Programming Models

Patrick McCormick (LANL), Pavan Balaji (ANL), Richard Barrett (SNL),
Bronis de Supinski (LLNL), Evi Dube (LLNL), Carter Edwards (SNL),
Paul Henning (LANL), Curtis Janssen (SNL),
Krishnan Manojkumar (PNNL), Steve Langer (LLNL), Allen McPherson (LANL),
Dan Quinlan (LLNL), John Shalf (LBNL)

Working Group Description

Primary Goals
- *Portability*
- *Productivity*
- *Performance*

- Scope (i.e. major sw stack elements responsible for)
  - Above system software, below (most) developer/user software
    - Includes runtimes, libraries, languages and compilers
- Key dependencies on other working groups
  - Hardware: Both vendors and simulators.
    - Need development/understanding of HW fault model(s)
  - System software: vendors centric features, support for memory/ thread affinity and locality, fault coordination, power management
    - At what level(s) do we expose these features?
  - User's/apps: (SAL, VDA, UQ, etc.)
    - Anyone developing code on the system.
    - Should we present layers for system software?
  - Tools: Coordinate presentation of model & details to users
  - I/O, Networking and Storage: Requests for better abstractions for I/O from apps

Exascale Challenges

- Fundamental challenges:
  - Leverage existing code investments
    - Keeping current codes functioning on emerging architectures
    - OpenMPI and OpenMP need to be supported
    - *Can we do this in a progressive **yet** revolutionary way? Is there a migration path to allow for interoperability with new models and requirements for data layout and movement?*
  - Existing programming models might be insufficient/problematic for exascale
    - Need better ways to express parallelism/concurrency, deal with complex memory hierarchies (data movement), resiliency, etc.
    - Need higher-level abstractions
    - Continuing with current low-level, sequential coding styles a risk
- Portability… Can we address apps/developers desire to address coding from "laptops to exaflops"?
  - And support this across the *swim lanes*?

Bottom line:
We can't afford to put all our eggs in a single basket...

Leveraging Existing Code Investments

- Proposed technologies
  - MPI, OpenMP (need support and new development)
  - OpenCL & other industry supported approaches (need evaluation)
- Initial next steps
  - Explorations/modification to existing models (FY12)
  - Evaluate interoperability of new and existing techniques
  - Identify strategies and address issues
- Timeline (Phase I, II or III; Evolutionary or Revolutionary)
  - Evolutionary and revolutionary
  - Phases I and II: Overlapping evaluation and deployment
  - Phase III: Maintain and deploy
- Required Partnership
  - Standards committees, vendors, industry, ASCR, academia
- Risks
  - Existing technologies fail at exascale
  - Industry's commitment in suitable programming models
  - Delays in adoption/implementation of critical needs by standards
  - Late start, late codes
  - Hardware uncertainties

New Programming Models (Higher Levels of Abstraction)

- Proposed technologies
  - Test beds and simulators for exploring architectures
  - Languages, compilers, runtimes, libraries that support higher level abstractions
- Initial next steps
  - Begin the development of mini-/compact-/skeleton apps and the supporting development tools (exploring multiple approaches to abstraction)
    - Evaluation on both test beds and simulators
    - Need to explore interoperability with existing codes
    - Leverage industry and academic activities
    - We **MUST** get our hands dirty sooner rather than later
- Timeline (Phase I, II or III; Evolutionary or Revolutionary)
  - Phases I, II, III: develop, evaluate and deploy (need very active lifecycle)
  - Evolutionary and revolutionary
- Required Partnership
  - Academia, industry, ASCR, DARPA
- Risks
  - Late start, late codes (amplified by development of new models)
  - Lack of adoption
  - Potential of significant software development costs
  - Hardware risks

Recommended Co-Design Strategy

- Critical steps/activities
  - **Must** work with SAL/Apps/SSW/etc… to define/develop programming models
    - Successful abstractions, future efforts will require fewer but more diverse teams
- Working with industry/vendors
  - Must have understanding of where architectures are heading
  - Must understand role of industry programming models and tools
  - Provide input to hardware architects on our needs
- Role of skeleton-/mini-/compact-apps
  - Foundation for understanding what works (and what doesn't) in timely fashion
  - Defining effective/efficient interfaces between working groups (domains)
- Concerns/suggestions
  - Vendor collaborations, intellectual property
  - Late start, late codes (amplified by development of new models)

Big Picture Issues

- Coordination
  - **Critical** to cross-cut, collaborate and coordinate between DOE offices, industry, and academia
    - Must be able to prioritize based on programmatic needs/goals?
    - We can't afford to negatively impact scientific production
- Test beds and simulators
  - We have **A LOT** to do -- getting our hands dirty ASAP is critical for success
- Investments in industry and academia
  - Need "path forward" opportunities and academic alliances
    - Must be done in close collaboration with national labs
- V&V and UQ and even just confidence in the results as we move forward
- *Is the evolutionary path feasible?*
- *Is the revolution path affordable?*