

PM/E Workshop Summary

Recap: High Level Workshop

Objectives

- Propose, discuss, and determine the required characteristics or properties of future programming models
- Review the application requirements for ECI PM/E(s)
- Identify the key capabilities and elements of future programming environments
- Determine the research questions that need to be addressed
- Propose the methods for evaluating PM/E(s) research results

Recap: Workshop Drivers

- Establish a conceptual framework to discuss and frame the relevant technical issues, their interrelationships, and their possible alternatives
- Understand differentiation between programming models and programming environments
- Identify and describe key concepts among alternative designs, avoiding premature standardization and maintaining separation of concerns such as functionality and interface
- Anticipate enabling technology opportunities and challenges
- Explore road map for incremental evolutionary approach and justify changes as required

Recap: Mission Drivers

- Discuss types of workflows that are supported intra-compute and inter-compute
- How to get useful application drivers that in many cases may require complete rewrites with new algorithms

Recap: Programming Models

- Discuss abstractions and their semantics; elevate key abstractions to first class consideration as a principal common goal of the ECI program plan and projects:
 - a. Parallelism
 - b. Global name space(s)
 - c. Locality/distribution/memory hierarchy
 - d. Synchronization
 - e. Introspection
 - f. Resilience
 - g. Communication
 - h. Time/space/energy management
 - i. Roles of runtime and compiler support

Recap: Programming Environment

- Understand invariants of program characterization retained for performance portability across systems of different type, scales, and generations
- Explore performance models and productivity models to motivate approaches, challenges and opportunities
- Investigate debugging strategies for correctness and performance
- Understand the needs for storage abstractions, from private local intranode storage to shared parallel file systems in future programming models and environments
- Establish needs and means for application interoperability to create compound jobs

Recap: Software Stack

- Determine roles of programming models and programming environments within the system stack; their requirements and interconnections
- Characterize nature and coupling of runtime system with programming model as well as responsibilities to programming environment workflow

What are the high level takeaways we can highlight from the last 2.5 days?

- Application teams are getting anxious – need to begin moving research into a hardened state for early adopters
- We need application teams and PM/E teams to partner on ideas (co-design)
- Defining metrics is difficult, but ultimately necessary – need to separate quality of implementation from quality of model
- High/Medium/Low PM abstractions provide a valuable framework for focusing discussions (Domain/Algorithm/Tuning Specialist)
- There is a great need for tools – both improving what we have, and creating entirely new genres
- Open question and skepticism about the productivity benefits of new programming models. Large upside potential, with commensurate learning curve
- Still working on understanding what aspects applications need first. Load balancing across nodes? Tasks models? Hierarchical memory abstractions?
- Complexity of programming is increasing, not decreasing – worrisome.
- Abstractions to storage and persistence still poorly understood
- Resilience and power concerns are increasingly being discussed in the context of pushing them down to the lower levels of the stack