Performance Optimization Under Thermal and Power Constraints For High Performance Computing Data Centers

Osman Sarood

PhD Final Defense
Department of Computer Science
December 3rd, 2013

PhD Thesis Committee

- Dr. Bronis de Supinski
- Prof. Tarek Abdelzaher
- Prof. Maria Garzaran
- Prof. Laxmikant Kale, Chair

Current Challenges

- Energy, power and reliability!
  - 235 billion kWh (2% of total US electricity consumption) in 2010
  - 20 MW target for exascale
  - MTBF of 35-40 minutes for exascale machine[1]

1. Peter Kogge, ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Agenda

- Applying thermal restraint to
  - Remove hot spots and reduce cooling energy consumption[1]
  - Improve reliability and hence performance[2]
- Operation under strict power budget
  - Optimizing a single application[2]
  - Maximizing throughput of the entire data center having multiple jobs[2]

1. Pre-Preliminary exam work
2. Post-Preliminary exam work

Thermal Restraint

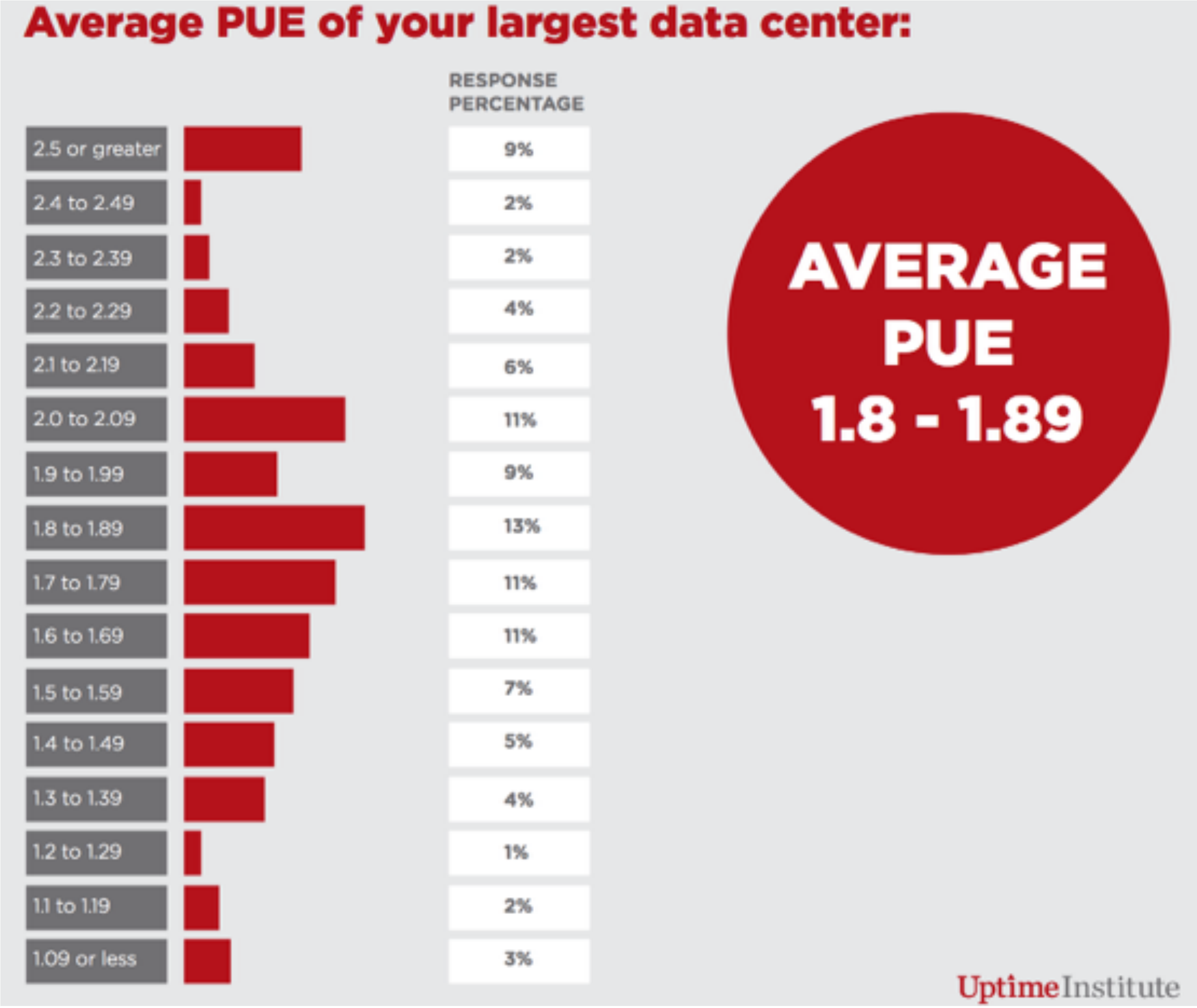
Reducing Cooling Energy Consumption

Publications

- **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.

Power Utilization Efficiency (PUE) in 2012

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}}$$



1. Matt Stansberry, Uptime Institute 2012 data center industry survey

PUEs for HPC Data Centers

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}}$$

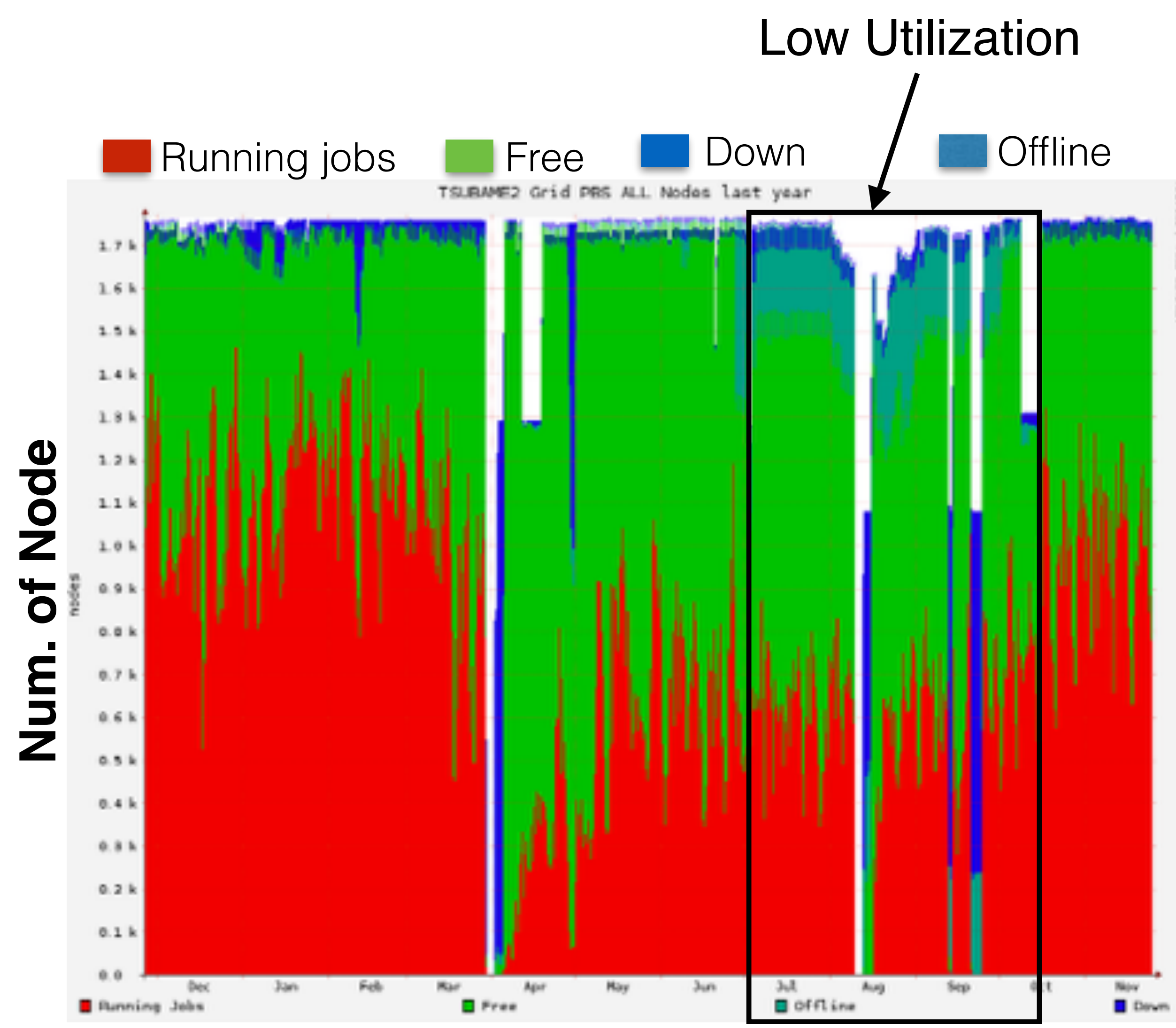
Supercomputer	PUE
Earth Simulator[1]	1.55
Tsubame2.0[2]	1.31/1.46
ASC Purple[1]	1.67
Jaguar[3]	1.58

- Most HPC data centers do not publish cooling costs
- PUE can change over time

1. Wu-chen Feng, The Green500 List: Encouraging Sustainable Supercomputing
2. Satoshi Matsuoka, Power and Energy Aware Computing with Tsubame 2.0 and Beyond
3. Chung-Hsing Hsu et. al., The Energy Efficiency of the Jaguar Supercomputer

Tsubame’s Cooling Costs

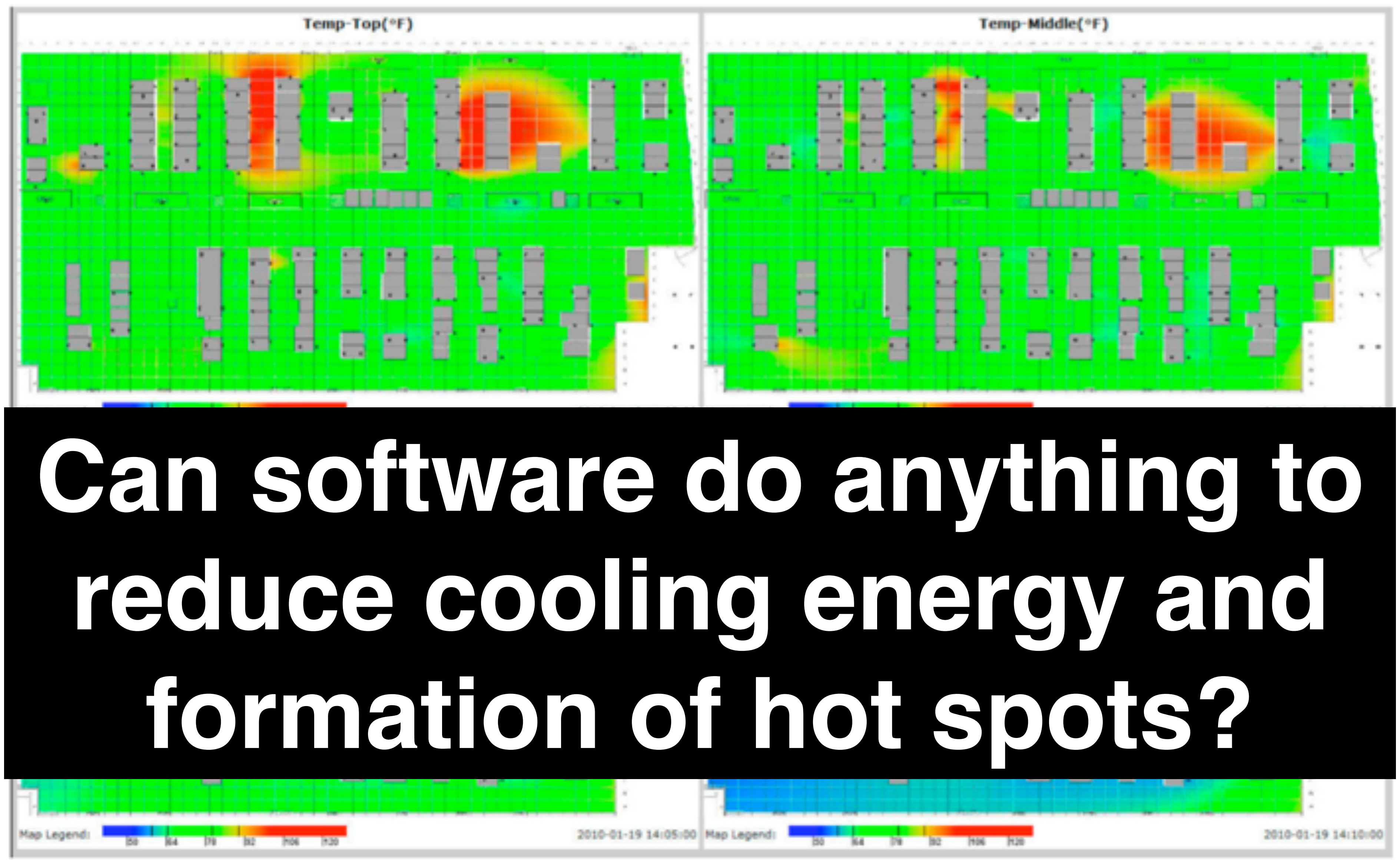
- Cooling costs generally depend:
  - On the environment (ambient temperature)
  - Machine utilization



Source: Tsubame2.0 monitoring portal, http://tsubame.gsic.titech.ac.jp/

Hot spots

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



Can software do anything to reduce cooling energy and formation of hot spots?

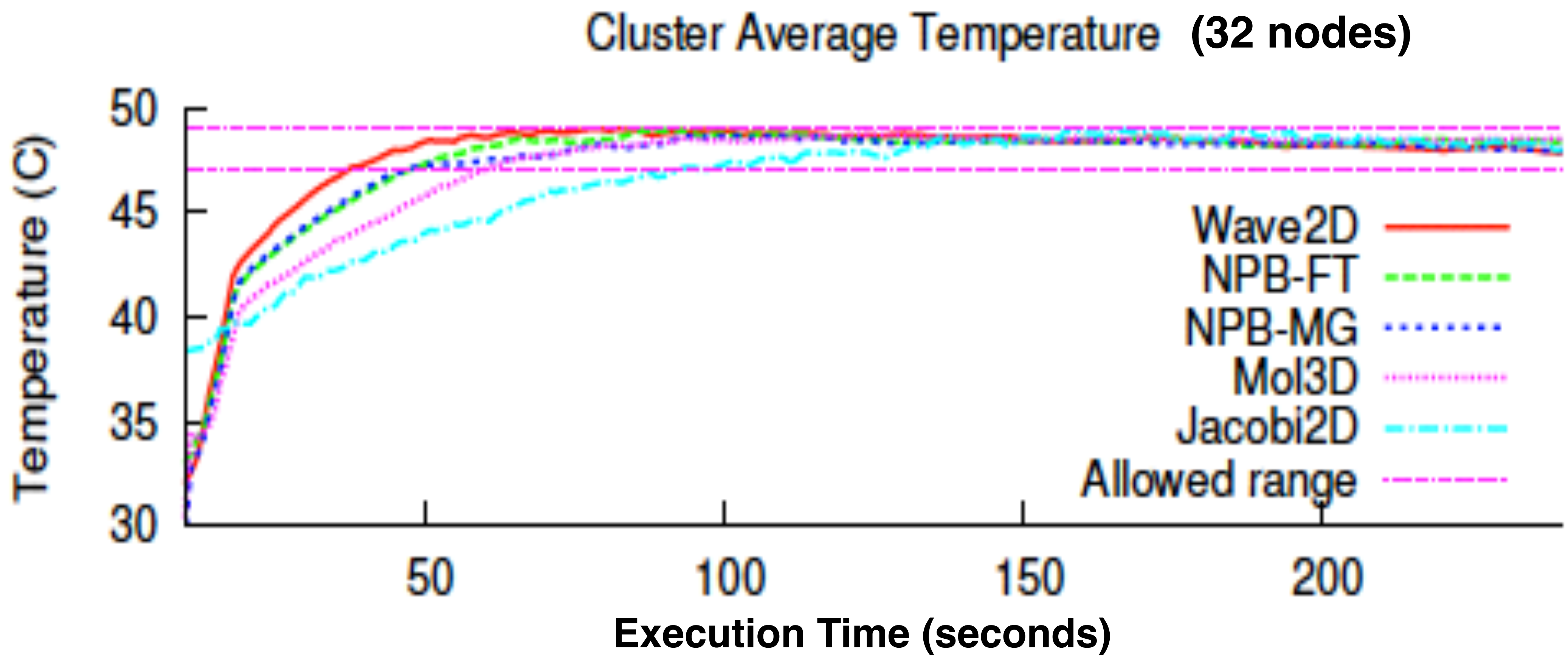
1. Dale Sartor, General Recommendations for High Performance Computing Data Center Energy Management Dashboard Display (IPDPSW 2013)

`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
- Details in dissertation

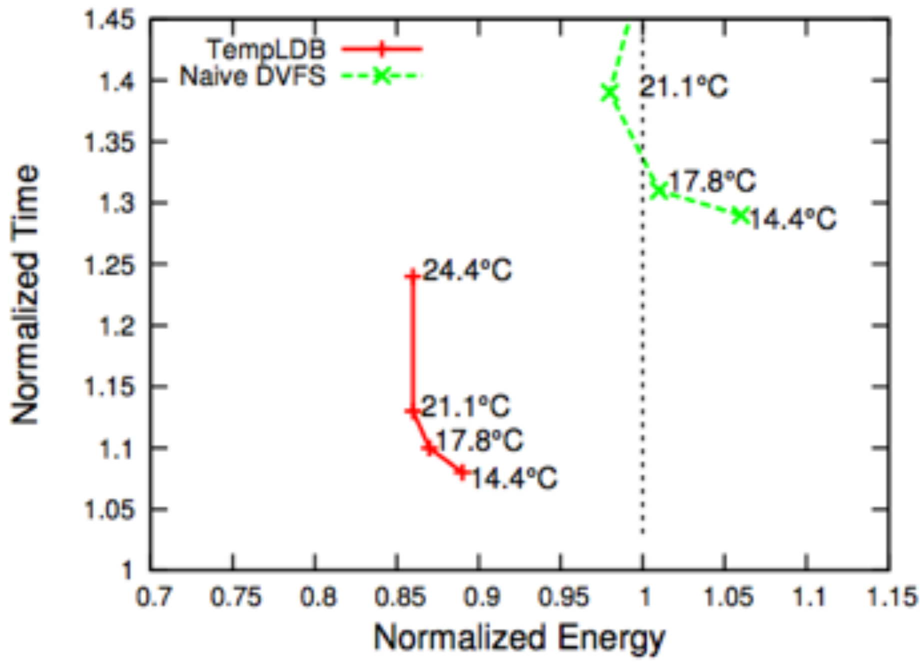
Average Core Temperatures in Check

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

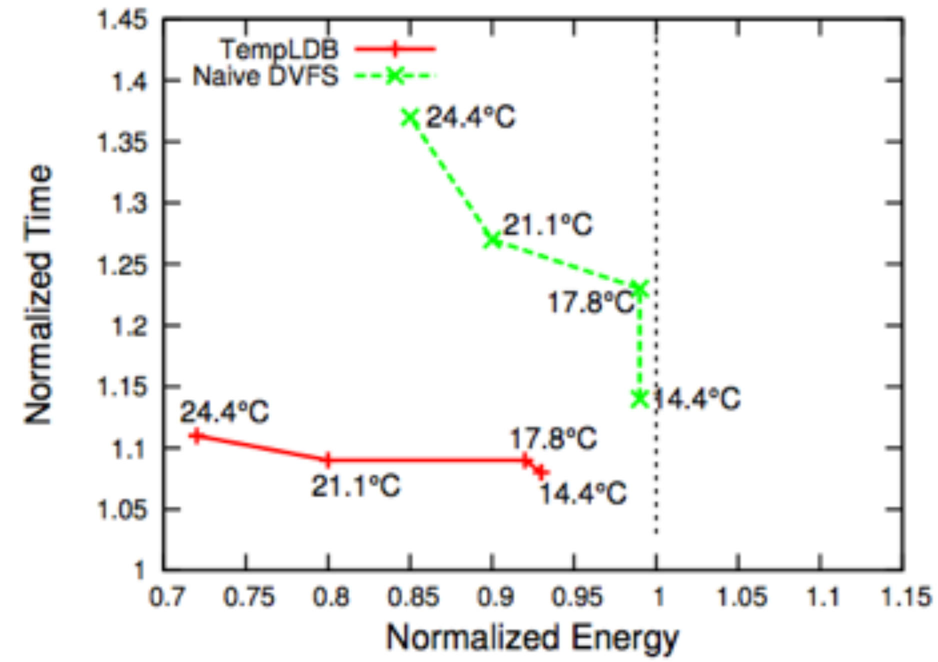


- Avg. core temperature within 2 C range
- Can handle applications having different temperature gradients

Benefits of `Cool' Load Balancer

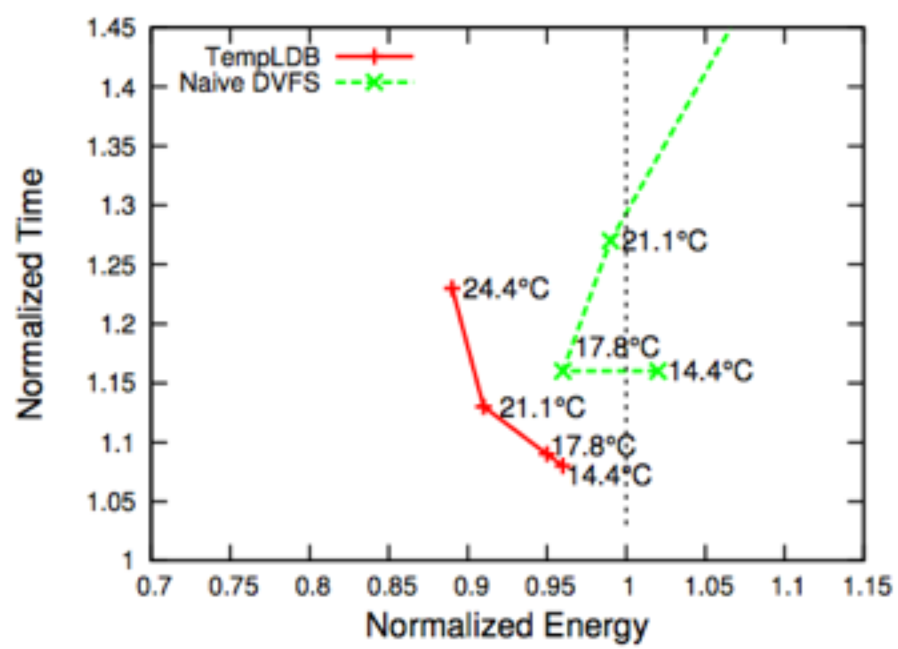


(a) Wave2D



(b) Mol3D

Normalization w.r.t run without temperature restraint



(c) NPB-MG

Thermal Restraint

Improving Reliability and Performance

Post-Preliminary Exam Work

Publications

- **Osman Sarood**, Esteban Meneses, and Laxmikant V. Kale. A `Cool' Way of Improving the Reliability of HPC Machines. Supercomputing'13 (SC'13).

Fault tolerance in present day supercomputers

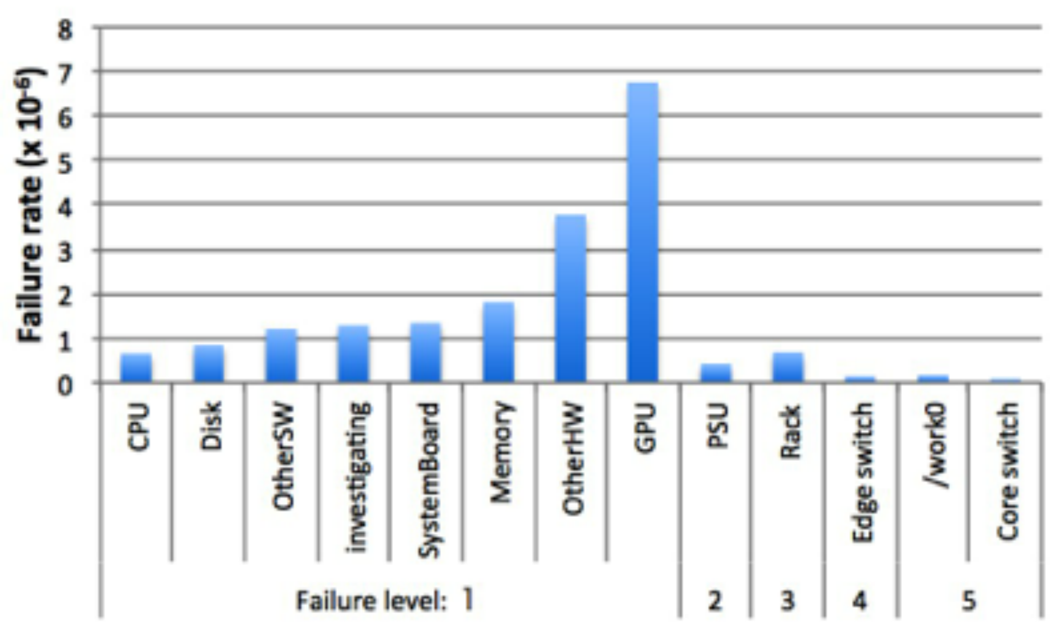
- Earlier studies point to per socket Mean Time Between Failures (MTBF) of 5 years - 50 years
- More than 20% of computing resources are wasted due to failures and recovery in a large HPC center[1]
- Exascale machine with 200,000 sockets is predicted to waste more than 89% time in failure/recovery[2]

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

Tsubame2.0 Failure Data[1]

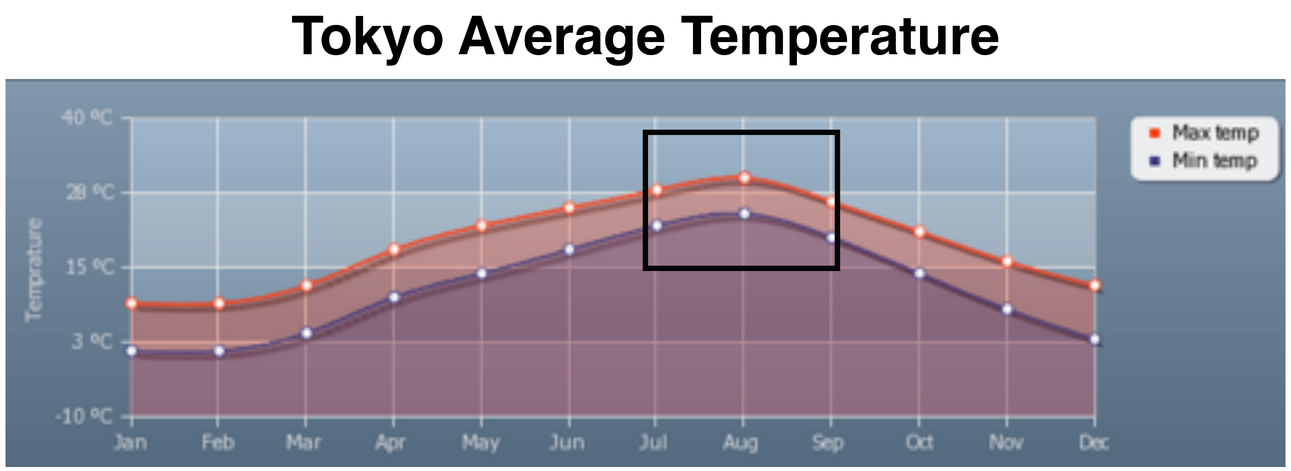
Component	MTBF
Core Switch	65.1 days
Rack	86.9 days
Edge Switch	17.4 days
PSU	28.9 days
Compute Node	15.8 hours

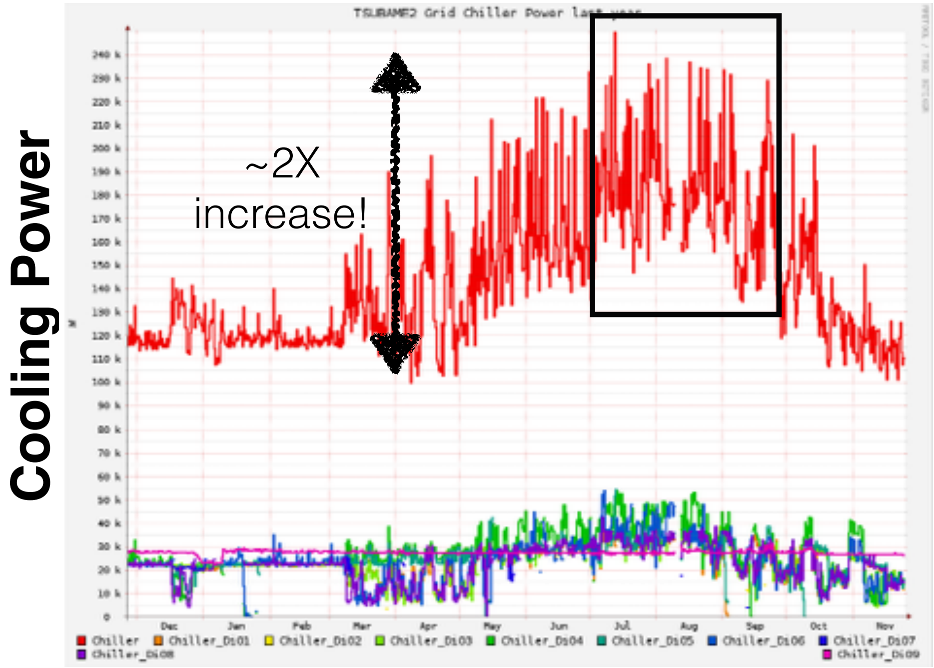
- Tsubame2.0 failure rates
  - Compute failures are much frequent
  - High failure rate due to increased temperatures

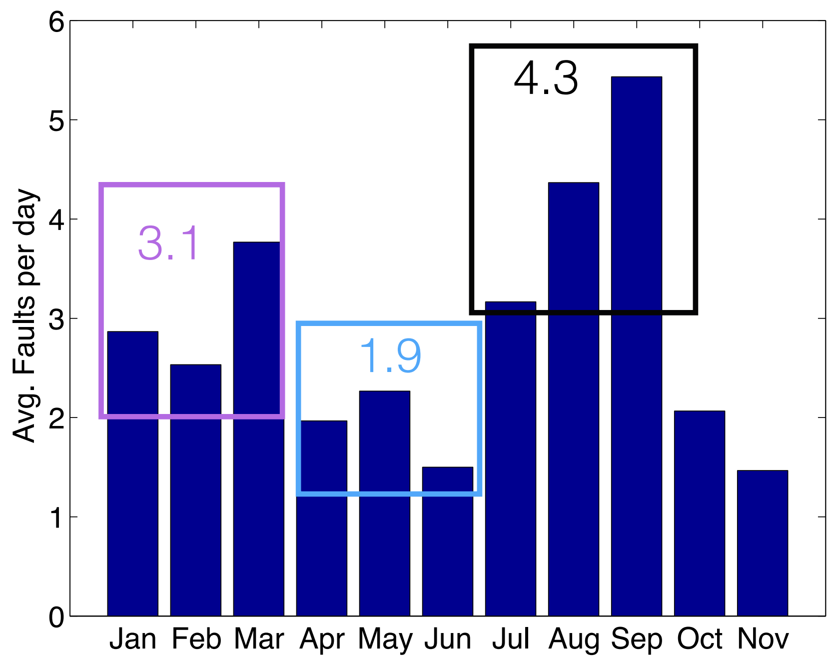


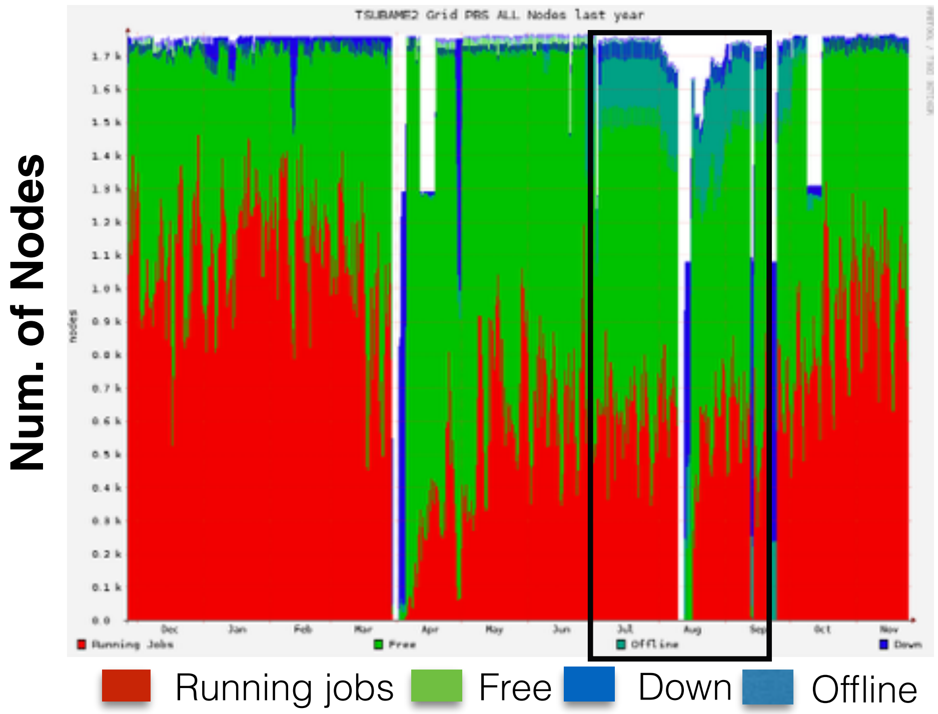
1. Kento Sato et. al., Design and Modeling of a Non-Blocking Checkpointing System, Supercomputing'12

Tsubame Fault Analysis









Source: Tsubame2.0 monitoring portal, http://tsubame.gsic.titech.ac.jp/

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:

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

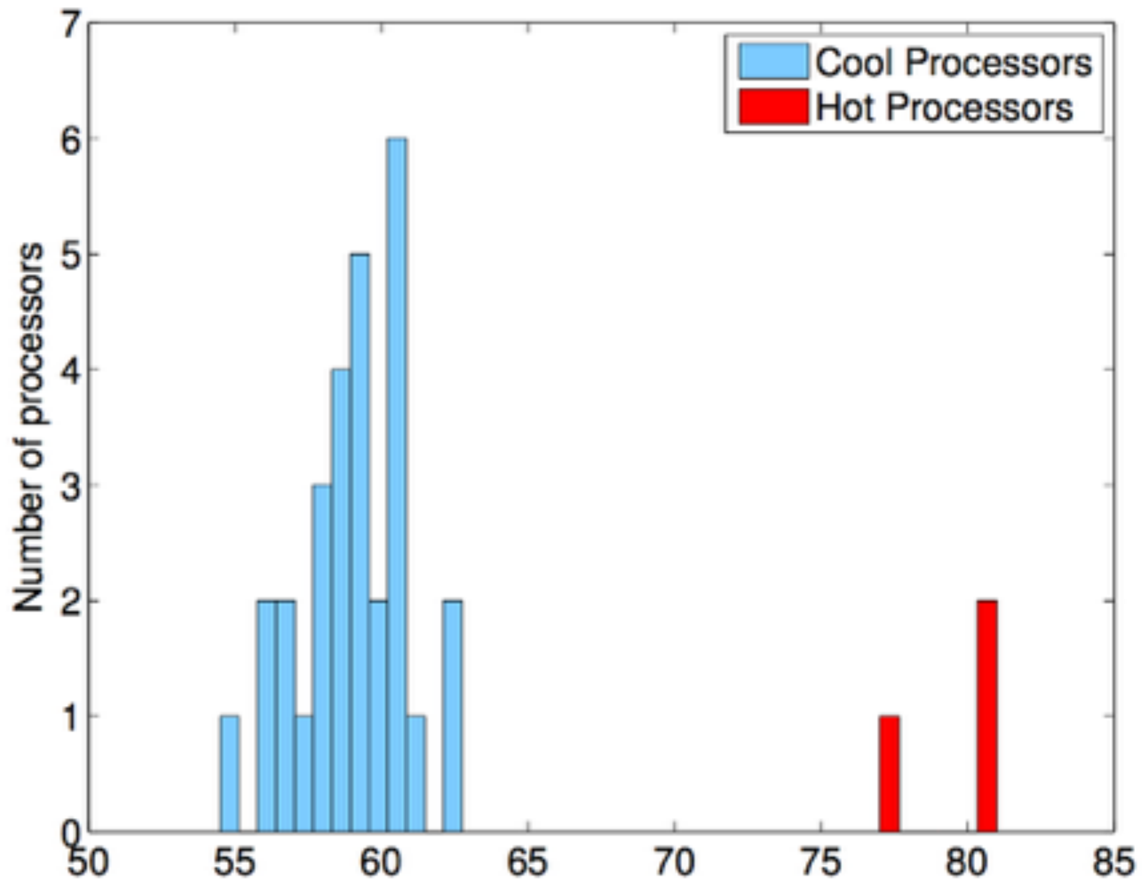
1. Wu-Chun Feng, Making a Case for Efficient Supercomputing, New York, NY, USA

Related Work

- Most earlier 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 any fault tolerance protocol

Distribution of Processor Temperature

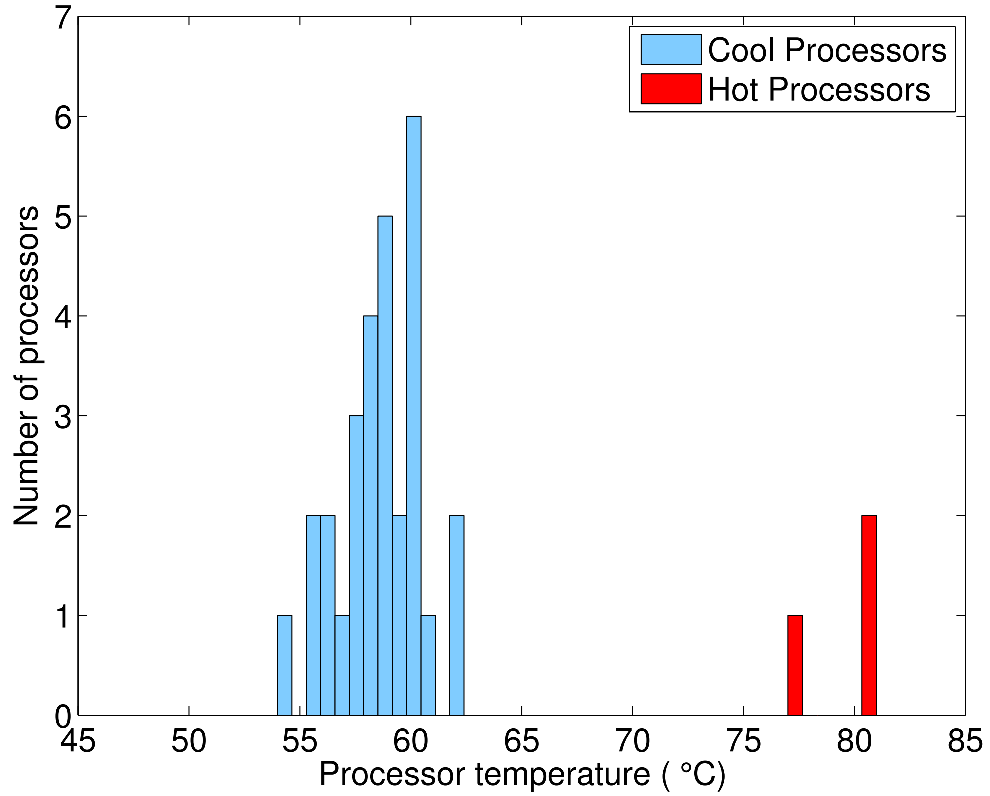
- 5-point stencil application (Wave2D from Charm++ suite)
- 32 nodes of our Energy Cluster[1]
- Cool processor mean: 59C, std deviation: 2.17C

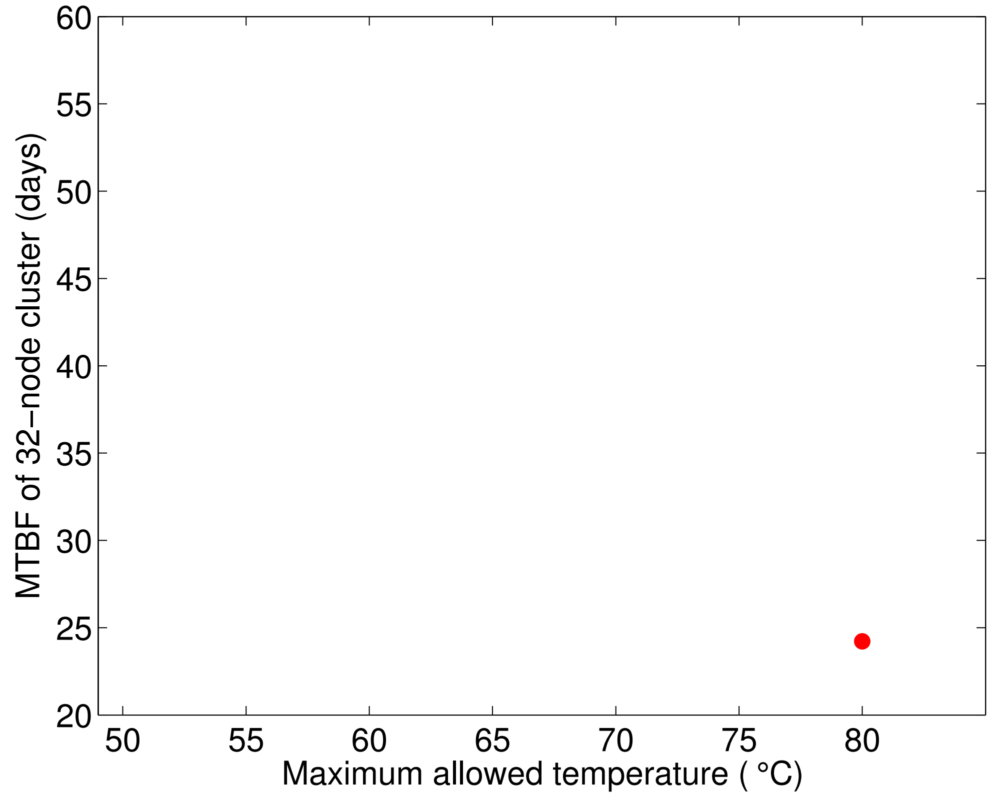


1. Thanks to Prof. Tarek Abdelzaher for allowing us to use the Energy Cluster

Estimated MTBF - No Temperature Restraint

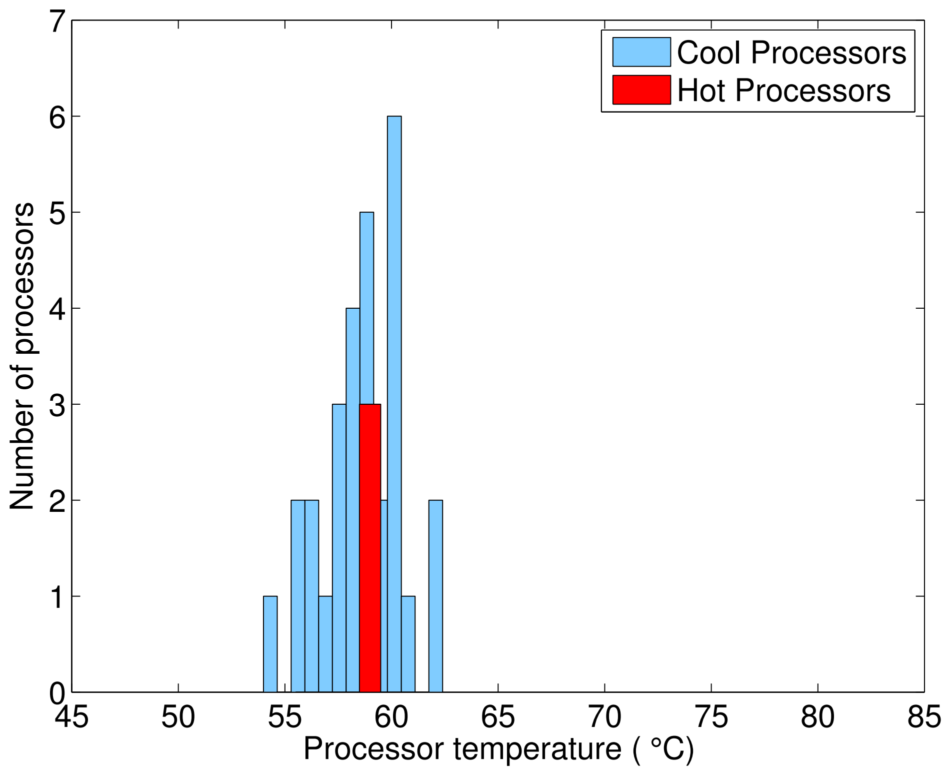
- Using observed max temperature data and per-socket MTBF of 10 years (cool processor mean: 59C, std deviation: 2.17C)
- Formula for M: $m = 160 * e^{-0.069T}$ $M = \frac{1}{\sum_{n=1}^{N} \frac{1}{m_n}}$

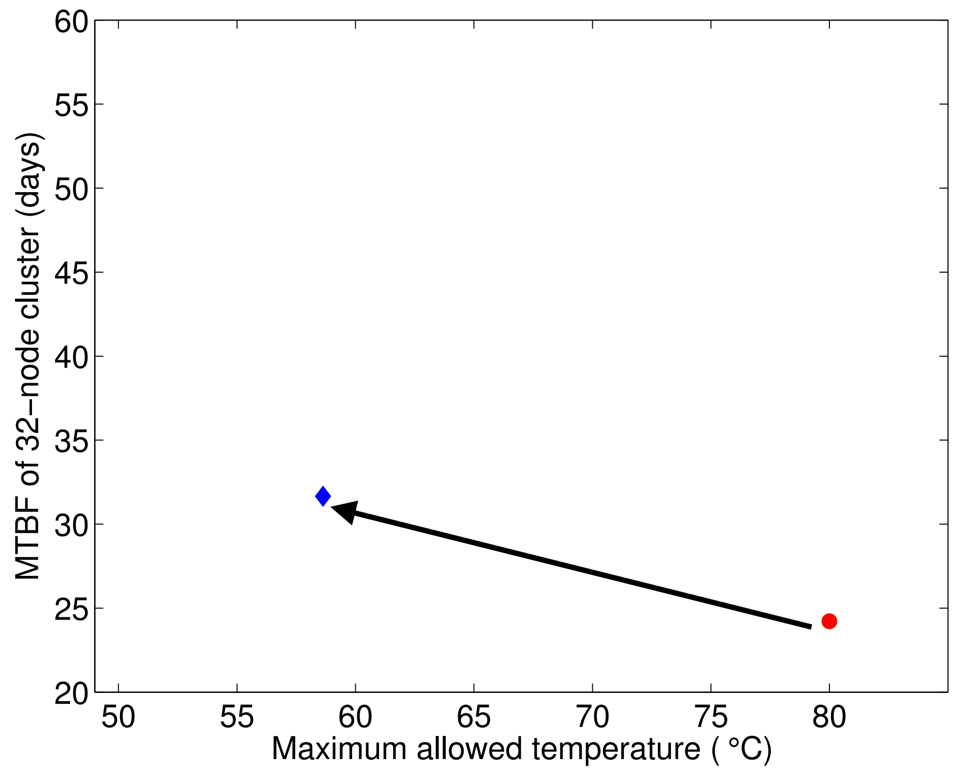




Estimated MTBF - Removing Hot Spot

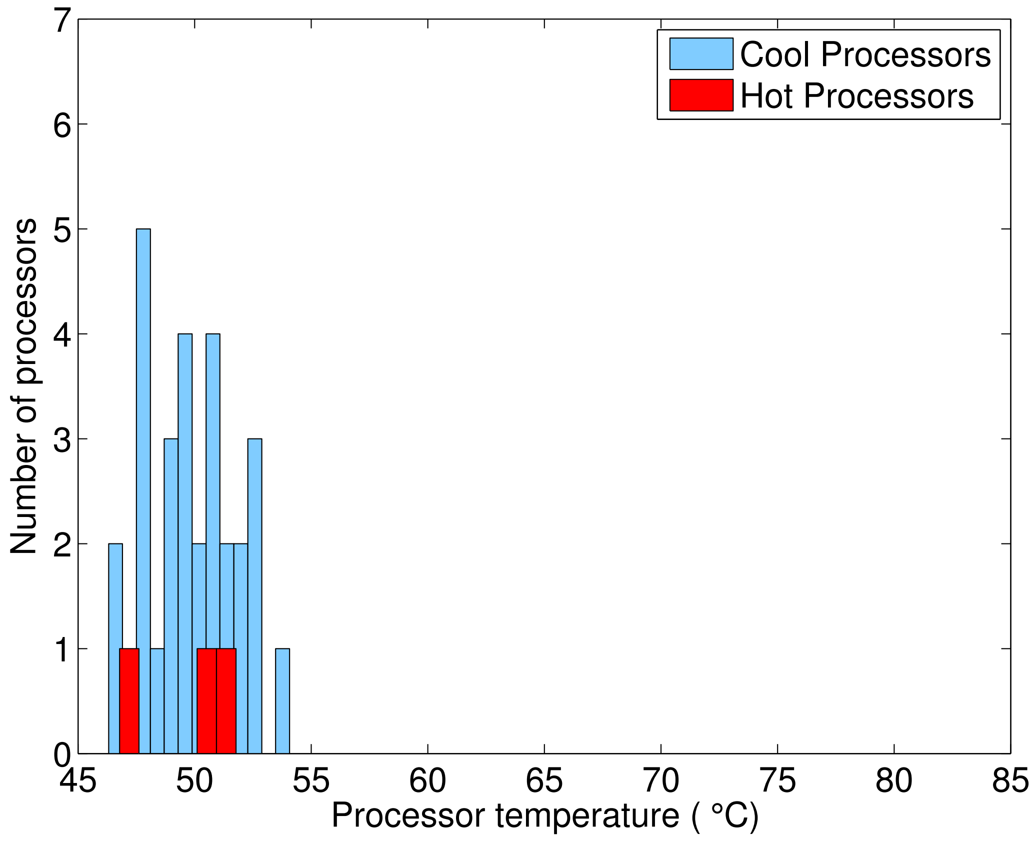
- Using measured max temperature data for cool processors and 59C (same as average temperature for cool processors) for hot processors

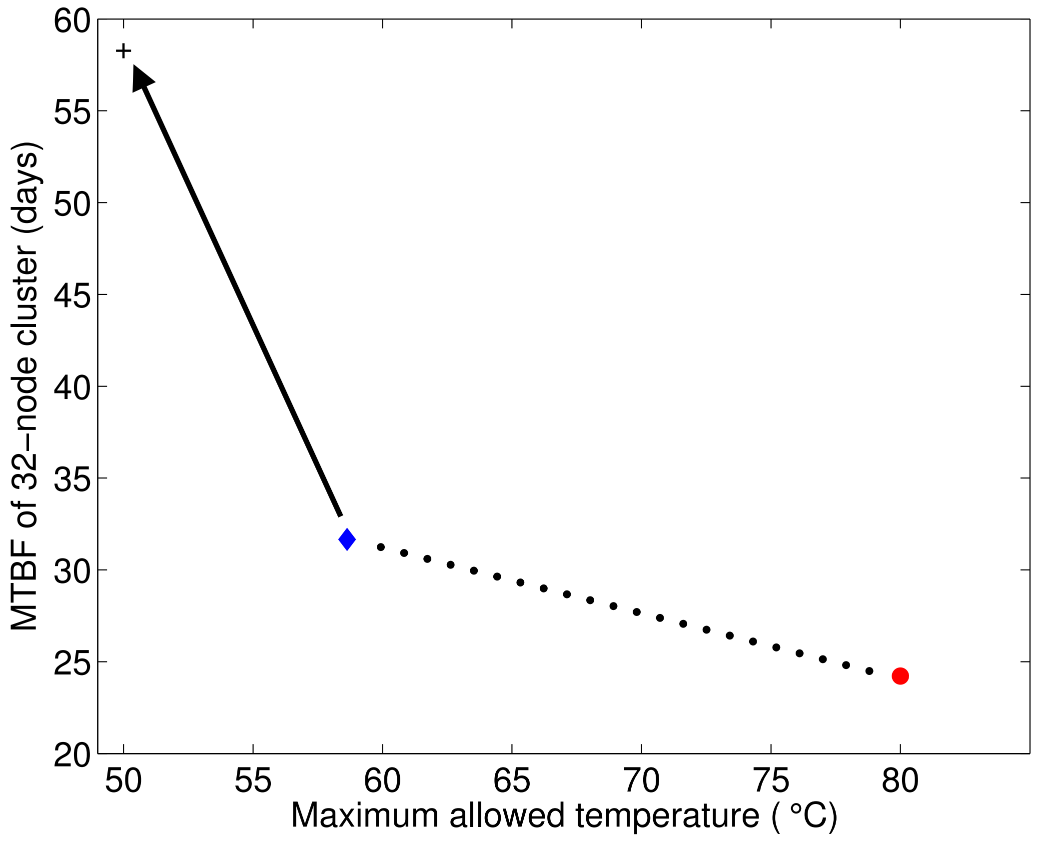




Estimated MTBF - Temperature Restraint

- Using randomly generated temperature data with mean: 50C and std deviation: 2.17C (same as cool processors from the experiment)





Recap

- Restraining temperature can improve the estimated MTBF of our Energy Cluster
    - Originally (No temperature control): 24 days
    - Removing hot spots: 32 days
    - Restraining temperature (mean 50C): 58 days
- How can we restrain processor temperature?
    - Dynamic Voltage and Frequency Scaling (DVFS)[5]?

5. Reduces both voltage and frequency which reduces power consumption resulting in temperature to fall

Restraining Processor Temperature

- Extension of `Cool' Load Balancer
- Specify temperature threshold and sampling interval
- Runtime system periodically checks processor temperature
- 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
- Extended by making it communication aware (details in paper):
    - Select objects (for migration) based on the amount of communication it does with other processors

Improving MTBF and Its Cost

- Temperature restraint comes along DVFS induced slowdown!
- Restraining temperature to 56C, 54C, and 52C for Wave2D 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:
  - decreases MTBF which in turn decreases check pointing, recovery, and restart times
  - increases time taken by useful work

Performance Model

Symbol	Description
T	Total execution time
W	Useful work
τ	Check point period
δ	check point time
R	Restart time
μ	slowdown

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

$$T = W\mu + \left(\frac{W\mu}{\tau} - 1\right)\delta + \frac{T}{M}\left(\frac{\tau+\delta}{2}\right) + \frac{T}{M}R$$ [1]

1. J. T. Daly, A higher order estimate of the optimum checkpoint interval for restart dumps

Model Validation

- Experiments on 32-nodes of Energy Cluster
- To emulate the number of failures in a 700K socket machine, we utilize a scaled down value of MTBF (4 hours per socket)
- Inject random faults based on estimated MTBF values using ‘kill -9’ command
- Three applications:
    - Jacobi2D: 5 point-stencil
    - LULESH: Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics
    - Wave2D: finite difference for pressure propagation

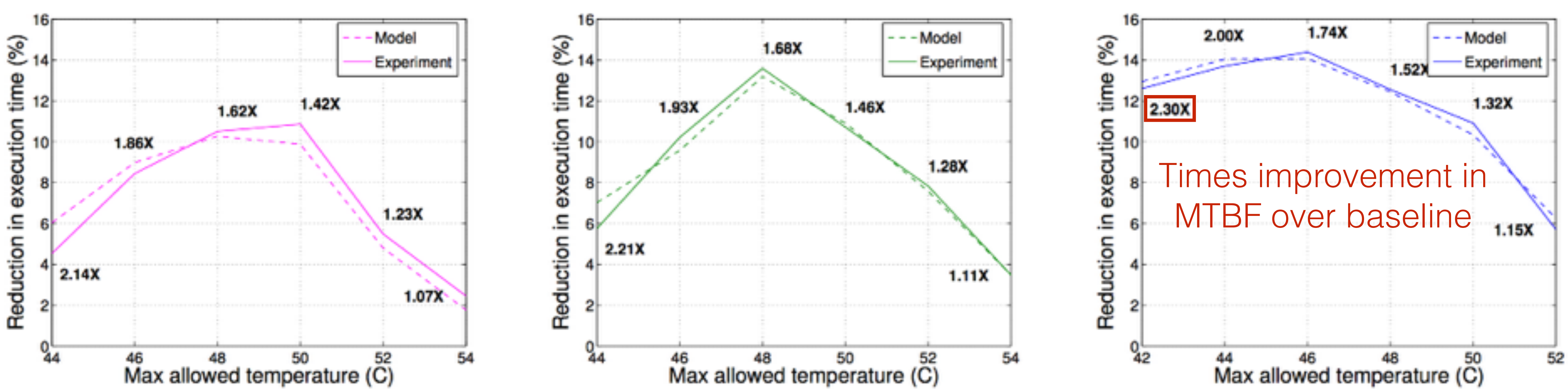
Model Validation

- Baseline experiments:
  - Without temperature restraint
  - MTBF based on actual temperature data from experiment
- Temperature restrained experiments:
  - MTBF calculated using the max allowed temperature

Reduction in Execution Time

- Each experiment was longer than 1 hour having at least 40 faults

- Inverted-U curve points towards a tradeoff between timing penalty and improvement in MTBF



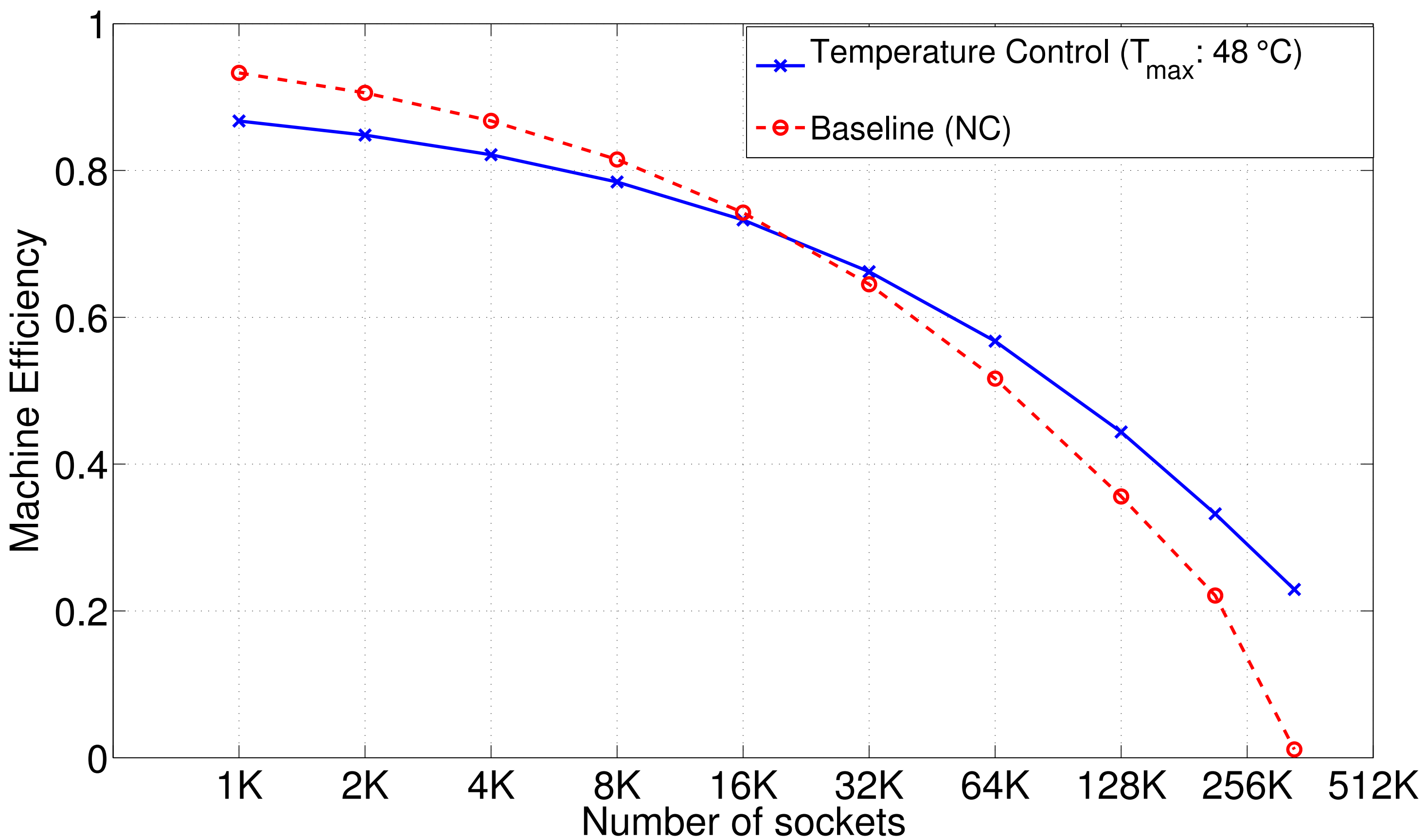
(a) Lulesh (b) Wave2D (c) Jacobi2D

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

Improvement in Machine Efficiency

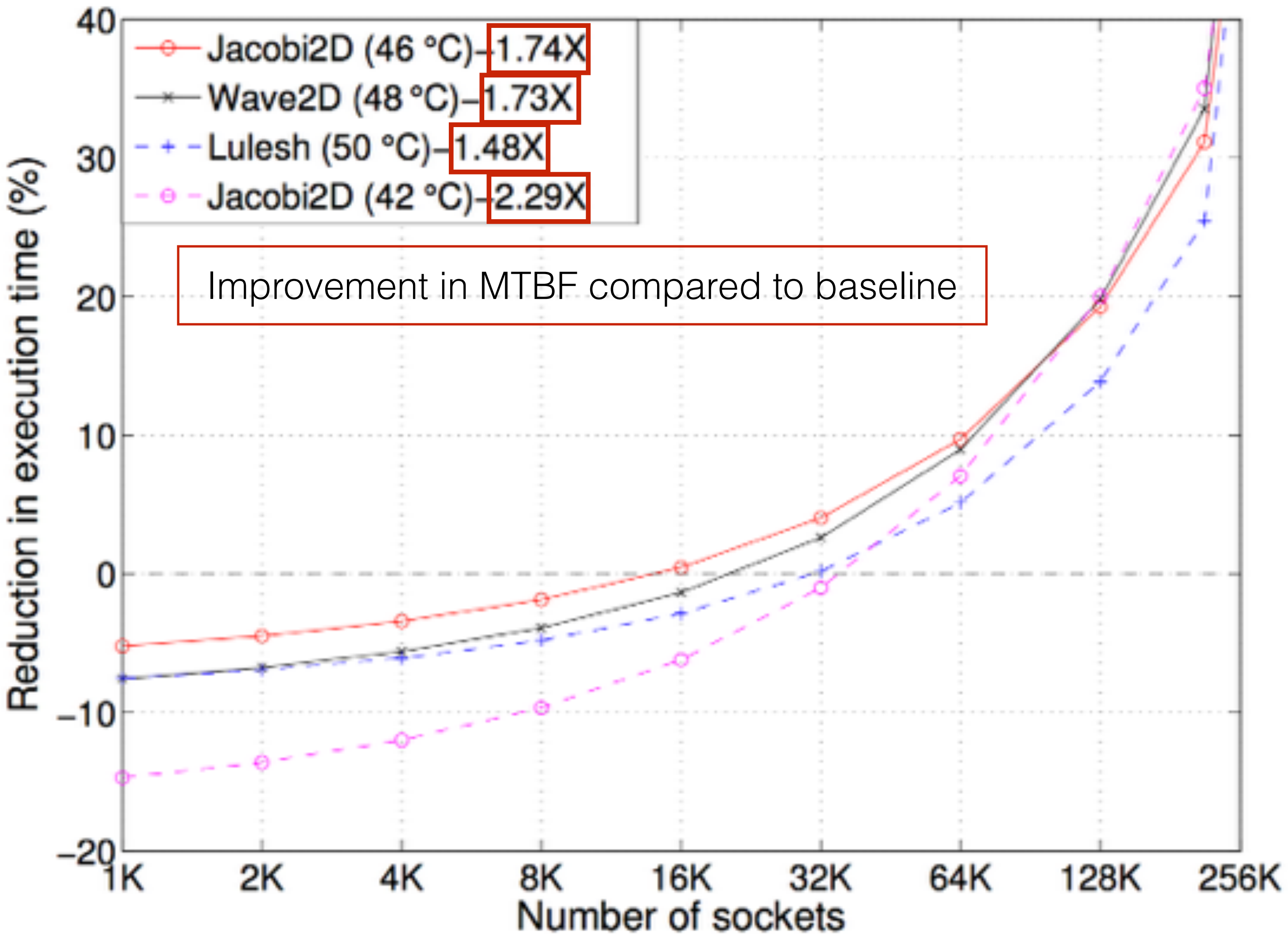
- Our scheme improves utilization beyond 20K sockets compared to baseline
- For 340K socket machine:
  - Baseline: Efficiency < 1% (un operational)
  - Our scheme: Efficiency ~ 21%

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



Predictions for Larger Machines

- Per-socket MTBF of 10 years
- Optimum temperature thresholds



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

Power Constraint

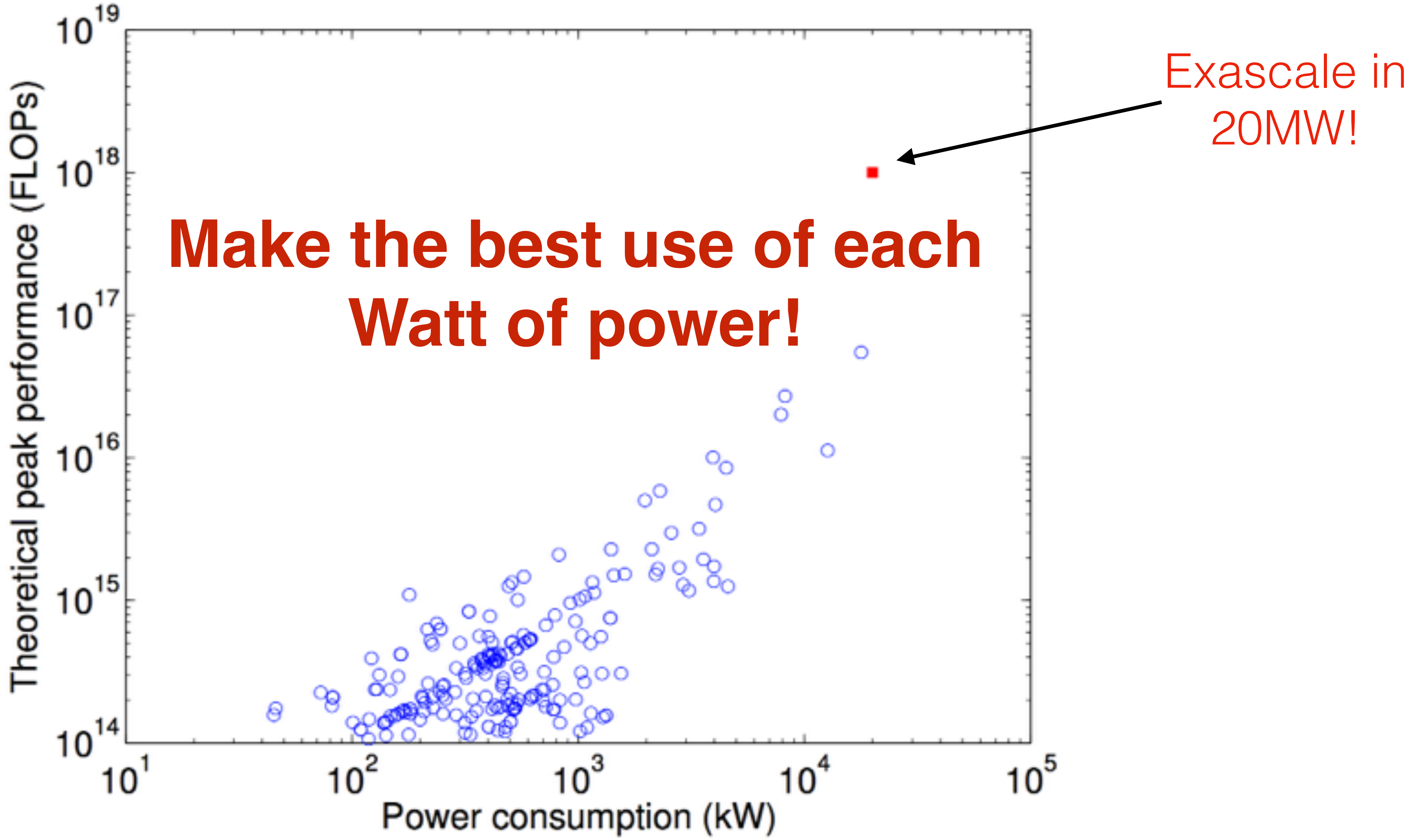
Improving Performance of a Single Application

Publications

- **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.

What's the Problem?

Power consumption for Top500



Overprovisioned Systems[1]

Example
10 nodes @ 100 W (TDP)
20 nodes @ 50 W

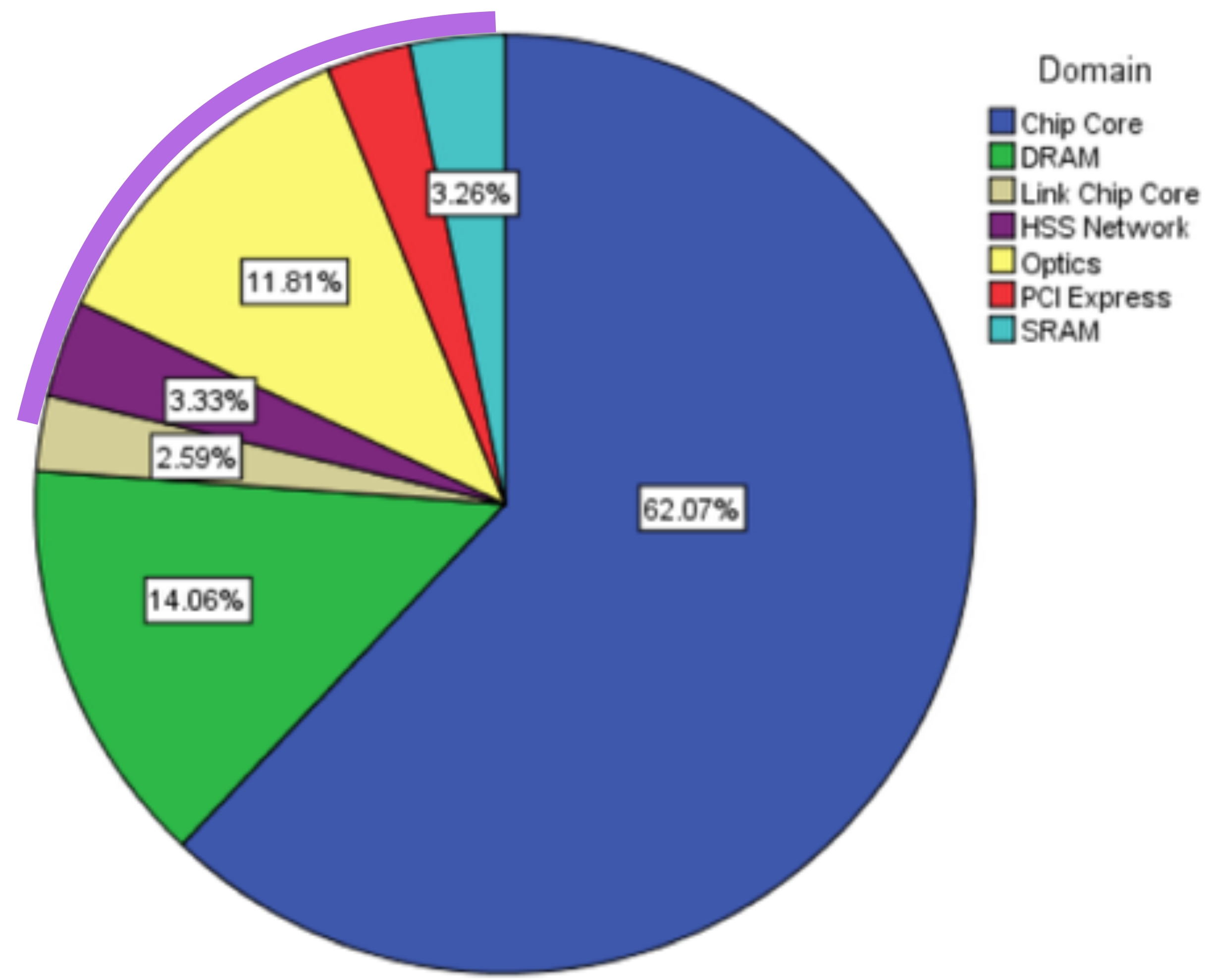
- What we currently do:
  - Assume each node consumes Thermal Design Point (TDP) power
- What we should do (overprovisioning):
  - Limit power of each node and use more nodes than a conventional data center
- Overprovisioned system: You *can't* run all nodes at max power simultaneously

1. Patki et. al., Exploring hardware overprovisioning in power-constrained, high performance computing, ICS 2013

Where Does Power Go?

Small with small variation over time

- Power distribution for BG/Q processor on Mira
    - CPU/Memory account for over 76% power
    - No good mechanism of controlling other power domains



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

Power Capping - RAPL

- Running Average Power Limit (RAPL) library
- Uses Machine Specific Registers (MSRs) to:
  - *measure* CPU/Memory power
  - *set* CPU/memory power caps
- Can report CPU/memory power consumption at millisecond granularity

Problem Statement

Optimize the numbers of nodes (n), the CPU power level (p_c) and memory power level (p_m) that minimizes execution time (t) of an application under a strict power budget (P), on a high performance computation cluster with p_b as the base power per node i.e. determine the best configuration (n x {p_c, p_m})

Applications and Testbed

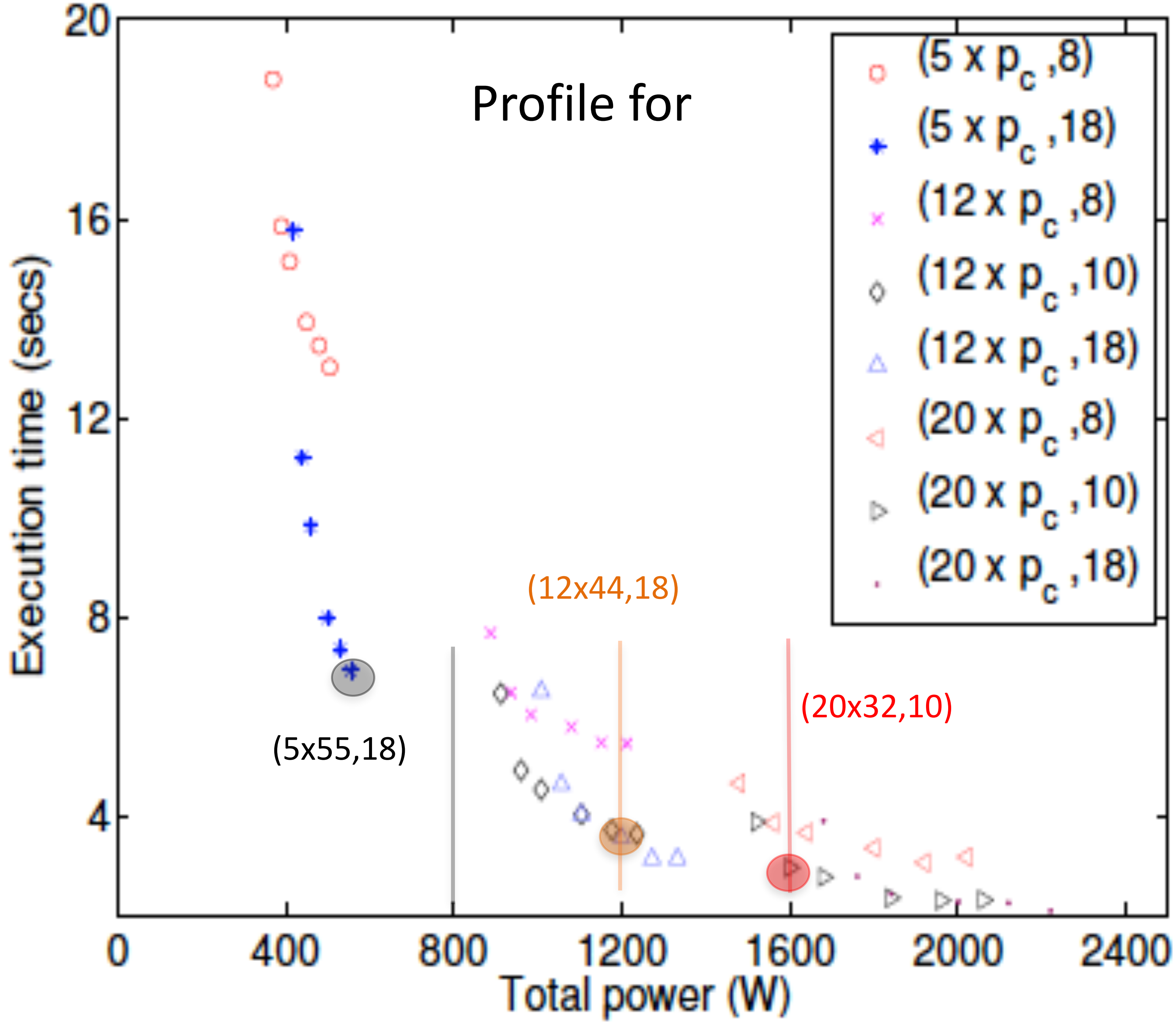
- Applications
  - Wave2D: computation-intensive finite differencing application
  - LeanMD: molecular dynamics simulation program
  - LULESH: Hydrodynamics code
- Power Cluster
  - 20 nodes of Intel Xeon E5-2620
  - Power capping range:
    - CPU: 25-95 W
    - Memory: 8-38W

Profiling Using RAPL

Profile configurations (n x p_c, p_m)

n: Num of nodes
p_c: CPU power cap
p_m: Memory power cap

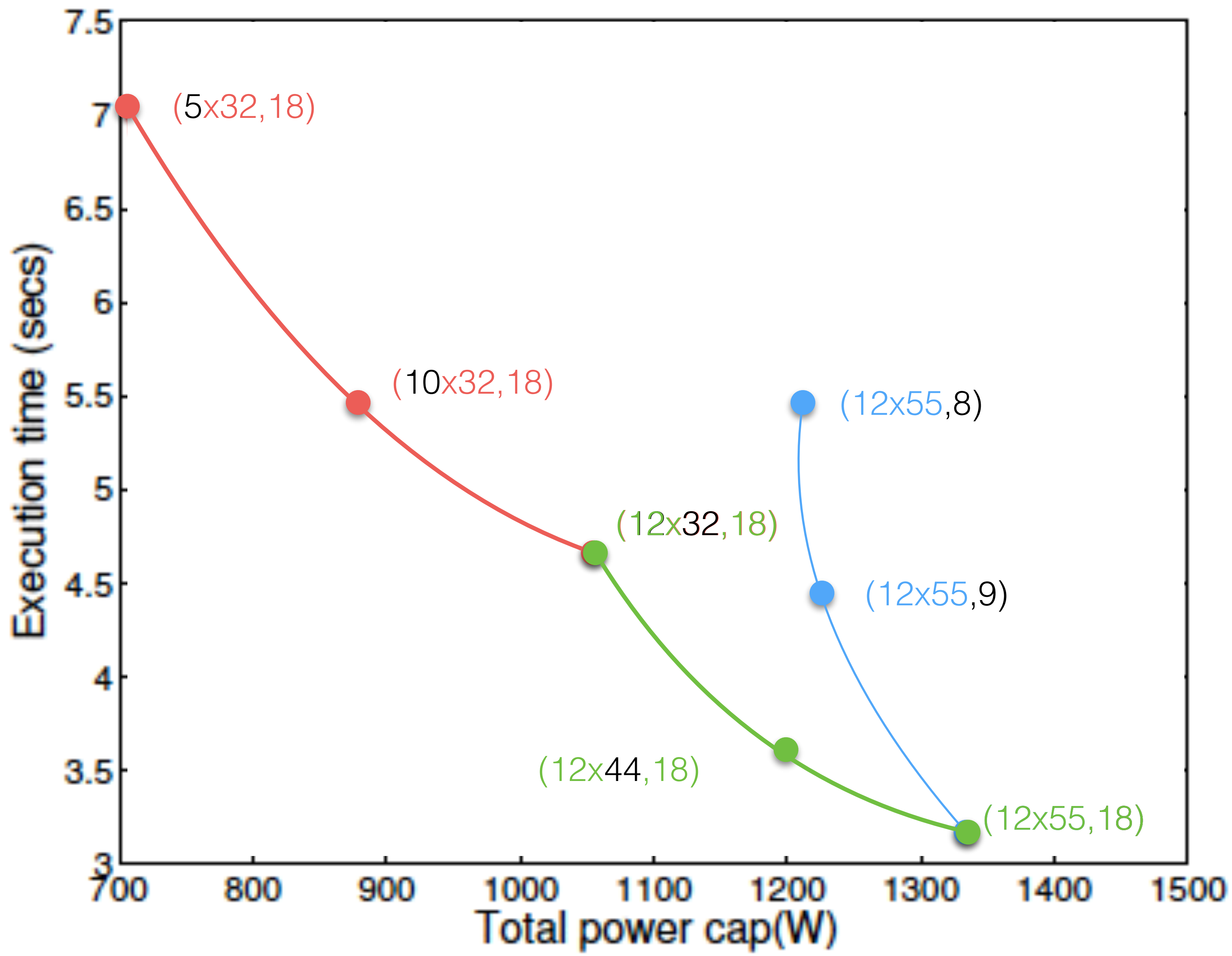
n: {5,12,20}
p_b: {28,32,36,44,50,55}
p_m: {8,10,18}
p_b: 38
Tot. power =
n * (p_c + p_m + p_b)



Can We Do Better?

- More profiling (Expensive!)
- Using interpolation to estimate all possible combinations

Interpolation - LULESH



Evaluation

- Baseline configuration (no power capping):

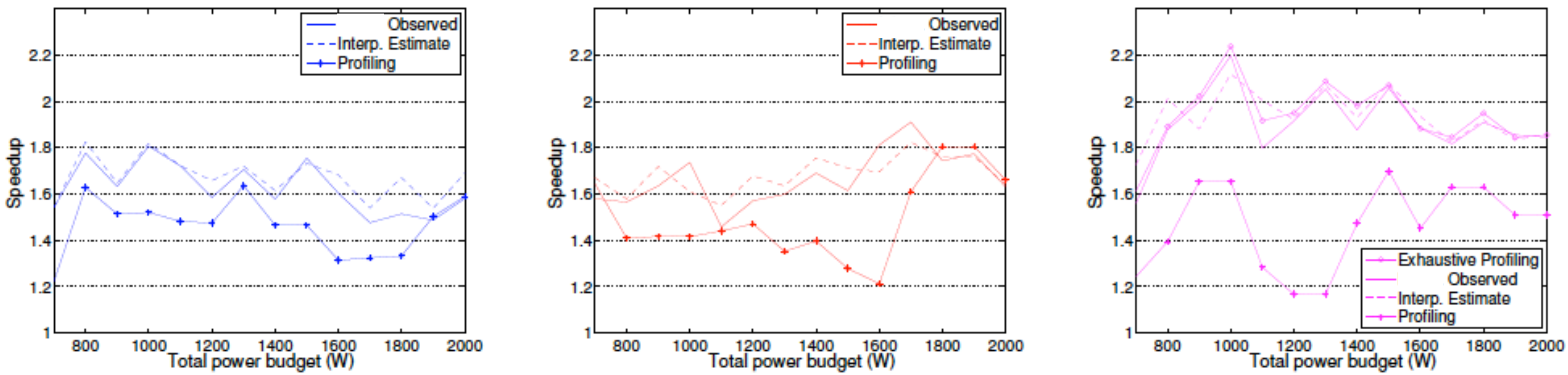
$$(n_b \times TDP_c, TDP_m)$$

$$\text{where } n_b = \left\lfloor \frac{P}{p_b + TDP_c + TDP_m} \right\rfloor$$

- Compare:
  - Profiling scheme: Only the profile data
  - Interpolation Estimate: The estimated execution time using interpolation scheme
  - Observed: Observed execution for the best configurations

Speedups Using Interpolation

Base case: Maximum allowed nodes working at TDP (max) power



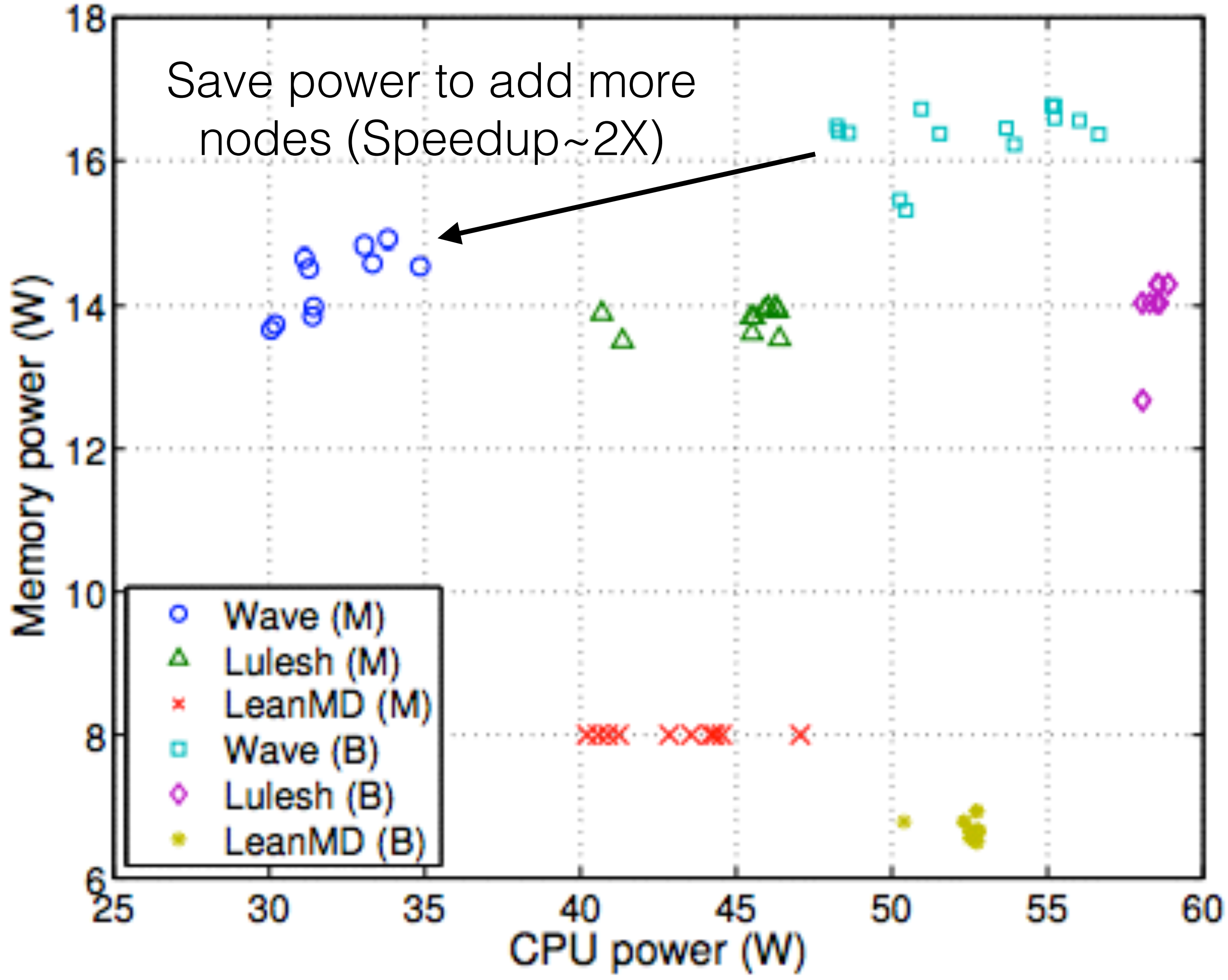
(a) Lulesh (b) Wave2D (c) LeanMD

- Interpolation speedups much better than ‘Profiling’ speedups
- Interpolation speedups close to best possible configurations i.e. exhaustive profiling

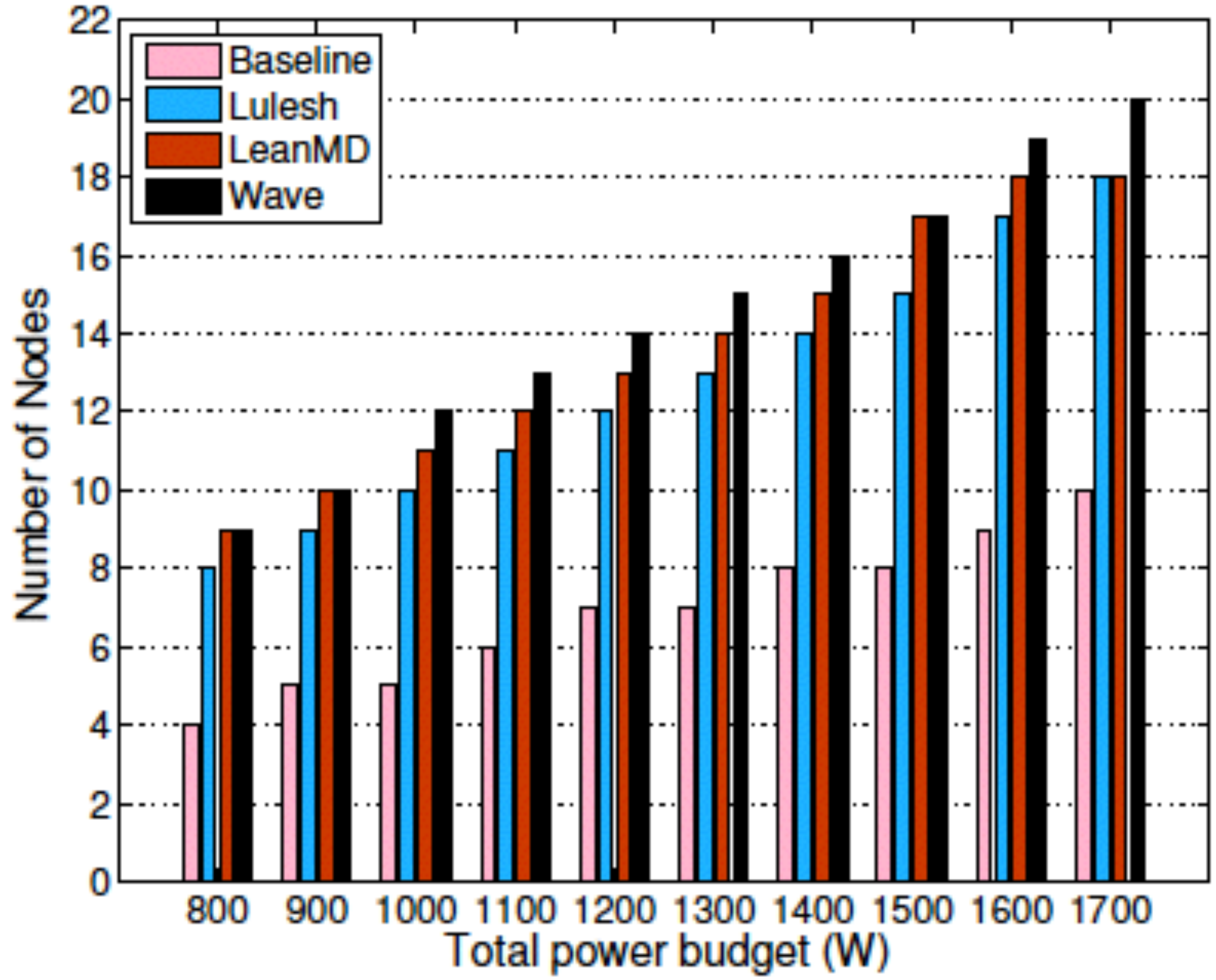
Optimal CPU/Memory Powers

CPU/Memory powers for *different* power budgets:

- M: observed power using our scheme
- B: observed power using the baseline



Optimal Configurations



- Power capping and overprovisioning allows adding more nodes
- Different applications allow different number of nodes to add

Power Constraint

Optimizing Data Center Throughput having Multiple Jobs

Publications

- **Osman Sarood**, Akhil Langer, Abhishek Gupta, Laxmikant Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. IPDPS 2014 (in submission).

Data Center Capabilities

- Overprovisioned data center
- CPU power capping (using RAPL)
- Moldable and malleable jobs

Moldability and Malleability

Moldable jobs

- Can execute on any number of nodes within a specified range
- Once scheduled, number of nodes can not change

Malleable jobs:

- Can execute on any number of nodes within a range
- Number of nodes can change during runtime
    - Shrink: reduce the number of allocated nodes
    - Expand: increase the number of allocated nodes

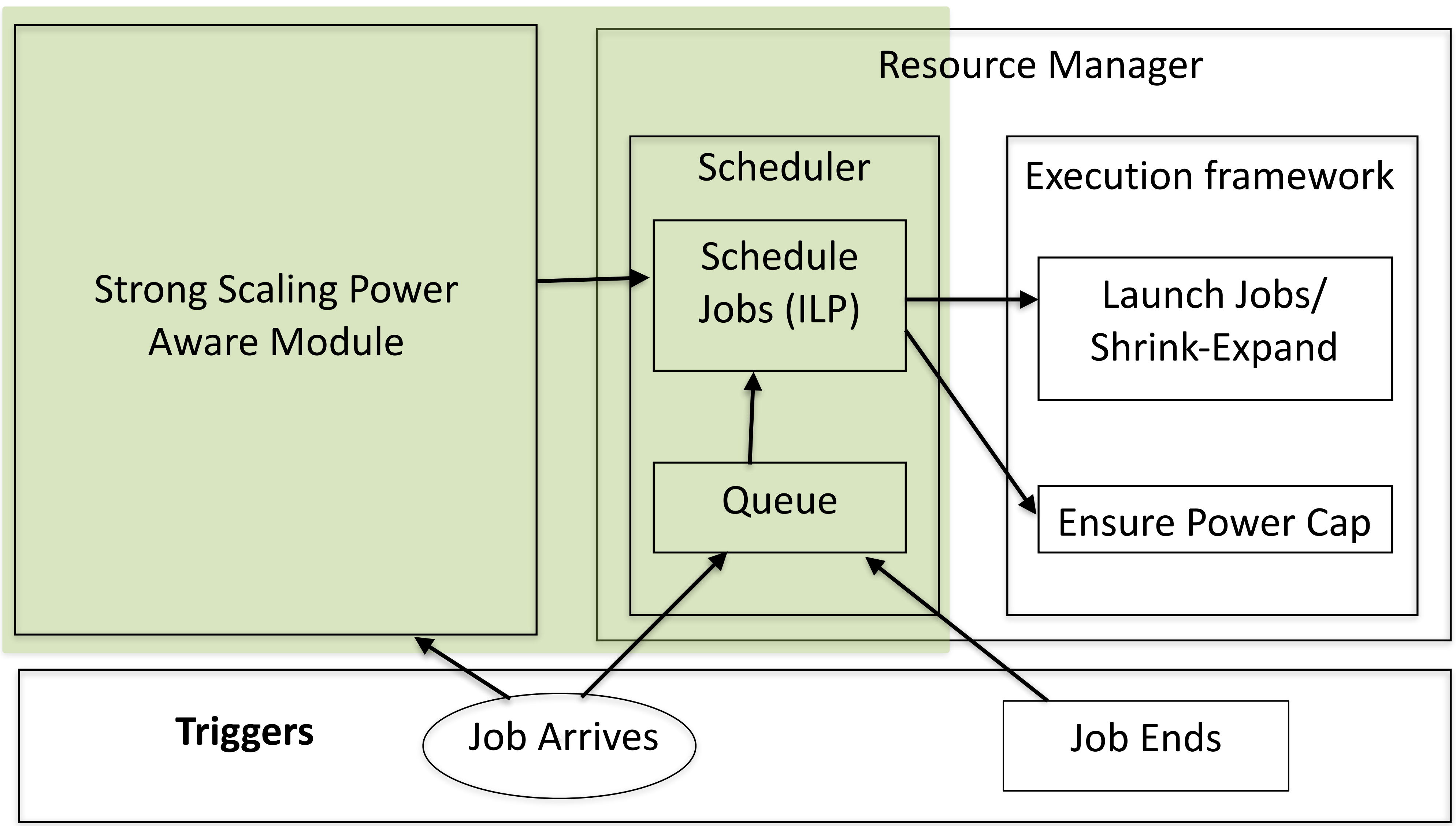
The Multiple Jobs Problem

Given a set of jobs and a total power budget, determine:

- subset of jobs to execute
- resource combination ($n \times p_c$) for each job

such that the throughput of an overprovisioned system is maximized

Framework



Strong Scaling Power Aware Module
Resource Manager
Scheduler
Schedule Jobs (ILP)
Queue
Execution framework
Launch Jobs/ Shrink-Expand
Ensure Power Cap
Triggers
Job Arrives
Job Ends

Throughput

- $t_{j,n,p}$: Execution time for job `j', operating on `n' nodes each capped at `p' watts

- Strong scaling power aware speedup for a job `*j'*, allocated `*n'* nodes each operating under `*p'* watts

$$s_{j,n,p} = \frac{t_{j,min(N_j),min(P_j)}}{t_{j,n,p}}$$

Exe. time using min resources

- Define throughput as the sum of strong scaling power aware speedups of all jobs scheduled at a particular scheduling time

Scheduling Policy (ILP)

Objective Function

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

Starvation!

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)

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

Making the Objective Function Fair

- Assigning a weight to each job `j’

Objective Function

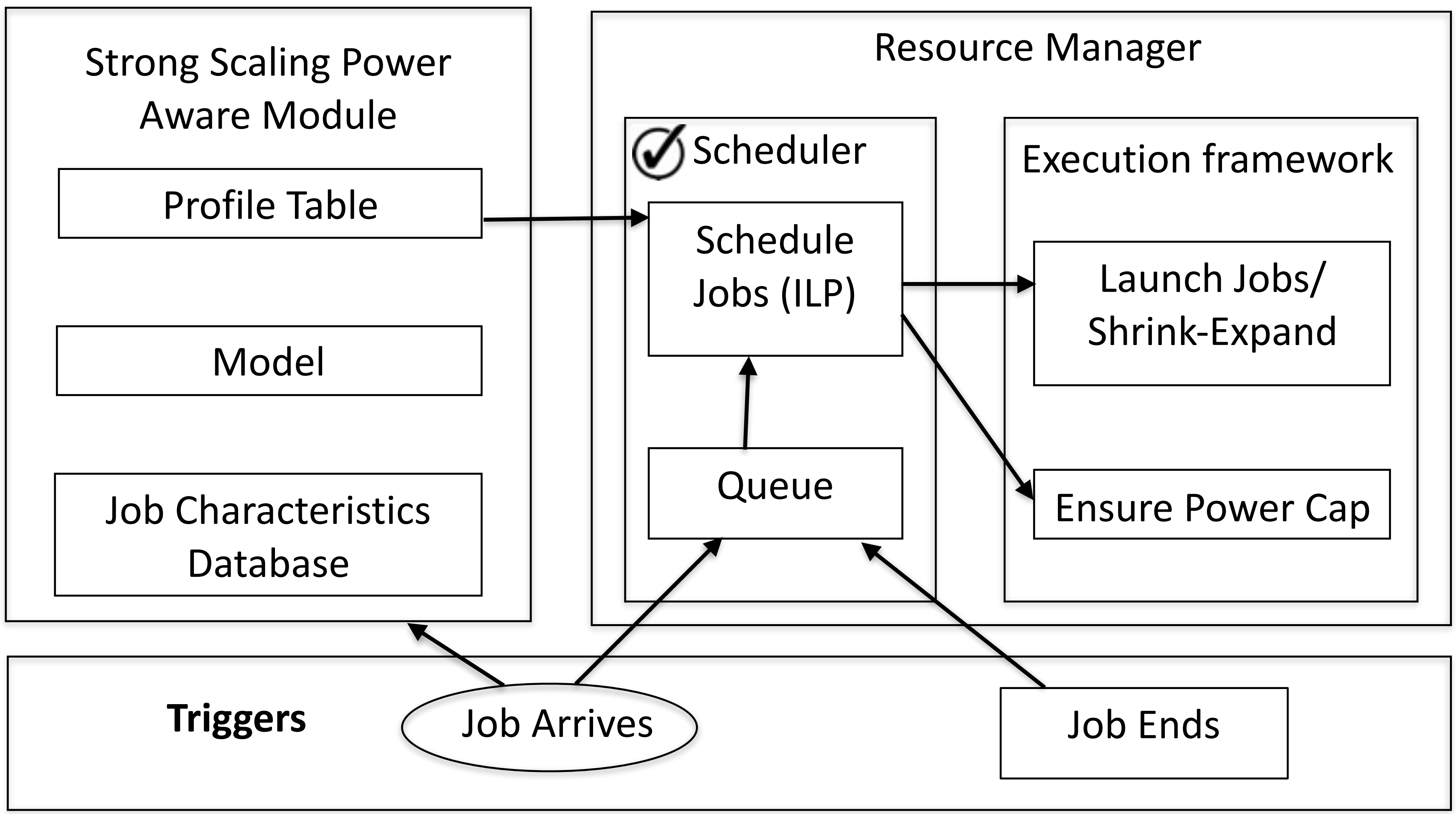
$$\sum_{j\in\mathcal{J}} \sum_{n\in N_j} \sum_{p\in P_j} w_j * s_{j,n,p} * x_{j,n,p}$$

$$w_j = (t^{rem}_{j,min(N_j),min(P_j)} + (t_{now} - t^a_j))^{\alpha}$$

Remaining time using min resources — Time elapsed since arrival — Extent of fairness

- t^a_j : arrival time of job `j’
- t_{now} : current time at present scheduling decision
- $t^{rem}_{j,min(N_j),min(P_j)}$: remaining time for job ‘j’ executing at minimum power operating at lowest power level

Framework



Strong Scaling Power Aware Module
Profile Table
Model
Job Characteristics Database
Resource Manager
Scheduler
Schedule Jobs (ILP)
Queue
Execution framework
Launch Jobs/ Shrink-Expand
Ensure Power Cap
Triggers
Job Arrives
Job Ends

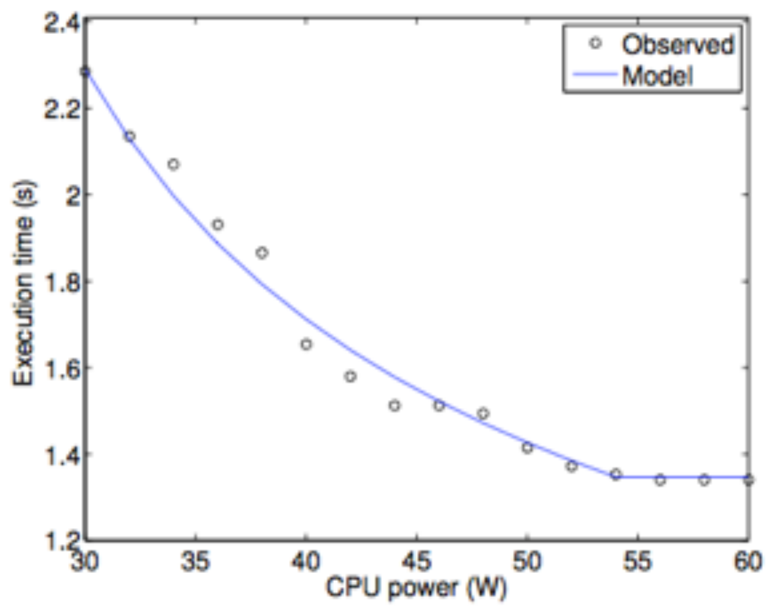
Power Aware Model

- Estimate exe. time for a given number of nodes `n' for varying CPU power `p'
  - Express execution time (*t*) as a function of frequency (*f*)
  - Express frequency (*f*) as a function of package/CPU power (*p*)
  - Express execution time (*t*) as a function of package/CPU power (*p*)

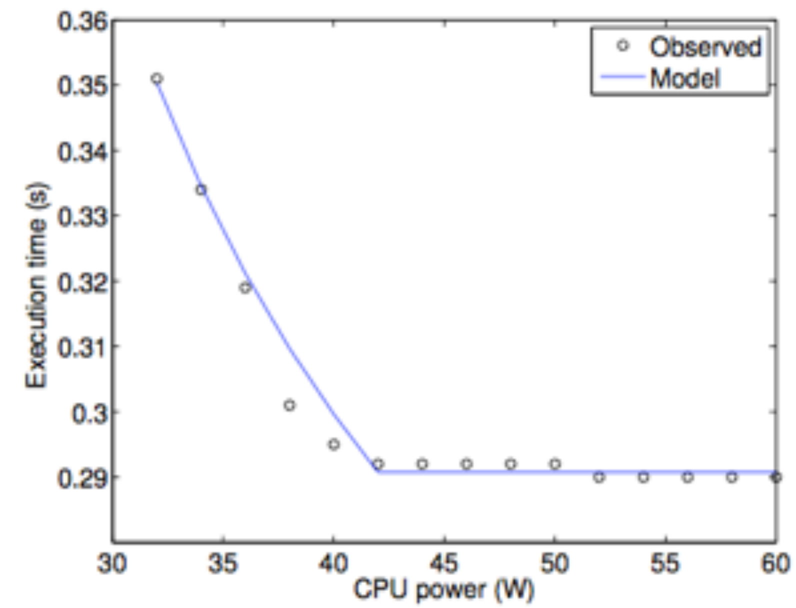
Power Aware Strong Scaling

- Extend Downey’s strong scaling model
- Build a power aware speedup model
- Combine strong scaling model with power aware model
- Given a number of nodes `n’ and a power cap for each node `p’, our model estimates execution time

Fitting Power Aware Model to Application Profile

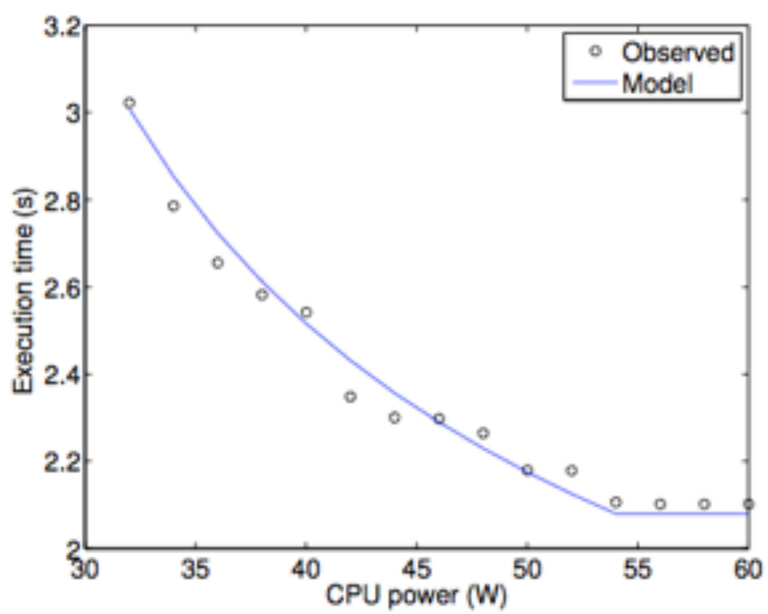


(a) LeanMD

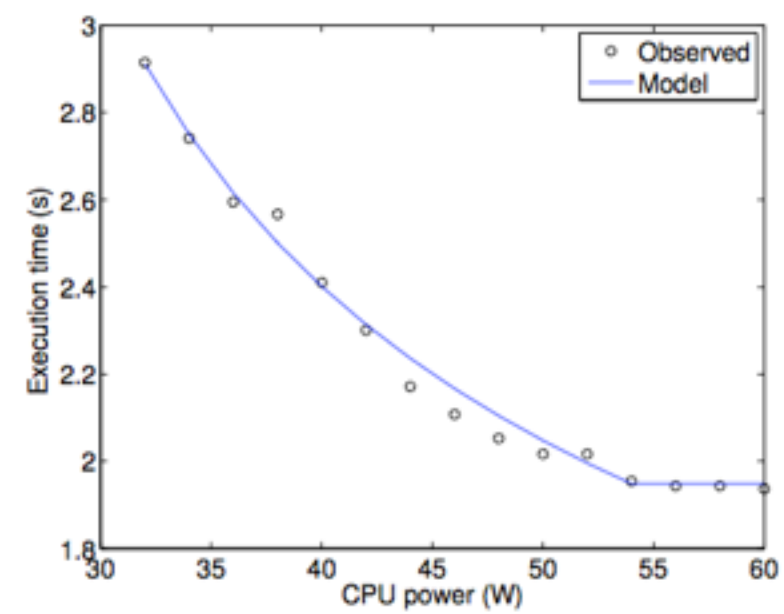


(b) Wave2D

Varying CPU power for 20 nodes



(c) Lulesh



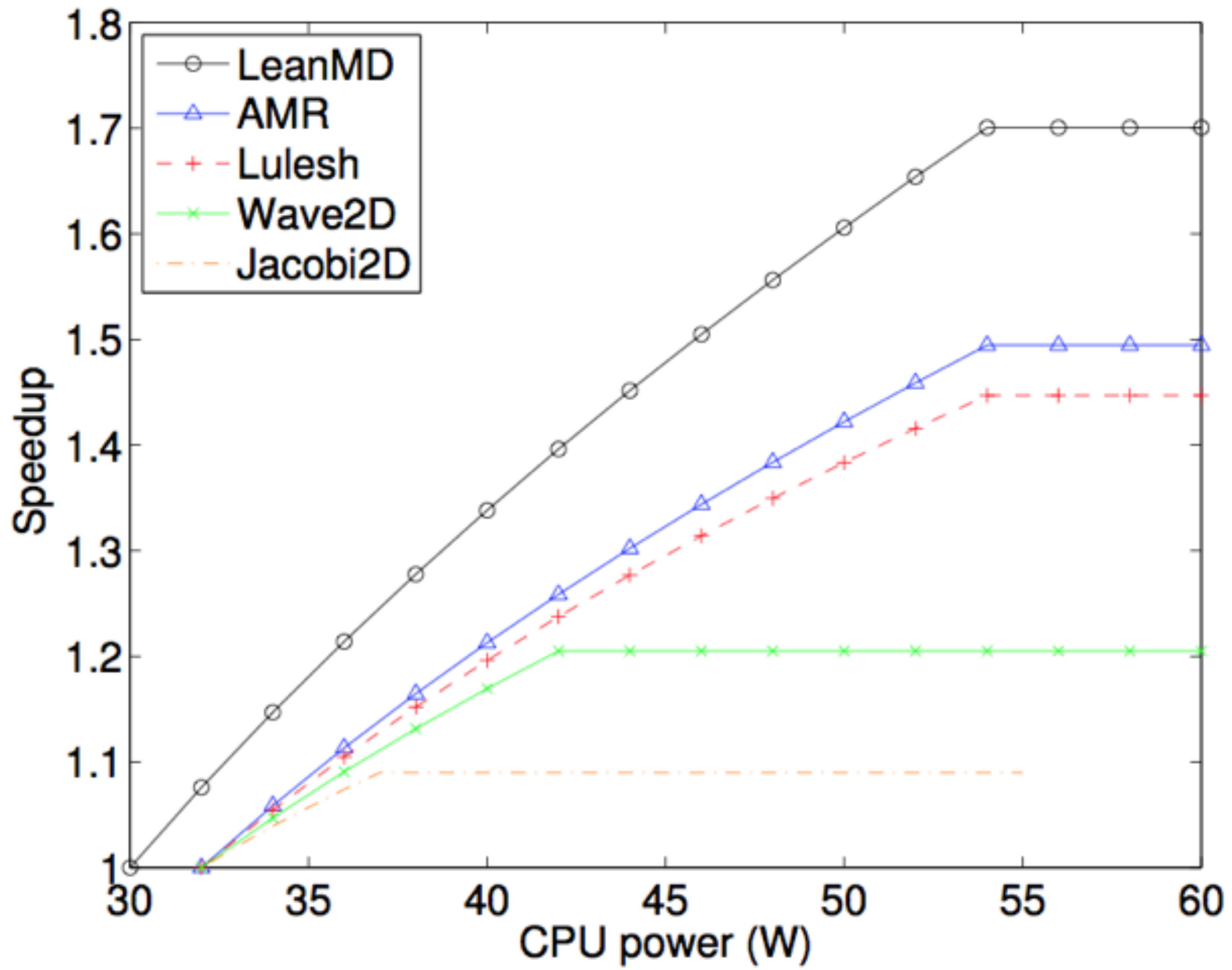
(d) AMR

Power Aware Speedup and Parameters

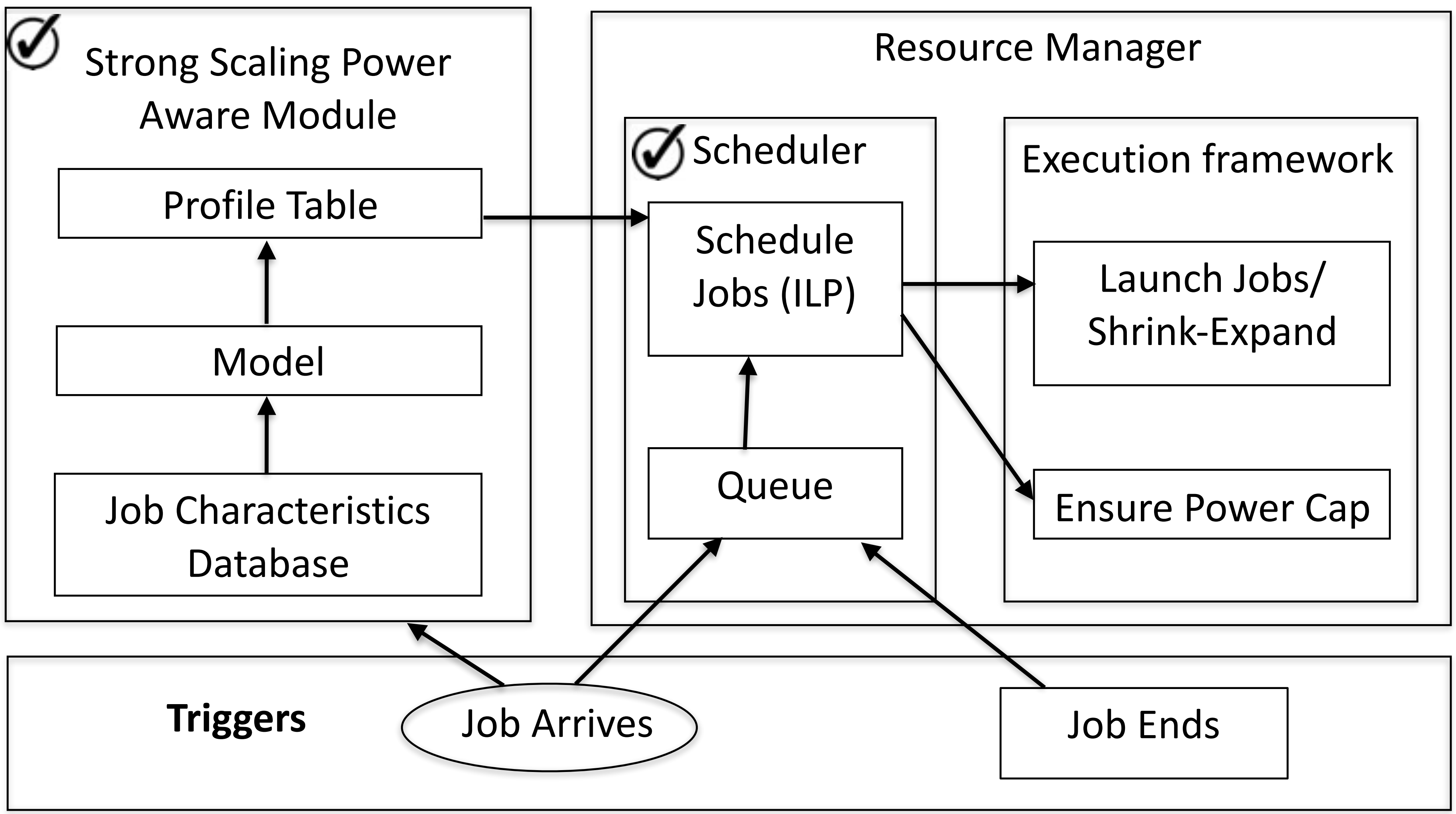
Estimated Parameters

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

Speedups based on execution time at lowest CPU power



Approach (Summary)



Experimental Setup

- Comparison with baseline policy of SLURM
- Using Intrepid trace logs (ANL, 40,960 nodes, 163,840 cores)
- 3 data sets each containing 1000 jobs
- Power characteristics: randomly generated
- Includes data transfer and boot time cost for shrink/expand

Experiments: Power Budget (4.75 MW)

- **Baseline** policy/SLURM: using 40,960 nodes operating at CPU power 60W, memory power 18W, and base power 38W. SLURM Simulator[1]
- **noSE**: Our scheduling policy with *only moldable* jobs. CPU power <=60W, memory power 18W and base power 38W, nodes > 40,960 nodes
- **wiSE**: Our scheduling policy with *both moldable jobs and malleable jobs* i.e. shrink/expand. CPU power <=60W, memory power 18W and base power 38W, nodes > 40,960 nodes

1. A. Lucero, SLURM Simulator

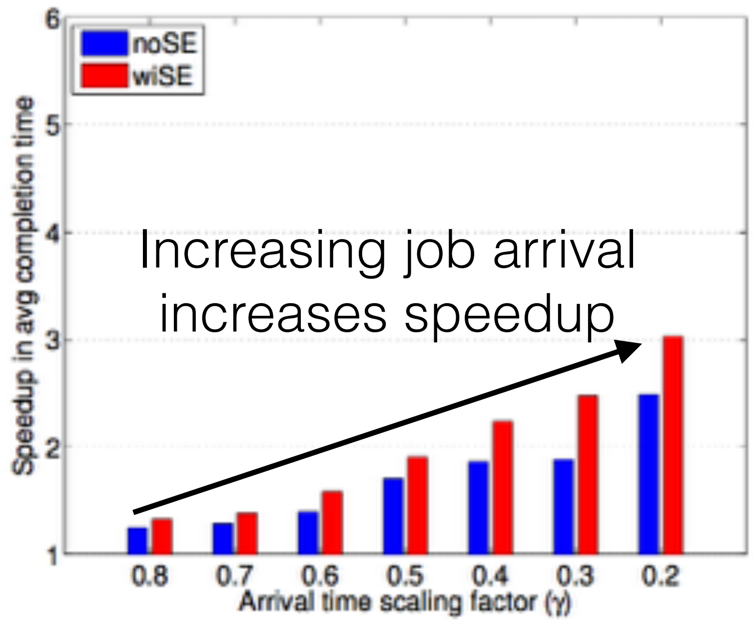
Metrics

- Response time: Time interval between arrival and start of execution
- Completion time: response time + execution time
- Max completion time: Largest completion time for any job in the set

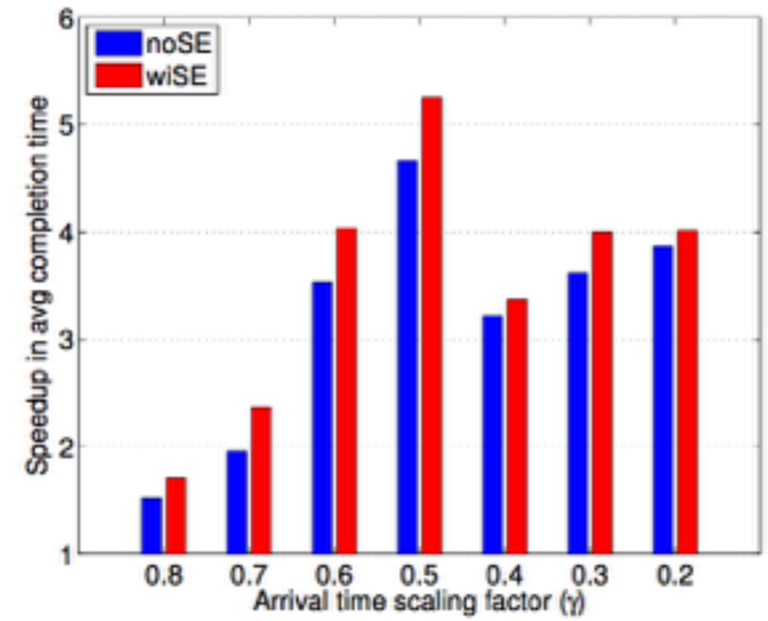
Changing Workload Intensity (γ)

- Impact of increasing job arrival rate
- Compressing data set by a factor γ
- Multiplying arrival time of each job in a set with $\gamma \in [0.2 - 0.8]$

Speedup

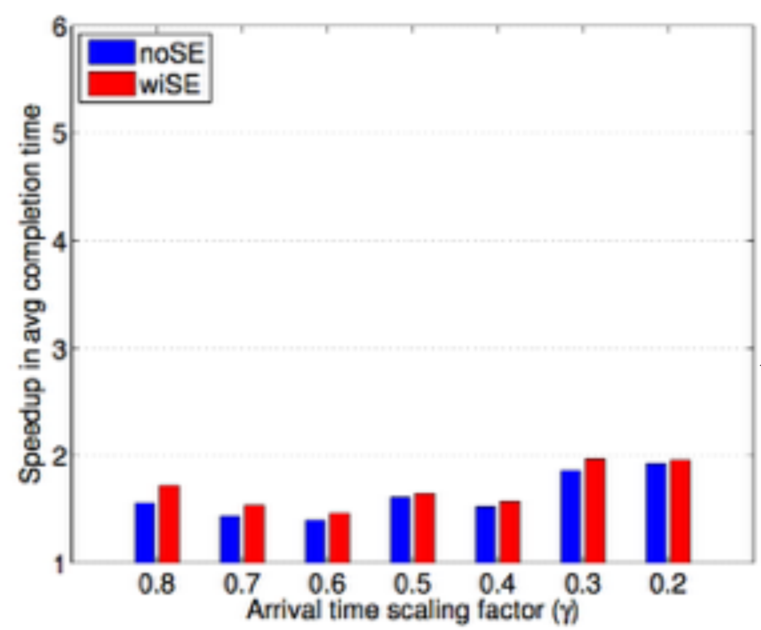


(a) Set1



(b) Set2

wiSE better than **noSE**



Not enough jobs: Low speedups

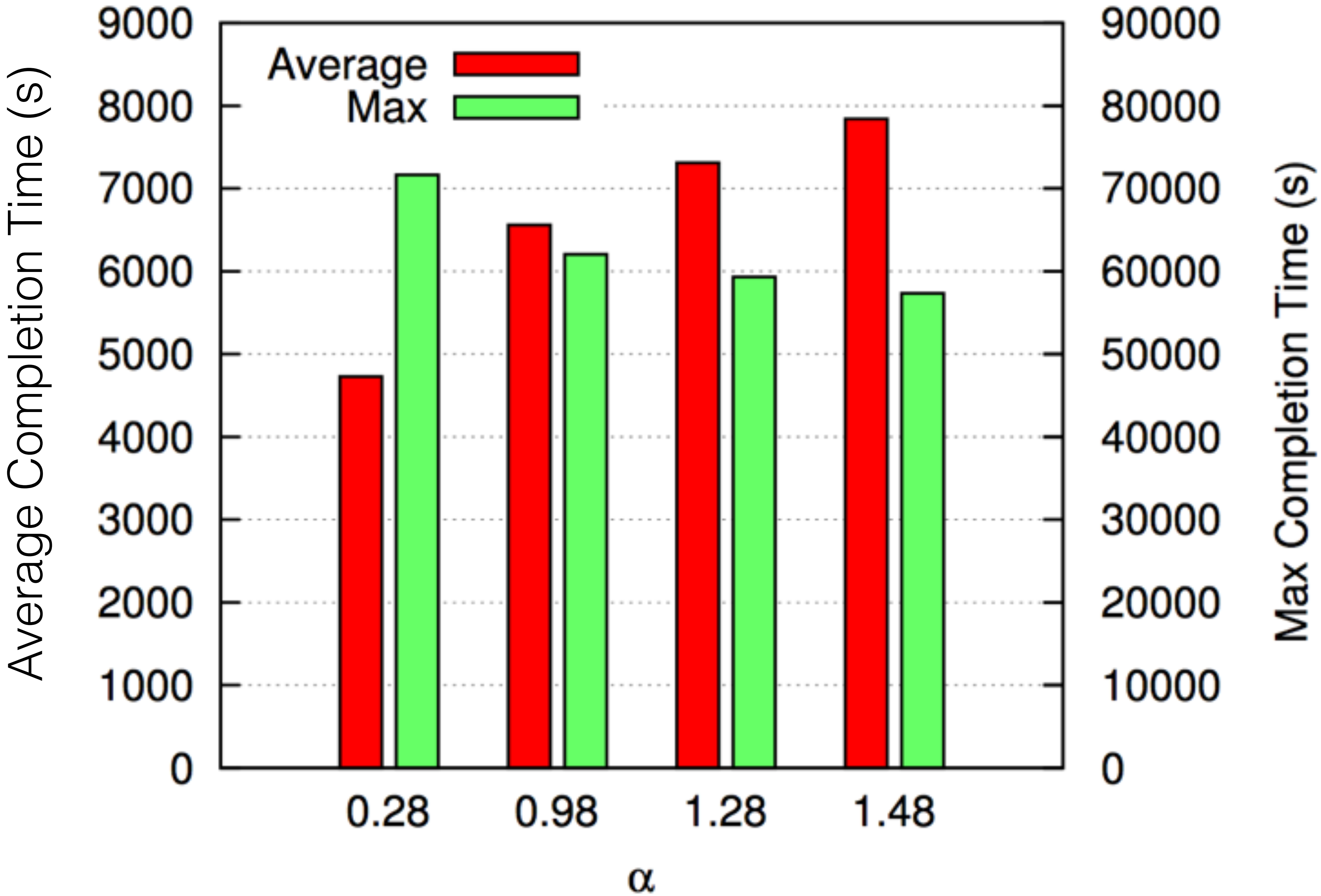
Speedup compared to baseline SLURM

Comparison With Power Capped SLURM

- Its not just overprovisioning!
- wiSE compared to a power capped SLURM policy using over provisioning for Set2
- Cap CPU powers below 60W to benefit from overprovisioning

CPU power cap	30	40	50	60
Speedup	4.32	1.86	2.33	5.25
Avg number of nodes	50332	42486	39700	37956

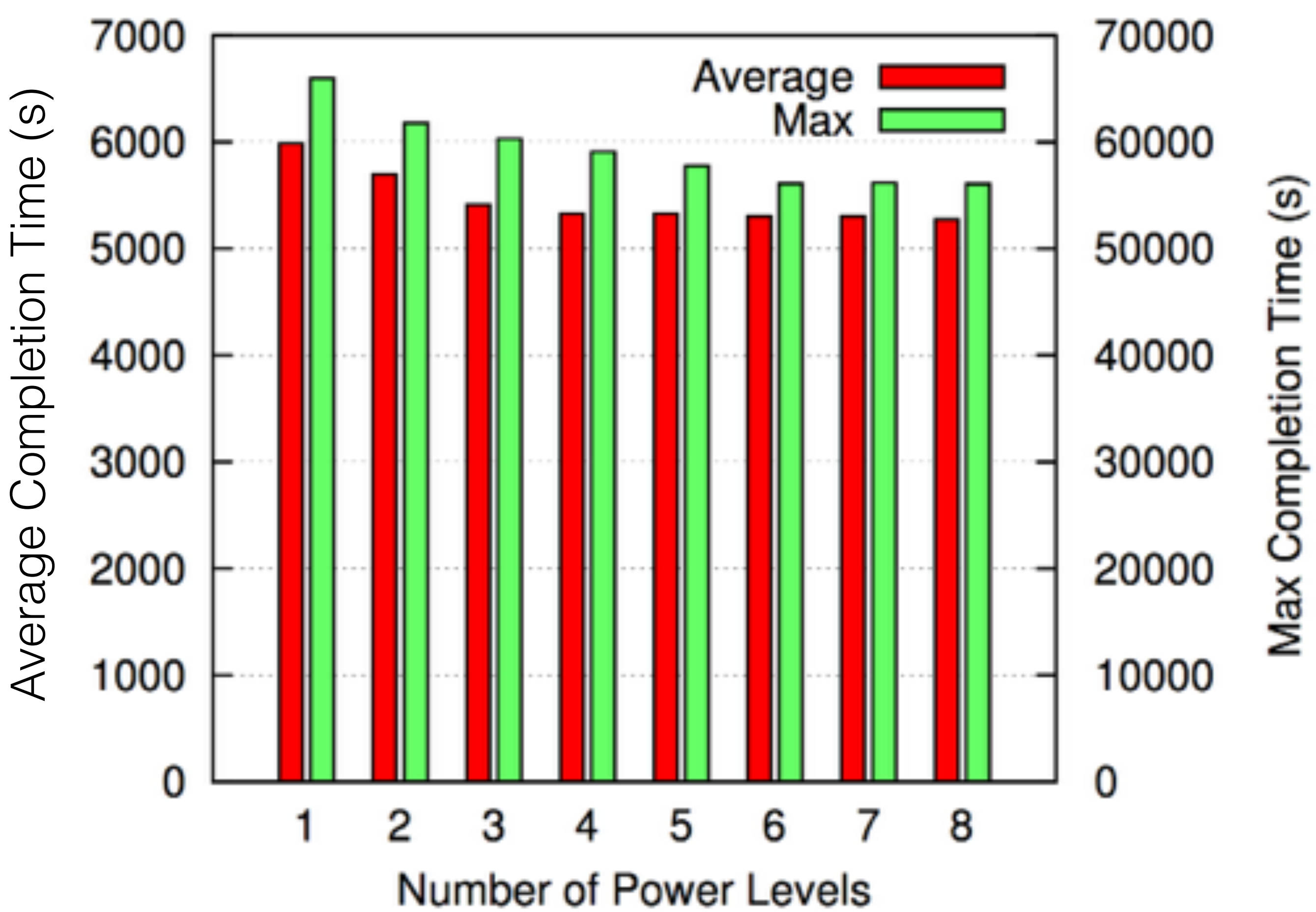
Tradeoff Between Fairness and Throughput



Varying Number of Power Levels

Increasing number of power levels:

- Increase cost of solving ILP
- Improve the average or max completion time



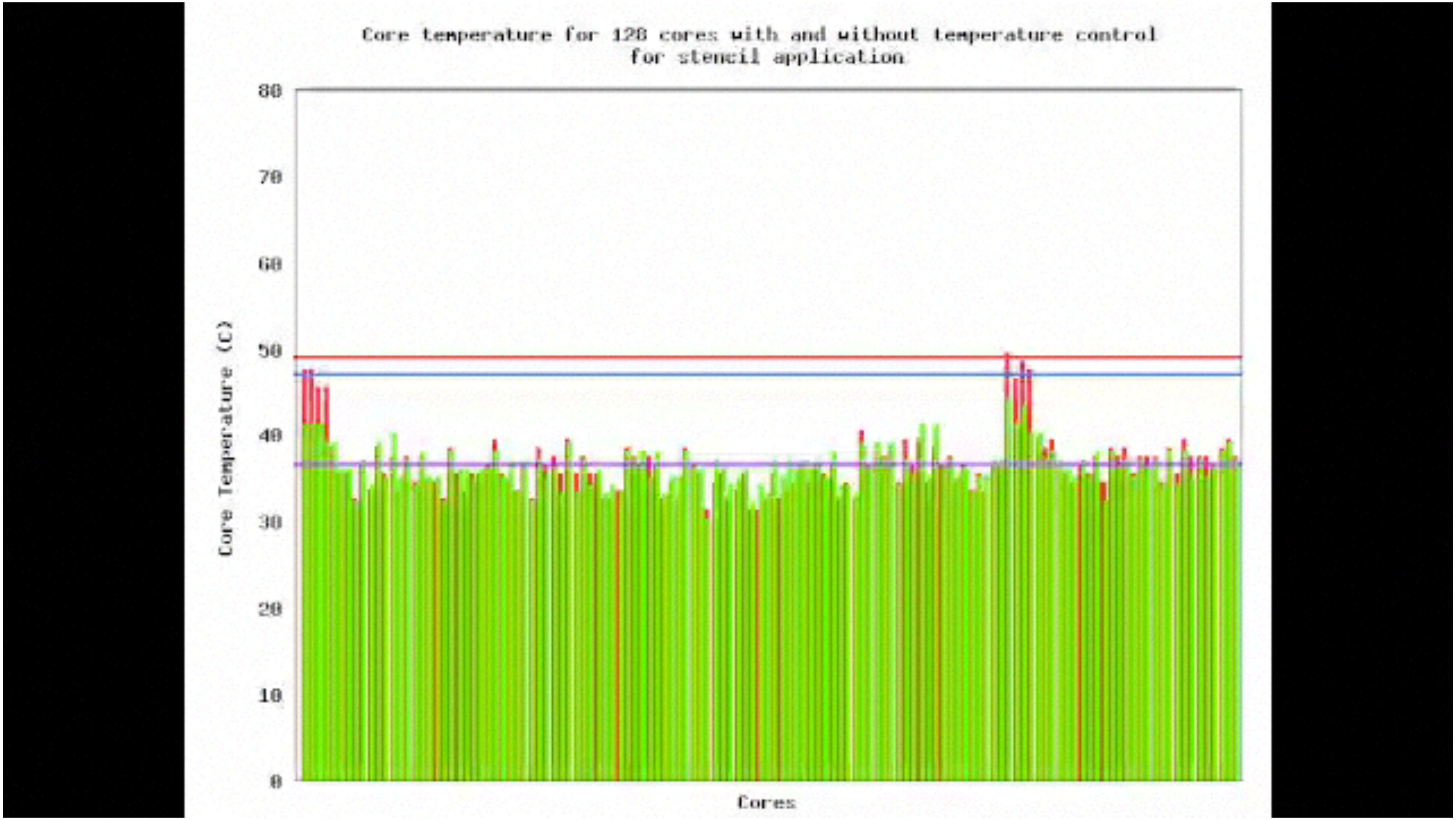
Major Contributions

- Use of DVFS to reduce cooling energy consumption
  - Cooling energy savings of up to 63% with timing penalty between 2-23%
- Impact of processor temperature on reliability of an HPC machine
  - Increase MTBF by as much as 2.3X
- Improve machine efficiency by increasing MTBF
  - Enables machine to operate with 21% efficiency for 340K socket machine (<1% for baseline)
- Use of CPU and memory power capping to improve application performance
  - Speedup of up to 2.2X compared to case that doesn't use power capping
- Power aware scheduling to improve data center throughput
  - Both our power aware scheduling schemes achieve speedups up to 4.5X compared to baseline SLURM
- Power aware modeling to estimate an application's power-sensitivity

Publications (related)

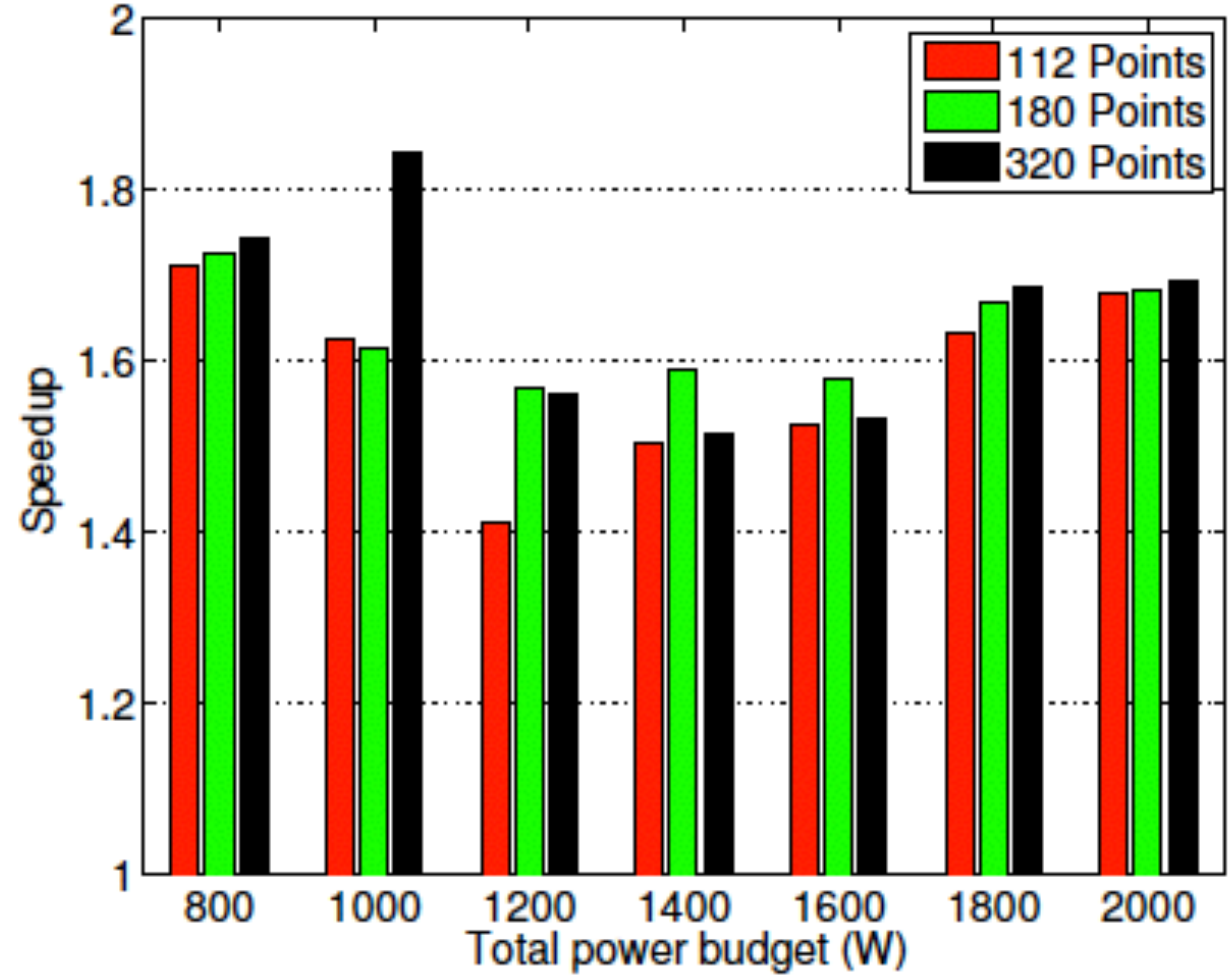
- **Osman Sarood**, Akhil Langer, Abhishek Gupta, Laxmikant Kale. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. IPDPS 2014 (in submission).
- Esteban Meneses, **Osman Sarood**, and Laxmikant V. Kale. Energy Profile of Rollback-Recovery Strategies in High Performance Computing. Elsevier - Parallel Computing (invited paper - in submission).
- **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

Varying Amount of Profile Data

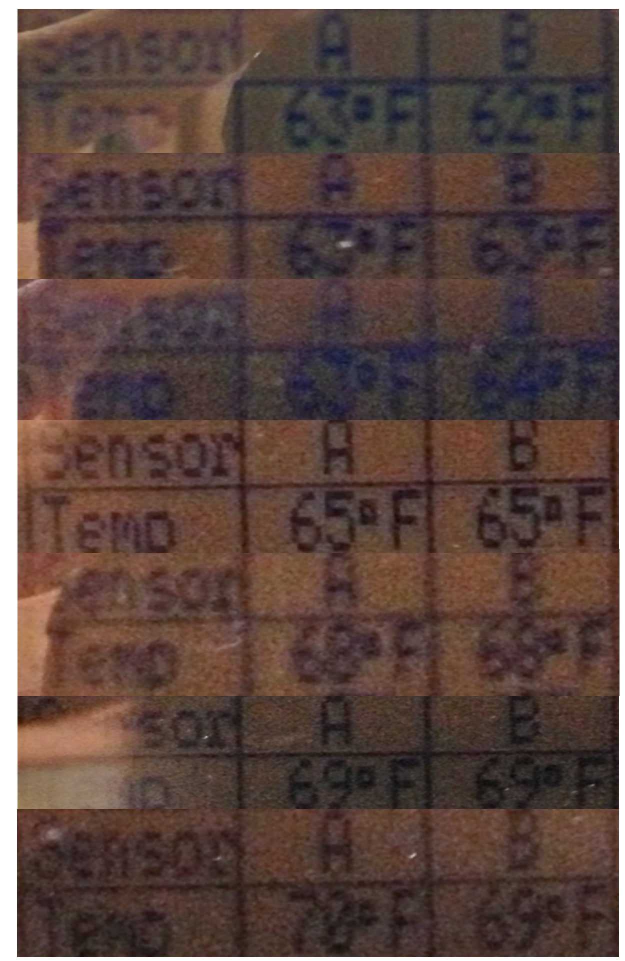


- Observed speedups using different amount of profile data
- 112 points suffice to give reasonable speedup

Blue Waters Cooling

Blue Waters Inlet Water Temperature for Different Rows

Row 1



63F **62F**

63F 63F

63F 64F

65F 65F

68F 68F

69F 69F

70F 69F

Row 7