SAND2012-3209P

Design Space Exploration and Co-design

Sudip Dosanjh

Extreme Scale Computing
Sandia National Laboratories

Exascale Research Conference
April 16, 2012
Portland, OR

Goal

Provide guidance/feedback to architects and algorithms/application researchers as we explore options for exascale and potential tradeoffs

Mantevo* Project

* Greek: augur, guess, predict, presage

- Multi-faceted application performance project.
- **Started 4 years ago.**
- Two types of packages:
  - **Miniapps**: Small, self-contained programs.
    - **MiniFE/HPCCG**: unstructured implicit FEM/FVM.
    - **phdMesh**: explicit FEM, contact detection.
    - **MiniMD**: MD Force computations.
    - **MiniXyce**: Circuit RC ladder.
    - **CTH-Comm: Data exchange pattern of CTH.**
  - **Minidrivers**: Wrappers around Trilinos packages.
    - **Beam**: Intrepid+FEI+Trilinos solvers.
    - **Epetra Benchmark Tests**: Core Epetra kernels.
    - **Dana Knoll working on new one.**
- Open Source (LGPL)
- Staffing: Application & Library developers.

Sandia National Laboratories

Can we trust mini-applications?

FY12 L2: Characterize the role of mini-applications in predicting key performance characteristics of real ASC applications.

Does information from miniapp exploration result in code changes?

Are they static?

For *diagnostics* $\{D\} = D_1, D_2, \ldots, D_n$,

baseline observations $\{B\} = B_1, B_2, \ldots, B_n$, and

miniapp measurements $\{A\} = A_1, A_2, \ldots, A_n$,

Then

$$X_i = || B_i - A_i ||_i \text{ , for all } i$$

$$V_i = \begin{cases} \textit{pass}, \text{ for } T^1_i < X_i < T^2_i \\ \textit{caution}, \text{ for } T^2_i < X_i < T^3_i \\ \textit{fail}, \text{ for } X_i > T^3_i \end{cases}$$

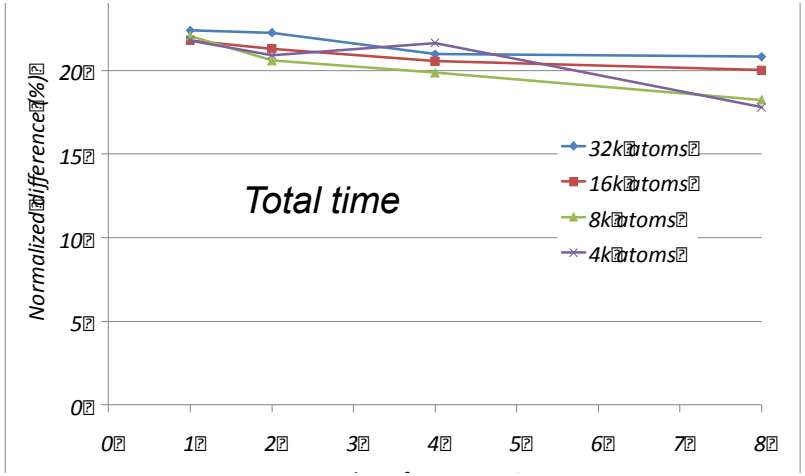
for thresholds T.

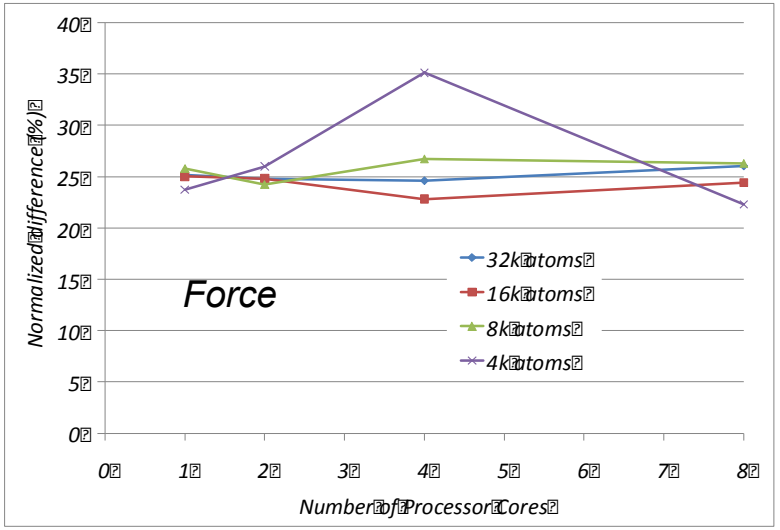
Potential O(1000) reduction in complexity

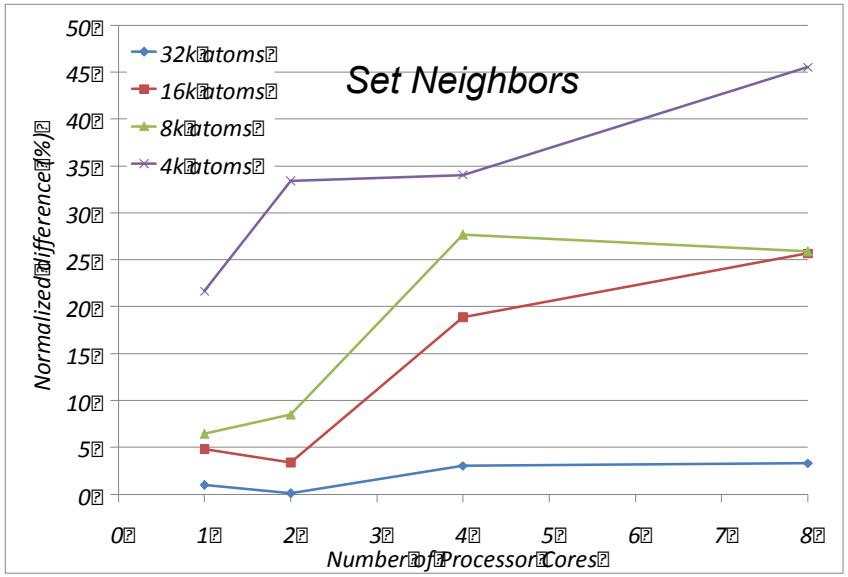
Sandia National Laboratories

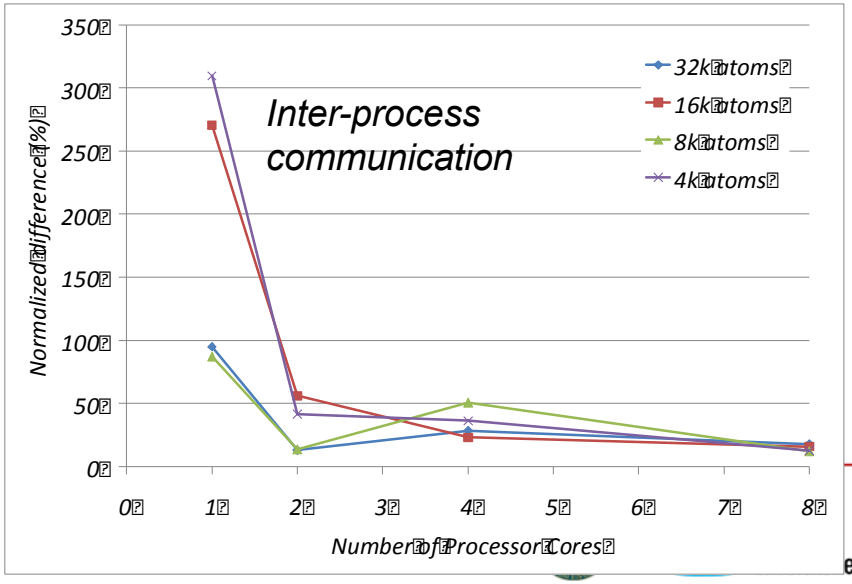
miniMD as predictor for LAMMPS?

time $X_i = || B_i - A_i ||_l / B_i$



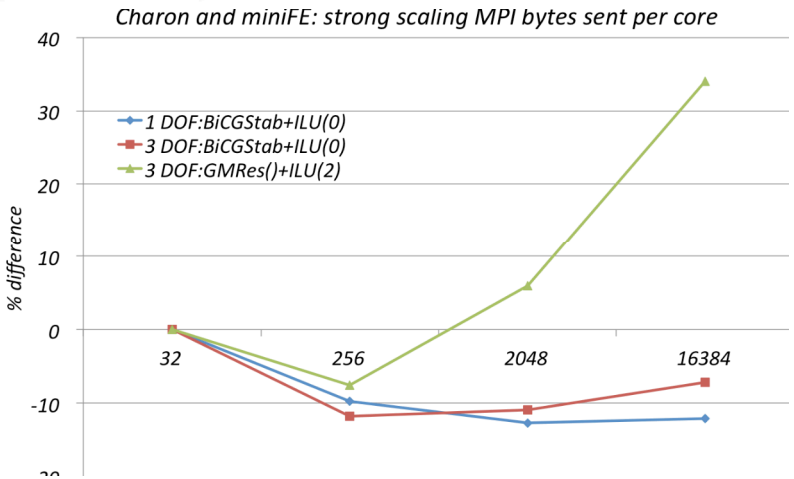




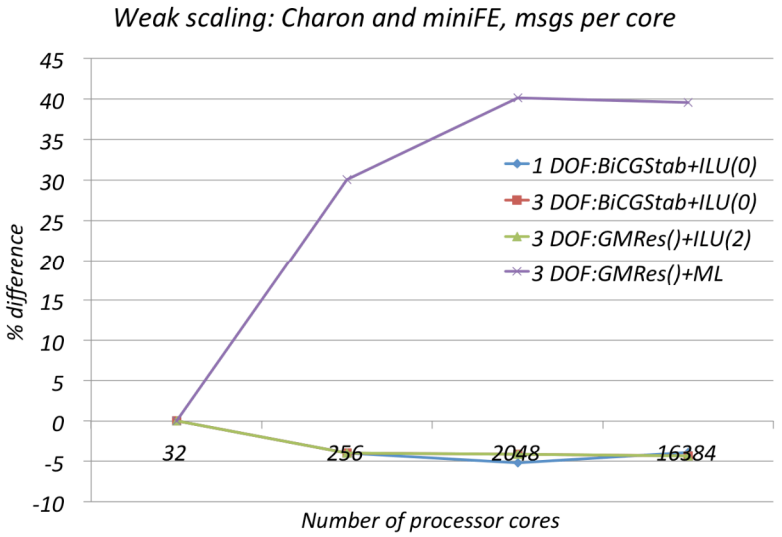


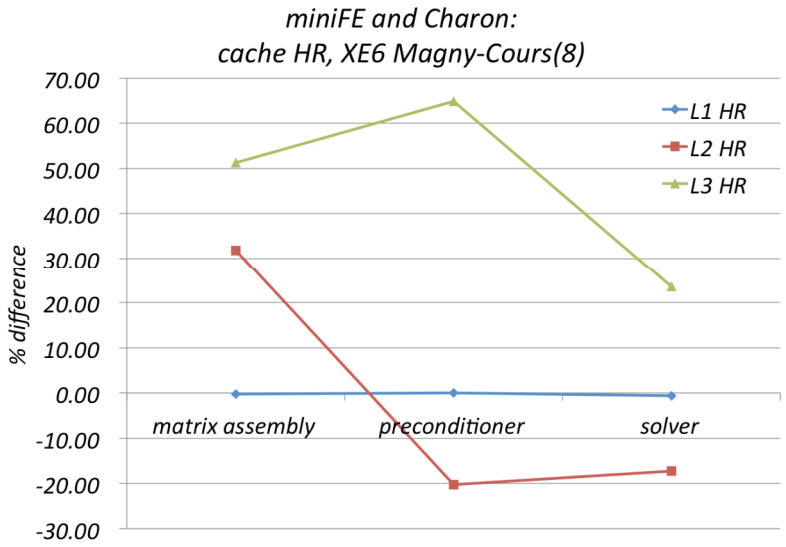
miniFE as predictor for Charon?

$$X_i = \| B_i - A_i \|_l / B_i$$



Execution on Cielo, Cray XE6





Sandia National Laboratories

Facets of Co-design

Measurement

- Miniapplications studies on testbeds/prototypes provides feedback to both architects and application developers

Experiments

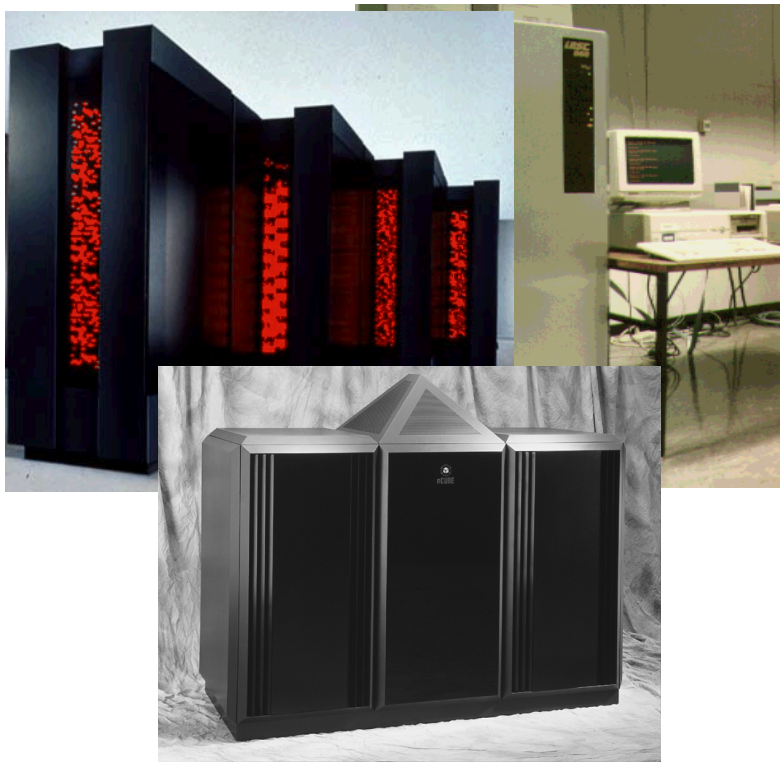
- Emulation
- Xstack (ParalleX, HPX, qthreads)
- Detuning studies

Prediction

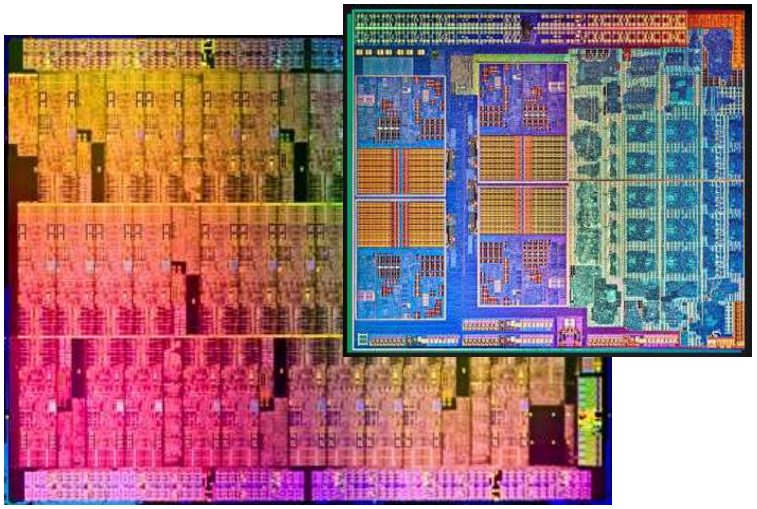
- Simulation/modeling, AMMs, miniapps

Measurement

1990s

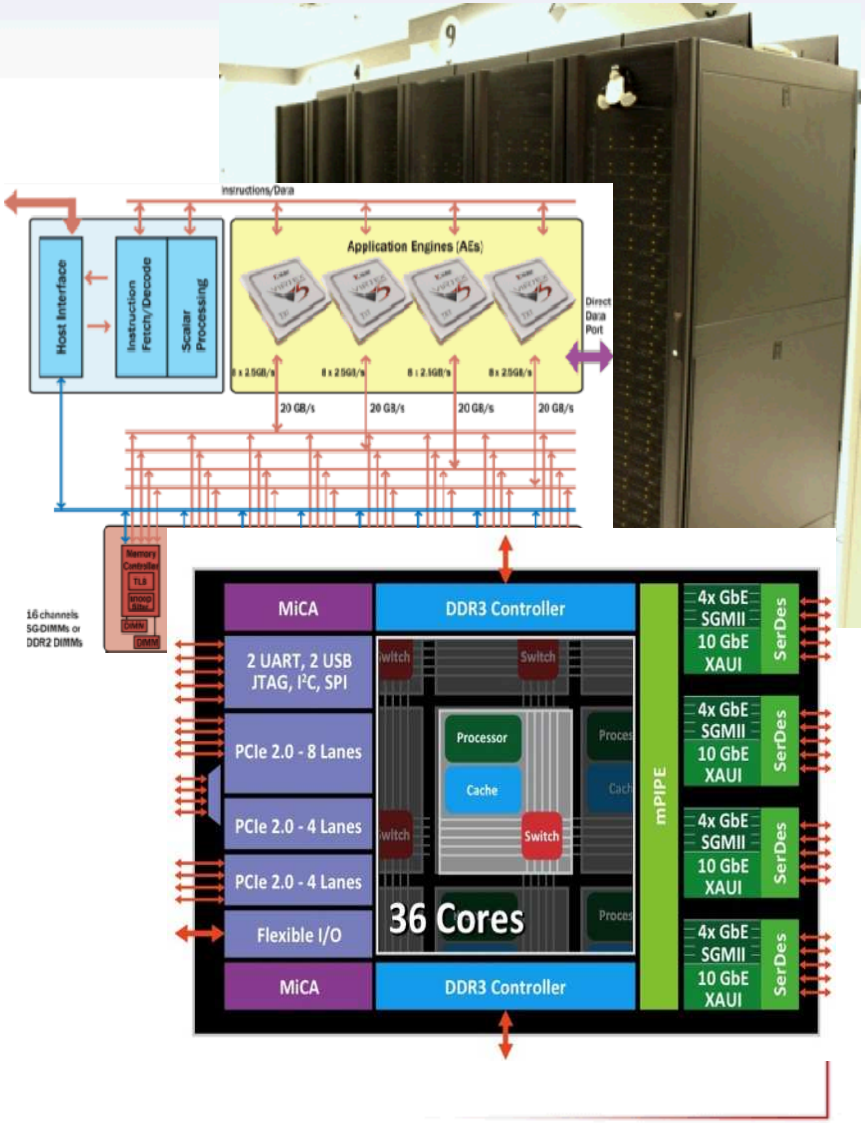


2010s



Recent Testbeds

- Intel Many Integrated Core (MIC) testbed with Knights Ferry co-processors
- Cray XK6 with Nvidia GPUs
- Convey HC-1ex
- AMD Fusion cluster: a heterogeneous CPU/GPU node with common memory address space
- Tilera TILE-Gx36 Processors



"miniFE" is a Finite-Element mini-application

Implements algorithms from an implicit finite-element application

1. **Assemble a sparse linear-system from the steady-state conduction equation on a domain of hexahedral elements**
2. **Solve the linear-system using the Conjugate Gradient algorithm**

   Per iteration:

   - **2 dot-products**
   - **3 axpys**
   - **1 sparse matrix-vector product**

$r_0 = b - Ax_0$

Loop {

If $k == 1$

$p_1 = r_0$

else

$\beta_k = r_{k-1}^T r_{k-1} / r_{k-2}^T r_{k-2}$

$p_k = r_{k-1} + \beta_k p_{k-1}$

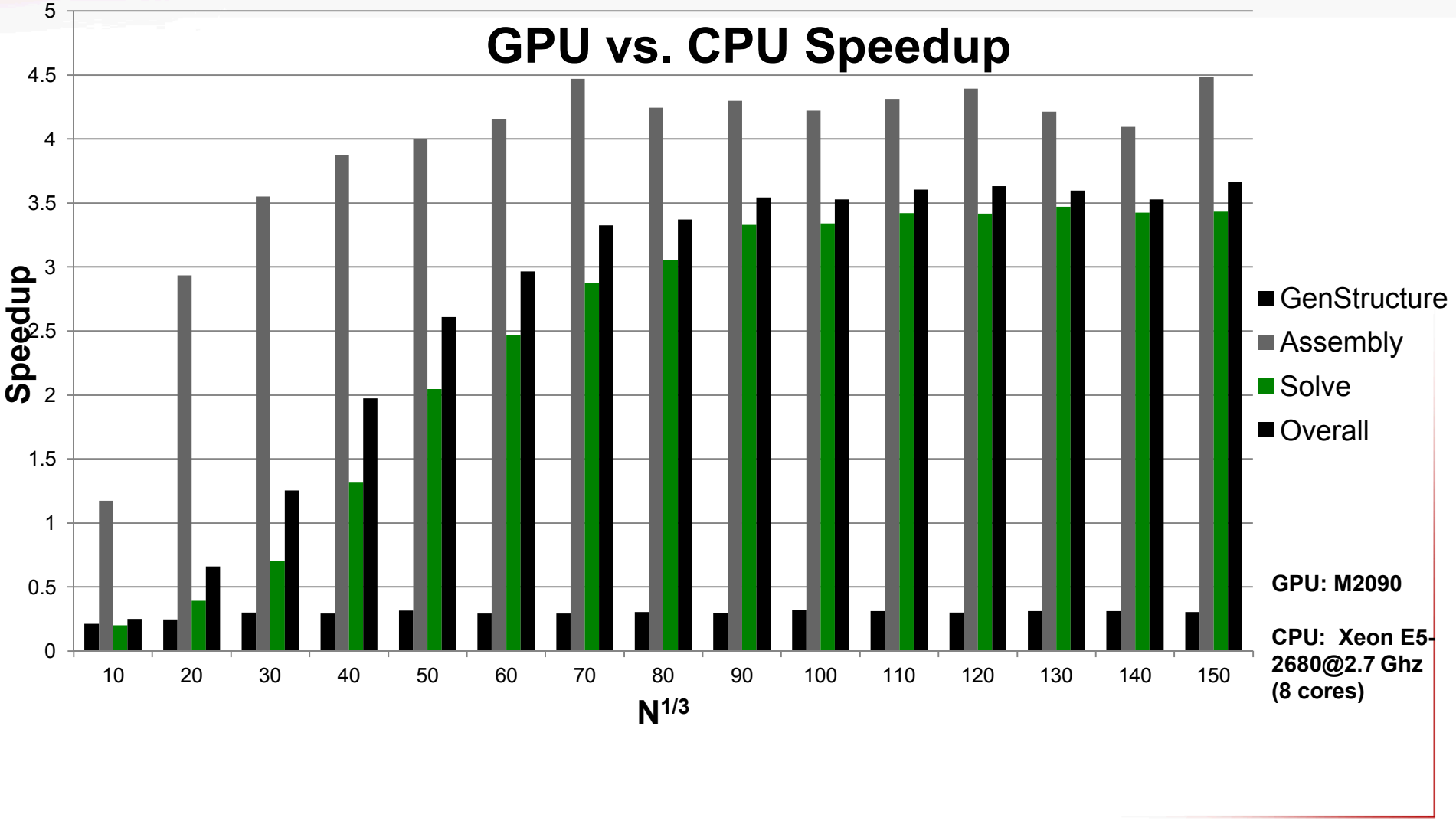
$\alpha_k = r_{k-1}^T r_{k-1} / p_k^T A p_k$

$x_k = x_{k-1} + \alpha_k p_k$

$r_k = r_{k-1} - \alpha_k A p_k$

}

Performance Results



Sandia National Laboratories

MiniFE/GPU Studies*

Mantevo/MiniFE = FE construction + assembly + solve

Activities:

- Port & refactor MiniFE.
- Study bottlenecks.

Findings:

- Simple lock-based assembly sufficient (<1%).
- FLOP-rich construction performance is suboptimal:
  - Requires a lot of state.
  - Leads to large amounts of register spilling.
  - Operation is bandwidth bound due to register spilling.

Take-away messages:

- (Nvidia) Increase max registers per thread:
  - More shared memory per thread.
  - Better register spilling.
  - Smarter compilers.
  - Faster register spilling.
  - Larger L1/L2 caches per thread.
- (Sandia) Reduce thread-state size:
  - Stripmine FE construction.
  - Explore other thread-to-work mappings.

*Luitjens, Williams, Heroux, *Optimizing miniFE an Implicit Finite Element Application on GPUs*, SIAM PP12, Savannah, GA, Feb, 2012.

Sandia National Laboratories

MiniMD on Testbeds

```
for (i = 0; i < nlocal; i++) {
    neighs = neighbor.firstneigh[i];
    numneigh = neighbor.numneigh[i];
    xtmp = x[i][0];
    ytmp = x[i][1];
    ztmp = x[i][2];

    for (k = 0; k < numneigh; k++) {
                j = neighs[k];
                delx = xtmp - x[j][0];
                dely = ytmp - x[j][1];
                delz = ztmp - x[j][2];
                rsq = delx*delx + dely*dely + delz*delz;

                if (rsq < cutforcesq) {
                            sr2 = 1.0/rsq;
                            sr6 = sr2*sr2*sr2;
                            force = sr6*(sr6-0.5)*sr2;

                            f[i][0] += delx*force;
                            f[i][1] += dely*force;
                            f[i][2] += delz*force;
                            f[j][0] -= delx*force;
                            f[j][1] -= dely*force;
                            f[j][2] -= delz*force;
                }
        }
 }
```

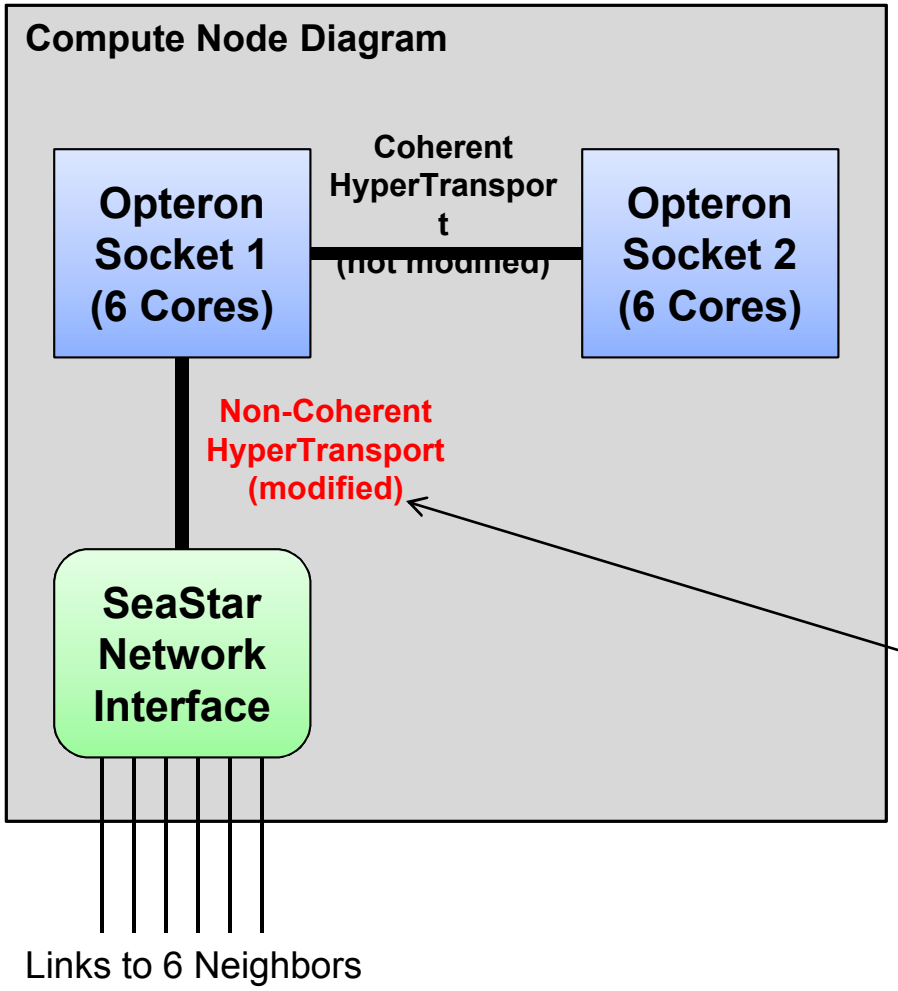
- **High proportion of time spent in force loop**
- **Does not vectorize automatically**
  - **Needs manual vectorization**
- **Threading is an even greater challenge**
- **Opportunity for gather/scatter memory operations – feedback from Intel**
- **Opportunity to employ masked vector operations to simplify code and improve vector code performance**

Sandia National Laboratories

Experiments and Emulation

Sandia National Laboratories

Network / Compute Balance Experiments on Red Storm and Cray XT5 Platforms



- **GOAL: Gather data to guide network bandwidth requirements for future systems**
- **Modified Cray BIOS to allow setting network injection bandwidth to speeds listed in table below**
- **Result is platform with configurable injection bandwidth**
  - **Tool for analyzing application sensitivity to system balance**
  - **Run full-scale experiments in real-time**

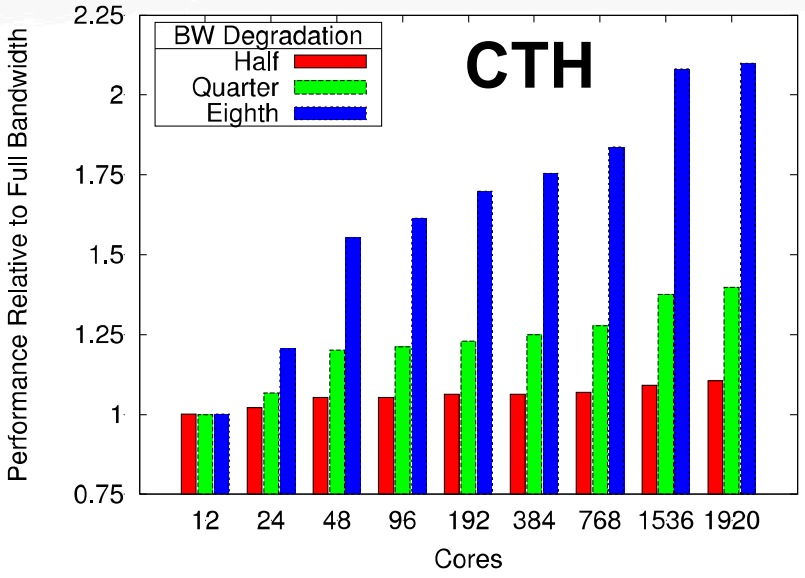
Link Frequency & Width	8 bits wide	16 bits wide
200 MHz	**400 MB/s**	**800 MB/s**
400 MHz	800 MB/s	**1600 MB/s**
800 MHz	1600 MB/s	**3200 MB/s**

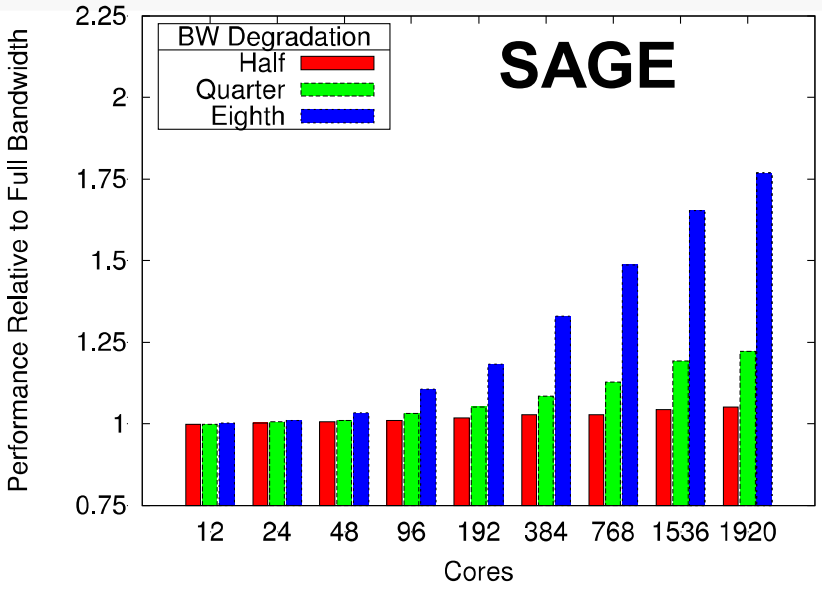
Theoretical Peak Bandwidths per-direction
HyperTransport is full-duplex

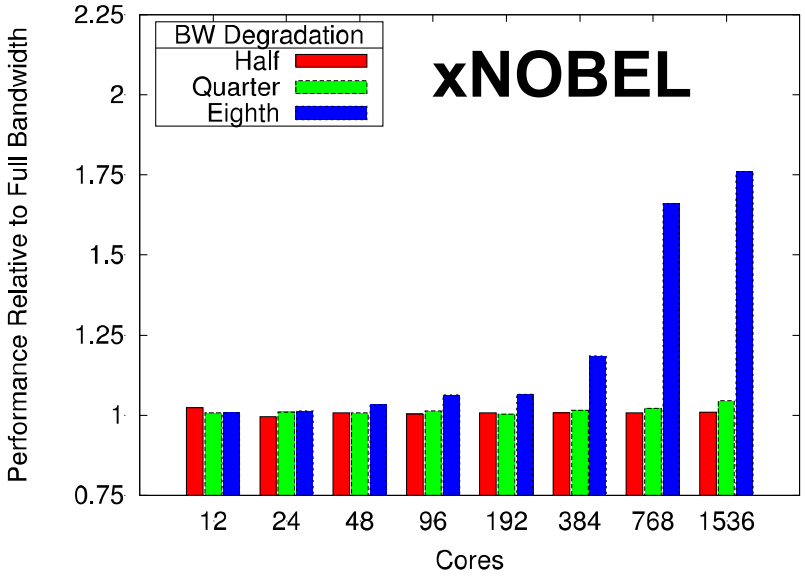
Sandia National Laboratories

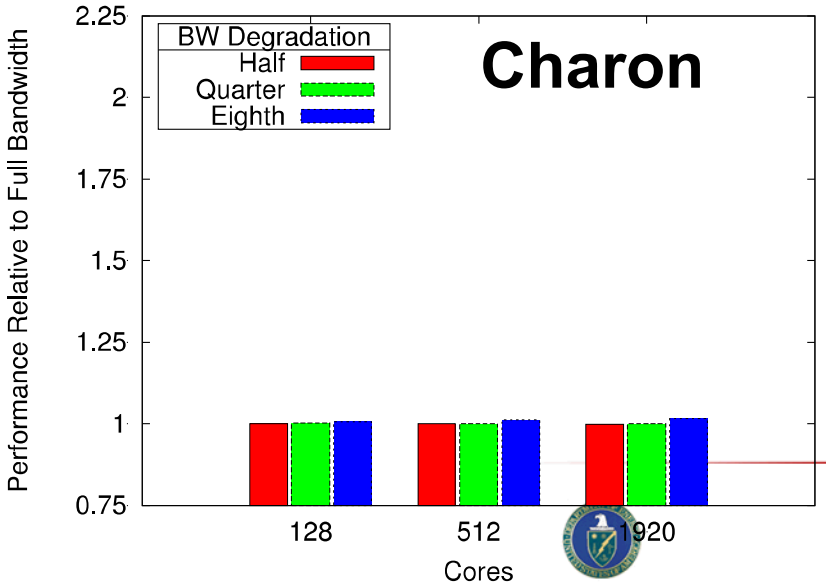
Application Results

CTH + Sage most impacted;
xNOBEL jumps suddenly with Eighth BW Degradation









Red Storm Network / Compute Balance Experiments

- **CTH 2048 nodes, 4096 cores:**

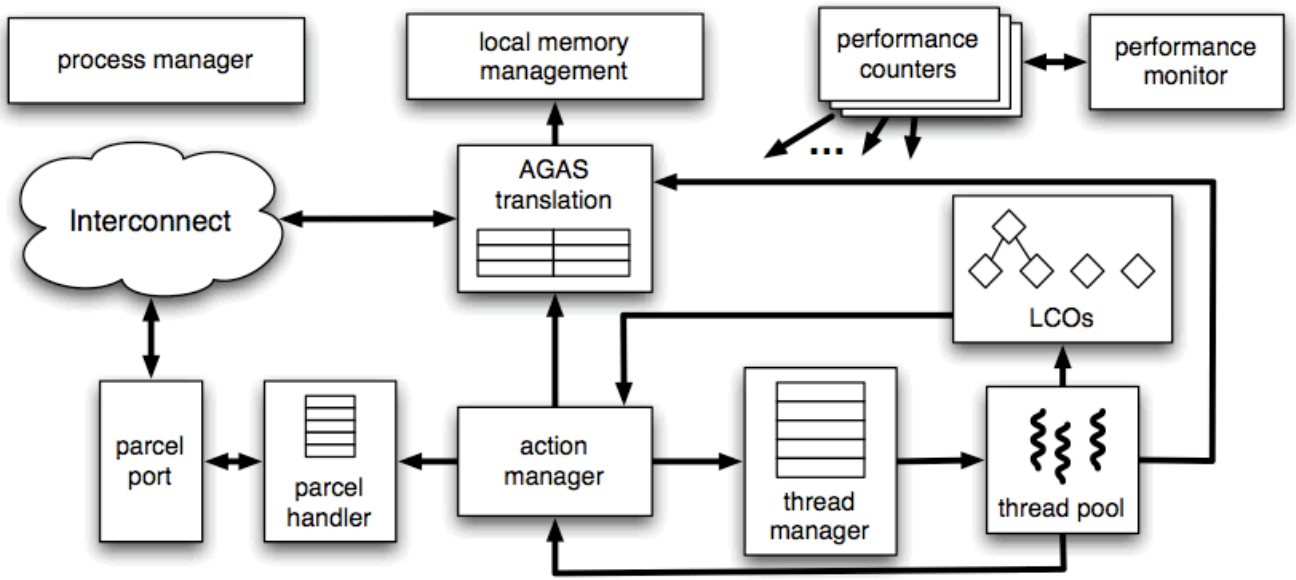
Network Speed	Runtime (s)	%Increase Runtime	%Increase Energy
Full	1519	--	--
Half	1671	10 %	(7 %)
Quarter	1975	30 %	**1 %**
Eighth	2127	40 %	**4 %**

- **AMG2006 2048 nodes, 4096 cores:**

Network Speed	Runtime (s)	%Increase Runtime	%Increase Energy
Full	859	--	--
Half	852	(1 %)	(16 %)
Quarter	858	0 %	**(23 %)**
Eighth	867	(1 %)	**(26 %)**

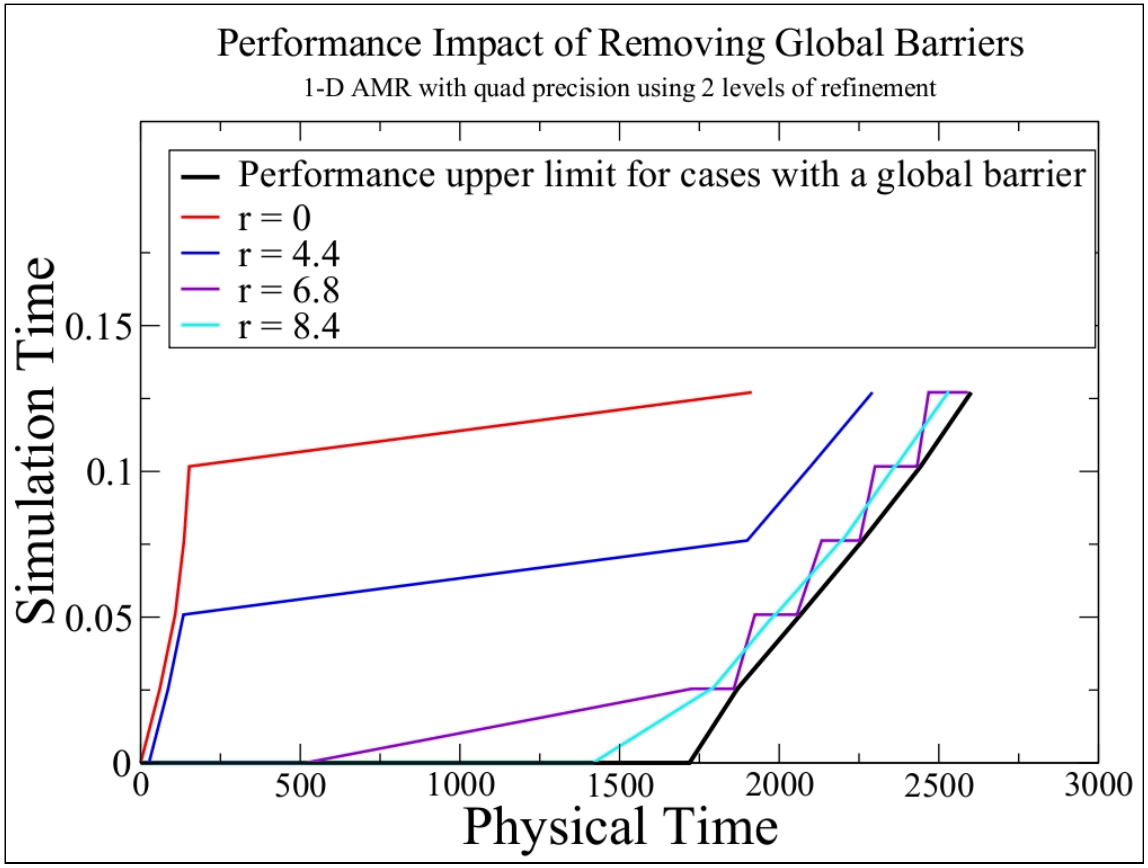
HPX Runtime Software System

- **Current version of HPX provides the following infrastructure as defined by the ParalleX execution model**
  - **Complexes (ParalleX Threads) and ParalleX Thread Management**
  - **Parcel Transport and Parcel Management**
  - **Local Control Objects (LCOs)**
  - **Active Global Address Space (AGAS)**



Courtesy of LSU and Indiana

HPX allows multiple simulation phases simultaneously

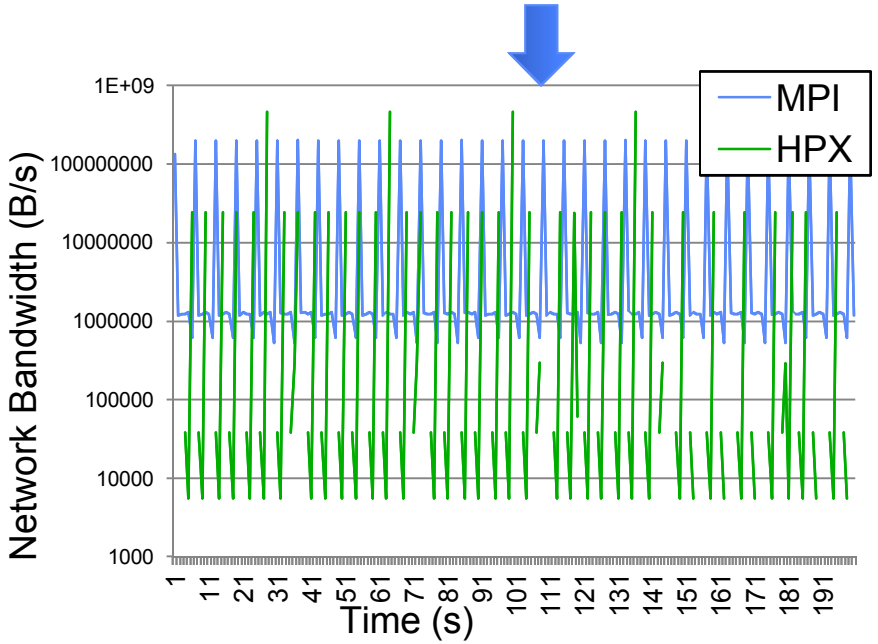


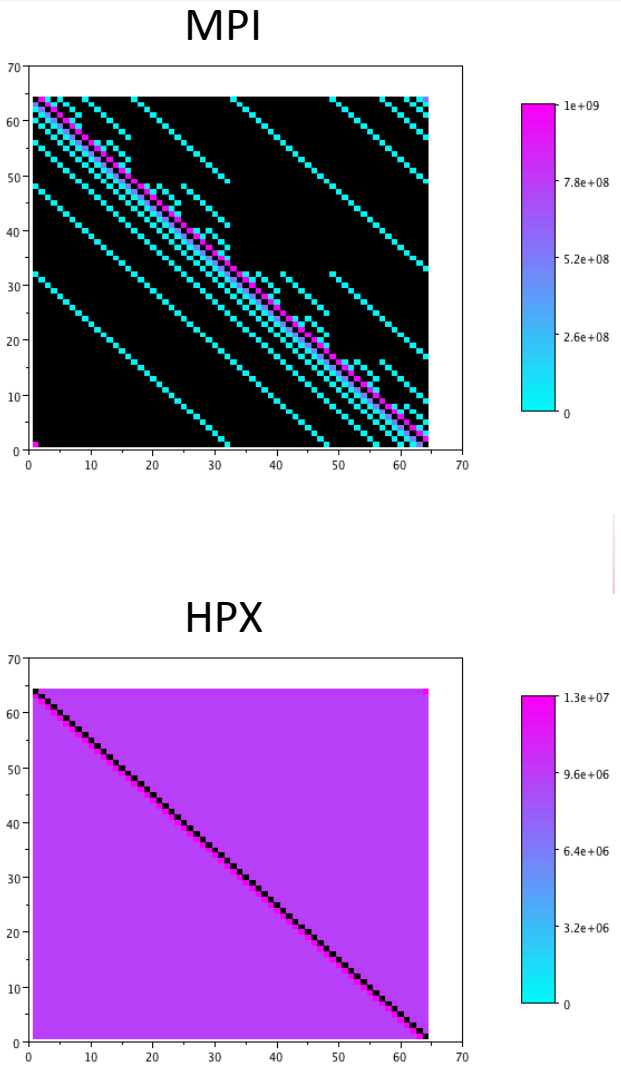
- **ParalleX based AMR removes all global computation barriers, including the timestep barrier (so not all points have to reach the same timestep in order to proceed computing)**

Sandia National Laboratories

GTC with static MPI vs. dynamically scheduled HPX

- **Preliminary experiments show asynchronous scheduling (HPX) changes the communication pattern vs. MPI**
- **Asynchronous communication (HPX) uses many more, much smaller messages, but less aggregate network bandwidth**

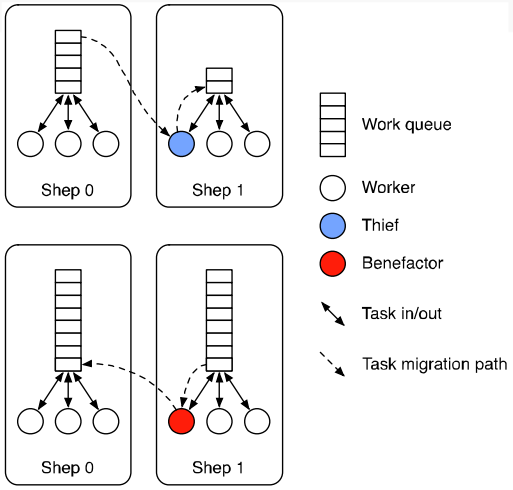




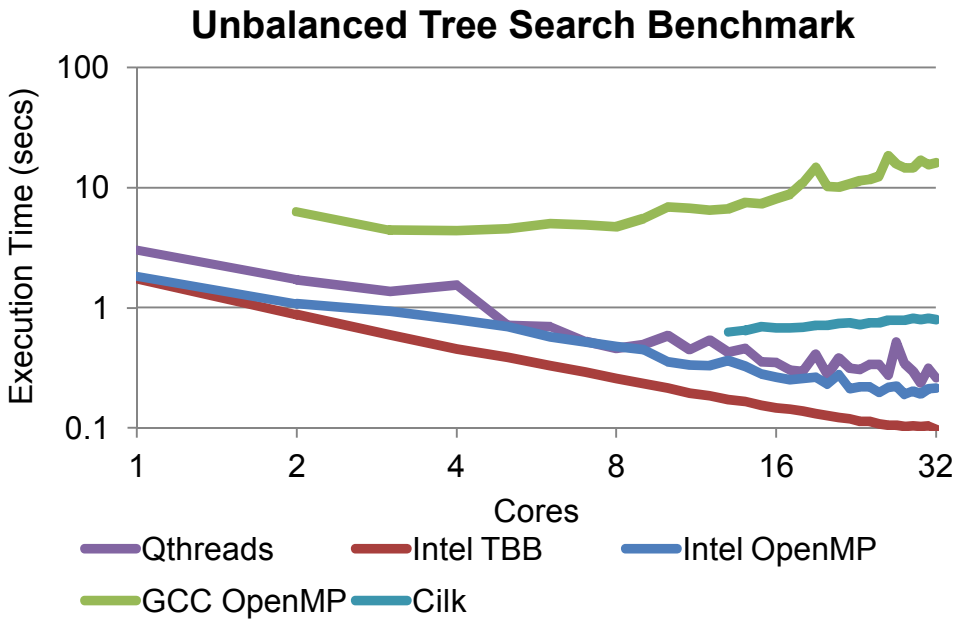
Sandia National Laboratories

Qthreads Lightweight Threading

- **Task-based runtime**
  - **Tool for programming model research**
  - **Supports both OpenMP-like models and more complex Chapel-like models**
  - **Presents simplified model of system to the application**
  - **High-performance scheduler**



High performance "sherwood" work-stealing scheduler effectively balances cache efficiency with load balancing.



Prediction

- Abstract Machine Models
- Proxy Applications
- Simulators

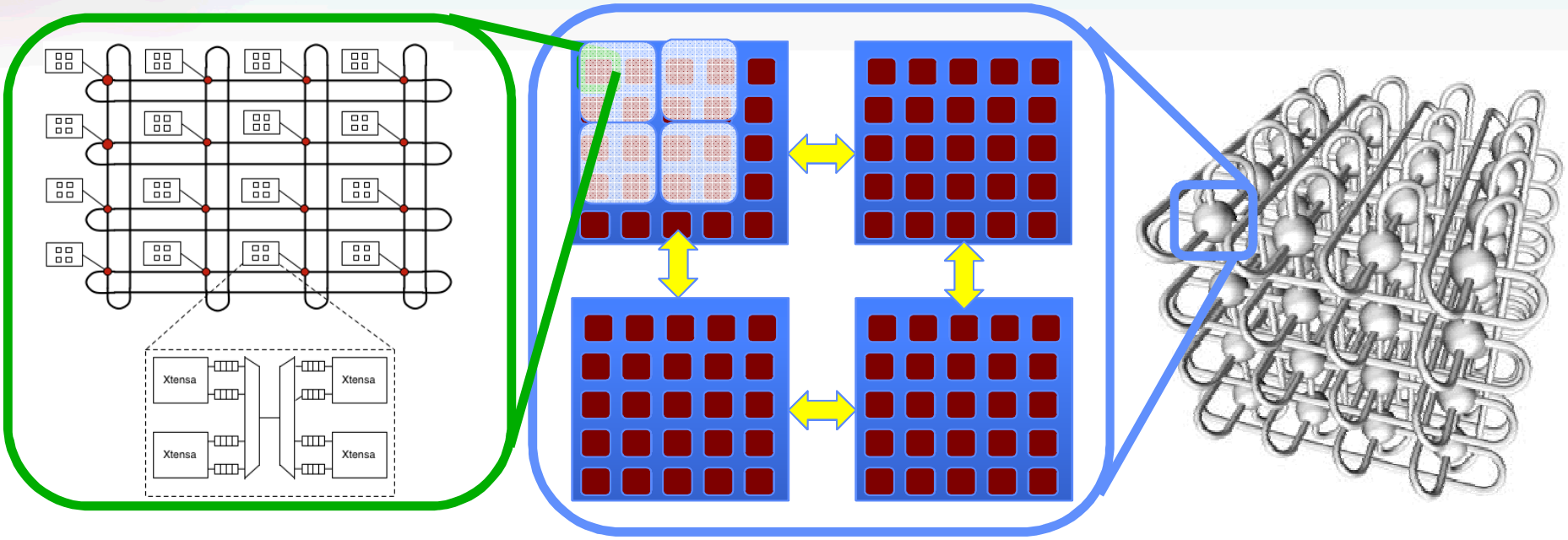
What is an Abstract Machine Model?

Definition: An Abstract Machine model represents the machine attributes that will be important to reasoning about code performance

- **Enables us to reason about how to map algorithm onto underlying machine architecture**
- **Enables us to reason about power/performance trade-offs for different algorithm or execution model choices**
- **Want model to be as simple as possible, but not neglect any aspects of the machine that are important for performance**

Notional Multi-Scale Abstract Machine Model

(what do we need to reason about when designing a new code?)



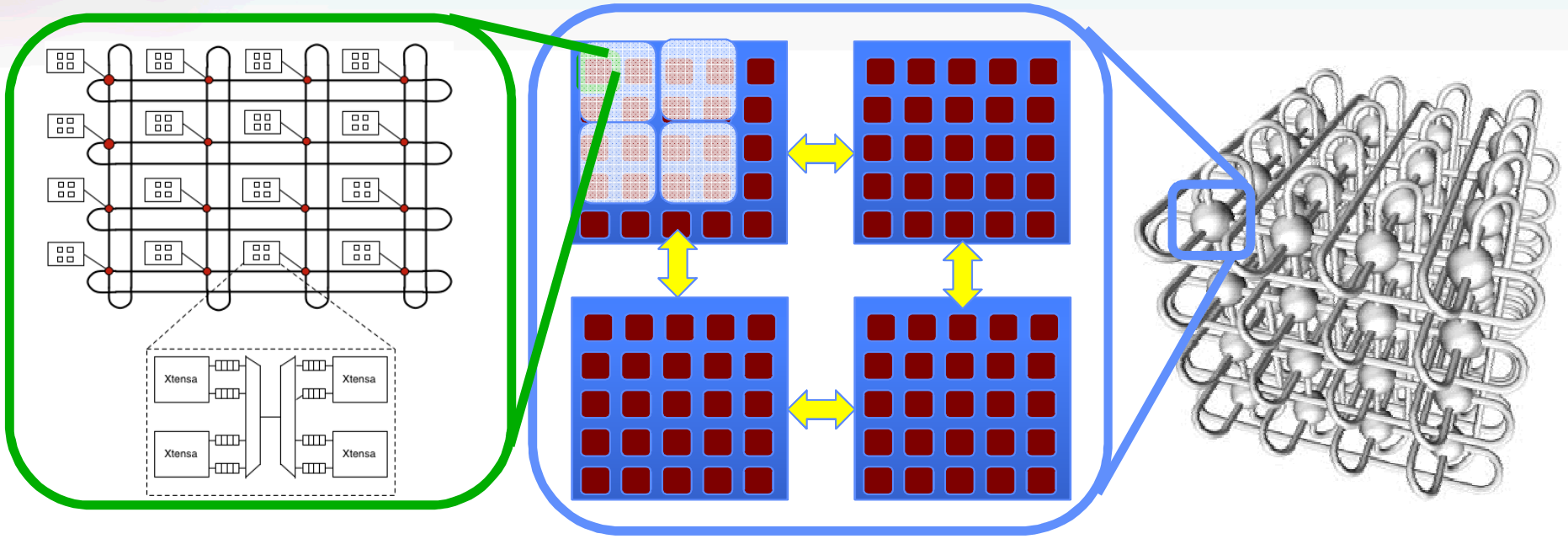
For each parameterized machine attribute, can

- **Ignore it:** If ignoring it has no serious power/performance consequences
- **Abstract it (*virtualize*):** If it is well enough understood to support an automated mechanism to optimize layout or schedule
  - **This makes programmers life easier (one less thing to worry about)**
- **Expose it (*unvirtualize*):** If there is not a clear automated way of make decisions
  - **Must involve the human/programmer in the process *(make pmodel more expressive)***
  - **Directives to control data movement or layout (for example)**

Want model to be as simple as possible, but not neglect any aspects of the machine that are important for performance

Notional Multi-Scale Abstract Machine Model

(what do we need to reason about when designing a new code?)



Cores
- •How Many
- •Heterogeneous
- •SIMD Width

Network on Chip (NoC)
- •Are they equidistant or
- •Constrained Topology (2D)

On-Chip Memory Hierarchy
- •Automatic or Scratchpad?
- •Memory coherency method?

Node Topology
- •NUMA or Flat?
- •Topology may be important
- •Or perhaps just distance

Memory
- •Nonvolatile / multi-tiered?
- •Intelligence in memory (or not)

Fault Model for Node
- • FIT rates, Kinds of faults
- • Granularity of faults/recovery

Interconnect
- •Bandwidth/Latency/Overhead
- •Topology

Primitives for data movement/sync
- •Global Address Space or messaging?
- •Synchronization primitives/Fences

Sandia National Laboratories

An SoC Model

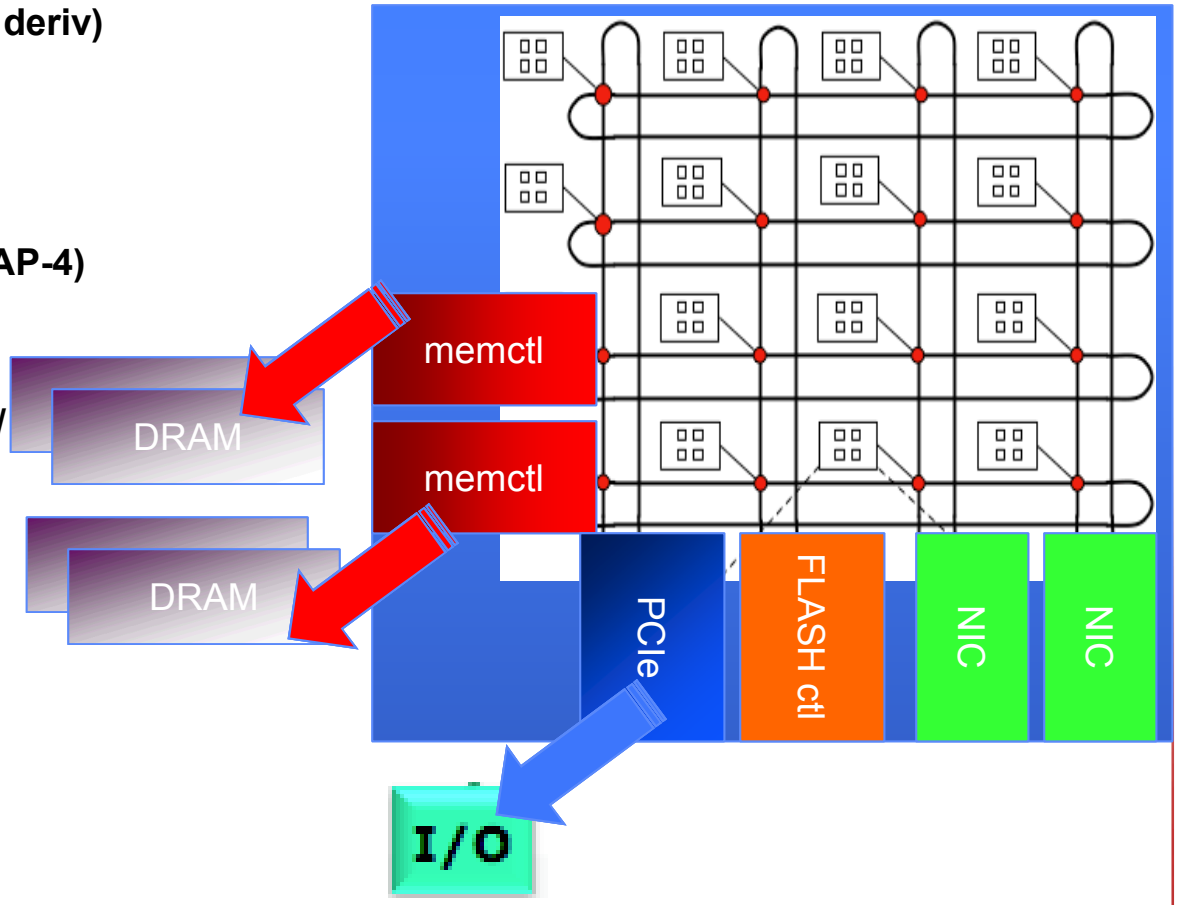
Processor Core (ARM, Tensilica, MIPS deriv)
With extra "options" like DP FPU, ECC

NoC Fabric: (Arteris, Denali, other OMAP-4)

DDR3 1600 memory controller (Denali / Cadence, SiCreations)
+ Phy and Programmable PLL

PCIe Gen3 Root complex

Integrated FLASH Controller

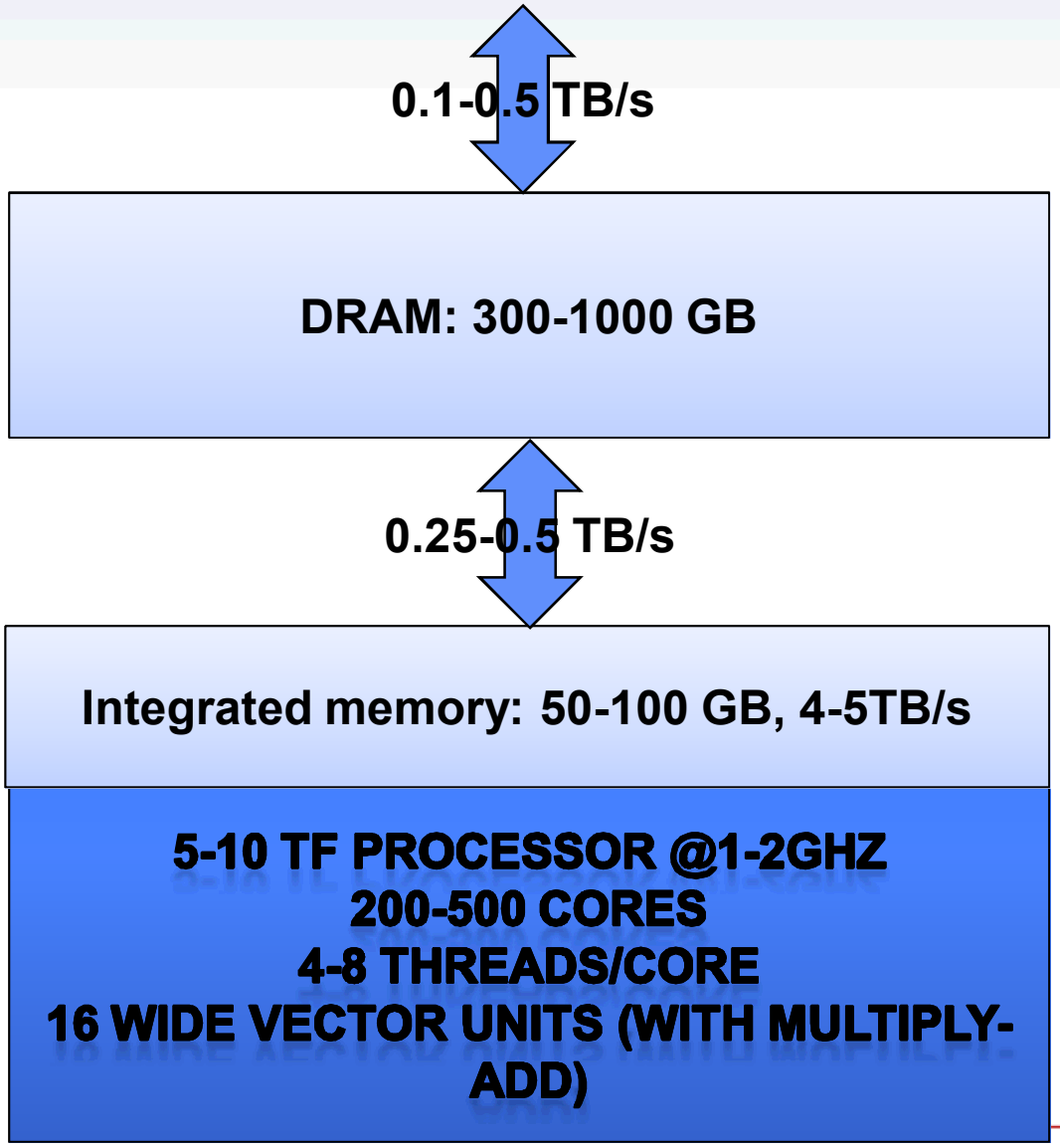


10GigE or IB DDR 4x Channel

Sandia National Laboratories

Draft FY13 L2: Study key performance Issues of ASC Applications executing on emerging technologies

- Testbeds
- Miniapps
- AMMs
- Simulation/Modeling



Sandia National Laboratories

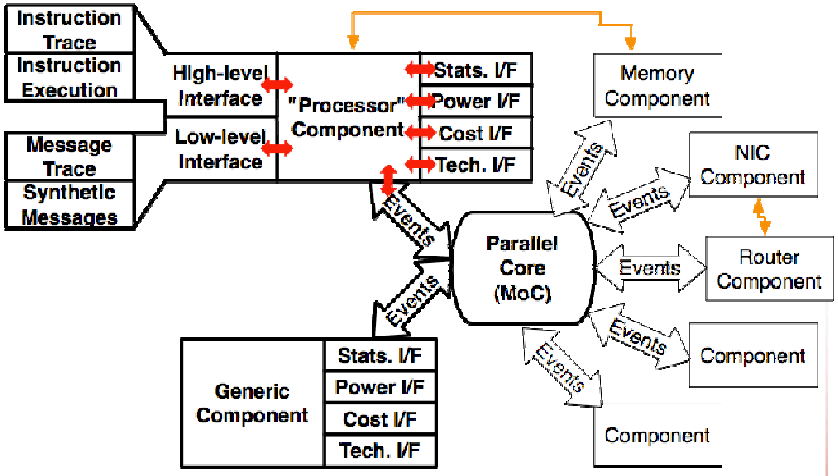
Hierarchical co-simulation is a key for co-design

Structural Simulation Toolkit

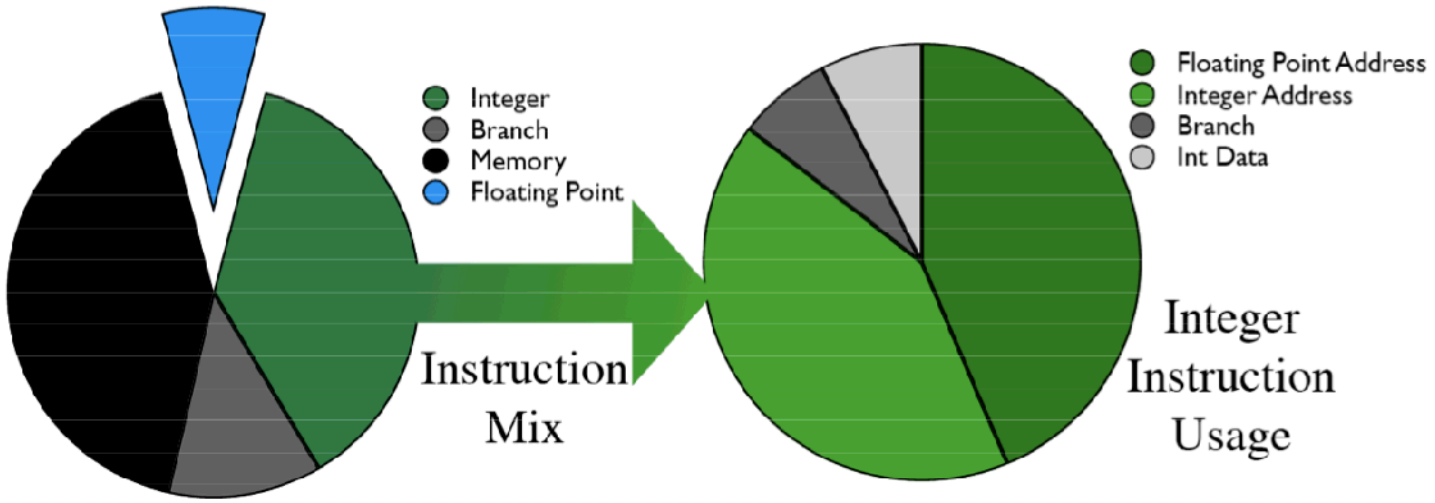
- Parallel
- Parallel Discrete Event core with conservative optimization over MPI
- Holistic
- Integrated Tech. Models for power
- McPAT, Sim-Panalyzer
- Multiscale
- Detailed and simple models for processor, network, and memory
- Current Release (2.0) at

http://www.cs.sandia.gov/sst/

- Includes parallel simulation core, configuration, power models, basic network and processor models, and interface to detailed memory model



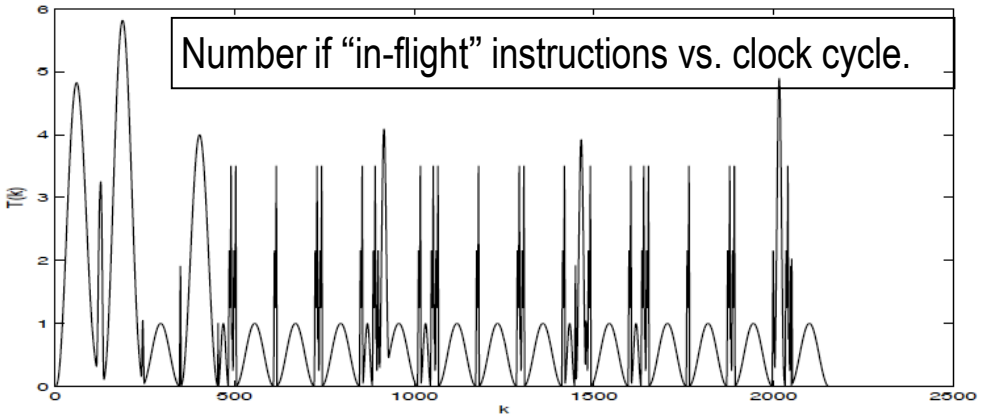
SST simulations have quantified the impact of the Memory Wall

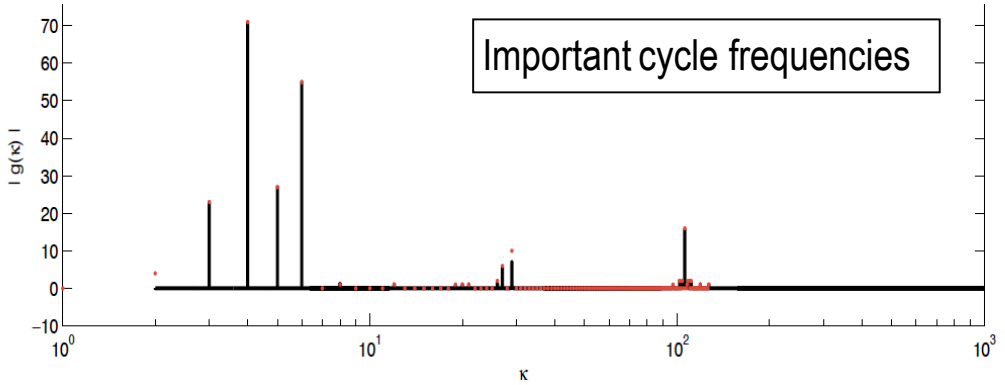


- Most of DOE's Applications (e.g., climate, fusion, shock physics, …) spend most of their instructions accessing memory or doing integer computations, not floating point
- Additionally, most integer computations are computing memory Addresses
- Advanced development efforts are focused on accelerating memory subsystem performance for both scientific and informatics applications

SST is providing architectural insights to algorithms developers

- **Input: SST Trace for SpMV.**
- **Lots of instruction stream data.**
- **Model: Use restricted *sin²* function to mark start/finish of each instruction.**
- **Use FFTs to analyze behavior.**





Trace fragment from SpMV inner loop

j	I_j	issue	complete	κ
59	bc	737	741	4
60	lwz	738	744	6
61	lfd	740	746	6
62	addi	742	746	4
63	addi	742	746	4
64	rlwinm	743	746	3
65	lfdx	744	850	106
66	fmadd	849	854	5
67	bc	850	854	4
68	lwz	851	857	6
69	lfd	853	859	6
70	addi	855	859	4
71	addi	855	859	4
72	rlwinm	856	859	3
73	lfdx	857	886	29
74	fmadd	885	890	5
75	bc	886	890	4
76	lwz	887	893	6
77	lfd	889	895	6
78	addi	891	895	4
79	addi	891	895	4
80	rlwinm	892	895	3
81	lfdx	893	899	6
82	fmadd	898	903	5
83	bc	899	903	4

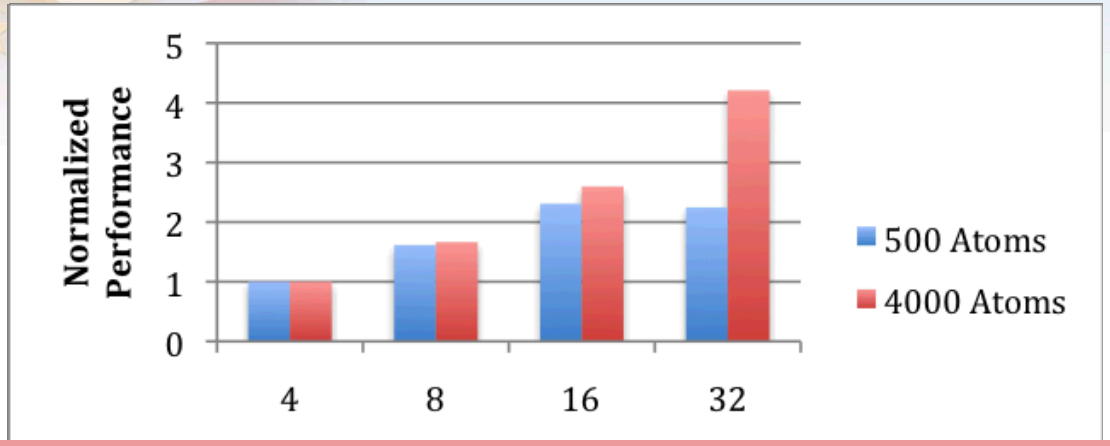
Component Validation

- Strategy: component validation in parallel with system-level validation
- Current components validated at different levels, with different methodologies
- Validation in isolation

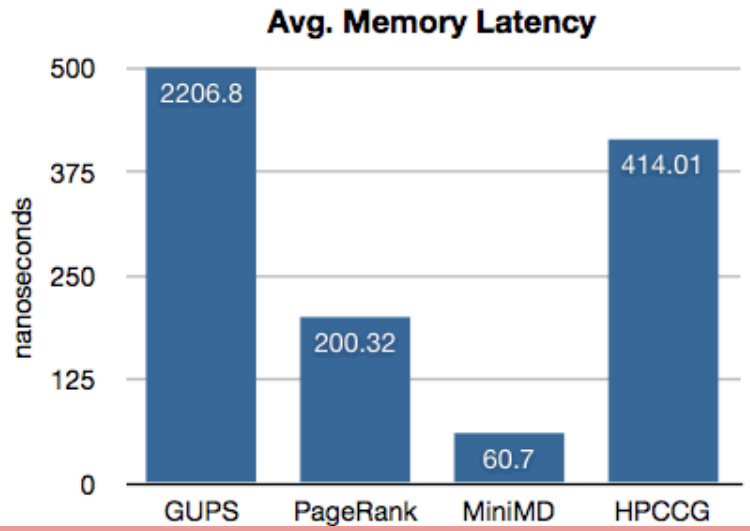
- What is needed
  - Uniform validation methodology (apps)
  - System (multi-component) level validation

Component	Method	Error
DRAMSim	RTL Level validation against Micron	Cycle
Generic Proc	Simplescalar SPEC92 Validation	~5%
NMSU	Comparison vs. existing processors on SPEC	<7%
RS Network	Latency/BW against SeaStar 1.2, 2.1	<5%
MacSim	Comparison vs. Existing GPUs	Ongoing <10% expected
Zesto	Comparison vs several processors, benchmarks	4-5%
McPAT	Comparisons against existing processors	10-23%

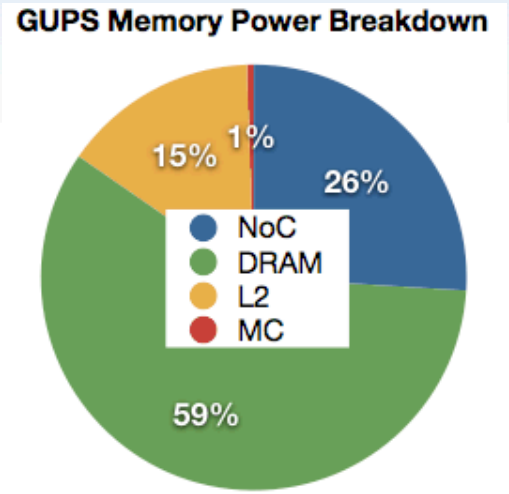
Sample Results –Node Level

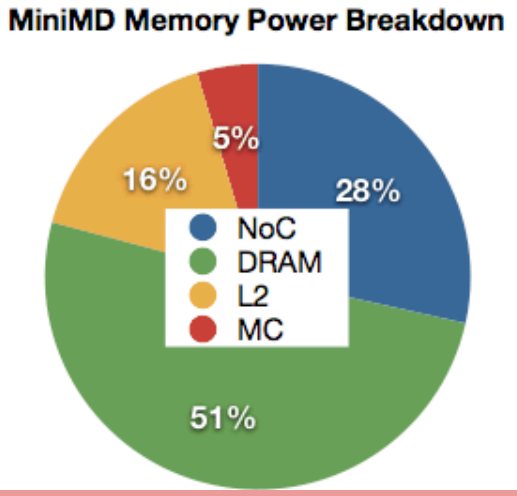


SST Simulation of MD code shows diminishing returns for threading on small data sets



Detailed component simulation highlights bottlenecks





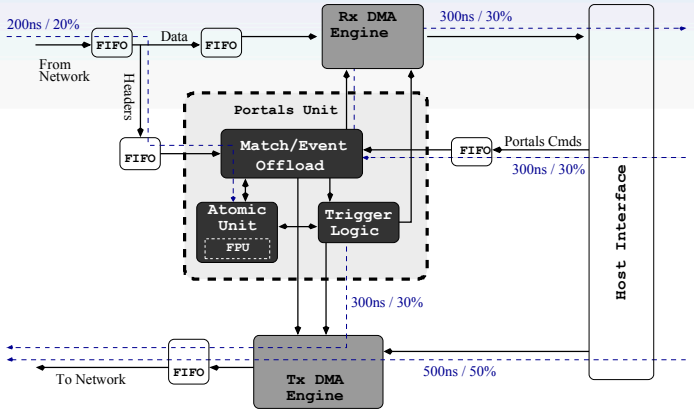
Power analysis help prioritize technology investments

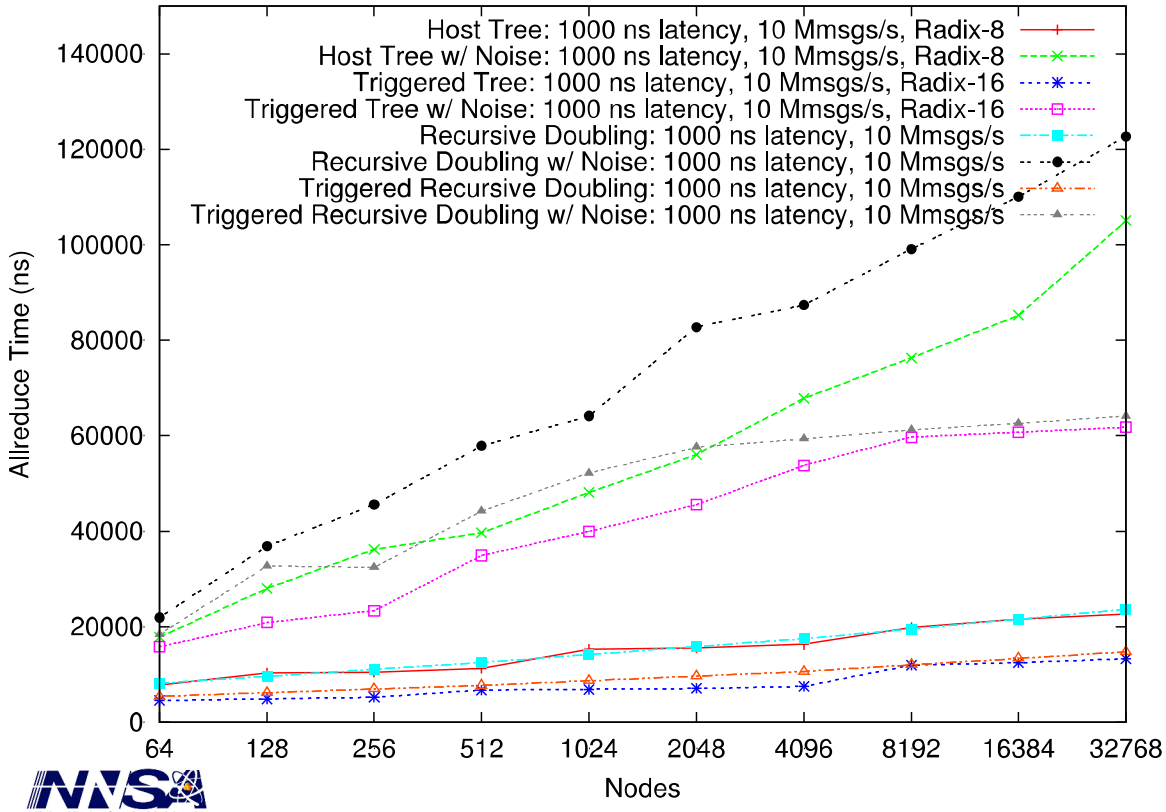
NNSA National Nuclear Security Administration

Sandia National Laboratories

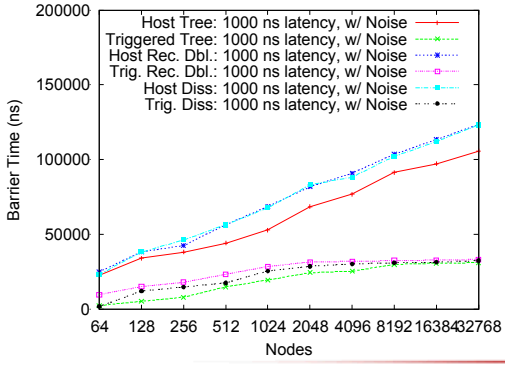
Sample Results – System Level

Simulation of new network API semantics (triggered operations) enabling flexible collective offload shows advantages in latency and noise tolerance





Simulation uses validated Red Storm router model coupled with a block-level NIC model(shown above) and a high level processor model



National Nuclear Security Administration

Sandia National Laboratories

Conclusions

- Co-design needs to impact architectures and applications/algorithms
- Our strategy for co-design has 3 elements: Measurements, experiments and prediction
- We need to speak with one voice to have any chance of influencing industry (especially processors and memory)
- Abstract machine models are one mechanism for engaging the broader research community and overcoming IP issues

Sandia National Laboratories

Backup Slides

Related Sandia L2 Milestones

FY12 L2:

Characterize the Role of the Mini-Application in Predicting Key Performance Characteristics of Real Applications. The Mantevo project includes a set of application proxies, referred to as mini-apps, and designed by code developers to represent key runtime performance characteristics of their applications. SNL will analyze two of these mini-apps to determine how well they represent the full application programs. Specifically, SNL will profile the runtime performance of the mini-app and application, characterizing the relationship between the two on at least two HPC platforms (including Cielo).

Draft FY13 L2:

Study of Key performance Issues of ASC Applications Executing on Emerging Technologies. Next generation computing platforms are expected to present significantly different architectural designs from the previous several generations. In preparation for these changes, we will explore the potential computing environments from processor core, to node, to inter-node. Our tools include a set of application proxies (called miniapps), a set of testbeds, simulation capabilities provided by the Structural Simulation Toolkit, abstract machine models, and analytic performance models. The outcome will be a better understanding of the characteristics and capabilities within the context of the computational science and engineering simulations of interest to the ASC program on emerging and future architectures and will inform hardware and software requirements.

Miniapps: Specs

- **Size: O(1K) lines.**
- **Focus: Proxy for key app performance issue.**
- **Availability: Open Source.**
- **Scope of allowed change: Any and all.**
- **Intent: Co-design: From HW registers to app itself.**
- **Developer & owner: *Application team*.**
- **Lifespan: *Until it's no longer useful.***

Sandia National Laboratories

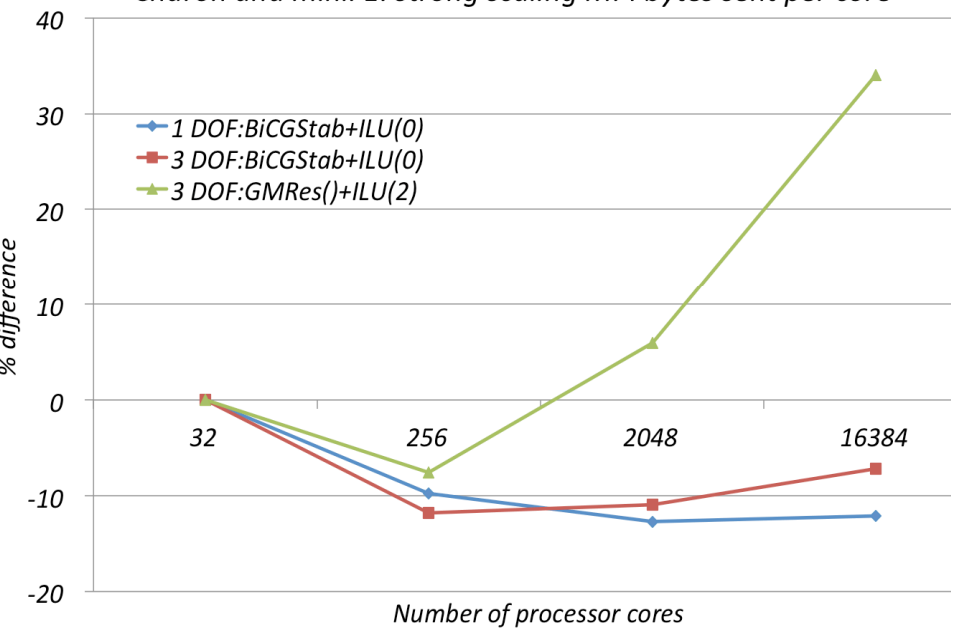
Charon Complexity

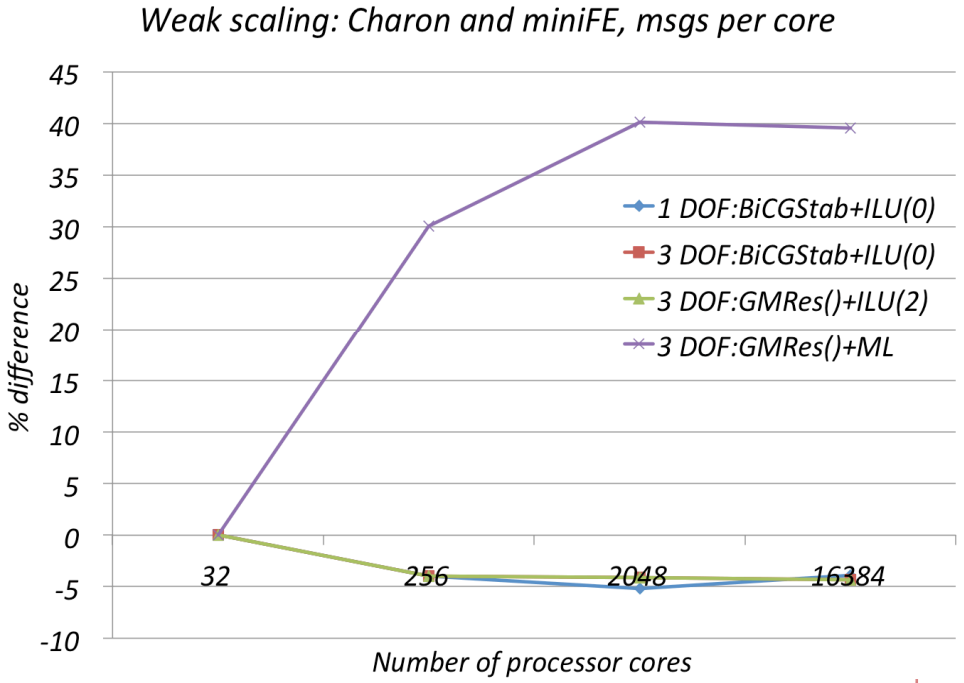
- SLOCCOUNT (tool from David A. Wheeler)
  - Charon physics: 191,877 SLOC.
  - Charon + nevada framework 414,885 SLOC
  - Charon_TPL 4,022,296 SLOC
- Library dependencies:
  - 25 Trilinos package.
  - 15 other TPLs.
- Requires “heroic effort” to build
- MPI-only, no intranode parallelism
- Export controlled

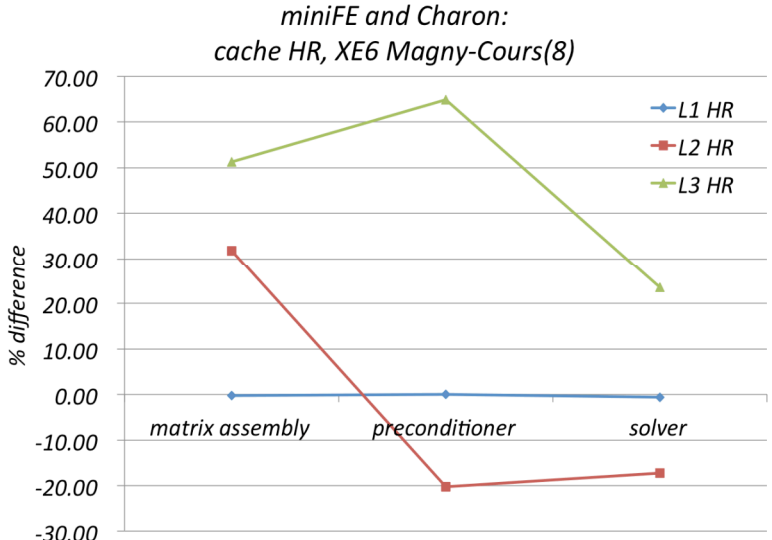
MiniFE Complexity

- **SLOCCOUNT:**
  - **Main code: 6,469 SLOC**
  - **Optional libraries (from Trilinos): 37,040 SLOC**
- **Easy to build:**
  - **Multiple targets:**
    - **Internode: MPI or not.**
    - **Intranode: Serial, Pthreads, OpenMP, TBB, CUDA.**
  - **Dialable properties:**
    - **Compute load imbalance.**
    - **Communication imbalance.**
    - **Data types: float, double, mixed.**
- **Open source**

Sandia National Laboratories



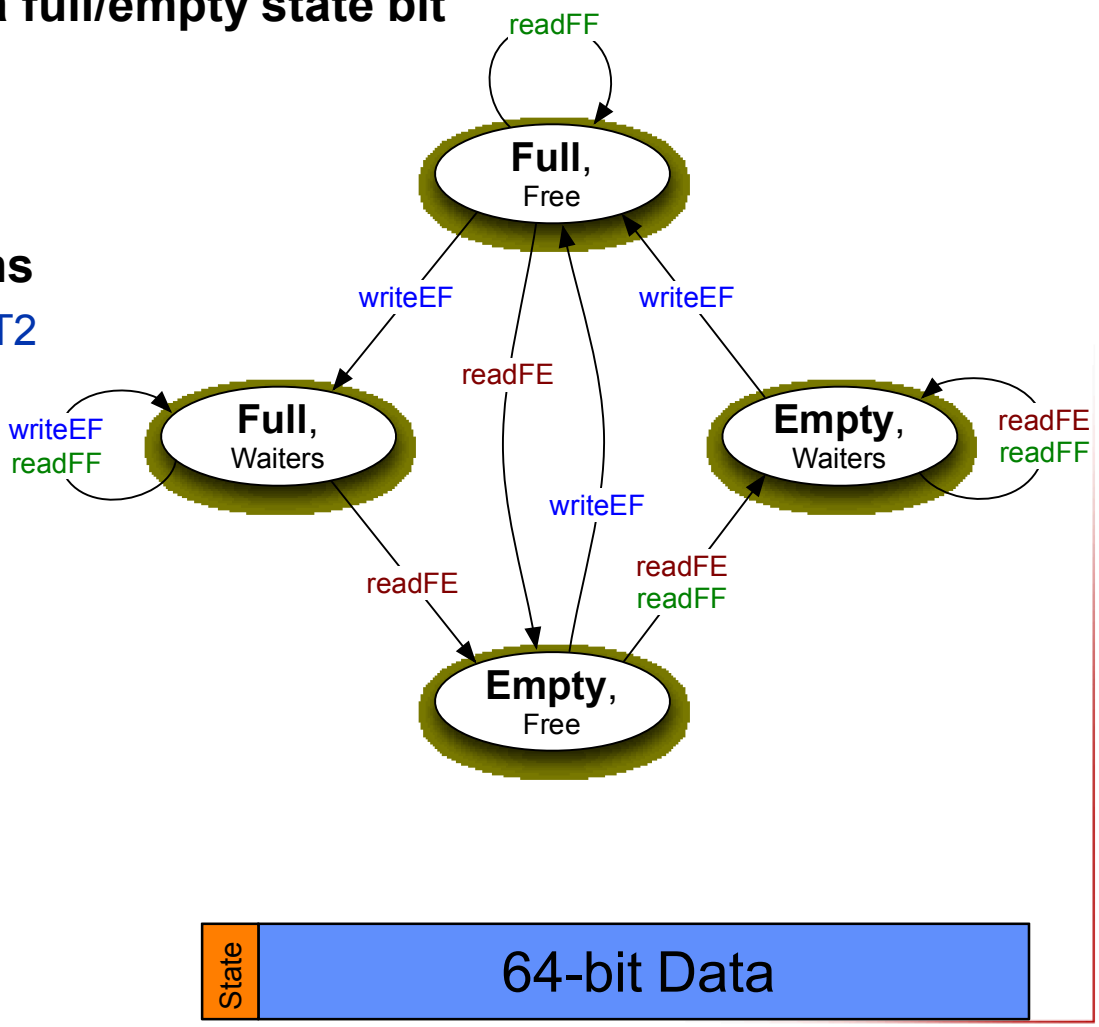




Sandia National Laboratories

HW Synchronization: Full/Empty Bits

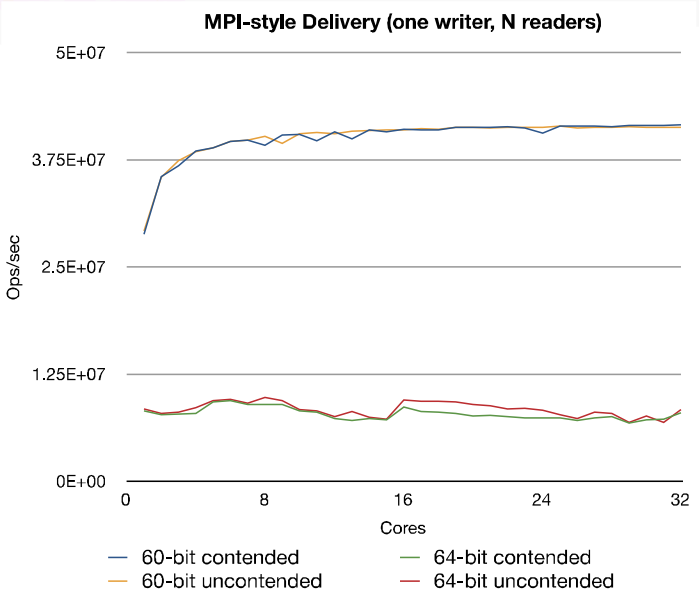
- **Every word in memory gets a full/empty state bit**
- **Reads/writes can:**
  - wait for a precondition state
  - modify state atomically
- **Previous HW Implementations**
  - Tera/Cray MTA, Cray XMT, XMT2
  - MIT Alewife
  - Denelcor HEP

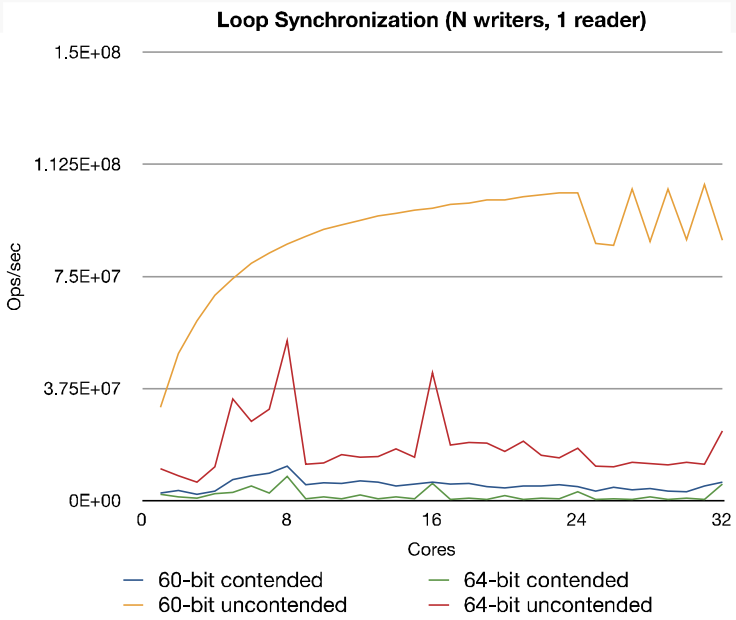


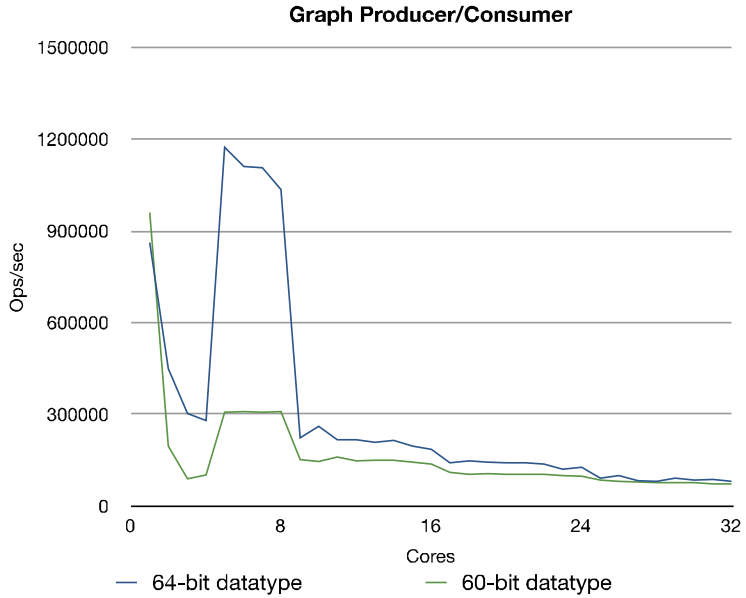
FEB Use Cases & Users

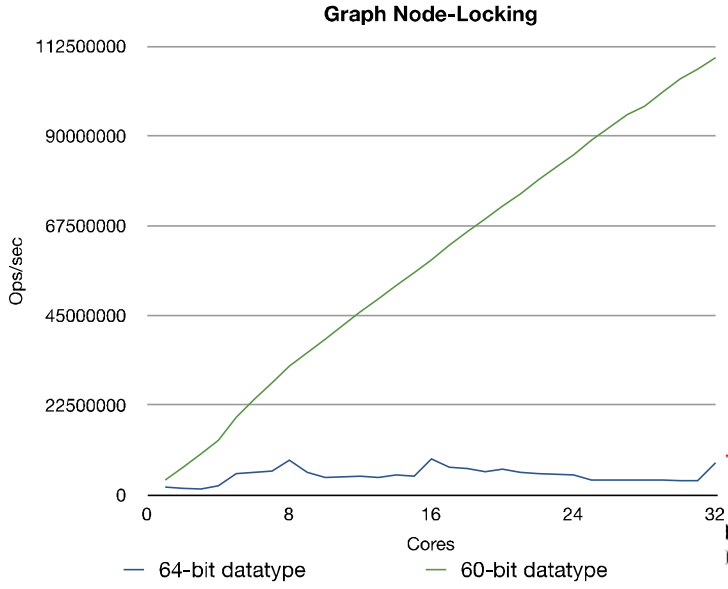
- **Use Cases**
  - Producer/Consumer
    - Communication/Computation Overlap
    - Loop Synchronization
  - Low-Contention Low-footprint Safety
    - Graph node locking
  - Combination
    - Graph node prod/cons
- **Users**
  - Multi-Threaded Graph Library
    - Based on Cray MTA/XMT
    - Safety
  - Chapel
    - General purpose synchronization primitive
    - Basic datatype modifier
  - Convey CHOMP Runtime
    - Provides for “nearly-atomic” operations
    - Better semantics than “try again”
    - Simplifies lock-free algorithms

Implementation Comparisons









Does MiniFE Predict Charon Behavior? Processor Ranking: 8 MPI tasks; 31k DOF/core

- Charon steady-state drift-diffusion BJT
- Nehalem (Intel 11.0.081 –O2 –xsse4.2; all cores of dual-socket quadcore)
- 12-core Magny-Cours (Intel 11.0.081 –O2; one socket, 4 MPI tasks/die)
- Barcelona (Intel 11.1.064 –O2; use two sockets out of the quad-socket)
- 2D Charon (3 DOF/node) vs. 3D MiniFE; match DOF/core and NNZ in matrix row
- Charon LS w/o or w/ ps: GMRES linear solve without/with ML precond setup time
- Try to compare MiniFE “assembling FE”+”imposing BC” time with Charon equivalent

MiniFE

	CG	FE assem+BC
1	Nehalem	Nehalem
2	MC(1.7)	MC(1.7)
3	Barc(2.7)	Barc(1.8)

Charon

	LS w/o ps	LS w/ ps	Mat+RHS
1	Nehalem	Nehalem	Nehalem
2	MC(1.7)	MC(1.8)	MC(1.46)
3	Barc(2.8)	Barc(2.5)	Barc(1.52)

Number in parenthesis is factor greater than #1 time

MiniFE Predict Charon? Multicore Efficiency Dual-Socket 12-core Magny-Cours : 124k DOF/core

- Charon steady-state drift-diffusion BJT; Intel 11.0.081 –O2
- Weak scaling study with 124k DOF/core
- 2D Charon (3 DOF/node) vs. 3D MiniFE; match DOF/core and NNZ in matrix row
- Efficiency: ratio of 4-core time to n-core time (expressed as percentage)
- Charon LS w/o or w/ ps: GMRES linear solve without/with ML precond setup time
- 100 Krylov iterations for both MiniFE and Charon (100 per Newton step)

MiniFE

cores	CG eff
4	Ref
8	89
12	73
16	61
20	54
24	45

Charon

cores	LS w/o ps eff	LS w/ ps eff
4	Ref	Ref
8	87	89
12	74	78
16	61	66
20	49	54
24	40	45

MiniMD/CPU Studies: Sandia

Mantevo/MiniMD = Molecular Dynamics Lennard - Jones

Activities:

- Optimizing miniMD
- Prepare code for future architectures

Findings:

- Performance critical kernels do not vectorize automatically
- Challenging environment for programming models, compiler and application
- Direct impact on LAMMPS and other key MD codes
- Can improve performance of codes on existing machines

Take-away messages:

- (Intel) Need to improve methods for vectorization
- (Sandia) Investigate code vectorization on existing platforms (and prepare for the future):
  - Look at code structure, pointer restriction, manual inlining etc
  - Introduce intrinsics to manually vectorize