Performance Modeling Under a Power Bound: A Tour of the Near Future

Martin Schulz and Barry Rountree
Lawrence Livermore National Laboratory

Performance Modeling: Methods & Applications – Workshop @ ISC 2015 ◆ July 16th, 2015

LLNL-PRES-674802

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.

The Need For Performance Modeling Under a Power Bound: A Tour of the Near Future

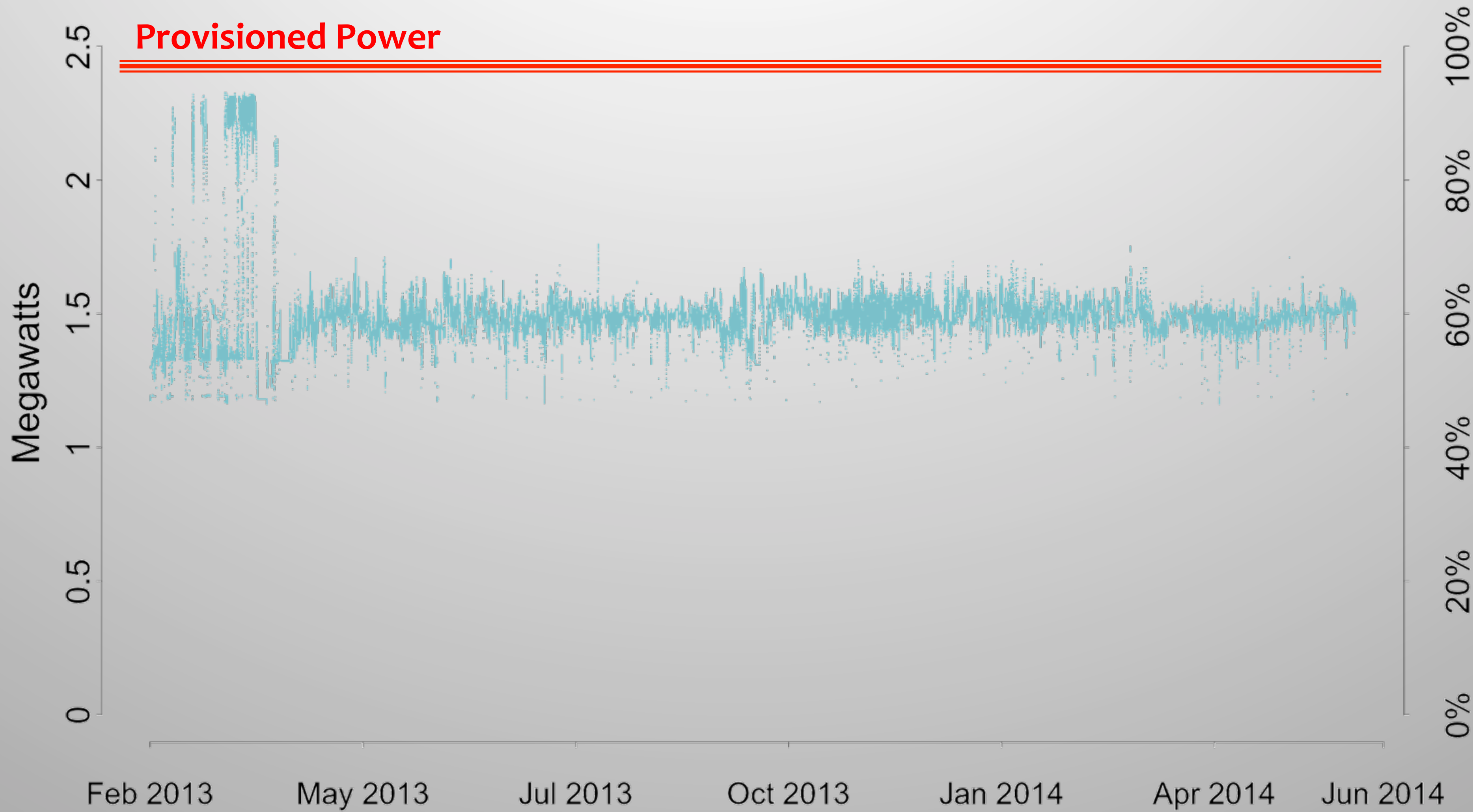
Martin Schulz and Barry Rountree
Lawrence Livermore National Laboratory

Performance Modeling: Methods & Applications – Workshop @ ISC 2015 ◆ July 16th, 2015

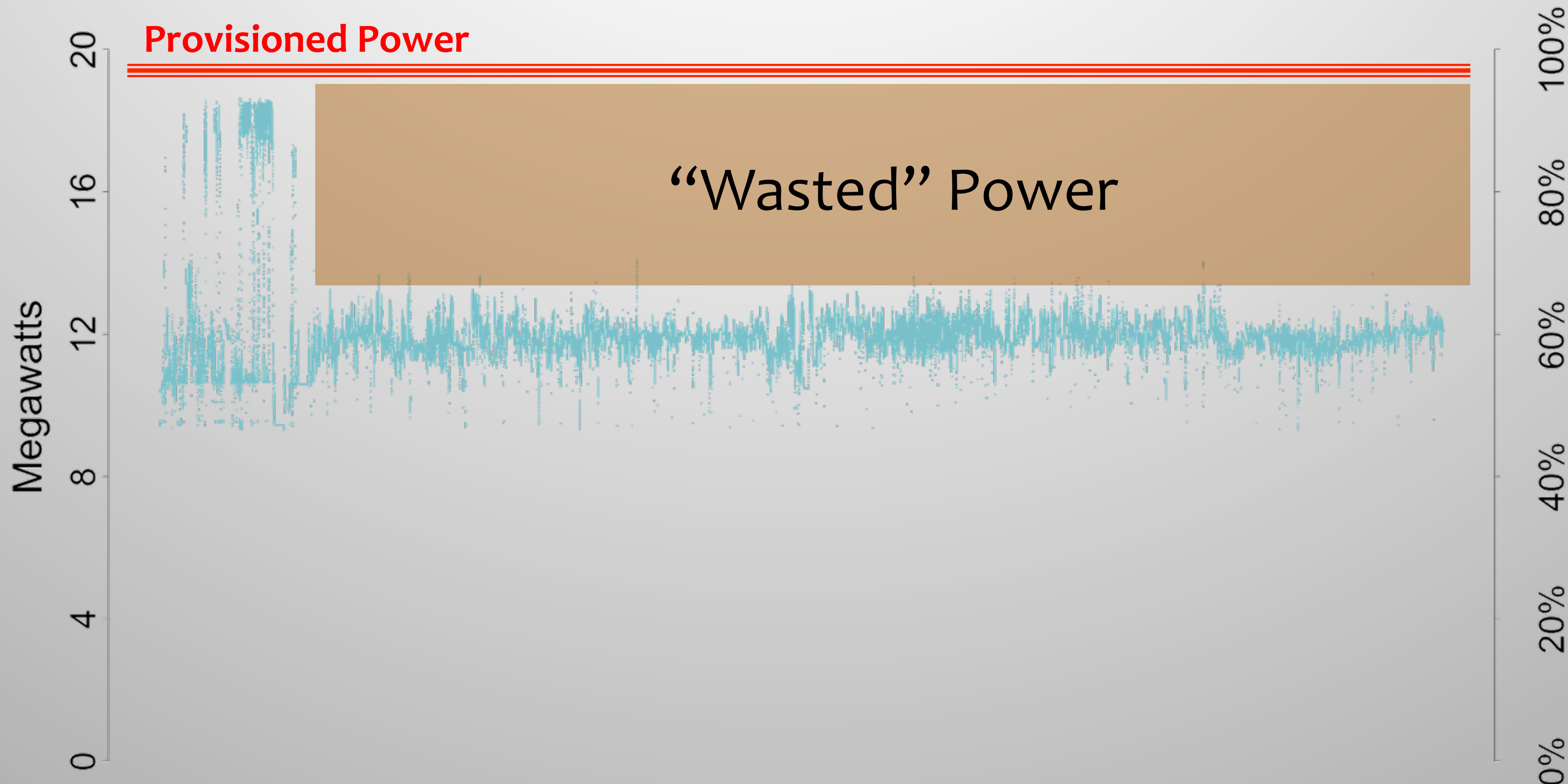
http://scalability.llnl.gov/

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

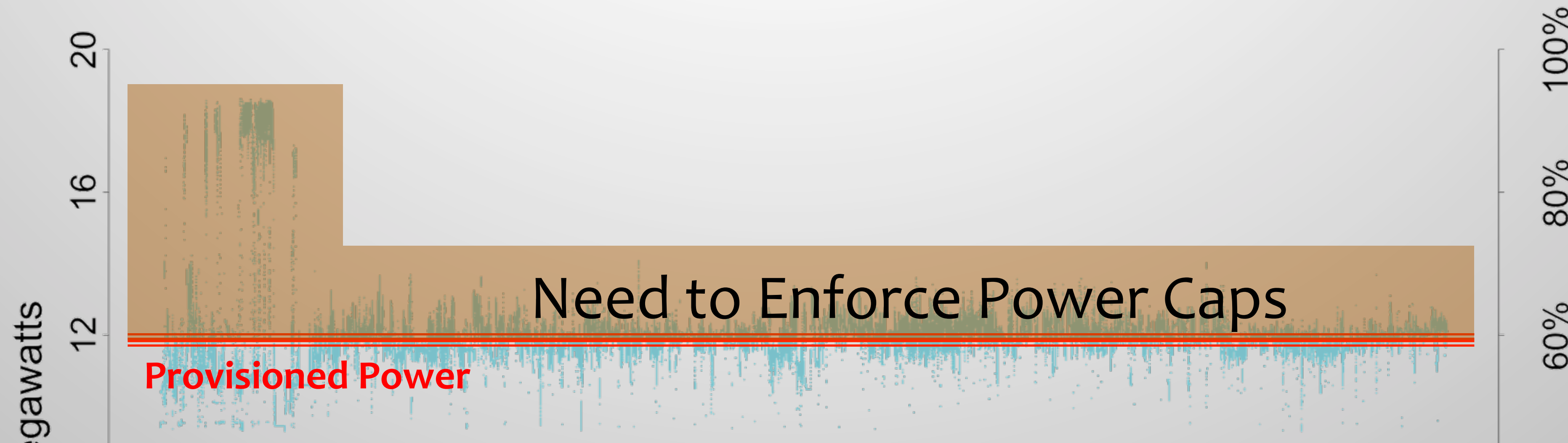
Power Consumption of Current Systems (e.g., BG/Q)



Goal: Exascale @ 20MW



Goal: Exascale @ 20MW



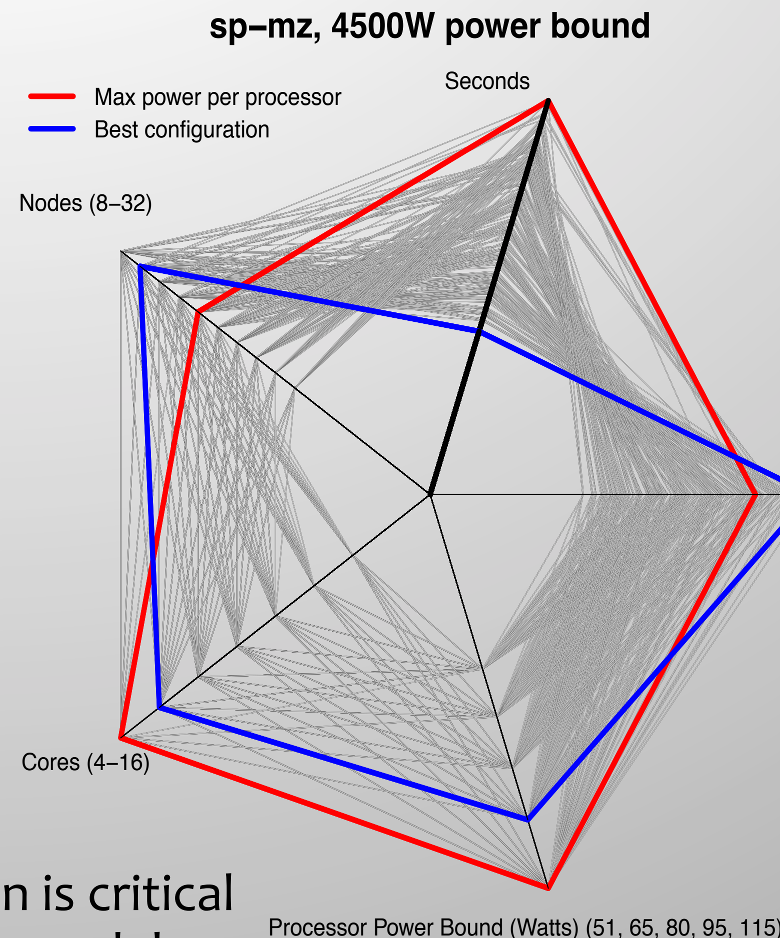
- **Power as a constraint, not an optimization goal**
- **Overprovisioned systems**
  - More hardware that can be powered at once
  - Need mechanisms to control and cap power
  - Need runtime systems to maximize power usage within cap
  - Need Cross-job scheduling control

Rethink utilization in terms of power, not nodes

- **Overprovisioning has large impact on applications**
  - Need to execute under strict node level power bounds
    - — Different performance behavior and tradeoffs
  - Steer power where it is needed to make most progress
  - Avoid wasted power, i.e., maximize power utilization

Importance of Configurations

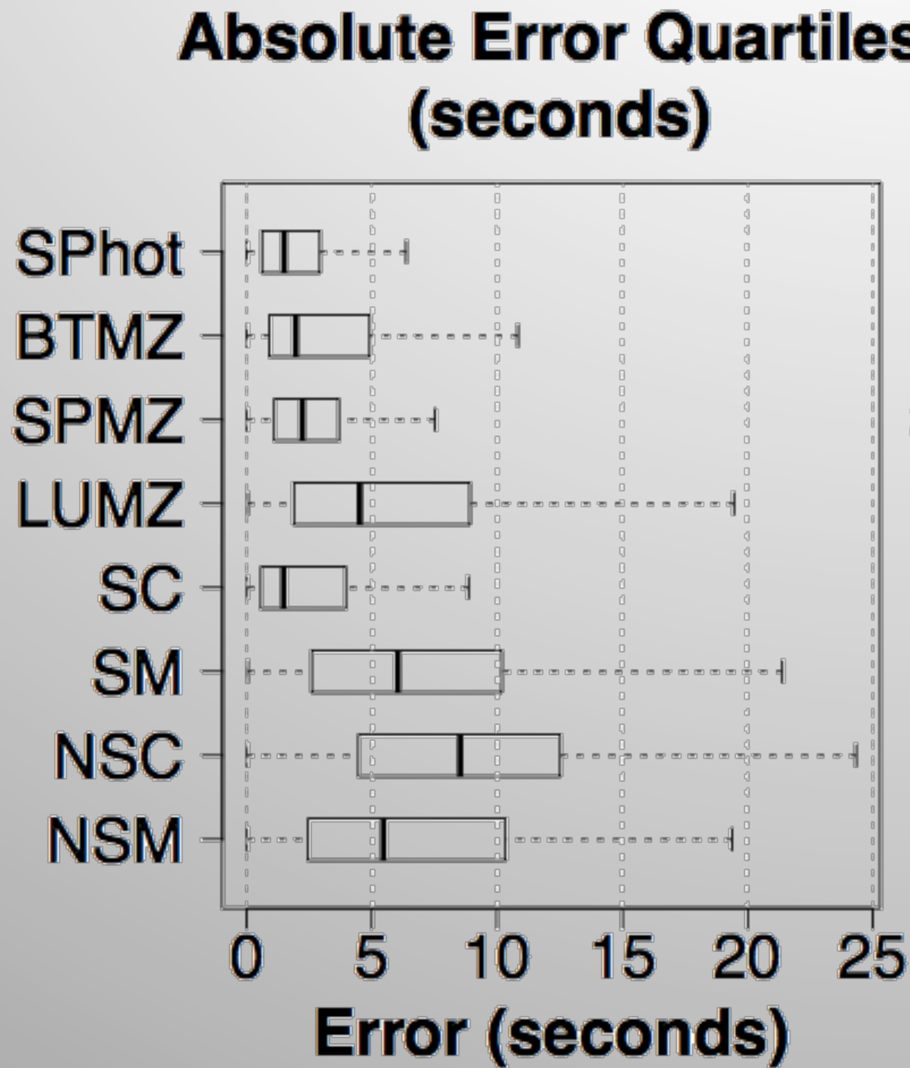
- **Experiment on 32 nodes on LLNL's TLCC system with a global power bound**
- **Naïve configuration in red**
  - Running all cores
  - This limits number of nodes
- **Best configuration in blue**
  - Moderate per node power bound
  - Reduced number of cores
- **Difference: > 2x**
- **Consequences:**
  - Determining the right configuration is critical
  - Intuition is insufficient -> need new models

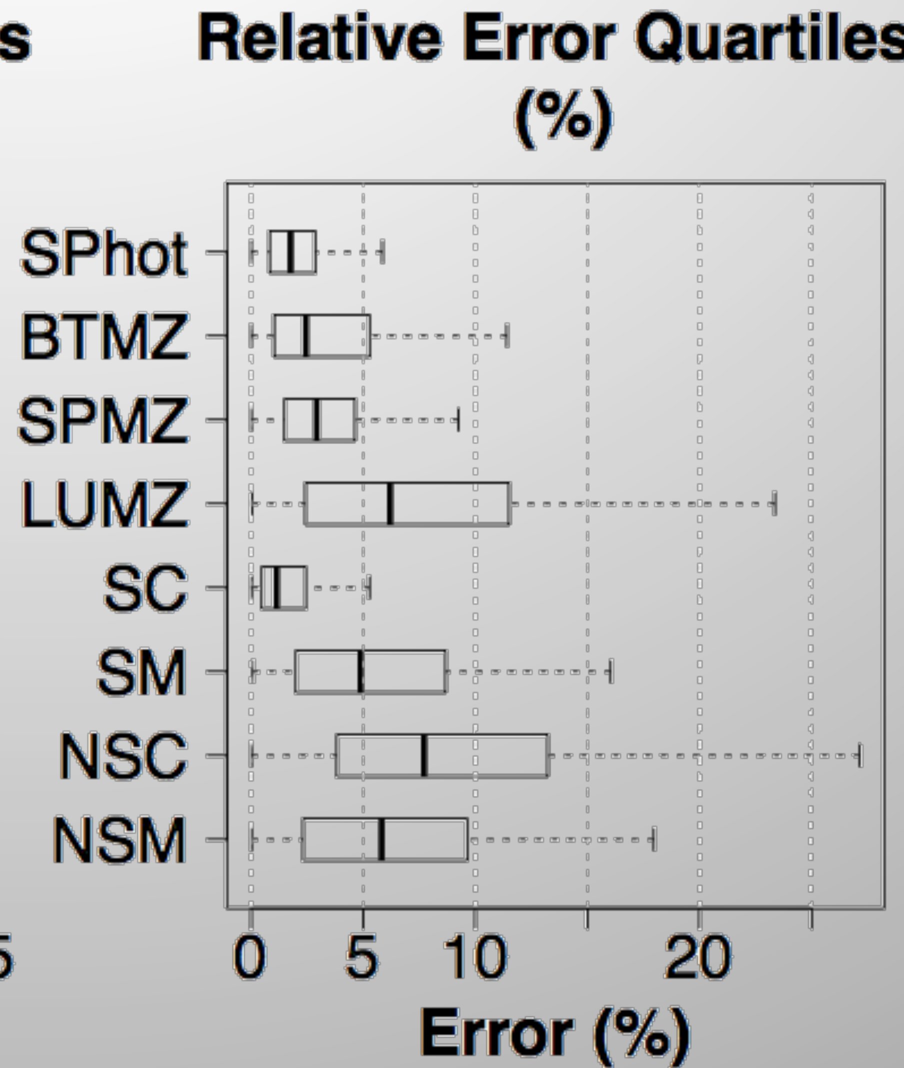


Rethink utilization in terms of power, not nodes

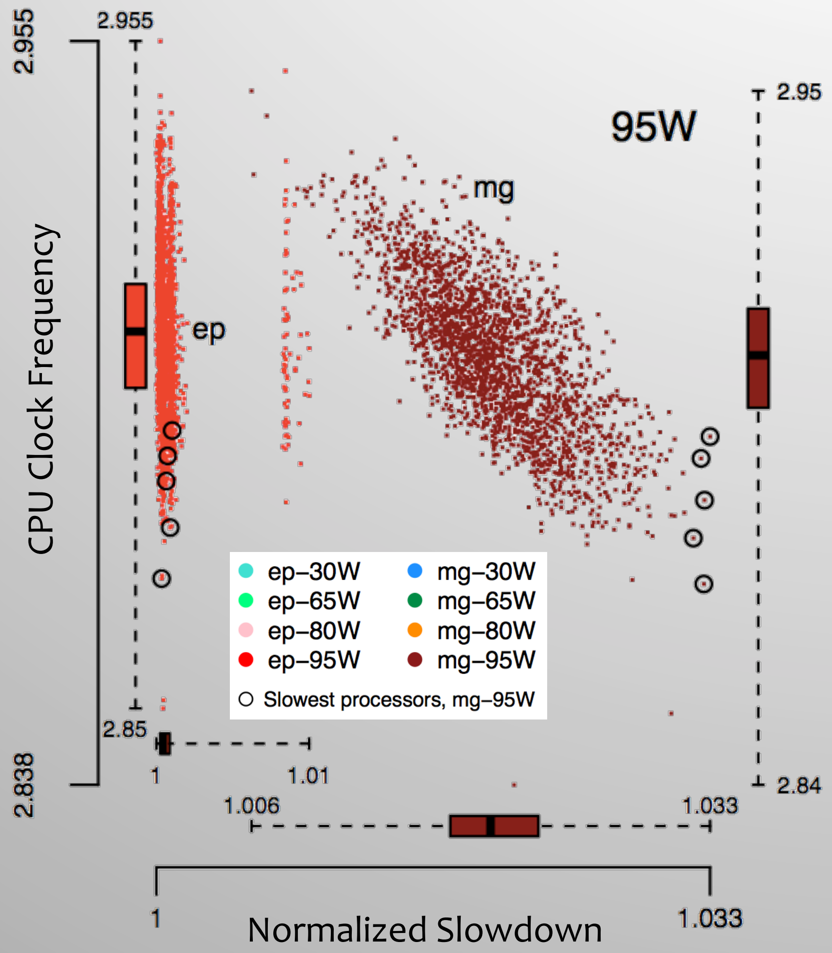
- **Overprovisioning has large impact on applications**
  - Need to execute under strict node level power bounds
    - — Different performance behavior and tradeoffs
  - Steer power where it is needed to make most progress
  - Avoid wasted power, i.e., maximize power utilization
- **Need a new power/performance model**
  - Different for each power bound
  - Depends on workload characteristics
  - So far, (some) success with adhoc models
    - — Sampling of configuration space (~3000 points)
    - — Linear regression to construct model (using 10%)

Modeling Results (8-64 nodes, 51W-115W, SandyBridge)



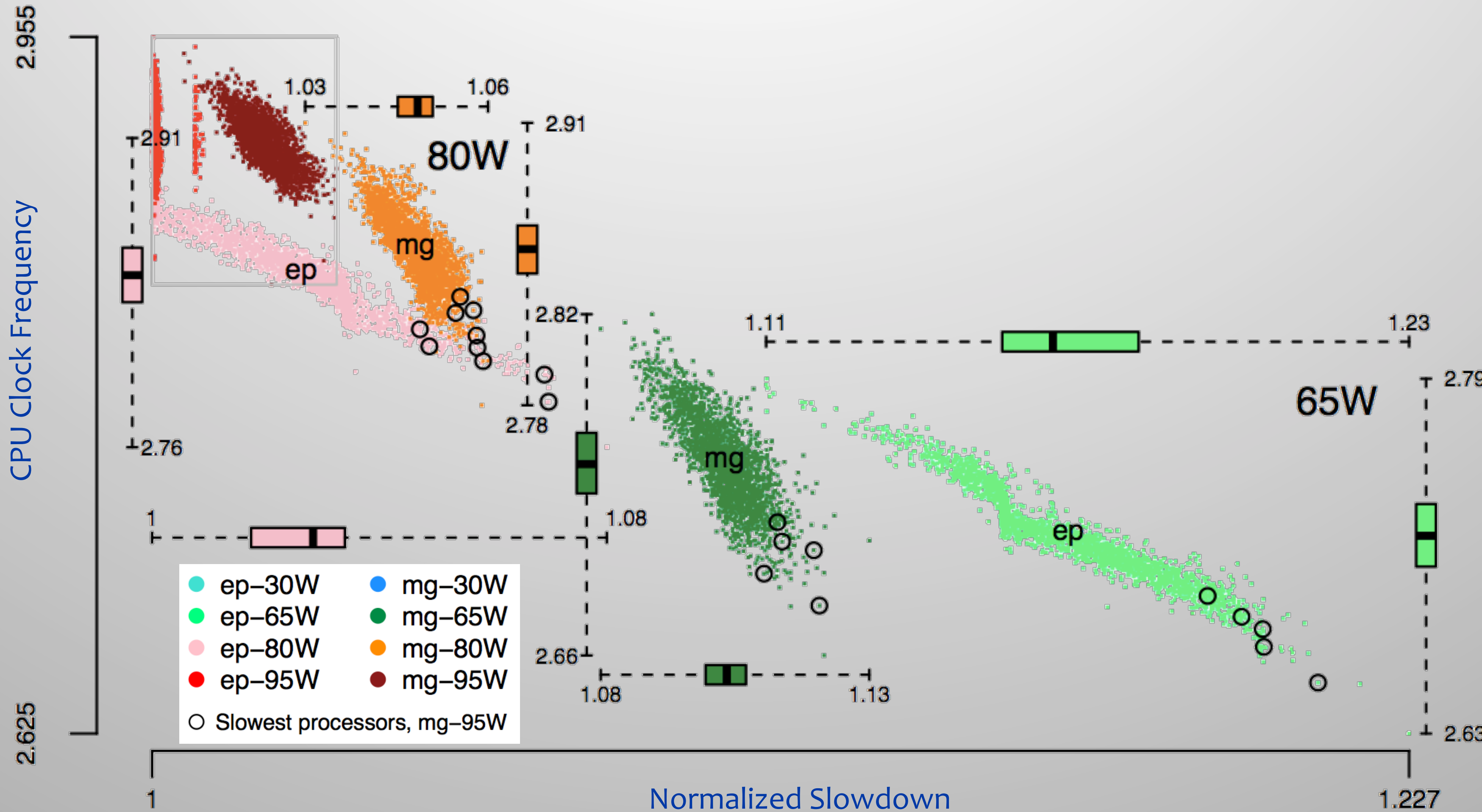


Impact of Processor Manufacturing Variability

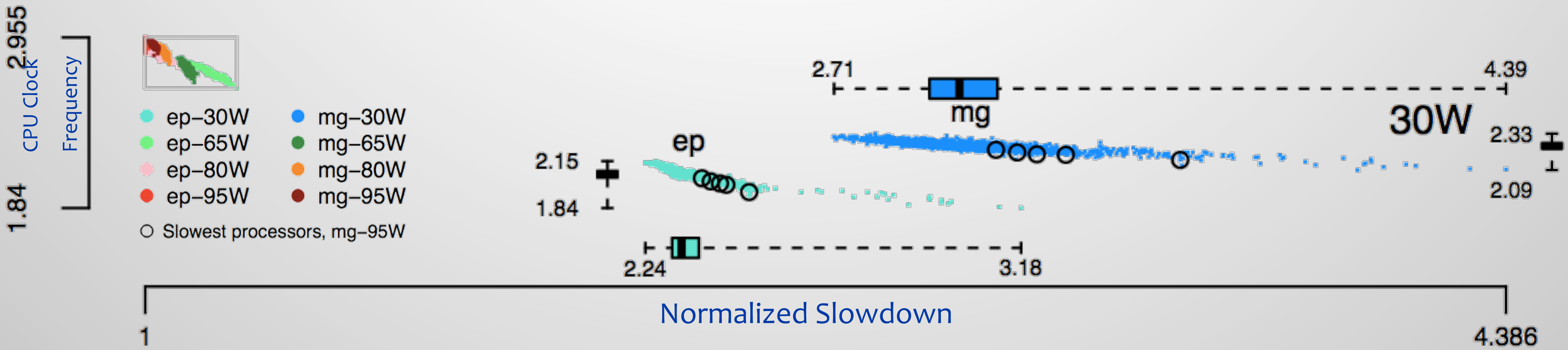


- **Census across 2386 processors**
  - mg (multigrid)
    - Runs at 105W
  - ep (embarrassingly parallel)
    - Runs at 90W
- **Chart showing one point for each processor in the system**
  - Performance normalized to fastest unbounded run
  - X-Axis: Slowdown
  - Y-Axis: CPU clock
  - Slowest processors circled

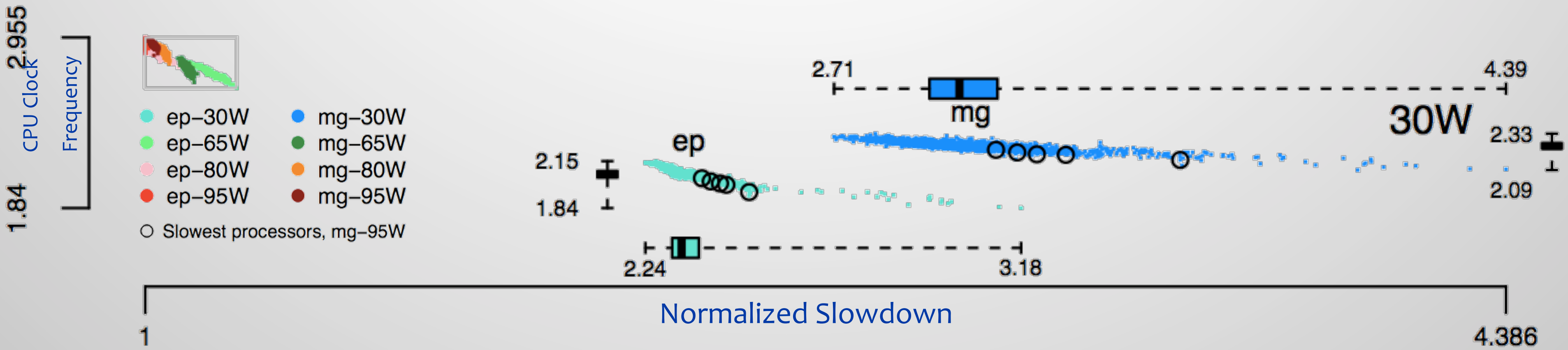
Large Scale Power Capping Experiments: 80W / 65W



Large Scale Power Capping Experiments: < 51W



Large Scale Power Capping Experiments: Conclusions



- **Power capping makes systems heterogeneous**
  - Need more flexible task scheduling and ability to absorb slack
  - Needs to be taken into account during load balancing
- **Slowdown under power caps application specific**
  - Can't use a single knob "processor/silicon efficiency"
  - Depends on application's instruction mix and memory intensity
- **Runtime systems needed for more efficient scheduling**

Conductor: A Runtime System for Overprovisioning

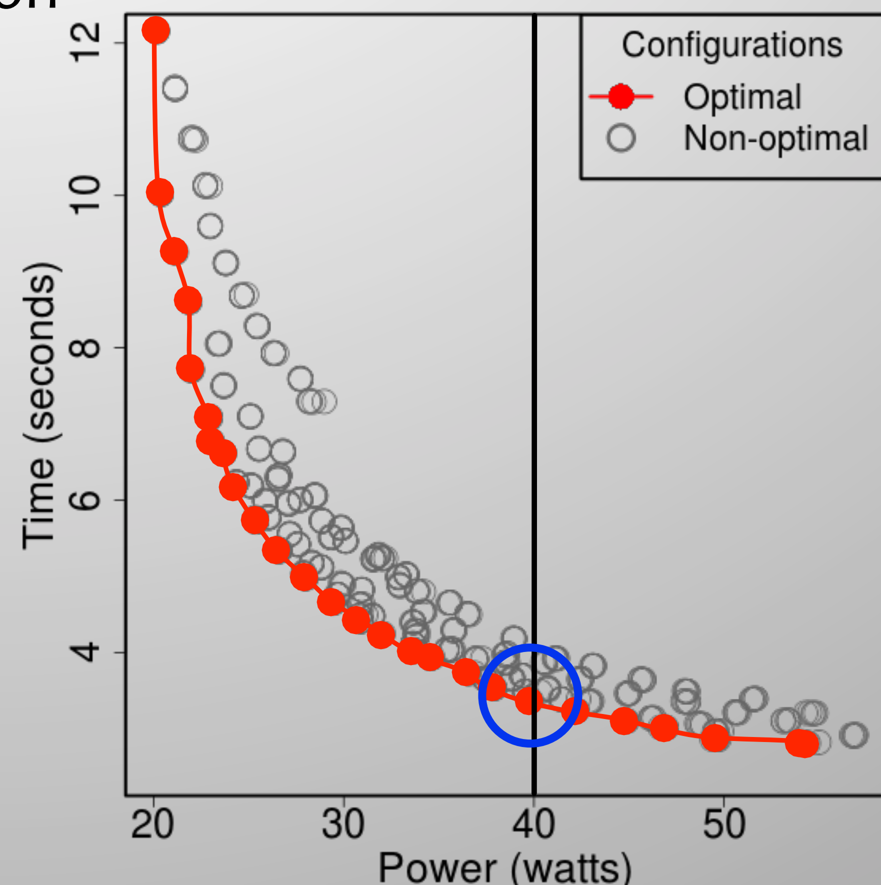
- **Central question**

  Given a job-level power constraint,
  how do we optimize application performance?

- **Main idea**
  - Identify critical path
    - Only the critical path needs full power
    - The rest can work with reduced power
  - Measure power headroom
    - Execute application for controlled period of time and measure power
    - Can be distributed based on process criticality
  - Execute repeatedly during application execution
    - Typically on time step boundaries
    - Intended for repetitive applications

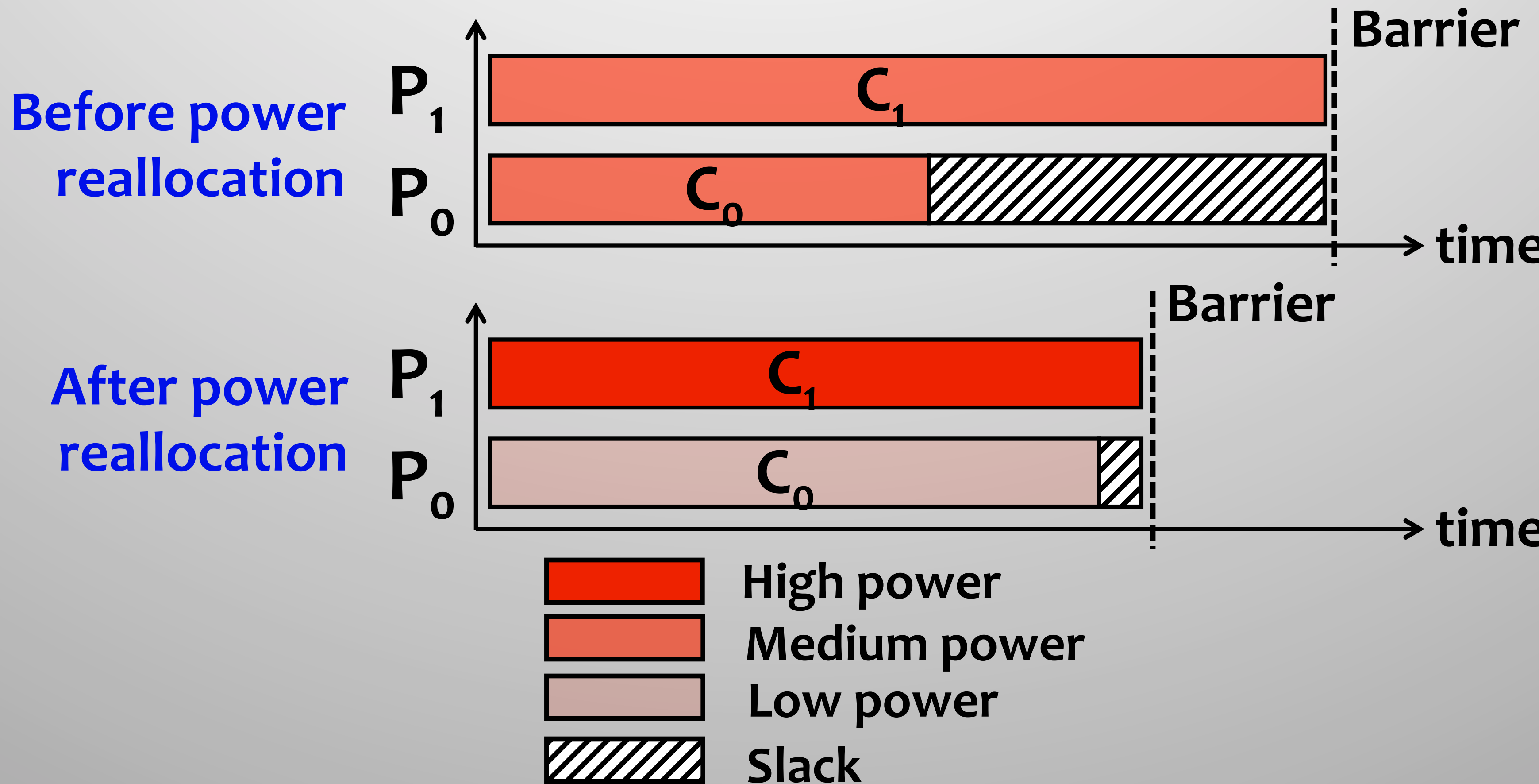
Step I: Configuration Selection

- **Profile the configuration space on-line**
  - Run each computation operation on individual nodes at distinct configurations
    - Exploit parallelism
  - Record the power/perf. profile characteristics of each computation operation
  - Construct Pareto frontier
  - Pick best configuration under a given power bound

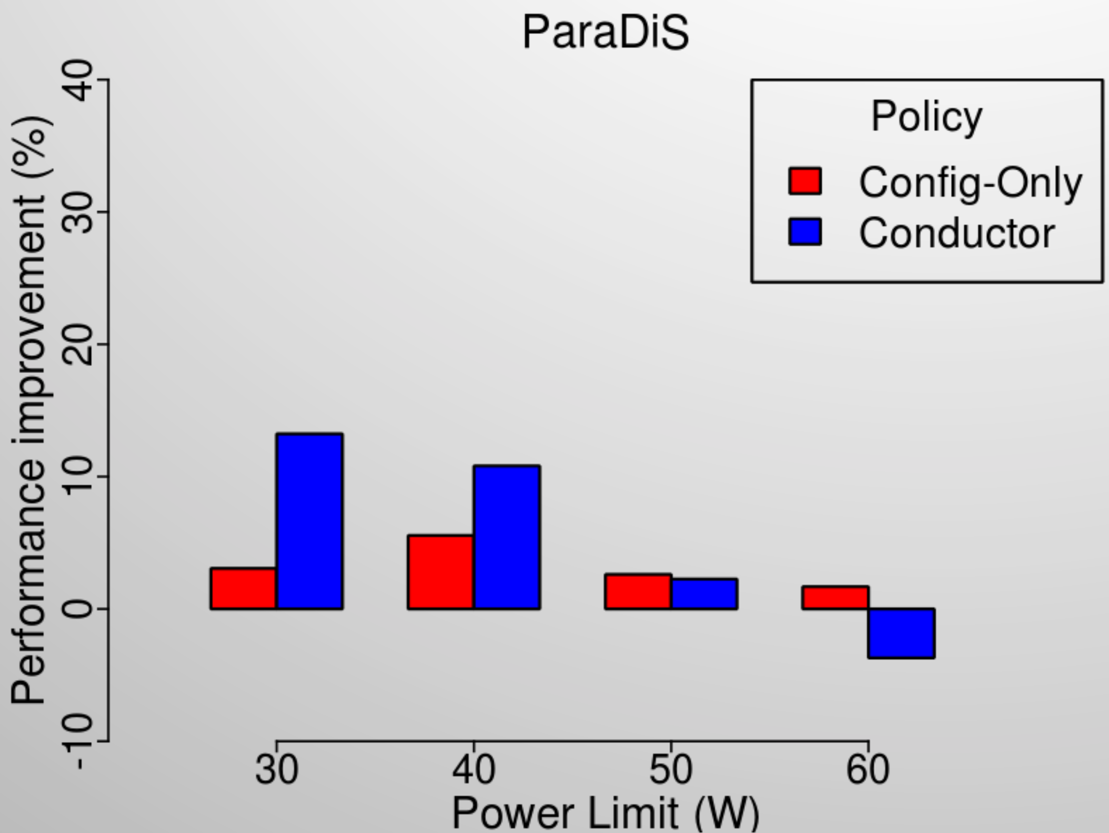


Step II: Power Reallocation

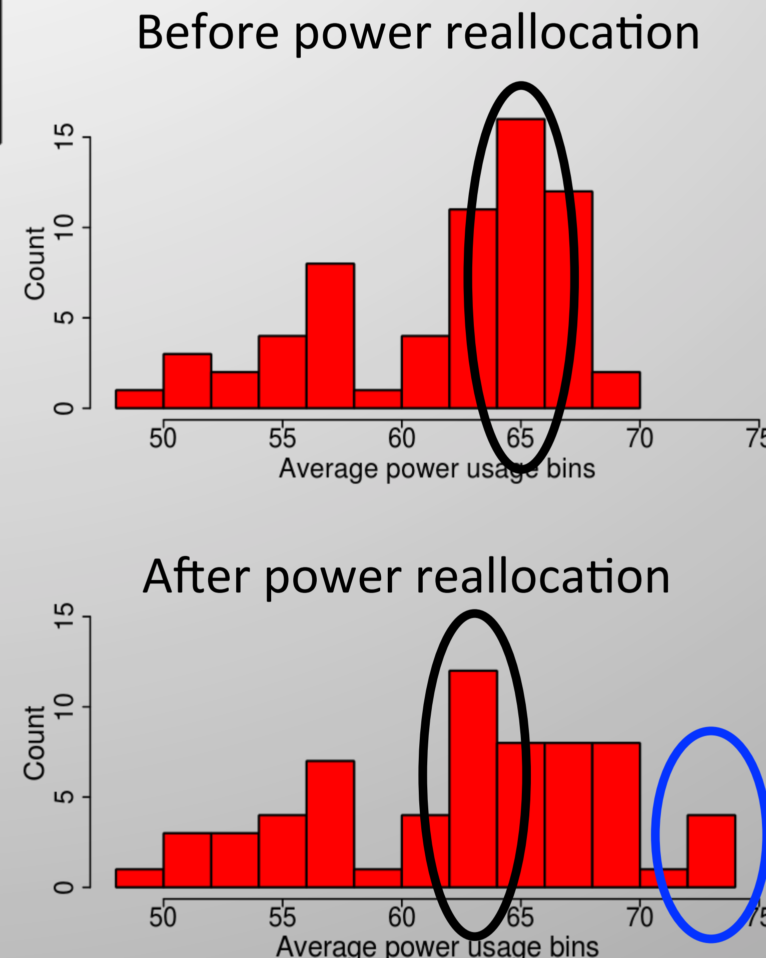
- How can we allocate power to the critical operations in an application and improve performance?



Conductor Benefits Dynamic Applications

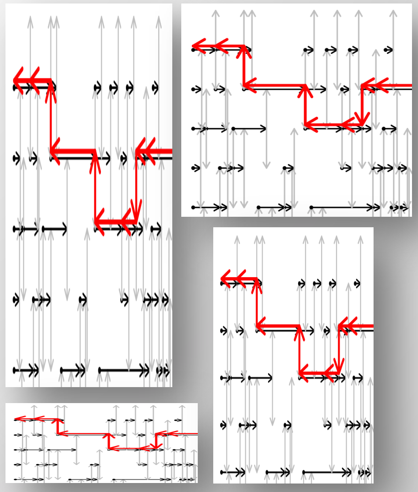


64 nodes / 512 processes
on Intel SandyBridge with RAPL



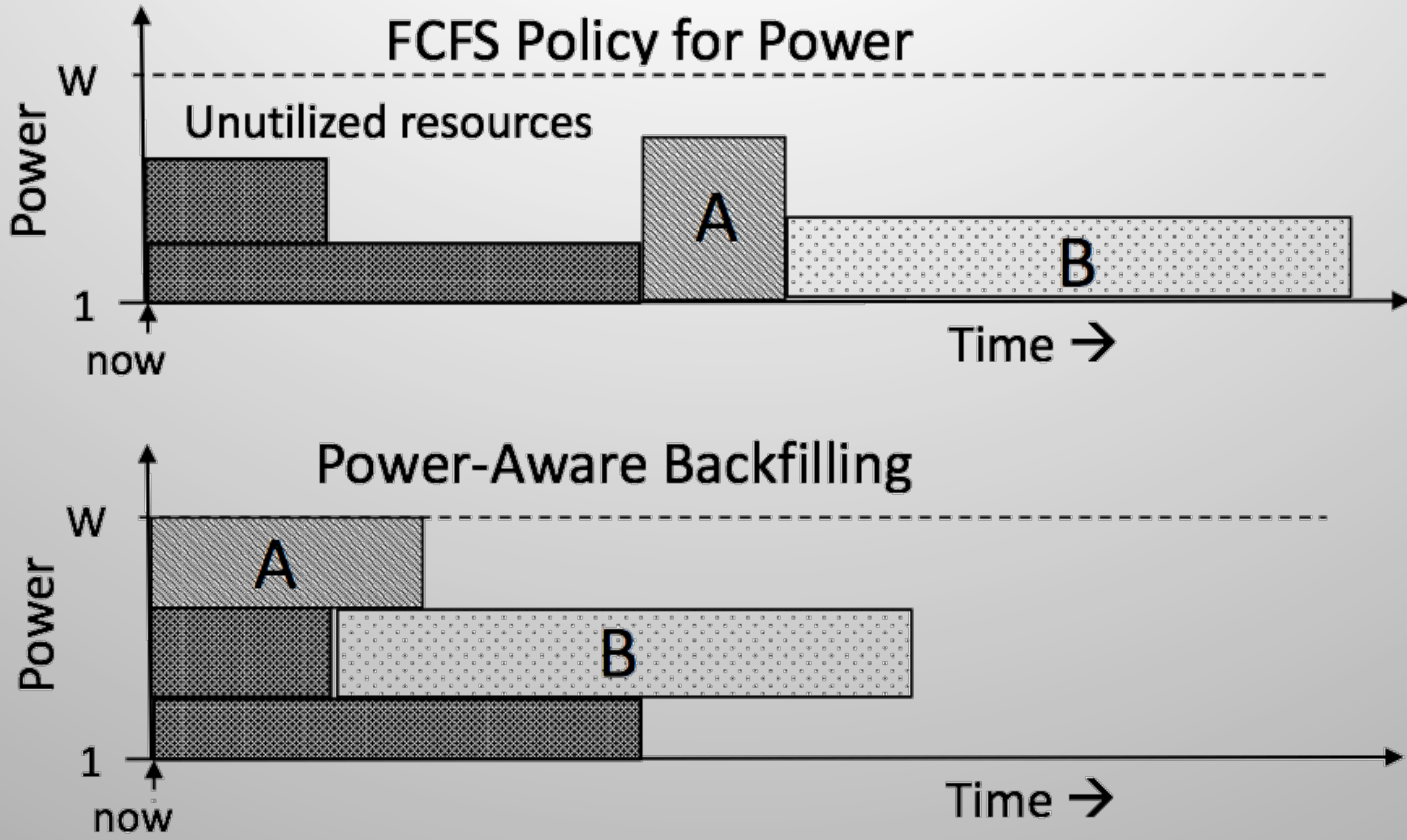
- Up to 13% speedup over Static scheme
- Benefits from process-level imbalance of power usage

Power-aware Resource Management

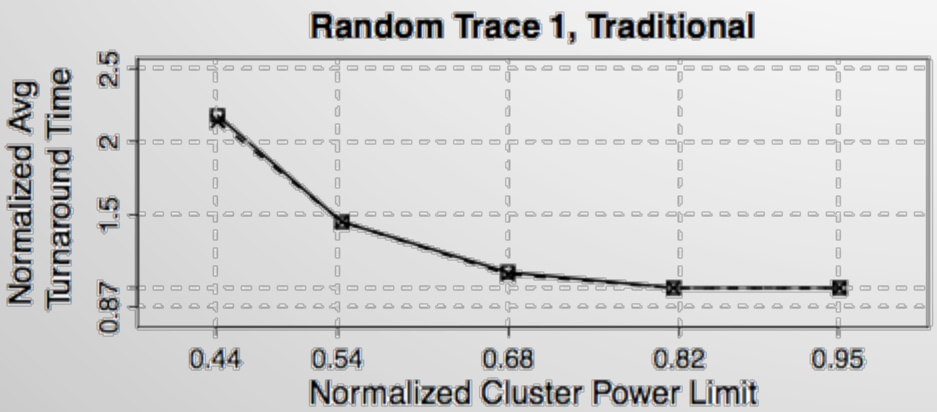


- **Power is a global resource**
  - The system power cap must be divided among jobs
  - Static division results in power fragmentation
  - Dynamic management can utilize open resources
- **Direction 1: Power-aware Resource Management**
  - Power as a controlled resource that is allocated
  - Initial step: power aware backfilling in P-SLURM

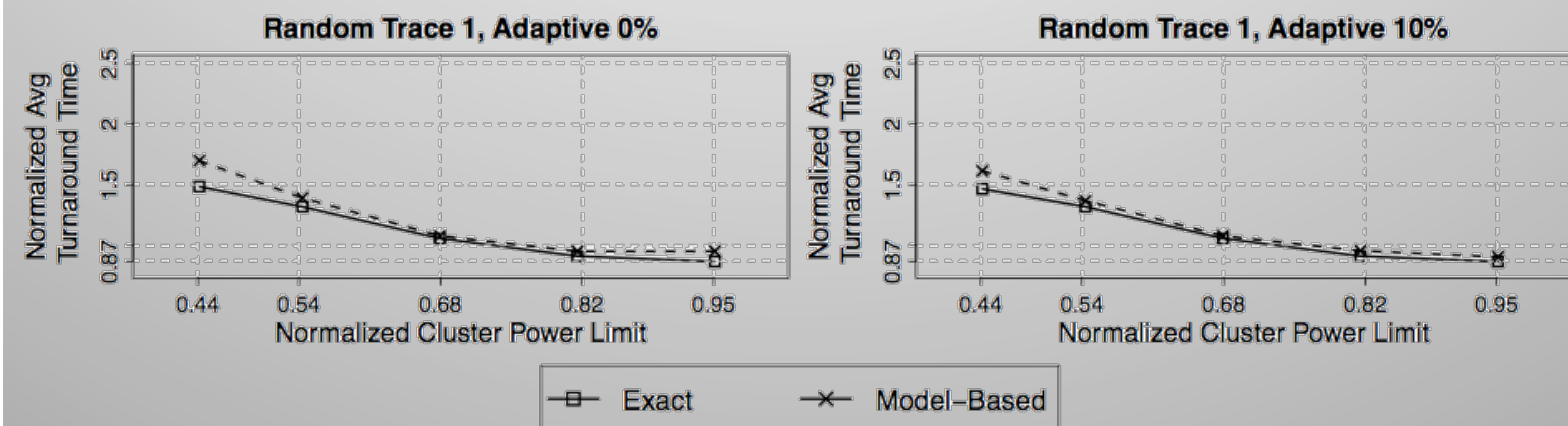
Power Aware Backfilling



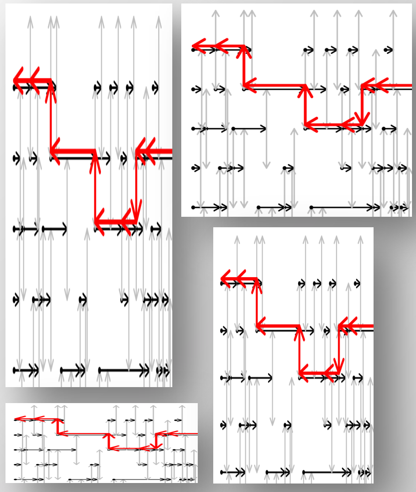
Results of Turnaround Times for Different Policies



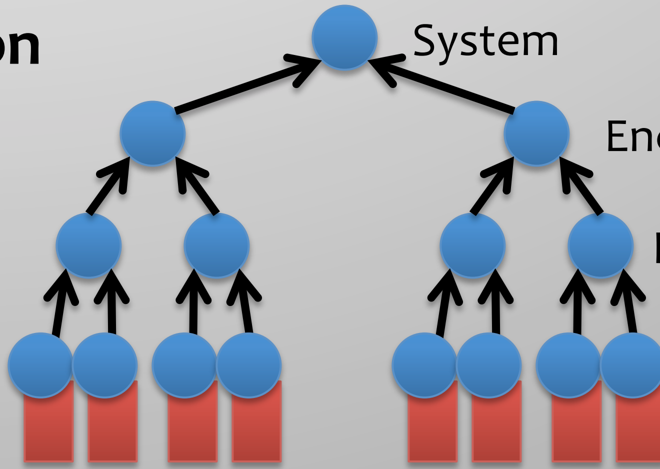
- **Traditional**
  - Require nodes at full power
- **Adaptive**
  - Adjust node power
  - Limit slowdown



Power-aware Resource Management

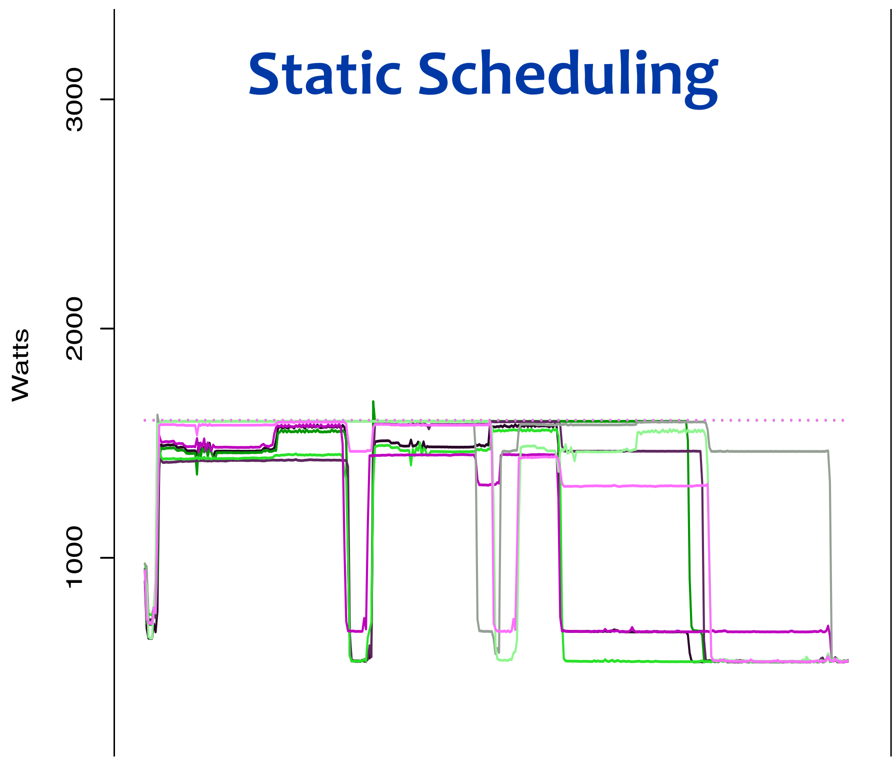


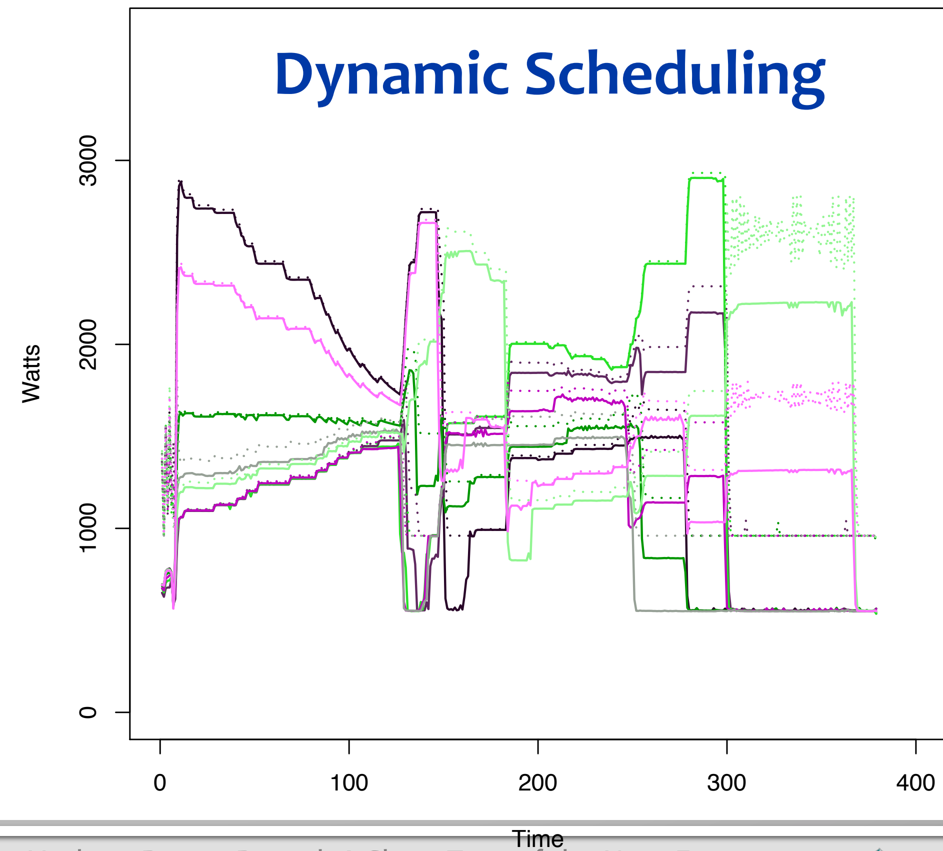
- **Power is a global resource**
  - The system power cap must be divided among jobs
  - Static division results in power fragmentation
  - Dynamic management can utilize open resources
- **Direction 1: Power-aware Resource Management**
  - Power as a controlled resource that is allocated
  - Initial step: power aware backfilling in P-SLURM
- **Direction 2: Runtime Adaptation**
  - Part of a global operating system
  - Detection and reallocation of unused power
  - Transparent to application
  - Need to maintain fairness



POWsched: Power Scheduler for the Exascale

- **8 Enclaves with different job mixes**
  - Static vs. dynamic scheduling under same power bound
  - Dynamic power measurement and control





Conclusions

- **Hard power limits will lead us to overprovisioned systems**
  - More hardware than we can power
    - Leads to power capping
    - Exposes inhomogeneity
  - Selectively distribute power to the right place
    - Within applications using adaptive runtime control (Conductor)
    - Across applications by the OS (POWsched)
    - At job allocation by the resource manager (P-SLURM)
- **Needs to be driven by power/performance models**
  - Complex relationships
  - Inhomogeneity is application dependent
  - Current models are very empirical
    - Work well in current settings and achieve promising results
    - Long term: need more accurate understanding of such models
- **Basis for efficiently utilizing overprovisioned systems!**

The Scalability Team
http://scalability.llnl.gov/

Abhinav Bhatele | David Boehme | Todd Gamblin | Tanzima Islam | Ignacio Laguna | Kathryn Mohror | Barry Rountree | Martin Schulz

- **Main topics**
  - Performance analysis tools and optimization
  - Correctness and debugging (incl. STAT, AutomaDeD, MUST)
  - Tool infrastructures (incl. P^{n}MPI, GREMLINs)
  - **Power-aware and power-limited computing (incl. P-SLURM & Conductor)**
  - Resilience and Checkpoint/Restart (incl. SCR)
- **Close collaboration with Universities of Arizona and Oregon & LMU/LRZ**
- **Funding sources involved in presented work:**

Conclusions

http://scalability.llnl.gov/

- **Hard power limits will lead us to overprovisioned systems**
  - More hardware than we can power
    - — Leads to power capping
    - — Exposes inhomogeneity
  - Selectively distribute power to the right place
    - — Within applications using adaptive runtime control (Conductor)
    - — Across applications by the OS (POWsched)
    - — At job allocation by the resource manager (P-SLURM)
- **Needs to be driven by power/performance models**
  - Complex relationships
  - Inhomogeneity is application dependent
  - Current models are very empirical
    - — Work well in current settings and achieve promising results
    - — Long term: need more accurate understanding of such models
- **Basis for efficiently utilizing overprovisioned systems!**