PARM : POWER-AWARE RESOURCE MANAGER

Maximizing Data Center Performance Under Strict Power Budget

*Osman Sarood, **Akhil Langer**, Abhishek Gupta, Laxmikant Kale*

Parallel Programming Laboratory

Department of Computer Science

University of Illinois at Urbana-Champaign

25th March 2015

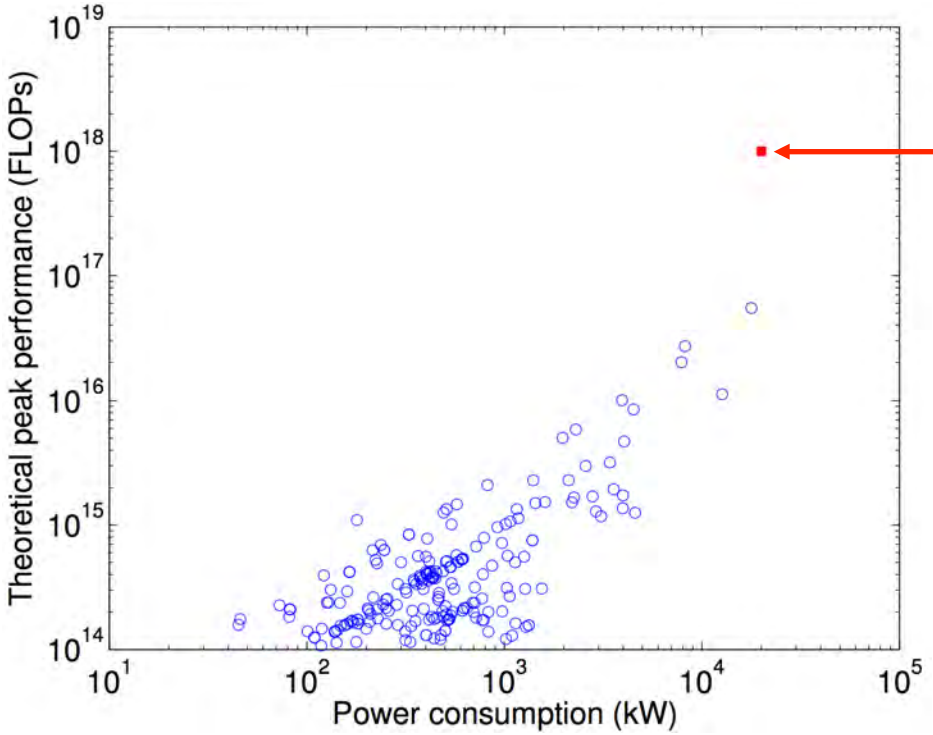
EE HPC WG Webinar on

Dynamic Power Management for MW-sized Supercomputer Centers

PARALLEL PROGRAMMING LABORATORY

Major Challenge to Achieve Exascale

Power consumption for Top500



Exascale in 20MW!

Data Center Power

How is power demand of data center calculated?

- ❑ Using Thermal Design Power (TDP)!

However, TDP is hardly reached!!

Constraining CPU/Memory power

Intel Sandy Bridge

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

Constraining CPU/Memory power

Intel Sandy Bridge

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

Solution to Data Center Power

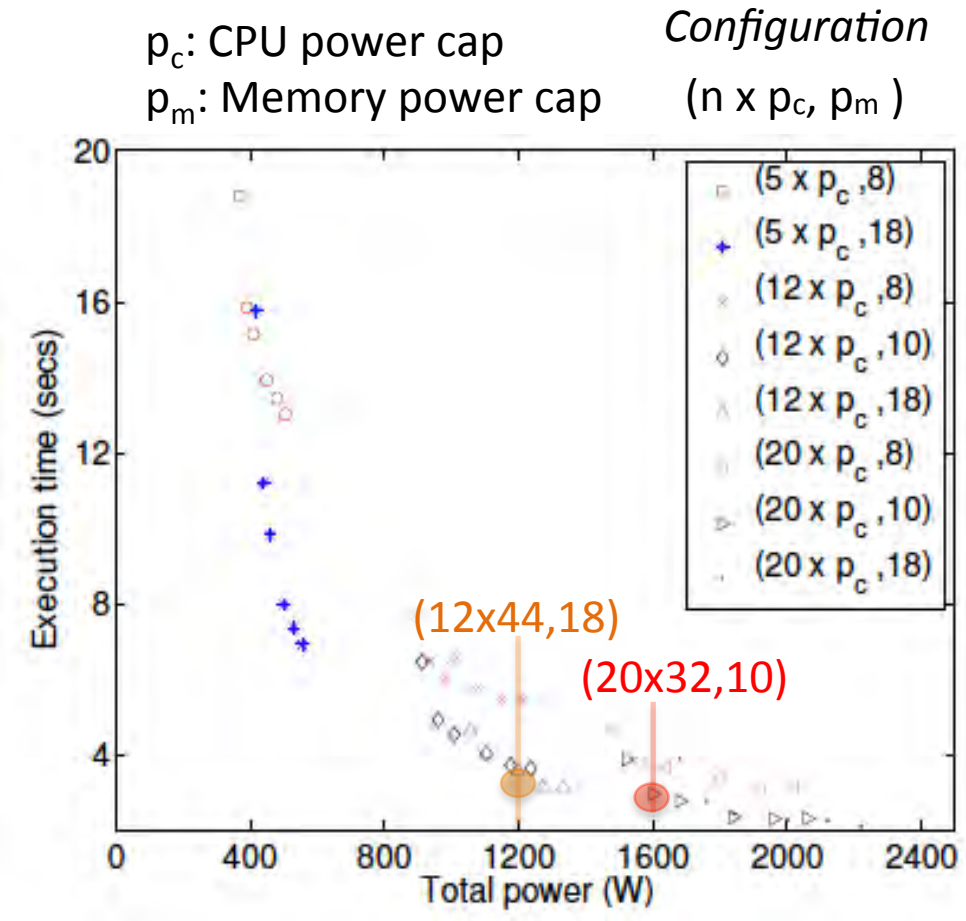
- Constrain power consumption of nodes
- *Overprovisioning* - Use more nodes than conventional data center for same power budget

Application Performance with Power

Application performance does not improve proportionately with increase in power cap

Run on larger number of nodes each capped at lower power level

Performance of LULESH at different configurations



Problem Statement

Maximizing Data Center Performance Under Strict Power Budget

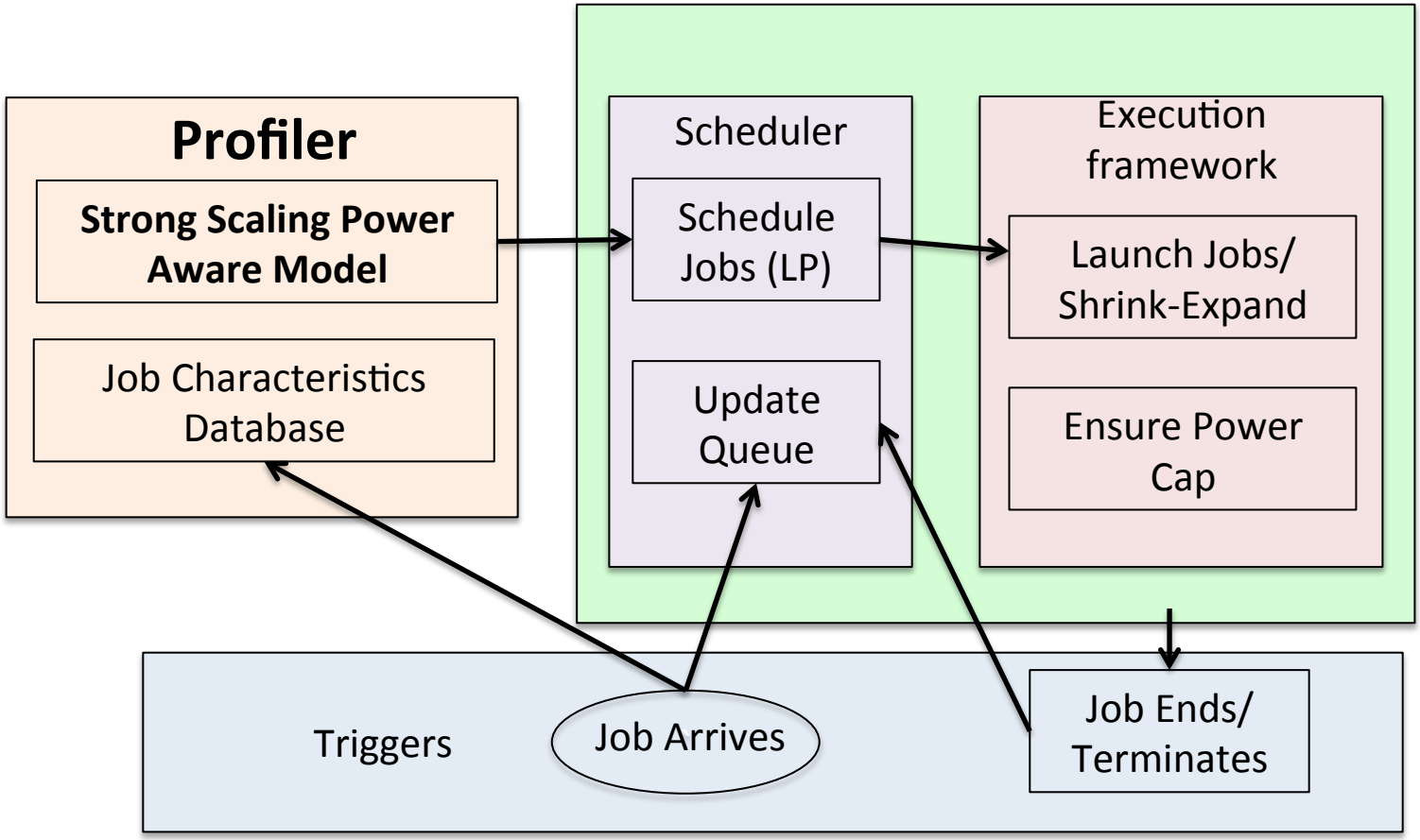
Data center capabilities

- ❑ Power capping ability
- ❑ Overprovisioning

Job features (Optional)

- ❑ Moldability
- ❑ Malleability
  - ➢ Charm++

PARM POWER-AWARE RESOURCE MANAGER



PASS Power Aware Strong Scaling Model

Execution time as a function of power and number of nodes

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

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

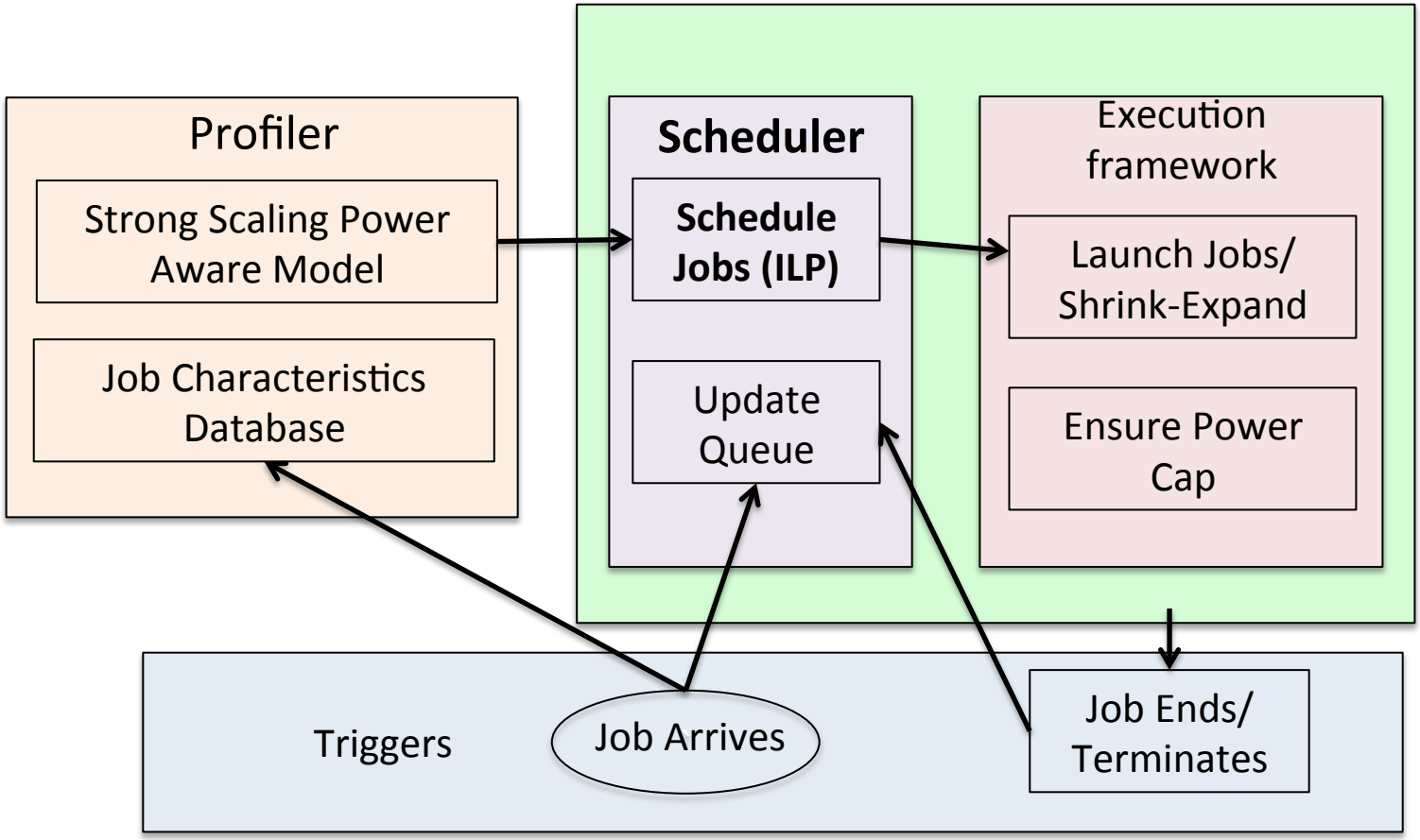
- W_{cpu}: CPU work
- T_{mem}: memory work
- T_h : minimum exec time

Power vs Frequency

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

- 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

PARM POWER-AWARE RESOURCE MANAGER



Scheduler - Integer Linear Program (ILP) 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

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

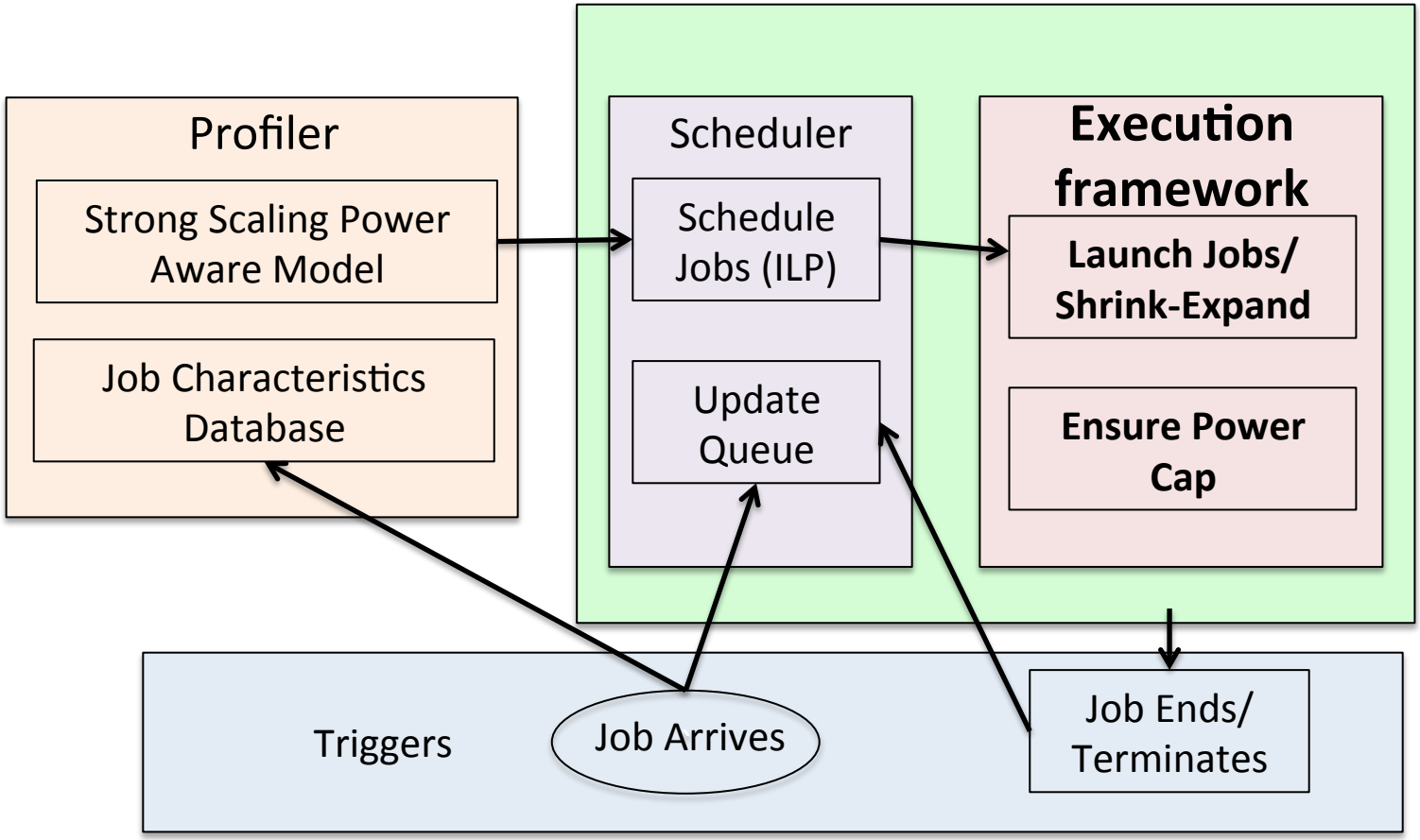
Disable Malleability (Optional)

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

Maximizing throughput makes online ILP optimization intractable, instead

maximize sum of power-aware speedup of selected jobs

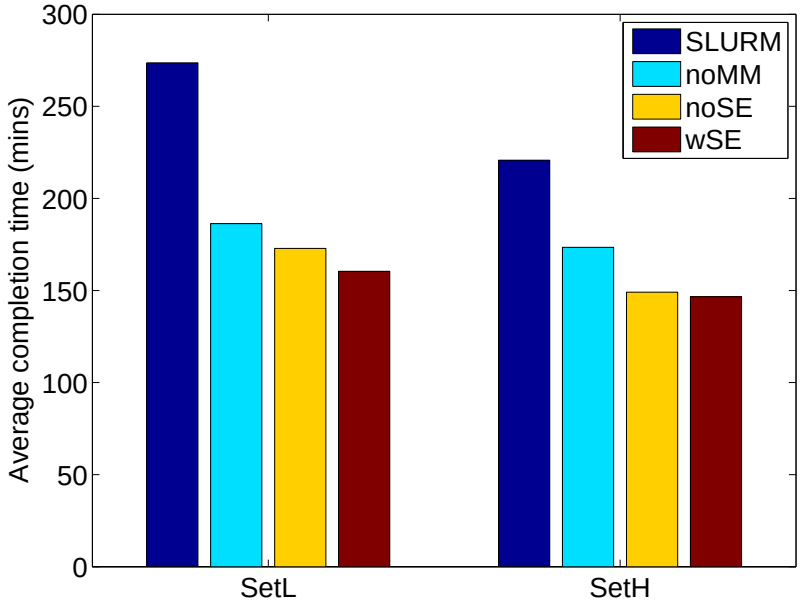
PARM POWER-AWARE RESOURCE MANAGER



Performance Results

Lulesh, AMR, LeanMD, Jacobi and Wave2D

38-node Intel Sandy Bridge Cluster, 3000W budget

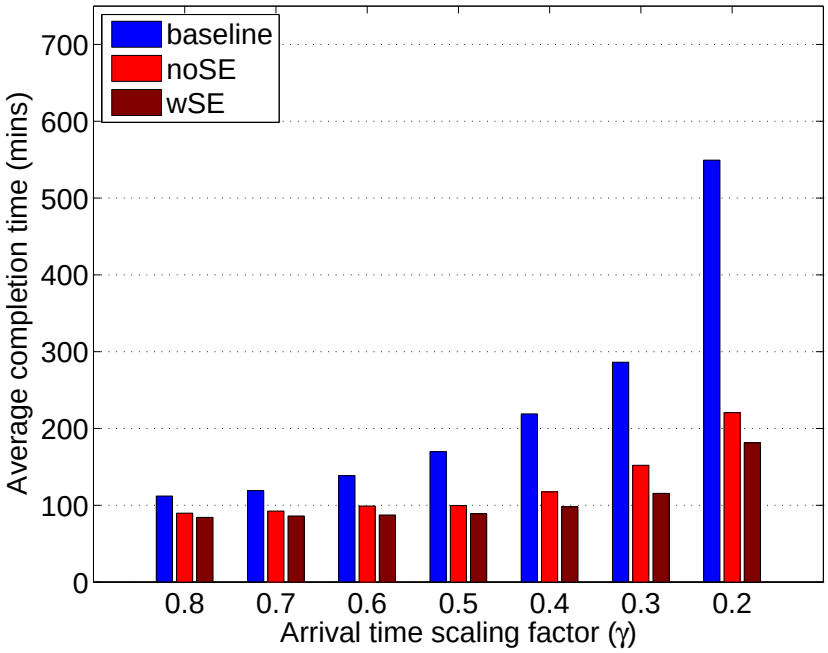


Description

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

1.7X improvement in throughput

Large Scale Projections



***Power Budget**: 4.75MW*

Description

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

- Arrival times multiplied by *γ*
  - Gives diversity in job arrival rates

5.2X speedup with wSE!

Comparison with Naïve Overprovisioning

CPU power cap (W)	30	40	50	60
Speedup of wSE over naive	4.32	1.86	2.33	5.25
Num. of nodes in naive strategy	55248	49493	44824	40960

Takeaways

- Significant improvement in throughputs
  - Power-aware characteristics (PASS model)
  - CPU power capping
  - Overprovisioning
- Sophisticated ILP scheduling methodology useful for resource assignment
- Adaptive runtime system further increases benefits by allowing malleability

References

http://charm.cs.uiuc.edu/research/energy

- [SC 14]. Maximizing Throughput of Overprovisioned Data Center Under a Strict Power Budget. Sarood et al. pdf
- [TOPC 14]. Power Management of Extreme-scale Networks with On/Off Links in Runtime Systems. Ehsan et al. pdf
- [SC 14]. Using an Adaptive Runtime System to Reconfigure the Cache Hierarchy. Ehsan et al. pdf
- [SC 13]. A Cool Way of Improving the Reliability of HPC Machines. Sarood et al. pdf
- [CLUSTER 13]. Thermal Aware Automated Load Balancing for HPC Applications. Harshitha et al. pdf
- [IEEE TC 12]. Cool Load Balancing for High Performance Computing Data Centers. Sarood et al. pdf
- [SC 12]. A Cool Load Balancer for Parallel Applications. Sarood et al. pdf
- [CLUSTER 12]. Meta-Balancer: Automated Load Balancing Invocation Based on Application Characteristics. Harshitha et al. pdf

Thank you!

PARM - POWER-AWARE RESOURCE MANAGER

Maximizing Data Center Performance Under Strict Power Budget

*Osman Sarood, **Akhil Langer**, Abhishek Gupta, Laxmikant Kale*

Parallel Programming Laboratory

Department of Computer Science

University of Illinois at Urbana-Champaign

25th March 2015

EE HPC WG Webinar on

Dynamic Power Management for MW-sized Supercomputer Centers

PARALLEL PROGRAMMING LABORATORY