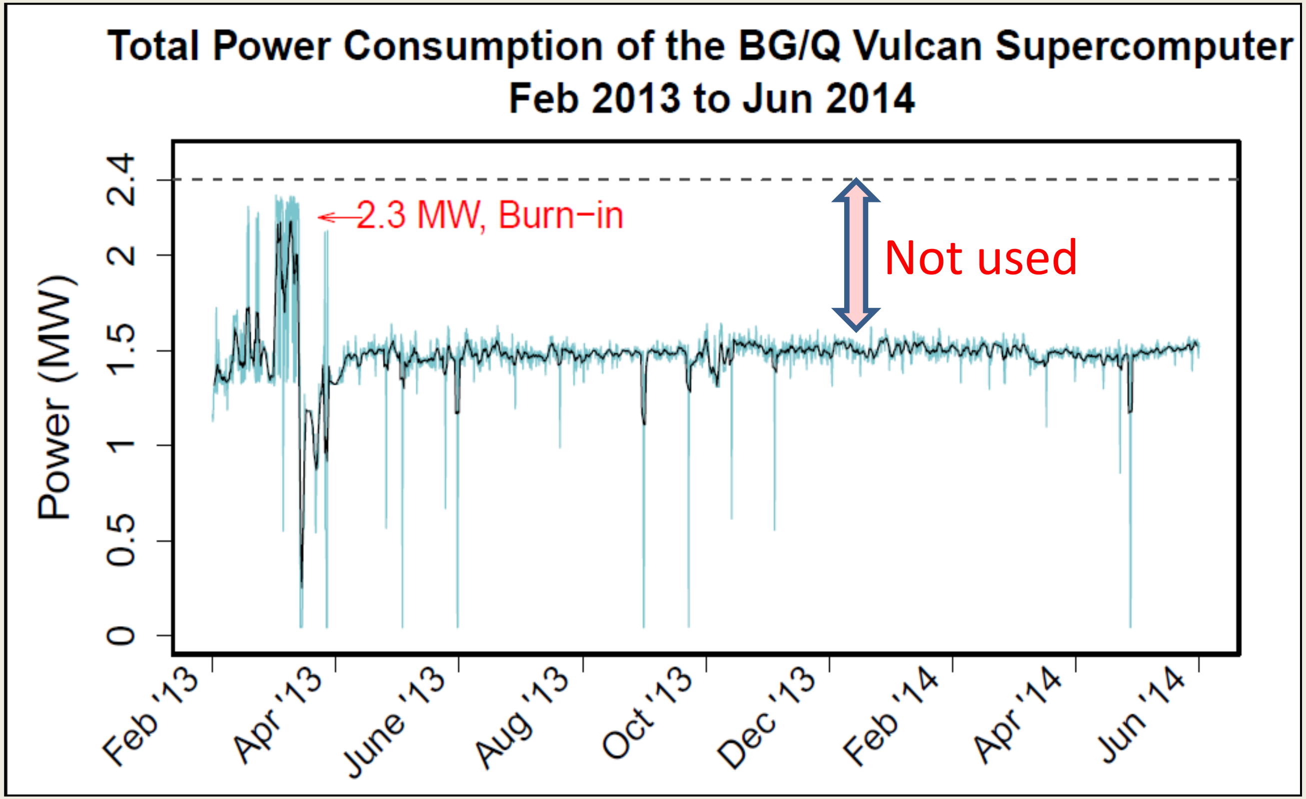
Practical Resource Management in Power-Constrained, High Performance Computing

Tapasya Patki*, David Lowenthal, Anjana Sasidharan, Matthias Maiterth, Barry Rountree, Martin Schulz, Bronis R. de Supinski

June 18, 2015

The Power Problem



40% Procured Power Unused!

Exascale Power Problem

20

Megawatts

0

5% Desirable range

100%

Power Utilization

0%

Problem is power utilization, not just power procurement

Do Not:

"Save" Power

Do :

Use power to improve performance and system throughput

Accomplish more science

Hardware Overprovisioning

Worst-case provisioning (traditional):

All nodes can run at peak power simultaneously

Hardware overprovisioning:

- Buy more capacity, limit power per node
- Reconfigure dynamically based on application's memory and scalability characteristics
- Moldable applications: flexible in terms of node and core counts on which they can execute

Configurations

Good performance relies on choosing an *application-specific configuration*

- Number of nodes, cores per node and power per node, *(n x c, p)*

In our case, improves performance under a power bound by 32% (1.47x) on average compared to worst-case provisioning

Hardware Overprovisioning Example

SP-MZ CFD kernel, Bound of 3500 W

Configuration	(n x c, p)	Time (s)	Total Power: CPU & DRAM (W)
Worst-case	(20 x 16, 115)	9.10	3250
Overprovisioned	(26 x 12, 80)	3.65	3497

CPU Power Cap

2.5x Speedup

Utilized all allocated power

Resource Management

What is the impact of overprovisioning when we have a real cluster with multiple users and several jobs?

Can we utilize the procured power better and minimize wasted power?

Power-Aware Scheduling Challenges

<u>User:</u> Users care about *fairness* and *turnaround time*

- Fair and transparent job-level power allocation
- Minimize execution time, reduce queue wait time

<u>System:</u> Admins care about *utilization* and *throughput*

- Maximize utilization of available nodes and power
- Minimize *average* turnaround time for job queue

Resource MAnager for Power (RMAP)

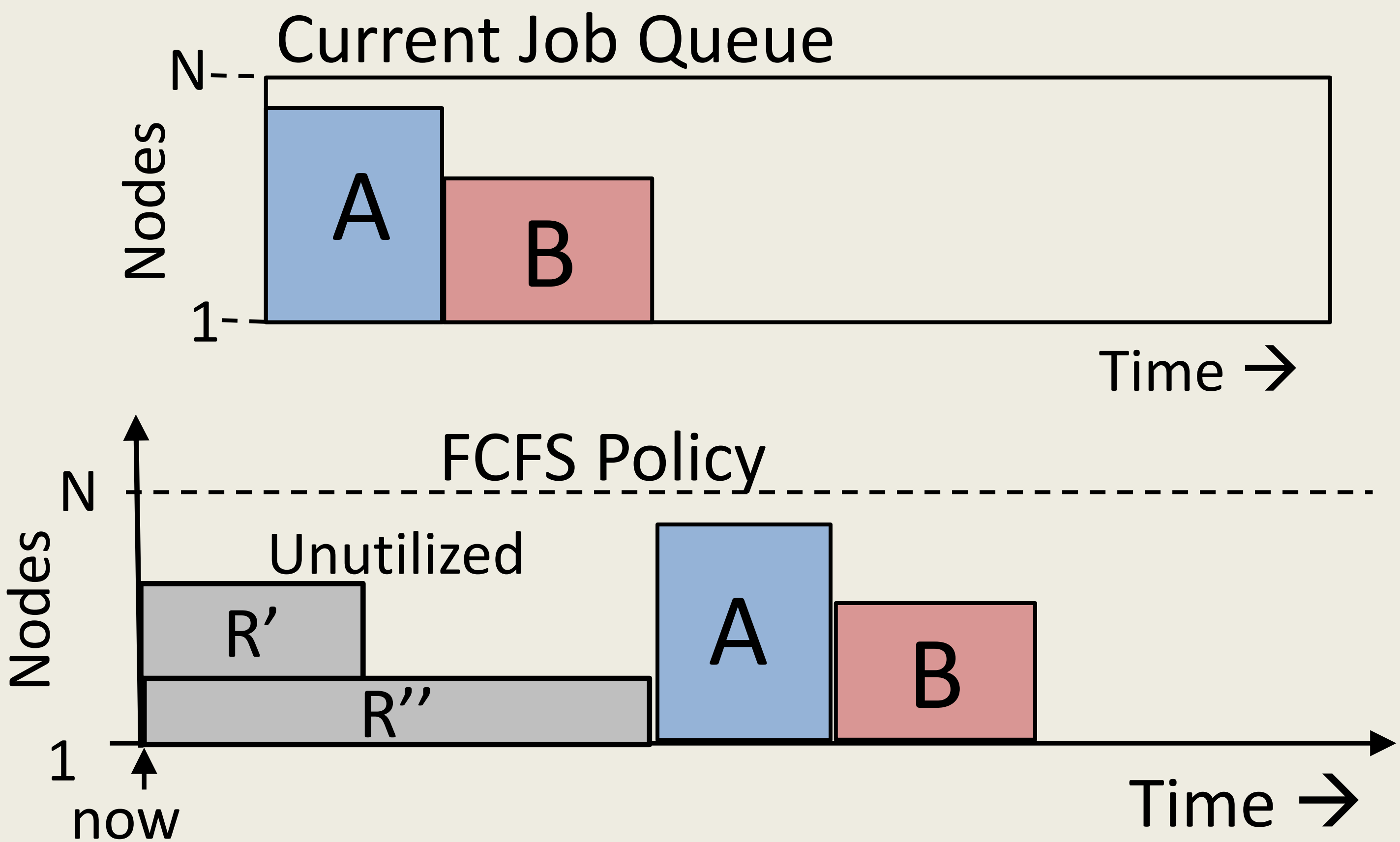
Aimed at future power-constrained systems

Implemented within SLURM

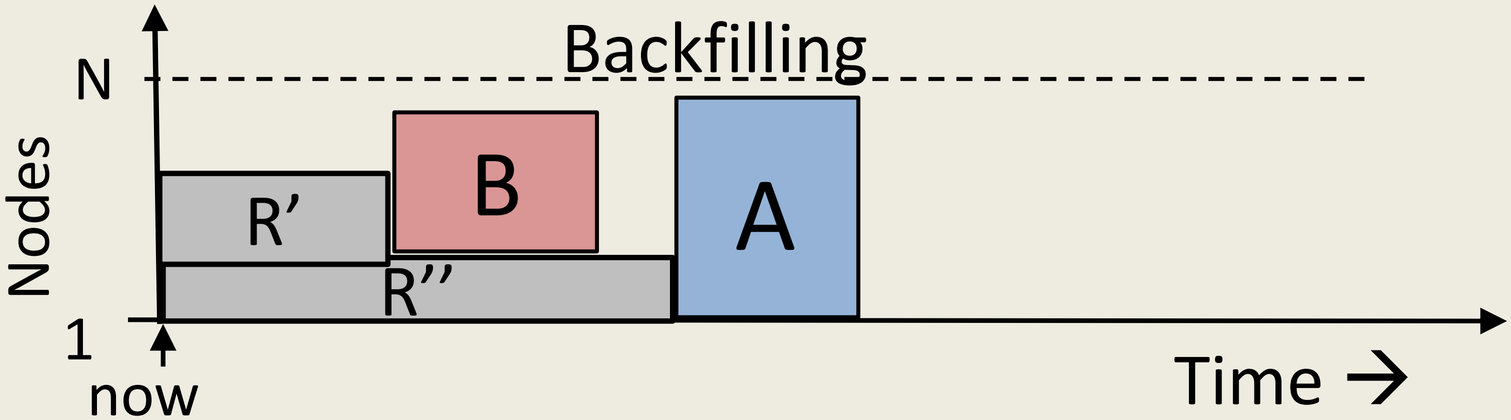
Novel Adaptive policy:

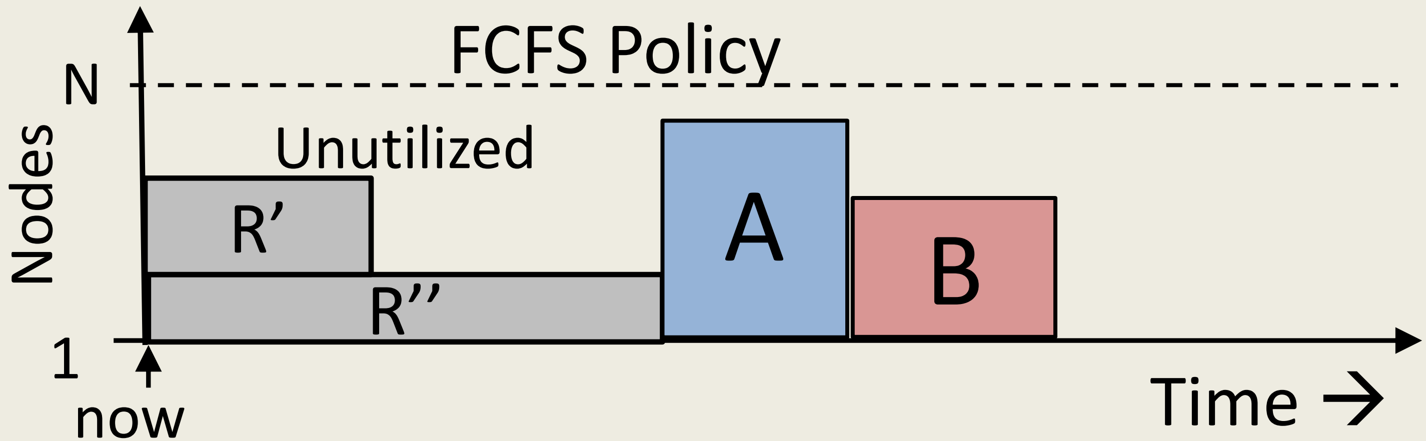
- Uses overprovisioning and *power-aware backfilling*
- Improves system power utilization and optimizes execution time under a job-level power bound
- Leads to 19% and 36% faster turnaround times than baseline *Traditional* and *Naïve*

First-Come First Serve Scheduling

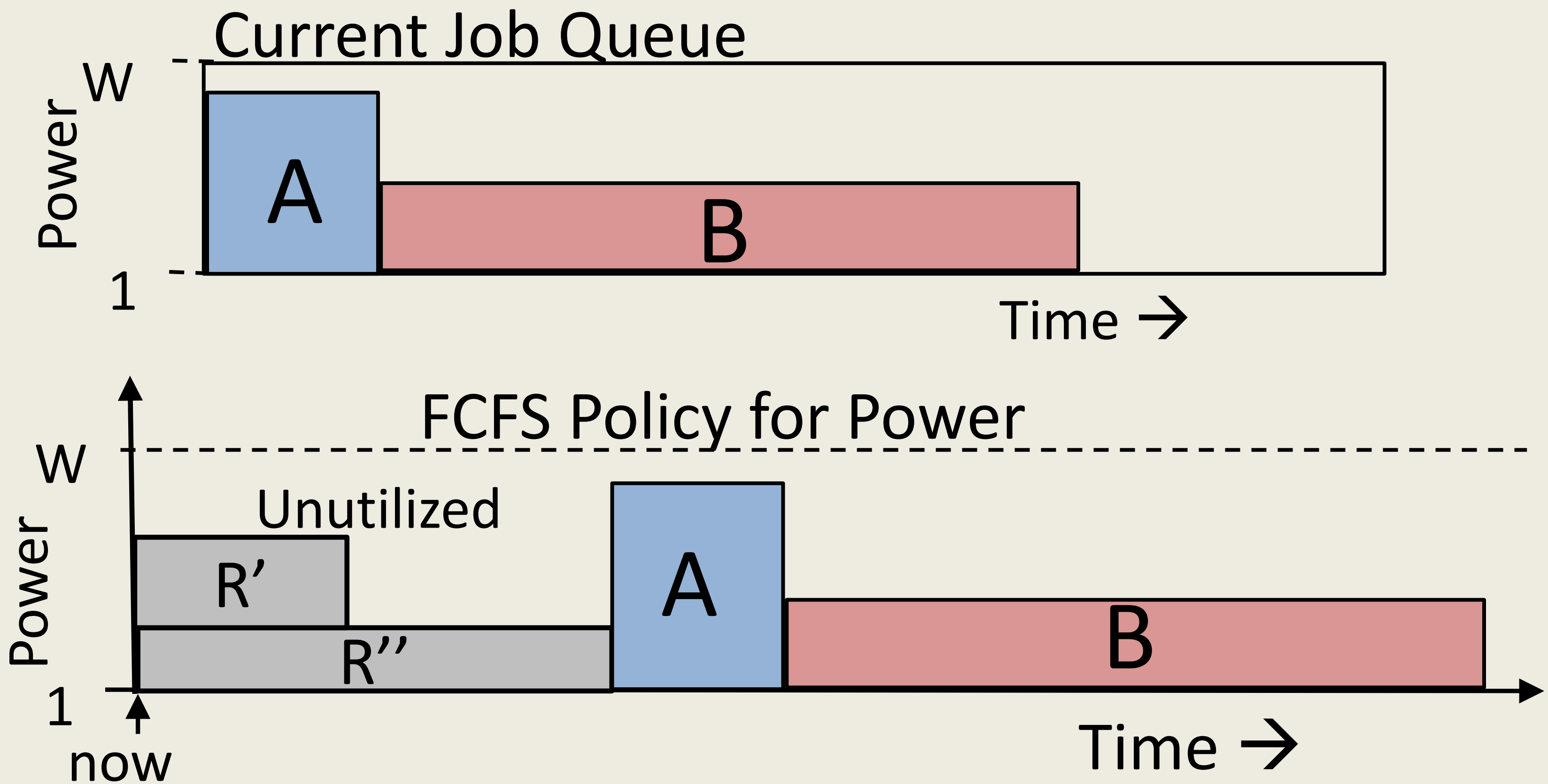


Backfilling

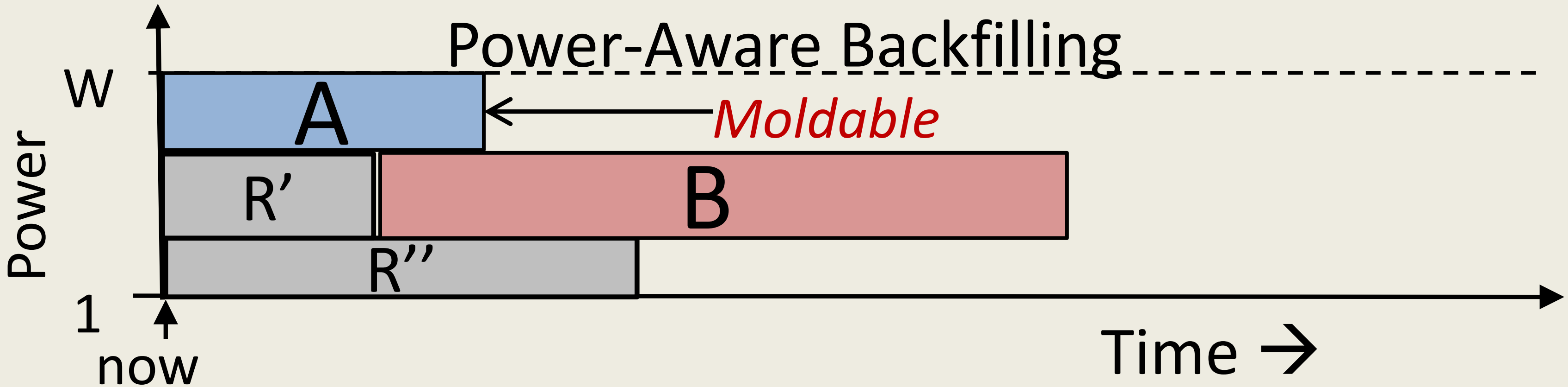


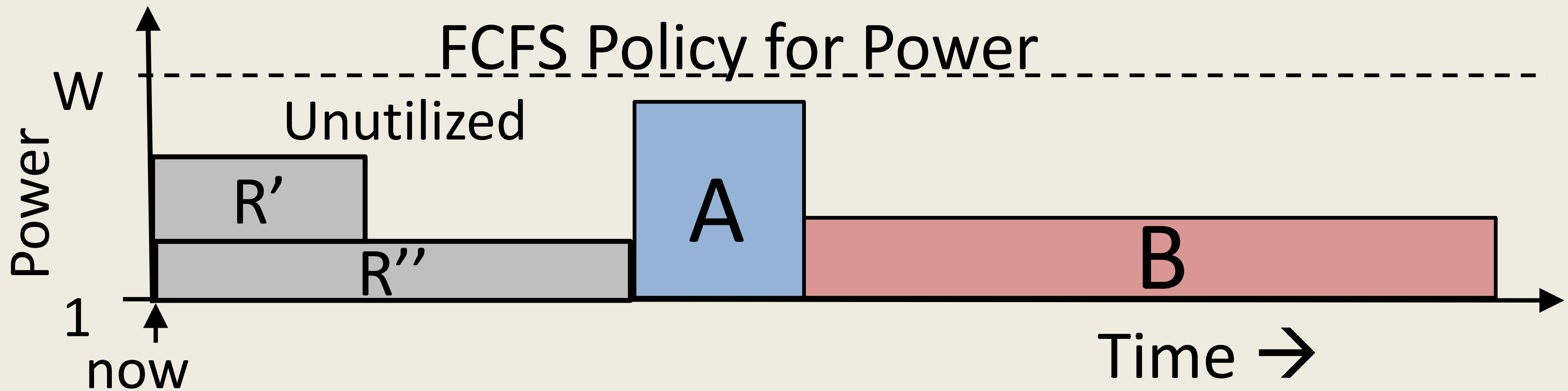


Insight: *Power-Aware* Backfilling



Insight: *Power-Aware* Backfilling





RMAP Policies: Inputs

Users request nodes and time

Job-level power bound:

- Fairly allocate power to each job based on the fraction of total nodes requested

We assume equal priority.

RMAP Adaptive Policy

- If *enough power is available*, allocate the *best overprovisioned configuration* under the derived job-level power bound
- Otherwise, *allocate a suboptimal overprovisioned configuration* with *available* power
- Users can specify an optional performance slowdown *threshold* for potentially faster turnaround times
  - Default is no slowdown (0%)

RMAP Baseline Policies

Policy	Description
Traditional	Not fair-share, allocates *requested* nodes with all cores at full power
Naïve	Greedily allocates best performing configuration under derived job-level power bound

**All policies use basic node-level backfilling.*

Experimental Details

Intel Sandy Bridge 64-node cluster

- 2 sockets per node, 8 cores per socket
- Min: 51 W, Max: 115 W

Intel RAPL for power measurement and control*

Moldable Applications

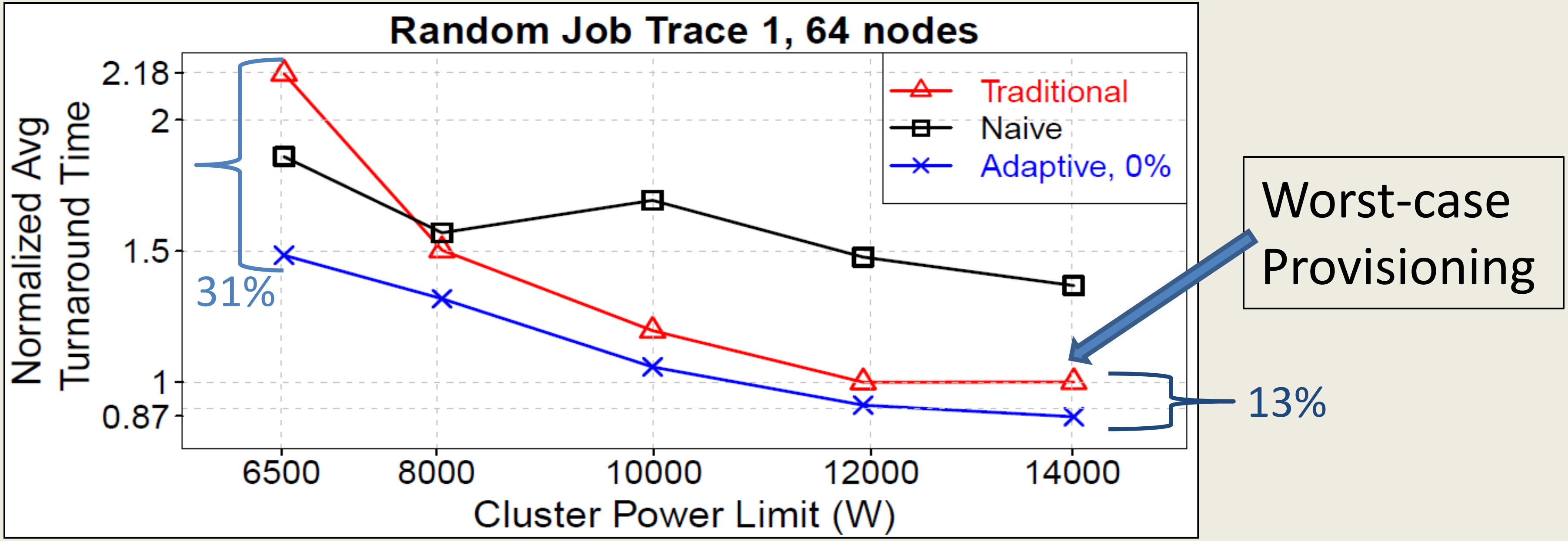
- SPhot , NAS-MZ (BT-MZ, SP-MZ and LU-MZ)
- Four synthetic

**DRAM power unavailable*

Evaluation

- SLURM Simulator, 64 nodes, 30 jobs per trace
- 5 global power bounds
  - 6500 W, extremely constrained
  - 14000 W, unconstrained
- Poisson process for dynamic job arrival

Random Trace Results



On average Adaptive with no slowdown does 19% better than Traditional, 36% better than Naïve

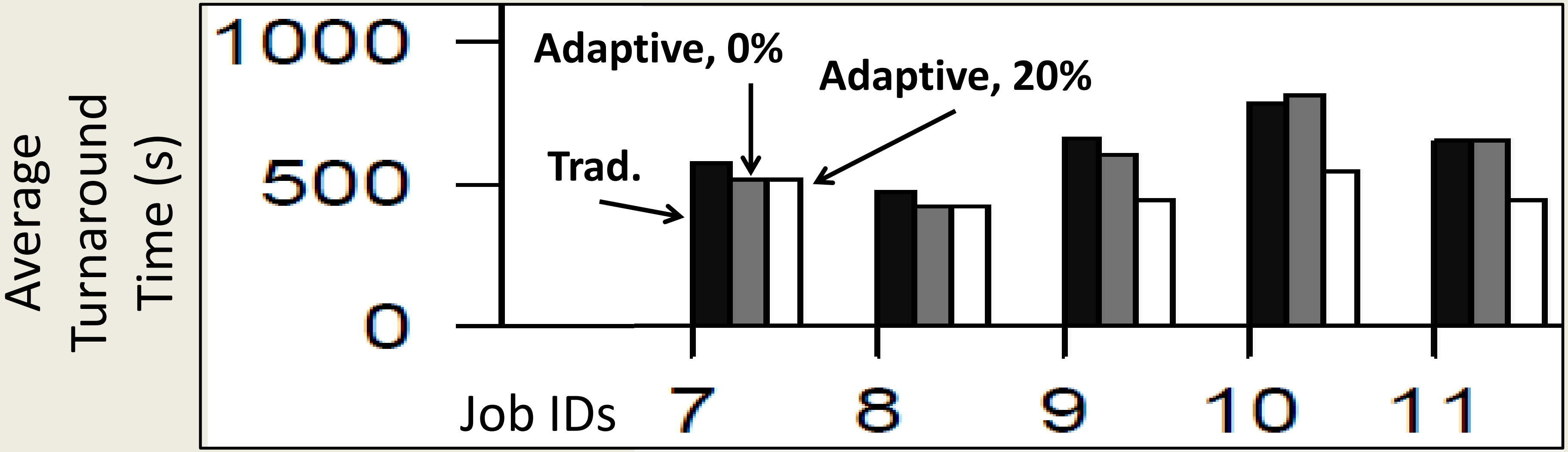
Detailed Results: 6500 W Bound

Extremely power-limited,
128 processors, 50 W per socket with fair-share

Each job requests at least 40 nodes

Policy	Average Turnaround Time (s)
Traditional	684
Naïve	990
Adaptive, 0%	636 (7% better than Traditional)
Adaptive, 20%	536 (21% better than Traditional)

Detailed Results: 6500 W Bound



- 22 of 30 jobs have faster turnaround times than Traditional
- 21% faster turnaround time (on average)

RMAP Summary

Adaptive policy

- Uses hardware overprovisioning and power-aware backfilling
- Leads to 19% and 36% faster queue turnaround times than *Traditional* and *Naïve*
- Improves individual application performance as well as system throughput

Acknowledgments

- Livermore Computing at Lawrence Livermore National Laboratory for MSR access
- Dr. Ghaleb Abdulla for help with Vulcan power data