Power Aware and Temperature Restraint Modeling for Maximizing Performance and Reliability

Laxmikant Kale, Akhil Langer, and Osman Sarood

Parallel Programming Laboratory (PPL)
University of Illinois Urbana Champaign (UIUC)

Yelp Dataset Challenge

- Currently working at Yelp
- Academic dataset from Phoenix, Las Vegas, Madison, Waterloo and Edinburgh!
  - 1,125,458 Reviews
  - 42,153 Businesses
  - 252,898 Users
- Your academic project, research and/or visualizations submitted by December 31, 2014
- yelp.com/dataset_challenge

Agenda

- Applying thermal restraint to
  - Remove hot spots and reduce cooling energy consumption
  - Improve reliability and hence performance
- Operation under strict power budget
  - Maximizing throughput of the entire data center having multiple jobs
- **End Goal**: Combining thermal and power constraints to optimize performance in faulty environment

Hot spots

Hardware, infrastructure people: Help!

Should we help ourselves?

HPC Cluster Temperature Map, Building 50B room 1275, LBNL

1. Dale Sartor, General Recommendations for High Performance Computing Data Center Energy Management Dashboard Display (IPDPSW 2013)

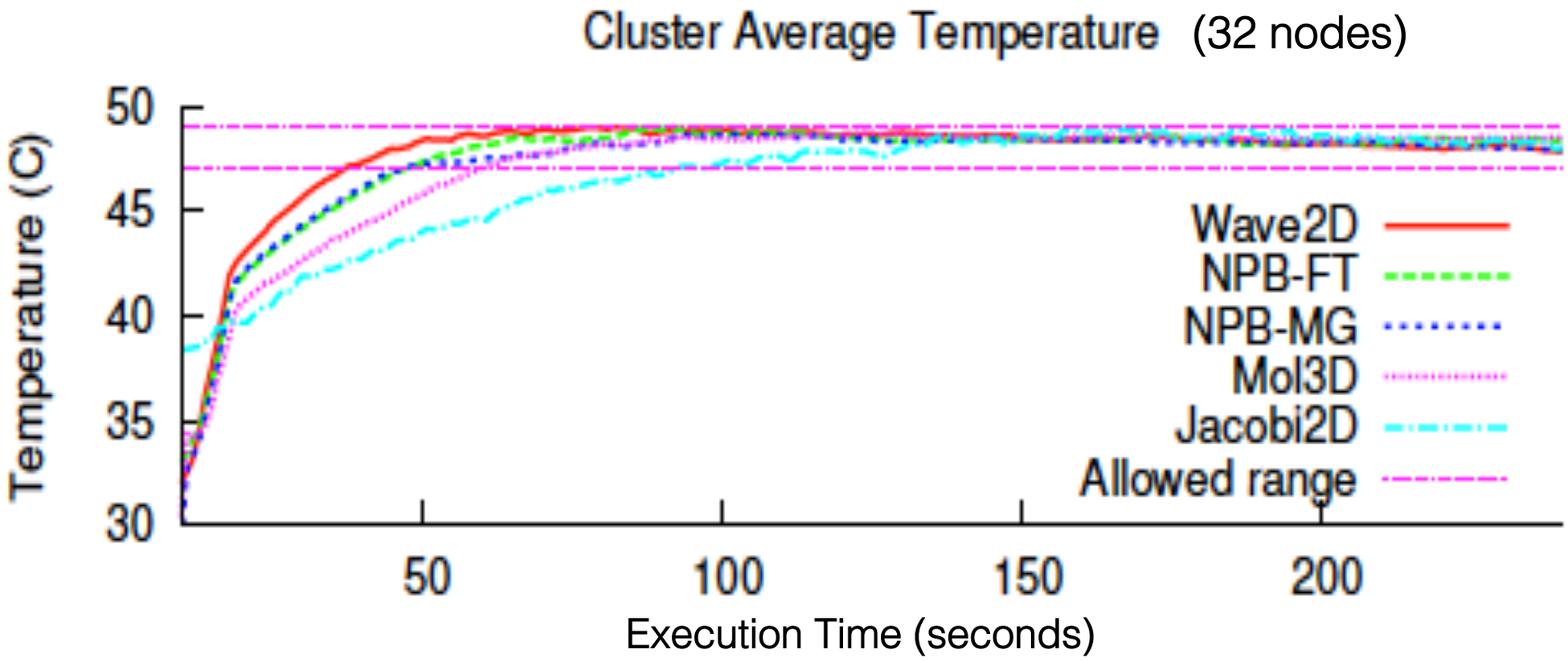
`Cool' Load Balancer

- Uses Dynamic Voltage and Frequency Scaling (DVFS)
- Specify temperature range and sampling interval
- Runtime system periodically checks processor temperatures
- Scale down/up frequency (by one level) if temperature exceeds/below maximum threshold at each decision time
- Transfer tasks from slow processors to faster ones
- Using Charm++ adaptive runtime system
- For details see SC'11 proceedings*

* O. Sarood, L. Kale. A `Cool' Load Balancer for Parallel Applications, Supercomputing'11 (SC'11)

Average Core Temperatures in Check

CRAC set-point = 25.6C Temperature range: 47C-49C



- Avg. core temperature within 2 C range
- Execution time penalty minimized using Charm++ load balancing
- Cooling energy savings of up to 63% with 11% delay in execution time (Mol3d: molecular dynamics application)

* O. Sarood, P. Miller, E. Totoni, L. Kale. `Cool' Load Balancing for HPC Data Centers, IEEE TC 2012

Fault tolerance in present day supercomputers

- Earlier studies point to per socket Mean Time Between Failures (MTBF) of 5 years - 50 years

- More than 20% of computing resources are wasted due to failures and recovery in a large HPC center[1]

- Exascale machine with 200,000 sockets is predicted to waste more than 89% time in failure/ recovery[2]

1. Ricardo Bianchini et. al., System Resilience at Extreme Scale, White paper
2. Kurt Ferreira et. al., Evaluating the Viability of Process Replication Reliability for Exascale Systems, Supercomputing'11

Fault Tolerance: What's new?

- Most earlier software research focusses on improving fault tolerance protocol (*dealing efficiently with faults*)

- Our work focusses on increasing the MTBF (*reducing the occurrence of faults*)

- Our work can be combined with most fault tolerance protocol

CPU Temperature and MTBF

- 10 degree rule: MTBF halves (failure rate doubles) for every 10C increase in temperature[1]
- MTBF (*m*) can be modeled as:

$$m = A * e^{-b*T}$$

  where '*A*', '*b*' are constants and '*T*' is processor temperature
- A single failure can cause the entire machine to fail, hence MTBF for the entire machine (*M*) is defined as:

$$M = \frac{1}{\sum_{n=1}^{N} \frac{1}{m_n}}$$

1. Wu-Chun Feng, Making a Case for Efficient Supercomputing, New York, NY, USA

Improving MTBF and Its Cost

- Temperature restraint comes along DVFS induced slowdown!
- Restraining temperature to 56C, 54C, and 52C for Wave2D (5 point stencil) application using `Cool' Load Balancer

How helpful is the improvement in MTBF considering its cost?

Threshold (C)	MTBF (days)	Timing Penalty (%)
56	36	0.5
54	40	1.5
52	43	4

Timing penalty calculated based on the run where all processors run at maximum frequency

Performance Model

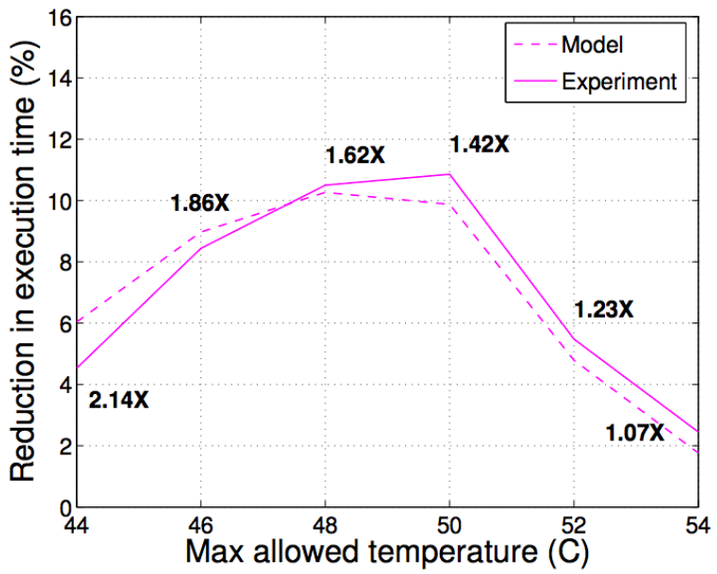
$$T = T_{Solve} + T_{Checkpoint} + T_{Recover} + T_{Restart}$$

- Execution time (T): sum of useful work, check pointing time, recovery time and restart time
- Temperature restraint:
  - increases MTBF which in turn decreases check pointing, recovery, and restart times
  - increases time taken by useful work

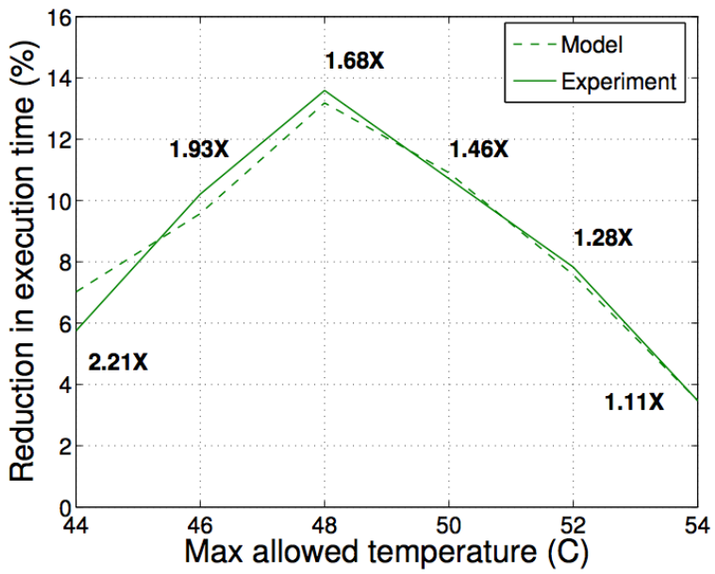
* O. Sarood, E. Meneses, L. Kale. A `Cool' Way of Improving the Reliability of HPC Machines, Supercomputing'13 (SC'13)

Reduction in Execution Time

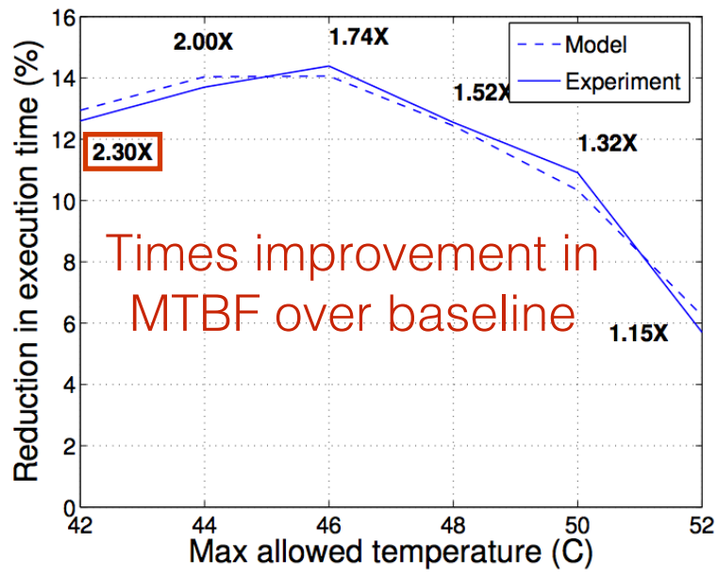
- Inverted-U curve points towards a tradeoff between timing penalty and improvement in MTBF
- ‘Sweet’ spot dependent on application characteristics



(a) Lulesh



(b) Wave2D



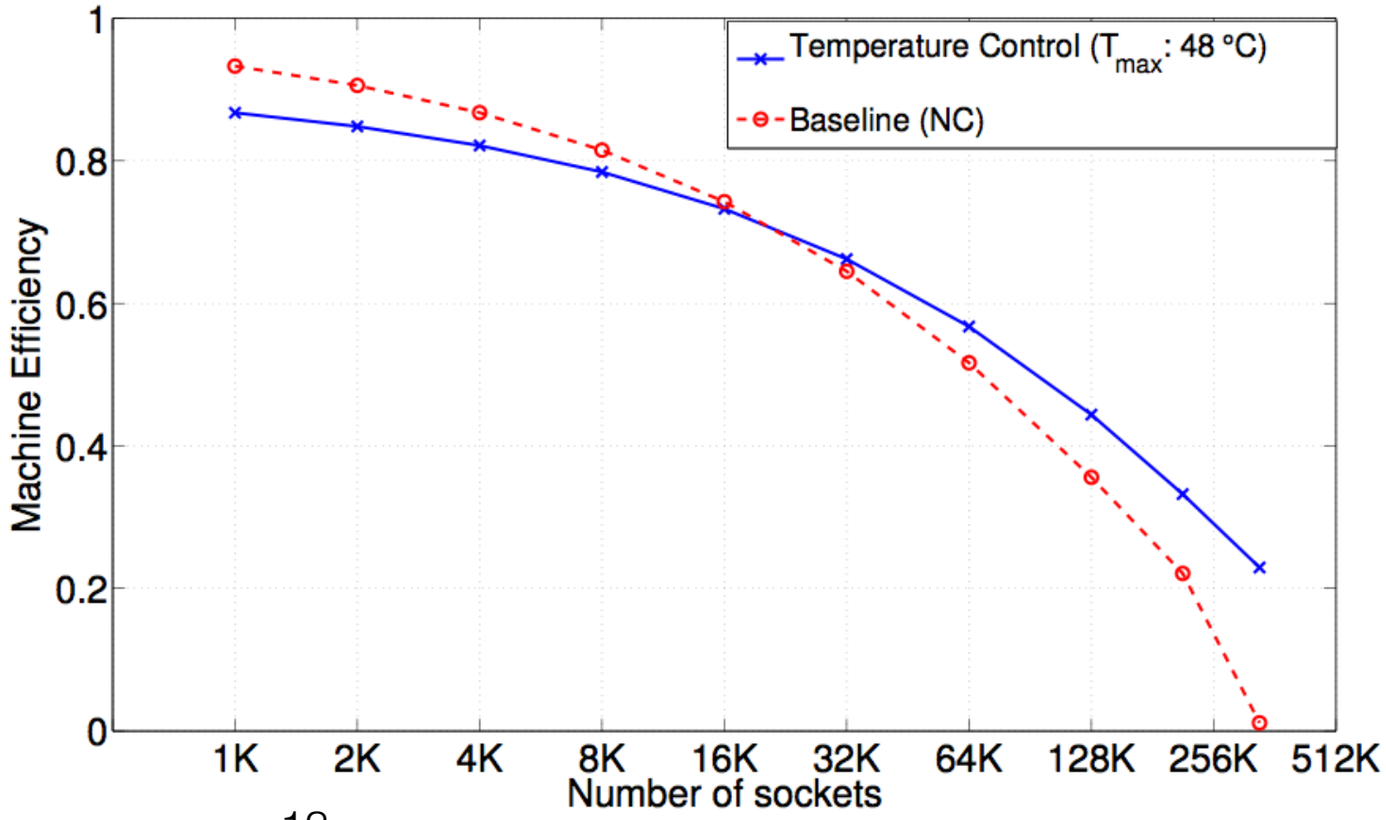
(c) Jacobi2D

Reduction in time calculated compared to baseline case with no temperature control

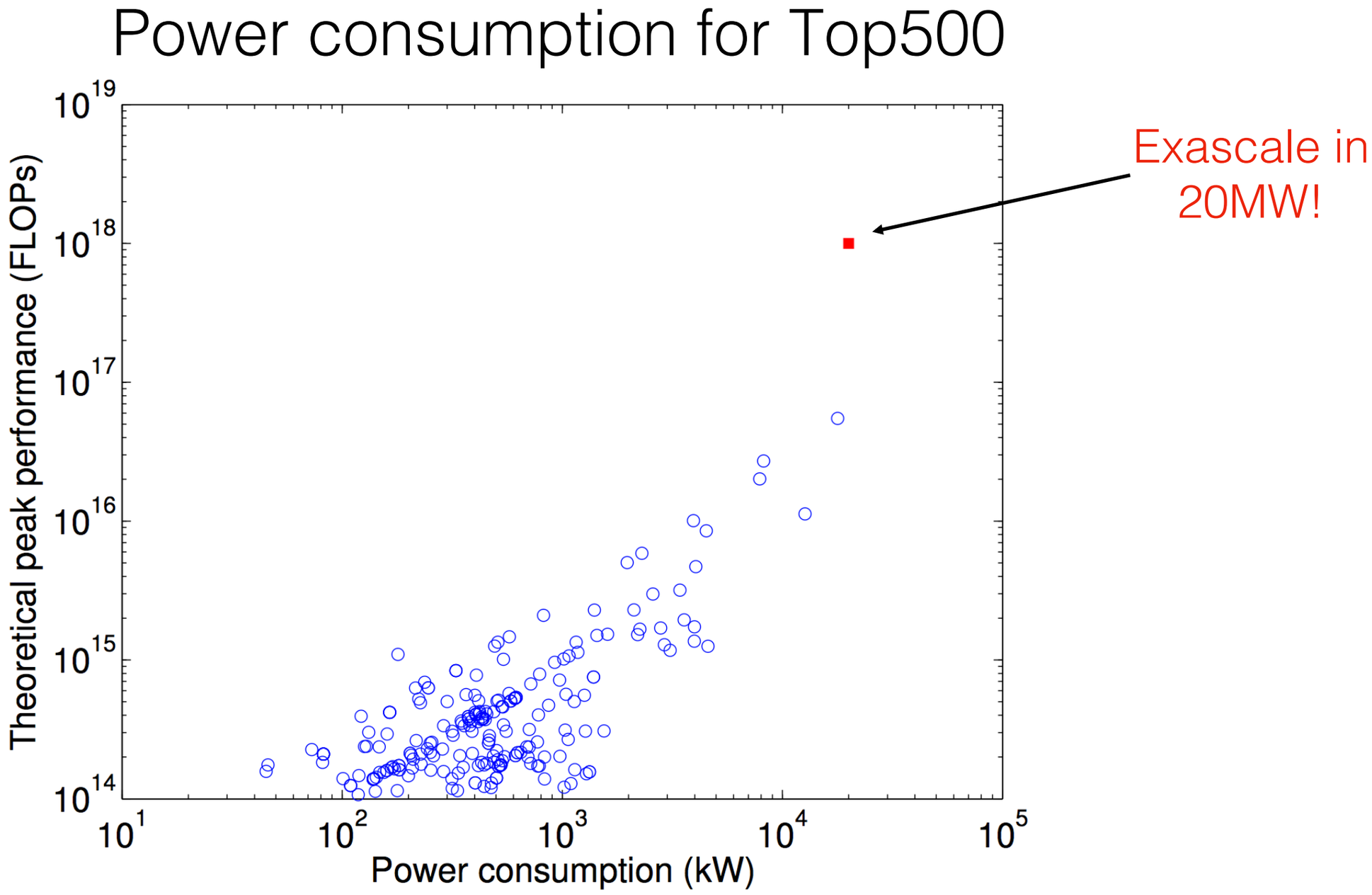
Improvement in Machine Efficiency

- Our scheme improves utilization beyond 20K sockets compared to baseline
- For 340K socket machine:
  - Baseline: Efficiency < 1% (un operational)
  - Our scheme: Efficiency ~ 21%

Machine Efficiency: Ratio of time spent doing useful work when running a single application



What's the Problem?



Data Center Power

How is data center power need calculated?

- using Thermal Design Power (TDP) of nodes

However, TDP is hardly reached!!

Solution

- Constrain power consumption of nodes
- Overprovisioning* - Use more nodes than conventional data center for the same power budget

* Patki et.al. Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing (ICS 13)

Constraining CPU/Memory Power

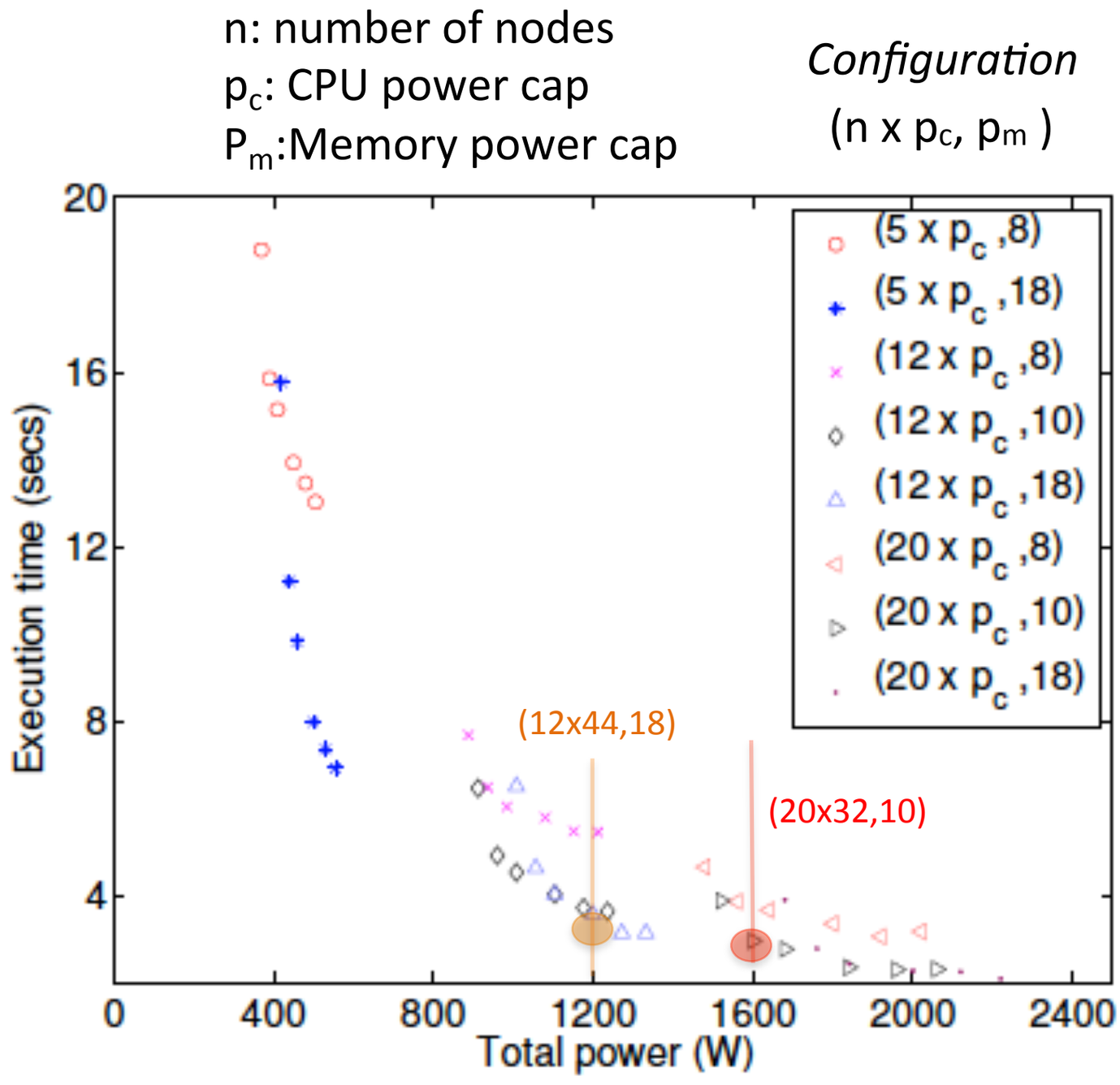
Intel Sandy Bridge

- Running Average Power Limit (RAPL) library
  - measure and set CPU/memory power

Application Performance with Power

- Application performance does not improve proportionately with increase in power cap

- Better to run on larger number of nodes each capped at lower power levels



Performance of LULESH at different configurations

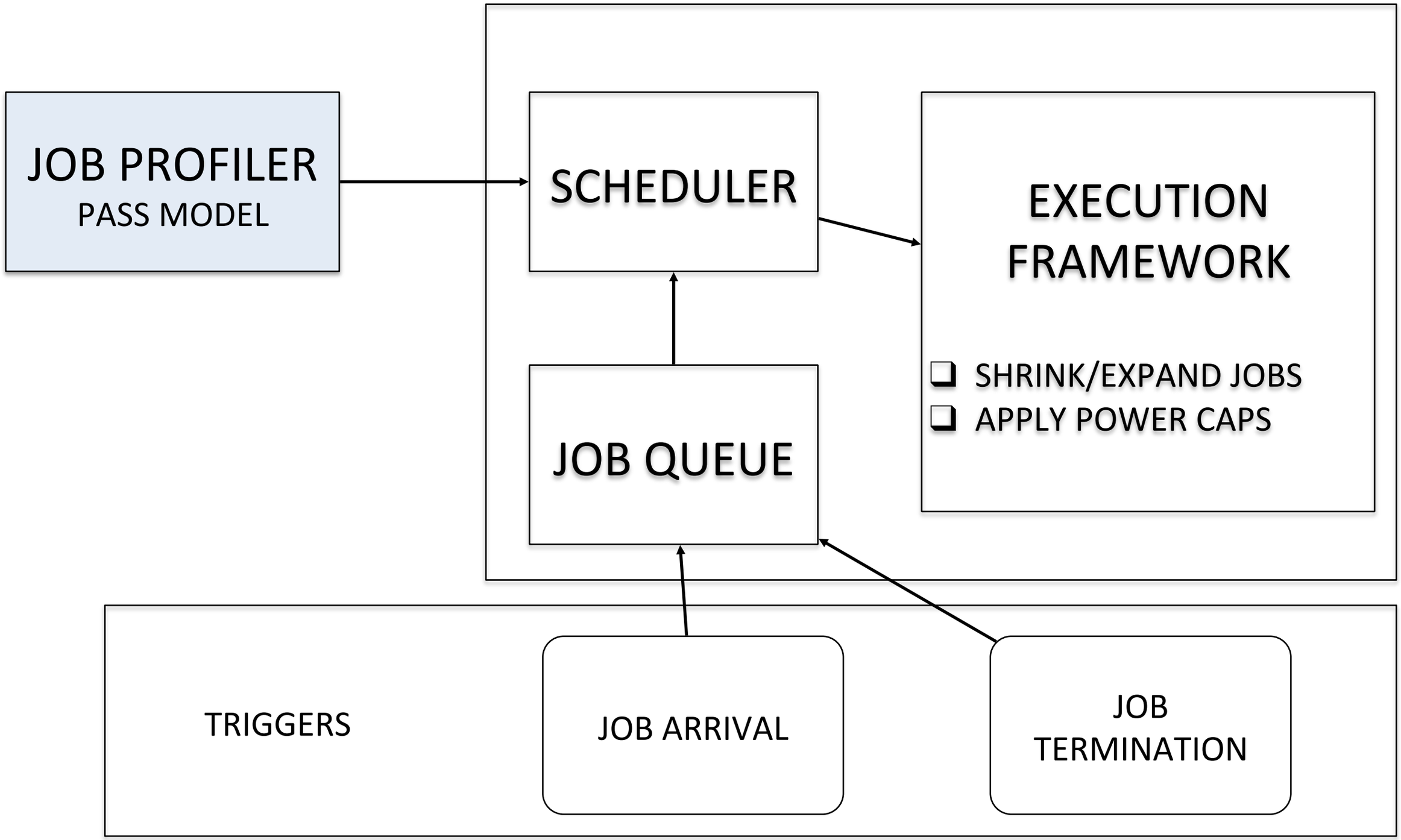
Problem Statement

Maximizing Data Center Performance Under Strict Power Budget

Data center capabilities and job features

- Power capping ability
- Overprovisioning
- Job moldability (Optional)
- Job malleability (Optional)
  - Charm++
  - Dynamic MPI

Power Aware Resource Manager (PARM)



JOB PROFILER

- Measure job performance at various scales and CPU power levels

- Power Aware Strong Scaling (PASS) Model
  - Predict job performance at any (n, p)
    - n: number of nodes
    - p: CPU power level

Power Aware Strong Scaling (PASS) Model*

Time vs Scale

Downey's strong scaling

$$t = F(n, A, \sigma)$$

- n: number of nodes
- A: Average Parallelism
- σ : duration of parallelism A

Time vs Frequency

$$t(f) = \begin{cases} \frac{W_{cpu}}{f} + T_{mem}, & \text{for } f < f_h \\ T_h, & \text{for } f \geq f_h \end{cases}$$

- W_{cpu}: CPU work
- T_{mem}: memory work
- T_h : minimum exec time

Frequency vs Power

- p_{core}: core power
- g_i: cost level I cache access
- L_i: #level I accesses
- g_m: cost of mem access
- M: #mem accesses
- p_{base}: idle power

$$p = p_{core} + \sum_{i=1}^{3} g_i L_i + g_m M + p_{base}$$

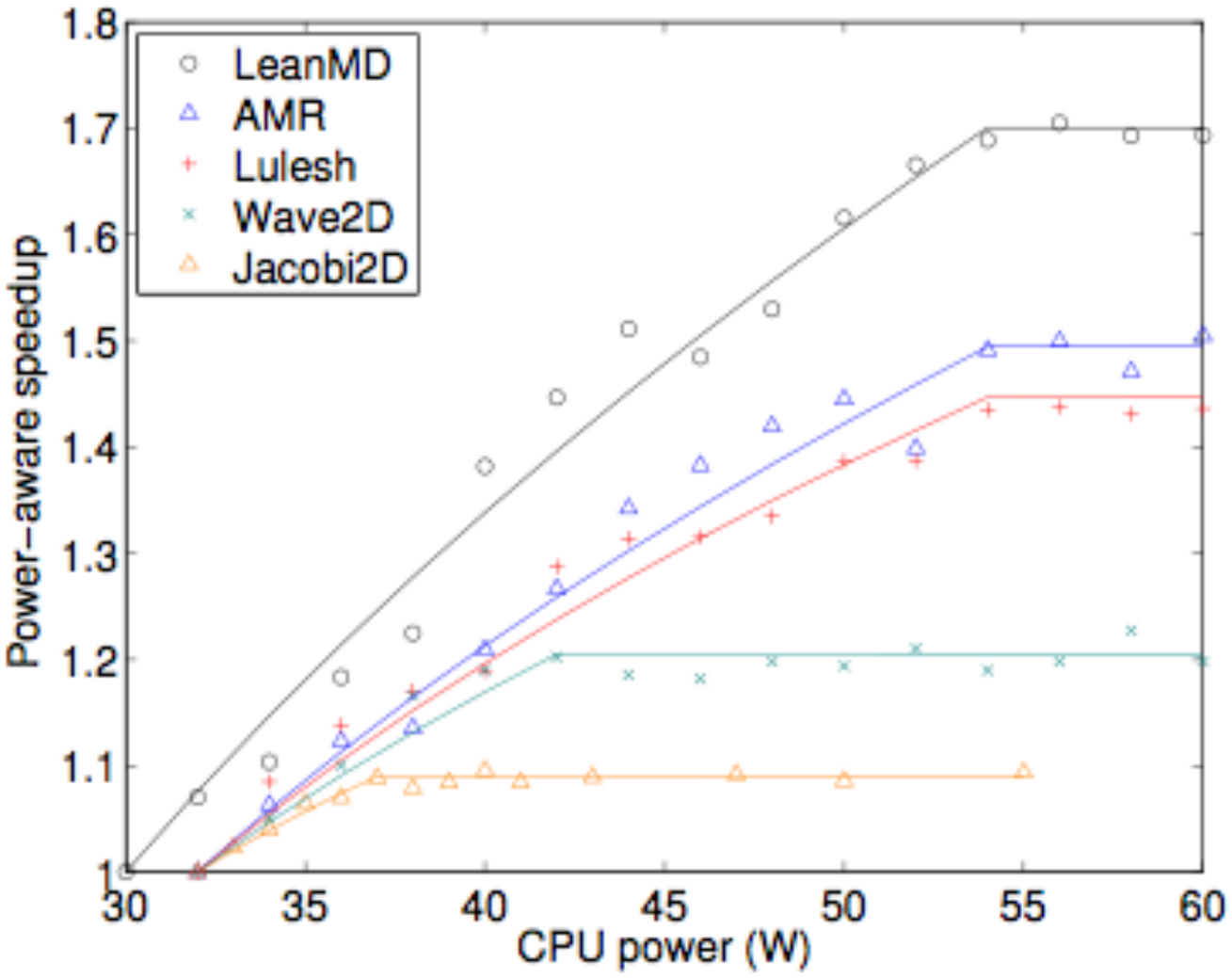
Time as a function of power and number of nodes

*O. Sarood, A. Langer, A. Gupta, L. Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. SC'14

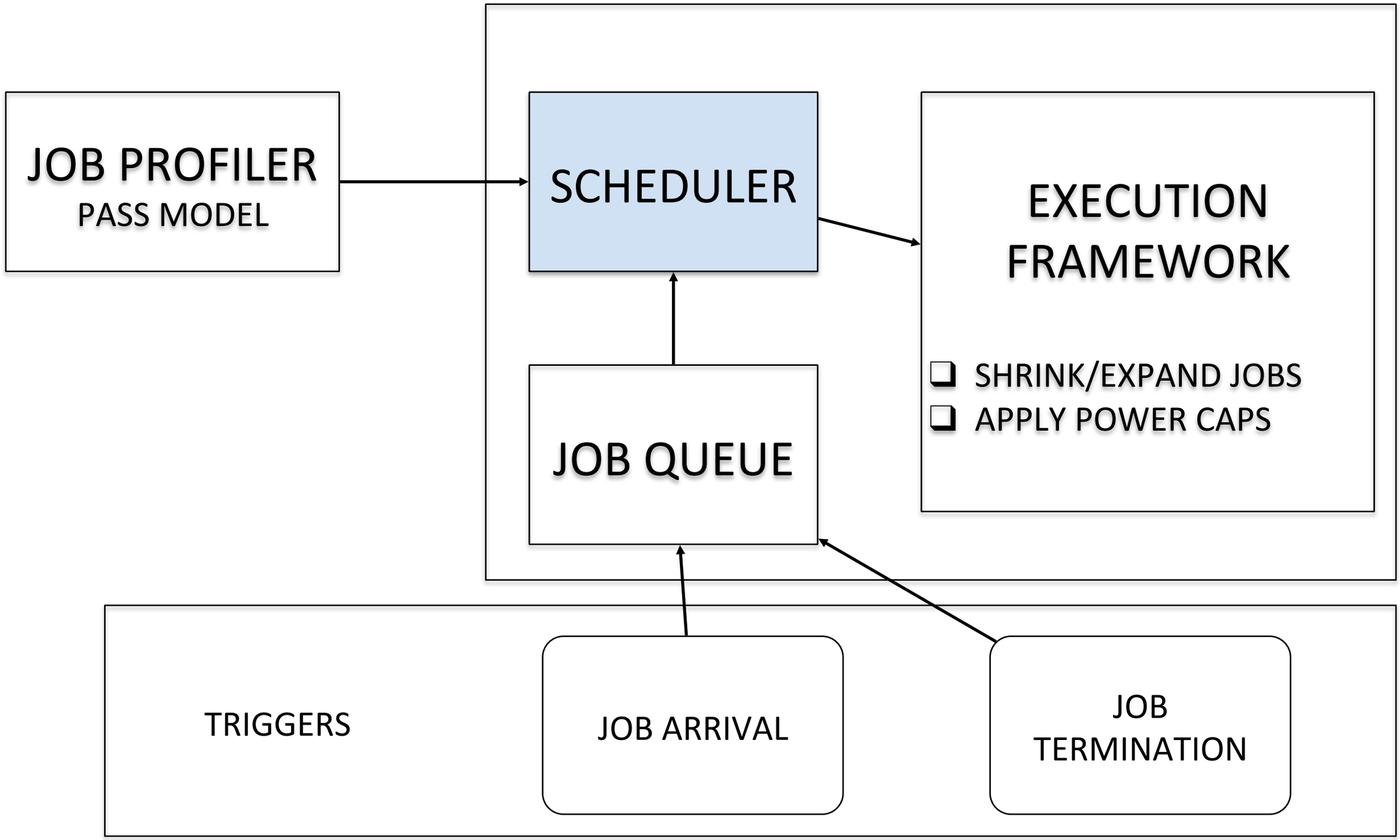
Estimating Performance using PASS

Model Parameters

Application	a	b	p_l	p_h	β
LeanMD	1.65	7.74	30	54	0.40
AMR	2.45	6.57	32	54	0.33
Lulesh	2.63	8.36	32	54	0.30
Wave2D	3.00	10.23	32	42	0.16
Jacobi2D	1.54	10.13	32	37	0.08



Power Aware Resource Manager (PARM)



Scheduler: Integer Linear Program Formulation

Objective Function

$$\sum_{j \in \mathcal{J}} \sum_{n \in N_j} \sum_{p \in P_j} w_j * s_{j,n,p} * x_{j,n,p}$$

Select One Resource Combination Per Job

$$\sum_{n \in N_j} \sum_{p \in P_j} x_{j,n,p} \leq 1 \qquad \forall j \in I$$

$$\sum_{n \in N_j} \sum_{p \in P_j} x_{j,n,p} = 1 \qquad \forall j \in \mathcal{I}$$

Bounding total nodes

$$\sum_{j \in \mathcal{J}} \sum_{p \in P_j} \sum_{n \in N_j} n x_{j,n,p} \leq \mathbf{N}$$

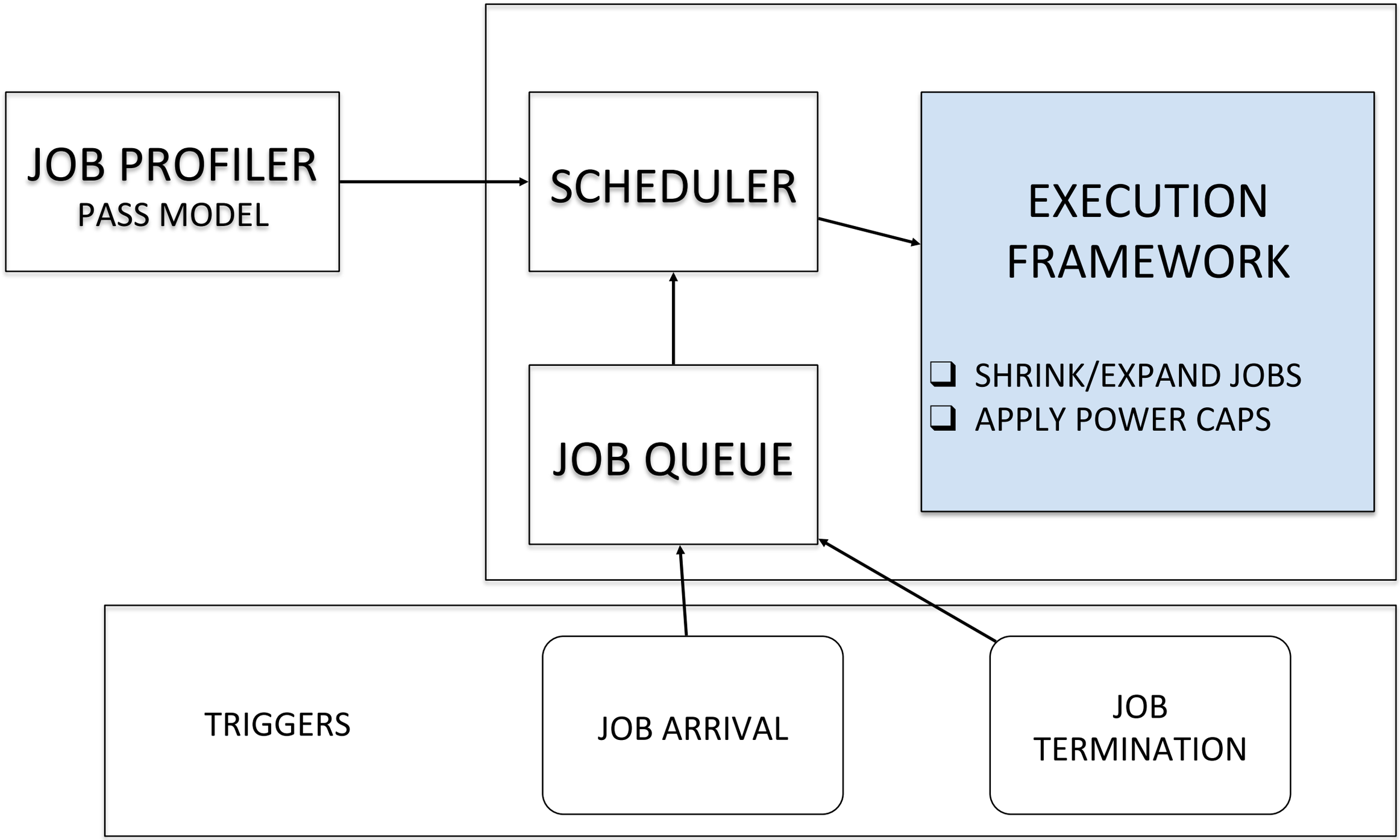
Bounding power consumption

$$\sum_{j \in \mathcal{J}} \sum_{n \in N_j} \sum_{p \in P_j} (n * (p + W_{base})) x_{j,n,p} \leq W_{max}$$

Disable Malleability (Optional)

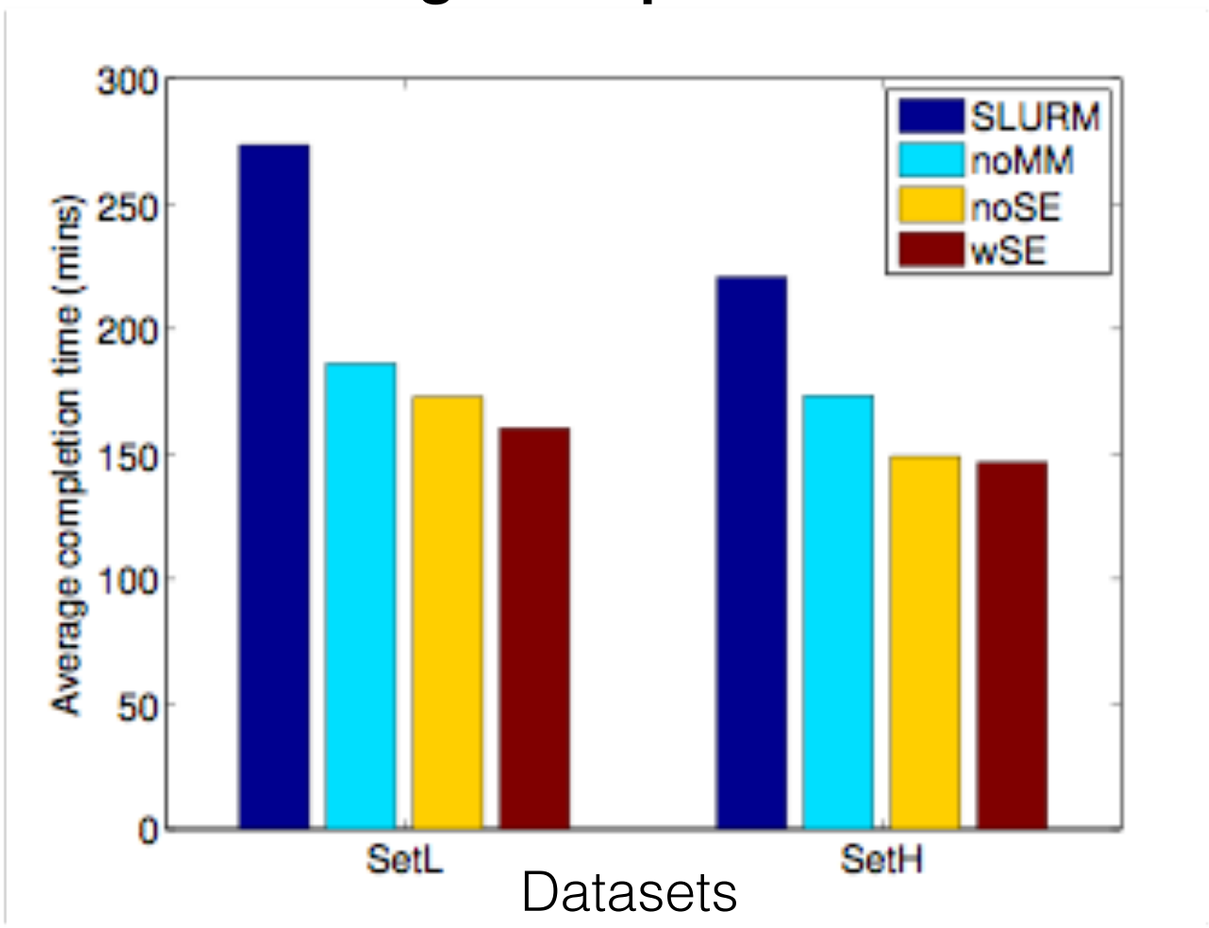
$$\sum_{n \in N_j} \sum_{p \in P_j} n x_{j,n,p} = n_j \qquad \forall j \in \mathcal{I}$$

Power Aware Resource Manager (PARM)



PARM Performance Results

Average Completion times



Description

- ***noMM***: without Malleability and Moldability
- ***noSE***: with Moldability but no Malleability
- ***wSE***: with Moldability and Malleability

Performance

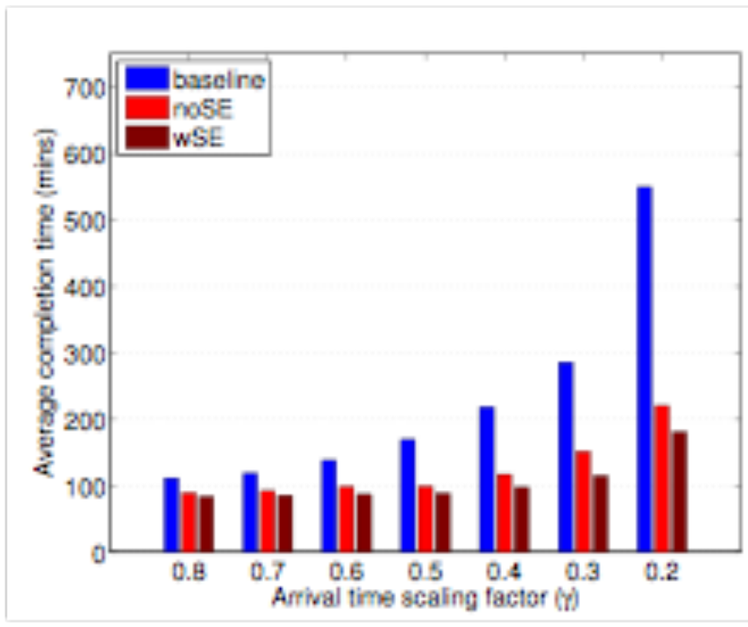
- 32% improvement with nMM over SLURM
- 13.9% improvement with noSE over noMM
- 7.5% improvement with wSE over noSE
- 1.7X improvement in throughput

Malleability: changing number of nodes at runtime
Moldability: assigning number of nodes from within a range (at schedule time)

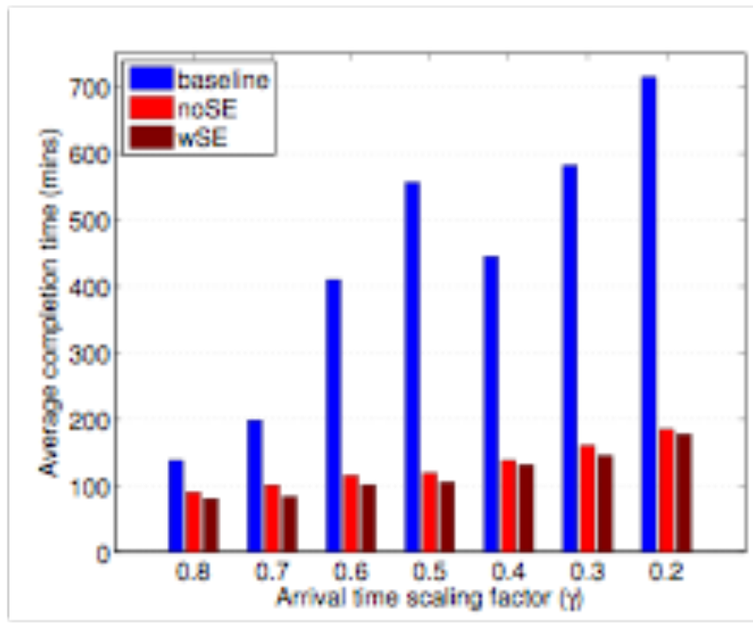
Large Scale Projections Performance

5.2X speedup with wSE using job logs from Intrepid!*

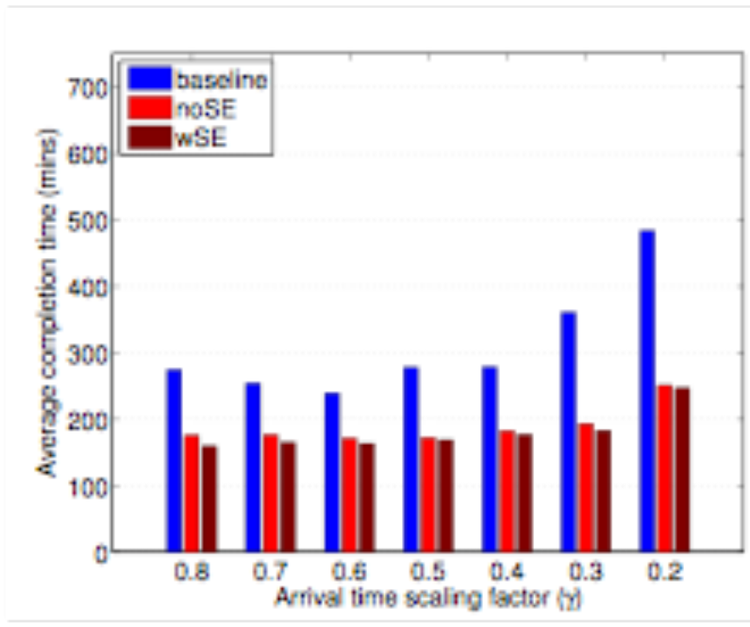
- ***baseline***: SLURM scheduling
- ***noSE***: with Moldability but no Malleability
- ***wSE***: with Moldability and Malleability



(a) Set1



(b) Set2

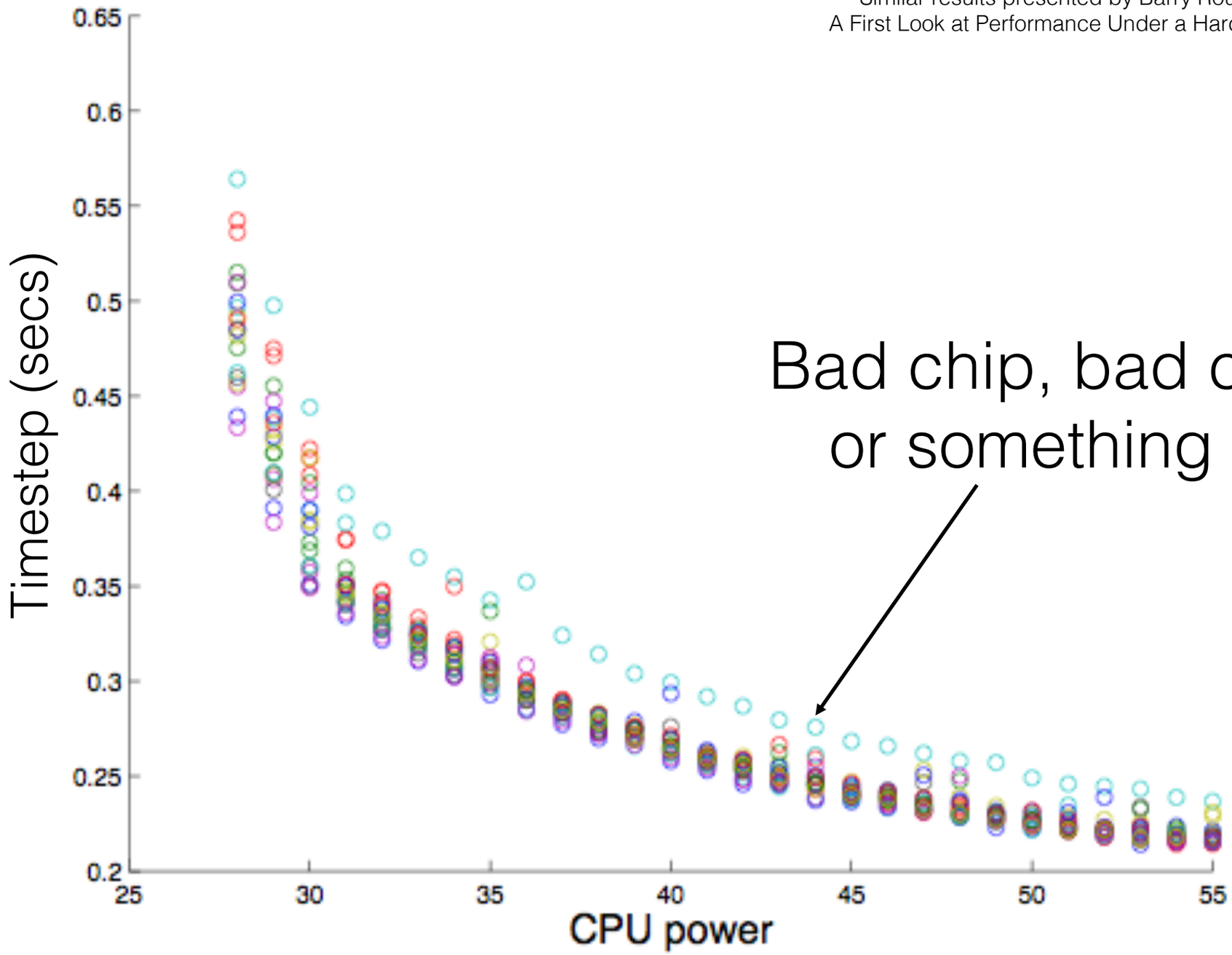


(c) Set3

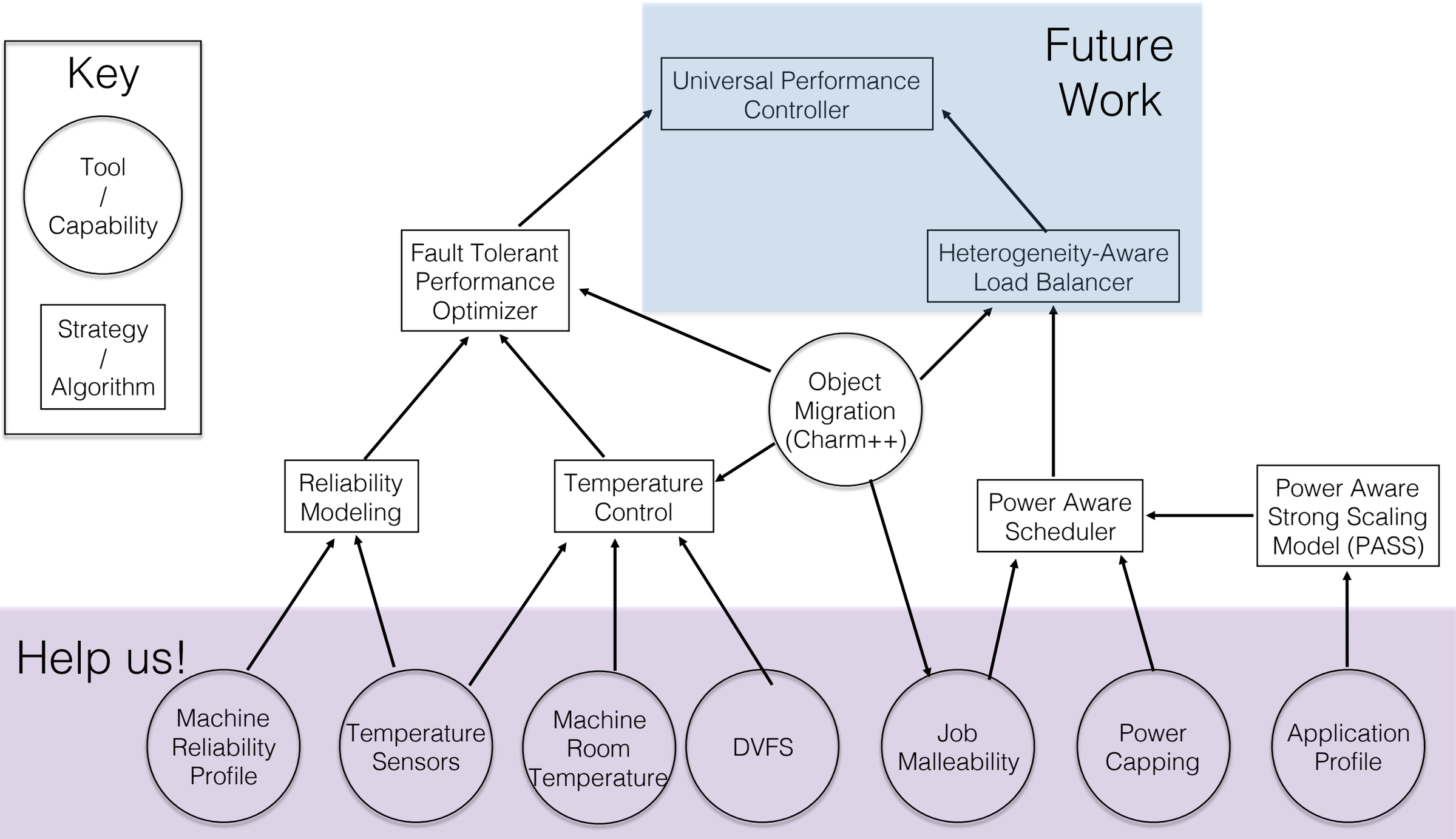
*To get diversity in job arrival rates, we multiplied job arrival times by γ

Heterogeneity in homogenous CPUs!

*Similar results presented by Barry Rountree earlier (Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound)

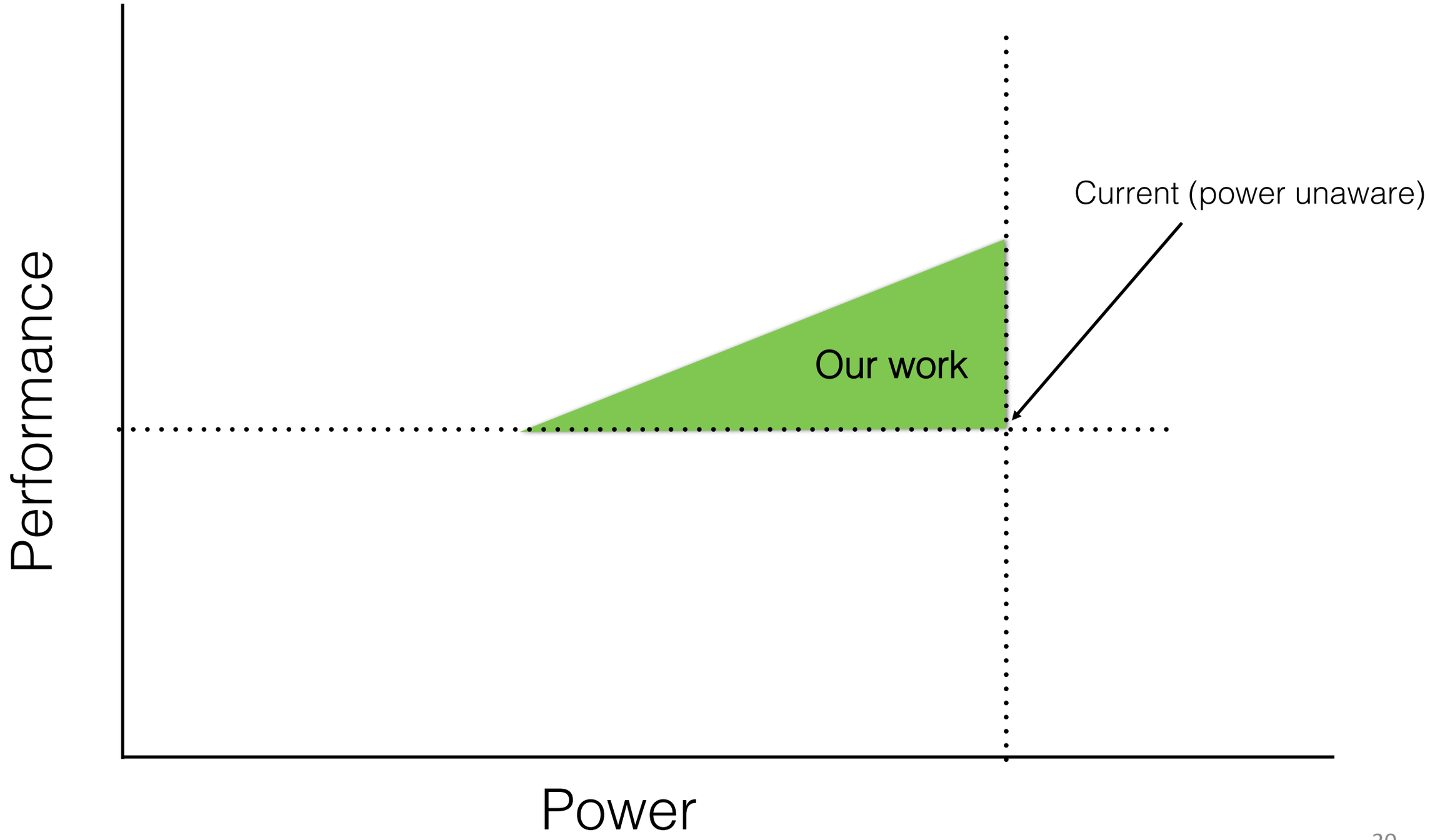


Exascale: Power, Thermal and Reliability Perspective



Power, Performance and our work

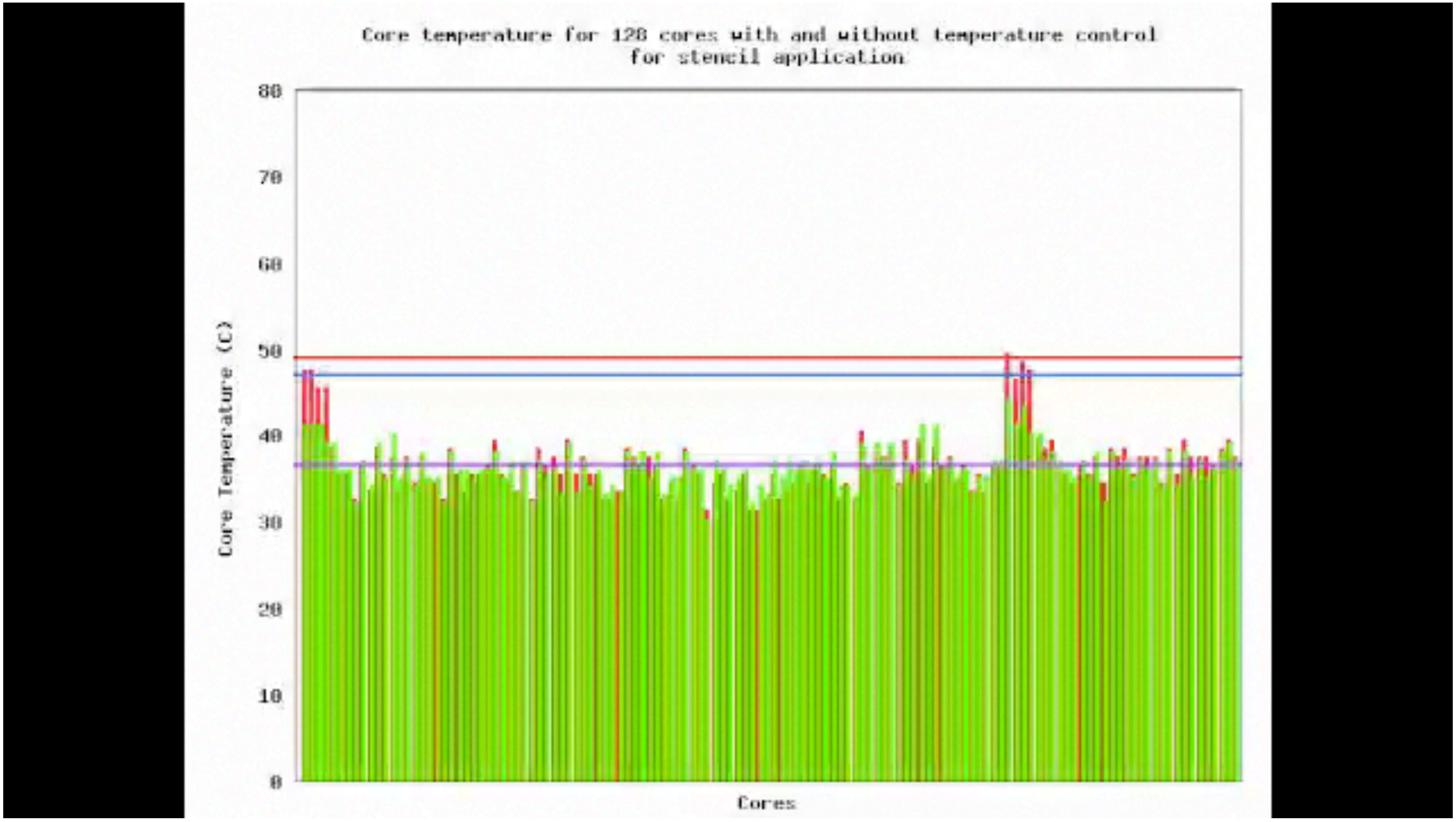
*Inspired by Prof. D.K. Panda's talk from MODSIM 2014



Publications (related)

- Osman Sarood, Akhil Langer, Abhishek Gupta, Laxmikant Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. SC'14.
- Ehsan Totoni, Joseph Torellas, Laxmikant Kale. Using an Adaptive HPC Runtime System to Reconfigure the Cache Hierarchy. SC'14.
- Esteban Meneses, Osman Sarood, and Laxmikant V. Kale. Energy Profile of Rollback-Recovery Strategies in High Performance Computing. Elsevier - Parallel Computing (PARCO 2014).
- Osman Sarood, Esteban Meneses, and Laxmikant V. Kale. A `Cool' Way of Improving the Reliability of HPC Machines. Supercomputing'13 (SC'13).
- Osman Sarood, Akhil Langer, Laxmikant V. Kale, Barry Rountree, and Bronis de Supinski. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. IEEE Cluster 2013.
- Harshitha Menon, Bilge Acun, Simon Garcia de Gonzalo, Osman Sarood, and Laxmikant V. Kale. Thermal Aware Automated Load Balancing for HPC Applications. IEEE Cluster.
- Esteban Meneses, Osman Sarood and Laxmikant V. Kale. Assessing Energy Efficiency of Fault Tolerance Protocols for HPC Systems. IEEE SBAC-PAD 2012. Best Paper Award.
- Osman Sarood, Phil Miller, Ehsan Totoni, and Laxmikant V. Kale. `Cool' Load Balancing for High Performance Computing Data Centers. IEEE Transactions on Computers, December 2012.
- Osman Sarood and Laxmikant V. Kale. Efficient `Cool Down' of Parallel Applications. PASA 2012.
- Osman Sarood, and Laxmikant V. Kale. A `Cool' Load Balancer for Parallel Applications. Supercomputing'11 (SC'11).
- Osman Sarood, Abhishek Gupta, and Laxmikant V. Kale. Temperature Aware Load Balancing for Parallel Application: Preliminary Work. HPPAC 2011.

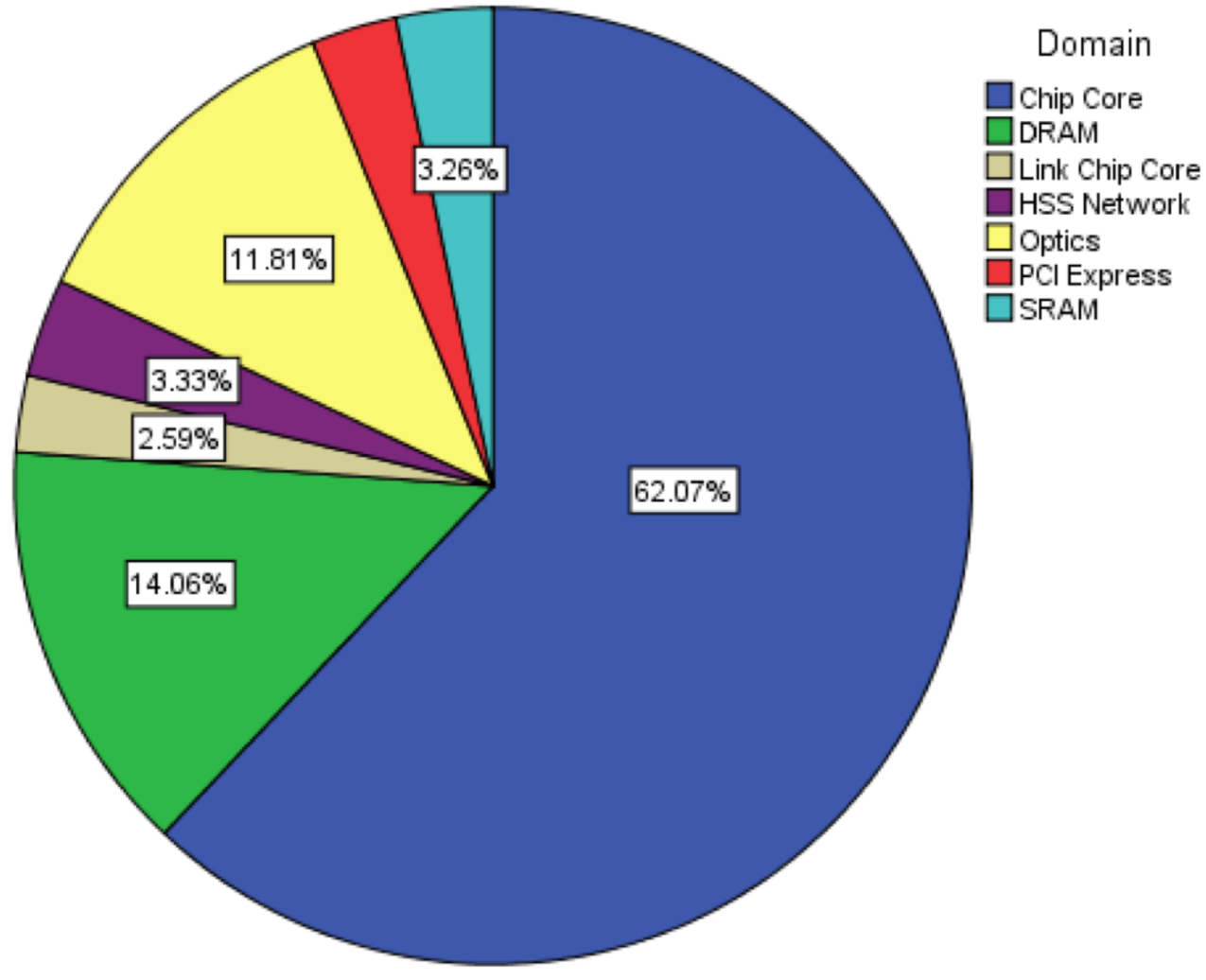
Thank You!


Core temperature for 128 cores with and without temperature control
for stencil application
Core Temperature (C)
80
70
60
50
40
30
20
10
0
Cores

Distribution of Node Power Consumption

Power distribution for BG/Q processor on Mira

- 76% by CPU/Memory
- No good mechanism for controlling other power domains



Pie Chart: Sean Wallace, Measuring Power Consumption on IBM Blue Gene/Q