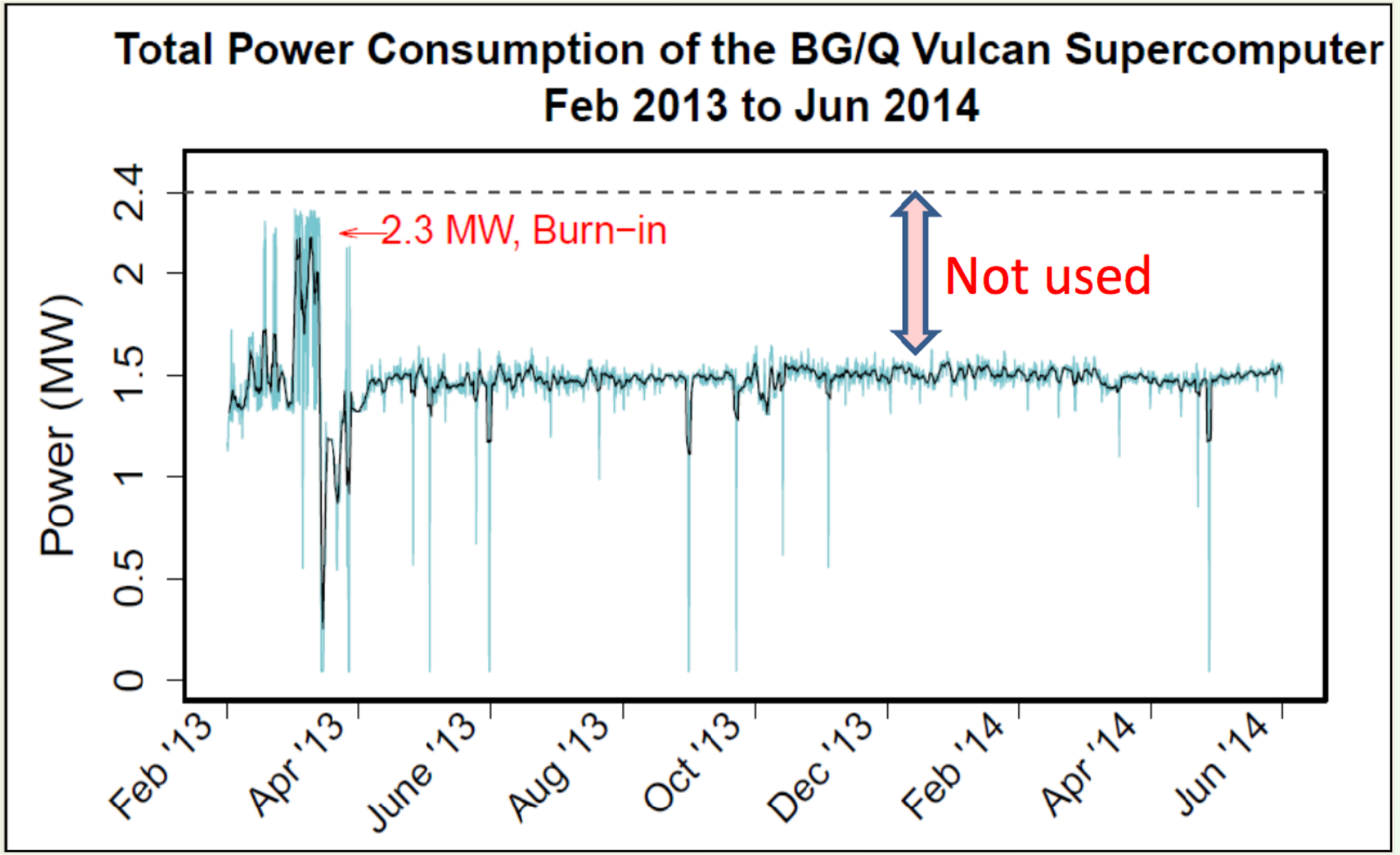
Dynamic Power Management for Hardware Over-provisioned Systems

Daniel Ellsworth
dellswor@cs.uoregon.edu

LLNL-PRES-705519

Observed Power Consumption



Patki, et al; HPDC 2015

Why Hardware Over-provision?

30% more computer for the same power cost
(more science per unit time)

— or —

Save 30% on power for the same size computer
(reduce annual opex by $1 million/MW)

Work Contributions

- Formalization of what a power scheduler must do
- PowSched - An application agnostic power scheduling solution
- A simulator for evaluating dynamic power scheduling strategies
- Behavior comparisons for high-level power scheduling strategies

Power Scheduler Invariant

$$\forall t, \sum_{i=1}^{n} c_i^t \leq L$$

L	System-wide power limit
n	Number of components
t	Time
c_i^t	Power consumed by component i at time t
a_i^t	Power allocated to component i at time t

$$\forall (t,i), c_i^t \leq a_i^t \text{ and } \forall t, \sum_{i=1}^{n} a_i^t \leq L \implies \forall t, \sum_{i=1}^{n} c_i^t \leq \sum_{i=1}^{n} a_i^t \leq L$$

Three Basic Approaches

- System Static - Set all allocations at machine install time and never again

$$\forall(t,i), a_i^t = \frac{L}{n}$$

- Job Static - Set the allocations for participating nodes when job is started

$$\text{Let } a_j \text{ be the estimated power a job will consume.}$$

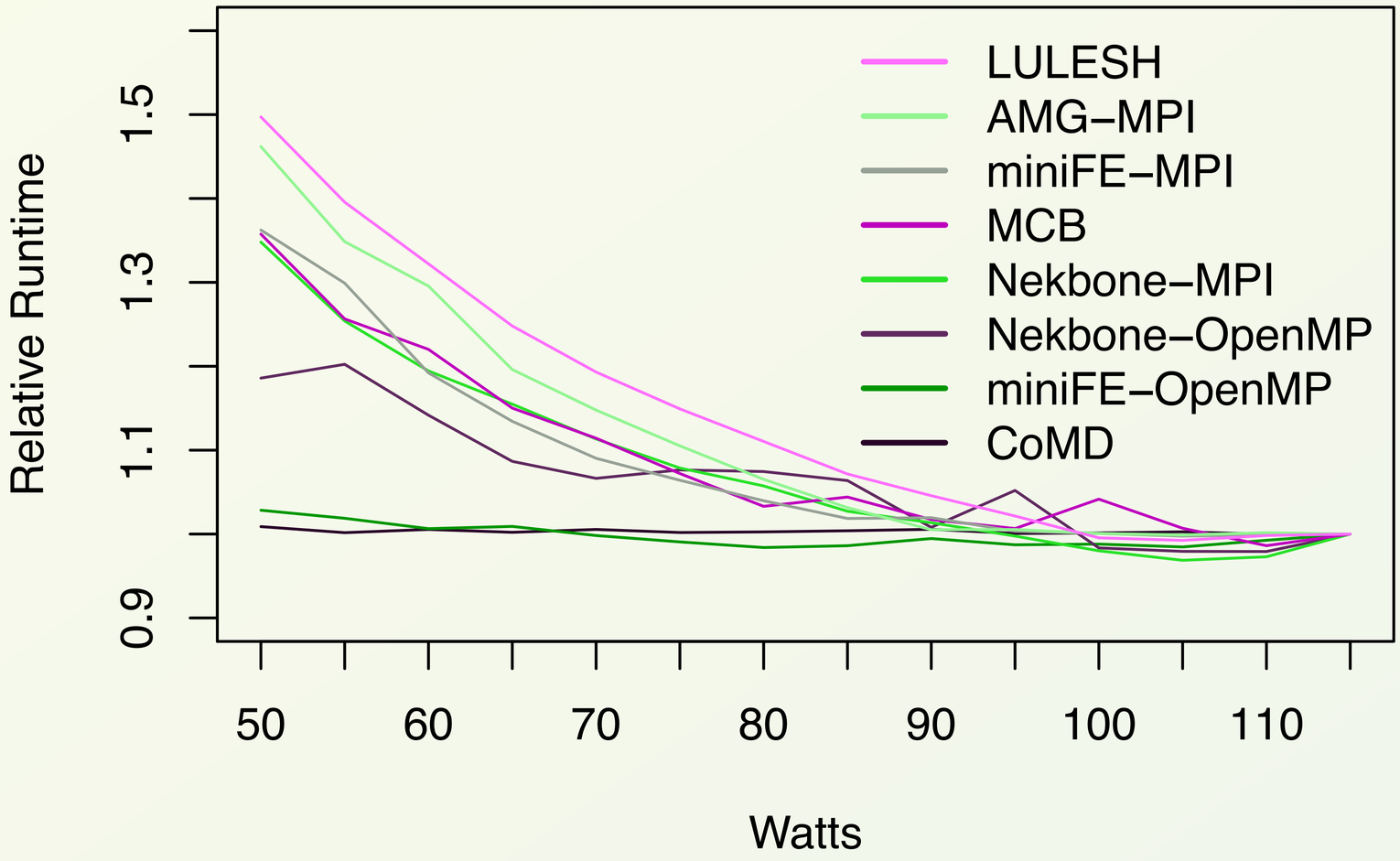
$$\text{Schedule concurrent jobs only when } \forall t, \sum_{j=1}^{J} a_j \leq L$$

$$\text{At job launch, set } a_i \text{ for all nodes in job } j \text{ such that } \sum a_i = a_j$$

- Dynamic - Set the allocations at anytime

$$\text{At any time } t \text{ set new } a_i \text{ such that } \sum_{i=1}^{n} a_i^t \leq L$$

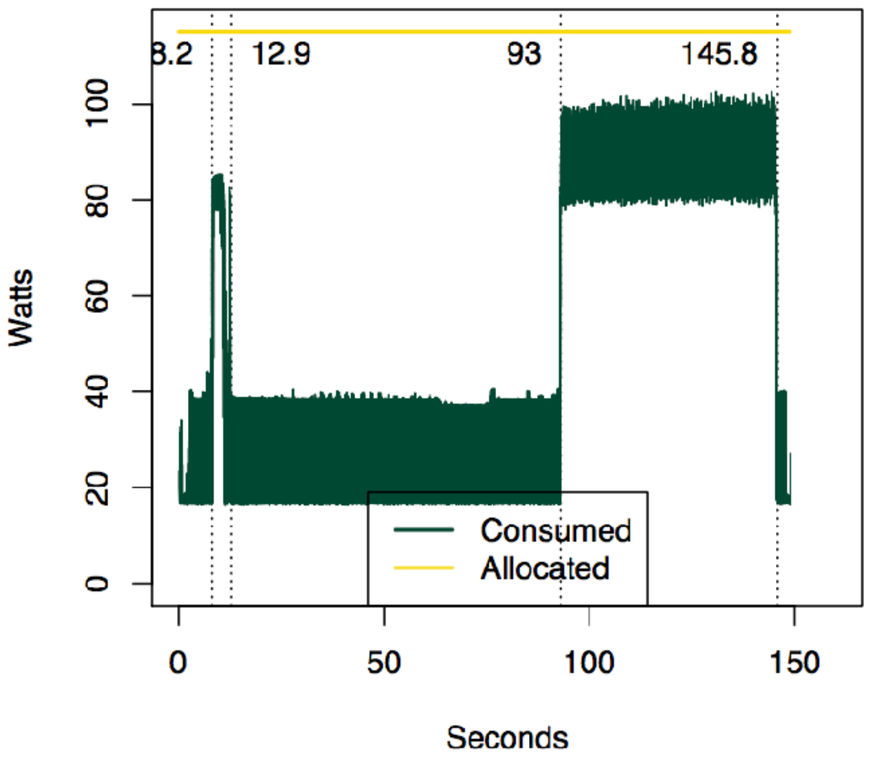
System Static Challenge



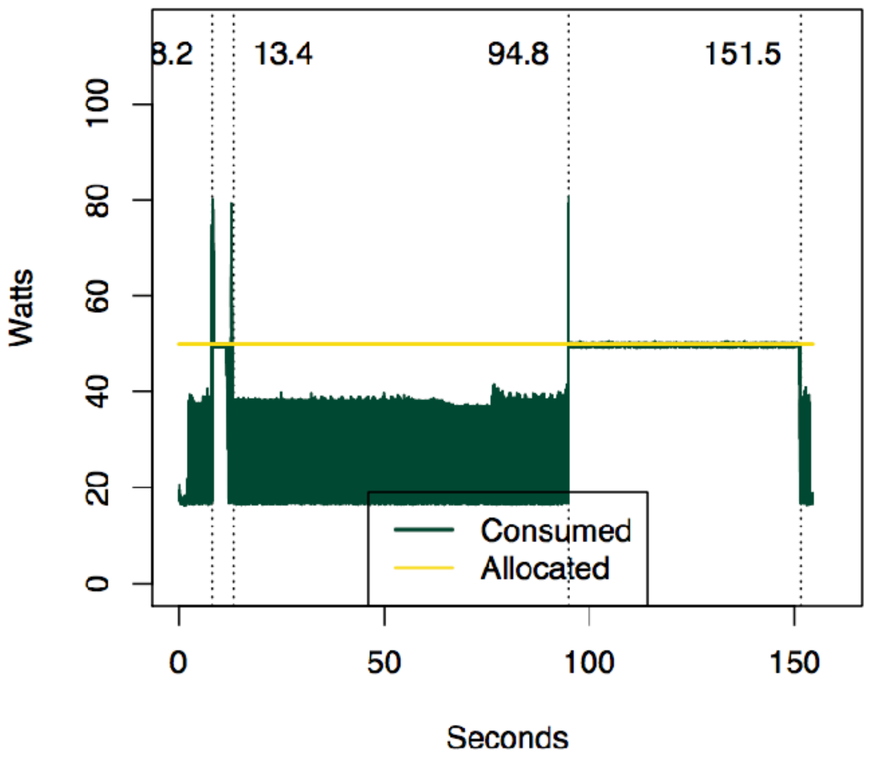
Different applications see performance degradation at different power caps

Job Static Challenge

118 kJ, 148 s -> 50W/sock



89 kJ, 153 s -> 36W/sock



Power consumption may be variable

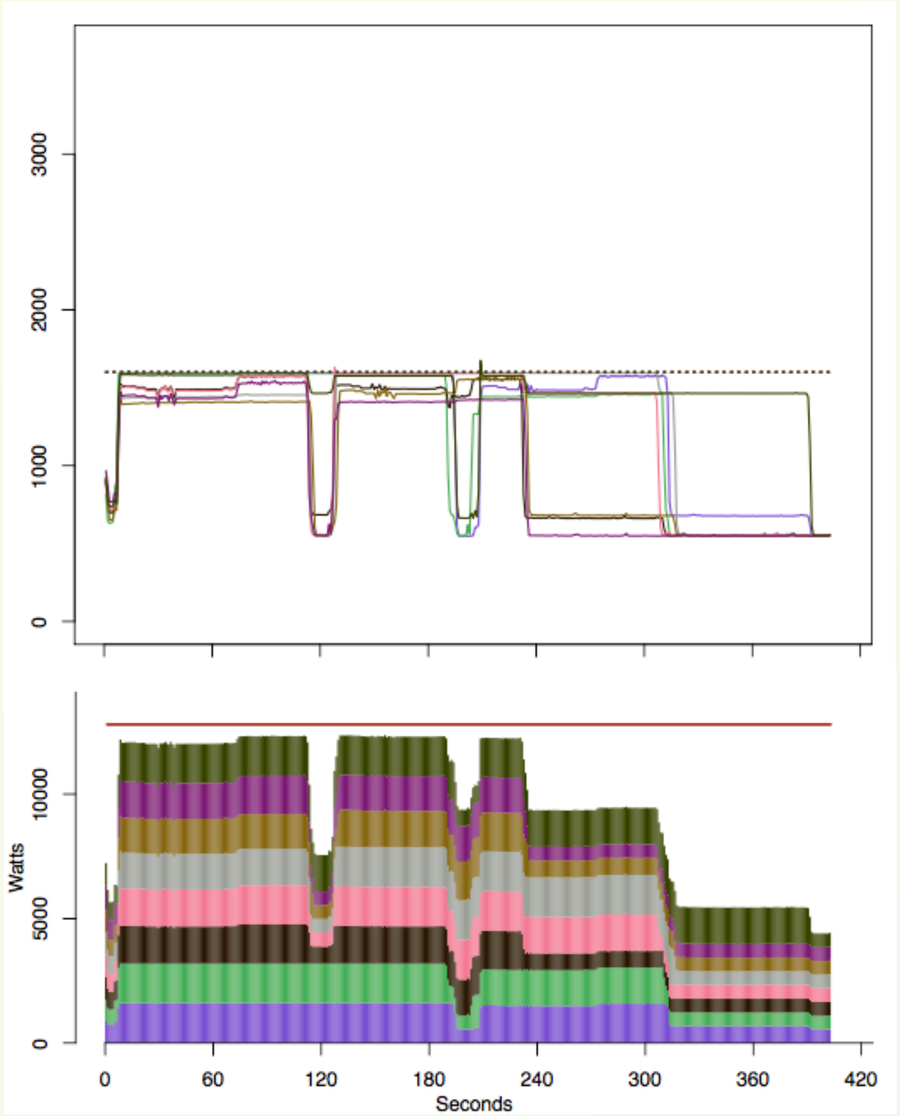
A Dynamic Solution - PowSched

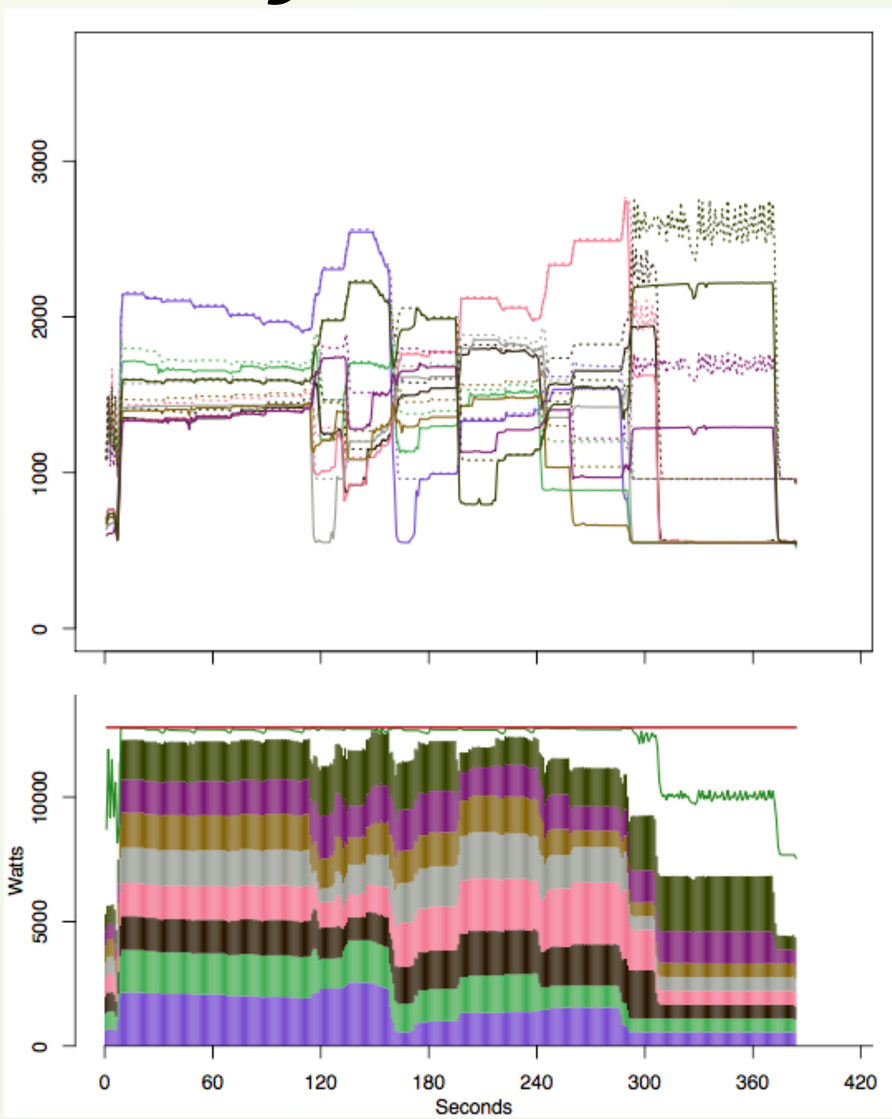
- A dynamic power scheduling solution that is application agnostic
- Basic Scheduler Logic:
  Sockets that use all the power allocated last iteration should get more.
  Take power from sockets that have unused allocated power.
- Experimental results from an implementation in C

Static vs Dynamic

Experiment	Runtime	Stddev	Improvement
115W static	278.26	9.57	
115W dynamic	276.24	4.84	0.7%
90W static	284.63	3.20	
90W dynamic	277.13	5.04	2.6%
70W static	323.83	4.90	
70W dynamic	278.02	4.97	14.1%
50W static	407.21	18.00	
50W dynamic	371.92	13.23	8.7%

50W Static and Dynamic



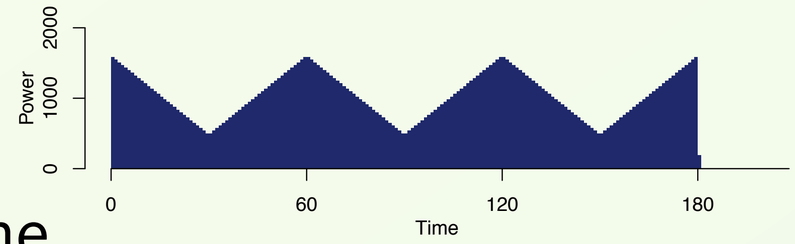


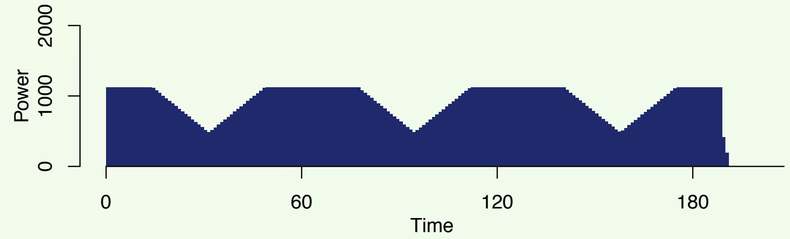
Challenges to Experimentation

- Experiments on real systems take too long and are too variable...
  - Getting a lot of nodes takes a bunch of time
  - Reality is noisy
- Simulation solves both of these problems...
- No suitable simulator existed
- Generated a model and implemented a simulator

Simulator

- For each simulation step
  - If the processor is program bound
    - advance the program clock by the step time
  - If the processor is power bound
    - compute the instruction work done during the step time
    - advance the program clock by an amount corresponding to the compute instruction work done





Ongoing Work

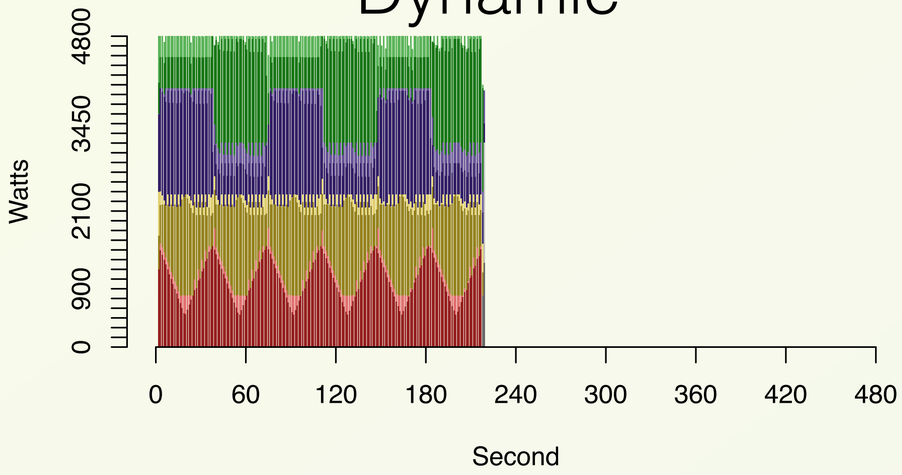
- Comparison of power scheduling strategies
- SLURM Extensions
  - Experimental evaluation on existing HPC hardware
  - See our E2SC workshop paper for details
- Simulation Evaluations
  - What are the characteristic behaviors of different strategies?
  - Under what circumstances does a specific strategy work well?

Generalized Strategies

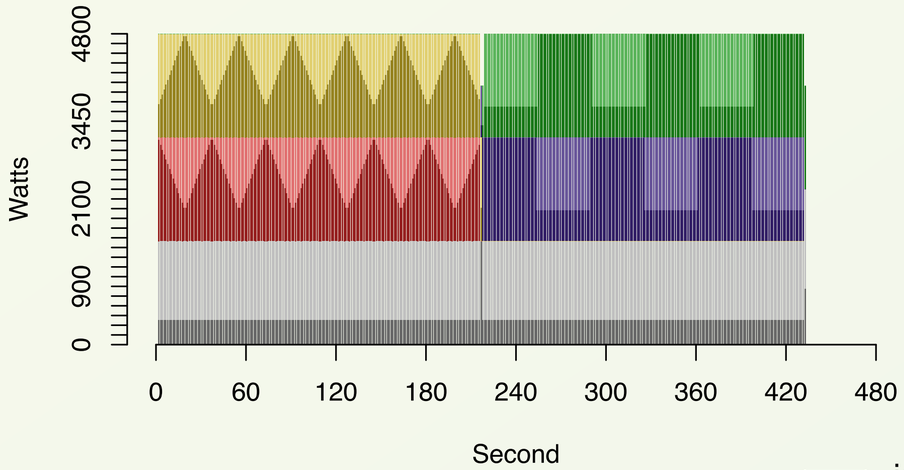
- System Static - Set caps evenly once at machine install time
- Job Static - Set caps for a job once when the job starts
- Dynamic Priority - Set caps periodically, preferring the highest priority job
- Dynamic - Set caps periodically, treat all sockets equally

Preliminary Results

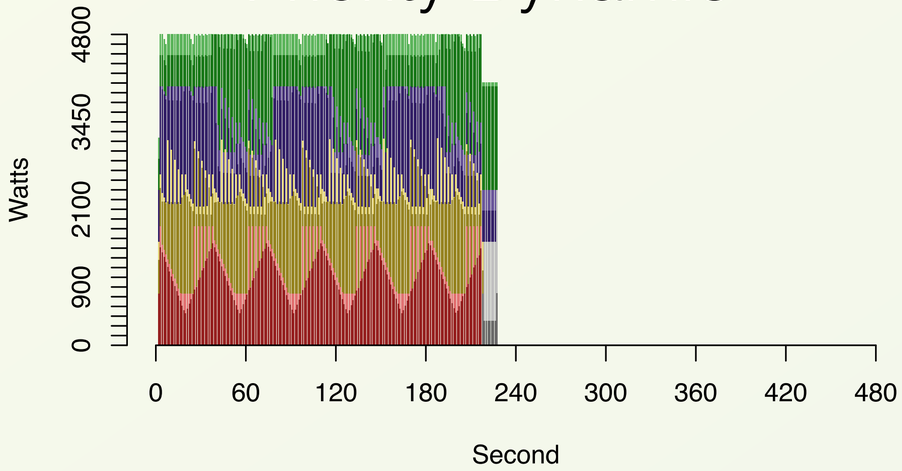
Dynamic



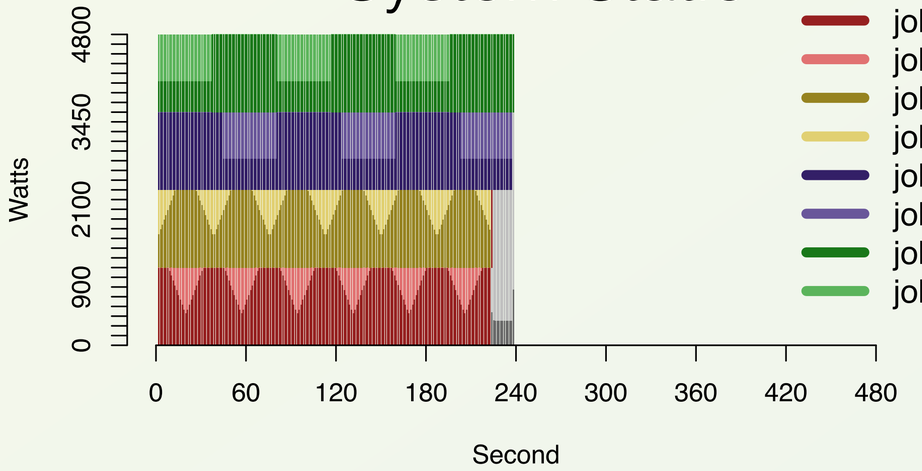
Job Static



Priority Dynamic



System Static



Conclusion

- Hardware over-provisioning creates a power management challenge
- Power Scheduler Invariant formalizes the minimum capability needed
- Dynamic solutions, like PowSched, likely outperform static solutions
- Ongoing study aims to explore comparative performance

Thank You

Contact Information:

Daniel Ellsworth

dellswor@cs.uoregon.edu

Publications:

Unified Platform for Exploring Power Management Strategies. E2SC 2016

Systemwide Power Management With Argo. HPPAC 2016

Dynamic Power Sharing of Higher Job Throughput. SC2015

POW: Systemwide Dynamic Reallocation of Limited Power. HPDC2015

Funding Acknowledgement

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

- Work by the University of Oregon is supported by the DOE Office of Science, through a Sub-Contract No. 3F-32643 from the University of Chicago, Argonne, LLC (as operator of Argonne National Laboratory), under Prime Contract No. DE-AC02-06CH11357.