POWER-AWARE JOB SCHEDULING

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

29th April 2014

PPL UIUC PARALLEL PROGRAMMING LABORATORY

12th Annual Workshop on Charm++ and its Applications

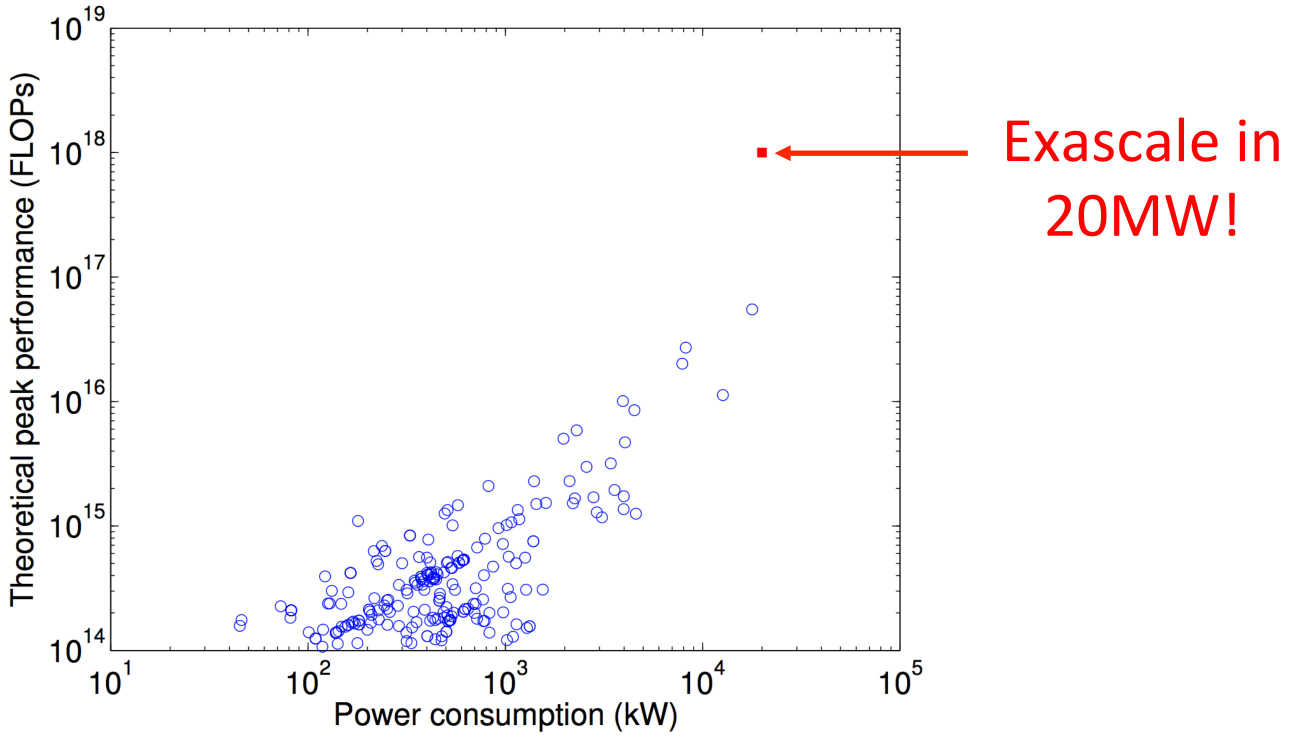
Major Challenges to Achieve Exascale[1]

- ☐ Energy and Power Challenge
- ☐ Memory and Storage Challenge
- ☐ Concurrency and Locality Challenge
- ☐ Resiliency Challenge

Kogge, Peter, et al. "Exascale computing study: Technology challenges in achieving exascale systems." (2008).

Major Challenges to Achieve Exascale[1]

Power consumption for Top500



Exascale in 20MW!

Kogge, Peter, et al. "Exascale computing study: Technology challenges in achieving exascale systems." (2008).

Data Center Power

How is data center power need calculated?

- ☐ using Thermal Design Power (TDP) of nodes

However, TDP is hardly reached!!

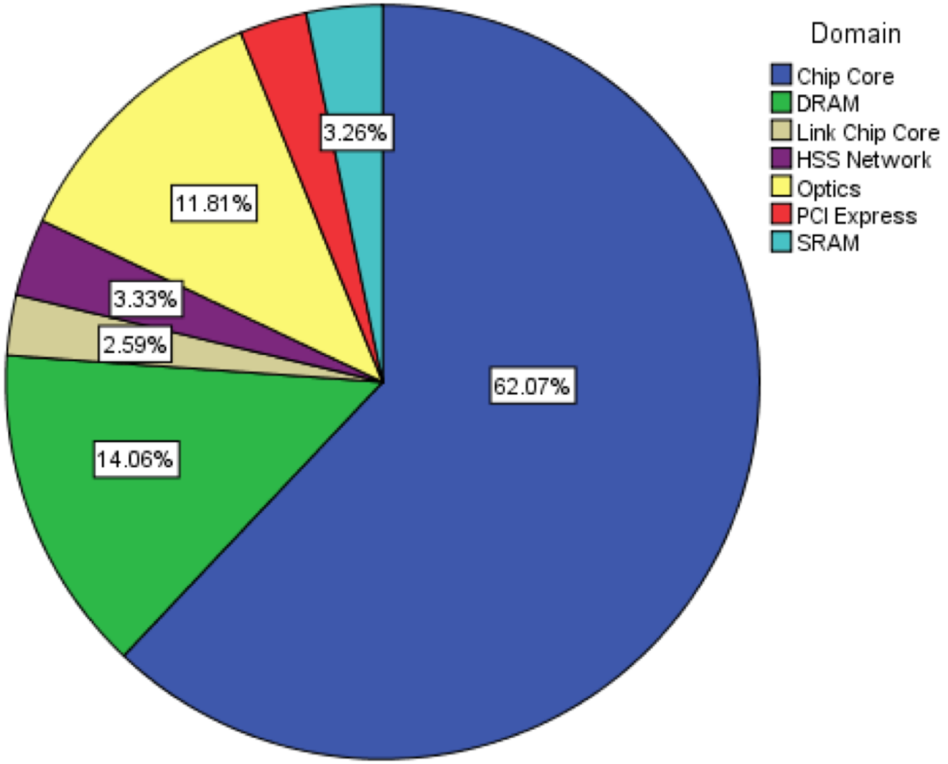
Solution

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

Distribution of Node Power Consumption

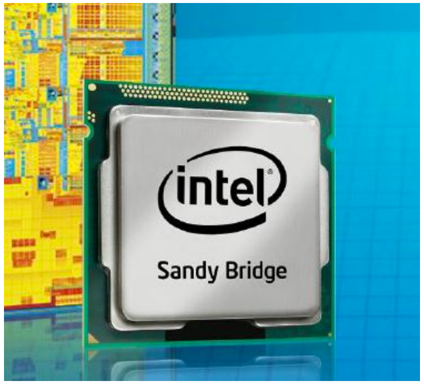
Power distribution for BG/Q processor on Mira

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



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

Constraining CPU/Memory Power



Intel Sandy Bridge

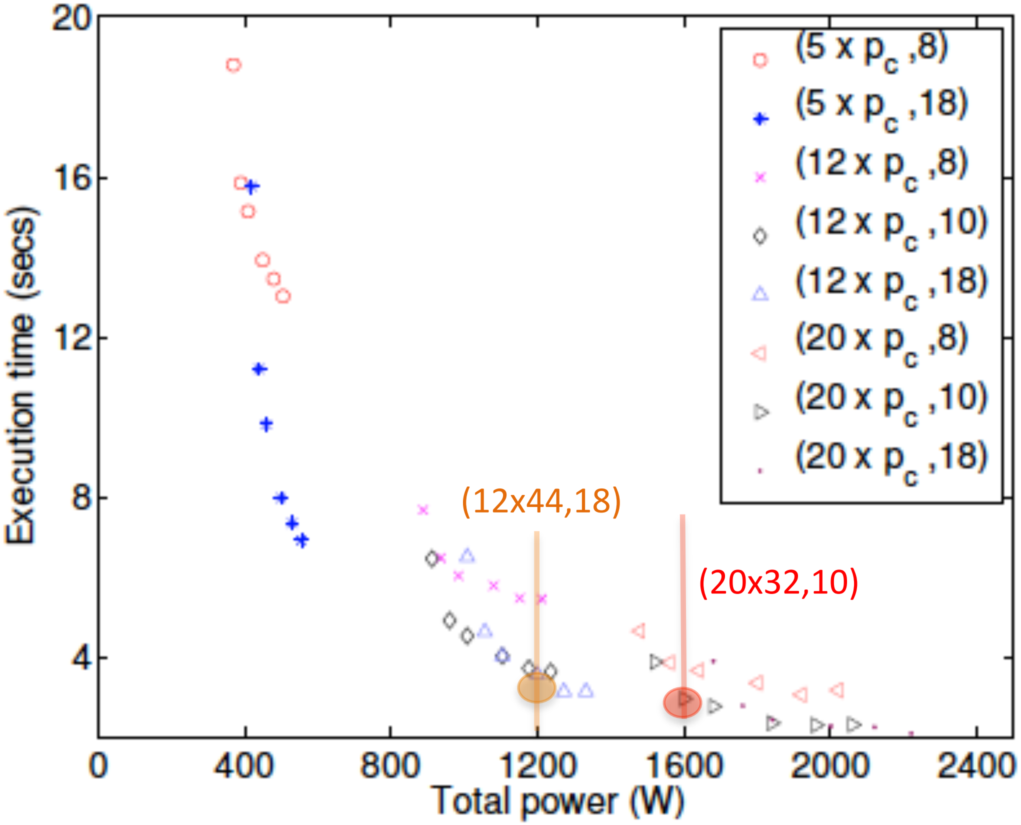
- Running Average Power Limit (RAPL) library
  - measure and set CPU/memory power

Application Performance with Power

- Application performance does not improve proportionately with increase in power cap
- Better is to run on larger number of nodes each capped at lower power level

p_c: CPU power cap
P_m:Memory power cap

Configuration
$(n \times p_c, p_m)$



Performance of LULESH at different configurations

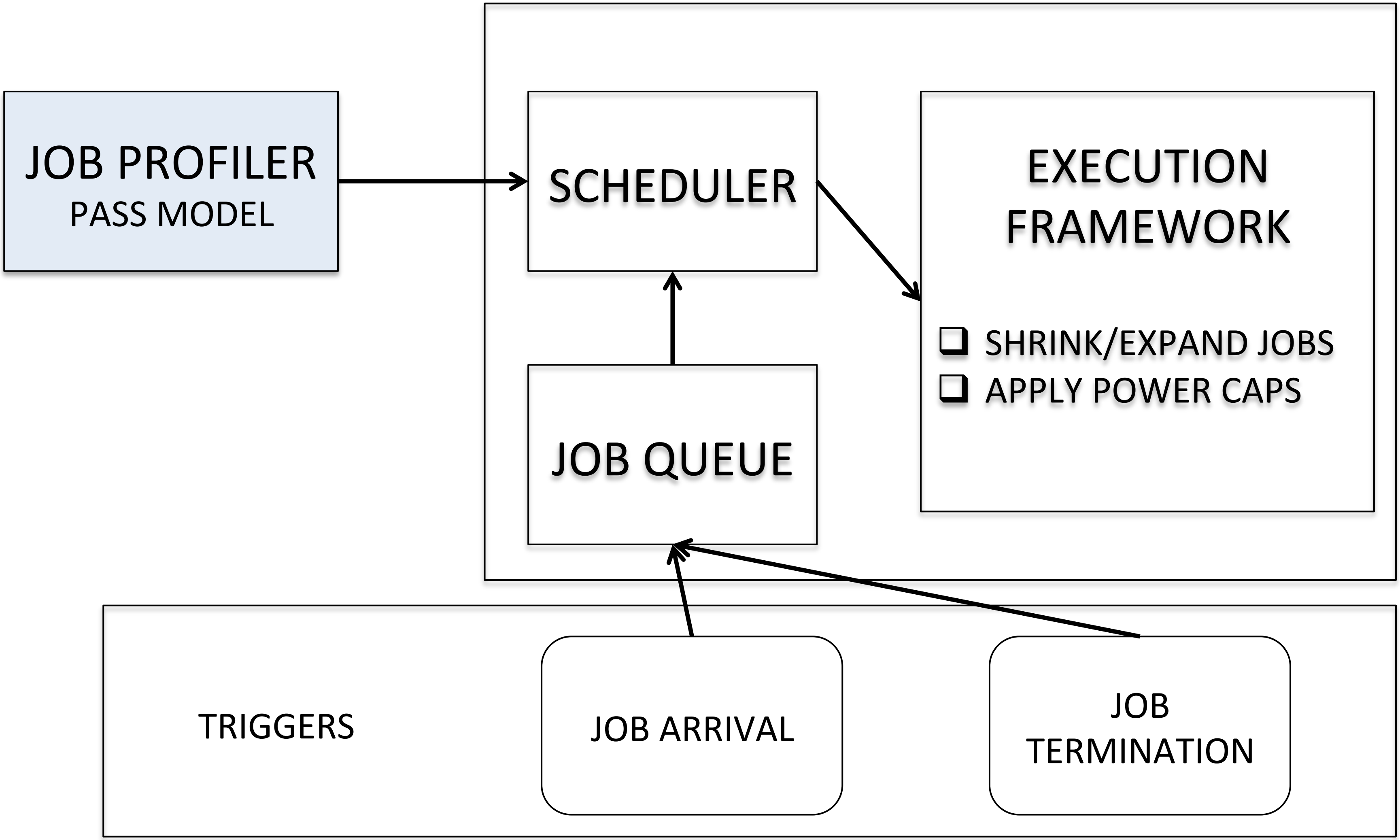
Problem Statement

Maximizing Data Center Performance Under Strict Power Budget

Data center capabilities and job features

- ☐ Power capping ability
- ☐ Overprovisioning
- ☐ Moldability (Optional)
- ☐ Malleability (Optional)
    - ➢ Charm++
    - ➢ Dynamic MPI

Power Aware Resource Manager (PARM)



JOB PROFILER
PASS MODEL

SCHEDULER

EXECUTION FRAMEWORK

- ☐ SHRINK/EXPAND JOBS
- ☐ APPLY POWER CAPS

JOB QUEUE

TRIGGERS

JOB ARRIVAL

JOB TERMINATION

JOB PROFILER

- ❑Measure job performance at various scales and cpu power caps

- ❑Power Aware Strong Scaling (PASS) Model
  - ➢Predict job performance at any (n, p)

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

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

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

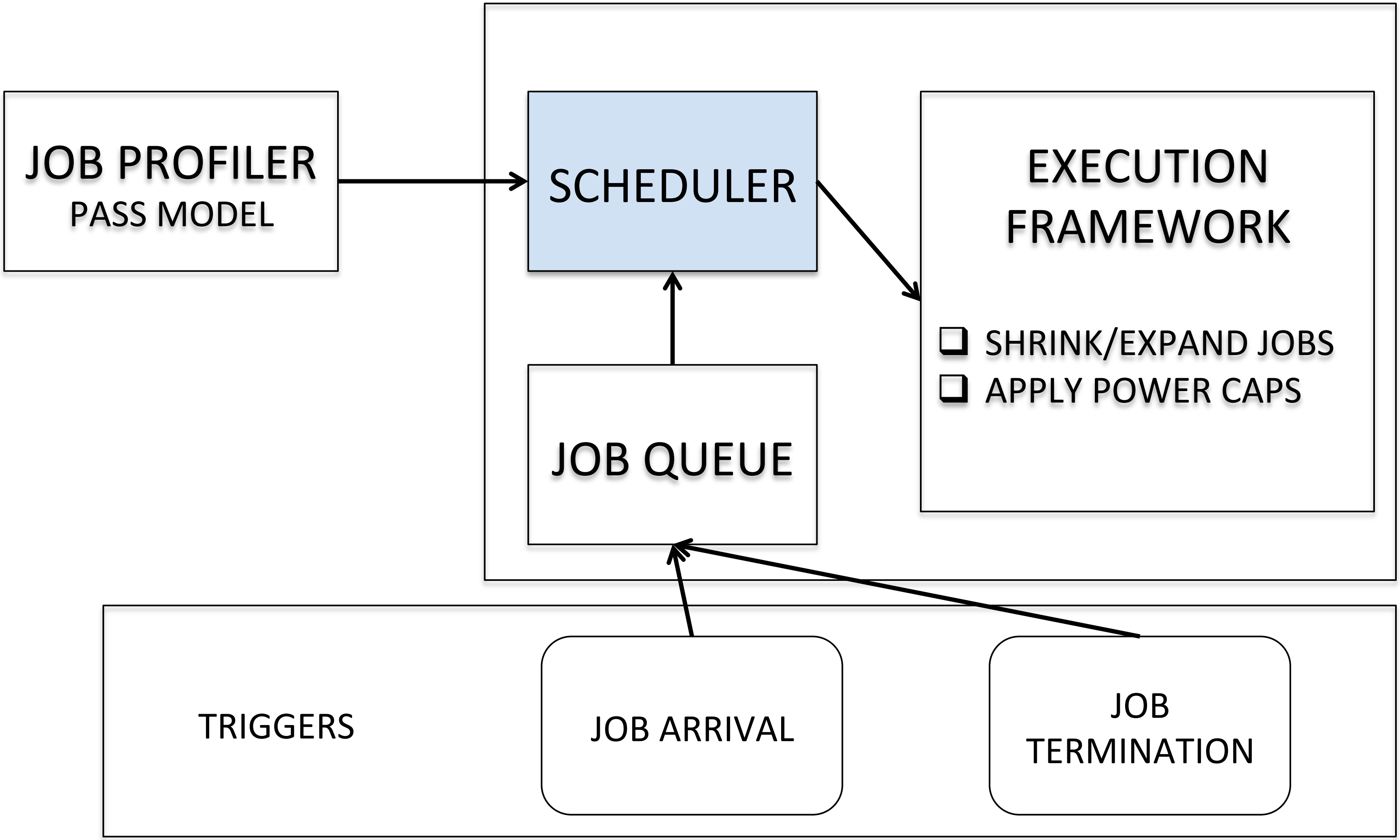
- ☐ p_{core}: core power
- ☐ g_i: cost level I cache access
- ☐ L_i: #level I accesses
- ☐ g_m: cost of mem access
- ☐ M: #mem accesses
- ☐ p_{base}: idle power

Frequency vs 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

Power Aware Resource Manager (PARM)



JOB PROFILER

PASS MODEL

SCHEDULER

EXECUTION FRAMEWORK

- ☐ SHRINK/EXPAND JOBS
- ☐ APPLY POWER CAPS

JOB QUEUE

TRIGGERS

JOB ARRIVAL

JOB TERMINATION

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)

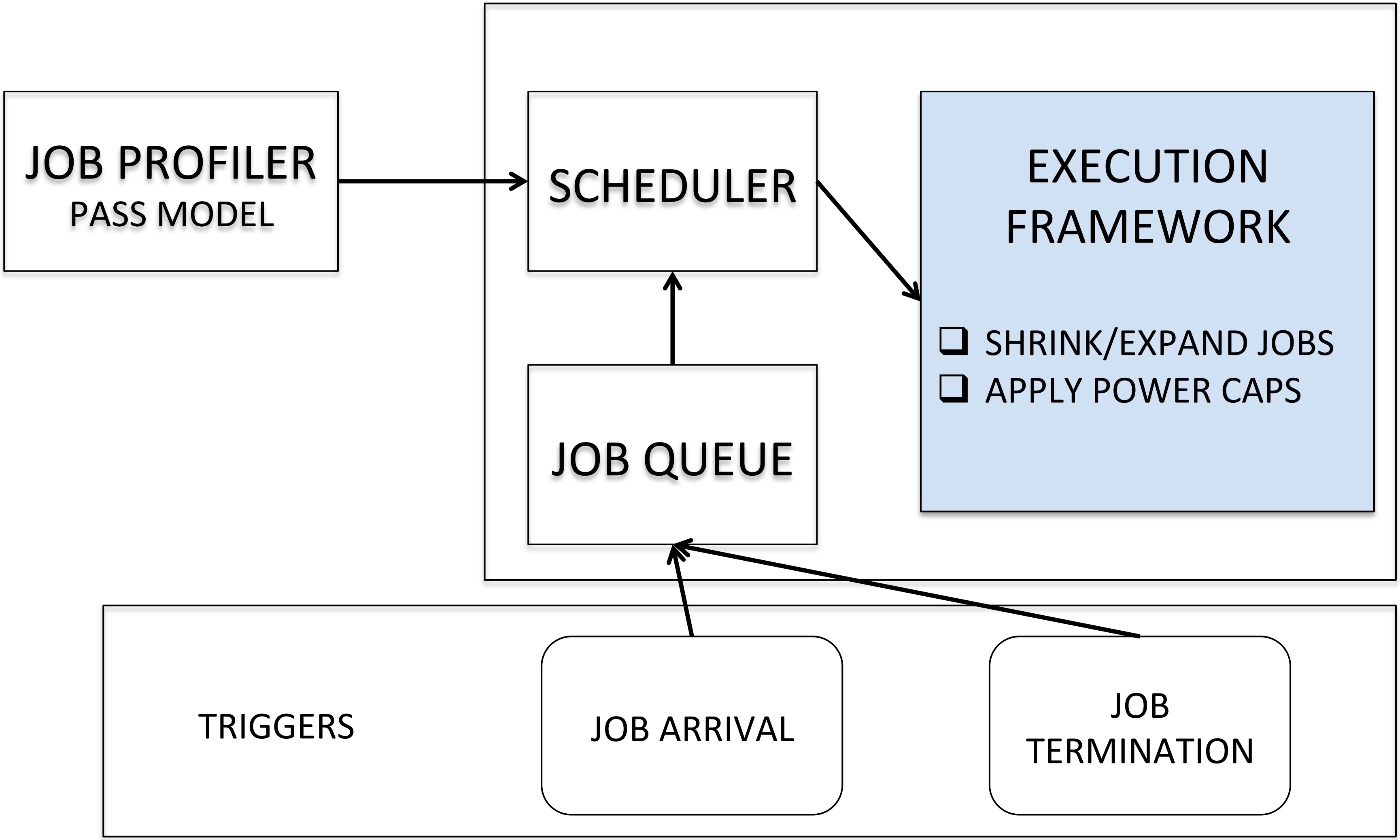
$$\sum_{n \in N_j} \sum_{p \in P_j} n x_{j,n,p} = n_j \qquad \forall j \in \mathcal{I}$$

Scheduler: Objective Function

- Maximizing throughput makes ILP optimization infeasible
- Maximize sum of power-aware speedup of selected jobs:

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

Power Aware Resource Manager (PARM)



JOB PROFILER
PASS MODEL

SCHEDULER

EXECUTION FRAMEWORK

- ☐ SHRINK/EXPAND JOBS
- ☐ APPLY POWER CAPS

JOB QUEUE

TRIGGERS

JOB ARRIVAL

JOB TERMINATION

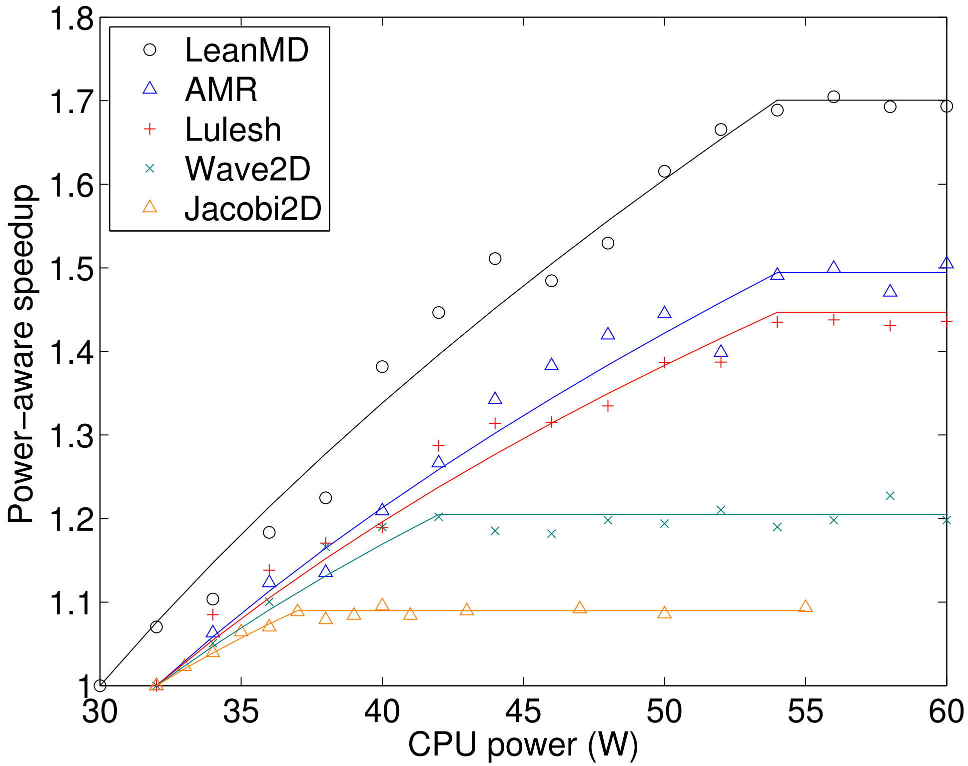
Experimental Setup

- Applications
  - Memory-intensive
    - Jacobi and Wave2D
  - Computation-intensive
    - LeanMD
  - Mixed
    - AMR and Lulesh
- Testbed
  - 38-node Intel Sandy Bridge
  - 6 physical cores, 16GB RAM
  - Power capping using RAPL
  - CPU power cap range [25-95]W
- Job Dataset
  - β corresponds to CPU sensitivity
  - SetL: Mix of apps with average β=0.1
  - SetH: Mix of apps with average β=0.27
- Power Budget
  - CPU power levels={30, 32, 34, 39, 45, 55}W
  - Node power consumption= 116W
  - Power Budget = 3000W
  - #nodes in traditional data center = 28

Estimating Performance using PASS

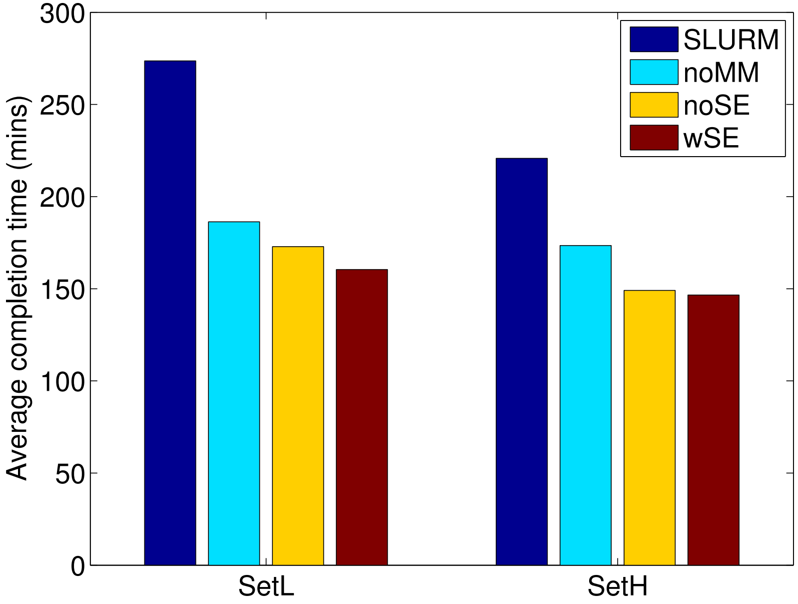
Model Parameters

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



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

Large Scale Projections

- SLURM simulator vs PARM simulator
- Modeling cost of shrinking and expansion of jobs
  - Boot times

$$t_b(\text{in seconds}) = (n_t - n_f) * 0.01904 + 72.73$$

  - Communication cost for data transfer

$$t_c = \frac{(\frac{m_j}{n_f} - \frac{m_j}{n_t}) * n_f}{2 * b * n_f^{\frac{2}{3}}}$$

  - Total cost

$$t_{se} = t_c + t_b$$

Large Scale Projections Experimental Setup

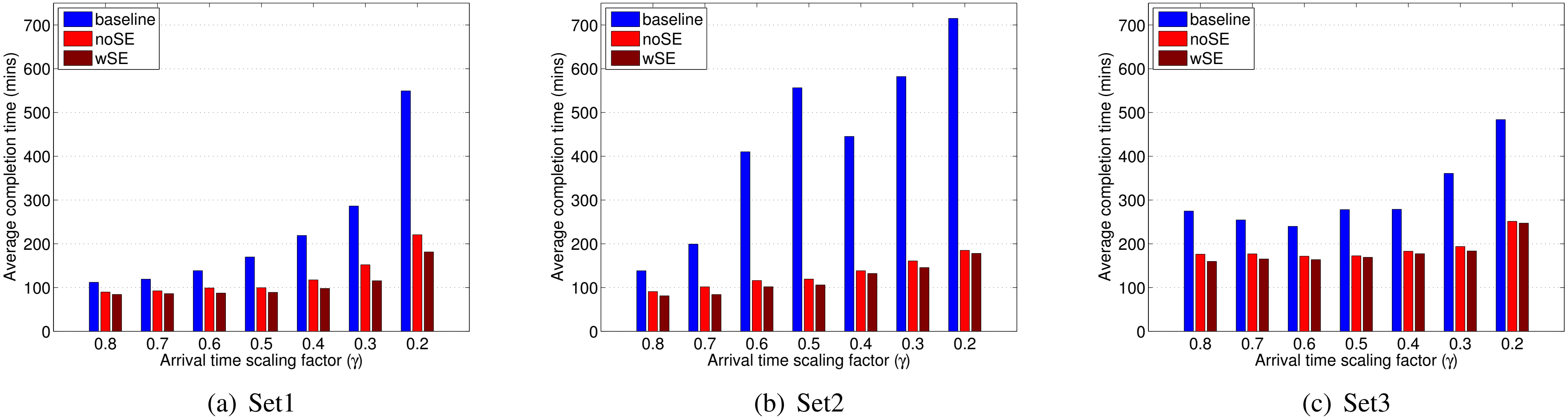
- ❑ Job Datasets
  - ➢ Intrepid job traces
  - ➢ 3 subsets: Set 1, Set 2, Set3
  - ➢ 1000 jobs
- ❑ Application Characteristics
  - ➢ Model parameters chosen randomly from range defined by computationally and memory intensive apps
- ❑ Node Range for Moldable/ Malleable jobs
  - ➢ min nodes = θ*max(N)

    θ ε[0.2, 0.6]
- ❑ Power Budget
  - ➢ 40,960 nodes -> 4.75MW
  - ➢ CPU power levels ={30,33,36,44,50,60}W

Large Scale Projections Performance

Description

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

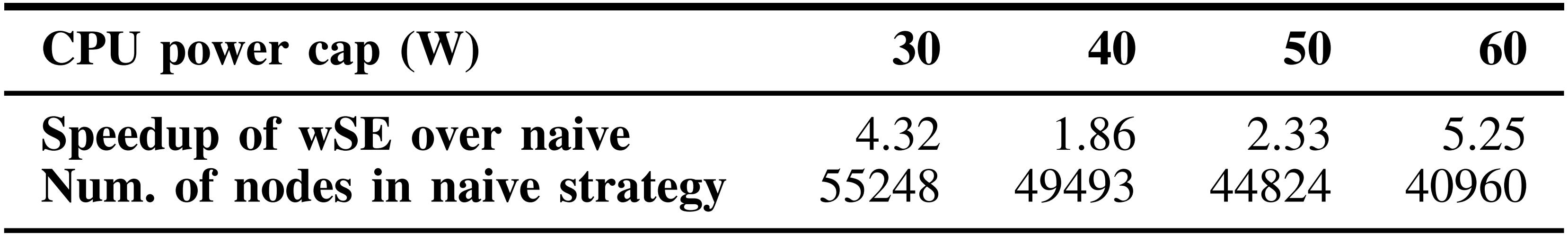


(a) Set1 (b) Set2 (c) Set3

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

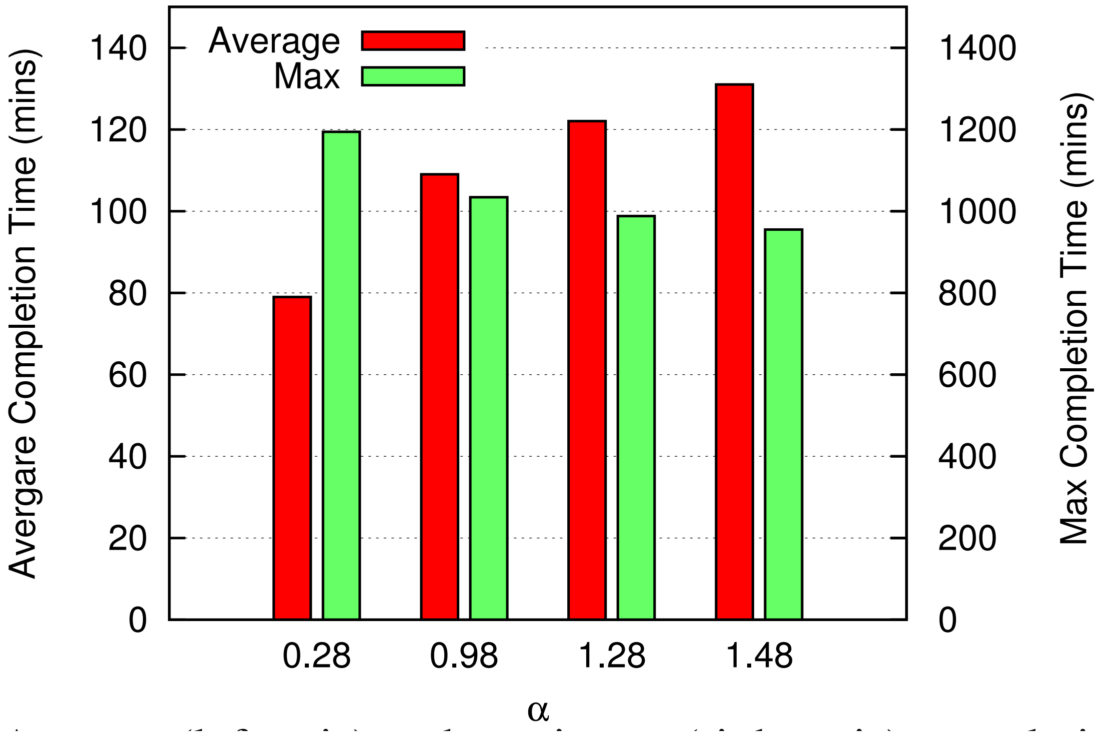
5.2X speedup with wSE!

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

Tradeoff between Throughput and Job Fairness



Objective function multiplier: **ω^{α}**

CONCLUSIONS/TAKEAWAYS

Conclusion

- ❑ 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
- ❑ Non-malleable jobs also benefit

Future Work

- ❑ Enable/disable caches
- ❑ Thermal constraints
    - ➢ To improve system reliability and improve cooling costs
- ❑ Rich support for user priorities

THANK YOU!

POWER-AWARE JOB SCHEDULING

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

29th April 2014

PPL UIUC
Parallel Programming Laboratory

12th Annual Workshop on Charm++ and its Applications