

# Application Requirements for Exascale PM/Es

*Rob Hoekstra, Lori Diachin, Marc Snir, Dave Richards,  
Wilf Pinfold, Pat McCormick, Josh ?, Robert Clay,  
Stephane Ethier, Rob Egan, John Shalf, Ray Bair, Lucy  
Nowell, Bill Harrod, Rich Brower, Phil Colella, Jackie  
Chen, Rich Hornung, Amarasinghe, Tina Macaluso*

# Topics

1. At a high level (not specific to any one application) catalog the application requirements at exascale that should be addressed by the combination of Programming Models and Environments. This will serve to conduct gap and coverage analysis of proposed PM/E(s), and as a basis for measuring progress.
2. Consider extending parallelism through non-conventional semantic constructs to enhance efficiency and scalability.
3. Describe the nature and importance of performance portability, productivity/usability, performance, robustness, interoperability with other PM/Es, and vendor support.
4. Discuss issues with trying to extend existing models (MPI+X), and any barriers to adopting new PM/Es,

# Topics V2

- Catalog application requirements
- Revolutionary semantic constructs
- Portability vs. Performance vs. Productivity vs. Robustness vs. Interoperability vs. Vendor Support
- Extending MPI+X vs. Revolutionary PM/Es

# Application Requirements

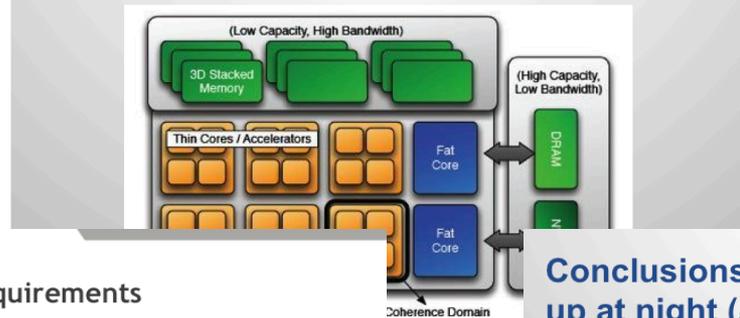


## Need better vector abstractions

### What is an ideal programming model?

- A homogeneous programming model that hides the complexity of multicore hosts, manycore accelerators, and inter-node messaging
  - Nice but... not going to happen, not realistic
- For future systems, non-disruptive programming models with lightweight runtime systems would be preferable
- Leave it to the application developer
  - Application developers should not know the inner workings of programming for performance, scalable codes
  - = **TRANSPARENT PROGRAMMING**
  - Should have the option to access any hardware
- Okay... maybe not entirely. I think we are promising for widely-used algorithms that they can do for Particle-in-Cell

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



- **SIMD code is hero work**
  - Relatively simple loop unrolling, but complex code changes
  - Bioinformatics can't expect many programmers at that level
- **Performance portability**
  - Different code (for a paper) necessary for each generation of processor
    - SW implementations for SSE2, SSE3, SSE4, GPU, Phi, FPGA

o bly for hand-tuning  
ns  
add features (like double-affine SW)  
nam applications

### Some climate modeling requirements

- A "reproducible" mode on exascale machines where repeated runs of the same executable on the same number of threads is guaranteed to produce bit-for-bit identical answers. For process-level development, not necessarily production.
- Programming model and environment that allows frequent and deep modification/replacement of highly specific/detailed algorithms.
- "support for teams" e.g. MPI communicators
- "mechanisms [that] permit the automated localized rewriting of software to use DSLs"
- "hardened, tested and supported production-quality software"

### Conclusions – what's keeping us up at night (at least that'll fit on one slide)



- **Multi-level memory models**
  - Coherent memory spaces ease transition, but consensus that application "hints" or explicit direction will be necessary.
- **OpenMP scalability and portability**
  - Even the best implementations have too much overhead to allow fine-grained parallelism
  - Will OpenMP 4.x be sufficient for targeting GPUs?
- **Asynchronous Task Models**
  - Research needed in how applications can easily express tasking
  - Growing consensus in its viability, but lots of unanswered questions
  - Will domain-scientists revolt? Yes, if we don't have good...
- **Tools**
  - Need for cross-platform, easy-to-use, scalable tools that focus on insight

chitecture for Exascale  
Office of Science

# Application Requirements

- Notion that we should start with PATTERNS
  - Dwarves
  - Motifs           # Intersection of these three?
  - Proxies           # Union of these three?
- This would better allow the community to distill down their priorities
- We have a long way to go in this regard...

# Application Requirements

- Why?
  - Edge cases dominate
  - The sum of the pieces is not equal to the whole
  - Are the proxies too ingrained with current HW constraints?
- What can we do?
  - Strive for a more mathematically rigorous definition of our requirements
  - Tighter interaction between domain scientists, computer scientists and vendors
  - Abstract machine models can map out PM requirements

# Application Requirements

- What can we agree on?
  - Task parallelism is our future...
    - Whether we like it or not...
    - But what does it look like?
  - Open standards are needed
    - But are we ready to define those standards?
  - Hierarchy of PM/Es
    - Task/data parallel low level constructs all the way up to DSLs at the high level
    - Application developers want to be able to drill down when it is necessary (They are control freaks just like the rest of us!)

# Revolutionary Semantic Constructs?

- More natural capture of parallelism?
  - Are task-based parallel constructs the panacea
    - Probably not... but they can help
- Can we win on all fronts?
  - Performance, Portability & Productivity???
  - Probably not, but they are not orthogonal
- One way or the other, propose something concrete and let the application developers beat on it.
  - Acknowledge this can be detrimental to fundamental research goals but it is a crucial bridge.

“We are so opposed to a new language that we would invent one inside of comments!”  
Comment on OpenMP

# Revolutionary Semantic Constructs?

- Low Level
  - DAG-based tasking?
    - Ubiquitous parallelism & Dependencies
  - Does it work for all algorithms? Should it?
  - Is the community ready to define a common abstraction/API?  
Probably not...
- High Level
  - Are DSLs our future?
  - Can application developers relinquish that much control over their code base?
  - Are general purpose DSLs a pipe dream?
  - Or are they becoming a more viable alternative to libraries?

Portability vs. Performance vs. Productivity vs.  
Robustness vs. Interoperability vs. Vendor Support

- We are always willing to give up some degree of performance for productivity & portability
  - How Much? It Depends....
- Other communities place a premium on “productivity”.
  - Can we learn from them?
  - Does this drive us away from the (1+epsilon) rewrite philosophy?
  - Can auto-tuning play a bigger role?

# Portability vs. Performance vs. Productivity vs. Robustness vs. Interoperability vs. Vendor Support

- Interoperability
  - Crucial when you are coupling disparate code bases (ala libraries, etc.)
  - Much less important when you own your code base
  - Partitioning of hardware resources to avoid contention is the starting point
- Vendor Support
  - Why we need to drive the open standards, but can they keep up?
  - Vendors will optimize the subset of features that win them the contract 😊
  - Take advantage of other leverage points?

# Extending MPI+X vs. Revolutionary PM/Es

- If we drive MPI+X models for now, the vendors and community will respond with solutions in this domain.
  - We can get pretty far down the road on that!
- Big question is when must we change
  - Optimization of resources problem
  - Most application developers will wait until the perceived payoff is big enough
  - Application developers have to eat; Pay them to make the leap of faith...
  - ...because volunteer early adopters are probably not enough

# Themes/Closing Thoughts

- Can we utilize patterns/abstractions more effectively?
  - Rigorous definition of constructs/patterns
- Range of PM/Es from low to high
- Try real apps in prototype PM/Es ASAP
- Co-design communication is crucial
- Revolutionary algorithms could still be the biggest winner

# Themes/Closing Remarks

- Engage standards communities
  - There will be failures but we can learn from them
- Pay people to try the new stuff
- What scares us silly!
  - Lack of mature implementations of PEs
  - Lack of tools
- Overriding issue: HUMAN CAPITAL
  - Domain Expert \* Computer Scientist = ~0

# Themes/Closing Remarks

- Urgency has escalated dramatically in the last few years
- If the codes are expected to run effectively on the pre-Exascale platforms they need to dramatically change “yesterday”.
- OR will the folks who play “wait and see” be the winners?