Computational Sprinting

Arun Raghavan[*], Yixin Luo[+], Anuj Chandawalla[+],
Marios Papaefthymiou[+], Kevin P. Pipe[+#],
Thomas F. Wenisch[+], **Milo M. K. Martin**[*]

University of Pennsylvania, Computer and Information Science[*]
University of Michigan, Electrical Eng. and Computer Science[+]
University of Michigan, Mechanical Engineering[#]

- **You are free:**
  - to **Share** — to copy, distribute, display, and perform the work
  - to **Remix** — to make derivative works
- **Under the following conditions:**
  - **Attribution**. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  - **Share Alike**. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

- For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to:

  http://creativecommons.org/licenses/by-sa/3.0/us/

- Any of the above conditions can be waived if you get permission from the copyright holder.
- Apart from the remix rights granted under this license, nothing in this license impairs or restricts the author's moral rights.

Overview of My (Other) Research

- **Multicore memory systems**
  - Adaptive cache coherence protocols
  - Memory consistency: specification & implementation
  - "Why On-chip Cache Coherence is Here to Stay" *Communications of the ACM*, July 2012
- **Transactional memory**
  - Semantics (what does "atomic" really mean?)
  - Extending transaction sizes & handling overflow
  - Conflict avoiding hardware via repair (true & false sharing)
- **Hardware support for security**
  - Goal: C/C++ as safe and secure as Java
  - Hardware/compiler co-design to provide memory safety

Talk Overview: Computational Sprinting

- Computational Sprinting
  - **Unsustainable power for short, intense bursts of compute**
- Feasibility study [HPCA'12]
  - Explored thermal, electrical, and architectural feasibility
  - Simulation results:
    - Significant responsiveness improvements in short bursts
    - With same dynamic energy consumption
- Preliminary results with sprinting on prototype-proxy
  - Characterize real energy/performance behavior
  - Sprinting can improve energy efficiency due to race to halt

Computational Sprinting and Dark Silicon

- A Problem: "Dark Silicon" a.k.a. "The Utilization Wall"
  - Increasing power density; can't use all transistors all the time
  - Cooling constraints limit mobile systems
- One approach: Use few transistors for long durations
  - Specialized functional units [Accelerators, GreenDroid]
  - Targeted towards sustained compute, e.g. media playback
- Our approach: Use many transistors for short durations
  - Computational Sprinting by activating many "dark cores"
  - **Unsustainable power for short, intense bursts of compute**
  - Responsiveness for bursty/interactive applications
- Our goal: responsiveness of 16W chip in 1W platform

Is this feasible?

Sprinting Challenges and Opportunities

- Thermal challenges
  - How to extend sprint duration and intensity?

    Latent heat from *phase change material* close to the die

- Electrical challenges
  - How to supply peak currents? **Ultracapacitor/battery hybrid**
  - How to ensure power stability? **Ramped activation (~100µs)**

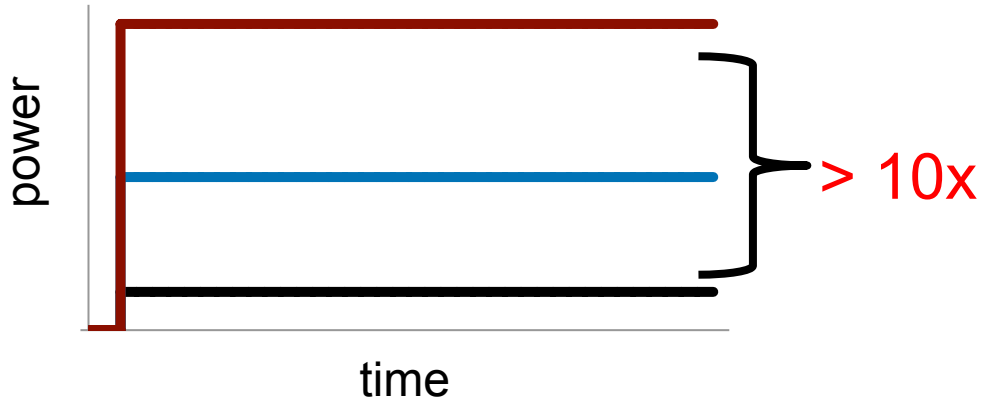
- Architectural challenges
  - How to control sprints? **Thermal resource management**
  - How do applications benefit from sprinting?

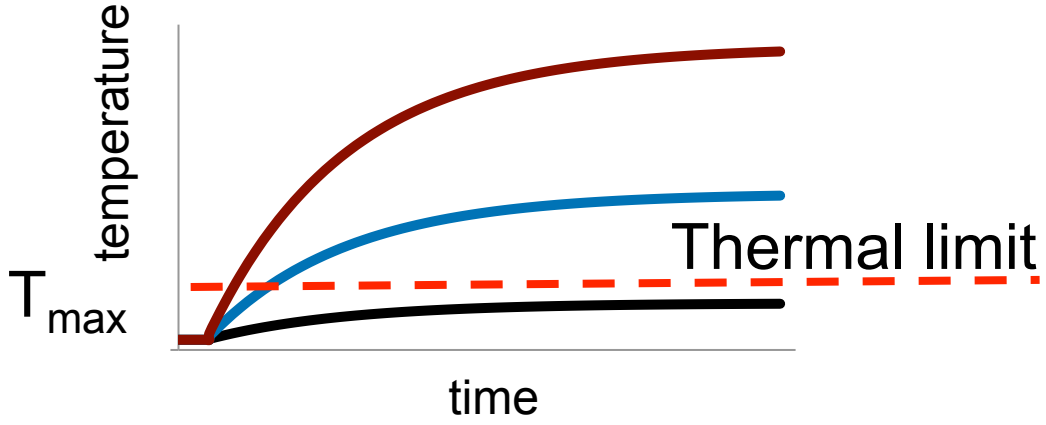
    10.2x responsiveness for vision workloads
    via a 16-core sprint within 1W TDP

Outline

- **Motivation: "Dark Silicon" and interactive apps**
- Computational Sprinting
- Feasibility Study
- Performance Evaluation
  - Simulation results
  - Characterization of a real system
- Conclusion

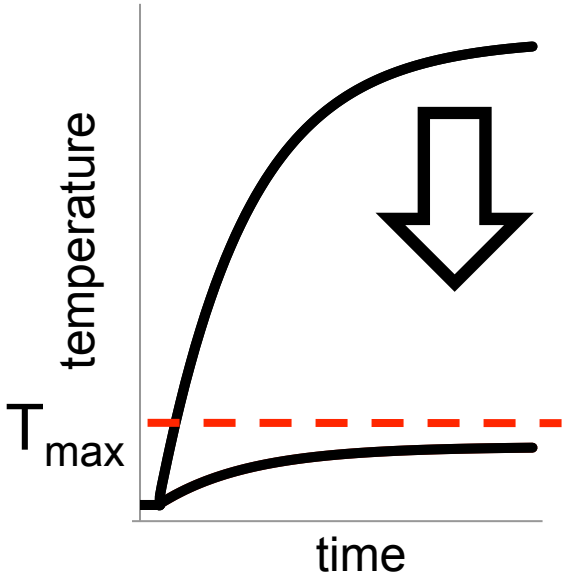
Power Density Trends for Sustained Compute

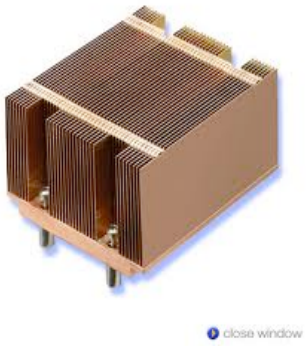




How to meet thermal limit despite power density increase?

Option 1: Enhance Cooling?



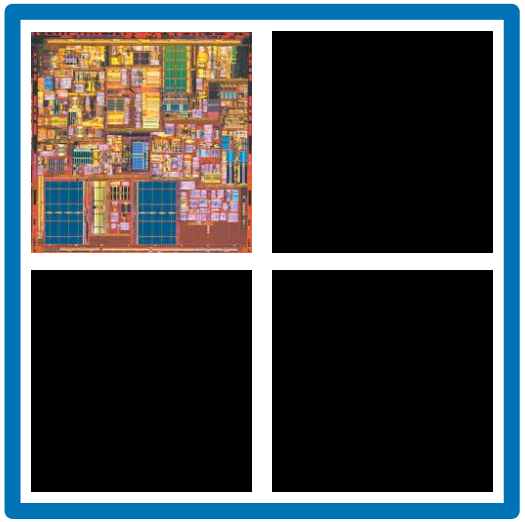


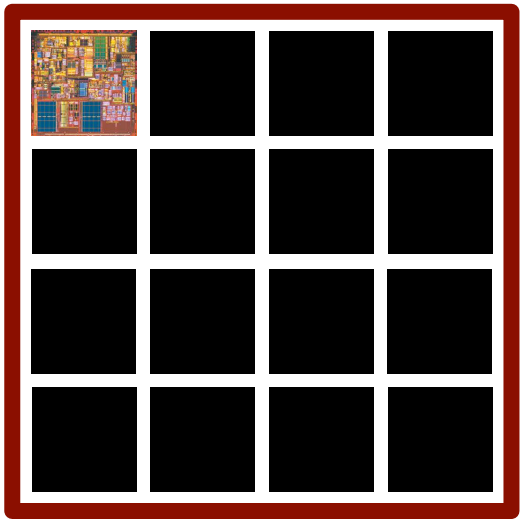
Mobile devices limited to passive cooling

Option 2: Decrease Chip Area?

Reduces cost, but sacrifices benefits from Moore's law

Option 3: Decrease Active Fraction?

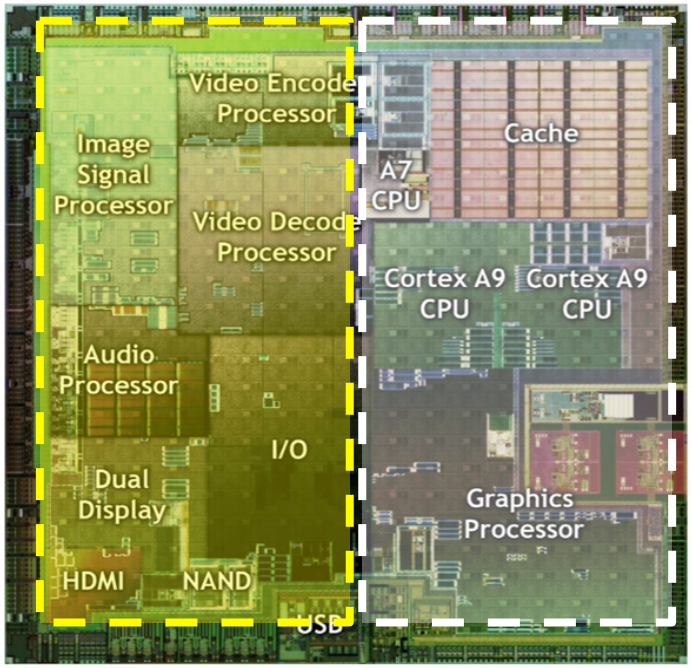




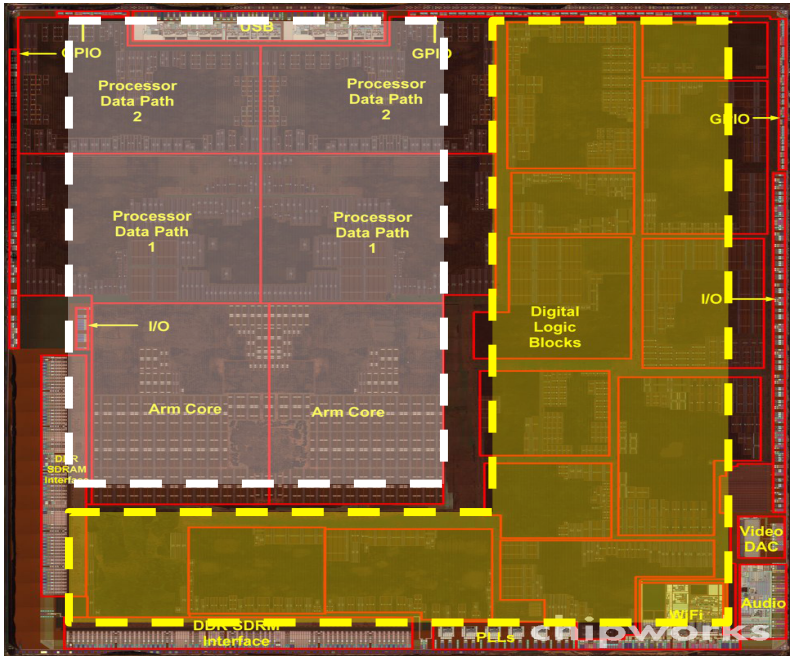
How do we extract application performance from this "dark silicon"?

Accelerator Cores?

- **Heterogeneous cores**
  [Conservation Cores ASPLOS'10, GreenDroid IEEE Comm., QsCores MICRO'11]
  - Activate different parts of chip based on application
- **Mobile chips already employ accelerators**



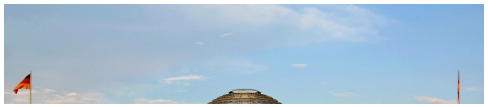
NVIDIA Tegra 2 (49 mm^2)

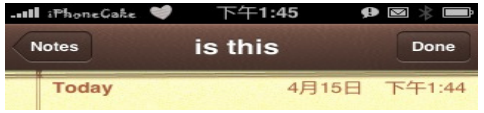


Apple A5 (122 mm^2)

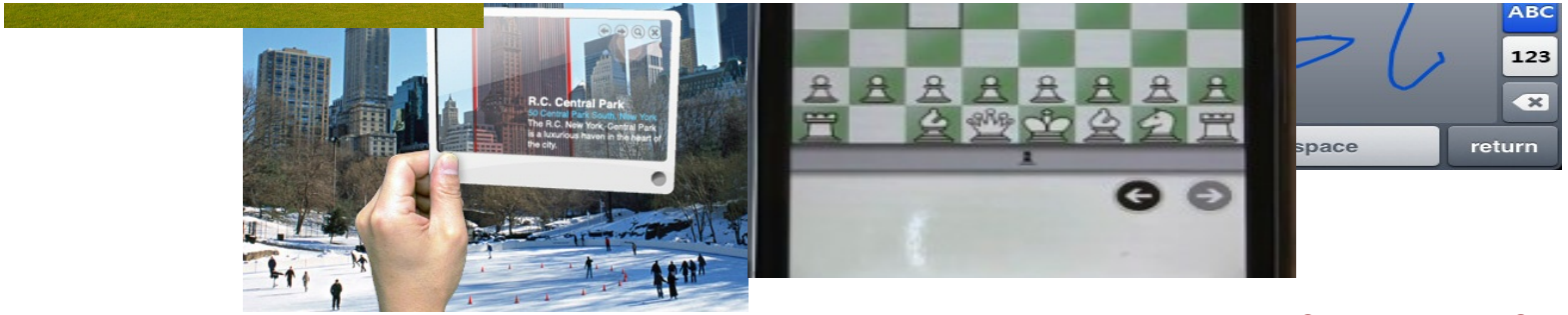
Design for Responsiveness

- **Observation: today, design for sustained performance**
- **But, consider emerging interactive mobile apps…**
  [Clemons+ DAC'11, Hartl+ ECV'11, Girod+ IEEE Signal Processing'11]
  - Intense compute bursts in response to user input, then idle
  - Humans demand sub-second response times
    [Doherty+ IBM TR '82, Yan+ DAC'05, Shye+ MICRO'09, Blake+ ISCA'10]



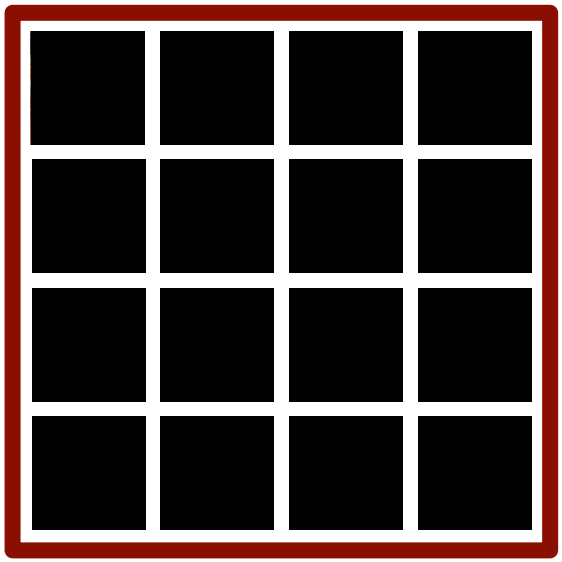
Peak performance during bursts limits what applications can do



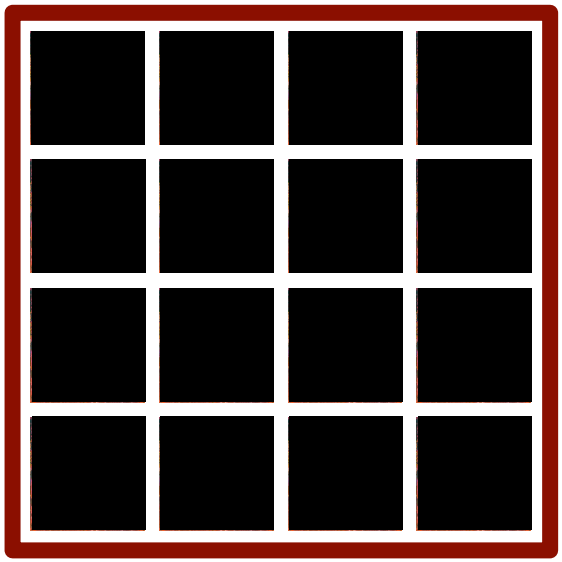
Designing for Responsiveness

Computational Sprinting

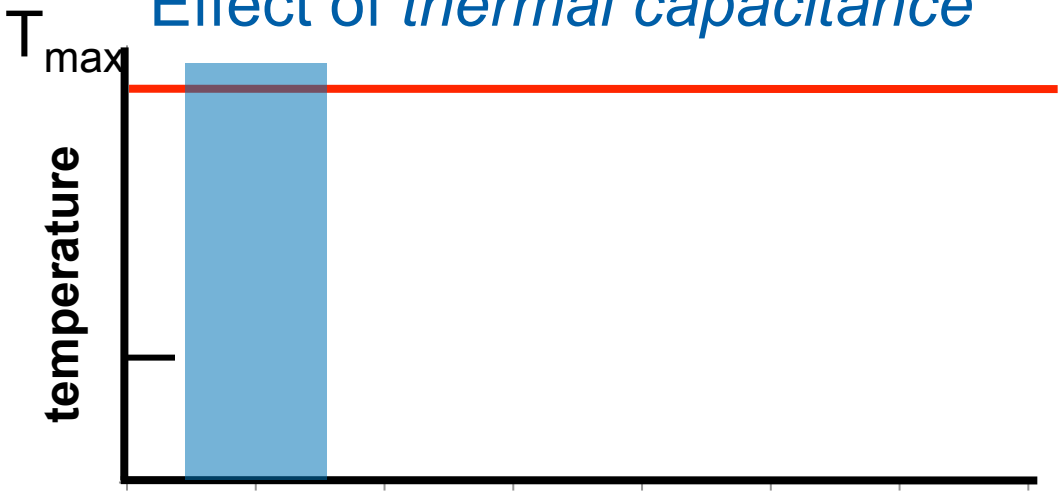
Parallel Computational Sprinting



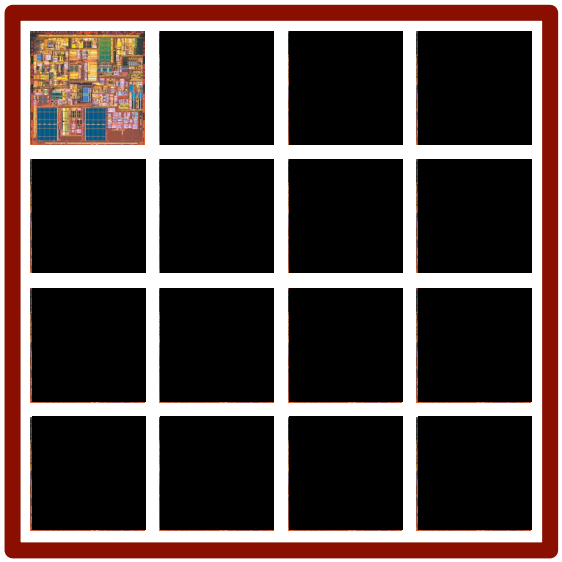
Parallel Computational Sprinting



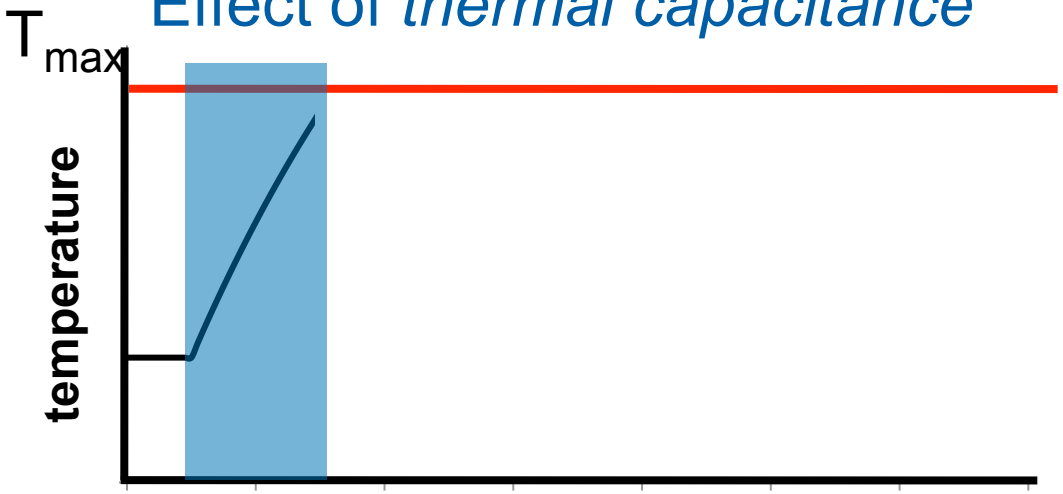
Effect of *thermal capacitance*



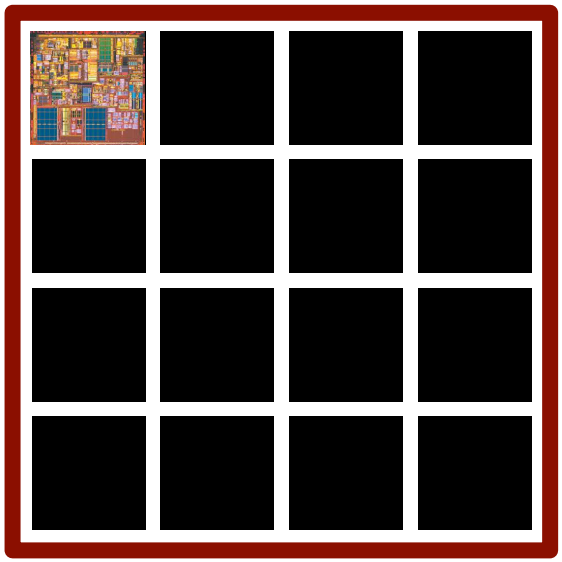
Parallel Computational Sprinting

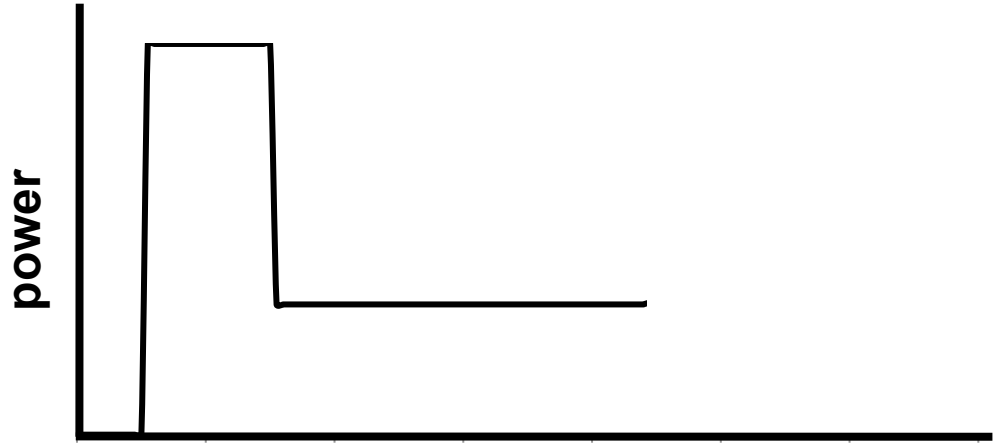


Effect of *thermal capacitance*

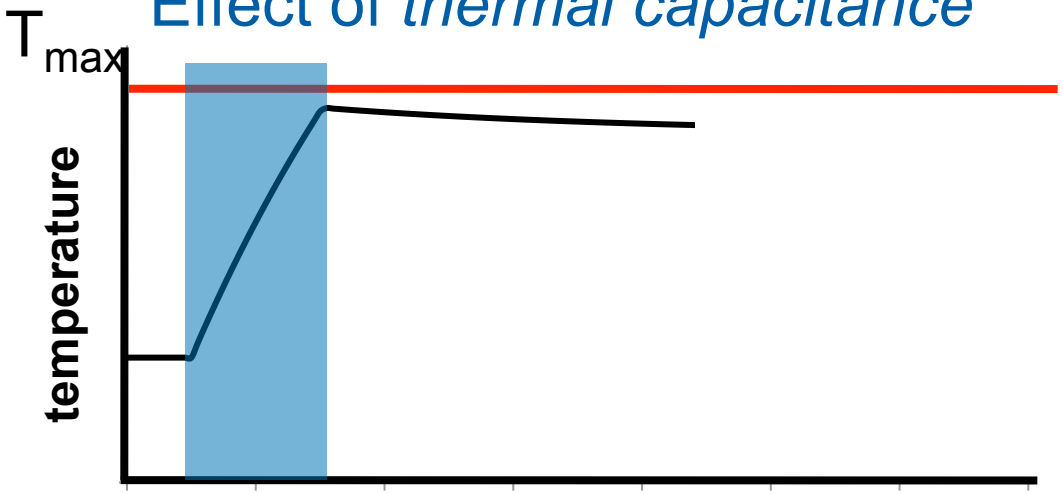


Parallel Computational Sprinting

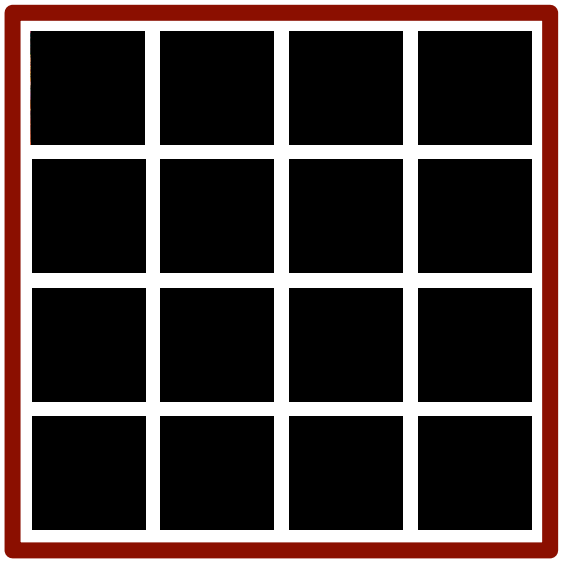


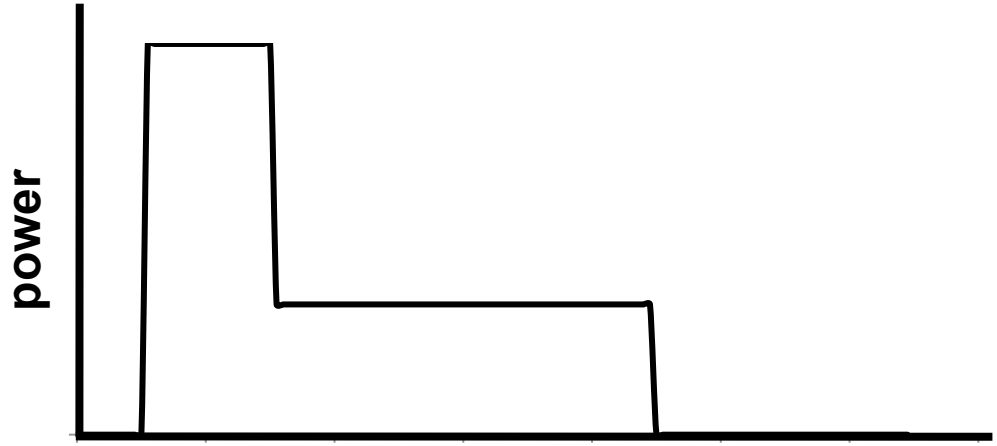


Effect of *thermal capacitance*



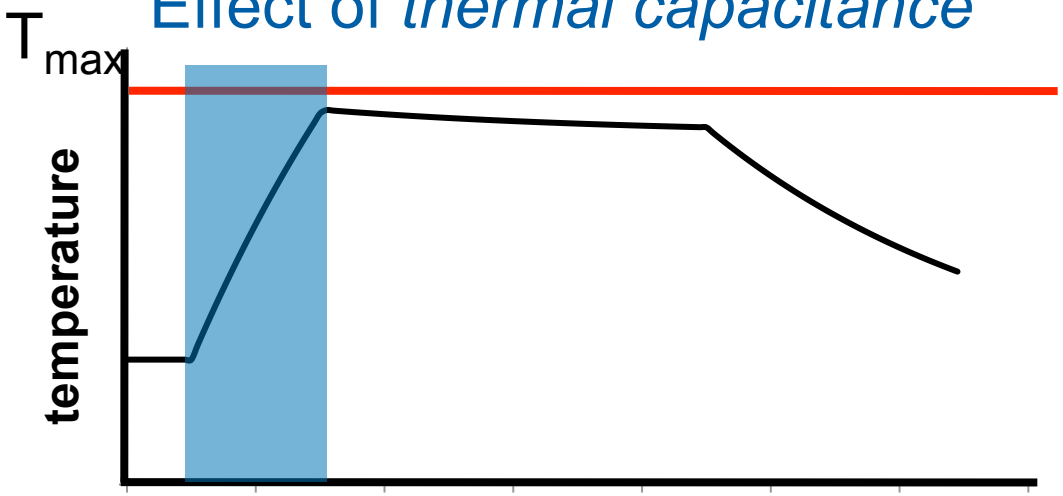
Parallel Computational Sprinting





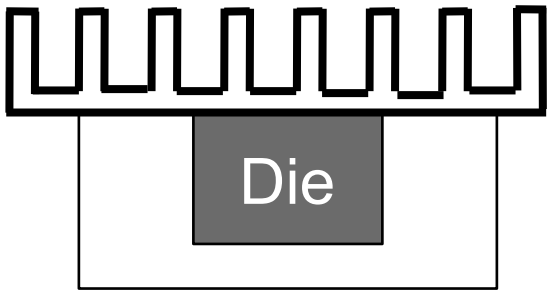
State of the art:
Turbo Boost 2.0
exceeds
sustainable power
with DVFS
(~25% for 25s)
Our goal: 10x, ~1s

Effect of *thermal capacitance*

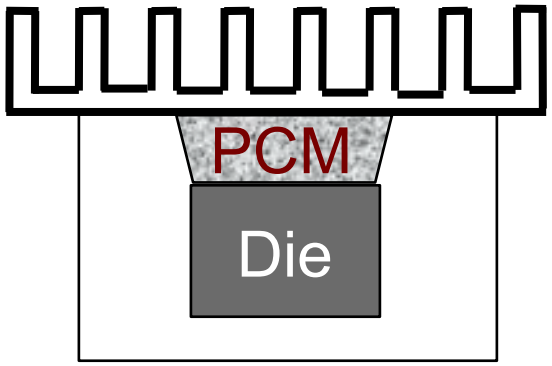


Extending Sprint Intensity & Duration: Role of Thermal Capacitance

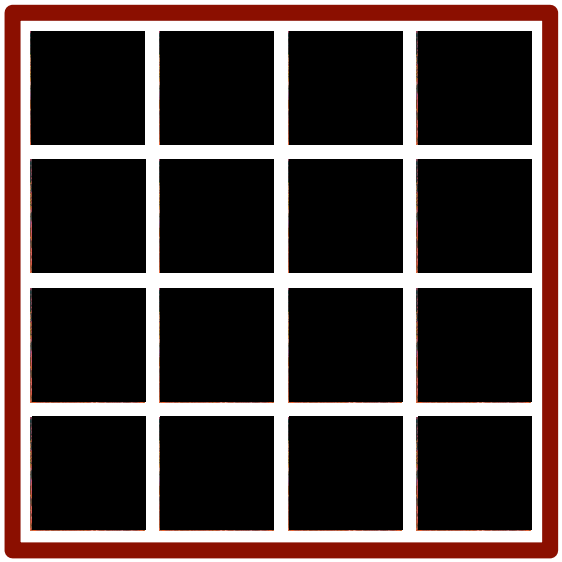
- Current systems designed for thermal ***conductivity***
  - Limited capacitance close to die

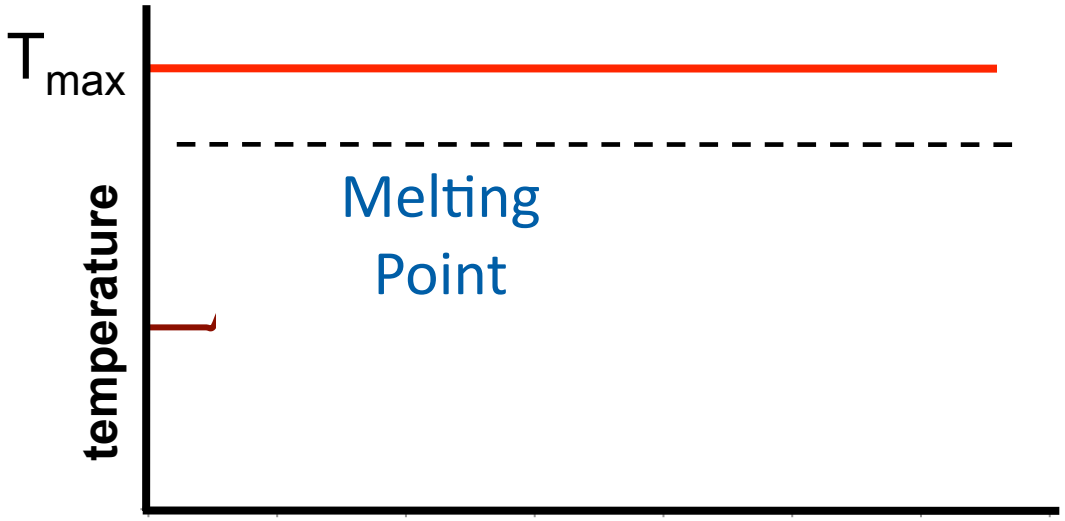


- To explicitly design for sprinting, add thermal ***capacitance*** near die
  - Exploit latent heat from phase change material (PCM)

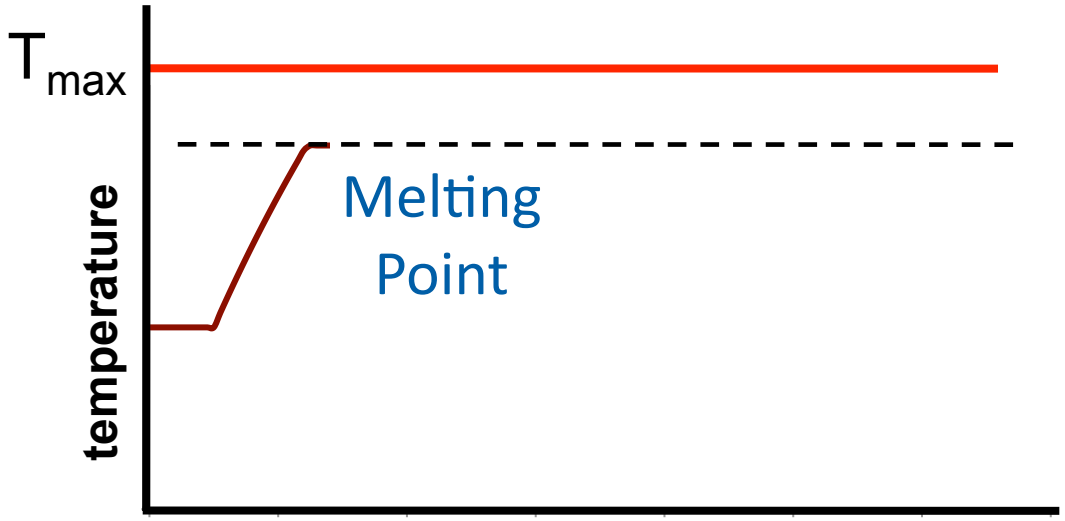


Augmented Sprinting with PCM

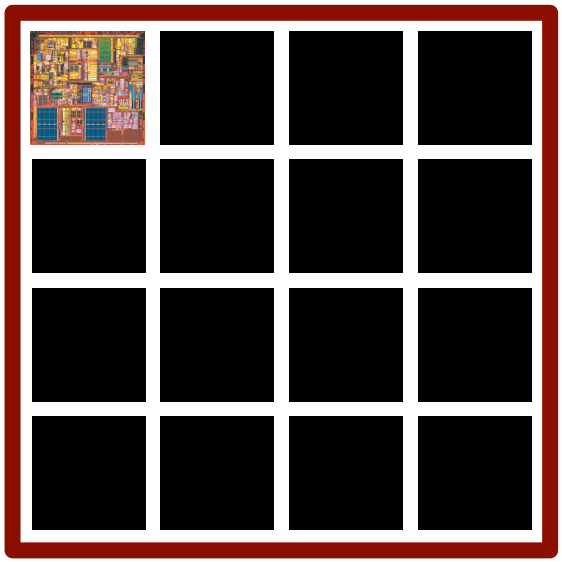


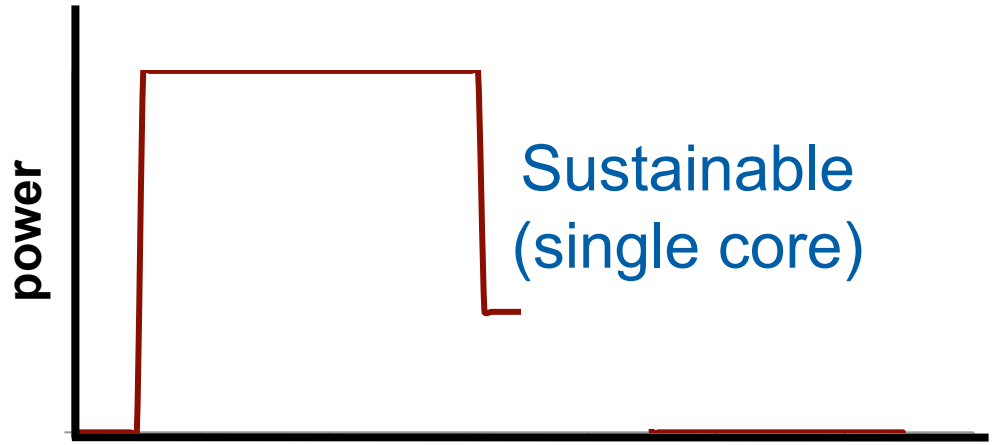


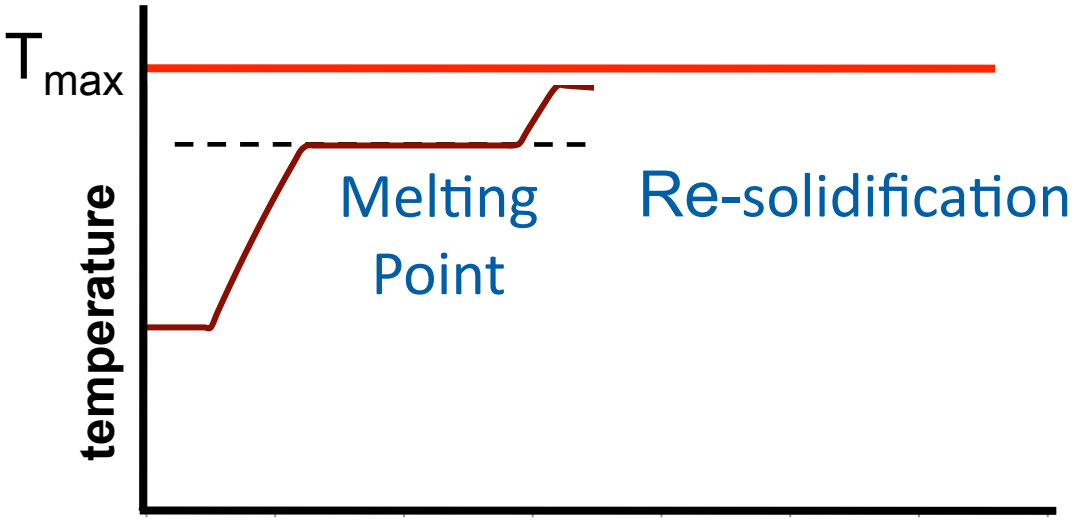
Augmented Sprinting with PCM



Augmented Sprinting with PCM





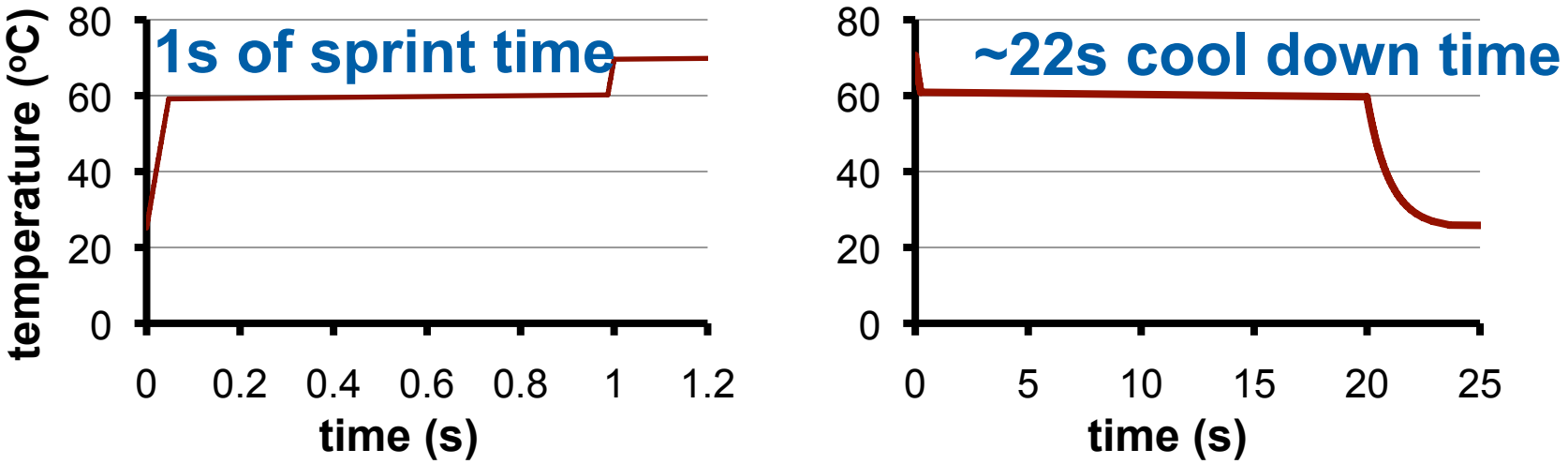


Outline

- Motivation
- Computational Sprinting
- **Feasibility Study**
  - **Thermal**
  - **Electrical**
  - **Hardware/software**
- Performance Evaluation
- Conclusion

Thermal Challenges

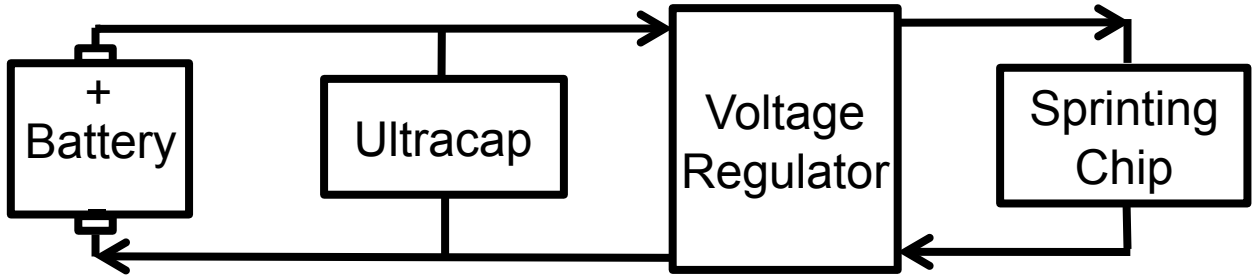
- Goal: 1s of 16x sprinting (16 1W cores)
- How much thermal capacitance does sprinting need?
  - 16W for 1s = 16J of heat, for PCM with latent heat 100J/g
  - 150mg, which is 2mm thick on 64mm^2 die
- Study based on thermal model of mobile phone



Heat flux and transient similar to desktop chips

Electrical Challenge #1: Peak Current Demands

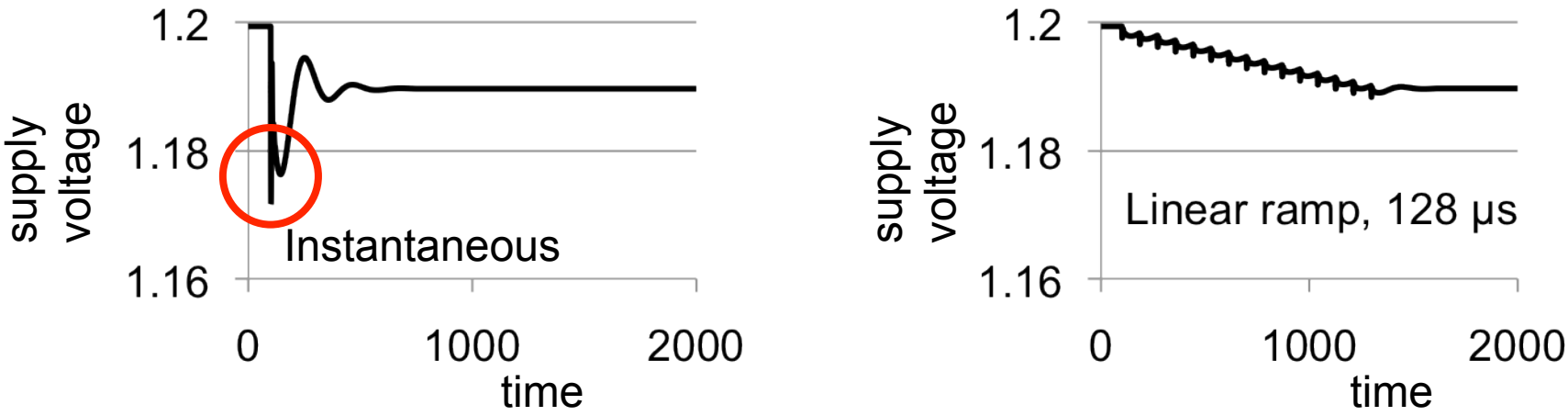
- 16x sprinting exceeds limits of today's phone batteries
  - Requires 16x peak current over baseline
- In contrast, ultracapacitors have high peak currents
  - Promising for sprinting (25F, 6.5g, 182J)
  - But today, lower energy density than batteries



- Leverage recent research on battery-ultracap hybrids
  [Mirhoseini+ '11, Palma+ '03, Pedram+ '10]
  - Active research at all levels (batteries, ultracaps, hybrids)
- Cost of extra power/ground pins

Electrical Challenge #2: On-chip Voltage Stability

- 16x current spike can cause supply voltage instability
  - Potential timing errors and state loss
- Study of core activation induced instability
  - SPICE model of board, package, chip
  - Abrupt activation violates; gradual activation ok



Core activation latency << sprint duration

Hardware/Software Challenges

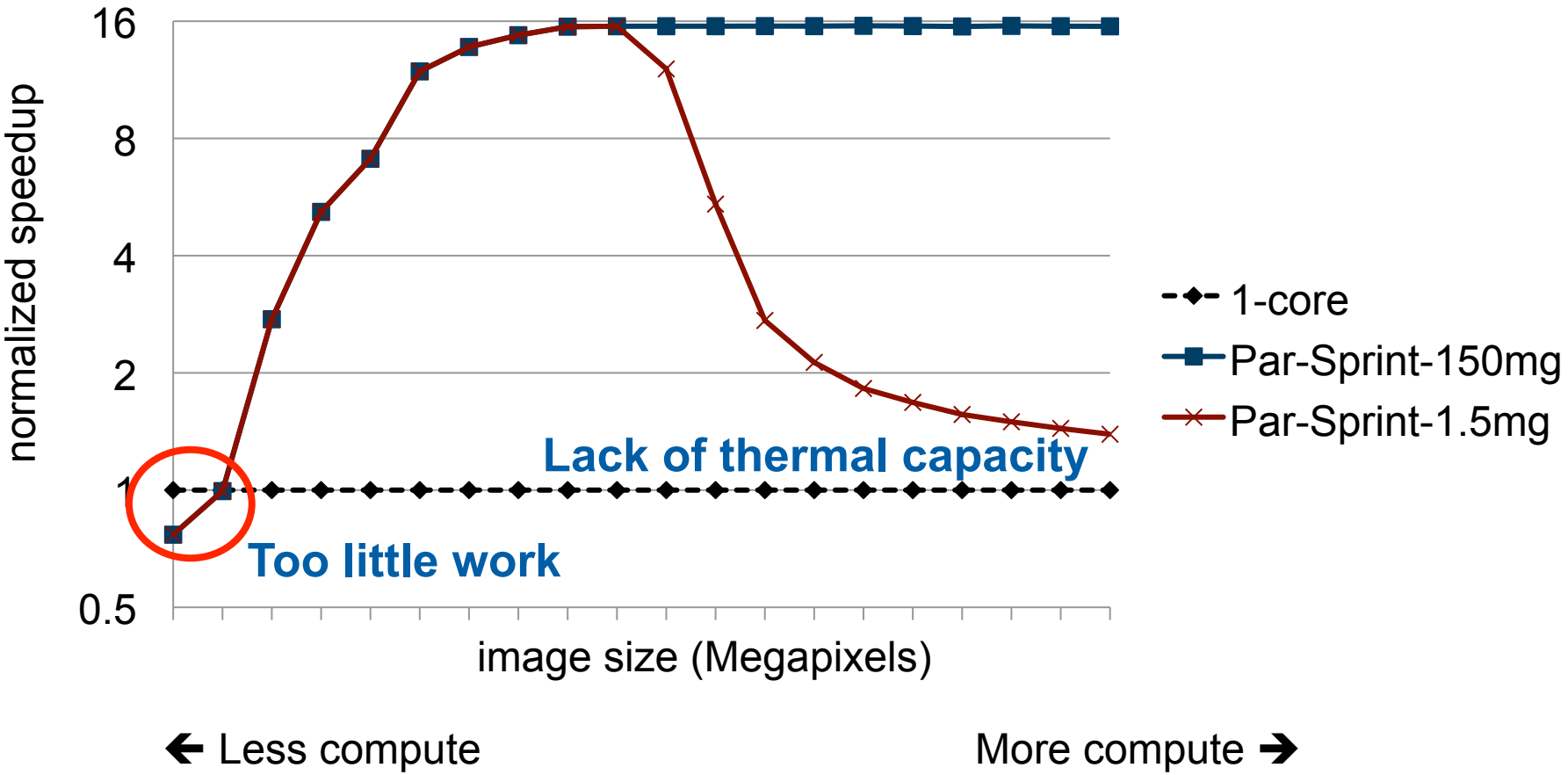
- Activation
  - Sprinting activated when parallel work available
- Deactivation
  - Hardware detects impending overheating
    - Monitor thermal budget
    - Energy from activity count + thermal model of system
  - If thermal budget nearing, ask runtime to migrate
  - If software is unable to respond
    - Drastically cut frequency to sustainable
    - Throttle by factor of "number of cores"

Performance Evaluation

Methodology

- In-order x86 many-core simulator
  - 16 cores
  - 32K, 8-way L1, 4MB 16-way shared LLC, directory cache coherence, 60ns memory latency, dual-channel 4GB/s memory interface
- Energy estimates from McPAT (1GHz, 1W, LOP)
  - Used to drive thermal model
- Workloads:
  - Vision kernels [SD-VBS], feature extraction app. [MEVBench]

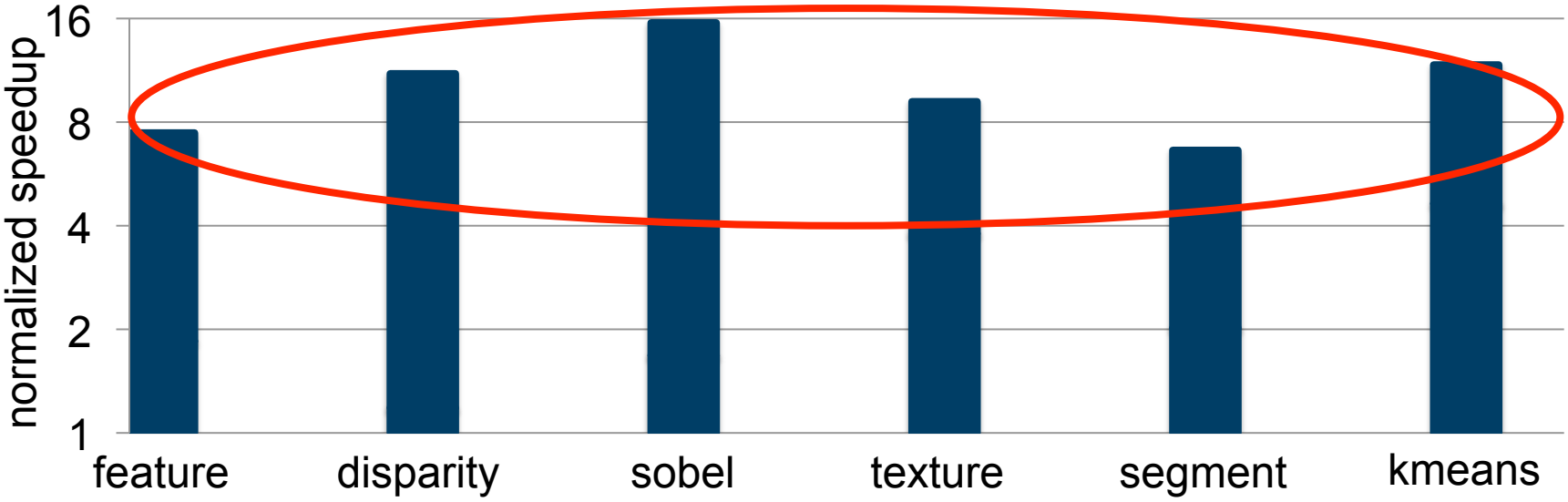
Sprinting Responsiveness



← Less compute

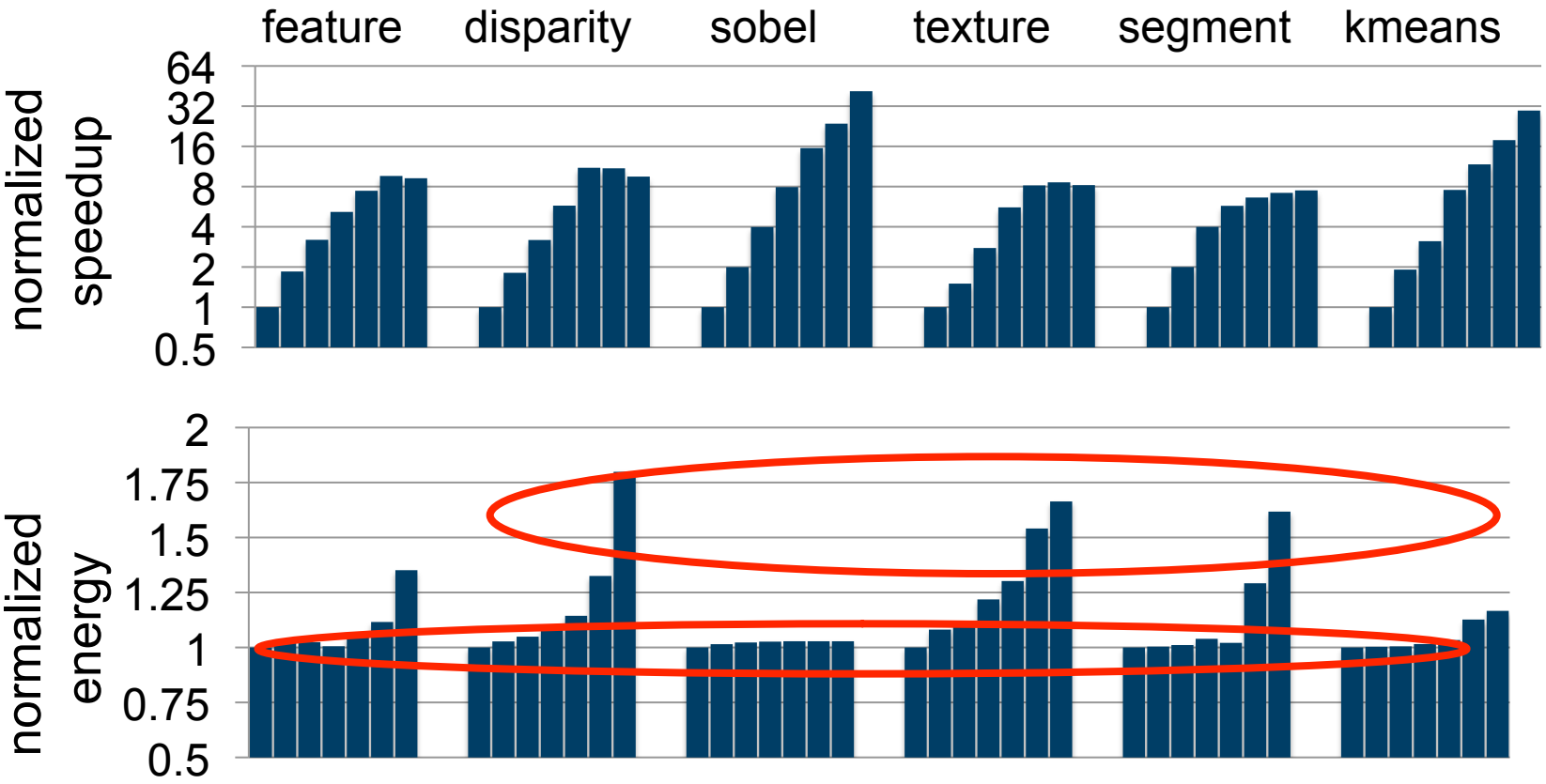
More compute →

Responsiveness Evaluation



Average 10.2x improvement in responsiveness

Parallelism & Energy



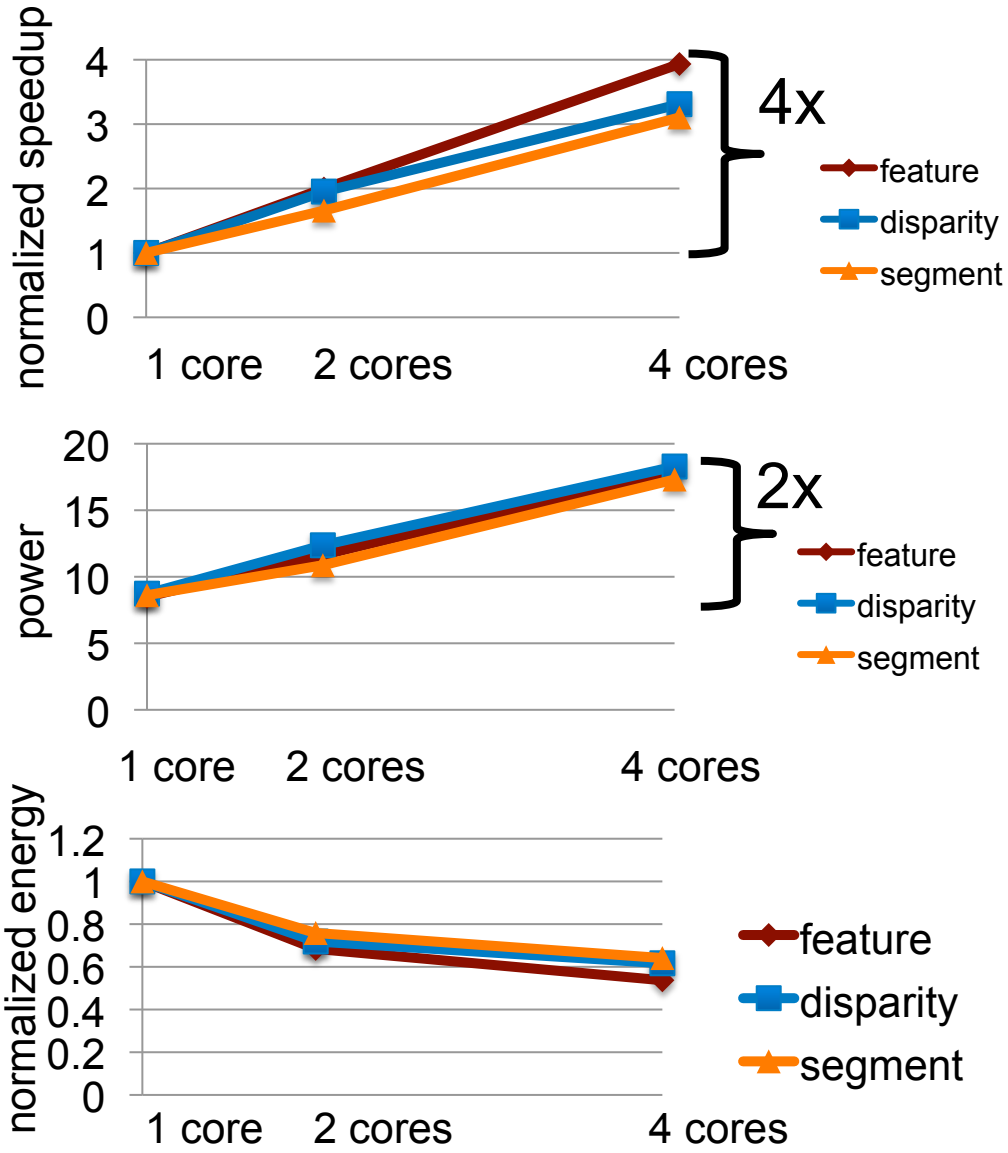
- No dynamic energy penalty when speedup linear
- Overall, 12% average dynamic energy increase

Moving Beyond Simulation:
Can we make a real system sprint?

Characterize real system energy/performance

How might sprinting behave on real system?

Multiple Cores: Energy and Performance



Power increases with core count

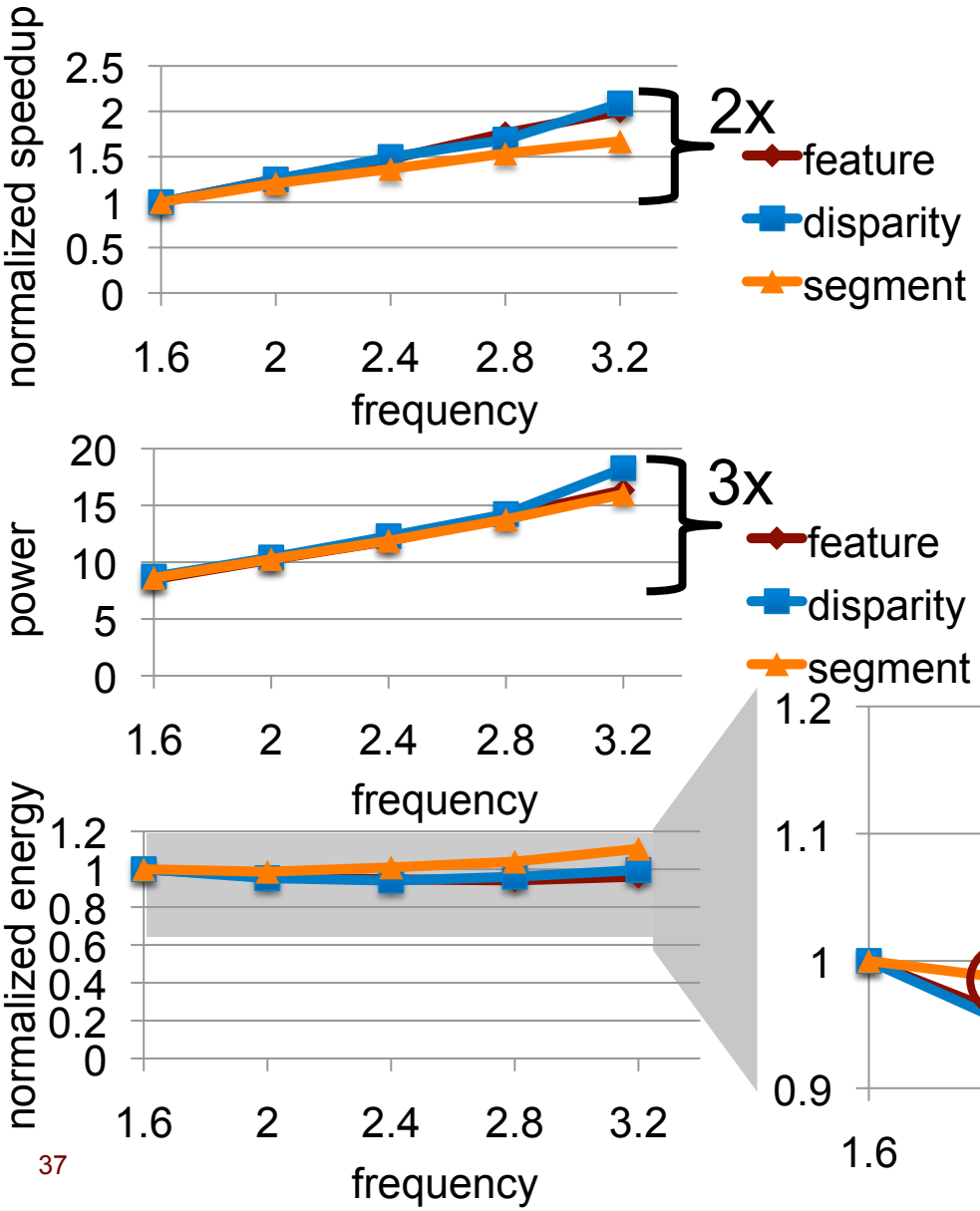
But < 2x for 4 cores

Why? background power

Energy consumption improves due to early completion

Race to halt

DVFS: Energy and Performance

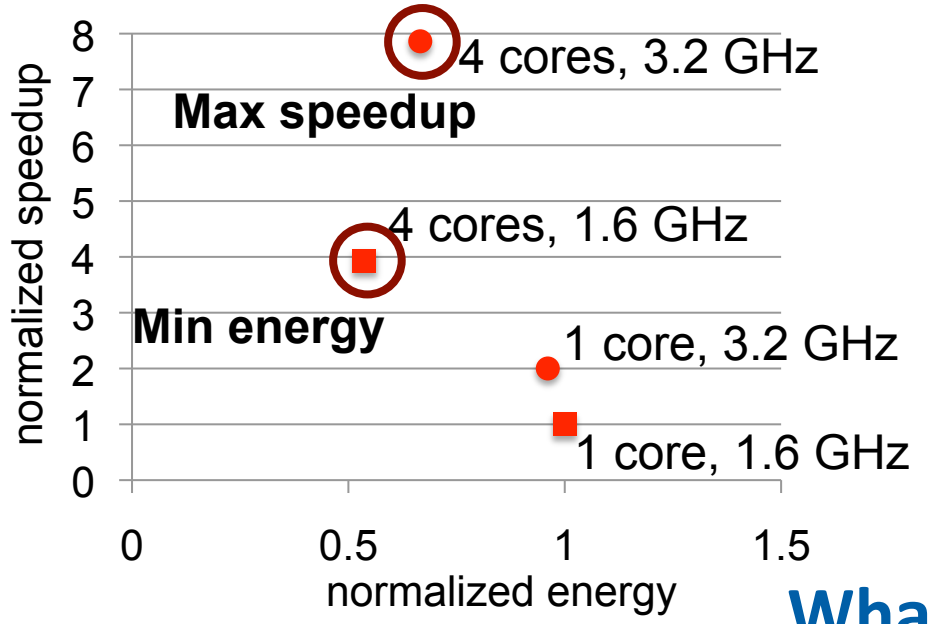


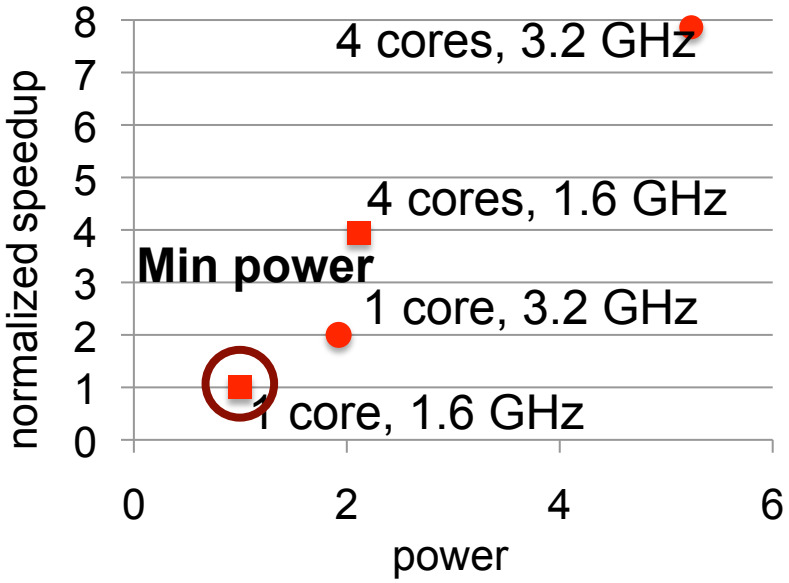
Power increases by ~3x for 2x speedup

Frequency	Voltage
1.6 GHz	0.95V
3.2 GHz	1.25V

Min frequency is not always min energy

Energy/Performance/Power Tradeoffs





What if this is the maximum sustainable power?

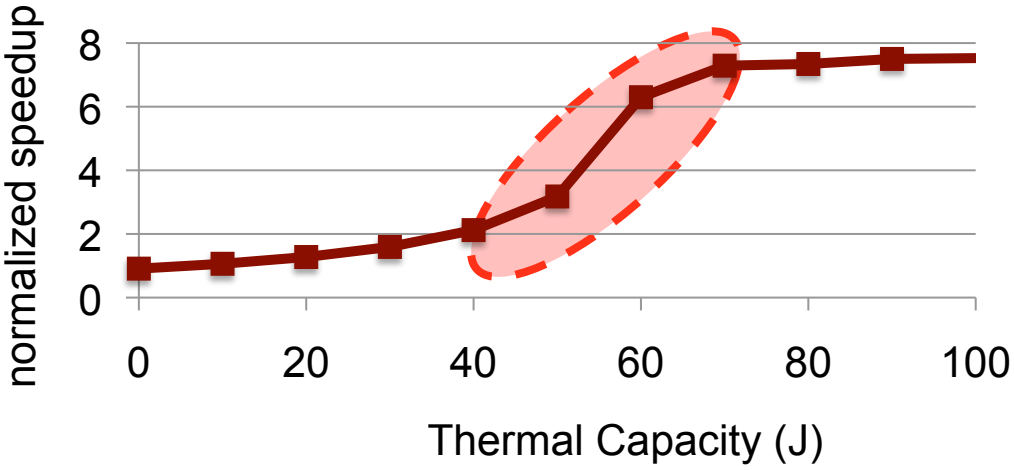
Sprinting enables higher responsiveness and greater energy efficiency

Emulating Sprinting with Limited Energy Budget

- Emulate effect of being thermally constrained
  - Not currently physically limiting cooling constraints
  - Estimate thermal capacity for sprinting based on energy

- Sprint operation:
  - Execute with all cores at maximum frequency
  - Monitor energy consumption via MSR
  - Terminate sprinting when energy capacity is exceeded
    - Migrate threads to single core
    - Shutdown additional cores
    - Lower frequency to minimum

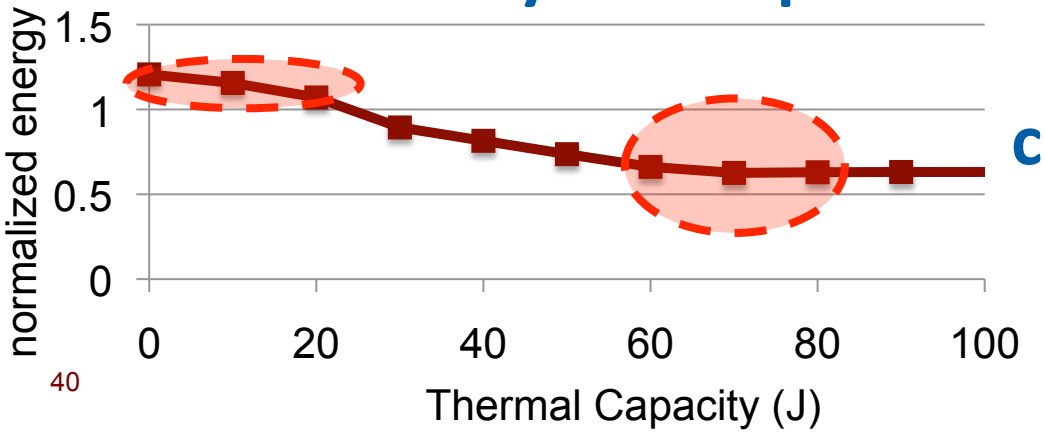
Effect of Thermal Capacity



Speedup sensitive to amount of computation within sprint

Longer sprints enable greater energy saving

Energy overhead from extremely short sprints



Caveat: mobile system background power characteristics likely differ

Work in progress: constrain cooling system

Conclusions

- **Computational Sprinting**
  - Targets responsiveness by far exceeding sustainable operation
  - Exploit phase change material as thermal buffer
- **Explored feasibility of sprinting**
  - Promising avenues for managing electrical, thermal and architectural barriers to sprinting
- **Order-of-magnitude improvements in responsiveness**
  - Within the constraints of a 1W device
- **Opportunity to rethink the stack around responsiveness**