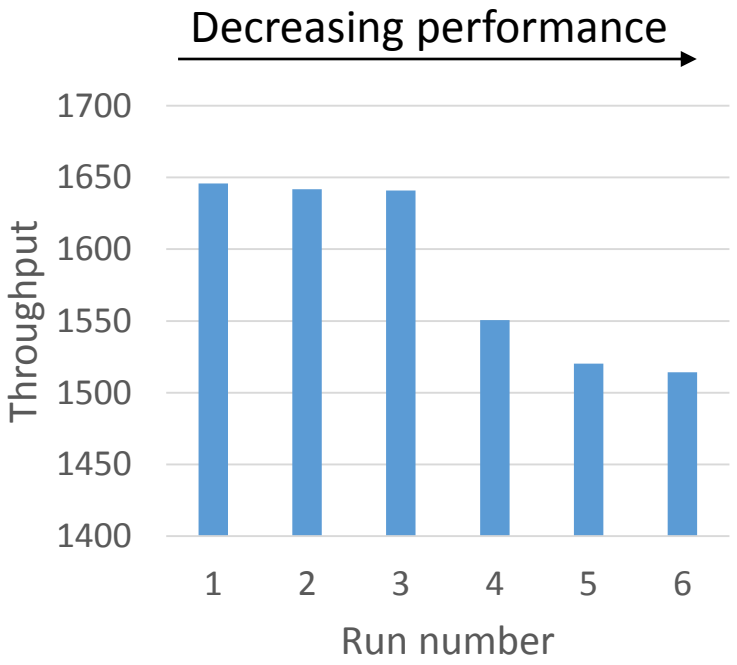
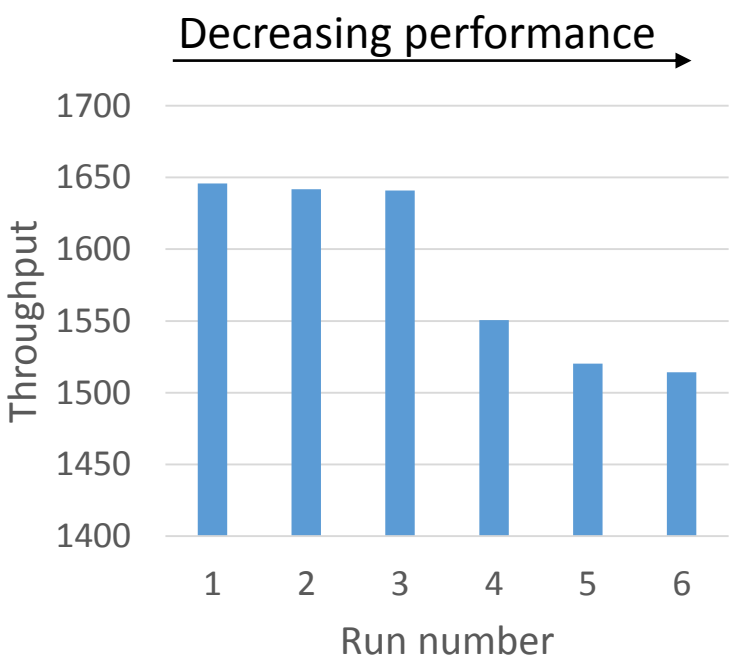
Pitfalls of Benchmarking Thermally Adaptive Systems

Laurel Emurian[*], Arun Raghavan[§], Lei Shao[#], Jeffrey M. Rosen[+]
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[#]
Oracle Labs[§]


Decreasing performance
1700
1650
1600
1550
1500
1450
1400
Throughput
1
2
3
4
5
6
Run number


12
3
6
9


Decreasing performance
1700
1650
1600
1550
1500
1450
1400
Throughput
1
2
3
4
5
6
Run number

Thermally adaptive system

- **Vary performance over time to meet thermal constraints**
  - Intel Turbo Boost 2.0, AMD Turbo Core
  - Computational Sprinting
- **Thermally adaptive systems cause benchmarking pitfalls:**
  - Failing to consider recent activity
  - Extrapolating steady state performance from a short run
  - Comparing programs of different lengths
- We explore yet another way of: ***Producing wrong data without doing anything obviously wrong!*** **[Mytkowicz, ASPLOS '09]**

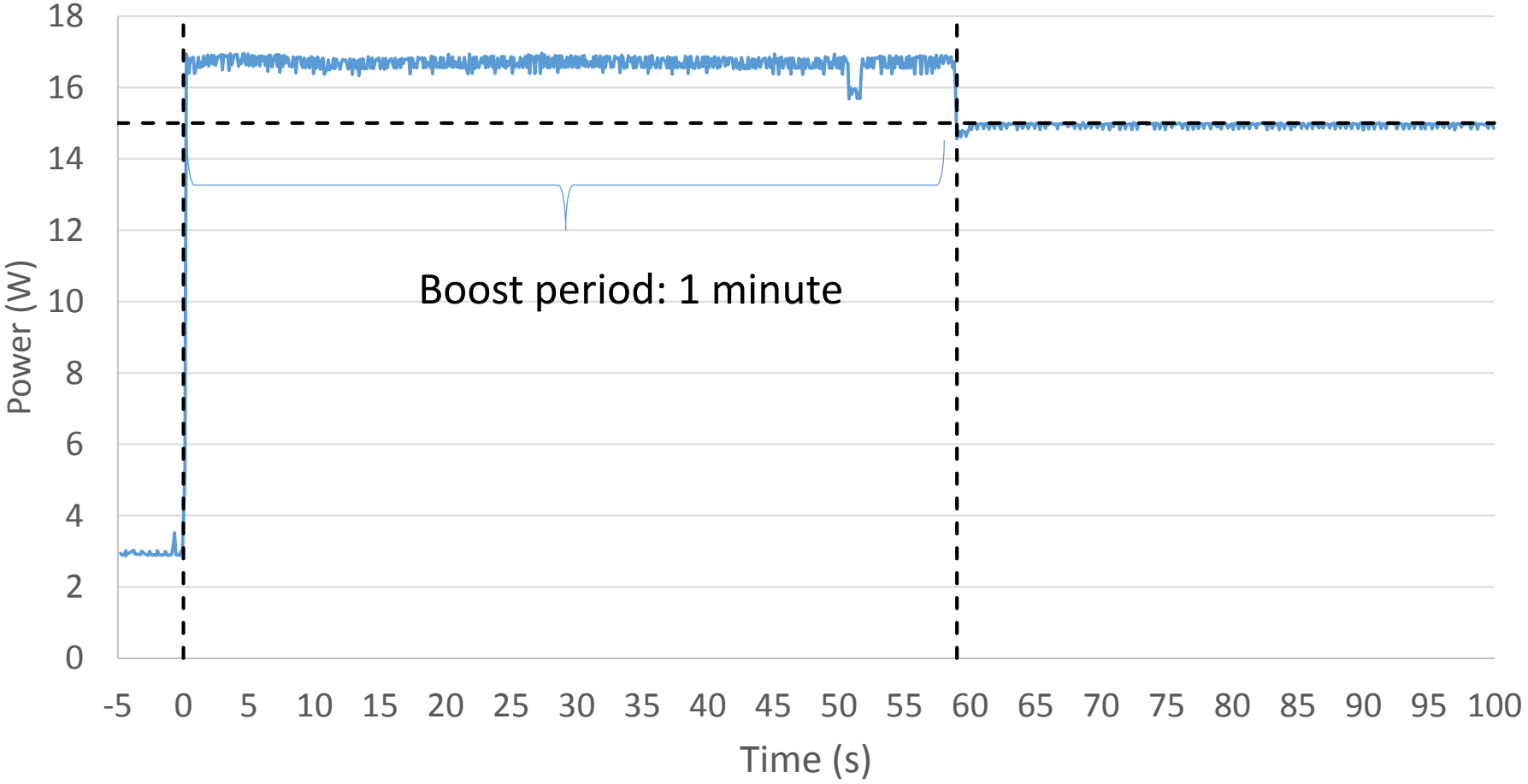
Overview

- How do thermally adaptive systems affect performance measurement?
  - Intel's Turbo Boost
- Benchmarking Perils
- Conclusion

Intel Turbo Boost

- First generation: Stays within sustainable thermal power
  - Shifts thermal budget among components
  - E.g., if cores are idle, increase GPU frequency
- Second generation: Exceeds sustainable thermal power
  - Starts at a frequency with unsustainable power
  - Throttles down after an interval to prevent over heating
- Understand Turbo Boost to learn how to avoid benchmarking pitfalls

How does a boost behave?

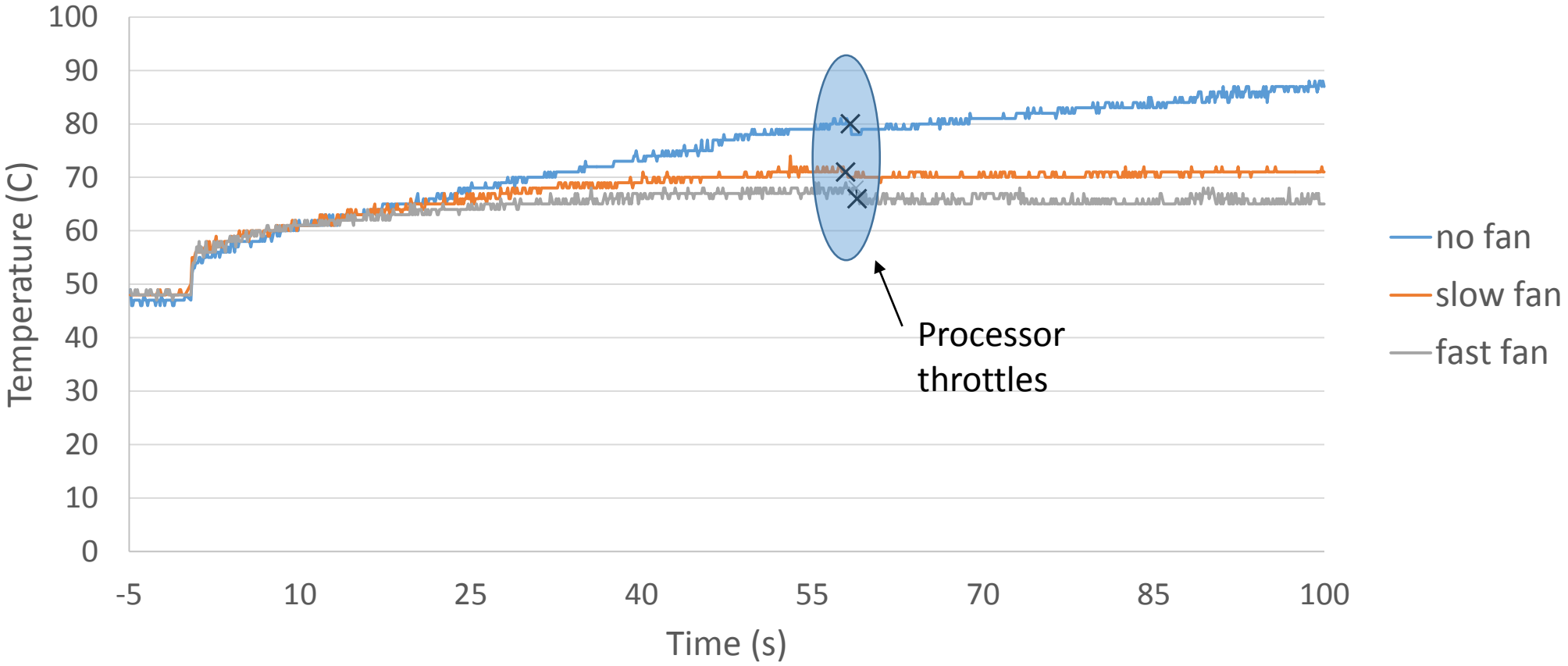


12.5% difference between boost and throttle frequency

Platform: Dell laptop, Intel i7-4500U

Is throttling based on temperature?

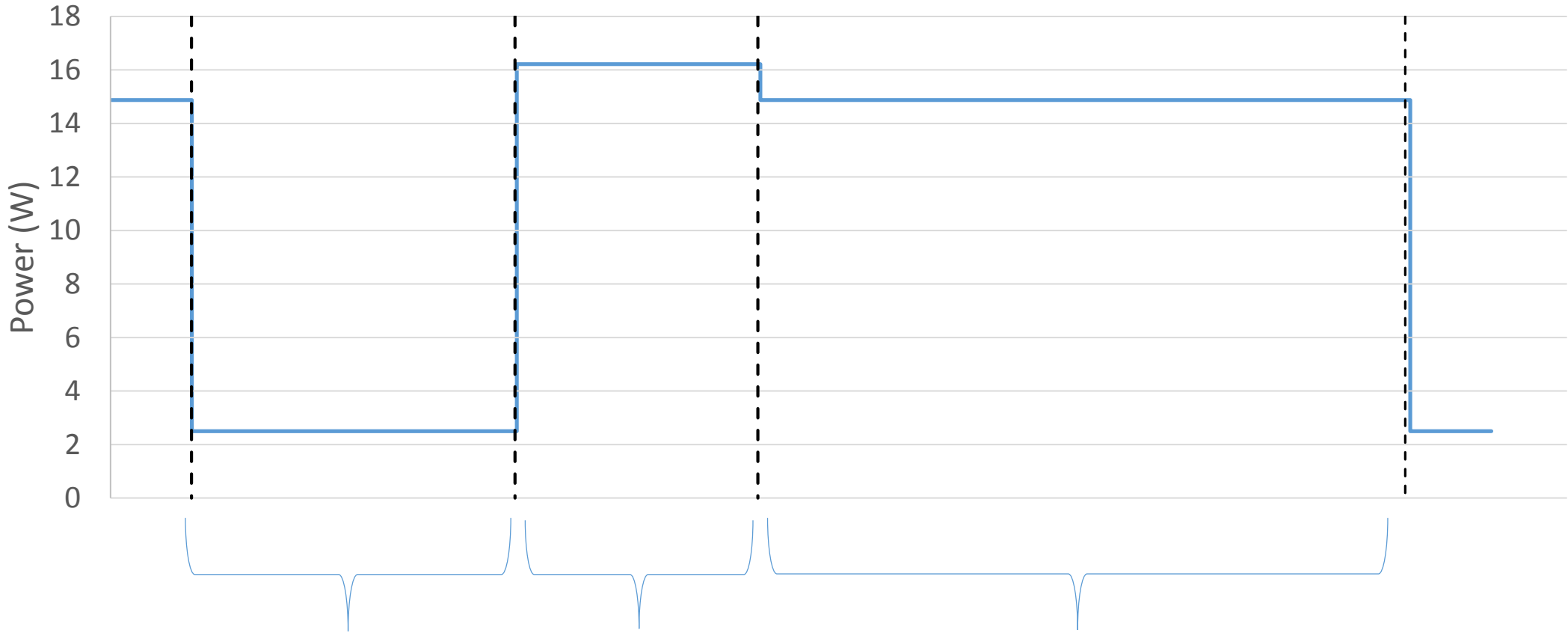
Experiment: Run with varied fan speeds



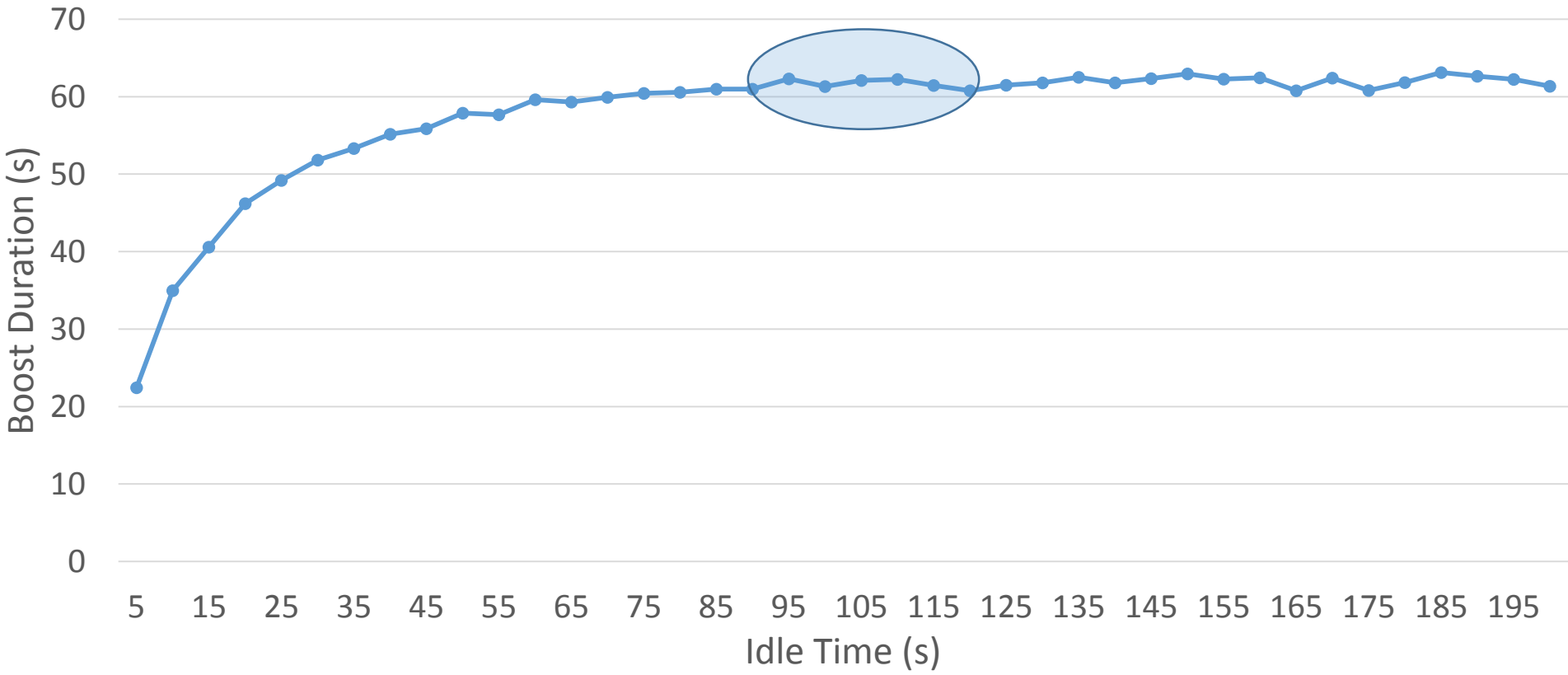
Runs with varied fan speeds throttle after same duration with different temperatures

How can we determine the control policy?

Experiment: Measure boost period after idling



What is the boost duration after varying idle?



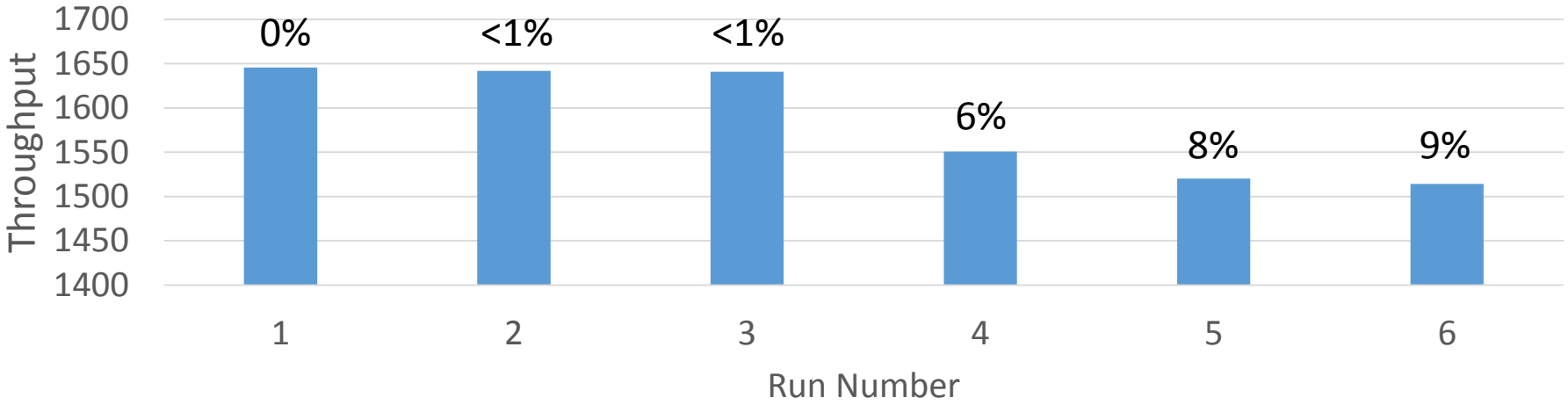
After approximately two minutes, no power history effects

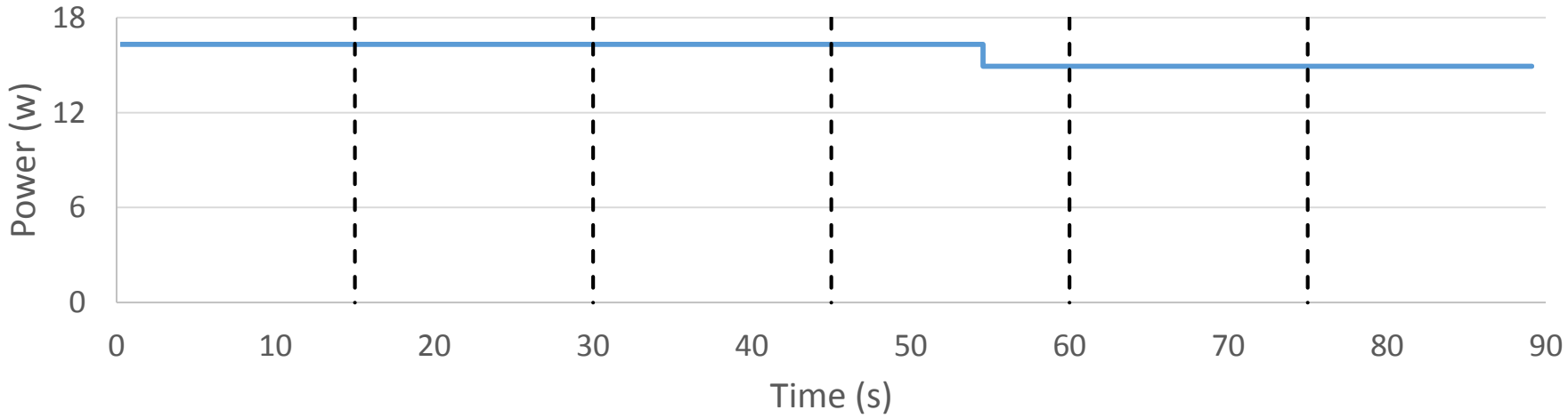
Overview

- Turbo Boost Analysis
- Benchmarking Perils
  - Why does each peril occur?
  - How can the perils be mitigated?
- Future Impact

Pitfall 1: Failing to consider recent activity

Scenario: Repeat a program multiple times back-to-back





This system: measurement error of up to 9%

Pitfall 1: Mitigation

- Disable Turbo Boost
  - Significantly decreases performance
  - May not be feasible (shared systems, no root privileges)
- Start workloads with a clean power history
  - Idle for the duration of the power window
- Fully saturate the power history so boost does not happen
  - Run for the duration of the power window before taking measurements

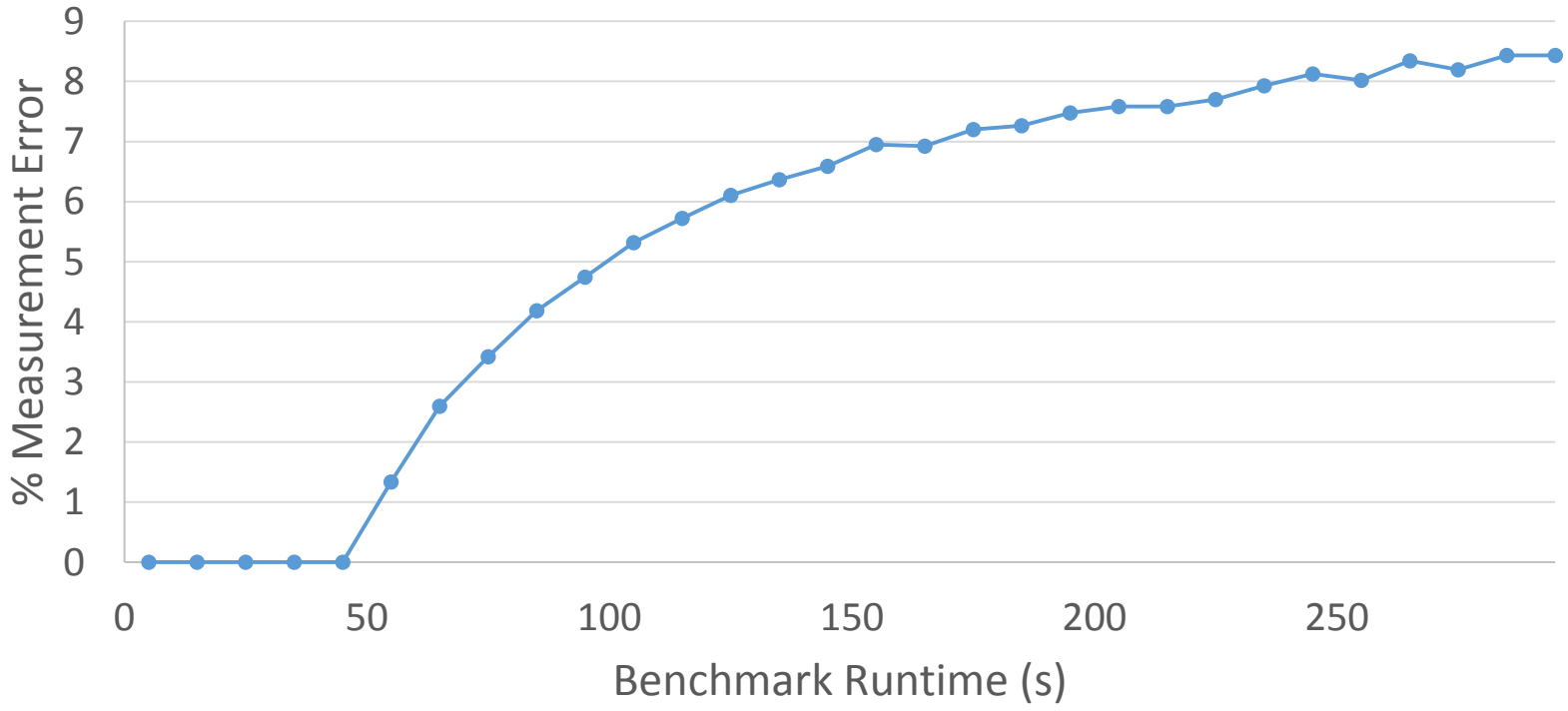
Pitfall 2: Extrapolating steady state performance from a short run

Scenario: Testing a program that will make 20 queries per second

- Deploy for 20 seconds -> throughput as expected
- Deploy for 5 minutes -> throughput lower than expected
  - Turbo Boost throttled performance after a minute

Pitfall 2: Extrapolating steady state performance from a short run

Experiment: Run the same benchmark for increasing runtimes



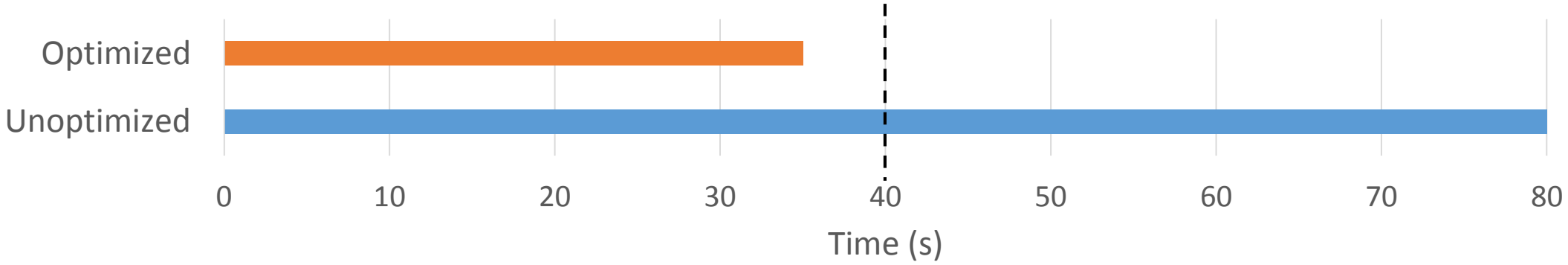
This system: measurement error of up to 8%

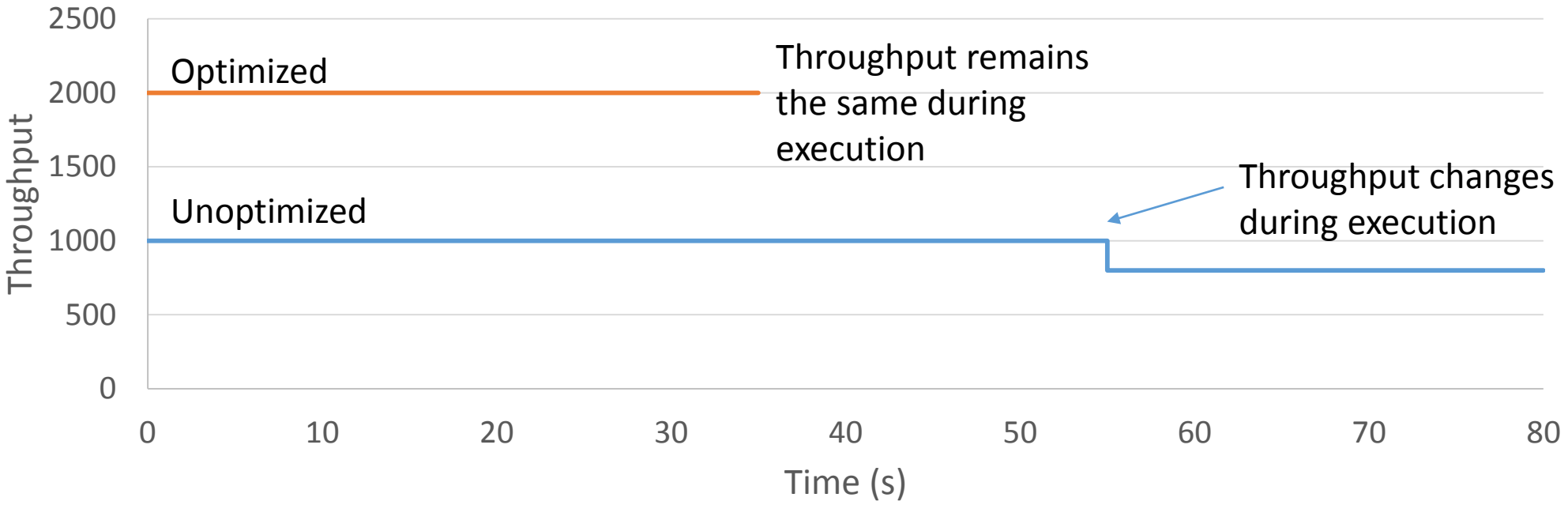
Pitfall 2: Mitigation

- Saturate the power history before running
- Don’t extrapolate past the length of the boost period
- Run for ‘long enough’

Pitfall 3: Comparing programs of different runtimes

Scenario: Compare throughput of an unoptimized and optimized program





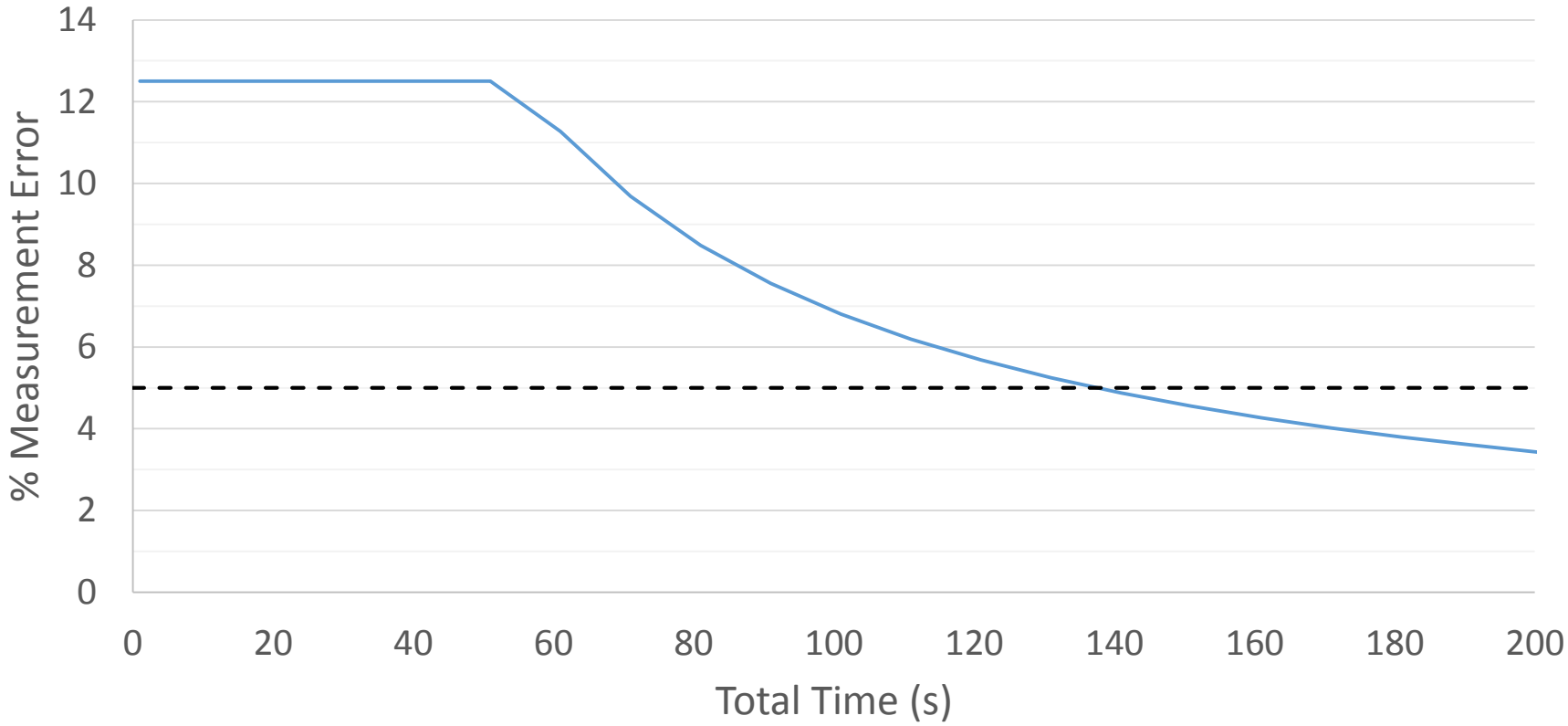
Optimized programs get an extra ‘boost’

Pitfall 3: Mitigation

- Ensure fixed frequency during runs
  - Don’t run for longer than the boost window
  - Pin cores to a fixed frequency
- Measure throughput over fixed duration
  - Not measure fixed work over variable duration
- Run a workload ‘long enough’

How long is ‘long enough’?

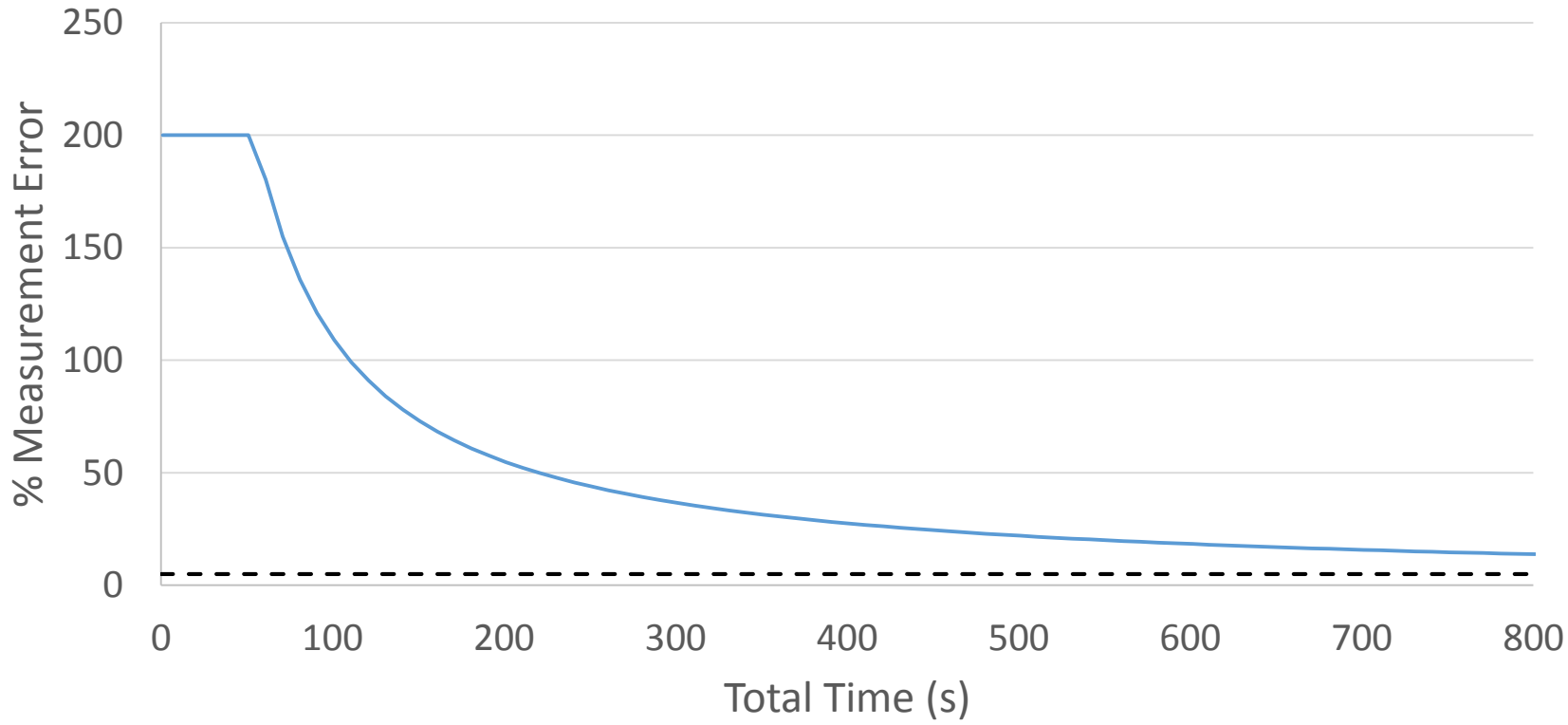
Model: When is measurement error less than 5%?



2 minutes for error below 5%

Looking forward: more intense boosts

Scenario: Boost is 3x throttled frequency



Significantly increases error – up to 200%

Run for 37 minutes for error of less than 5%

Conclusions

- Thermally adaptive systems have long start up transients
  - Much larger than e.g. cache warm up
- Benchmarking errors occur when:
  - Failing to consider prior system activity
  - Extrapolating workload performance from a short run
  - Comparing programs of different lengths
- Our system: Already shows significant errors of (9%) occur
- Boost intensities -- and hence error -- projected to increase over time

www.cis.upenn.edu/acg/sprinting/