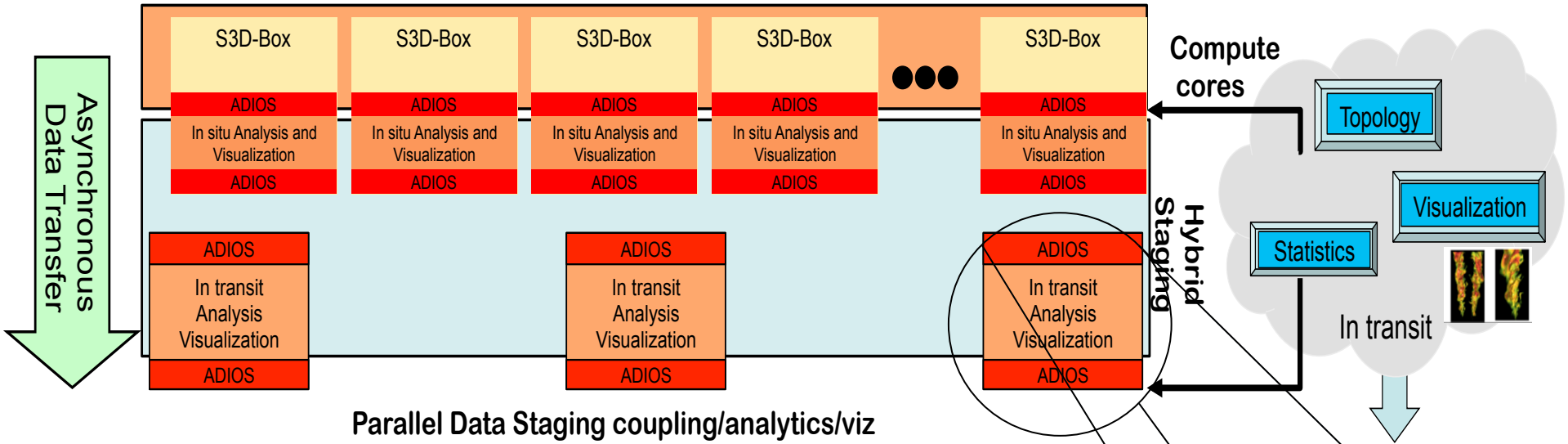
Short story:

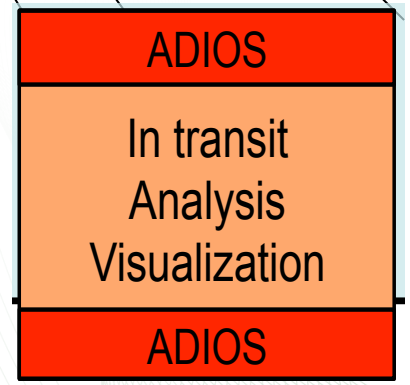
Computing systems provide new opportunities for integrating computation with data movement



OAK RIDGE National Laboratory

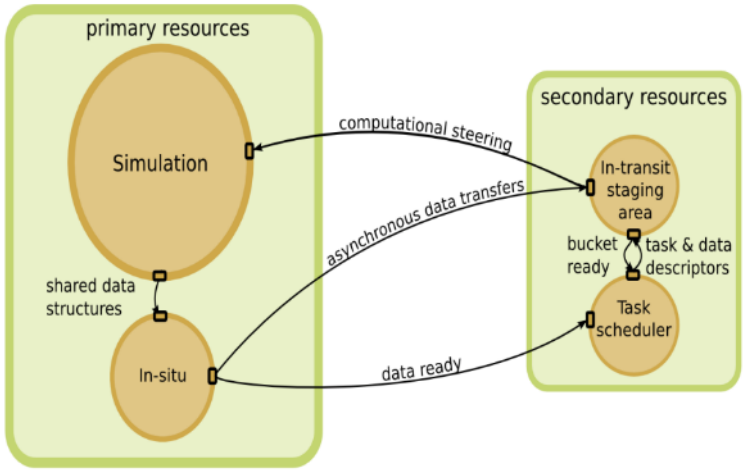
Hybrid Staging

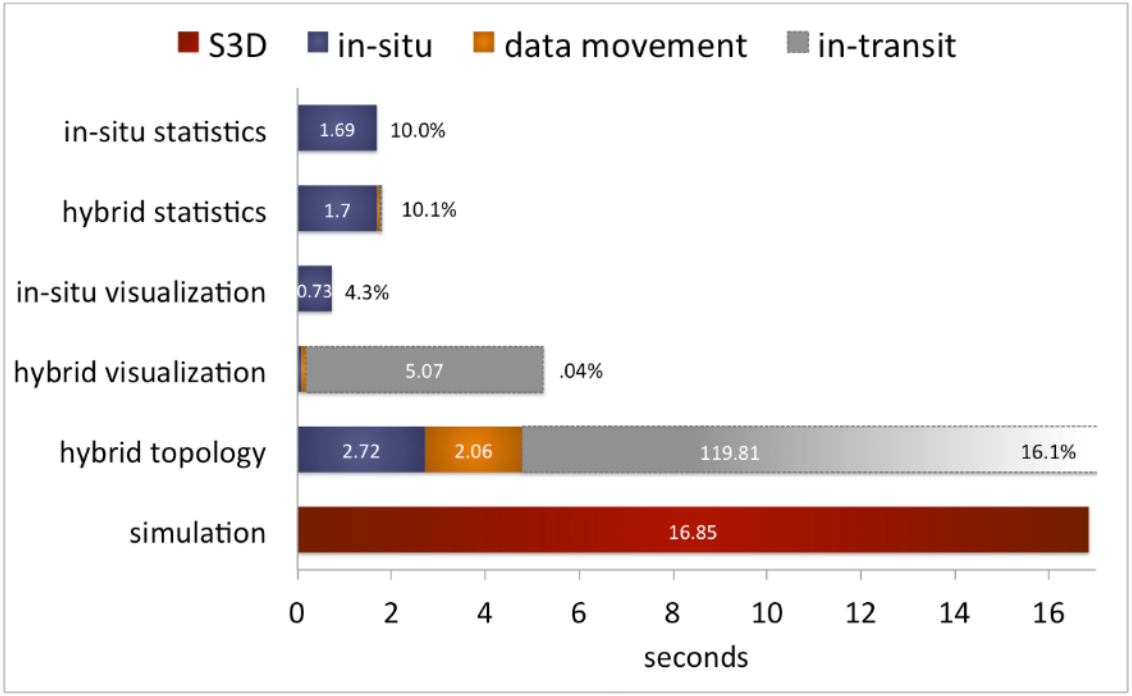




- Use compute and deep-memory hierarchies to optimize overall workflow for power vs. performance tradeoffs
- Abstract complex/deep memory hierarchy access
- Placement of analysis and visualization tasks in a complex system
- Impact of network data movement compared to memory movement

In-situ and In-transit Partitioning of Workflows





- Primary resources execute the main simulation and in situ computations
- Secondary resources provide a staging area whose cores act as buckets for in transit computations

- 4896 cores total (4480 simulation/in situ; 256 in transit; 160 task scheduling/data movement)
- Simulation size: 1600x1372x430
- All measurements are per simulation time step

Understanding the on-node components

Contributions from:

Jeffery Vetter

Short story:

On-node interconnects are becoming more and more complicated.

We need to understand the endpoints: memories and processors.

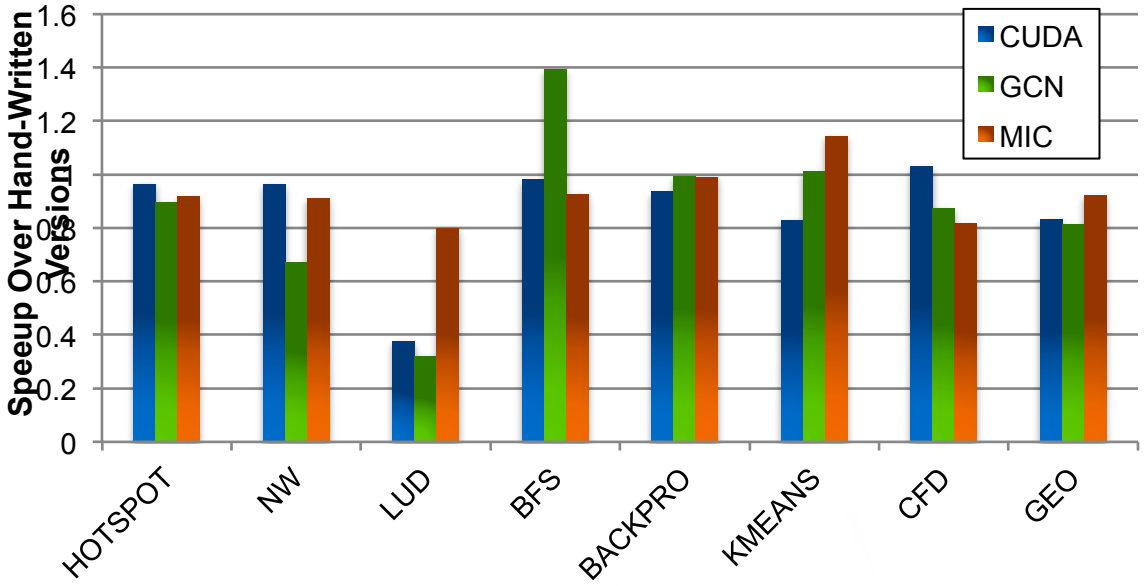
OAK RIDGE National Laboratory

Contemporary Heterogeneous Architectures

Property	CUDA	GCN	MIC
Programming models	CUDA, OpenCL	OpenCL, C++ AMP	OpenCL, Cilk, TBB, LEO, OpenMP
Thread Scheduling	Hardware	Hardware	Software
Programmer Managed Cache	Yes	Yes	No
Global Synchronization	No	No	Yes
L2 Cache Type	Shared	Private per core	Private per core
L2 Total Size	Up to 1.5MB	Up to 0.5 MB	25MB
L2 Line-size	128	64	64
L1 Data Cache	Read-only + Read-write	Read-only	Read-write
Native Mode	No	No	Yes

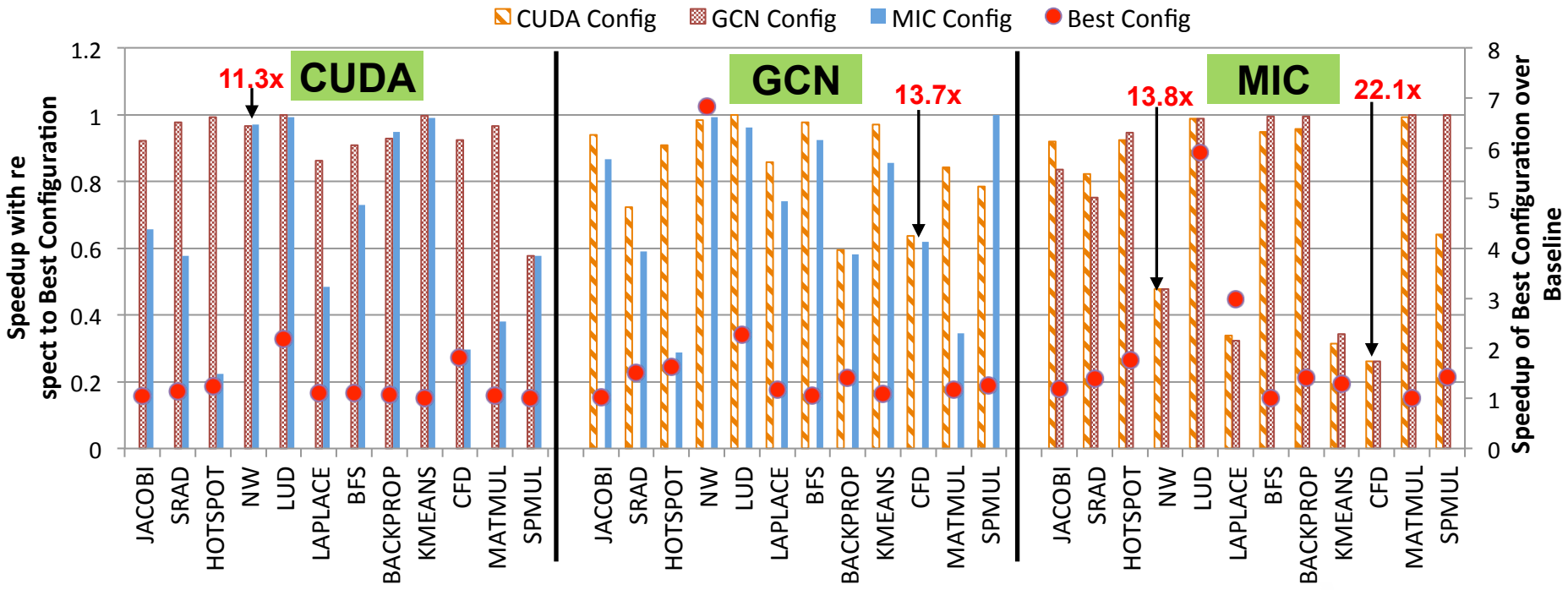
Comparison to Hand-written CUDA/OpenCL Programs

- Geo. mean indicates that 82%, 81%, 92% of the performance of hand written code can be achieved by tuned OpenACC programs on CUDA, GCN and MIC resp.



- LUD : CUDA, GCN are slower as they don't use on-chip shared-memory. No issues on MIC, as there is no shared-memory
- KMEANS : On MIC, OpenACC performs better → unrolling
- CFD : Performs better on CUDA → Texture memory
- BFS : Better perf. due to parallelism arrangement on GCN

Overall Performance Portability



- Better perf. portability among GPUs
- Lesser across GPUs and MIC
- Main reasons
  - Parallelism arrangement
  - Compiler optimizations : e.g. device-specific memories, unrolling etc.

Performance Portability Matrix

Best Program version of	Executed on: CUDA	Executed on: GCN	Executed on: MIC
CUDA	100	84	65
GCN	91	100	67
MIC	58	68	100

A. Sabne, P. Sakhnagool *et al., "Evaluating Performance Portability of OpenACC," in 27th International Workshop on Languages and Compiler for Parallel Computing (LCPC)* Portland, Oregon, 2014

Speculation (apologies to Dave Resnick)

- We need to define API(s) (abstract protocols) that focus on data movement
- Can we unify on-node and off-node APIs into a single framework?
  - What about I/O?
- Virtualization
  - configurable isolation
- Resilience (at rest, in-transit, or transformation)
- In-transit data manipulation (stream oriented computation)
- Abstract, but enable access to hardware capabilities?