ASC Panel on Runtime System Topics: Optimizing Runtimes, Tools and Interfaces

Martin Schulz

Lawrence Livermore National Laboratory

2015 ECI Runtime Systems Workshop ◆ March, 2015

LLNL-PRES-668449

http://scalability.llnl.gov/

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Bridging the Gap Between Systems and Applications

- **Rising complexity on both sides**
  - Architectures are becoming more heterogeneous and diverse
    - Large scale, complex networks, accelerators, deep caches, NUMA, …
    - Multiple, quite different target architectures at the same time
  - Applications are complex, multi-physics simulation frameworks
    - Wide range of independently developed modules
    - Trend towards higher levels of abstraction
  - Mapping applications to systems while getting efficiency is hard
- **Tools and runtime research to support this mapping task beyond the actual OS and runtime systems**
  - Auto-tuning and adapting system components
  - Support libraries and techniques, e.g., for resilience and load balancing
  - New tool chains to help with deployment, debugging & optimization

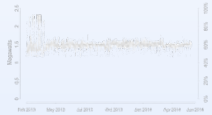
Wide Spectrum of Challenges Beyond Only Runtimes

Performance Tools & Optimization

Debugging & Correctness Tools

Pro-Active & Defensive Resilience

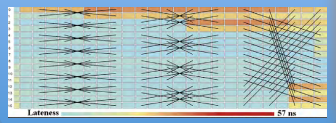
Power-Constrained Systems



Supportive Infrastructure

New Generation of Tools and Tool Infrastructures

Performance Tools & Optimization



Debugging & Correctness Tools

Pro-Active & Defensive Resilience

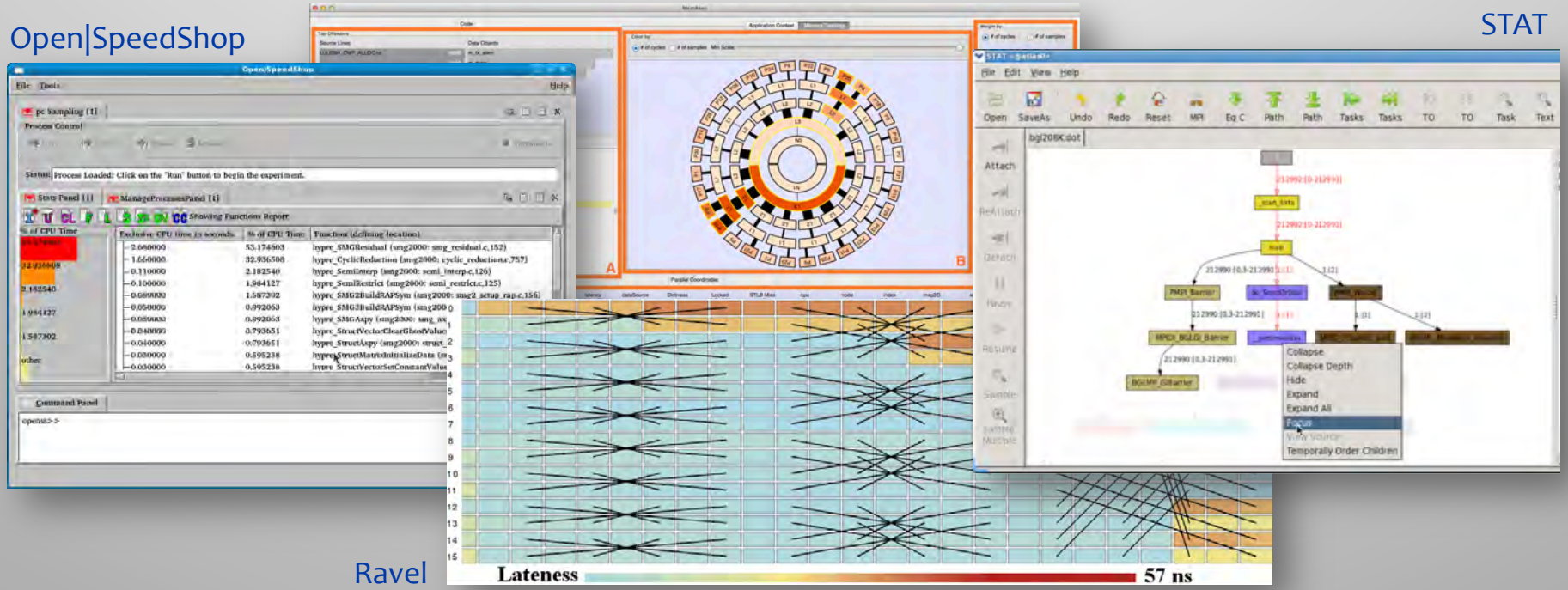
Power-Constrained Systems

Supportive Infrastructure

MemAxes

Open|SpeedShop

STAT



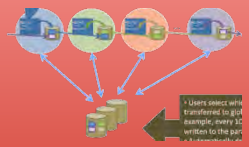
Ravel

Resilience Approaches

- **Resilience abstractions in programming models**
  - Our current focus: MPI
- **Efficient checkpointing**
  - Using existing programming systems

Do We Need/Want a Resilient MPI?

Pro-Active & Defensive Resilience

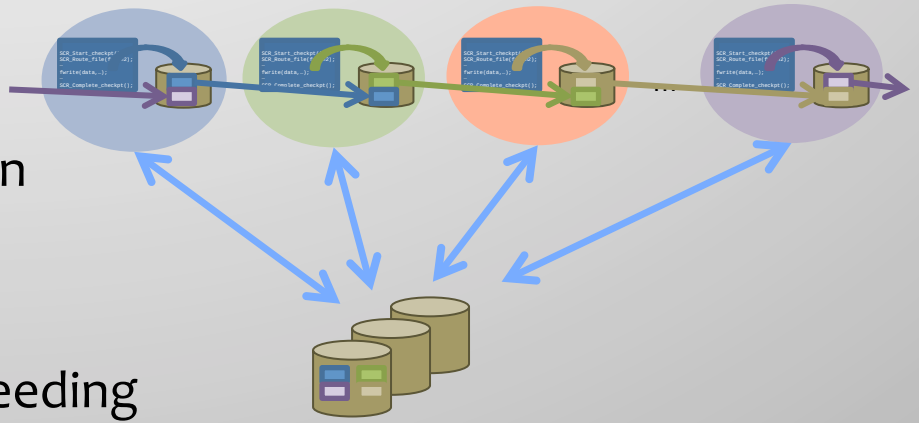


- **Inclusion into programming systems, like MPI, requires *careful* design**
  - Current activities in the MPI forum: ULFM
    - Covers process faults
    - Proposal for shrinking recovery
    - Code has to re-organize its data decomposition after a fault
  - Hard pill to swallow for many existing codes
    - Is a re-decomposition on the fly even possible?
    - Increased code complexity and significant rewriting needs
    - Data consistency requires global restart anyway
- **Is this really a benefit over existing checkpoint/restart?**
  - Are we doing more harm than good?
  - How does this change with new programming models?
  - Who is really the target and does this change the story?

Is global checkpoint/restart really that bad?

SCR: Scalable Checkpoint Restart

- **Efficient checkpointing approach**
  - Decouple applications from checkpoint implementation
    - Application only specifies what has to be checkpointed
  - How and to where is decided by the library
    - Optimal checkpoint intervals
    - Multi-level storage options
  - Multiple avenues for optimization
    - Checkpoint location
    - Data compression
    - Asynchronous checkpoint bleeding



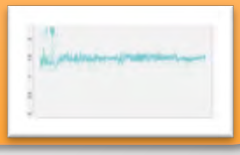
- **Research Challenges**
  - Efficient use of new hierarchical storage options
  - Impact on new, more asynchronous programming models
  - Tradeoffs between performance, power and reliability

Treating Power as a Constraint

- **Overprovisioned systems are in our future**
  - Provision the system on average not peak power
  - Leads to more hardware than we can power
  - Consequence: power caps
- **Challenges: How can extract the most performance under a power constraint?**

Power is (just another) a Performance Issue

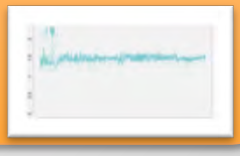
Power-Constrained Systems



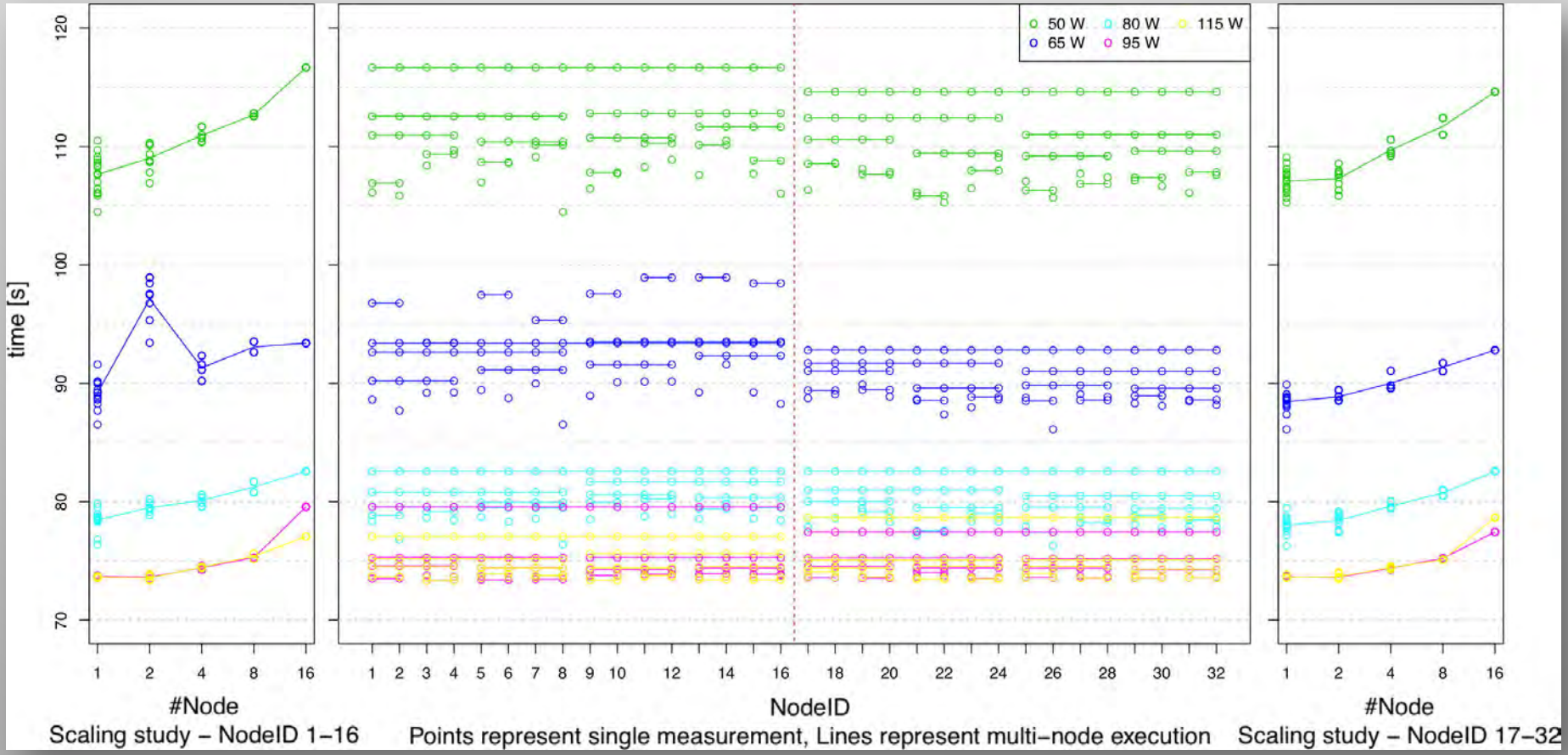
- **Impact on applications**
  - Exposes power inhomogeneity
    - Currently 10%
    - Predictions: rising

Power is (just another) a Performance Issue

Power-Constrained Systems

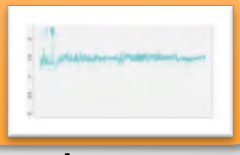


- **Impact on applications**
  - Exposes power inhomogeneity

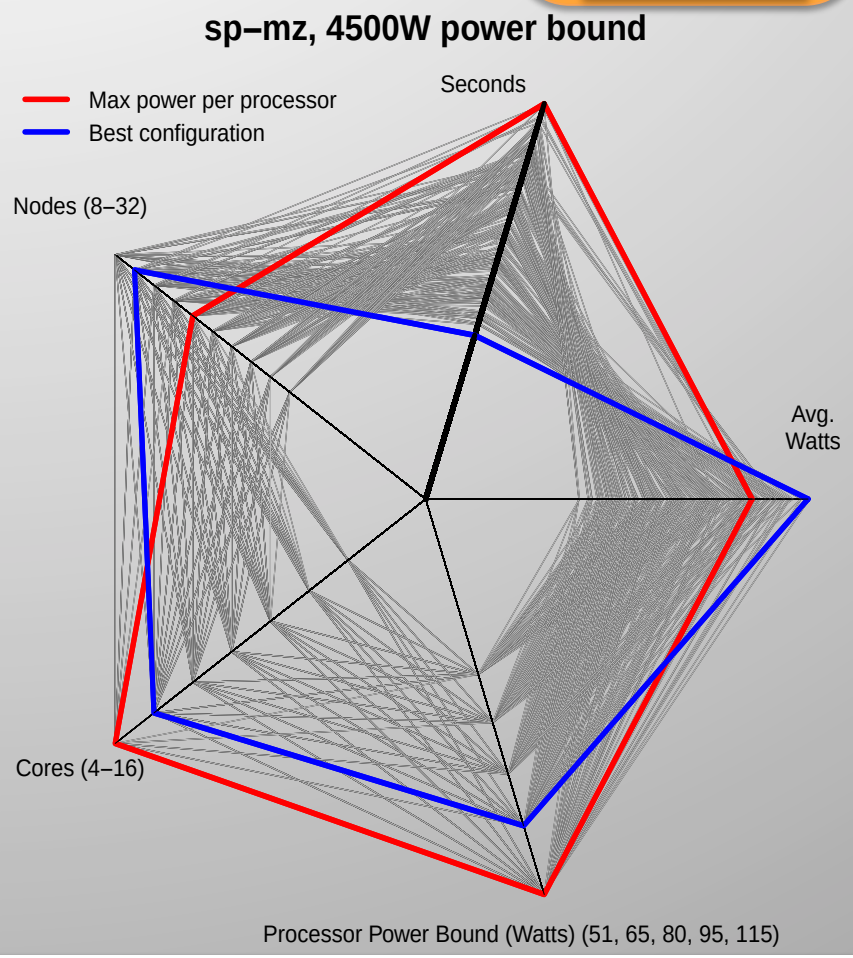


Power is (just another) a Performance Issue

Power-Constrained Systems

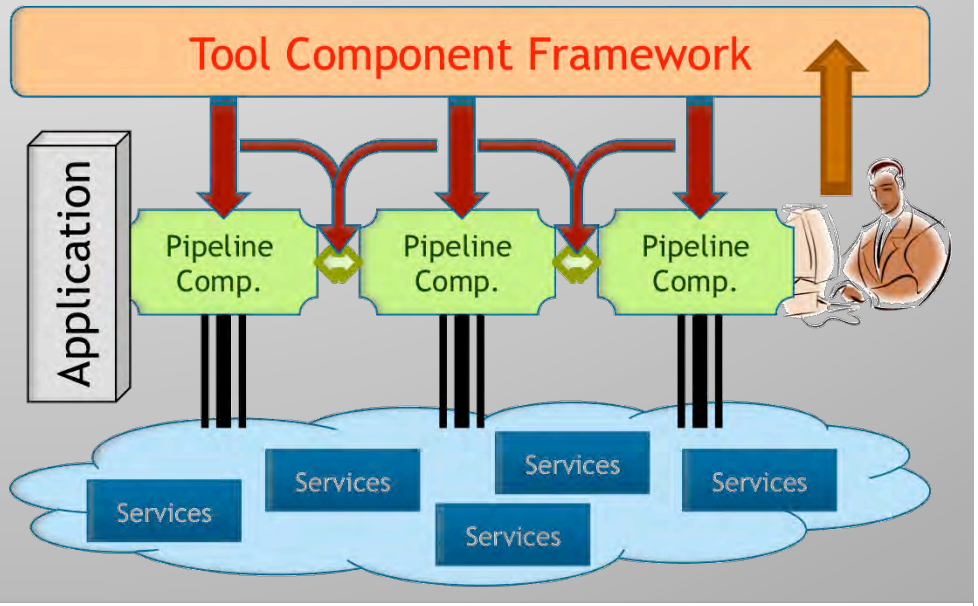


- **Impact on applications**
  - Exposes power inhomogeneity
    - Currently 10%
    - Predictions: rising
  - Performance reduction is application dependent
    - Based on capping level
    - Based on memory patterns
  - Need accurate modeling
- **Research Challenge: Adaptive, power optimizing runtime systems**
  - Balance/Exploit inhomogeneity
  - Direct resources to critical path
  - Find optimal configuration(s)
  - Introspection into placement



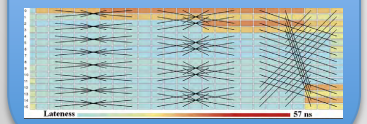
Crosscutting Issue: Scalable Infrastructure

- **Many tools with similar requirements & workflows**
  - Performance analysis
  - Debugger
  - Continuous regression
  - Checkpoint management
  - In-situ visualization
  - Data analytics
- **Should be OS/R service**



Missing Component

Performance Tools & Optimization

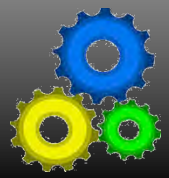


Debugging & Correctness Tools

Pro-Active & Defensive Resilience

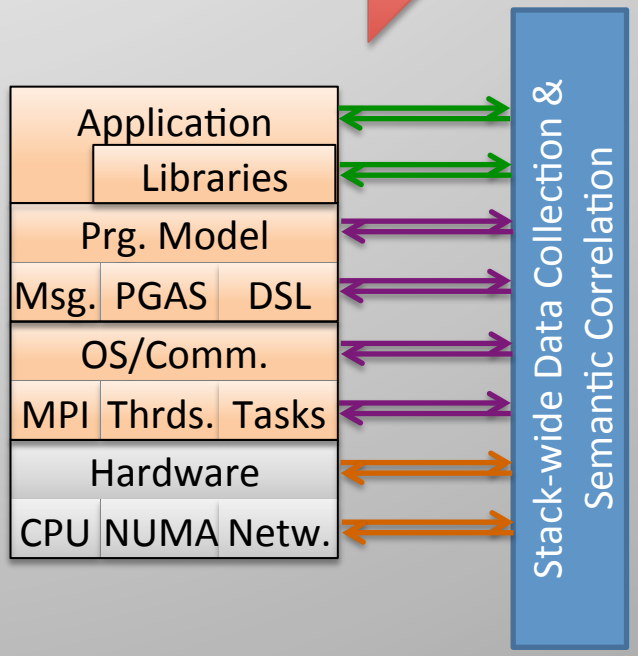
Power-Constrained Systems

Scalable Infrastructure



Missing Component: Compatible Interfaces

- **Introspection across the SW stack**
  - Application
  - Programming models
  - OS/R
  - Hardware
- **Doing this with a NxN approach is not feasible**



Interfaces: Not Sexy, But Essential

- **Tools (in a broad sense) need interfaces to extract information**
  - Best example: the MPI profiling interface
    - Allows interception of all MPI calls
    - Basis for wide and highly successful tool infrastructure
  - Need to apply lessons to other programming models/systems
- **Recent successes:**
  - MPI_T is part of MPI 3.0
  - OMPT has been sanctioned by the OpenMP ARB
  - Active discussion on OMPD
- **Need to extend to new programming models and runtimes**
  - Interfaces to allow introspection
  - Preferably same/similar interfaces
  - Can't afford a N/tool to M/runtime complexity

This cannot be an afterthought - Co-Design with users!

Wide Spectrum of Challenges Beyond Only Runtimes

- New generation of performance and debugging tools
- Multi-pronged resilience approach
- Performance optimizations under power constraints
- Shared infrastructure supported provided by OS/R
- Introspection interfaces co-designed among all communities

Need: close coordination within/with/beyond the runtime stack