A POWER CAPPING APPROACH FOR HPC SYSTEM DEMAND RESPONSE

Kishwar Ahmed, Research Aide
MCS Division
Argonne National Laboratory, IL, USA

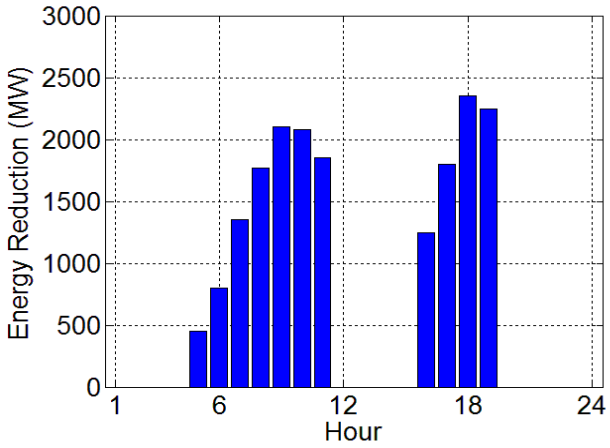
Mentor: Kazutomo Yoshii
MCS Division
Argonne National Laboratory, IL, USA

Outline

- Motivation
  - Why demand response is important?
  - HPC system as demand response participant?
  - Related works
- Applications, Tools and Testbed
- Model and Simulator
  - How we model HPC demand response participation?
  - How we simulate the proposed model?
  - Cooling energy model
  - How we compare our model with existing policies?
- Conclusions

What is Demand Response (DR)?

- **Overall objective**: Enable an overall HPC system demand response participation through job scheduling and resource allocation (e.g., power capping)
- DR: Participants reduce energy consumption
  - During transient surge in power demand
  - Other emergency events
- A DR example:
  - Extreme cold in beginning of January 2014
  - Closure of electricity grid
  - Emergency demand response in PJM and ERCOT



Energy reduction target at PJM on January 2014

Demand Response Getting Popular!

SUSTAINABILITY

Why Apple Is Getting into the Energy Business

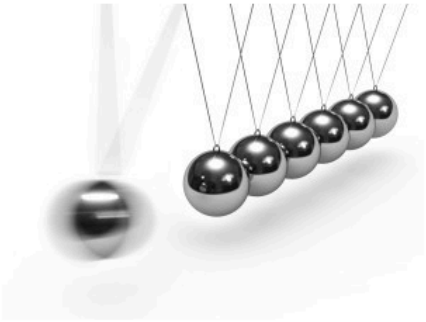
by Peter Fox-Penner

NOVEMBER 25, 2016

But solar electricity is only the beginning of the future energy marketplace. Many companies are already in markets where "demand-response" contracts enable them to sell the right to manage a portion of their power use, allowing them to be paid for reducing their energy during hours when the spot price of power is high. In the future, in addition to selling actual

Equinix 'in R&D phase' of demand response experiments

OCTOBER 16, 2015 BY BRENDAN COYNE — 1 COMMENT



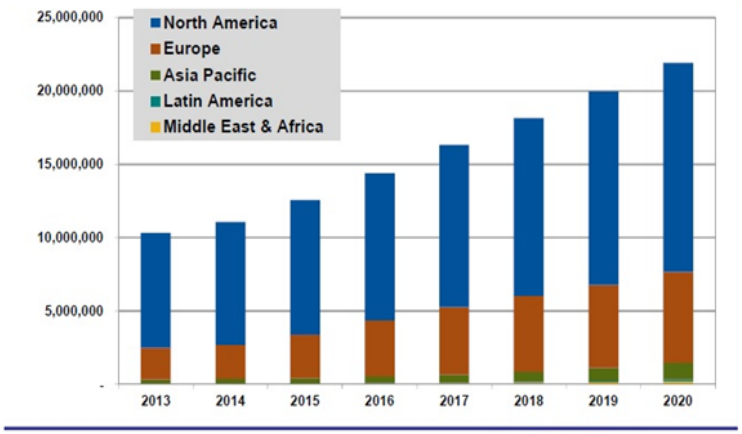
Equinix is "in the R&D phase" of testing demand response technologies, according to UK MD Russell Poole.

National Grid wants more businesses to help balance the electricity system by turning power off or on, or adjusting loads, in return for payment. However, mission critical sites have traditionally been reluctant to increase any perceived risk factors for relatively low rewards. National Grid though, is rethinking its mechanisms and contract structures.

DEMAND RESPONSE WILL DOUBLE BY 2020: HERE'S WHY

BY JESSICA KENNEDY SEPTEMBER 3, 2014

Chart 1.1 Total DR Sites by Region, World Markets: 2013-2020



(Source: Navigant Research)

HPC System as DR Participant?

- HPC system is a major energy consumer
  - China's 34-petaflop Tianhe-2 consumes 18MWs of power
    - Can supply small town of 20,000 homes
  - The power usage of future HPC system is projected to increase
    - Future exascale supercomputer has power capping limit
    - But not possible with current system architecture
- Demand response aware job scheduling envisioned as possible future direction by national laboratories ["Intelligent Job Scheduling" by Gregory A. Koenig]

HPC System as DR Participant? (Contd.)

- A number of recent surveys on possibility of supercomputer's participation in DR program
- Patki et al. (in 2016)
  - A survey to investigate demand response participation of 11 supercomputing sites in US
  - "*…SCs in the United States were interested in a tighter integration with their ESPs to improve Demand Management (DM).*"
- Bates et al. (in 2015)
  - "*…the most straightforward ways that SCs can begin the process of developing a DR capability is by enhancing existing system software (e.g., job scheduler, resource manager)*"

Power Capping

- What is power capping?
  - Dynamic setting of power budget to a single server
- Power capping is important
  - To achieve global power cap for the cluster
  - Intel's Running Average Power Limit (RAPL) can combine good properties of DVFS
- Power capping is common in modern processors
  - Intel processors support power capping through RAPL interface
  - Intel Node Manager, an Intel server firmware feature, gives capability to limit power at the system, processor and memory level

Related Works

- Job scheduling and resource provisioning in HPC
  - [Yang et al.] Reduce energy cost through executing
    - Low power-consuming jobs during on-peak periods
    - High power-consuming jobs during off-peak periods
- Green HPC
  - Reducing brown energy consumption
    - GreenPar: adopts different job scheduling strategies (e.g., dynamic job migration, resource allocation)
- Energy saving techniques in HPC system
  - CPU MISER
    - DVFS-based power management scheme
  - Adagio
    - Exploits variation in the energy consumption during computation and communication phases

Related Works (Contd.)

- Data center and smart building demand response
  - Workload scheduling: such as load shifting in time, geographical load balancing
  - Resource management: server consolidation, speed-scaling
- However,
  - These approaches are applicable for internet transaction-based data center workload
  - Service time for data center workload are assumed uniform and delay-intolerant
- HPC system demand response
  - Recently, we proposed an HPC system demand response model
- However the current work,
  - Does not consider real-life applications running on clusters
  - Considers DVFS, not power capping
  - Does not perform job allocation to processors
  - Does not consider cooling energy model

Outline

- Motivation
  - Why demand response is important?
  - HPC system as demand response participant?
  - Existing works
- Applications, Tools and Testbed
- Model and Simulator
  - How we model HPC demand response participation?
  - How we simulate the proposed model?
  - Cooling energy model
  - How we compare our model with existing policies?
- Conclusions

Applications and Benchmarks

Benchmark Type	Description	Applications	Application Description
Scalable science benchmarks	Expected to run at **full scale** of the CORAL systems	HACC, **Nekbone**, etc.	Compute intensity, small messages, allreduce
Throughput benchmarks	Represent large ensemble runs	UMT2013, AMG2013, SNAP **LULESH**, etc.	Shock hydrodynamics for unstructured meshes.
Data Centric Benchmarks	Represent emerging **data intensive** workloads – Integer operations, instruction throughput, indirect addressing	Graph500, **Hash,** etc.	Parallel hash benchmark
Skeleton Benchmarks	Investigate various **platform characteristics** including network performance, threading overheads, etc.	CLOMP, **XSBench,** etc.	Stresses system through memory capacity.

Applications and Benchmarks (Contd.)

Benchmark Type	Description	Applications	Application Description
NAS Parallel Benchmarks	A small set of programs designed to help evaluate the **performance of parallel supercomputers**	IS, EP, FT, **CG**	CG - Conjugate Gradient method
Dense-matrix multiply benchmarks	A simple, multi-threaded, dense-matrix multiply benchmark. The code is designed to measure the sustained, floating-point computational rate of a single node	MT-DGEMM, Intel MKL DGEMM	MT-DGEMM: The source code given by NERSC (National Energy Research Scientific Computing Center) Intel MKL DGEMM: The source code given by Intel to multiply matrix
Processor Stress Test Utility	N/A	FIRESTARTER	Maximizes the energy consumption of 64-Bit x86 processors by generating heavy load on the execution units as well as transferring data between the cores and multiple levels of the memory hierarchy.

Measurement Tools

- etrace2
  - Reports energy and execution time of an application
  - Relies on the Intel RAPL interface
  - Developed under DOE COOLR/ARGO project
  - Used inside Chameleon cluster

- An example run

```
../tools/pycoolr/clr_rapl.py --limitp=140
etrace2 mpirun -n 32 bin/cg.D.32

../tools/pycoolr/clr_rapl.py --limitp=120
etrace2 mpirun -n 32 bin/cg.D.32
```

```
Output:
p0 140.0
p1 140.0

 NAS Parallel Benchmarks 3.3 -- CG Benchmark

 Size:    1500000
 Iterations:   100
 Number of active processes:    32
 Number of nonzeroes per row:       21
 Eigenvalue shift: .500E+03

   iteration          ||r||                 zeta
       1       0.73652606305295E-12   499.9996989885352
...
# ETRACE2_VERSION=0.1
# ELAPSED=1652.960293
# ENERGY=91937.964940
# ENERGY_SOCKET0=21333.227051
# ENERGY_DRAM0=30015.779454
# ENERGY_SOCKET1=15409.632036
# ENERGY_DRAM1=25180.102634
```

Measurement Tools (Contd.)

- pycoolr
  - Measure processor power usage and processor temperature
  - Use Intel RAPL capability to measure power usage
  - Power capping limit change capability
  - Reports data in json format
- An example run

```
../tools/pycoolr/clr_rapl.py --limitp=140
mpirun -n 32 ./nekbone ex1

./coolrs.py > nekbone.out
```

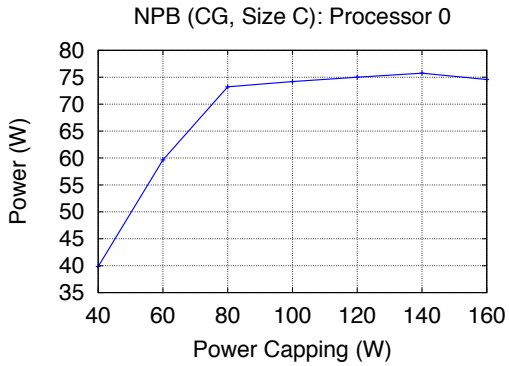
{"sample":"**temp**","time":
1499822397.016,"node":"protos","p0":{**"mean":
34.89** ,"std":1.20 ,"min":33.00 ,"max":36.00 ,"0":
33,"1":33,"2":35,"3":36,"4":35,"5":36,"6":36,"7":
34,"pkg":36}}

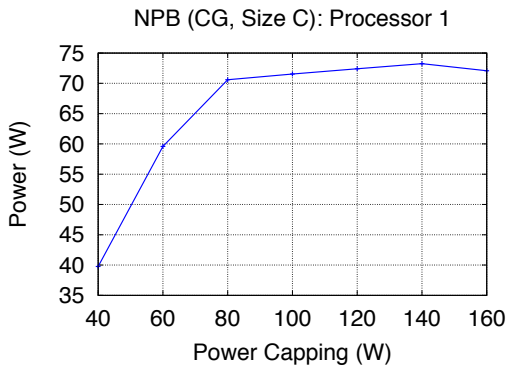
{"sample":"**energy**","time":
1499822397.017,"node":"protos","label":"run","energ
y":{"p0":57706365709,"p0/core":4262338717,"p0/
dram":62433931283,"p1":15467688771,"p1/core":
18329000806,"p1/dram":55726072673},"power":
{**"p0":16.3**,"p0/core":4.6,"p0/dram":1.4,**"p1":
16.7**,"p1/core":4.8,"p1/dram":0.9,"total":
35.3},"powercap":{"p0":140.0,"p0/core":0.0,"p0/
dram":0.0,"p1":140.0,"p1/core":0.0,"p1/dram":0.0}}

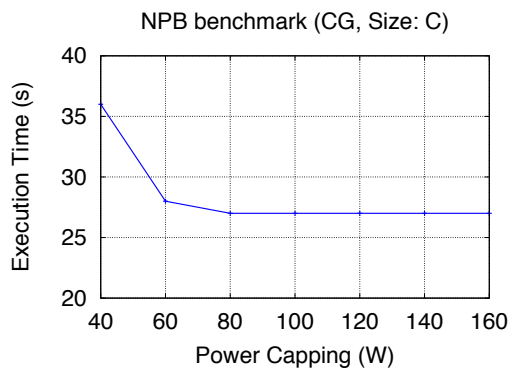
Experimental Testbeds

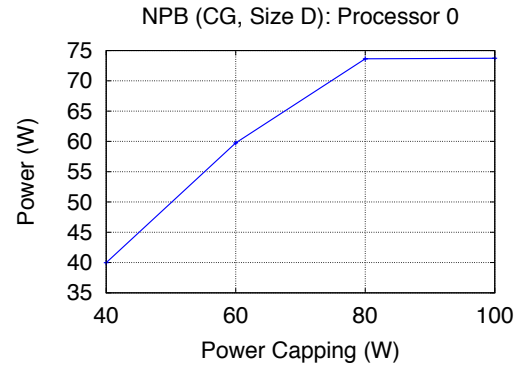
- Chameleon cluster
  - An experimental setup for large-scale cloud research
  - Deployed at the University of Chicago and the Texas Advanced Computing Center
  - Hosts around 650 multi-core cloud nodes
  - Used 6-node cluster to run applications
  - However, power capping experiments still not supported; limited by Dell server's BIOS
- Experimental node@Tinkerlab
  - Intel Sandy Bridge processor
  - Provide power-capping capability
  - Consists of 2 processors with 32 cores
- JLSE@ANL
  - We ran applications on multiple nodes and measured power and temperature data

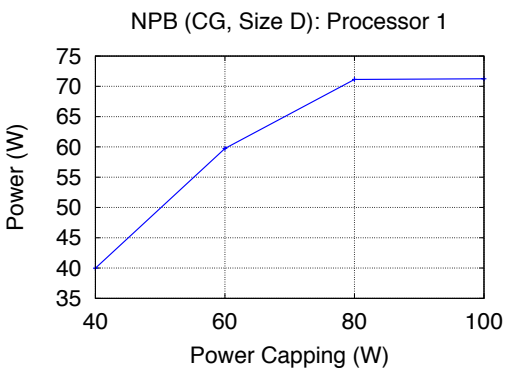
Experiment Results@Tinkerlab

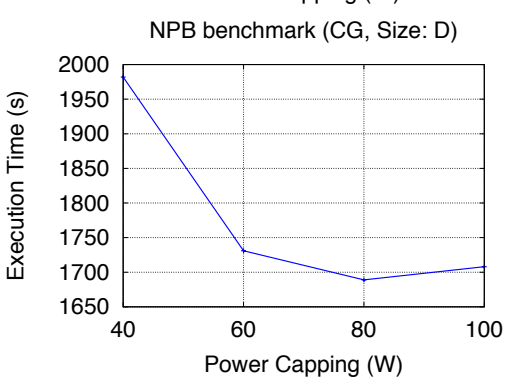


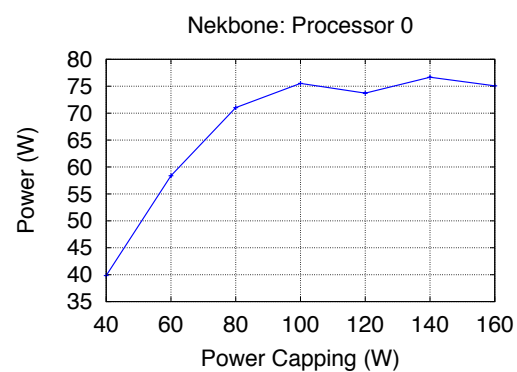


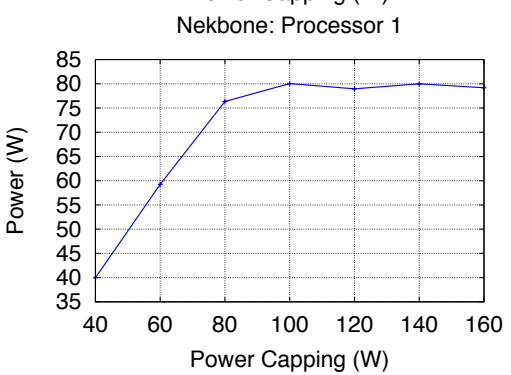


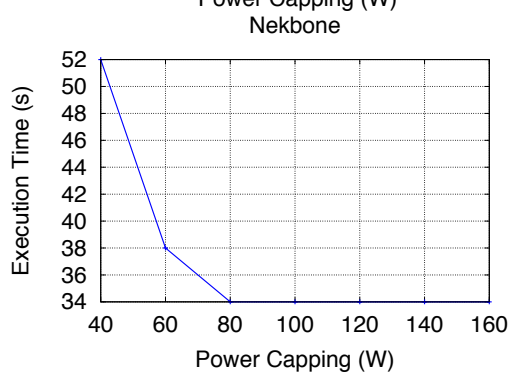




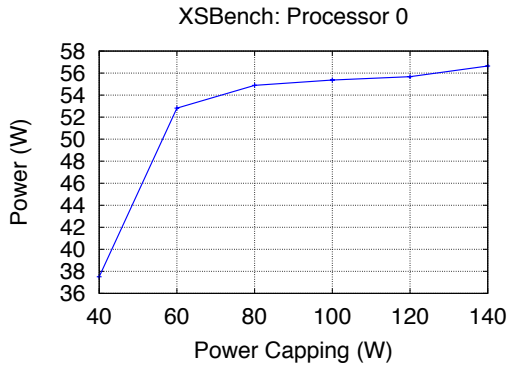


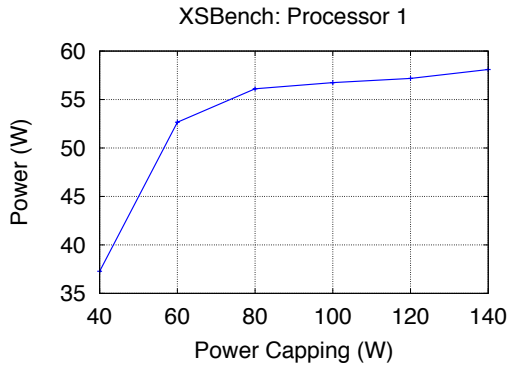


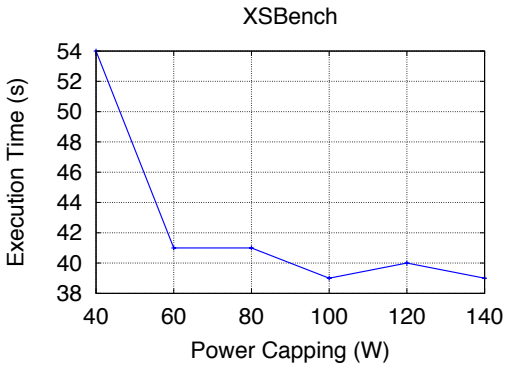


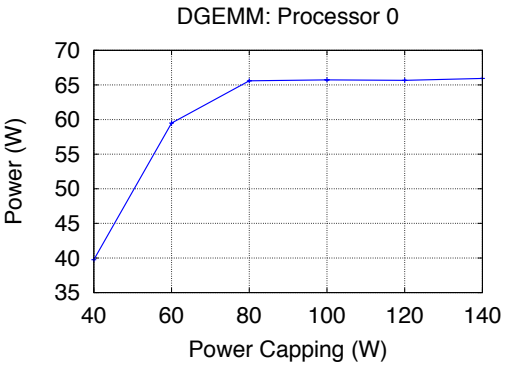


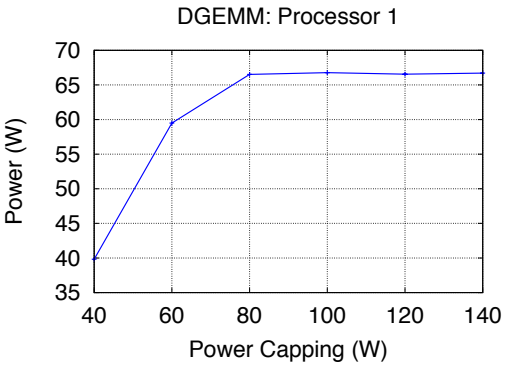
Experiment Results@Tinkerlab (Contd.)

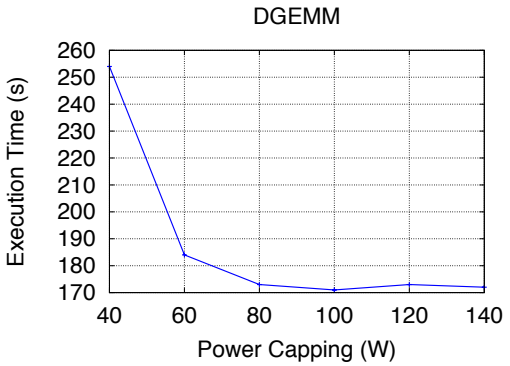


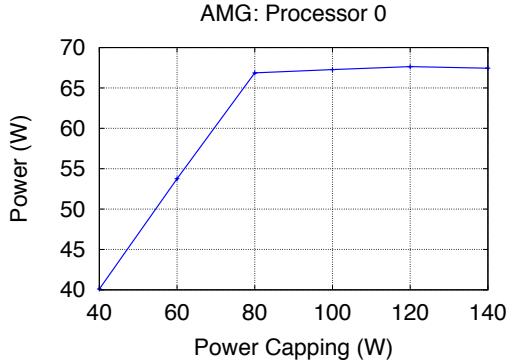


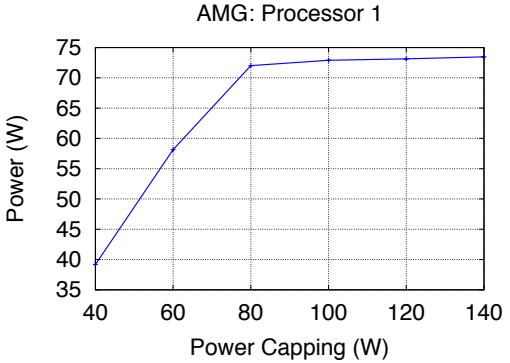


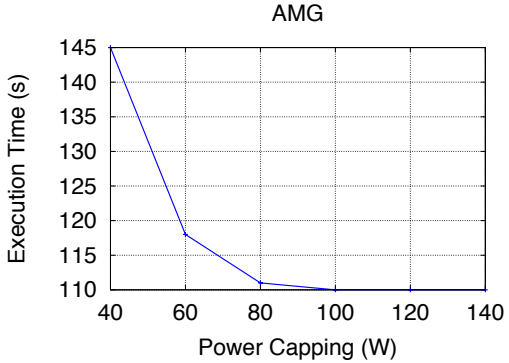




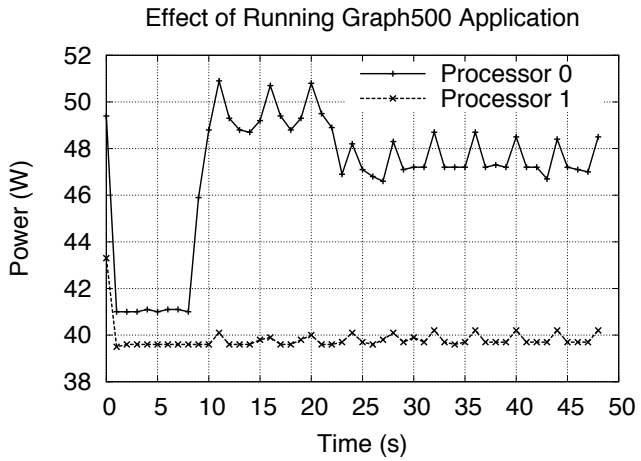


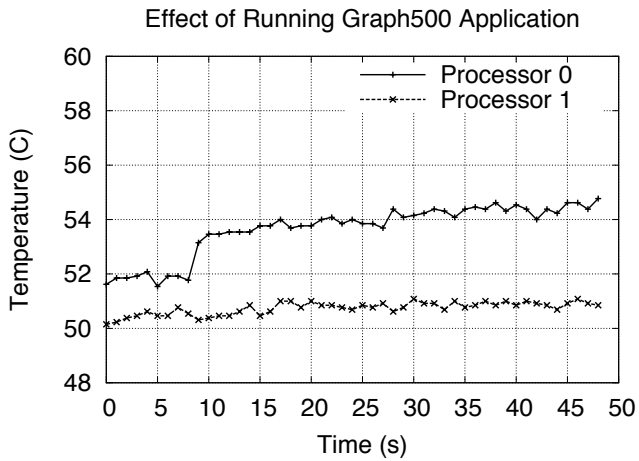


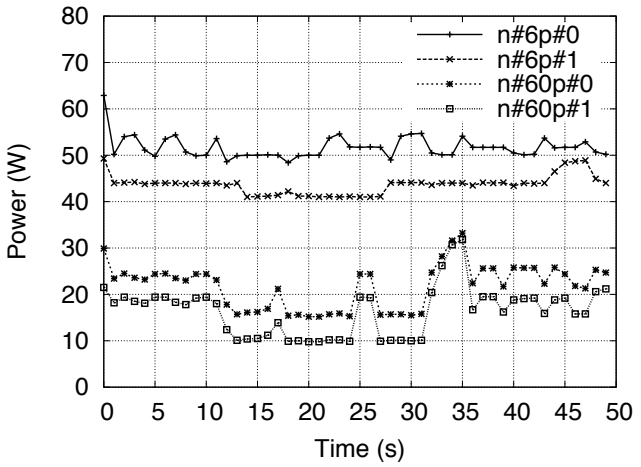


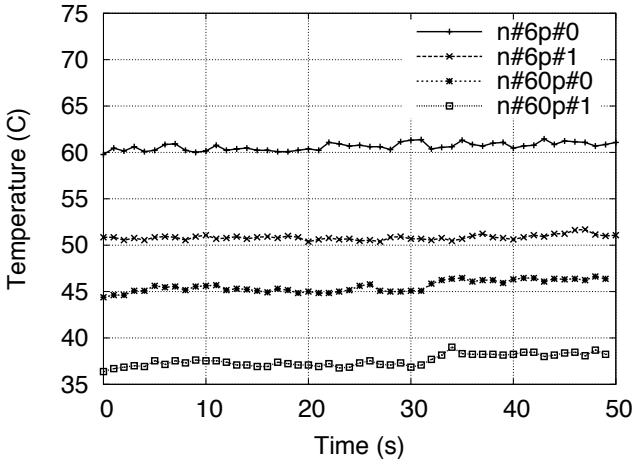


Experiment Results@Chameleon









Outline

- Motivation
  - Why demand response is important?
  - HPC system as demand response participant?
  - Existing works
- Applications, Tools and Testbed
- Model and Simulator
  - How we model HPC demand response participation?
  - How we simulate the proposed model?
  - Cooling energy model
  - How we compare our model with existing policies?
- Conclusions

Demand Response Model

- Power and performance prediction model
  - Based on regression models for power capping
- Resource provisioning
  - Determine processors' optimal power allocation to run the job
  - Determine optimal set of processors with thermal-awareness
- Job scheduling
  - Based on FCFS with possible job eviction (to ensure power bound constraint)

Power and Performance Prediction

- We use third-order polynomial function to determine power usage of job *j* running at processors' power-cap limit p_c:

$$p(j, p_c) = a + b \cdot p_c + c \cdot {p_c}^2 + d \cdot {p_c}^3$$

- We use exponential regression function to determine execution time:

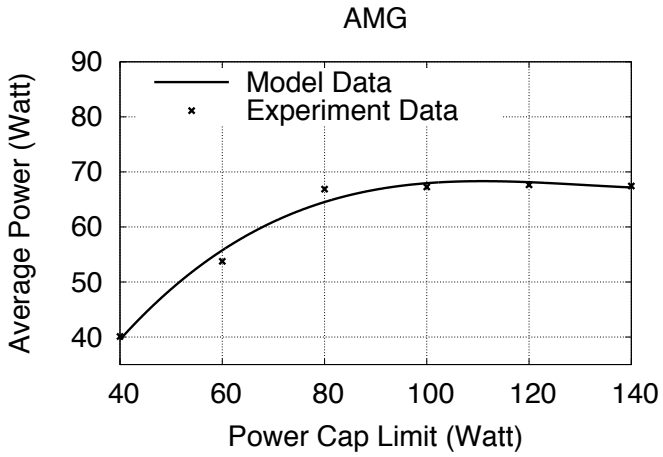
$$t(j, p_c) = \alpha \cdot e^{\beta \cdot p_c} + \gamma$$

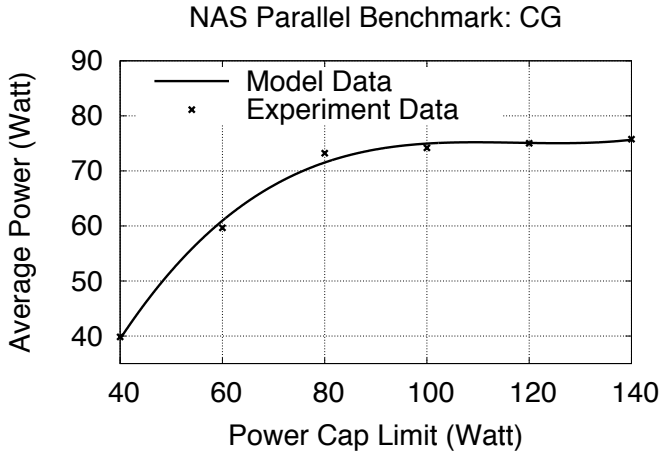
- Total energy consumption for job j can be determined as following:

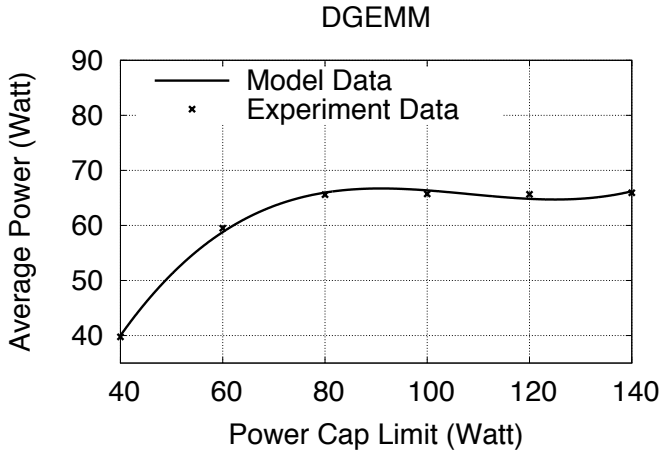
$$e(j, p_c) = n_j \cdot p(j, p_c) \cdot t(j, p_c)$$

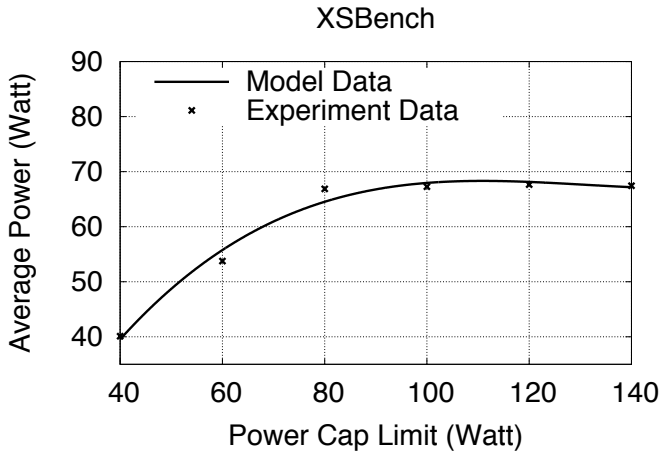
- Theorem: Minimization of the e(j, p_c) is a convex optimization
  - Proof outline: Exponential and polynomial functions are convex

Power and Performance Prediction Results

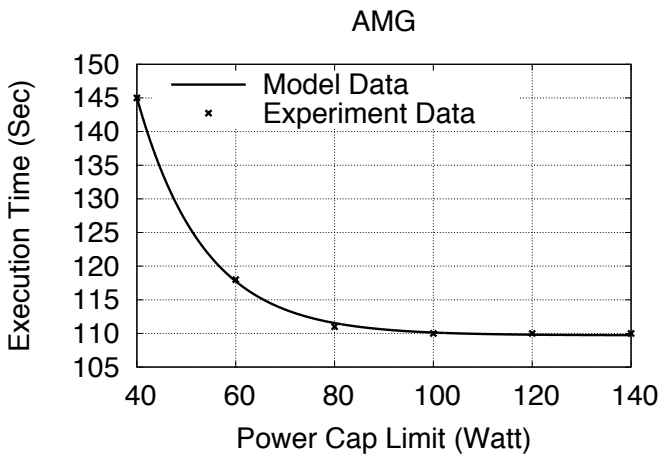


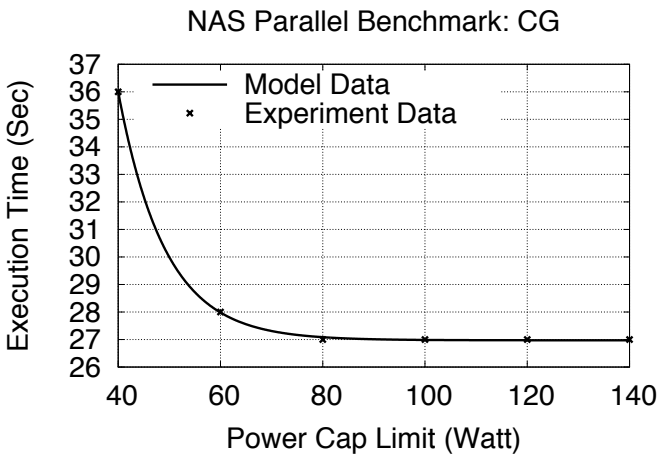


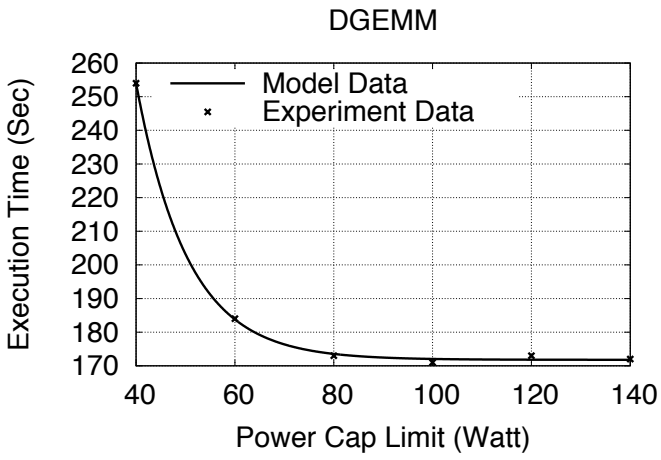


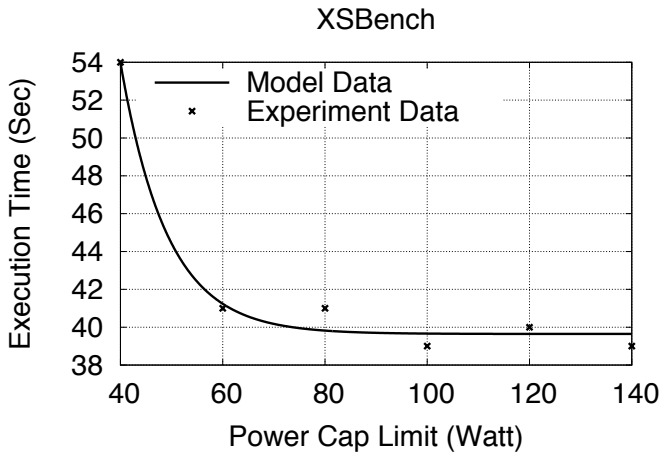


Power and Performance Prediction Results (Contd.)









Job Scheduling and Resource Provisioning

Algorithm 1 HPC Demand Response Job Scheduler

1: find the first eligible job j in wait queue Q
2: **if** job j exists **then**
3: dequeue job j from Q
4: allocate n_j processors to run job j
5: $R \leftarrow R \cup \{j\}$, where R is execution queue
6: **goto** line 1
7: **end if**
8: determine optimal power cap $\forall j \in R$
9: **if** no optimal solution exist **then**
10: evict jobs to reduce power consumption
11: **goto** line 8
12: **end if**
13: reset optimal power cap if changed $\forall j \in R$

Minimize: $\sum_{j \in R} e_R(j, p_c^j)$
subject to constraints (9) and (10)

$$e_R(j, p_c^j) = (1 - \alpha_j) \cdot n_j \cdot p(j, p_c^j) \cdot t(j, p_c^j) \quad (11)$$

$$p_c^{min} \leq p_c \leq p_c^{max} \quad (9)$$

$$p_{run} = \sum_{j \in R} p(j, p_c^j) \leq \hat{p} \quad (10)$$

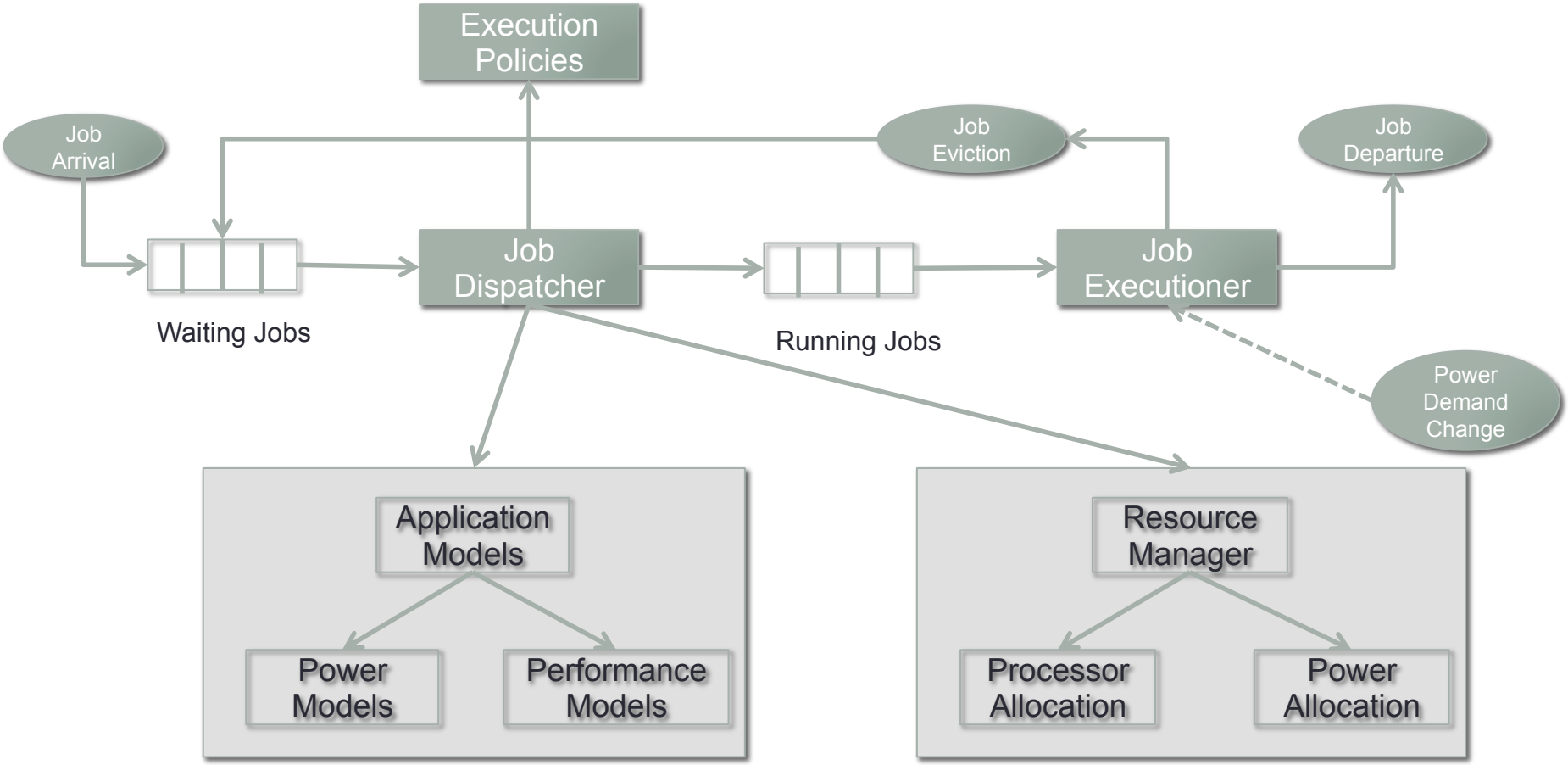
Maximize: $\sum_{j \in R} (x_j \cdot e_X(j))$
subject to $\sum_{j \in R} \left(x_j \cdot p(j, p_c^{min})\right) \leq \hat{p}$

$$e_X(j) \approx \alpha_j \cdot n_j \cdot p(j, p_c^j) \cdot t(j, p_c^j)$$

Job Scheduler Simulator

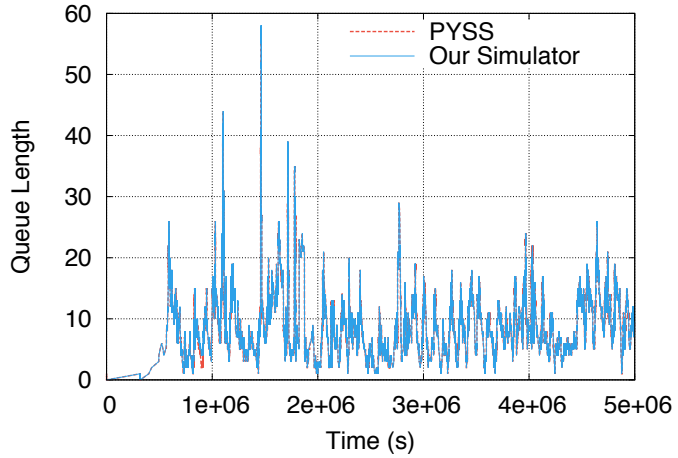
- We use our own scheduler simulator developed earlier
  - Trace-driven capability
  - Flexibility to incorporate new scheduling functions, power-aware methods, as well as demand response models
- Based on Simian
  - An open-source, process-oriented parallel discrete-event simulation engine
  - Some unique features
    - A minimalistic design (only 500 lines of code base)
    - For some models, outperformed simulators using compiled languages such as C or C++
    - Recent significant effort on models based on Simian
      - For example, GPU models (Chapuis et al.), interconnection models (Ahmed et al.)

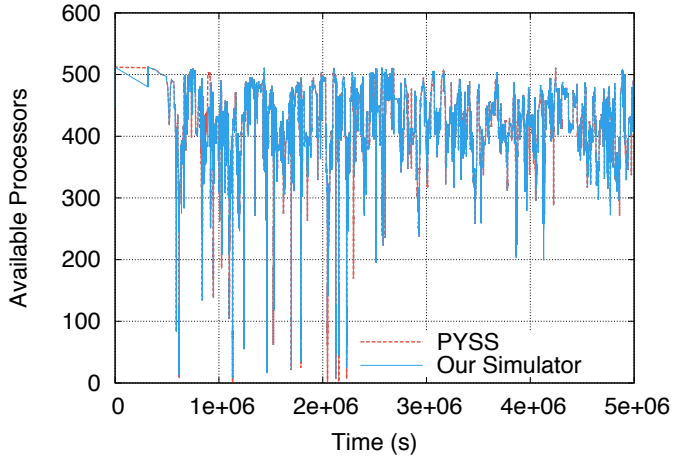
Job Scheduler Simulator (Contd.)



Job Scheduler Simulator (Contd.)

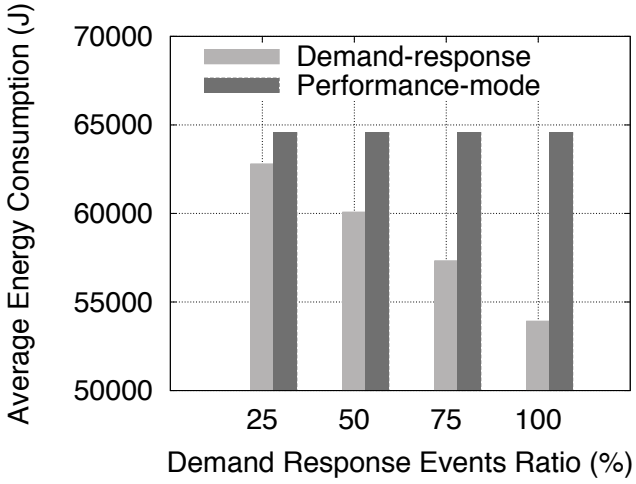
- Validated against PYSS
  - A python-based scheduler simulator for HPC workload
  - Has been used to study various scheduling algorithms in HPC system
- Collected workload trace
  - Parallel Workloads Archive
  - Contains information such as job start time, job run time, number of requested processors, etc.

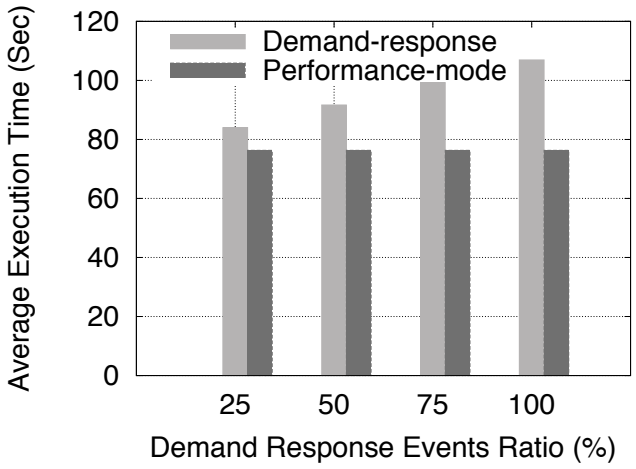




Energy vs. Performance

- Workload trace collected from Parallel Workloads Archive
- Power and performance data collected after running real-life HPC applications on clusters
- Resource allocation policy
  - Performance-mode
    - Always chooses maximum power cap limit to ensure best application runtime





Thermal-aware Job Placement

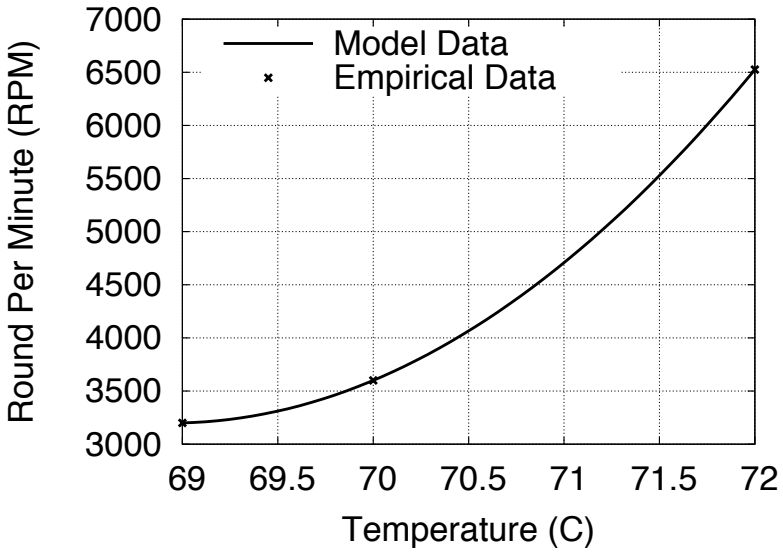
- Determine subset of processors to place the jobs
- Select the cooler processors to execute the jobs
- Assume x_p ε {0, 1} denotes the selection of processor p
  - x_p = 1 means processor p is selected
  - x_p = 0, otherwise
- Jobs are distributed to processors according to following optimization:

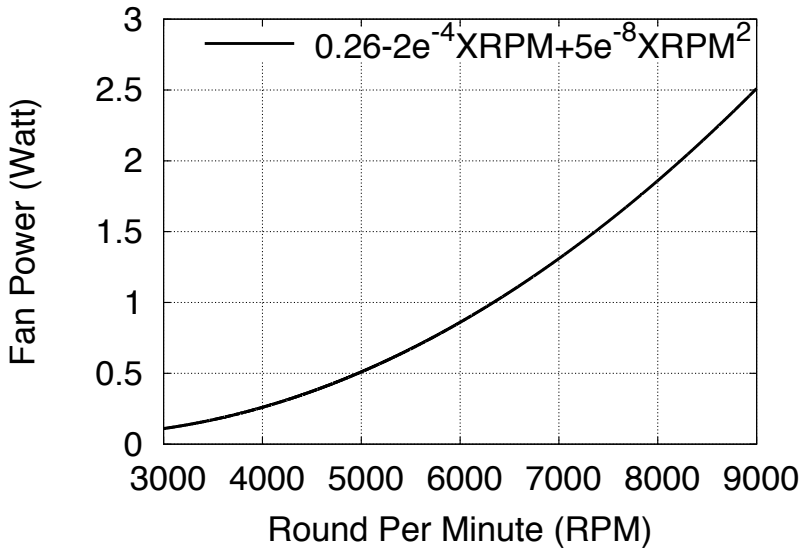
$$\text{Minimize: } \sum_{p \in N}(x_p \cdot T_p)$$
$$\text{subject to } N_r \leq \sum_{p \in N}(x_p) \leq N_t$$

- Where,
  - T_p denotes temperature of processor p
  - N_r denotes number of requested processors
  - N_t denotes number of available processors

Power Prediction Results

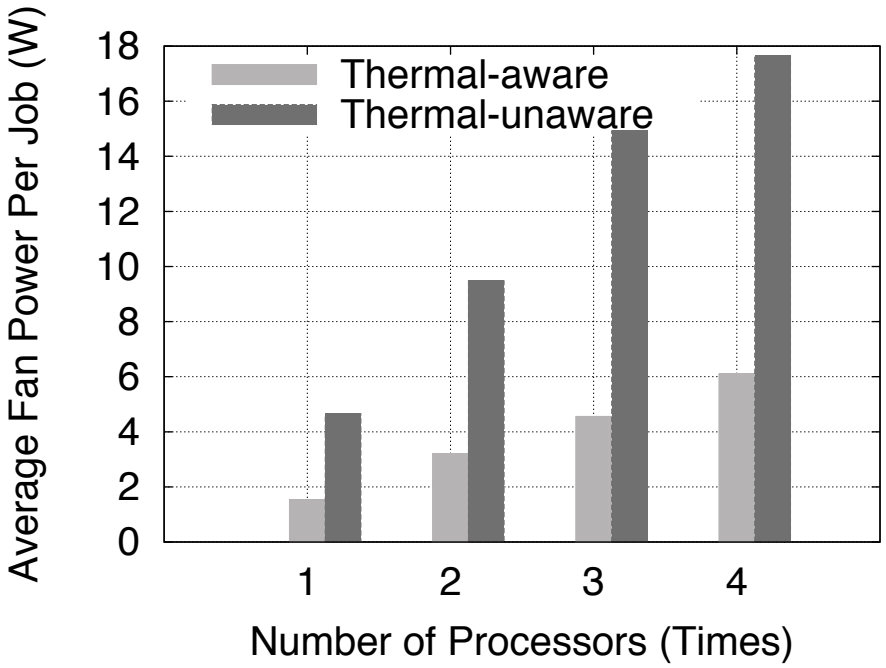
- How to relate temperature to processor power consumption?





Thermal-aware vs. Thermal-unaware

- **Thermal-aware**: Determines processor subset using our algorithm
- **Thermal-unaware**: Places jobs to first available processors



Cooling System Model

- Determine optimal thermostat temperature during demand response periods
- The cooling power consumption is formulated as

$$P_{AC} = \frac{P_C}{CoP(T_{sup})}$$

- Coefficient of Performance (CoP) depends on supply temperature (T_{sup})

$$CoP(T_{sup}) = 0.0068 \cdot T_{sup}^2 + 0.0008 \cdot T_{sup} + 0.458$$

- The power change through thermostat temperature change

$$\Delta P_{AC} = \frac{P_C}{CoP(T_{sup})} - \frac{P_C}{CoP(T'_{sup})}$$

Cooling System Model (Contd.)

- Inlet (t^{in}) temperature for each processor depends on
  - supply air temperature vector and power consumption for processors

$$\vec{t}^{in} = \vec{t}^{sup} + D\vec{p}$$

- Inlet temperature to each processor should be within redline threshold temperature (t^{red})

$$\vec{t}^{in} \preceq \vec{t}^{red} \qquad (5)$$

- The optimization for cooling energy consumption can be formulated as following

$$\text{Maximize:} \Delta P_{AC}$$
$$\text{subject to constraint (5)}$$

Conclusions

- We studied
  - Possibility of HPC system's demand response participation
- We proposed a demand-response model which ensures
  - Demand response participation through power capping and processor allocation
  - Energy reduction in processor, memory and cooling system
- We experimented
  - Real-life scientific applications on experiment cluster
  - Demonstrated effectiveness of our proposed approaches
- Difficulty
  - Chameleon cluster experiments could not be completed due to BIOS issue
- Future works
  - Experiments on cooling energy model
  - A prediction model for prediction of unknown HPC applications' characteristics (e.g., power usage)

Thank you all! Questions?

Many thanks to –
Kazutomo Yoshii*, Jason Liu**, Xingfu Wu*,
Misbah Mubarak*, Rob Ross*

* MCS, Argonne National Laboratory
** SCIS, Florida International University