Working Group on Exascale Tools

Martin Schulz, LLNL (lead)

Atinuke Arowojolu, DOE
Sean Blanchard, LANL
James Brandt, SNLs
Scott Futral, LLNL
John Mellor-Crummey, Rice University
Barton Miller, University of Wisconsin
David Montoya, LANL
Mahesh Rajan, SNLs
Kenneth Roche, PNNL
Allan Snavely, UCSD/SDSC
Mary Zosel, LLNL

Scope of the Tools WG

- Major Software Stack Elements the Group is Responsible for:
  - Tools for application development (debugging, correctness, performance)
    - Wide spectrum: memory, power, locality, resilience, …
    - Static analysis tools for code evaluation
  - Tools for SSW to evaluate the exascale stack itself
    - SSW, I/O, Network, File systems, Scheduler, …
    - Need to get away from ad-hoc tools, need whole system solution
  - Shared infrastructure for measurement, data gathering and presentation
    - Online analysis, data aggregation, shared across the system stack
    - Post-mortem, online, in site and batch tools
  - HW and SW APIs / information exchange with other WGs
    - APIs that we want to wrap and monitor
    - Introspection APIs (HW and SW)
    - Guidance for other system components (targeted, information isolation)
    - APIs exposing semantic information from the users to tools
  - Resources for testing/validation of the system (incl. tools)
- Not in scope: compilers (vendors!), resiliency techniques, runtimes

Key Dependencies with Other WGs (1)

- HWA
  - Measures of resource consumption: power, network, memory bandwidth, issue slots, …
  - Raw measures of inefficiency (exposed latency, lack of memory parallelism)
  - Identification of resources (e.g., for heterogeneous nodes, GPU versions)
  - Hardware instrumentation to emulate 2018 machine costs with 2015 machine
- SSW
  - Right APIs incl. RAS and debugger interfaces (incl. testing)
  - Expose all hardware features, don't hide anything
    incl. counters, power, resiliency, faults, HW topology
  - Timely reporting and precise attribution of asynchronous events
  - Interfaces to scheduler, scheduling of tool resources
  - SSW runtime monitoring, runtime must expose right abstractions
- I/O & I/O Networks
  - For tools: interfaces to capture and measure performance (MPI_T like)
  - Capture network and storage topologies
  - Tool needs: load balancing and striping, detect link contention
  - modeling vs. measurement to find bottlenecks
  - Tracing data movements and separate between system and user traffic
  - Provide building blocks to enable specialized I/O tools (generic tracers/profilers)
  - More discussion needed: storage approaches and formats for tools (SQL DBs?)

Key Dependencies with Other WGs (2)

- Visualization and Data Analysis (VDA)
  - Common needs, requirements on SSW (online analysis and data storage)
  - Exploit application knowledge available in Viz tools (data layout, …)
  - Provide building blocks to enable specialized VDA tools (e.g., in situ analysis)
  - Need VDA techniques for performance data analytics and visualization (outlier detection, equivalence groups, compression/data reduction, feature detection, …)
- Programming Models (PMs)
  - Compiler and runtime must provide information for tools to map costs back to PM abstractions
  - Translators/PMs/Compilers must expose abstractions to tools
  - PM runtime monitoring, runtimes must expose right abstractions
- Applications, Solvers, Algorithms, Libraries (Apps, SAL)
  - We are treating libraries as apps (exception: potential API interception)
  - List of expectations on tools – information that Apps/SAL people want to see
    - Data centric profiling – away from flop centric tools to memory centric tools
    - Memory locality and consumption
    - Data structures and access patterns
    - Opportunity analysis (concurrency, offload to accelerators, compiler feedback)
    - Delivering information on power and resiliency
  - Mini-Apps for testing of tools (for performance, complexity, SSW, …)
  - Application internal monitoring interfaces to capture semantic and performance data

Exascale Challenges for/around Tools

- Challenges in providing new capabilities
  - Scalability of measurement, analysis, and presentation
    - Incl. new metrics: memory, power, …
  - Turning information into insight
    - Despite flood and complexity of data from billions of threads
  - Dealing with new programming methodologies
    - Heterogeneous systems/architectures (HW and SW)
    - Coupled systems and applications
  - "What if" tools for Co-Design
- Challenges for tool implementations
  - Quick design of prototype tools for new scenarios
    - Agile development to keep up with PMs
    - Need them early, enable specialized tools in this and other areas
  - Getting right interfaces with the right abstractions
    - To SSW, HWA, Apps, Libraries, Runtimes, Compilers, …
  - Resiliency for tools and tool infrastructures

Path Forward: Memory Tools

- **New generation of tools to explore memory related metrics**
  - Data locality, cache pressure, data movements, efficiency of memory usage
  - Clear attribution to code regions and data structures
  - Away from flop centric, towards data centric tools
- Initial next steps
  - Explore new counters and measurement techniques
  - New data-centric visualization techniques
- Timeline: Now, Phase I and II (Evo -> Rev)
- Required Funding: $ for phase 1, $$ for phase 2
- Required Partnership
  - Apps&SAL (guidance, input), PM (abstractions), HWA (counters)
  - HW Vendors (HW support), OASCR/Academia
- Risks
  - Insufficient support from HW may lead to high-overhead tools, but we have little other chance (apps need this)

Path Forward: Tool Building Blocks

- **Modular and Separable Tool Components**
  - Reusable and interchangeable measurement and analysis systems
  - Enables retargetable & specialized tools (across archs, WGs, apps)
  - Avoid stove pipes and inefficient ad-hoc tools
- Initial next steps
  - Establish and disseminate APIs and backbone infrastructures
  - Transform existing tools as seed for component libraries
- Timeline: Now, Phase I, II, III (from research to maintenance)
- Required Funding: $-$$ per phase (influences speed of development)
- Required Partnership
  - Users: Apps, SALs, SSW, IO, VDA, Support: SSW
  - OASCR/Academia, International community
- Risks
  - Must create a community effort around component structure to guarantee interoperability (mitigation: focus on partnerships)

Path Forward: Application-Tool Interfaces

- **Interfaces to exchange performance and semantic information**
  - How to deliver semantic information from apps to tools?
  - How to expose application specific hooks to tools?
  - Integration with modular measurement system and use of components
- Initial next steps
  - Discussion among WGs on potential APIs
  - Experiment with a series of specialized tools on current systems
- Timeline: Now, Phase I (revisit in later phases)
- Required Funding: $
- Required Partnership
  - Apps, SALs
- Risks
  - Low risk activity, but potential for high payoff

Path Forward: Mini/Skeleton-Apps

- **Aid in the definition of the collection of Mini-Apps**
  - Ensure similar performance characteristics
  - Support for validation
  - Coverage of mini-apps beyond performance (startup, system issues, …)
- Initial next steps
  - Discussions with Apps/SAL teams on mini-apps extraction
  - Creation of validation infrastructure
- Timeline: Phase I (but driven by Apps/SALs)
- Required Funding: 0.5*$
- Required Partnership
  - Apps, SALs (as the creators of the mini-apps)
  - SSW (who will need the mini-apps for testing in the same way as tools)
- Risks
  - Low risk activity, but potential for high payoff

Path Forward: Power Tools

- **Inclusion of power metrics into application oriented tools**
  - Measure power and attribute it to application structures
  - Expose costs that are currently hidden
  - Next step: expand to other metrics (resiliency, health, …)
- Initial next steps
  - Expose and use new interfaces for power measurements
  - Integration into existing tools as new measurement components
- Timeline: Phase I, II, III (initial steps now)
- Required Funding: $ (phase 1), $$ (phase 2), $$$ (phase 3)
- Required Partnership
  - Apps, SALs (guidance), SSW/HWA (interfaces)
  - OASCR/Academia (existing approaches, HW access)
- Risks
  - Next generation measurement capabilities unclear
  - Testbed access, especially at labs (security concerns)

Path Forward: Correctness Tools

- **Verification of correct usage of PM abstractions**
  - Catch bugs/problems early – at scale
  - Example: MPI verification tools
  - Expand to other programming models
- Initial next steps
  - Continue development of MPI related tools with focus on scale
  - Study semantics and suitability for this approach for other interfaces
- Timeline: Phase I,II,III (initial steps now)
- Required Funding: $$ (phase I), $$$ (phase 2), $ (phase 3)
  - Early development (I), many interfaces (II), maintenance (III)
- Required Partnership
  - Apps, SALs (guidance), PMs (API specifications – existing and new ones)
  - Academia, International community
- Risks
  - Existing models: low risk, New models: suitability/feasibility unclear

Path Forward: Support for New Models

- **Investigation of support for new programming models**
  - Find code segments for accelerator offload
  - Support for coupled applications
  - Gather information and predictions on system utilization
- Initial next steps
  - Establish interfaces to coupled applications and accelerators
  - Performance models for heterogeneous models
- Timeline: Phase I ++ (unclear since models are unknown)
- Required Funding: $-$$ (depends on the number of models)
- Required Partnership
  - Apps, SALs & PMs
  - OASCR/Academia
- Risks
  - Effectiveness & usefulness of provided information
  - Longevity of new models

Recommended Co-Design Strategy

- Critical steps/activities
  - Memory tools
  - Establish tool building blocks (in collaboration with other WGs)
  - Establish interfaces
- Working with vendors
  - Interfaces and documentation of all HW and SW features
  - Low-level compilers should be vendor task
    - Requirements to expose function, unwind and attribution information
  - Dialogue on hardware and runtime support for measurements
    - Incl. power consumption, faults, resiliency data, …
- Role of skeleton/compact apps
  - To study performance and scaling expectations
  - Validation suite for SSW and tools
  - Understand tool requirements for emerging execution models
  - Establish requirements for communicating semantic annotations to tools
- Concerns/suggestions
  - Coordination between WGs to establish a single picture
  - How to encourage/incentivize WGs to work together?
  - How to merge people's independent research agendas?
  - Testbeds, integration tests, …

Big Picture Issues

- Coordination – must be a continuous, agile process
  - Among tool developers
    - Coordinate on common interfaces and components
    - Maintenance models
  - With Apps/SAL teams
    - Ensure their needs are met
    - Establish interfaces
  - With SSW, I/O, VDA
    - Share infrastructures
    - Avoid ad-hoc tools
  - With vendors
    - Need interfaces and documentation
    - Co-Design interactions on getting the right system hooks
- Test beds
  - Essential, need sufficient access for tools research
  - Work around security concerns (e.g., for power sensors)