Application (Code/Development) Requirements for PM/Es

James Belak (belak@llnl.gov)
www.exmatex.org

DOE Programming Models and Environments Workshop

Rockville, MD

9-11 March, 02015

'Algorithm research has been driven by hard to use machines'
-Rob Schreiber (HP Labs)

'People who are serious about software should make their own hardware.'
- Alan Kay (Xerox PARC)

LLNL-PRES-XXXXXX

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and supported by Office of Science Advanced Scientific Computing Research Program.

Use-inspired research and development involves a dynamic interaction between both considerations of use and fundamental understanding.

Outline:

- Application Requirement for PM/E
- A Day in the Life of an Application Developer
- Abstractions for Execution, or how doe the developer think about the architecture
- MPI + X
- PM/E, or the mapping between computational “motifs” and execution “motifs”
- Requirements from the Co-design Program Meeting

Application Requirements for PM/E: a (useable) Productive, Portable, and Performance programming System/Framework that allows the app developer to reason about the machine (there is no consensus on App Requirements!)

- Productivity: the **effectiveness** of productive effort (produce **quality** code) as measured in terms of rate of output per unit of input (man years)
- Portable: ability to be **transferred** from one machine or system to another (node multicore/gpu, disparate exascale systems, petascale in a rack or Beowolf cluster)
- Performance: an action, task or operation seen in terms of how **successfully** is was performed (not just time to solution, but time to a better solution, better physics, the computational task that optimizes workflow, new science, or a better bridge)

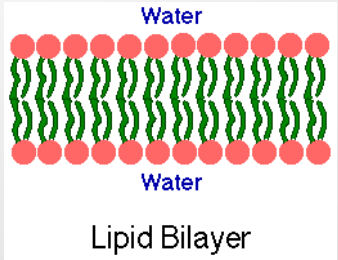
A Day in the Life of an Application Developer

- Check out current program (component) state: revision control system (svn, …)
- Edit the component I am working on (vi, emacs, …) and implement (translate) numerical algorithm to computer code for target machine from physics specification (this is where the programming model lives)
- Make (compile) on development resource: is the compiler informative if it does not make?
- Execute on development resource and test for correctness of physics, test on production system (at scale?)
- Check in component for production use

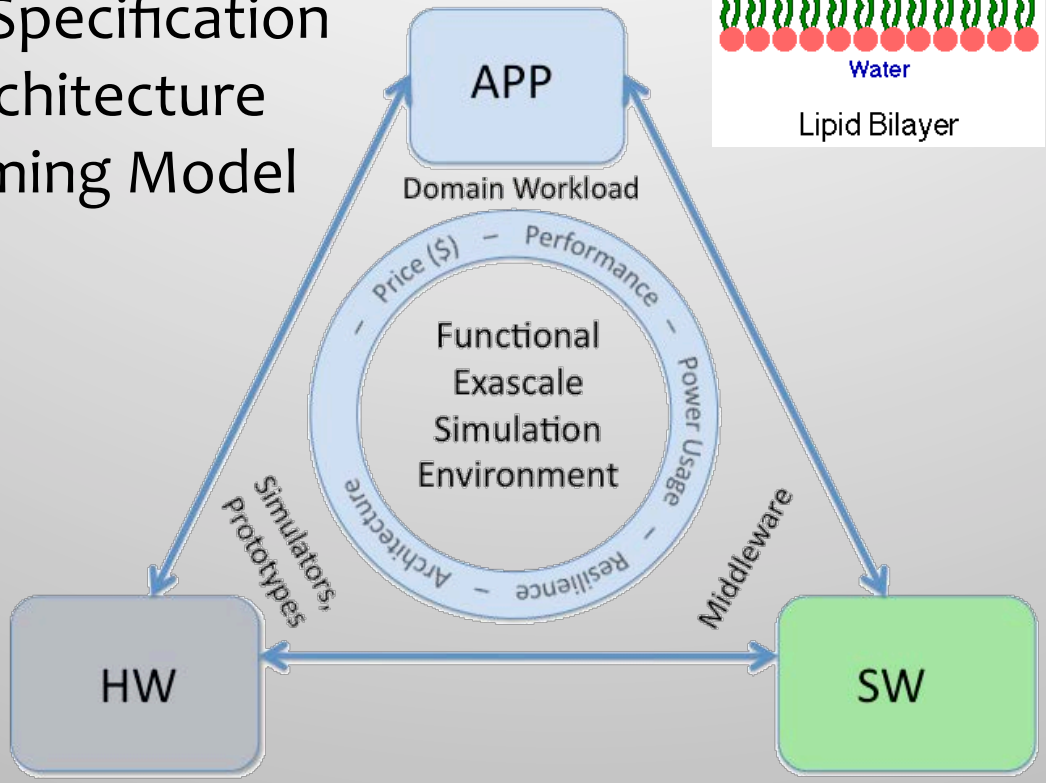
Creating Community Codes (1994) LAMMPS CRADA: Cray (Carpenter), Bristol-Myers (Stouch), Dupont (Lustig), LLNL (Belak), Sandia (Plimpton)

Three Essential Elements:

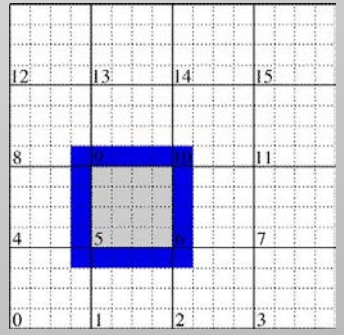
- Problem Specification
- Target Architecture
- Programming Model



Diffusion of a small drug molecule through a bio-membrane



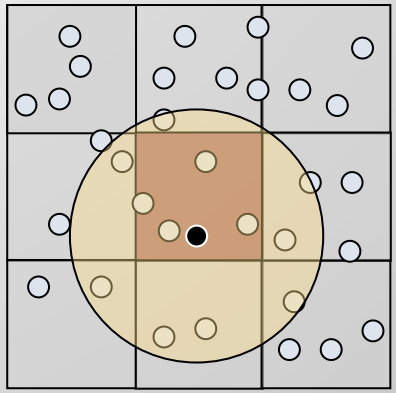
Cray T3D, Clock Speed: 0.15GHz, Date: 1994
Microprocessor Peak Teraflops: 0.02, Memory: 0.01TB
Processors: 128. Power Consumption: 41.40 kW



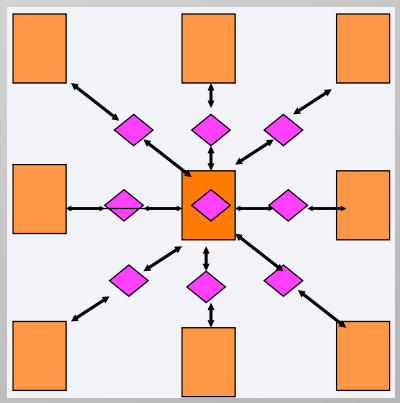
SPMD, MPI, Geometric domain decomposition

1997: LAMMPS became a successful community code because of Steve's idea to release it as open source, the ubiquitous Beowolf cluster and the stability of MPI.

- 20 Years Ago
  - Hardware: Linux Beowolf Cluster
  - Software Programming Model: MPI/SPMD
  - Application Code: LAMMPS
- 5+ Years in the Future
  - Hardware: Exascale (Petascale in a Rack)
  - Software Programming Model: MPI+X, Q? Task-based with "Control" of Data Locality (User and Runtime)
  - Application Codes???



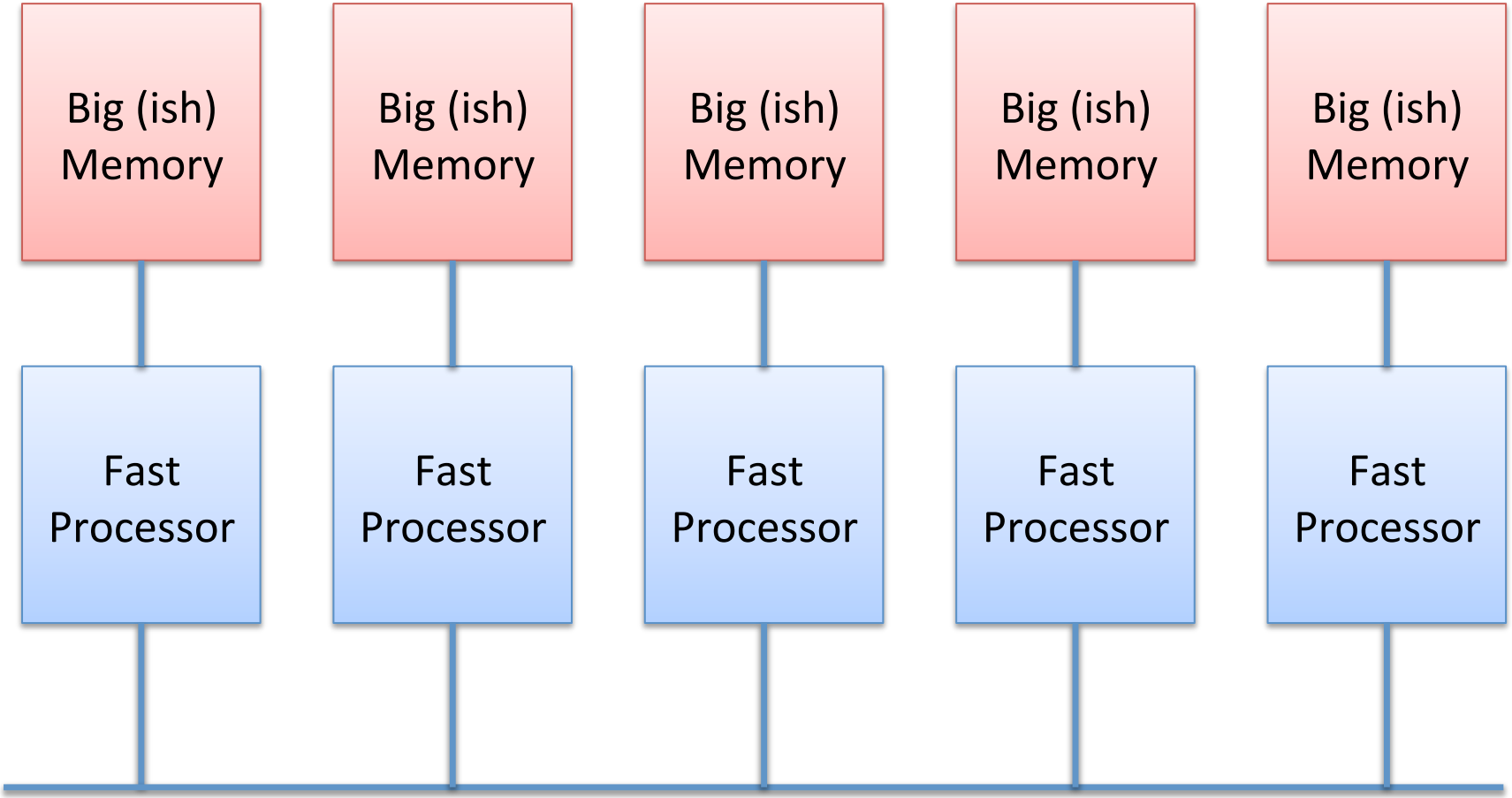
Worker



Work

How does the Application Developer think about the machine? (David Richards, LLNL)

Abstract machine model used for the last 20+ years



Network built from tin cans and string

What are the Abstractions of Execution (motifs) needed to represent this node?

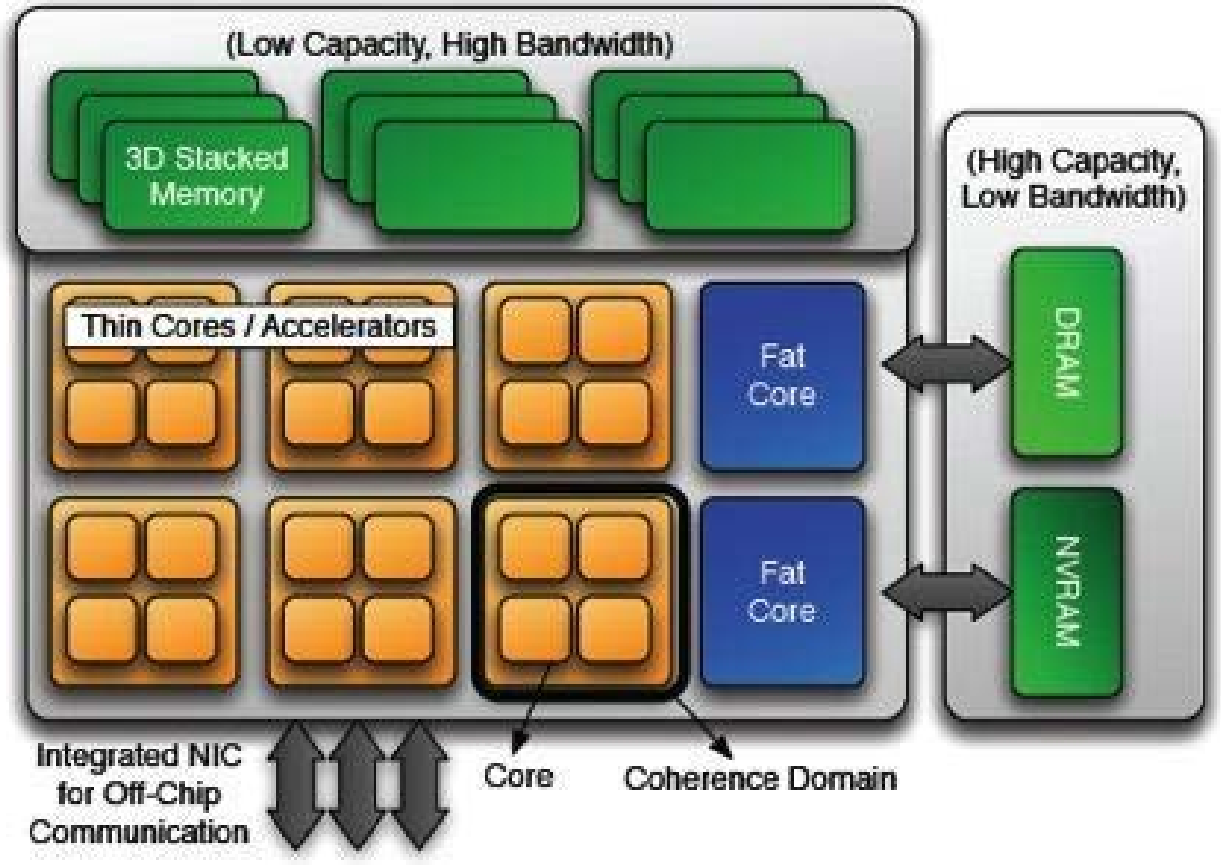


Figure 1: Abstract Node Architecture for Exascale

Petascale (exascale) application developers must optimize for a complex parallel machine

- (Inter-Node) **System-level**. No shared memory. Traditionally uses MPI to communicate data between disjoint address spaces.
- (Intra-Node) **Cores**. Modern nodes have multiple CPU cores. The work needs to be distributed across them. Old fashioned MPI is one option, but this increases surface to volume. Solutions such as OpenMP that acknowledge the shared address space (possibly NUMA) among the cores are probably preferred, especially in new code.
- (Intra-Node) **Threads**. Modern cores are supporting multiple hardware threads per node. Among other things, multiple threads per core cover latencies since some threads can typically proceed while others are stalled. Multiple threads may allow for better register usage, reduced pipeline stalls, etc.
- (Intra-Node) **SIMD**. We are now seeing quad-double SIMD units on intel and AMD hardware as well as BG/Q. Memory access need to be aligned to allow vector registers to be filled efficiently. Throwing away SIMD instructions is instantly giving away a factor of 4 in performance. On GPUs, warps are rather like SIMD instructions since all threads in a warp execute the same instruction.
- (Intra-Node) **Functional Units**. BG/Q has both an integer and a floating point unit for each core. It is the integer unit that loads data so in order to do useful processing you need to keep both active (to both load and process data). Note that any thread can only issue an instruction to one of the units per cycle so at least two threads are needed to fully exploit the units. It is also important to structure algorithms so that the use of the functional units is balanced.

MPI+X: What is a programming model? An execution model?

- I disagree with Bill Gropp. MPI is not a programming model. AMT, SPMD, MPMD, SIMD, SIMT, … are programming models (or are they execution models?).
- I agree with Bill Gropp. MPI is a programming system within which a code can be instantiated in a programming model.
- MPI was originally developed to support the distributed memory execution abstraction (i.e. the Beowolf cluster and was used to create the original LAMMPs MD code)
- Now, 20 years later, what execution abstractions does MPI support?
- What is the set of execution abstractions that an application developer needs to think about for exascale?

PM/E's (and associated RTs) are the mapping between computation motifs and execution motifs

Computation Motifs:

- Dense matrices
- Sparse matrices
- Structured grids
- Unstructured grids
- Particle methods
- Spectral methods
- Monte Carlo methods
- …

PM/Es

Execution Motifs:

- Inter node (distributed memory)
- Shared memory
- Multi-core
- SIMD / GPU
- Threads
- Functional units
- Multi-level memory
- …

Requirements from the Co-design Program Meeting

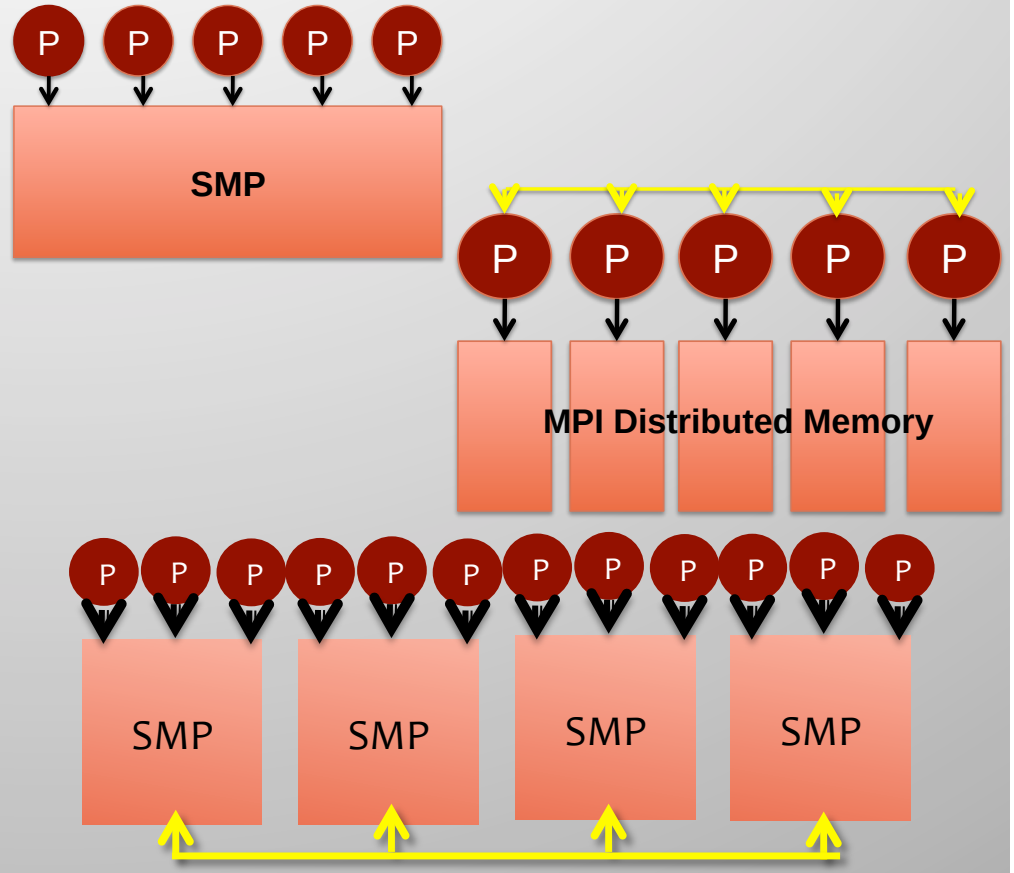
Wednesday, February 25, 2015
Day of Programming in the context of application & architecture

8:00	CORAL	Buddy Bland
8:30	Q&A	
9:00	Cori	Nick Wright
9:30	Q&A	
10:00	Break	
10:30	Panel: Progress on Programming	Alex Aiken, Tim Warburton, Ian Karlin, Jim Belak
11:10	Q&A, discussions	
12:00	lunch	
13:00	CAL	John Shalf
14:00	Theme 3: programming	3 Breakout Rooms
15:30	Break	
16:00	Report back from Theme 3; discussions	Breakout Facilitators
17:30	Day 2 adjourns	

The Programming Model is a Reflection of the Underlying *Abstract Machine Model (John Shalf)*

Martha Kim, Columbia U. Tech Report "Abstract Machine Models and Scaling Theory"
http://www.cs.columbia.edu/~martha/courses/4130/au13/pdfs/scaling-theory.pdf

- **Equal cost SMP/PRAM model**
  - No notion of non-local access
  - int [nx][ny][nz];
- **Cluster: Distributed memory model**
  - CSP: Communicating Sequential Processes
  - No unified memory
  - int [localNX][localNY][localNZ];
- **2-level (CTA in Martha Kim Taxonomy)**
  - Candidate Type Architecture (CTA)
  - MPI+X model (for all practical purposes)



- **Whats Next?**

2-Level MPI+X is dominant, but insufficient!

How does the Application Developer think about this node (David Richards, LLNL)

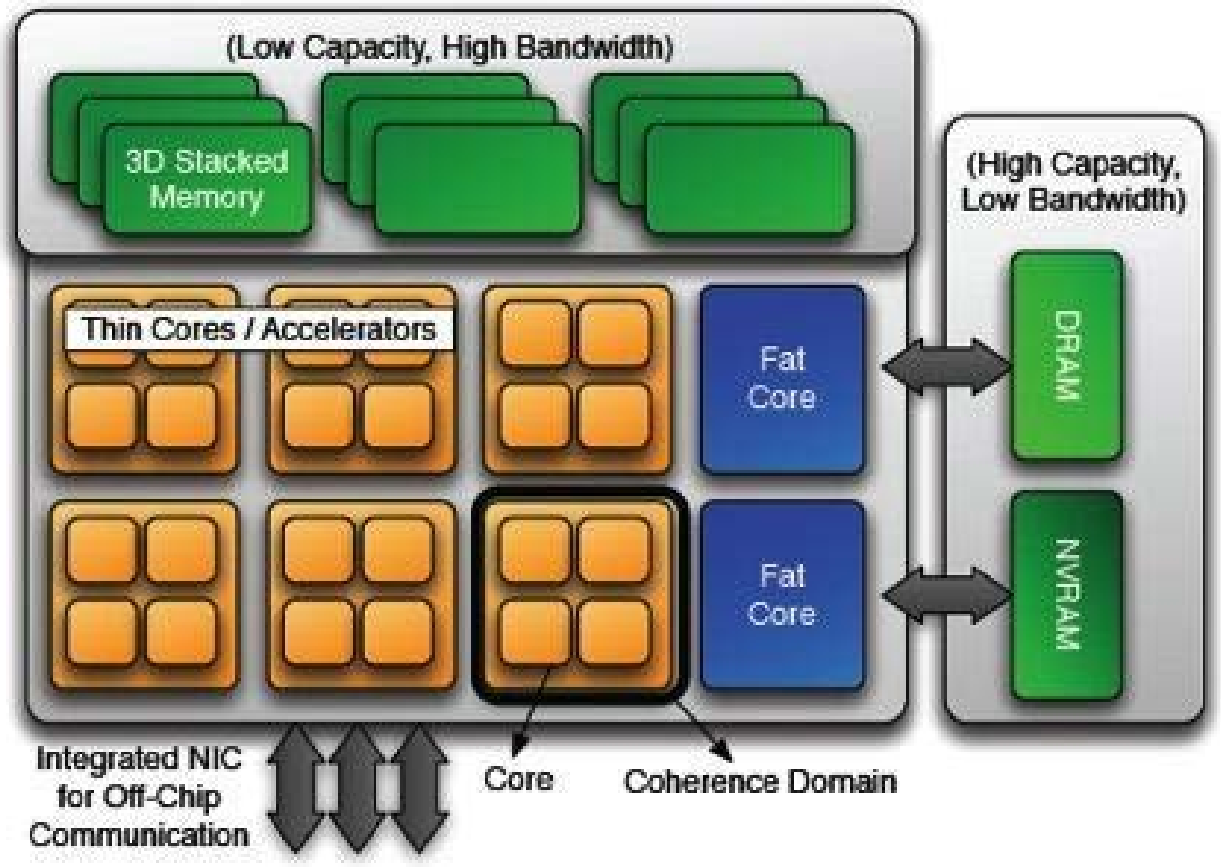


Figure 1: Abstract Node Architecture for Exascale

What Programming Model reflects the Execution Abstractions?

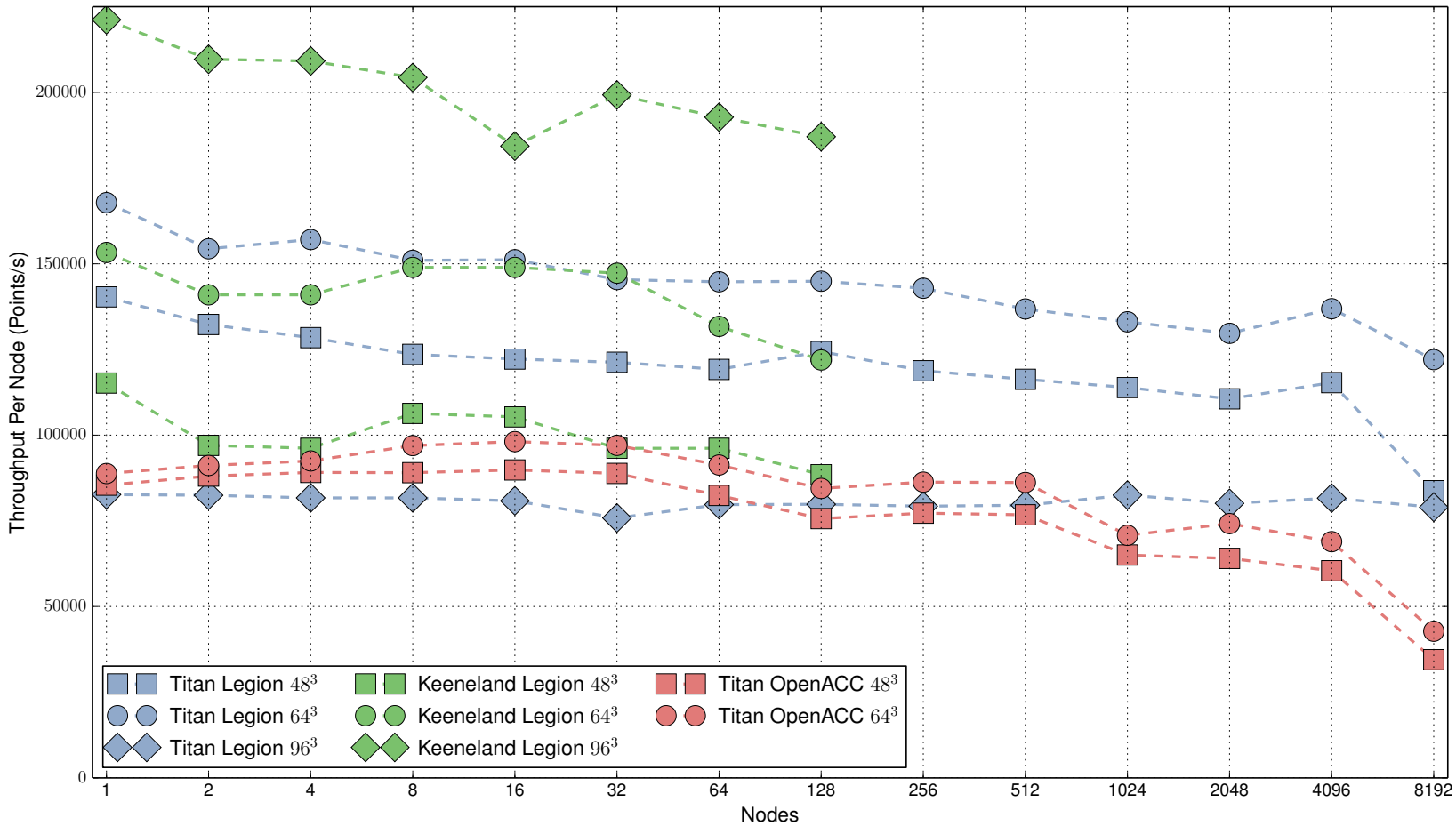
Legion

- A programming model for heterogeneous, distributed machines
- Heterogeneous
  - Mixed CPUs and GPUs
- Distributed
  - Large spread, and variability, of communication latencies
  - Caches, RAM, NUMA, network, …

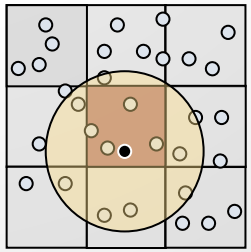
- A runtime system
  - Dynamic decision making
- Data-centric
  - Focus on giving the runtime info about the structure of the data
- Latency tolerant
  - Crucial on the target platforms

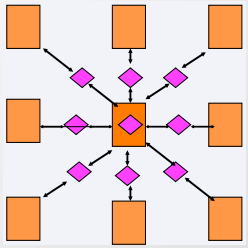
Legion S3D Heptane (Titan)

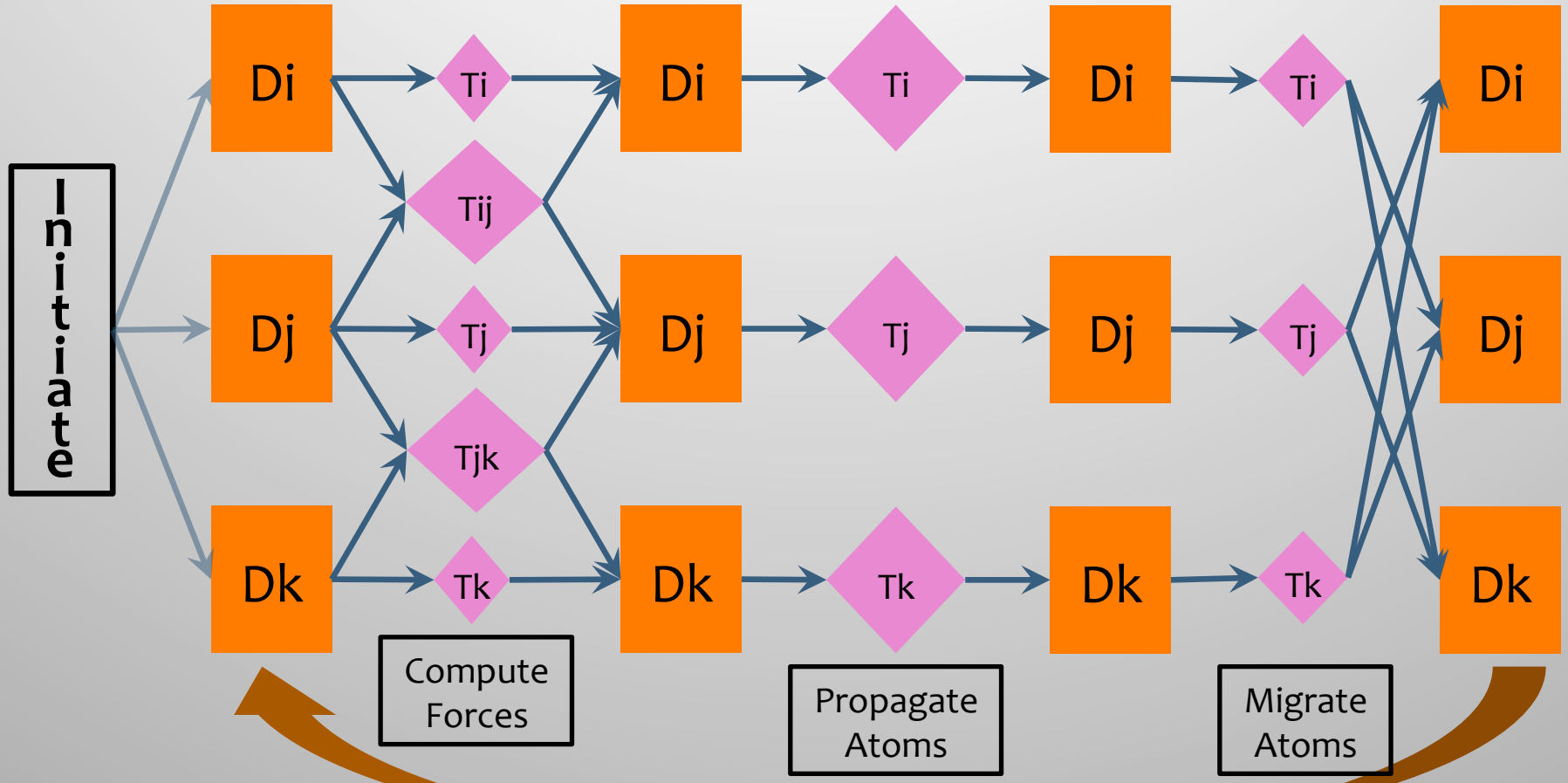
1.73X - 2.85X faster between 1024 and 8192 nodes



Task and data dependency graph for an MD simulation

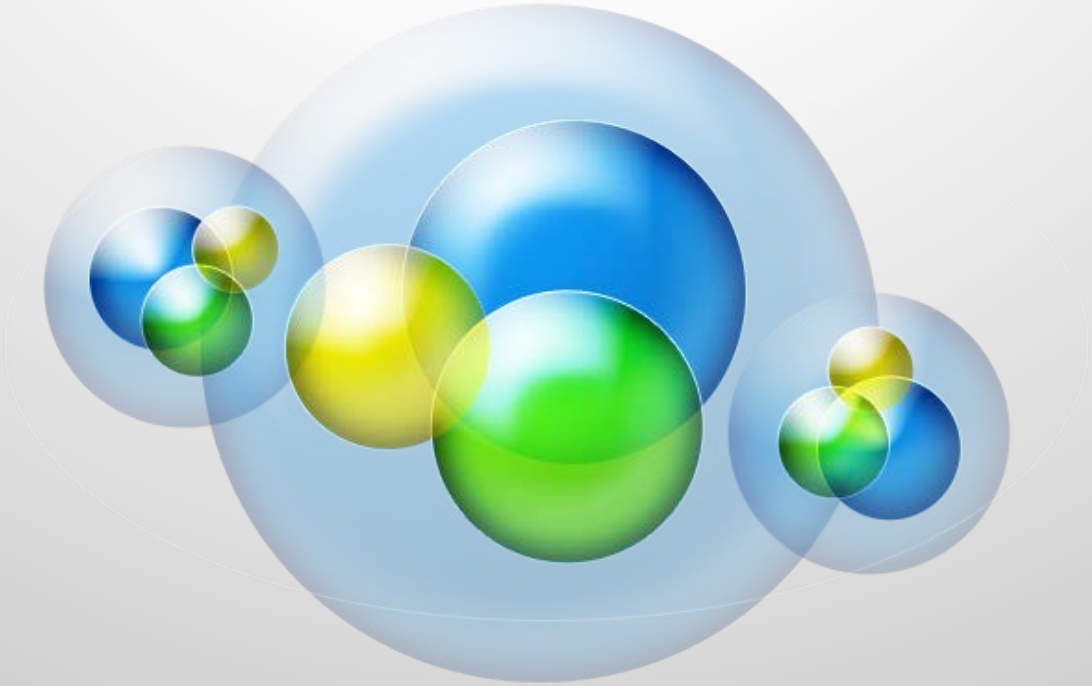






Loop until done

Breakout Summaries



Theme 3: Programming
Room A

Facilitators: Jim Ang and Dave Richards

There are more Programming Models than Codes – Mike Heroux

Theme 3: Programming

- Presenters:
  - Dave Richards
  - Pat McCormick
  - Rajeev Thakur
- Note Takers:
  - Tanzima Islam
  - Hemanth Kolla

What aspects of your application do you need to express to expose parallelism?

- Will bulk synchronous work?
- Do you need to express the algorithm in a fully asynchronous mode?
  - If so, what are your thoughts and/or experience with task-based programming models?
  - How well does your application map to such models?
- What size tasks work best for your application (coarse or fine)?
  - Are there aspects of your application that require different task sizes?
  - Do tasks need to be scheduled dynamically or will a static / semi-static scheduling model work (e.g. load balancing, in situ and/or in transit processing, multi-physics coupling, I/O operations, etc)?

What aspects of your application do you need to express to expose parallelism?

- Richards
  - Bulk sync is inevitable at some level (causality)
  - Expressing synchrony and data dependency is needed
  - Task based models reduce the number of inmates with shivs in the prison yard
- McCormick
  - Bulk sync will not work
  - Plenty of task parallelism
- Thakur
  - Bulk sync MPI still scales well
  - Applications seem to be converging on hybrid models
  - Lots of interest in tasking

What type of intranode programming model does your application need?

- Does OpenMP 4.0 meet your needs?
- Is there another model that would be better?
- Is expressing data locality important to your performance?
- What does the data layout for your application look like in the programming models you've explored?
- Is there a preferred data/memory paradigm (e.g. PGAS vs. distributed)?
- Can you achieve performance portability?

What type of intranode programming model does your application need?

- Richards
  - OpenMP doesn't have adequate support for non-data parallel workloads.
- McCormick
  - OpenMP/OpenACC work, but they aren't pretty
- Thakur
  - OpenMP has been successful
    - Some examples aren't actually openMP

What type of internode programming model does your application need?

- Will MPI work or do you need some other type of model, like PGAS?
  - Would a light-weight MPI work?
  - What MPI functionality do you actually need?
- What is your strategy for scheduling processor and memory locality and affinity?
- How about resilience?
- Can your application concentrate communication in a small number of latency optimized cores without creating a communications bottleneck?
- What features in the programming model do you need for dynamic load balancing?
- Do you have a good expression in one or more models?
  - If so, what do they look like?

What type of internode programming model does your application need?

- Richards
  - MPI probably adequate (but 50000 GPU threads sending messages is a challenge. So is strong scaling)
  - On-node heterogeneity a much bigger problem than node-to-node.
  - One-sided and fire/forget semantics potentially useful
  - Large standards result in poorly implemented features.
- McCormick
  - Two sided MPI won't work very well. Manual scheduling is difficult. Software engineering is too hard to schedule comm.
- Thakur
  - MPI is successful at largest possible scales today, and there is more than one way to do it.

What do you need in terms of languages, libraries and runtimes to support a programming model appropriate to your application?

- What level of support would be needed if you were to adopt a new programming model?
- Does a programming model implementation need to interoperate with legacy code?
- What would be the value proposition for your application to rewrite in a different language?
- What aspects of the hardware do you believe you need to control to obtain good performance?
- How well does a programming model need to reflect the specifics of the underlying hardware?
- What are the tradeoffs for your application between portability, expressiveness, usability, and performance?

What do you need in terms of languages, libraries and runtimes to support a programming model appropriate to your application?

- Richards
  - We want it all. Early adopters will need to be highly pain tolerant. Debugging is huge issue.
  - Unclear if a programming to any single abstract machine model can be performance portable
  - Separation of concerns features of Raja potentially bridge the lack of common abstract model
- McCormick
  - Programming model should hide hardware details as much as possible
  - Runtime should provide flexibility to map algorithms across the system
- Thakur

Programming Out-brief

Session B

Room 2

Jamal Mohd-Yusof
Cy Chan
Rusty Lusk
Kipton Barros
Rob Hoakstra

General Themes

- Cautious buy-in for task based programming
  - Need clear performance/productivity win
  - Concerns about overheads
  - Open standards needed
  - Some algorithms may not map to this model
    - Distinction between the algorithm and physics
  - Need to better understand data movement costs
  - Role for DSLs

Concerns

- Ownership of data layout
  - Adopting new programming model tied to abstraction of data layouts
  - Standards needed here also?
- Availability/ need for standards
  - Want some guarantee of persistence of the new model
- Task graph complexity

Disagreements

- Role of standards
- Role of new languages
  - Can implement tasks in existing language
- Need for tuning hooks in the model
- Interoperability with existing codes
- Incremental move to new model

Programming Models

Room 3
Beckman (ANL, CESAR)
Keasler (LLNL, ExMatEx)
Bergen (LANL, ASC)
Junghans (LANL, notes)
McPherson (LANL, facilitator)

Question 1 Tasks

- Will bulk synchronous work?
  - Codes are logically synchronous (timestep, reductions). Can do with tasking but with what overheads? More exploration needed. Some fear KoolAid being sold. Step back and take careful look at applicability and performance.
- Do you need to express the algorithm in a fully asynchronous mode? If so, what are your thoughts and/or experience with task-based programming models?
  - Don't NEED to but generally think there are benefits (variability, load balance, fault tolerance). With qualification and skepticism.
- What size tasks work best for your application (coarse or fine)?
  - Variable: a single cell (Pete, Andrew) to minutes (Jeff)
- Are there aspects of your application that require different task sizes?
  - Very fine grain for dynamically varying behavior of the node (Pete)
  - Open question. Perhaps because we can't do fine grain across the machine (Andrew?).
  - Natural variability of run-time of certain kernels (convergence) factor of 5
- Do tasks need to be scheduled dynamically or will a static / semi-static scheduling model work (e.g. load balancing, in situ and/or in transit processing, multi-physics coupling, I/O operations, etc)?
  - Static (fixed) may be required for some algorithms (SN)
  - Determinism may be necessary.

Question 2 Intra-

- What type of intra-node programming model does your application need?
  - MPI+X (Initial ExMatEx) write your own X
  - Legion+X (LANL NGC) write your own X
- Does OpenMP 4.0 meet your needs?
  - Likely not for fine scale tasking (Pete)
  - If we need fine scale tasking, we need OS and HW support
  - Maybe we need a new proxy to quantify and test OpenMP overhead.
- Is there another model that would be better?
  - Very fine grain on-node (cell-level) with OS support (Pete)
- Is expressing data locality important to your performance?
  - Yes, all the way down to the core
- Can you achieve performance portability?
  - Not if we need both fine and not-so-fine scale tasking (Jeff)
  - Differences between CPU and GPU requirements on task size

Question 3 Inter-

- What type of internode programming model does your application need?
  - MPI+X (LULESH early in ExMatEx)
  - Legion+X (Ben, ASC)
  - Charm++ (LULESH, tasking later in ExMatEx, mature)
  - LOCI (referenced by other tasking model providers?)
- Will MPI work or do you need some other type of model, like PGAS? Would a light-weight MPI work? What MPI functionality do you actually need?
  - Aside from legacy, MPI may be desirable (required) for efficient messaging underneath tasking programming models.
  - Lots of work and research in making this fast. Will need to do over for other technologies.
- What is your strategy for scheduling processor and memory locality and affinity? How about resilience?
  - Very fine grain tasking requires affinity and OS/HW support (Pete)
- What features in the programming model do you need for dynamic load balancing? Do you have a good expression in one or more models? If so, what do they look like?
  - Load imbalance will result from all the unknowns in the application (ExMatEx). Number of fine scale calls, convergence of fine scale calls, etc. Unpredictability of application flow.
  - Variations in very small task due to variability in OS (Pete).

Question 4 Model

- What do you need in terms of languages, libraries and runtimes to support a programming model appropriate to your application?

- What level of support would be needed if you were to adopt a new programming model? Does a programming model implementation need to interoperate with legacy code? What would be the value proposition for your application to rewrite in a different language?
  - Some questions (skepticism) about the viability of tasking for all algorithms and code (Neely, Ian). Especially in ASC. Questions about commitment reflect this.
- What aspects of the hardware do you believe you need to control to obtain good performance? Why? How well does a programming model need to reflect the specifics of the underlying hardware?
  - Strong feeling that standard HW support for DP atomics would enable performant reductions across platforms.
- What are the tradeoffs for your application between portability, expressiveness, usability, and performance?
  - Still seems to be focus on performance first (Jeff GPU and CPU support). Task-based overhead for SN (Ian, Rob). Not willing to sacrifice performance for the others?

Application Requirements for PM/E: a Productive, Portable, and Performance System/Framework

- Productivity: the **effectiveness** of productive effort (produce something) as measured in terms of rate of output per unit of input (man years)
- Portable: ability to be **transferred** from one machine or system to another (node multicore/gpu, disparate exascale systems, petascale in a rack or Beowolf cluster)
- Performance: an action, task or operation seen in terms of how **successfully** is was performed (not just time to solution, but time to a better solution, better physics, the computational task that optimizes workflow, new science, or a better bridge)