Energy-Efficient and Power-Constrained Techniques for Exascale Computing

Stephanie Labasan
Computer and Information Science
University of Oregon
17 October 2016

Overview

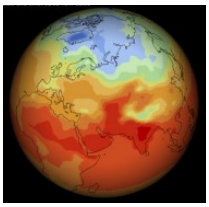
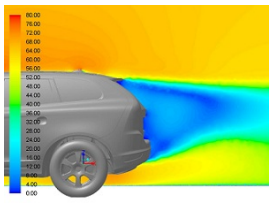
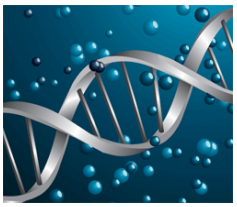
- **Motivation**: Power is becoming a leading design constraint in HPC
  - **Goal**: Save energy/power
- **Ideas**: Reduce CPU clock frequency or place a power cap on the CPU
- **Proposition for users**: Run **X%** slower, save **Y%** in energy/power

Outline

- Motivation & Background
- Energy-Efficient Techniques for HPC
- Power-Constrained Techniques for HPC
- Scientific Visualization
- Power-Aware Visualization & Future Work

What is High-Performance Computing?

- Enabling technology for scientific discoveries through simulation
  - Weather prediction, design simulation, genomics research, medical imaging, and more!
- Leverages computational capacity of several thousand interconnected processing units


Large Supercomputers

- Performance measured in FLOPS = **fl**oating-point **o**perations **p**er **s**econd
  - Linpack: highly-optimized compute-intensive standard benchmark
- 1 TeraFLOP = 10^{12} [1997]
- 1 PetaFLOP = 1K TeraFLOPS [2008]
- 1 ExaFLOP = 1M TeraFLOPS [exp 2023]

High Power Costs

- 1 MW of power costs $1M per year
- Supercomputing centers are paying ~$12M annually
- As machines get larger, cost rises, unless we innovate power-efficient techniques
  - Techniques may come from HW, but SW as well

Rank	System	Cores	Rmax (PFLOPS)	Power (MW)
1	Sunway TaihuLight, China	10.6M	93	15.4
2	Tianhe-2, China	3.12M	33.9	17.8
3	Titan, ORNL	560K	17.6	8.2
4	Sequoia, LLNL	1.57M	17.2	7.9

c/o Top500, Jun2016

What is Energy? What is Power?

Energy

- Units: Joules, kiloWatt-hour
- Total work done
- Monthly bill from electric company (e.g., $.11/kWh) – "power bill"

$$Energy = Power \cdot time$$

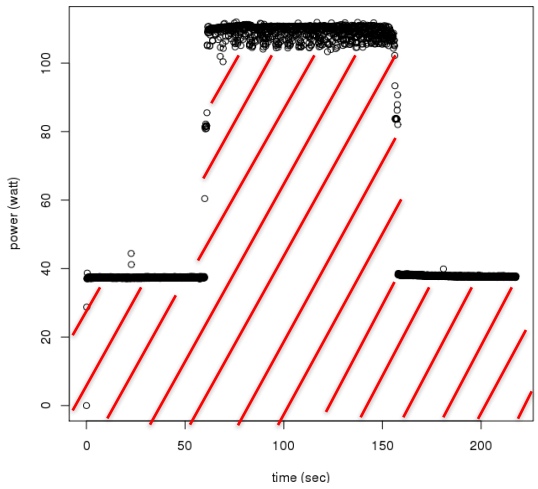
Power

- Units: Joules/second, Watts
- Instantaneous rate of energy usage
- Lightbulb ratings, microwaves, stereo systems

$$Power = \frac{Energy}{time}$$

CPU Power Usage Varies Over Time

- Base power usage for idle state
- When more HW components are engaged (cache, vector units, etc.), power rises
- Energy usage would be the area under this power curve



Computationally-intensive HPC benchmark (Linpack) running on small cluster

Save Energy? Save Power?

Energy

- For a single application execution, minimize energy consumption ($$)
- Energy-to-solution
- Mobile, desktop, workstations, data centers

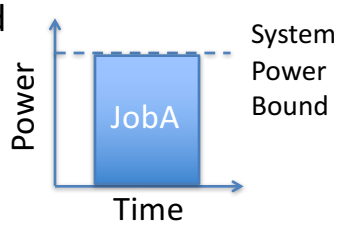
Power

- Reduce rate at which application consumes energy
- Save power != Save $$/energy
  - Run at 40W ↓ take 10s = 400J
  - Run at 30W ↓ take 15s = 450J ↑
- Supercomputers
- Increase concurrent jobs (maximize completed jobs)

Increasing Machine Throughput

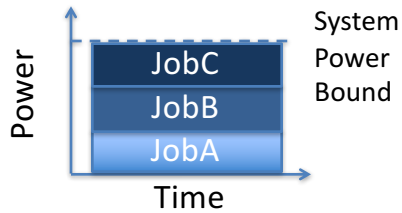
Scenario 1

- Launch a single job with maximum power allocation available
- No other concurrent jobs can be launched without exceeding the bound



Scenario 2

- Launch the same job at a lower power allocation, job runs longer
- Power available to launch other concurrent jobs → increase amount of science done



U.S. Department of Energy Perspectives on Exascale Challenges

Parameter	2010	"2018"	Difference Today & "2018"
System peak	2 PFLOPS	1 EFLOPS	O(1000)
Power	6 MW	~20 MW	~3
System memory	0.3 PB		
Node performance	125 GF		
Node memory BW	25 GB/s		
Node concurrency	12		
Total node interconnect BW	3.5 GB/s		
System size (nodes)	18,700		
Mean time to failure	Days	O(1 day)	- O(10)

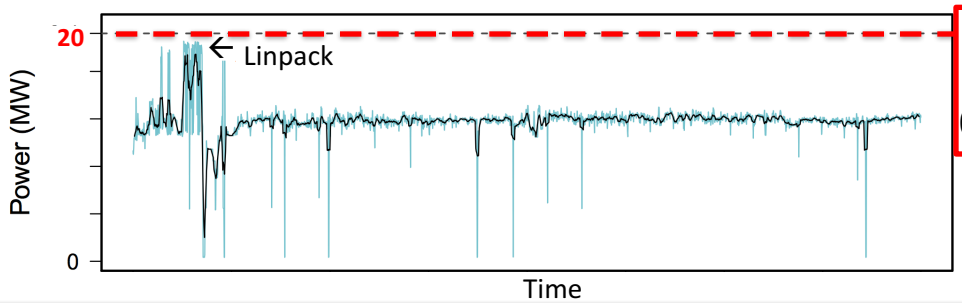
Parameter	2010	"2018"
System peak	2 PFLOPS	1 EFLOPS
Power	6 MW	~20 MW

Until now, innovations in power savings have been hardware driven. However, to reach 1 Exaflop/20 MW, software improvements may be needed.

Future Systems: Max Power ≤ 20MW

- Machine reaches maximum power assuming worst-case compute-bound application
  - i.e., even Linpack stays below 20 MW

Performance left unused!

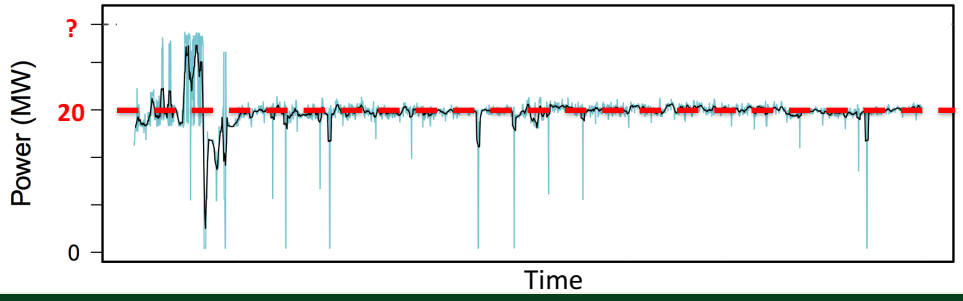


1 EF, peak

0.7 EF, "usable"

Future Systems: Max Power > 20MW

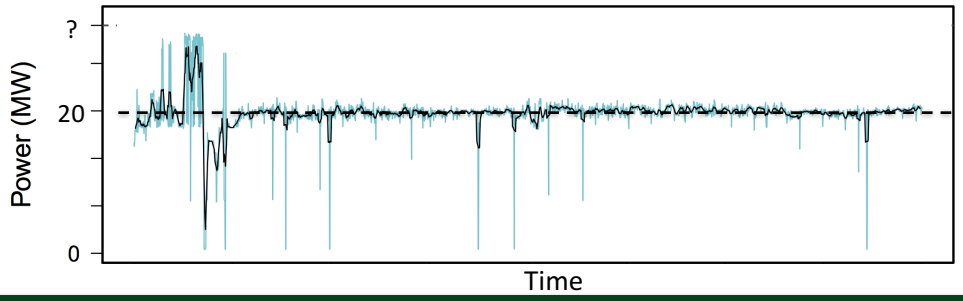
- Machine contains more compute capacity than can be running simultaneously under the power bound
- Controls needed to enforce power usage at 20 MW
- Power allocations are coordinated by a central manager



1 EF, "usable"

Exascale Power Challenges

- Power is the scarce resource
- Understanding performance under a power constraint is difficult
- Achieving max utilization of 1 exaflop/20MW may risk exceeding the power bound



1 EF, "usable"

Outline

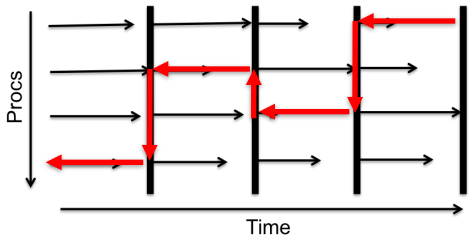
- Motivation & Background
- Energy-Efficient Techniques for HPC
- Power-Constrained Techniques for HPC
- Scientific Visualization
- Power-Aware Visualization & Future Work

What is Energy Efficiency?

- **Goal**: Use less energy to reach solution
  - i.e., reduce costs
- **How**: Exploit parallel execution behaviors, slow processor down with little to no change in runtime

Load Imbalances Create a Performance Challenge

- Massively-distributed bulk-synchronous applications
- Global synchronization barriers at computation milestones
- Performance determined by last processor to arrive at the barrier
  - Speed up slower processors
  - Slow down processors with slack (i.e., waiting on dependencies)



Power Saving Technique: Reduce CPU Frequency (DVFS)

- **Outcome**: Takes longer to run, but uses less power
- **Why**: Non-linear relationship between frequency and power → $P \propto f^3$
  - Reduced clock frequency results in less power consumption
  - but, subcomponents still consume power at the same rate

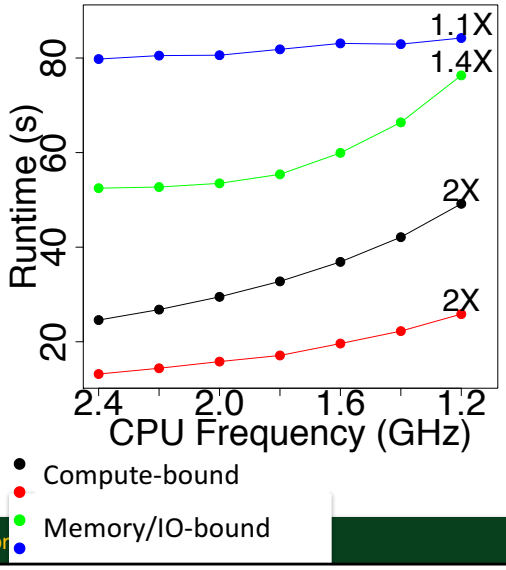
✔ Will lead to power savings

✔ May lead to energy savings, may not

✖ Does not guarantee a specific power consumption

Performance Under DVFS

- Trade-offs between energy savings and runtime are dependent on application bottlenecks
- DVFS typically only impacts the processor and private caches

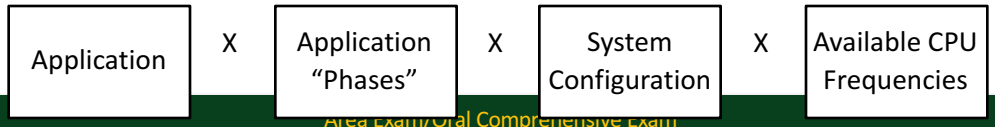


Energy-Efficient Approaches

- Offline Approaches
- Online Approaches
  - Interval-Based
  - Iteration-Based
  - Slack-Based

Offline Approaches

- Necessary step to understand value proposition
- Approaches:
  - Simple: Single CPU frequency per application
  - Advanced: Multiple CPU frequencies per application
- High overhead in collecting energy profiles, not ideal for production



Interval-Based Online Approaches

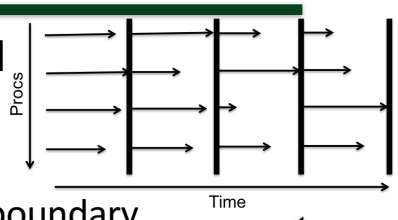
- Energy-savings decisions are assessed at regular intervals of time
- Future decisions are based on execution behaviors of previous interval
- What metrics to use?
  - Instructions per cycle (IPC) Low IPC = memory stalls slow exec rate
  - Misses per operation (MPO) High MPO = more memory accesses
  - Beta metric: intensity of off-chip accesses (all misses)
  - Leading loads: memory-bound due to *first* miss

Ex: CPUFreq/CPUSpeed/CPUPower

- Linux drivers to enable OS to statically/dynamically change CPU frequency to save power
- Several "governors" = software heuristics for changing frequency
  - Static
    - Performance
    - Powersave
    - Userspace
  - Dynamic
    - Ondemand
    - Conservative
- Not critical path aware, 15-25% power reduction can be achieved

Iteration-Based Online Approaches

- Assume application iterations are identical
- Decide frequency for next iteration
- **Ex: Jitter**
  - Must modify source code to identify iteration boundary
  - Slow processors off critical path to arrive just-in-time, 8% energy savings, 2.6% increase in runtime
- Works well on applications where critical path does not move between iterations
- Increasing number of applications are non-temporal



Slack-Based Online Approaches

- **Slack**: Time spent blocked in an MPI communication call
  - i.e., processes not on the critical path
- **Goal**: Find *lowest* frequency where no slack is incurred
  - i.e., processes arrive at the same time

Ex: Adagio

- No markup to application
- Approximates lowest frequency as it may be unavailable on given CPU
- Achieves 20% energy savings with <1% slowdown by being critical path aware

Comparing Approaches

Energy-efficient approaches explore trade-offs between energy usage and performance

Assume we want to apply an energy-efficient approach to an application:

Approach A:

- Saved 5% in energy usage
- Increased runtime by 2%

Approach B:

- Saved 8% in energy usage
- Increased runtime by 4%

Which approach is better?

Evaluation Metrics

- Create a single scalar metric quantifying slowdown and energy usage
- **Energy-delay product** (EDP) $ED^nP = E \times D^n$
  - *n* indicates preference towards acceptable slowdown
- **Weighted ED²P** $ED^2P_w = E^{(1-\delta)} \times D^{2(1+\delta)}$
  - User-specifies preference towards energy savings or performance slowdown

Additional Metric: Energy Reliability

- Exascale systems will experience more frequent failures (at least once a day)
- Energy usage increases to provide reliability mechanisms
  - e.g., cooling, checkpoints/restarts, backup components
- Slowdowns due to DVFS increase likelihood of encountering a failure, impacting overall energy savings

Outline

- Motivation & Background
- Energy-Efficient Techniques for HPC
- Power-Constrained Techniques for HPC
- Scientific Visualization
- Power-Aware Visualization & Future Work

What is Power-Constrained?

- Power is the constrained resource at exascale
- **Goal**: Maximize power efficiency by increasing system job throughput
- **How**: Applications will need to be flexible with resource allocations (e.g., power, nodes, execution time)

HPC System Design Approaches

- **Worst-Case Provisioning** (current approach)
  - Assume every node consumes peak power simultaneously
- **Hardware Overprovisioning**
  - More nodes exist than can be fully powered simultaneously under the bound
  - Benefits:
    - Dynamic system configuration based on application execution
    - Increase system throughput by reallocating power where it is most needed

Power Saving Technique: Reduce Concurrency

- Change number of threads executing in parallel region
- Less overhead than DVFS
- Beneficial on architectures with shared clock frequencies between CPU and shared cache

✔ Will lead to power savings

✔ May lead to energy savings, may not

✖ Does not guarantee a specific power consumption

Power Saving Technique: Power Capping

- Enforce an upper bound on power consumption, hardware toggles CPU frequency/voltage
- Supported by Intel, AMD, IBM processors

✔ Will lead to power savings

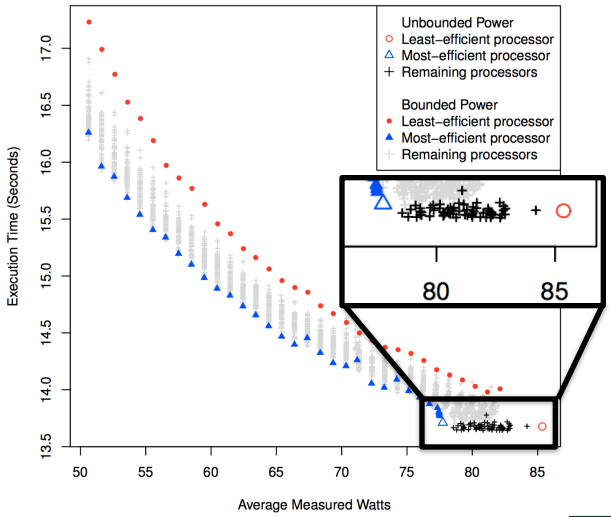
✔ May lead to energy savings, may not

✔ Guarantees specific power consumption

Performance under DVFS has been well-studied, but performance under a power cap will pose new challenges.

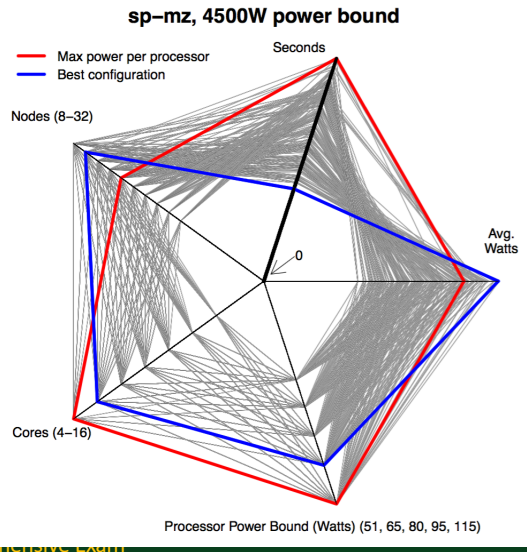
CPU Performance Under a Power Bound

- Processor manufacturing is imperfect
- **Unbounded power**: Constant execution time, variation in power draw
- **Bounded power**: Constant power draw, variation in execution time



Job Performance Under a Power Bound

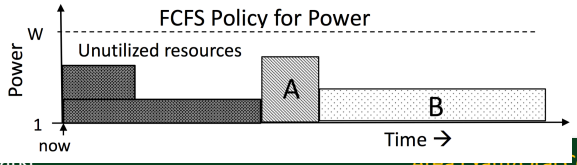
- Explore configuration space ≤ job-level power bound
  - 3 inputs: power bound, cores, nodes
  - 2 outputs: seconds, avg. watts
- Initial configuration: 115W, 16 cores/node, worst time
- Best configuration: 95W, 14 cores/node, 2X faster time



System Performance Under a Power Bound

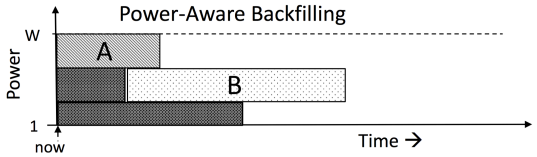
First Come First Serve ✖

- Goal: Create fairness across jobs on machine shared by many
  - Job A: power request exceeds current available resources → long queue time
  - Job B: request fits in resources, but must wait until Job A finishes



Power-Aware Backfilling ✔

- Goal: Use as much power when scheduling jobs
  - Job A: Adjust request → extend runtime, but reduce queue time
- Jobs submit a "laundry list" of tunable knobs (e.g., concurrency, algorithm, slowdown)



Dynamically Reallocate System Power

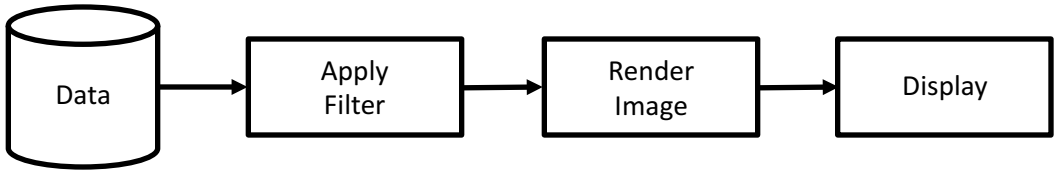
- Further benefits job performance *as they're running*
  - **Ex: powsched**
- Central manager dynamically shifts power between nodes based on current CPU power usage
  - Agnostic to jobs running on each node
- Must guarantee system power bound is not exceeded, while fully utilizing all power

Outline

- Motivation & Background
- Energy-Efficient Techniques for HPC
- Power-Constrained Techniques for HPC
- Scientific Visualization
- Power-Aware Visualization & Future Work

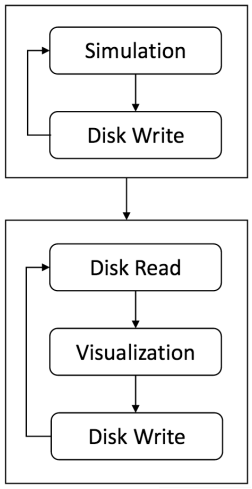
Scientific Visualization

- Central actor in scientific discovery process:
  - Communicate
  - Explore
  - Validate



Post-Processing (Traditional Model)

- Simulation writes data to disk at regular time steps
- Visualization reads in data, performs analysis, and writes results to disk



Time step 20, 40, 60, …

Apply filter, render image

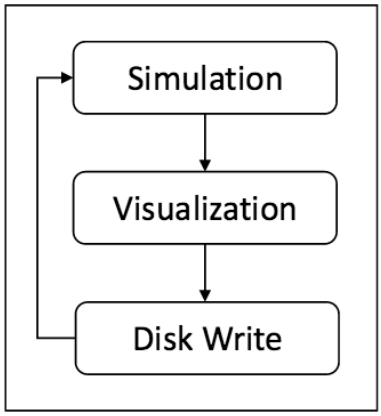
Challenges of Post-Processing Model

- Simulations are greatly increasing in the amount of data per time step they can generate
- I/O bandwidth limitations widen gap between time steps

Step 0 Step 500 Step 1000

Transition to In Situ to Bypass I/O Gap

- Data analysis and visualization occur while simulation is running
  - Share job resource allocations
- No storage needed between simulation and analysis



Outline

- Motivation & Background
- Energy-Efficient Techniques for HPC
- Power-Constrained Techniques for HPC
- Scientific Visualization
- Power-Aware Visualization & Future Work

Power-Aware Visualization

- Very limited research in this space
- Research efforts so far:
  - What is the difference in power consumption between traditional post-process approach and in situ?
  - How do software knobs (e.g., concurrency, algorithm, parallel programming model) affect energy and power savings of visualization algorithms?
  - How does in situ data movement (e.g., no movement, intra-node, inter-node) impact power?

Dissertation Proposal

- Interests lie in how power constraints will impact visualization
- **Goal**: Understand what parameters impact energy and power usage of visualization routines
- **How**: Use energy-efficient and power-constrained techniques to build a power-aware visualization framework for exascale computing

Future Work

- What tunable knobs exist, such that the application is moldable to the available resources?
- How do the knobs apply to different visualization routines?
- How much speedup can be gained from shifting power between nodes during phases in the pipeline?

Summary & Questions

- Power will be a key challenge at exascale
- All system components will need power-efficient innovations
- Power-aware scientific visualization is important to continued scientific discovery

"Energy-Efficient and Power-Constrained Techniques for Exascale Computing"
Stephanie Labasan
slabasan@cs.uoregon.edu

Backup

FLOPS-Focused

- Measuring performance in FLOPS has created two problems:
  1. Supercomputers consume high electrical power
  2. Supercomputers generate high amounts of heat, necessitating large cooling infrastructures for system reliability

A New Performance Metric

- Green500 list complements Top500
- Created in Nov 2007
- Ranks by performance-per-watt
- Promotes improved energy efficiency, system reliability

Rank	System	GFLOPS/ Watt	Power (kW)	Top500 Rank
1	Shoubu, Japan	6.7	150	94
2	Satsuki, Japan	6.2	46.9	486
3	Sunway TaihuLight, China	6.1	15371	1
4	GSI Helmholtz, Germany	5.3	57.2	440

c/o Green500, Top500, Jun2016

Moore's Law & Dennard Scaling

Moore's Law

- # of transistors per chip doubles every 18-24 months at constant cost
- The smaller the transistor size, the faster it can switch states (i.e., frequency)

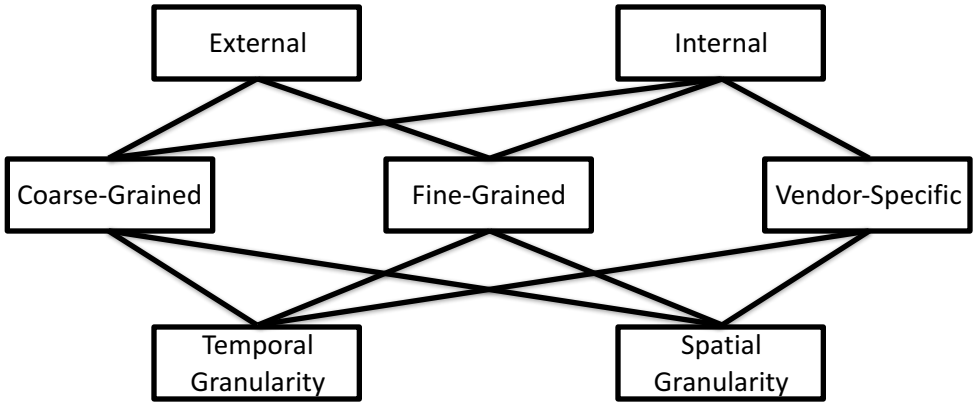
Dennard Scaling

- Voltage and current should be proportional to the linear dimensions of a transistor
  - Thus, as transistors shrank, so did necessary voltage and current; power is proportional to the area of the transistor.

Breakdown of Dennard Scaling

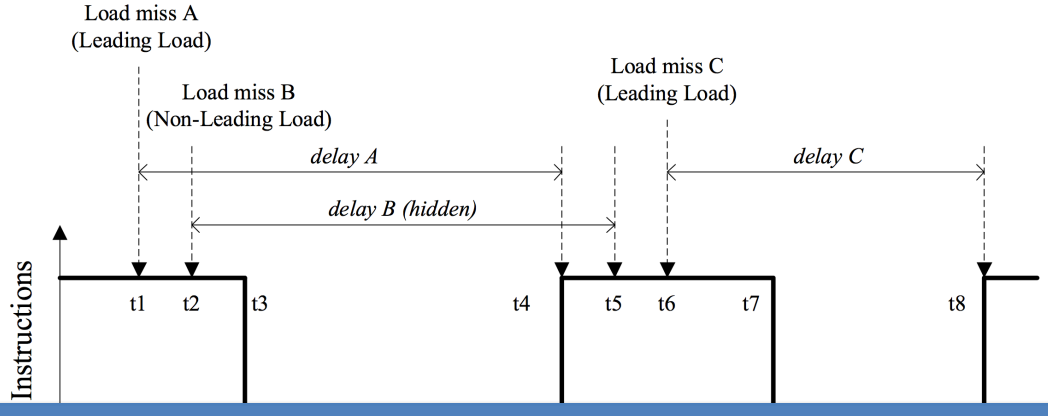
- Dennard ignored leakage power, which sets a base power consumption per transistor
  - As transistors shrank in size, power density increases leakage power does not scale with size
  - Hence, frequency has stagnated to around 4 GHz since 2006

Power Monitoring & Control



Lots of different tools for monitoring and controlling power. Not the focus of this presentation, but is relevant to enabling this research.

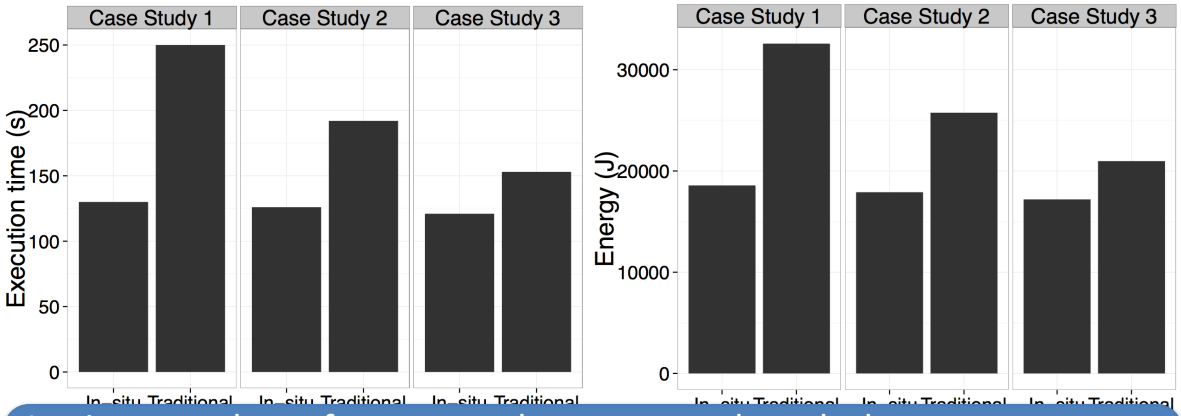
Leading Loads



Only first memory access can stall the pipeline. Do not count subsequent misses occuring before the leading load has returned.

c/o "Im
Predictor on Commodity Processors

Post-Process vs. In Situ Power Usage



In situ completes faster, uses less energy, though the average power usage is higher than post-processing. Reducing I/O times with in situ is more energy efficient.

In Situ Data Movement Strategies

Use the same compute resources as the simulation, but have different data movement patterns

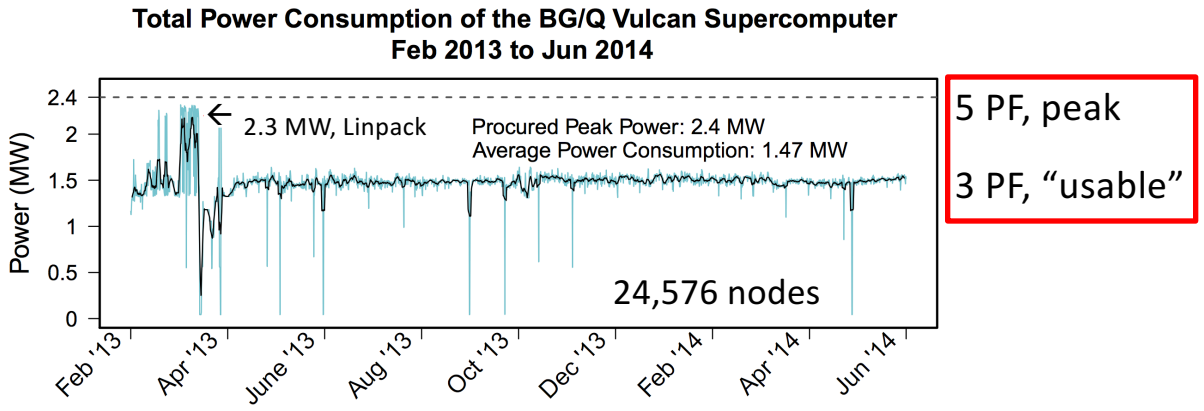
- **No data movement**: Analysis uses all the compute nodes allocated to simulation, which may be optimal for simulation, not analysis
- **Off-node**: Analysis uses a subset of nodes, frequency of data transfer limited by I/O bandwidth

Up to a certain point does using all cores per node benefit execution time and energy usage (with high power consumption). Using fewer cores per node is a more scalable solution for performance and energy and power usage.

Classifying Application Behaviors (cont'd)

- Beta metric:
  - Intensity of off-chip (memory) accesses
  - 1: Performance will scale with a reduction in frequency
  - 0: Performance is independent of the frequency
- Leading loads:
  - Track number of cycles spent performing non-speculative read resulting in LLC miss

Current System Power Usage



5 PF, peak

3 PF, "usable"

c/o T Patki