MITIGATING VARIABILITY IN HPC SYSTEMS AND APPLICATIONS FOR PERFORMANCE AND POWER EFFICIENCY

BILGE ACUN

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

DOCTORAL SHOWCASE– NOV, 2017

Dissertation Goal

To increase the performance and power efficiency of High Performance Computing (HPC) systems through mitigating various sources of variability without sacrificing from performance

- Analyze variability in large scale HPC systems
  - Frequency, power, temperature
- Address each of the sources of the variability
  - Via software and hardware techniques

Outline

1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Frequency, Temperature, Power
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

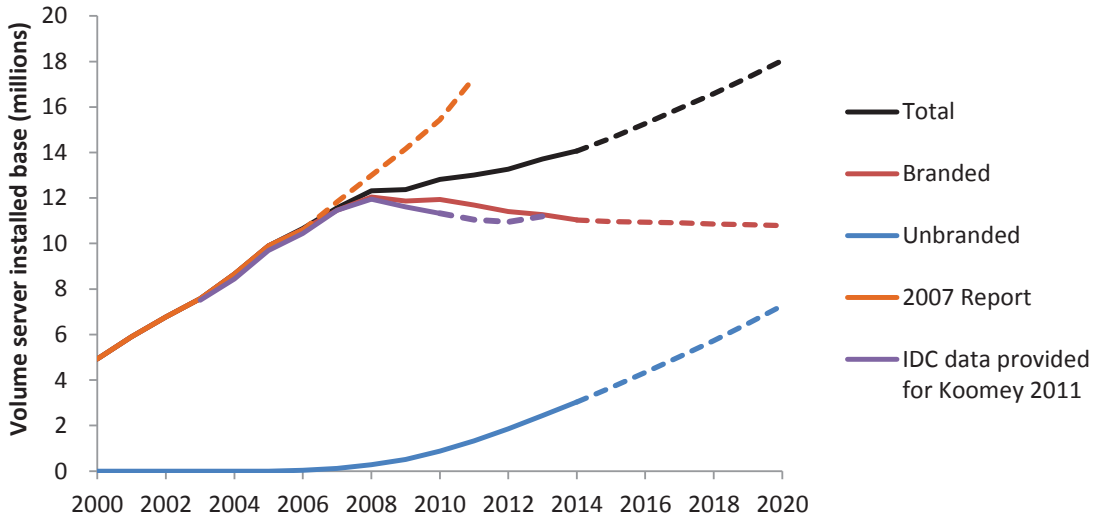
Outline

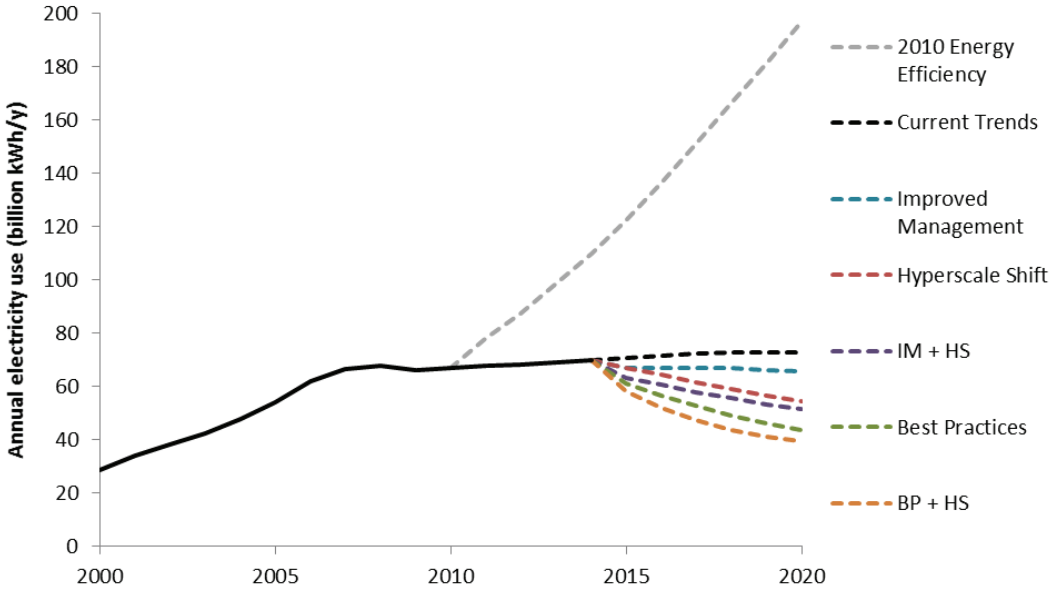
1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

2016 U.S. Data Center Energy Report

- Energy consumption has been flat-lined due to (*):
  - Improved operations
  - Hardware advancements

- Data center electricity is spent by (*):
  - Servers: ~40%
  - Infrastructure: ~40%
  - Network and storage: ~20%



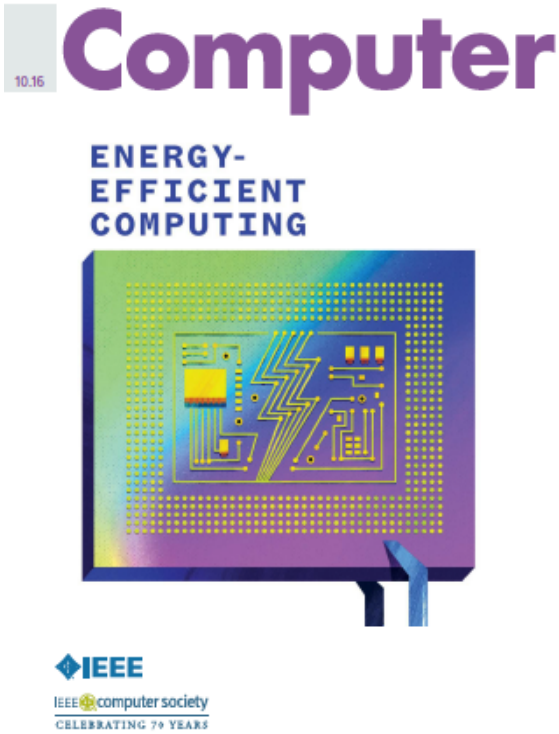


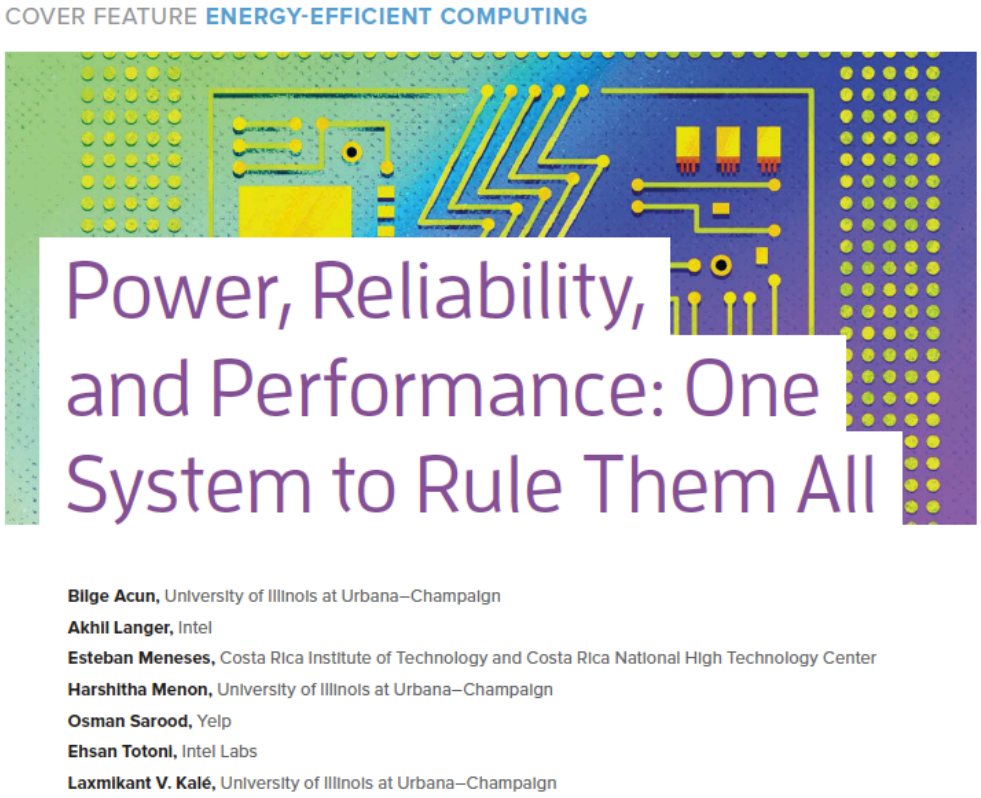
* Figures and data are taken from A. Shehabi et al. "United states data center energy usage report," Lawrence Berkeley National Laboratory. LBNL-1005775, vol. 4, 2016.

Outline

1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

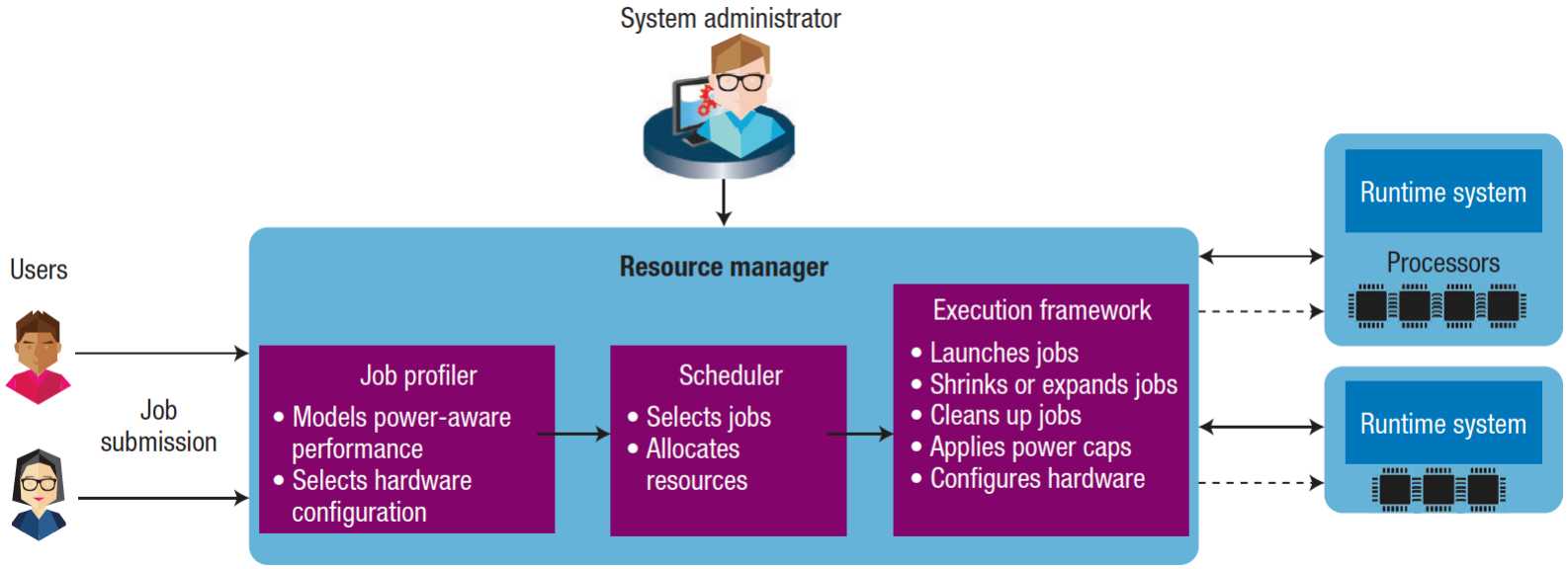
Charm++ as an Energy Efficient Runtime





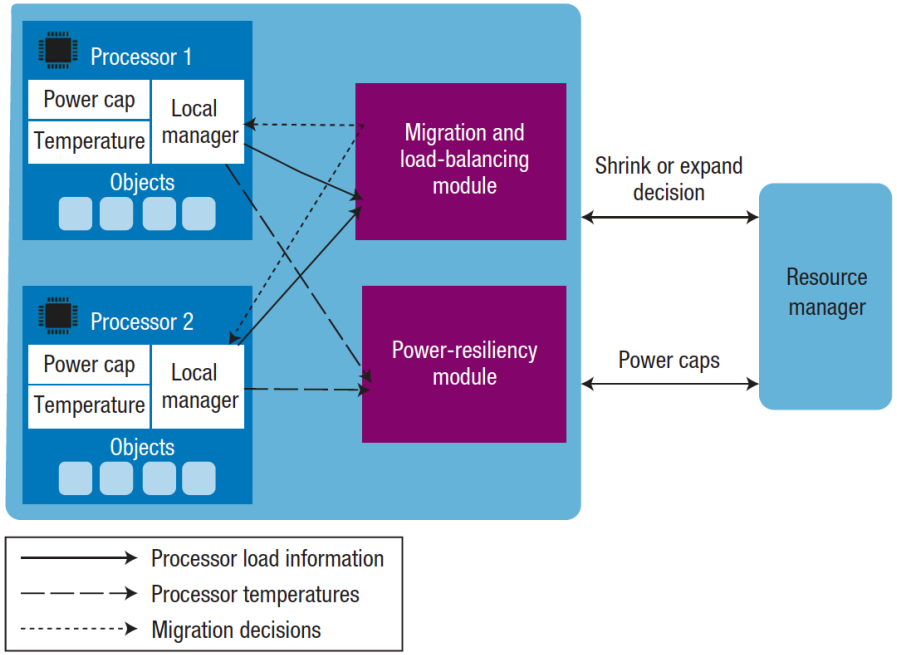
B. ACUN, A. LANGER, H. MENON, O. SAROOD, E. TOTONI, AND L. V. KALE. "POWER, RELIABILITY, PERFORMANCE: ONE SYSTEM TO RULE THEM ALL". IEEE COMPUTER, 2016.

Interaction Between the Runtime System and the Resource Manager



- ✓ Allows dynamic interaction between the system resource manager and the runtime system
- ✓ Meets system-level constraints such as power caps and hardware configurations
- ✓ Achieves the objectives of both datacenter users and system administrators

Components of Charm++ with Its Interactions



Charm++ has four main components:

- **Local manager:** tracks local information such as object loads, CPU temperatures
- **Load-balancing module:** makes load-balancing decisions and redistributes load
- **Power-resiliency module:** ensures that the CPU temperatures remain below the temperature threshold, change the power cap
- **Client-server interface:** Enables interactions with other programs

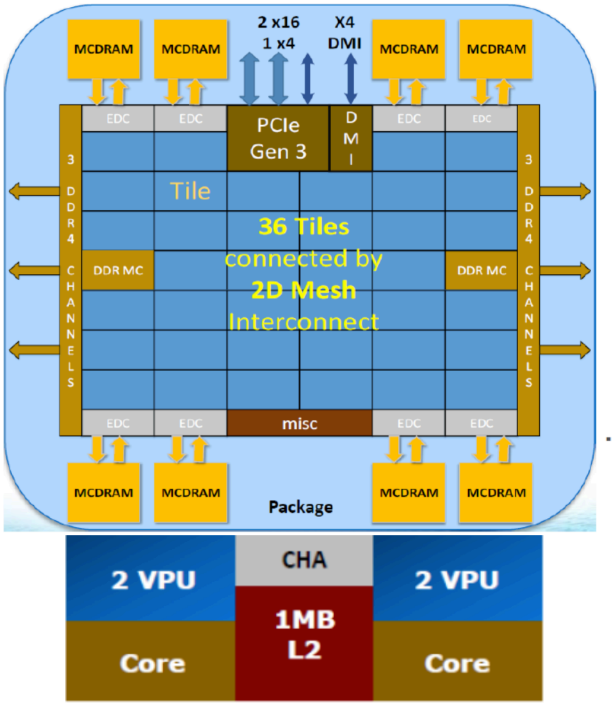
B. ACUN, A. LANGER, H. MENON, O. SAROOD, E. TOTONI, AND L. V. KALE. "POWER, RELIABILITY, PERFORMANCE: ONE SYSTEM TO RULE THEM ALL". IEEE COMPUTER, 2016.

Outline

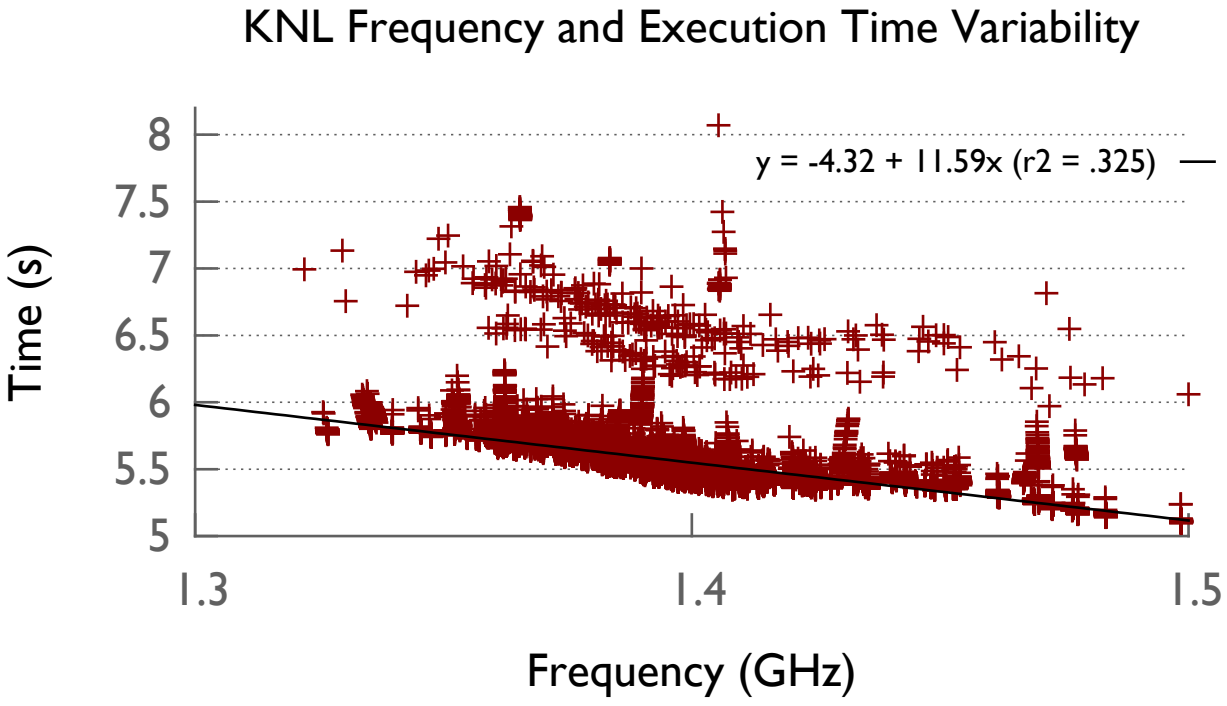
1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

Analysis of Performance Variability

- Intel's Xeon Phi - Knights Landing (KNL) processors on Cori supercomputer at NERSC
- High variability due to 4 reasons is reported (frequently 15%, up to 100%):
  - OS noise
  - Cache contention on tile (L2)
  - Memory variability due to cache mode page conflicts
  - Network variability



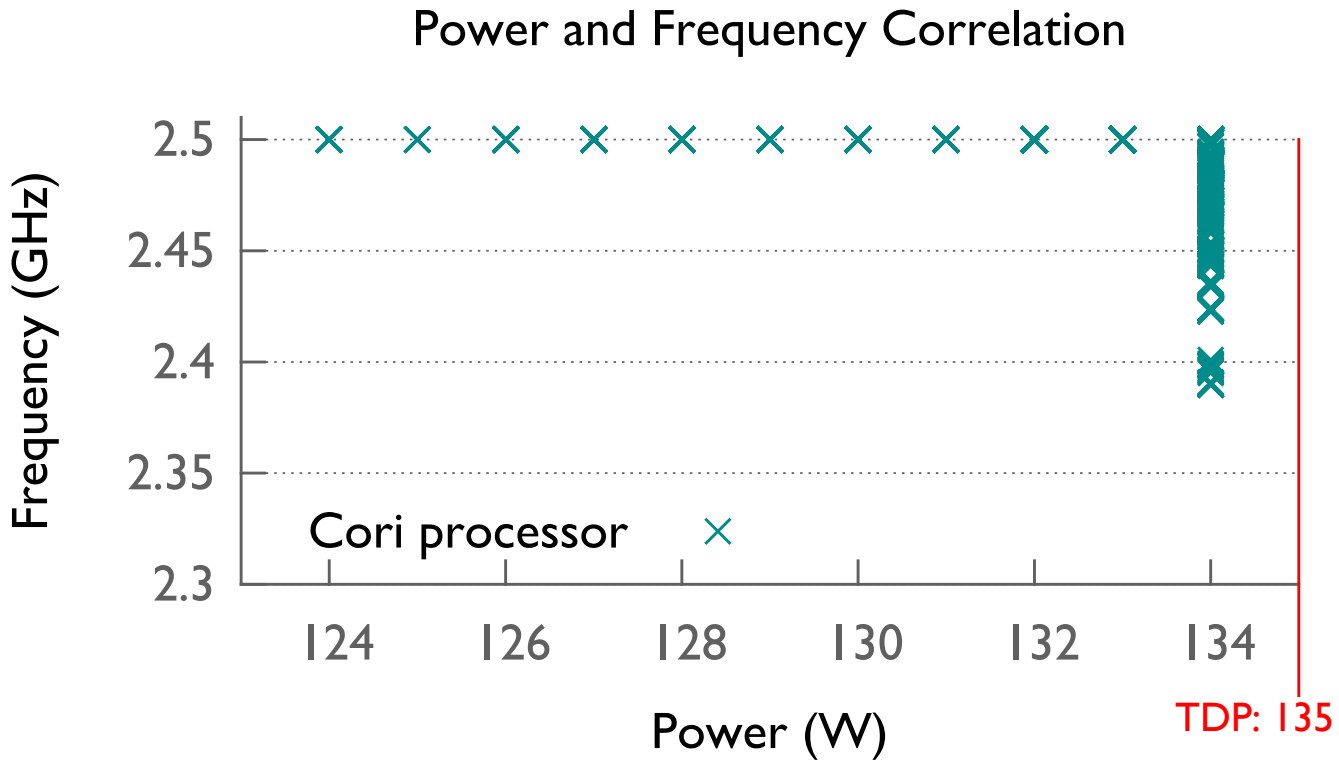
* Image source: https://insidehpc.com/2016/01/mcdram/



- ~ 50% performance variation on 256 KNL processors

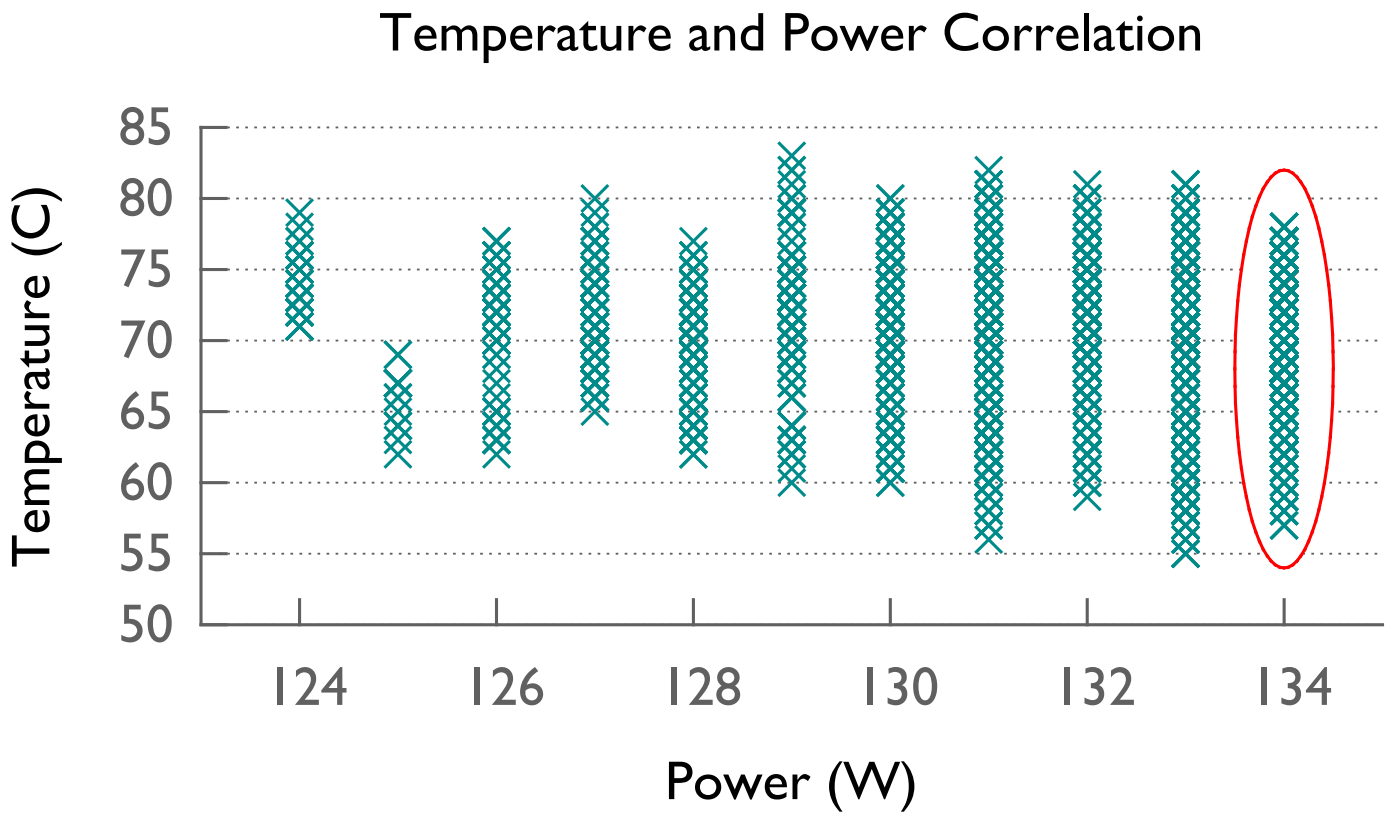
Power and Frequency Correlation

- The processors whose frequencies are throttled all hit the Thermal Design Power (TDP) of the chip (135 Watts)
- 256 Cori Intel Haswell processors is shown



Frequency Throttling is not Only Temperature Related

- The processors that hit TDP, have a wide range of temperature from 56 C to 78C (22 C difference)
- Processors that do not hit TDP have similar temperature ranges



Summary: Variation Analysis

- Large-scale systems exhibit power and temperature variations that are related to design and manufacture.
- The inherent differences manifest themselves as frequency, performance variations.
- Mitigating these variations enables to increase the performance and power efficiency without sacrificing from performance which is an important concern for HPC users.

Outline

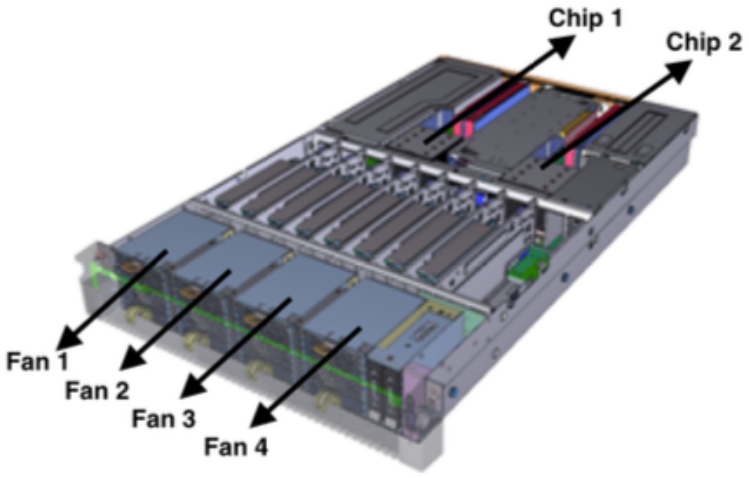
1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

Outline

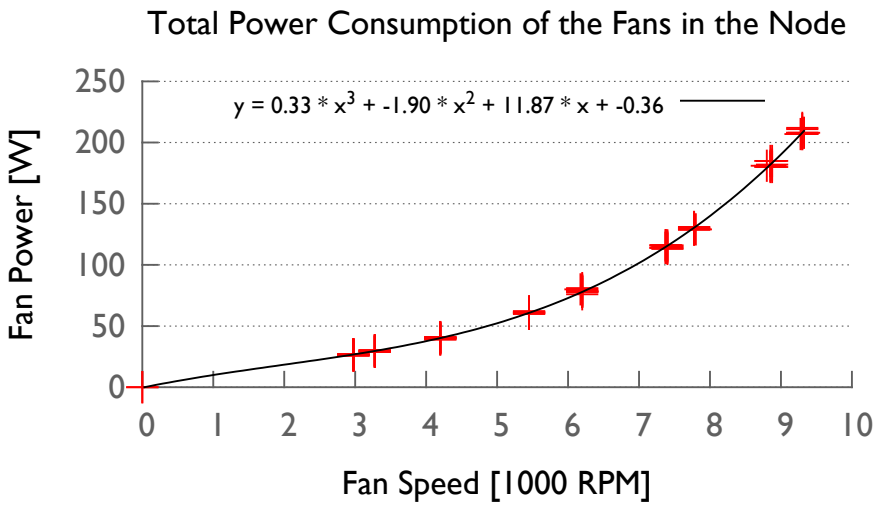
1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

Motivation

- Understand the temperature variation and cooling inefficiencies in large scale systems
- Find solutions to mitigate the variation in order to reduce cooling power



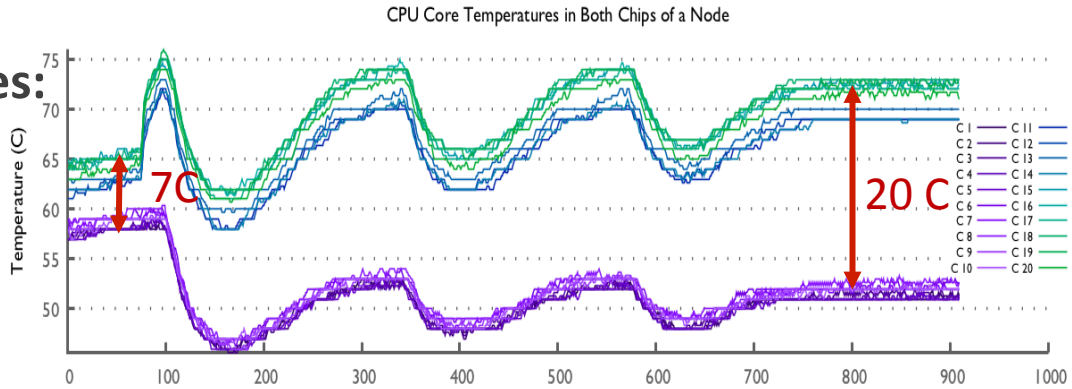
POWER8 Server Node Architecture

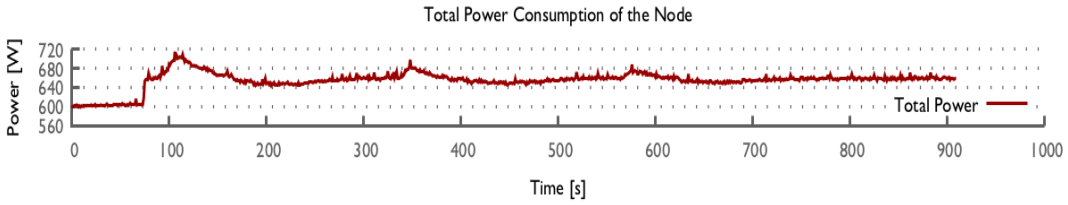


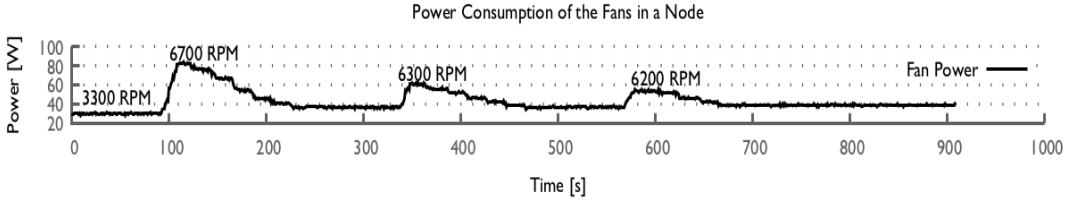
- Four fans that operates synchronously to cool the two chips, GPUs, and memory
- If any of the components hits their temperature limit (73C , 79C , and 74C respectively), fans trigger in a reactive way

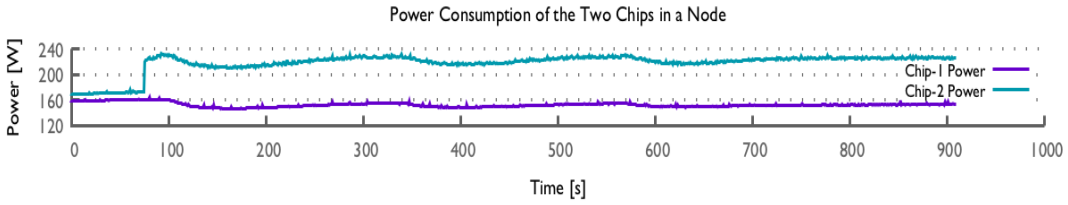
Motivation

- **Temperature variations among cores:**
  - 7 C in idle temperatures
  - 20 C idle/active mixed
  - 9 C in all active temperatures
- **Synchronous fan control:**
  - 4 independent fans in the node
  - Fans all act together and cause even further temperature variation
- **Reactive cooling behavior:**
  - 54 W jump in fan power
  - 10 minutes stabilization time with a regular workload



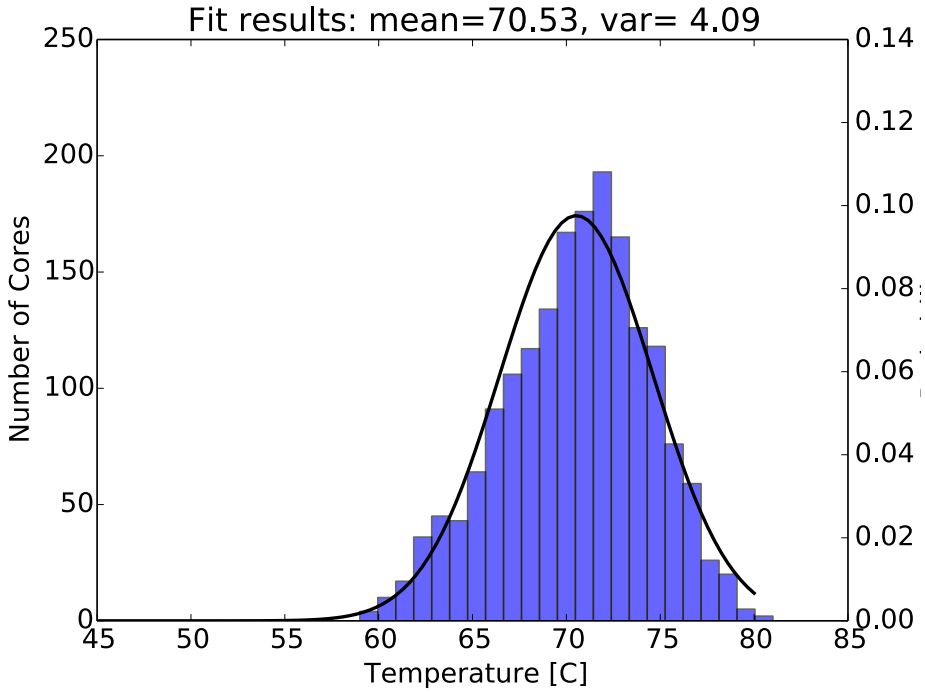




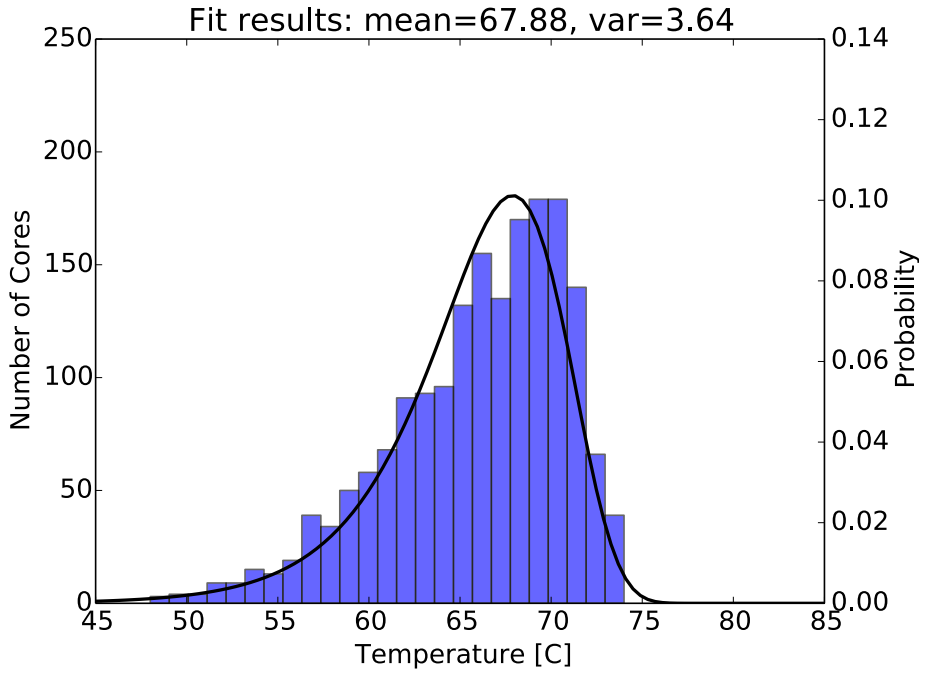


Temperature Variation in Large Scale

- Steady-state temperature distribution of **1,800 cores** in two different platforms:
  - 25 C difference among cores when running the same workload



Cori at NERSC – Intel Haswell

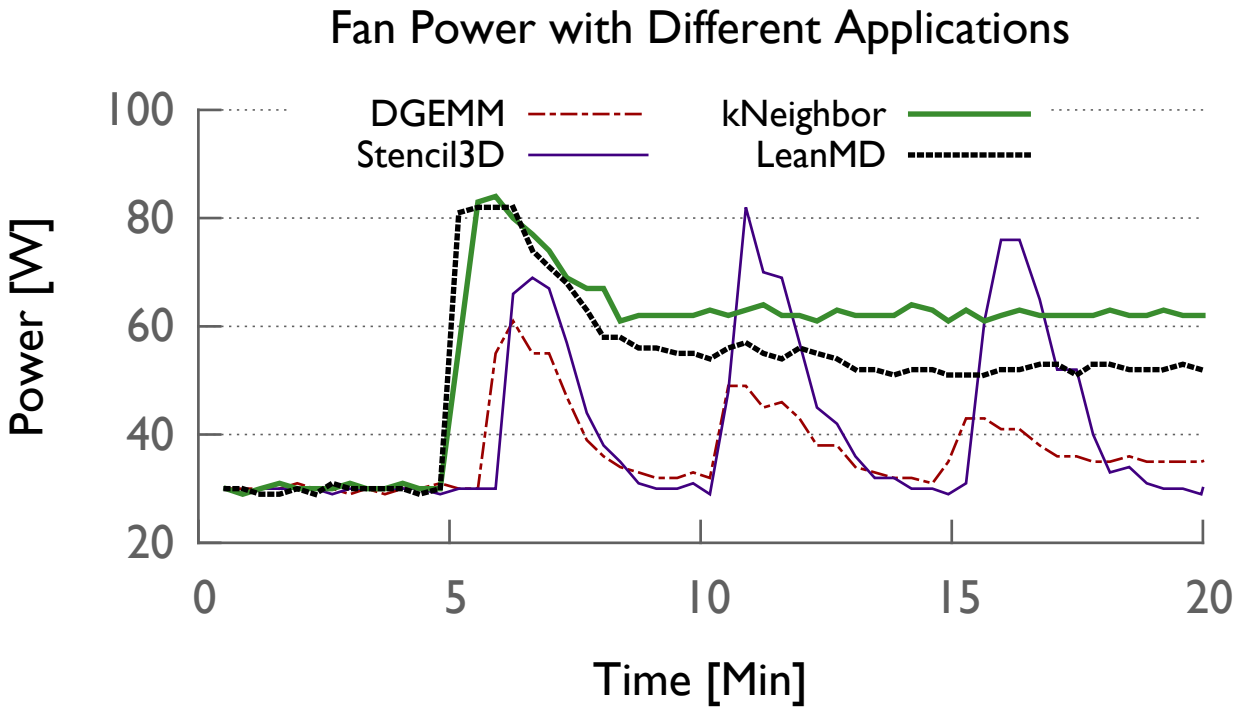


Minsky at IBM - POWER8

Oscillatory Cooling Behavior

Fan Behavior of Different Applications

- Some applications makes only single power peak from the application start
- Some applications keeps oscillating even after tens of minutes of execution

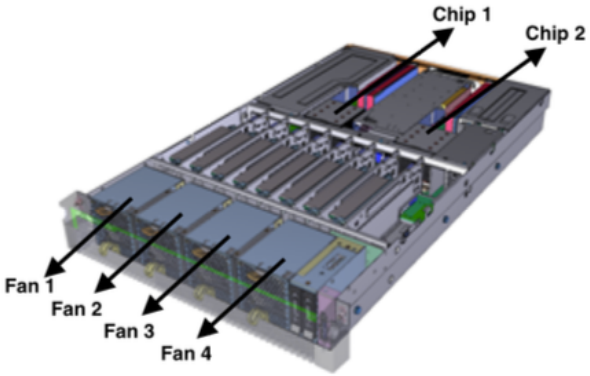


Why Temperature Modeling is Difficult?

- There are lots of parameters affecting the core temperatures:
  - Complex workloads
  - Ambient temperature
  - Core frequencies
  - Fan speed level
  - Physical layout
  - Hardware variations

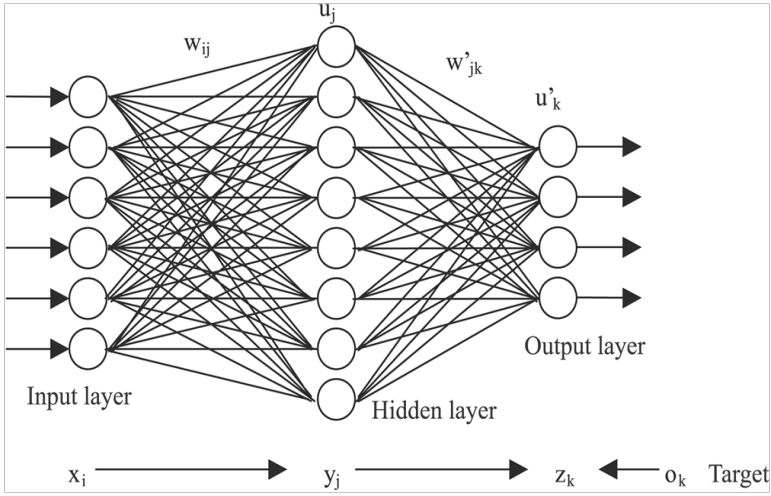
- Combination of these parameters creates an exponential modeling space
  - 10 different cores
  - 0-100 CPU utilization levels
  - 44 different frequency levels
  - 3000 RPM-10000 RPM fan speed levels
  - 4 fans

❖ (10^10) * 44 * (10^4) = ~ 2^52



Neural Networks for Temperature Modeling

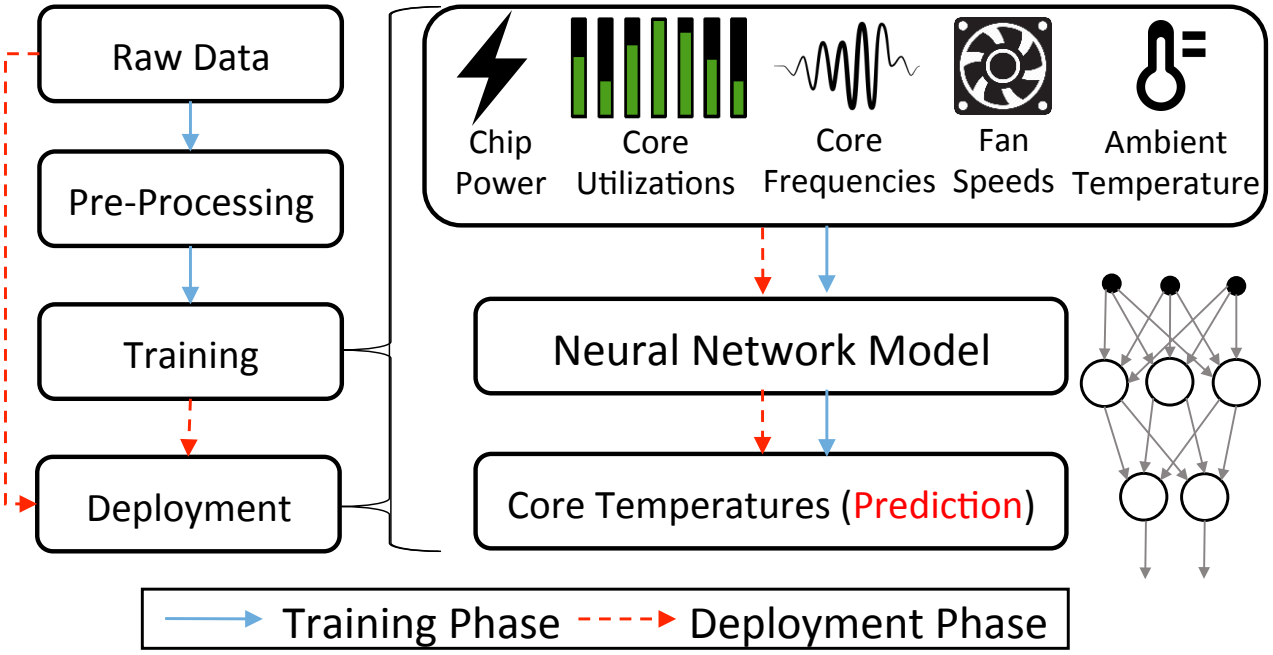
- Neural networks are good because:
  - They can capture linear and non-linear behavior between input and output parameters
  - They work well in noisy data
  - They do not need for formulation of an objective function
- Neural networks has been used in HPC for:
  - Energy and power modeling [1]
  - Performance modeling [2]
  - Temperature modeling:
    - For GPU temperature modeling [3]
    - For coarse-grained data center level modeling [4]



1. A. Tiwari, M. A. Laurenzano, L. Carrington, and A. Snavely. Modeling power and energy usage of HPC kernels. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW),* IEEE, 2012.
2. B. C. Lee, D. M. Brooks, B. R. de Supinski, M. Schulz, K. Singh, and S. A. McKee. Methods of inference and learning for performance modeling of parallel applications. In *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '07, 2007.
3. A. Sridhar, A. Vincenzi, M. Ruggiero, and D. Atienza. Neural network-based thermal simulation of integrated circuits on GPUs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.
4. L. Wang, G. von Laszewski, F. Huang, J. Dayal, T. Frulani, and G. Fox. Task scheduling with ann-based temperature prediction in a data center: a simulation-based study. *Engineering with Computers*, 2011.

Neural Networks for Temperature Prediction

- Proof-of-concept prototype model, other models can also be used in my solutions.

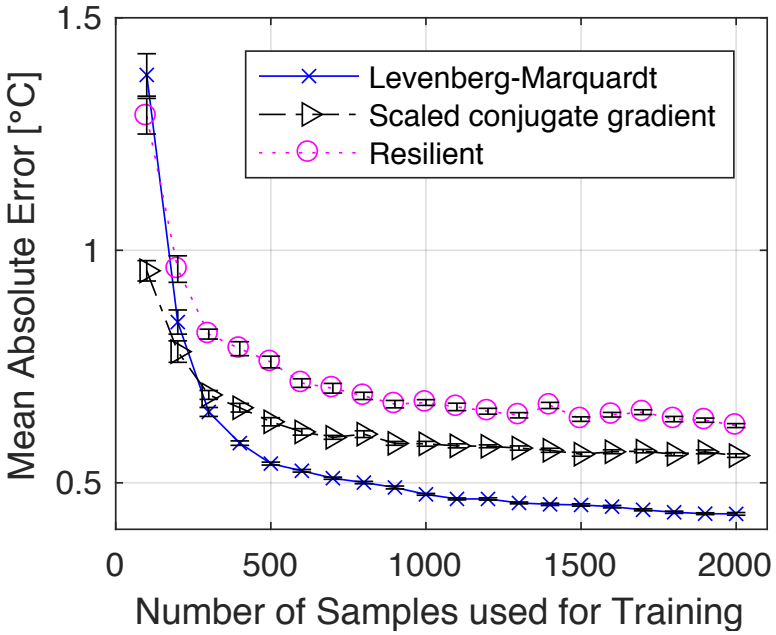


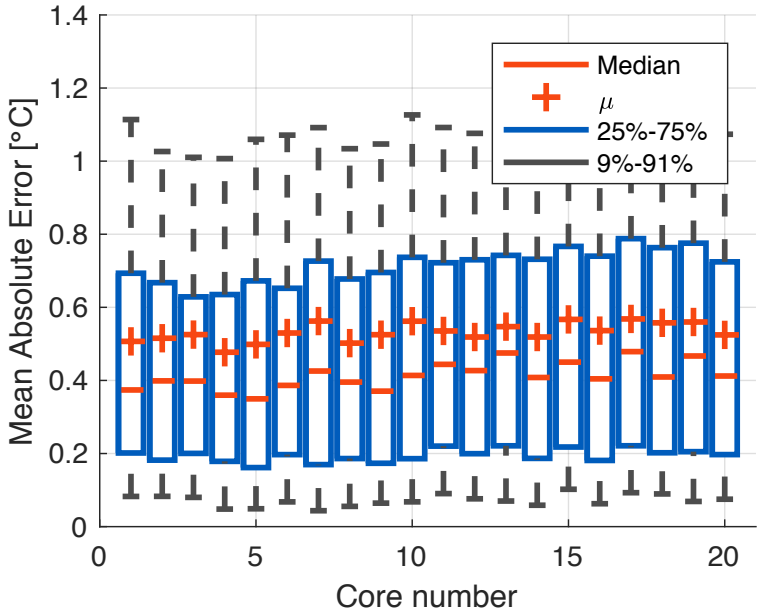
Experimental Setup:

- Firestone cluster at IBM with Power 8 processors
- 1 node = 2 sockets, 20 physical cores, 160 SMT cores
- OCC, and BMC for temperature, power readings

Neural Network Configuration and Validation

- We test different back-propagation algorithms with different time and memory requirements.





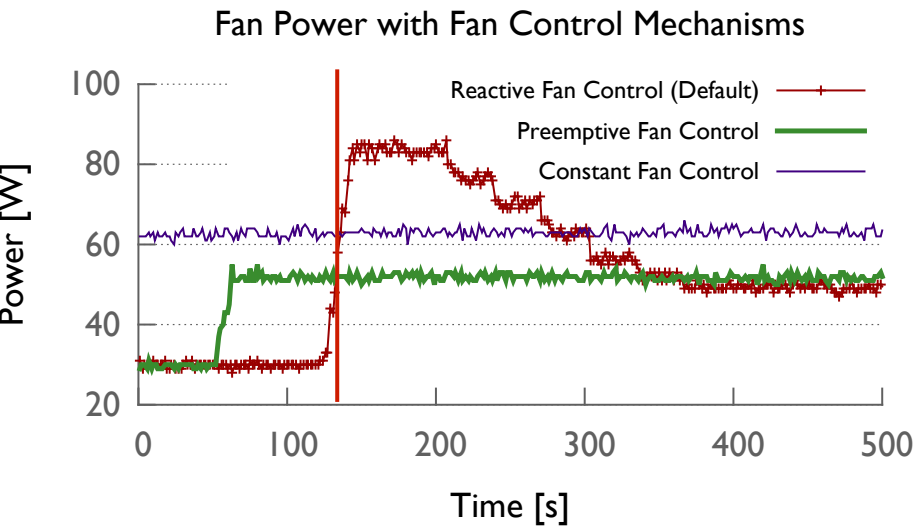
- Other configurations include number of layers, and number of neurons.

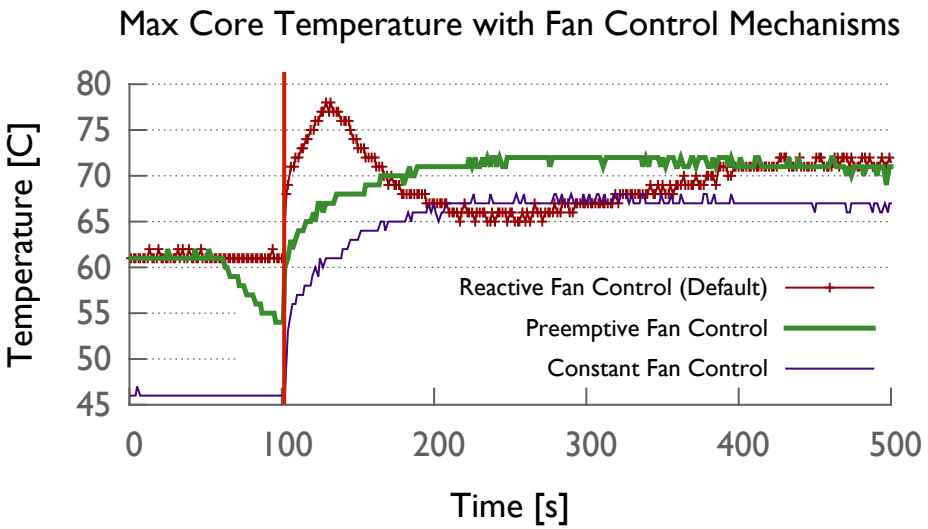
Model Guided Proactive Cooling Decisions

- Fan control
  - This can reduce oscillations and chip-to-chip temperature variations.
  - What should be the fan speed level to be able keep the chips at a certain temperature limit?
- Load balancing
  - This can remove core-to-core, as well as chip-to-chip temperature variations.
  - What would the core temperatures become if a certain amount of data is moved from one core to another?
- DVFS
  - Chip-level DVFS can reduce chip-to-chip, core level DVFS core-to-core temperature variations.
  - What frequency level we need to set for the cores to stay under a temperature limit for a workload?

Proactive Fan Control Mechanism

- The key idea behind *precooling* is to cool the processor proactively, for example, before the application starts.





45.6% reduction in fan power

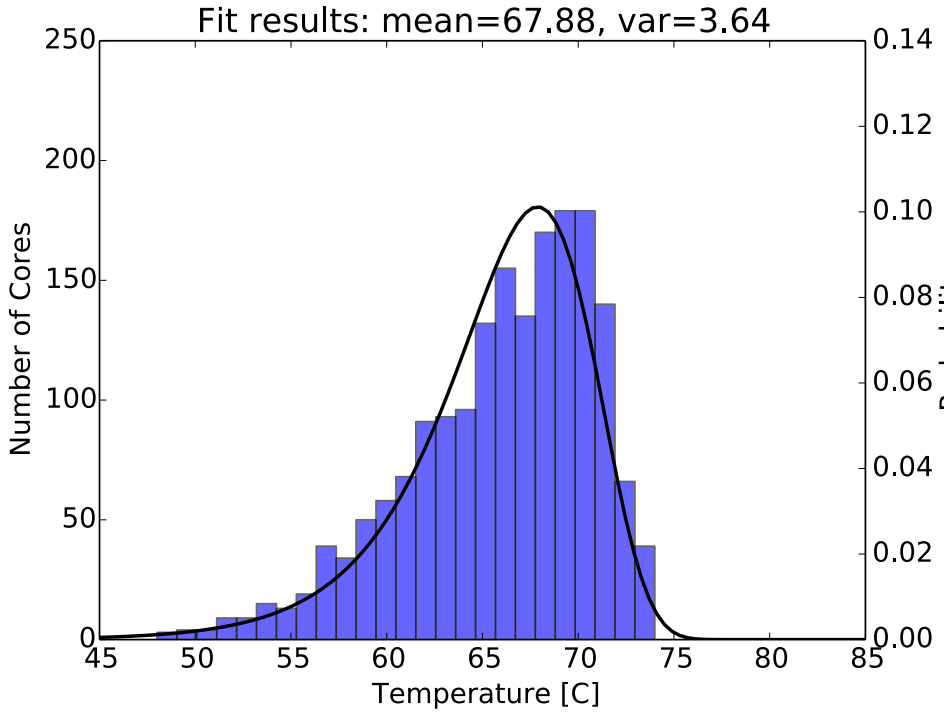
9.4% reduction in fan energy

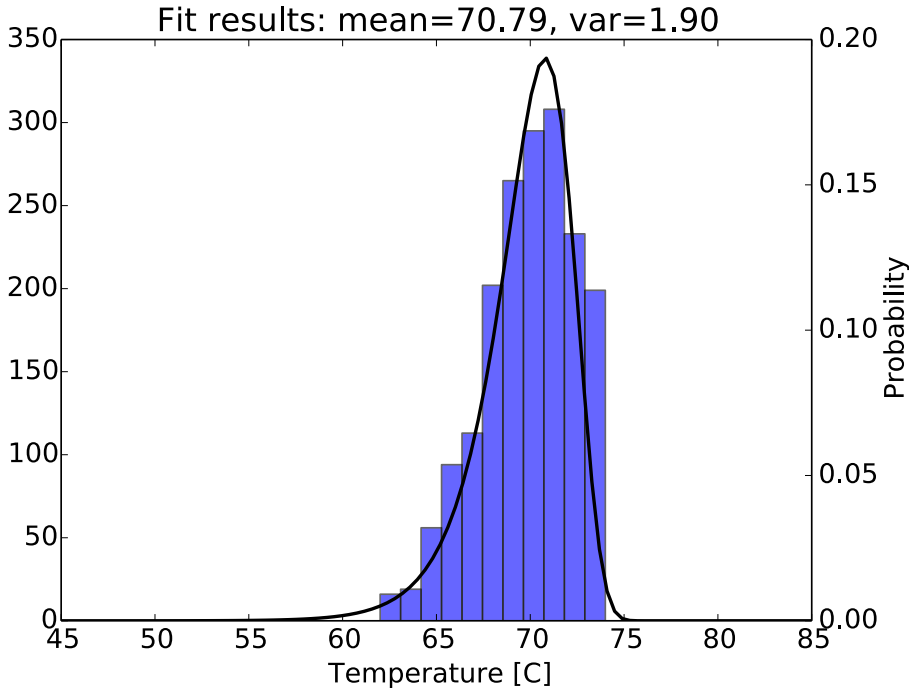
- Preemptive fan-control removes temperature peaks, and is able to keep the temperature as the same level as reactive fan control.
- It can be done via job scheduler, and/or runtime without taking over the total control.

Decoupling the Fans

7.7% reduction in fan power

13% reduction in fan energy





BEFORE

AFTER

Total Reduction in Fan Power

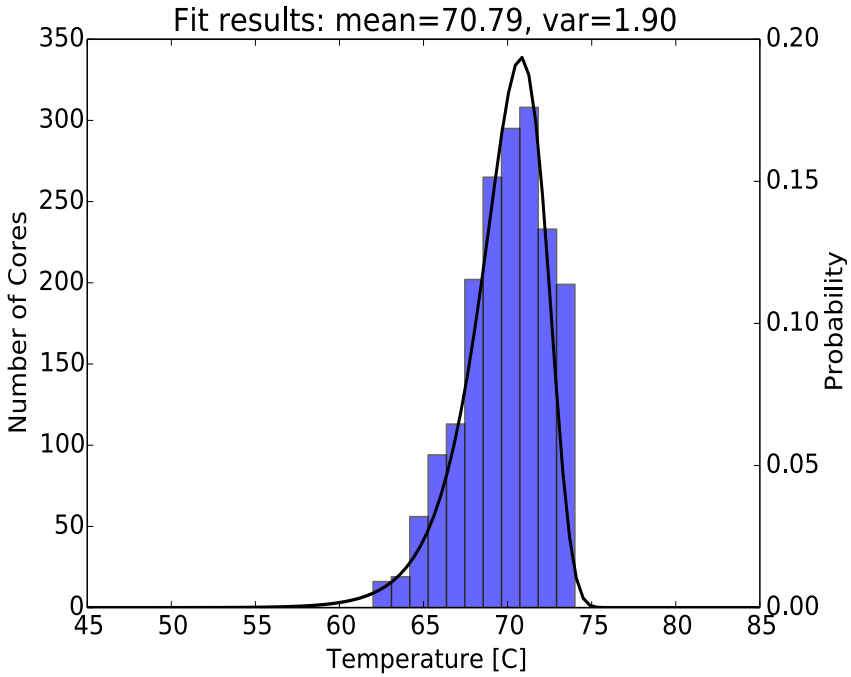
53% reduction in fan power on average

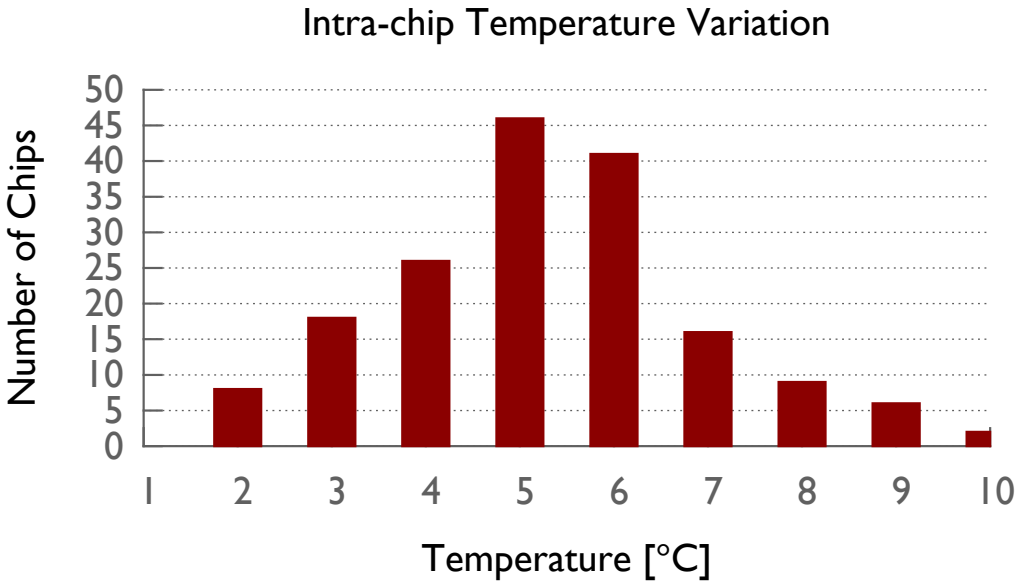
22% reduction in energy on average

Optimization \ Benchmarks	DGEMM	Stencil3D	kNeighbor	LeanMD	Average
Reactive Fan Control	5868 W	13433 W	6769 W	6770 W	8210 W
Preemptive Fan Control	3893 W	8526 W	4381 W	4224 W	5256 W
Preemptive and Decoupled Fan Control	3179 W	7972 W	3765 W	3569 W	4621 W
Total Power Reduction (%)	45.8	59.3	55.6	52.7	53.3

Remaining Temperature Variation

- There is up to 10 C intra-chip variation that cannot be mitigated by decoupled fans

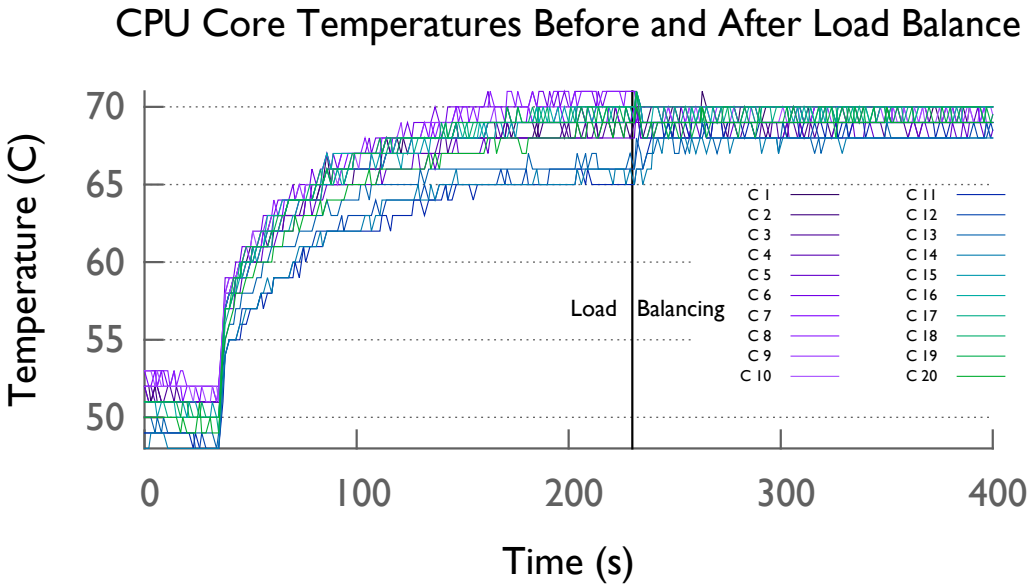




- How to mitigate intra-chip temperature variation?
  - DVFS: core-level is not supported in many architectures ☹
  - Load Balancing ☺

Temperature-Aware Load Balancing With Charm++

- Load balancing has potential to remove both chip and core level variations.
- It can help reduce the temperature variations, but how do we decide how much load to move?
- Charm++ has an runtime database which stores:
  - Number of tasks per process
  - Load of each object
- Load balancing is triggered periodically with customizable periods
- We implement our temperature-aware model guided load balancing algorithm.



Summary: Mitigating Temperature Variation

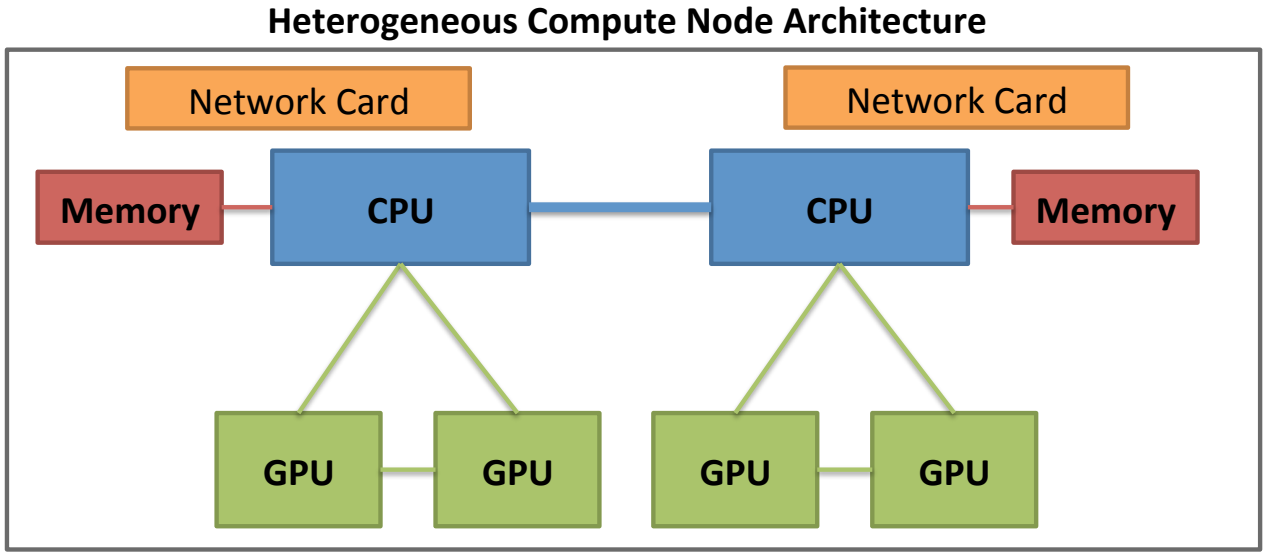
- Analyzed inefficiencies in cooling systems
- Proposed solutions based on a neural network based temperature prediction model:
  - Precooling
  - Decoupled fan control
  - Load balancing
- Our results shows:
  - We can accurately predict core temperatures
  - Peak fan power can be reduced by 53%, energy by 22%
  - As a result, air cooling systems can be made more efficient

Outline

1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

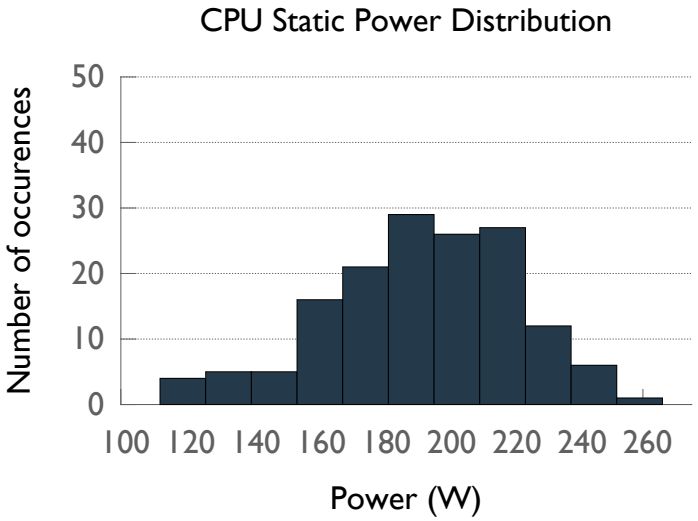
Mitigating Across Component Power Variation

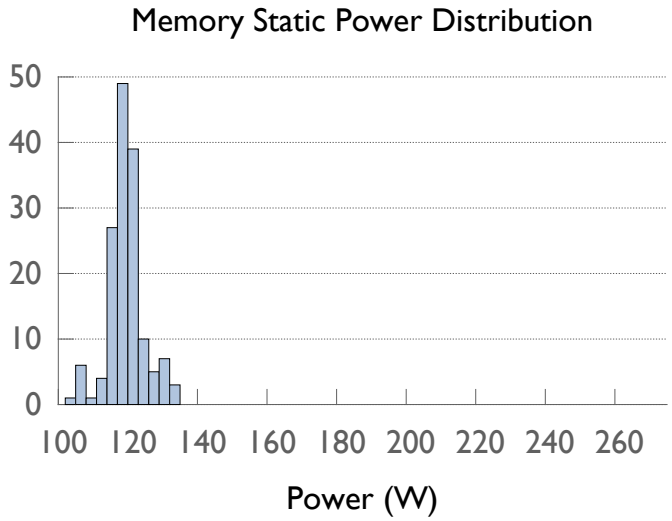
- An exascale architecture having fat and heterogeneous compute nodes
- Each of the node components have different power variations

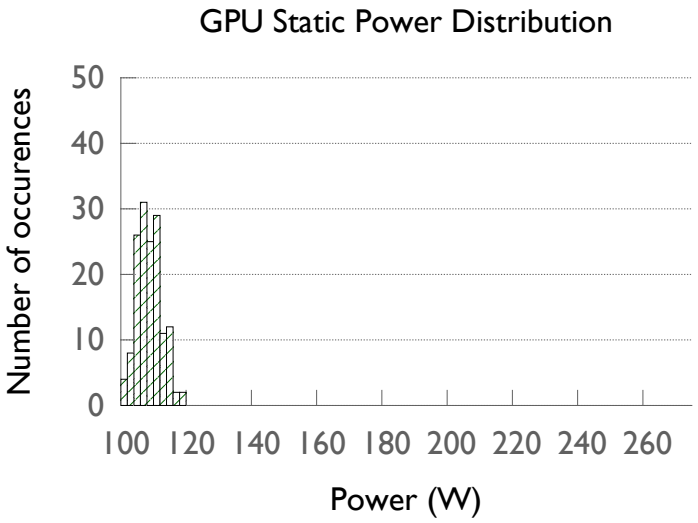


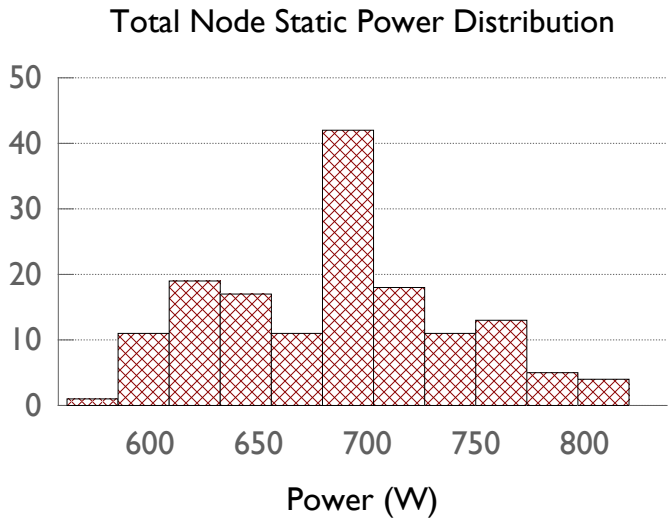
- Sierra and SummitDev node architecture.
- SummitDev has IBM POWER8 CPUs, NVIDIA Tesla P100 GPUs, DDR4 memory and Mellanox EDR Infiniband network adapters.

Idle Power Distribution of Node Components

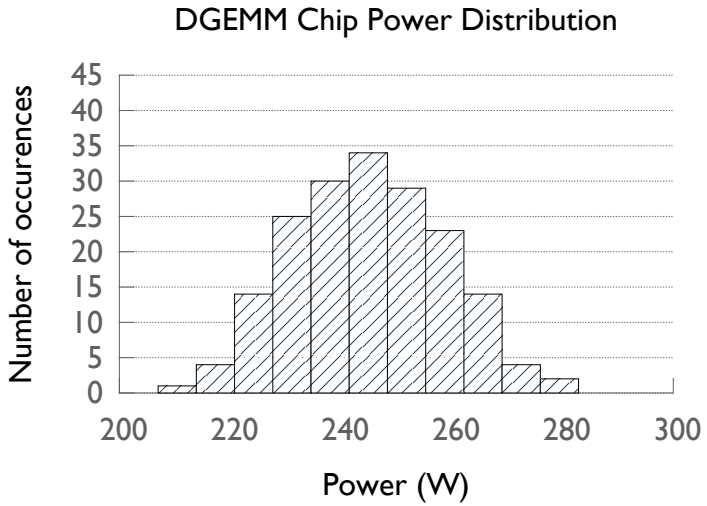


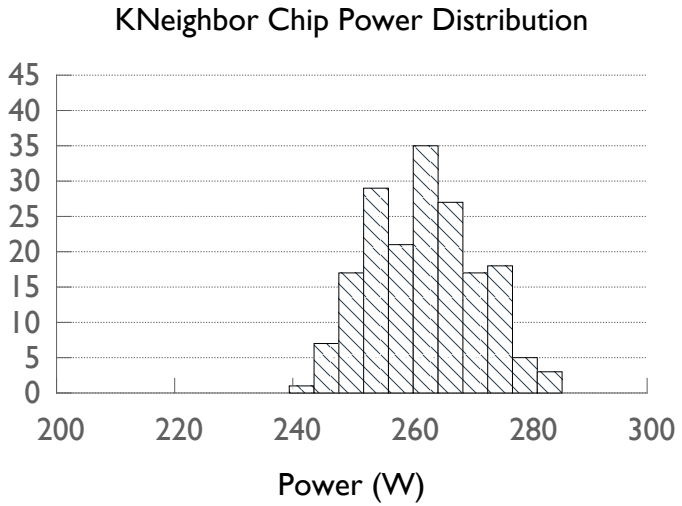


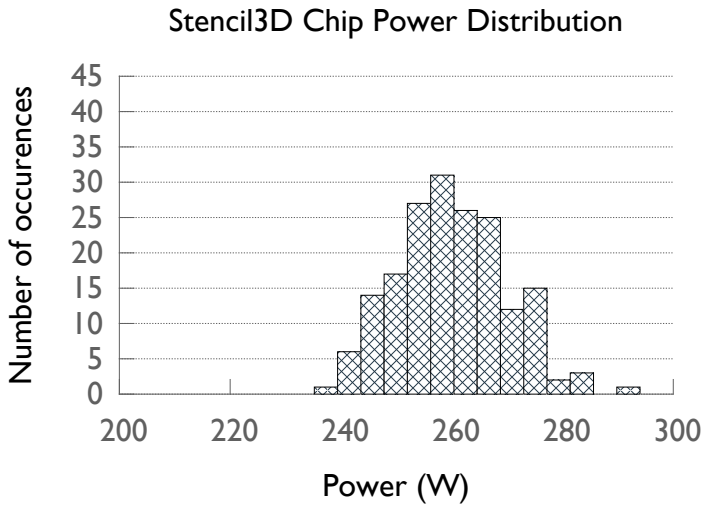


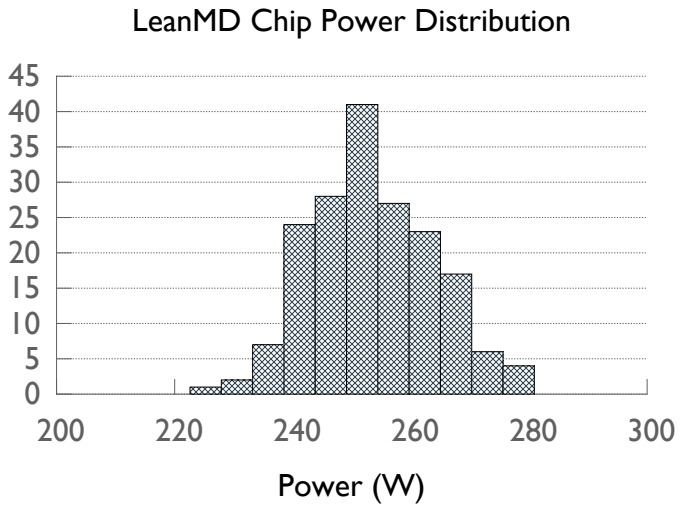


CPU Power Distribution of Different Benchmarks





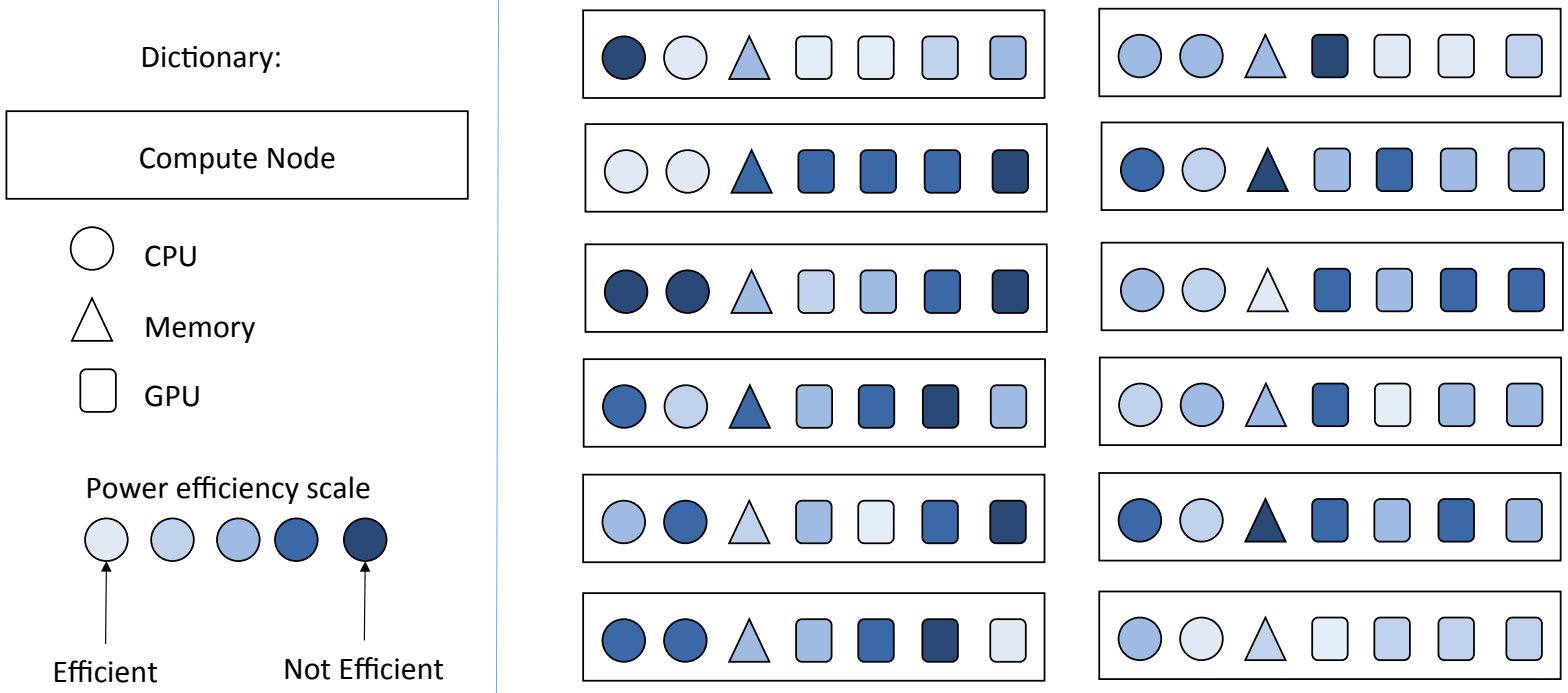




Random Node Assembly

- Efficient and non-efficient components may randomly show up in a node.

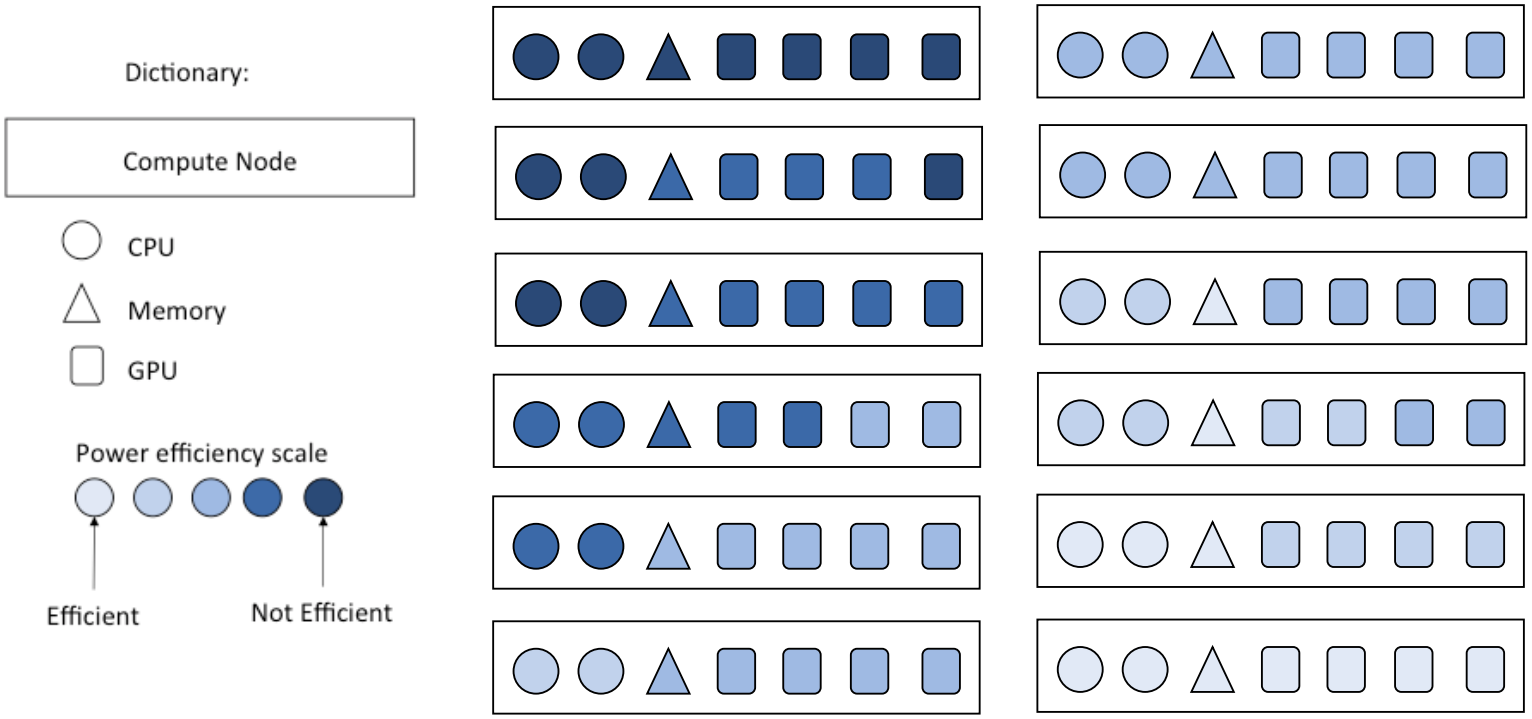
Illustration of Data Center Components' Efficiency in Random Assembly



Categorized Node Assembly

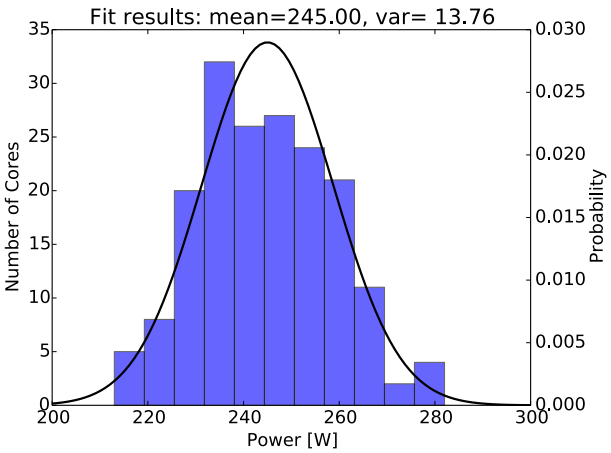
- Components having the same efficiency level are gathered in the same node
- Data center consumes less power if not at full load
- Customers can select to buy only efficient nodes

Illustration of Type-1 Node Assembly

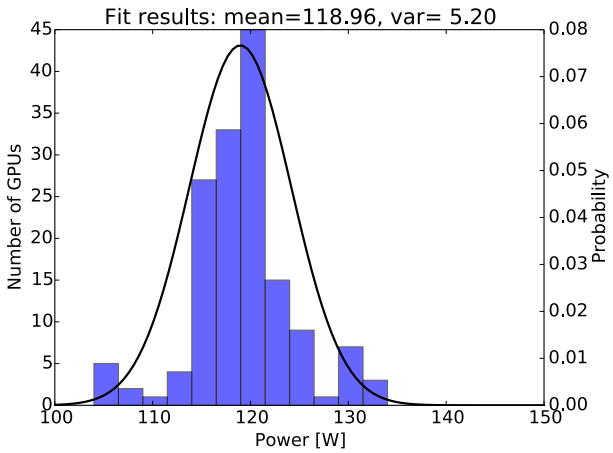


Active Power Distribution of Components

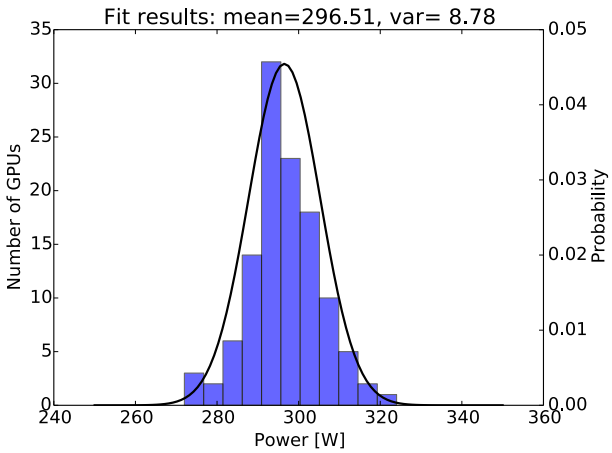
- The active distributions are fit into Gaussian distributions
- Extrapolated from ~90 to 5000 nodes in order to represent large scale



CPU

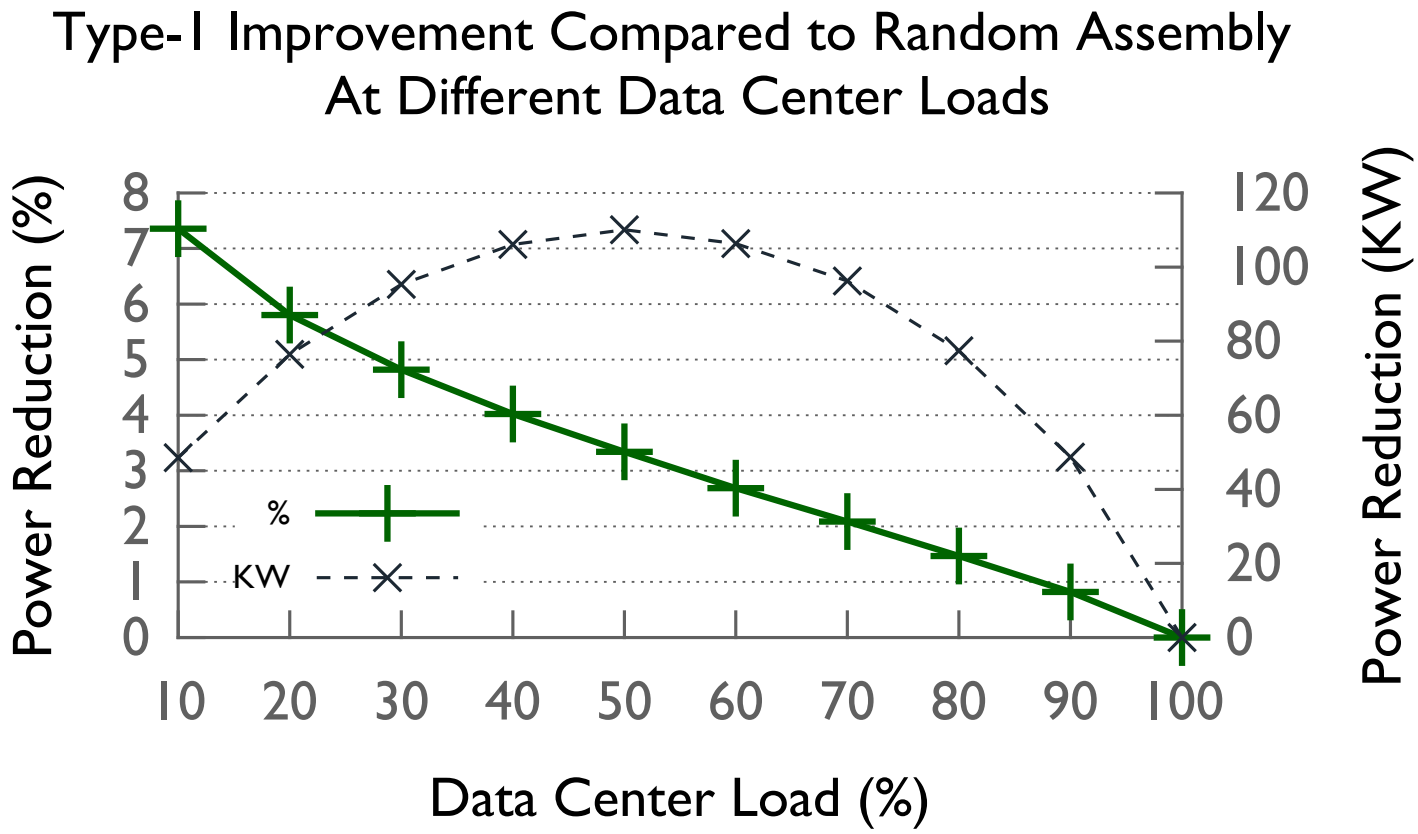


Memory



GPU

Categorized Assembly Power Reduction

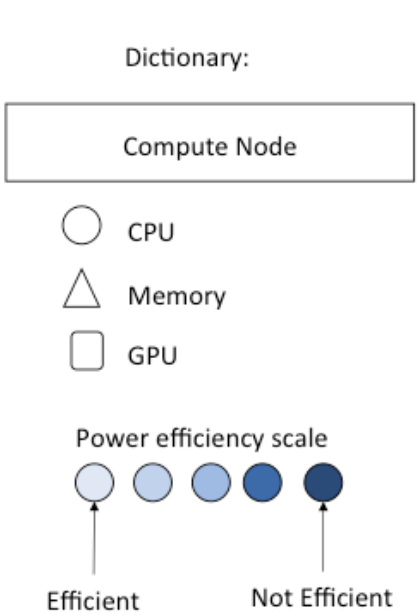


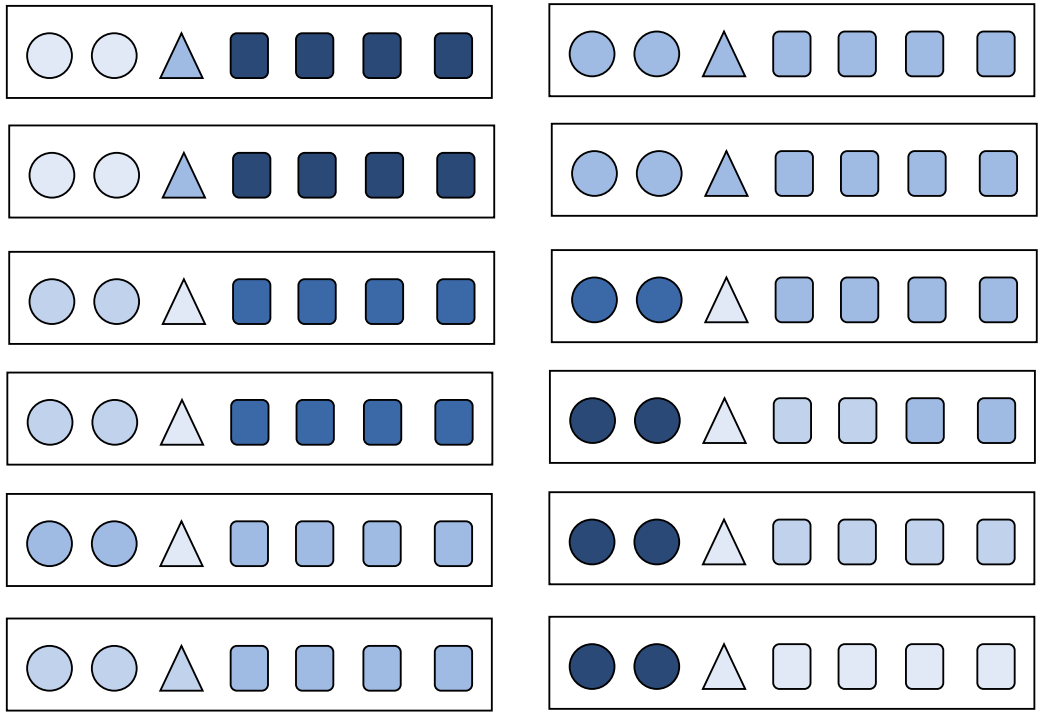
- Total power consumption of the components: 6.5 MW
- Summit is expected to consume 13 MW

Application Specific Node Assembly

- Node is assembled based on application characteristics:
  - a memory intensive application don't need efficient CPUs
- Performance variations can be mitigated (up to 16%)

Illustration of Type-2 Node Assembly

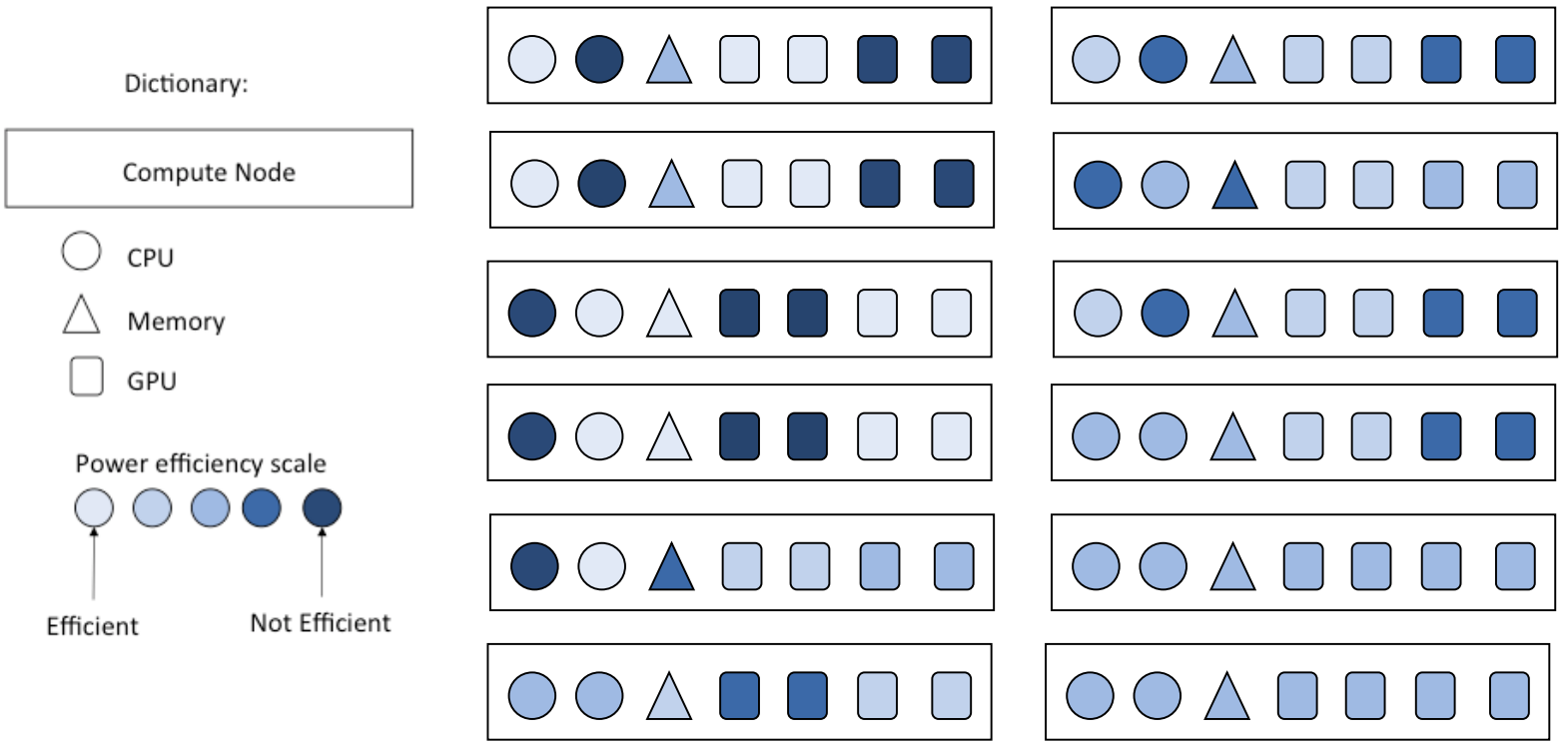




Balanced Power Node Assembly

- Makes total node power and average performance more predictable
- More suitable for cloud platforms

Illustration of Type-3 Node Assembly



Summary: Mitigating Power Variation

- Analyzed power variation of different components in the node
- Proposed three new node assembly techniques:
  - Categorized
  - Application Specific
  - Balanced Node Power

Outline

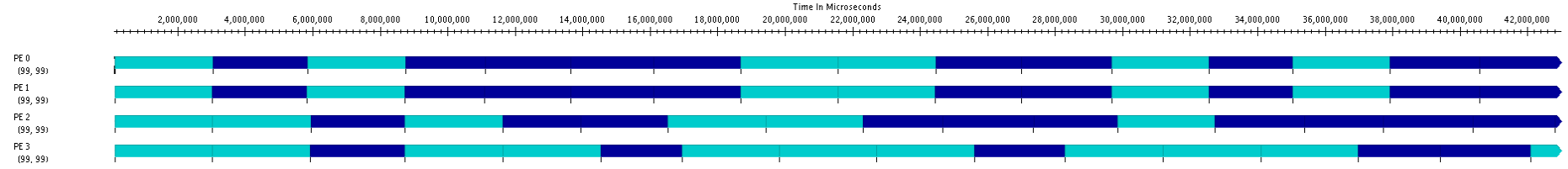
1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

Mitigating Application-Level Variations

- Applications might have different phases or kernels that execute simultaneously and have different “optimal frequency” levels
  - Unstructured or non-uniform applications
  - Applications with special threads: I/O, communication threads

- Runtime can decide the “optimal frequency” of each application task
  - Transparent from the application
  - No need for application modifications, insertion of directives
  - Higher efficiency by making automated fine-grain optimizations

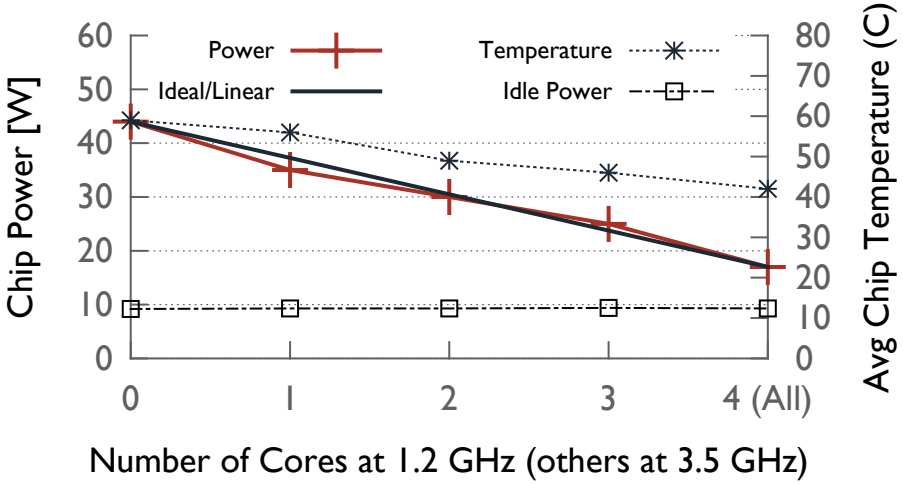
- What is “optimal frequency”?
  - Energy minimal frequency
  - Lower frequency without sacrificing from performance
  - Highest frequency under a power-constraint
  - Temperature restraining frequency



Per-core DVFS Support in Three Architectures

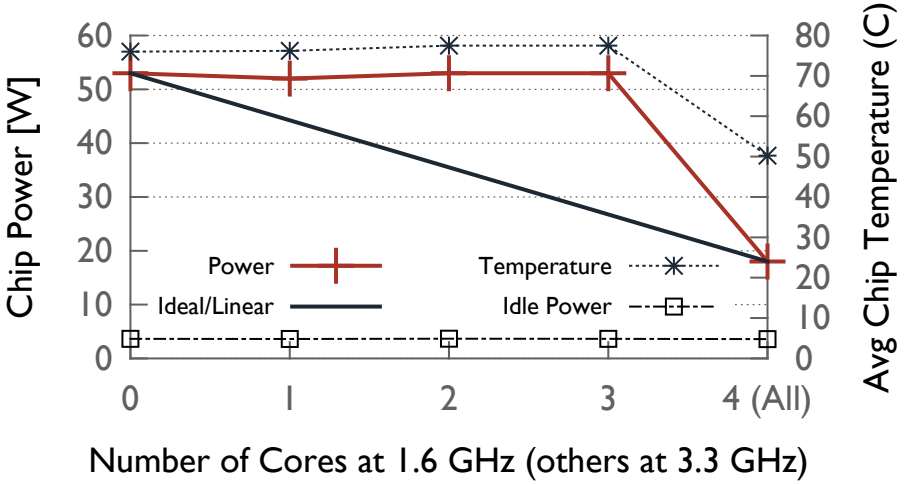
- Intel Haswell is the only platform that supports per-core DVFS in production.

Core Level Frequency Control Behavior on Haswell



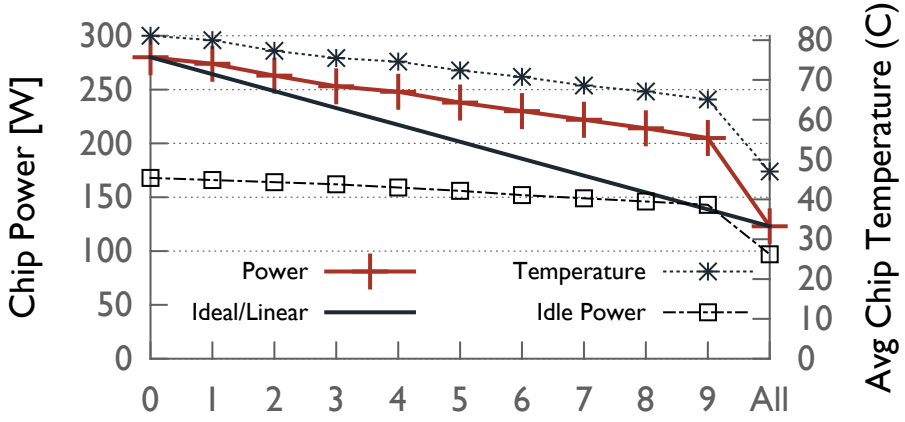
Number of Cores at 1.2 GHz (others at 3.5 GHz)

Core Level Frequency Control Behavior on Sandy Bridge



Number of Cores at 1.6 GHz (others at 3.3 GHz)

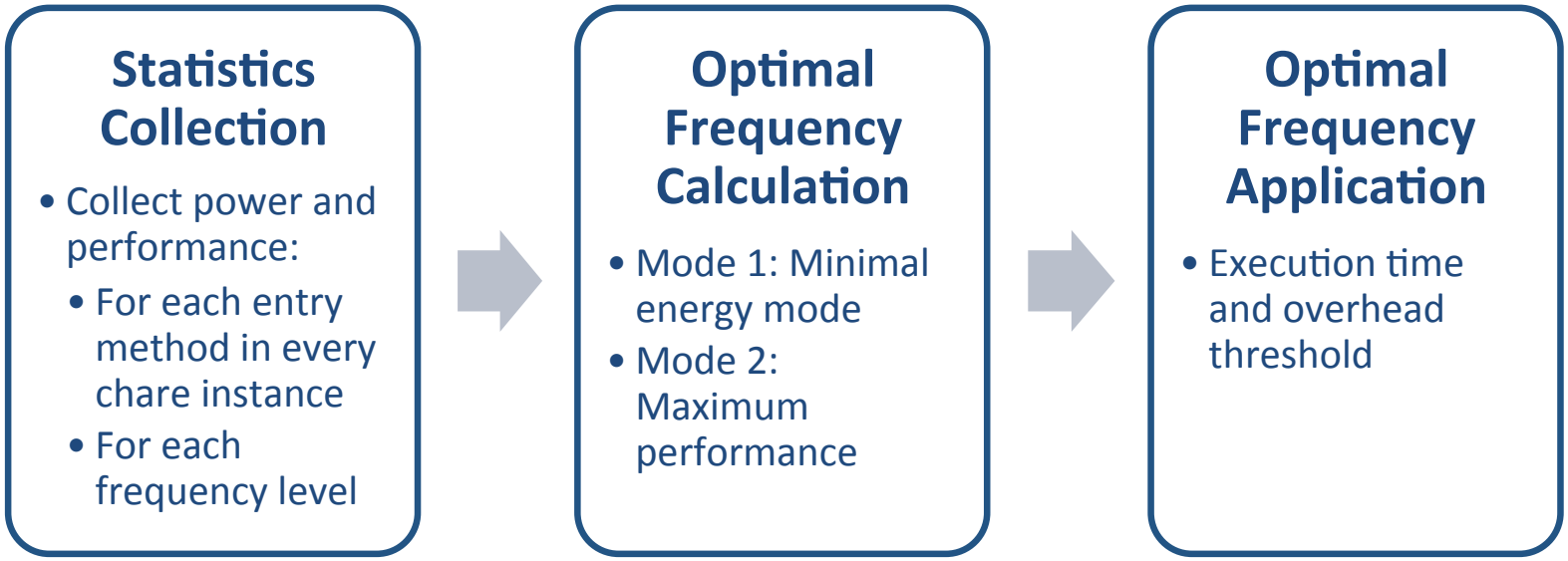
Core Level Frequency Control Behavior on Power8



Number of Cores with DVFS Applied

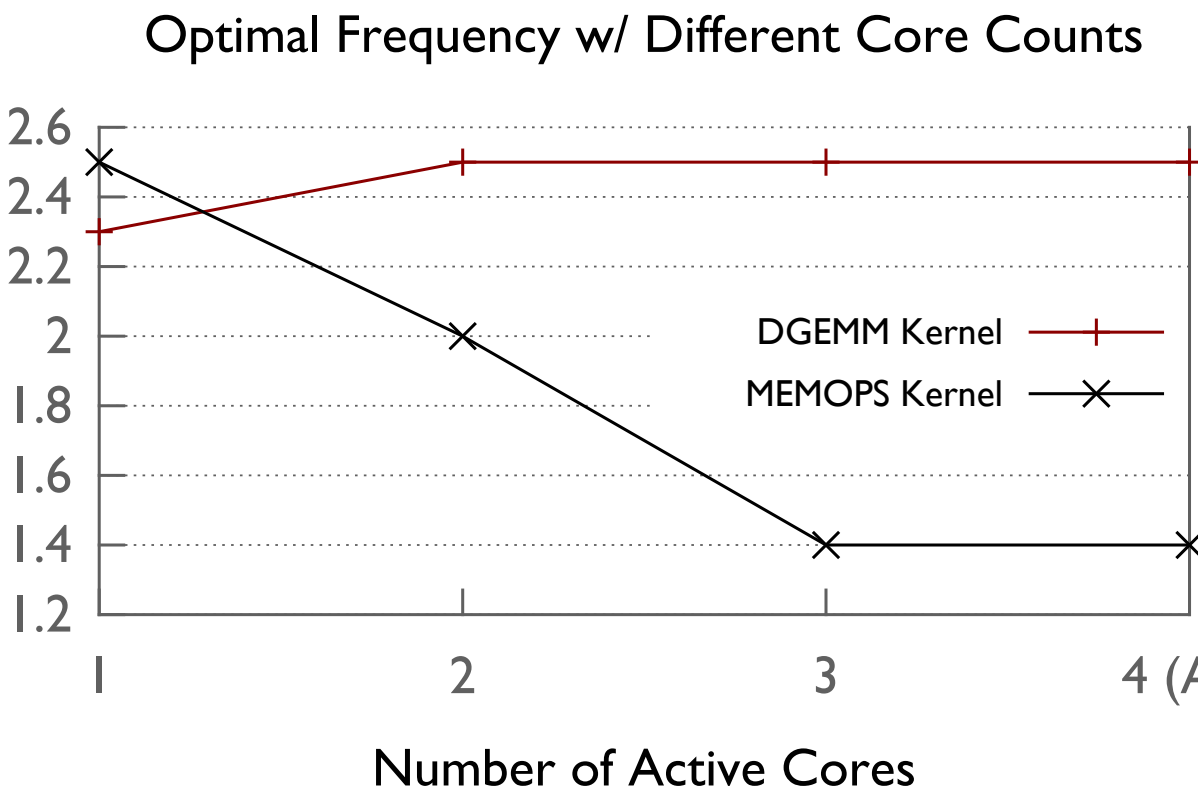
Runtime-based Function-Level Optimization Approach

- Charm++ “entry methods” naturally enables separation and control of different kernels of the applications.

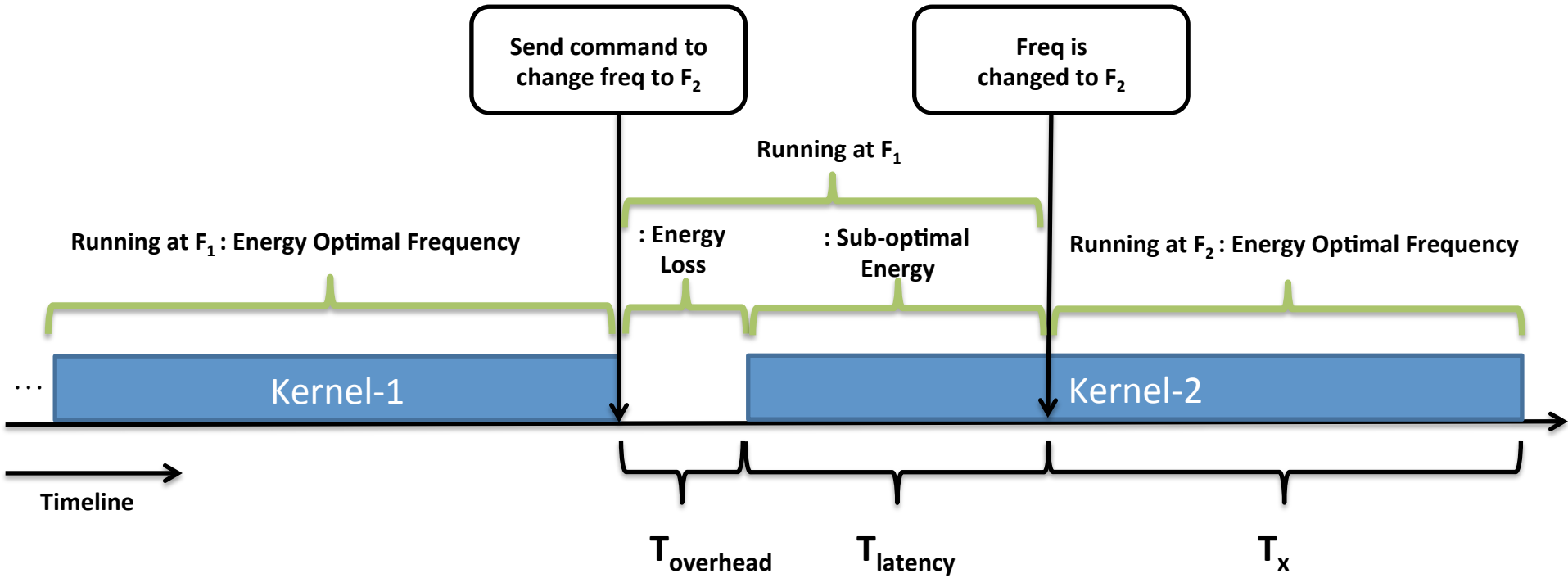


No Core-Level Power Data Available

- Use a locking mechanism − one core active at a time
  - Subtract static power of idle cores
- Does measurements one core reflect overall measurements?

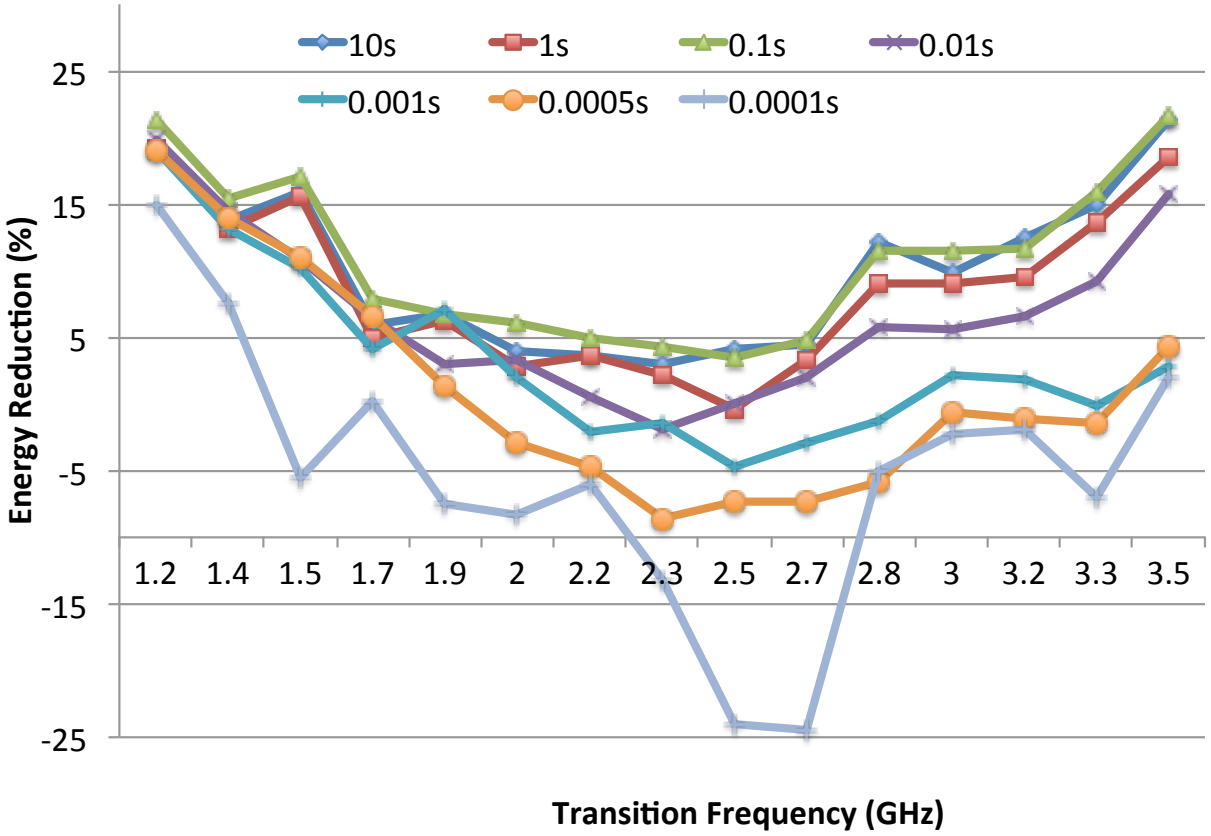


Entry-based DVFS



T_overhead = 2-5 microseconds
T_latency = up to 500 microseconds

Energy Reduction for Different Kernel Durations



- The optimal frequency of the target kernel is 2.3 GHz
- It does not worth transitioning from 1.7-2.8 GHz
- Shorter kernel durations have less benefit due to overhead and latency

Summary: Mitigating Application Variations

- Task-based runtime can do fine grained optimizations by optimizing each kernel of the application
- The performance of a kernel or task depends on what other cores are running
  - Need core-level power counters
- Per-core DVFS needs to have less latency and overhead to be practical

Outline

1. Introduction
2. A Dynamic Runtime Interacting with Data Center's Resource Manager
3. Variation Analysis: Power, Temperature, Frequency
4. Mitigating Frequency Variation
5. Mitigating Temperature Variation
6. Mitigating Power Variation
7. Mitigating Within Application Variations
8. Conclusion

Concluding Remarks

- Large scale systems exhibit variability due to various reasons and likely to continue exhibit in the future.
- Variability is bad for performance reproducibility and debugging.
- Removing the variability is inherently difficult.
- Instead runtimes or software systems should know and accommodate for variability.
- Support from HPC systems to enable power, temperature related measurements and controls are key in achieving this.

Thank you!