Performance Issues

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Announcements

- Assignment 3 has been posted
  - Due on: April 10 at 11:59 pm
- If you were a scribe for April 2, please pick a new slot
  - 6 open slots

Performance metrics

- Time to solution
- Time per step (iteration)
- Science progress (figure of merit per unit time)
- Floating point operations per second (flop/s)
- When comparing multiple data points:
  - Speedup, efficiency

What is the best performance we can get?

- Peak flop/s
- Peak memory bandwidth
- Peak network bandwidth
- Why do we not achieve peak performance?

What is happening in a program

- Integer operations
- Floating point operations
- Conditional instructions (branches)
- Loads/stores
- Data movement across the network (messages + I/O)

Performance issues

- Serial code performance issues*
  - Inefficient memory access: data movement in the memory hierarchy
  - Inefficient floating point operations
- Load imbalance
  - Some processes doing more work than most
- Communication issues / parallel overhead*
  - Spending increasing proportion of time on communication
- Algorithmic overhead / replicated work
  - More computation when running in parallel (e.g. prefix sum)

Performance issues

- Speculative loss
  - Perform extra computation speculatively but not use all of it for the result
- Critical paths*
  - Dependencies between computations spread across processes / threads
- Insufficient parallelism
- Bottlenecks*
  - Serial bottlenecks: one process doing some computation and holding everyone up

Serial code performance issues

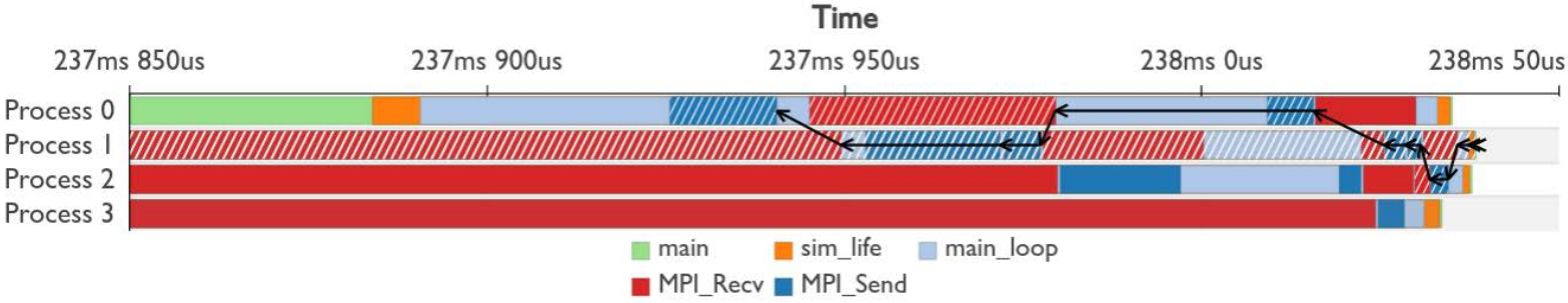
- Identify issues using performance tools
- Solutions:
  - Minimize data movement
  - Maximize data reuse
  - Optimize floating point calculations

Communication performance issues

- Overhead and grainsize (Lots of tiny messages or a fewer, larger messages)
- No overlap between communication and computation
- Increasing amounts of communication as we run with more processes/threads
- Frequent global synchronization

Critical paths

- A long chain of operations with consecutive dependencies across processes
- We want to identify and avoid having long critical paths
  - Eliminate completely if possible, or shorten it
  - Reduce time spent in a path by removing work on the critical path

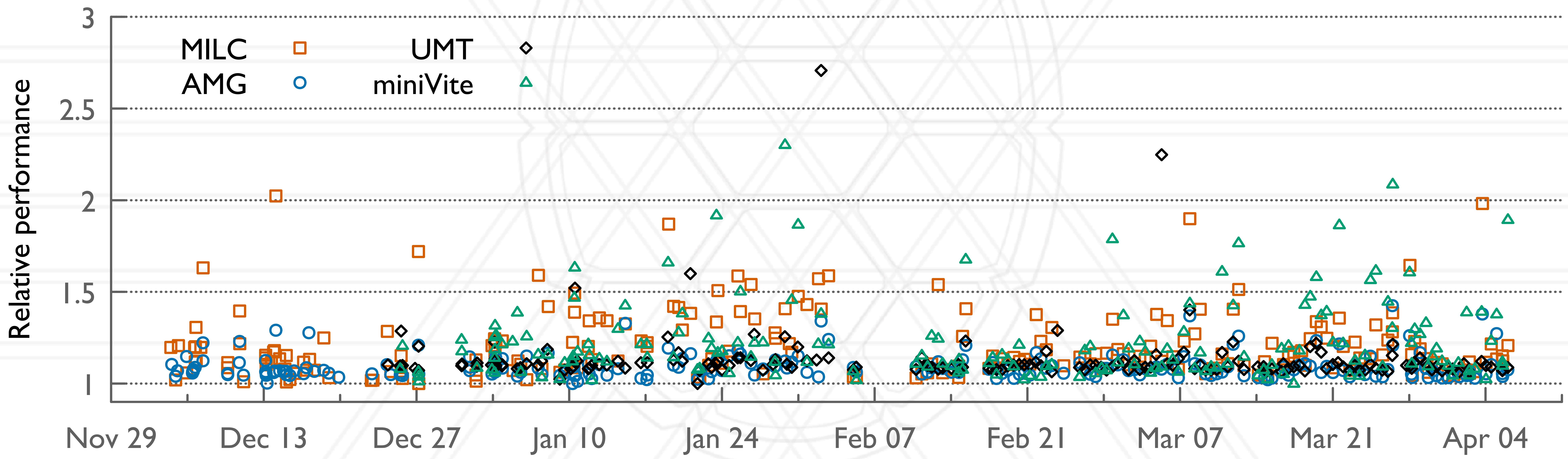


Serial bottlenecks

- Detect serial bottlenecks: things getting "serialized"
  - One process busy while all others wait
- Examples of serial bottlenecks:
  - Reduce to one process and then broadcast
  - One process responsible for input/output
  - One process responsible for assigning work to others
- Solutions:
  - Parallelize as much as possible, use hierarchical schemes

Performance variability is a real concern

Performance of control jobs running the same executable and input varies as they are run from day-to-day on 128 nodes of Cori in 2018-2019



Bhatele et al. The case of performance variability on dragonfly-based systems, IPDPS 2020

Leads to several problems …

- Individual jobs run slower:
  - More time to complete science simulations
  - Increased wait time in job queues
  - Inefficient use of machine time allocation
- Overall lower system throughput
- Increased energy usage/costs

Affects software development cycle

- Debugging performance issues
- Quantifying the effect of various software changes on performance
  - Code changes
  - System software changes
- Estimating time for a batch job or simulation

Sources of performance variability

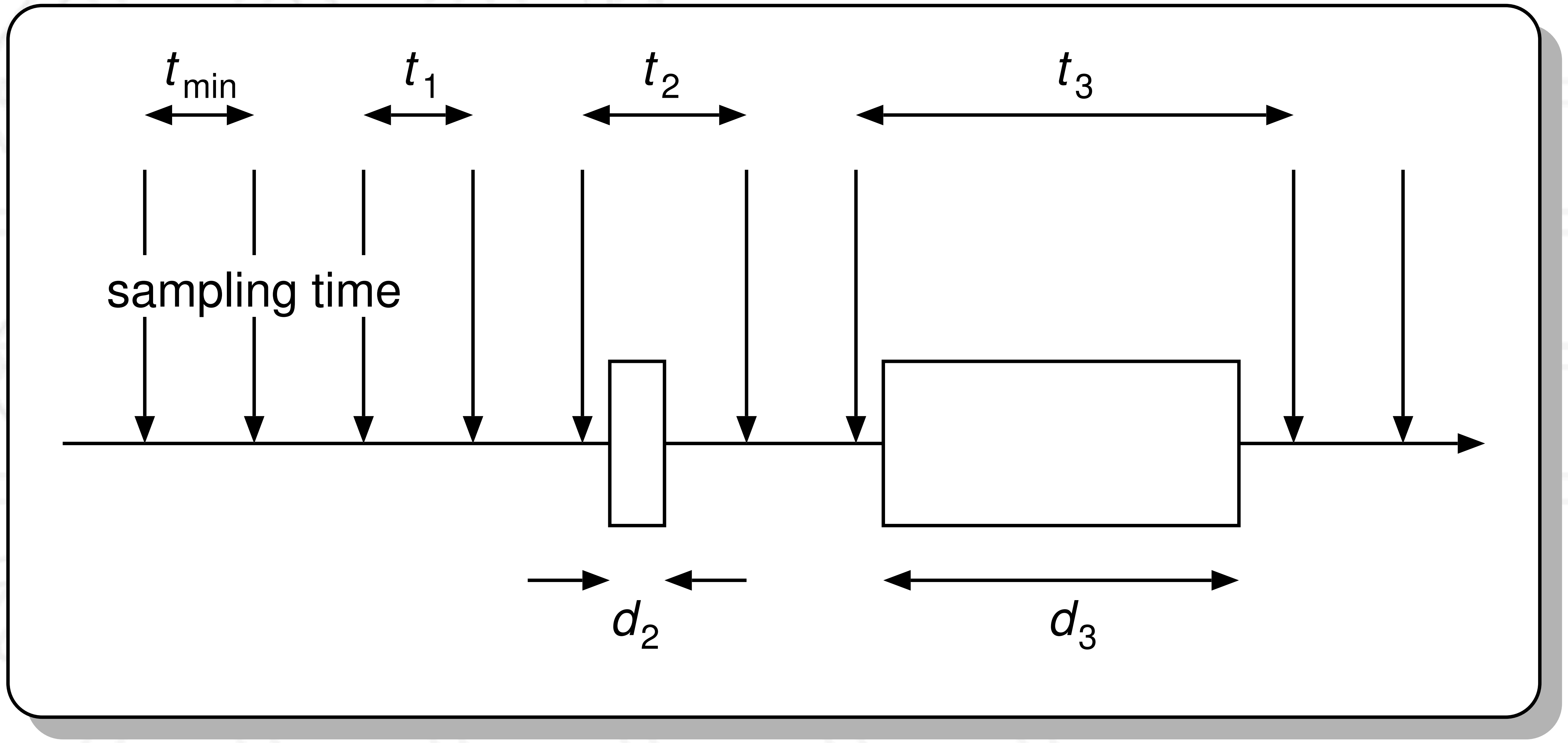
- Operating system (OS) noise/jitter
- Contention for shared resources
  - Network
  - Filesystem

Operating System

- Node on an HPC cluster may have:
  - A "full" linux kernel, or
  - A light-weight kernel
- This determines what services/daemons run
- Impacts performance predictability

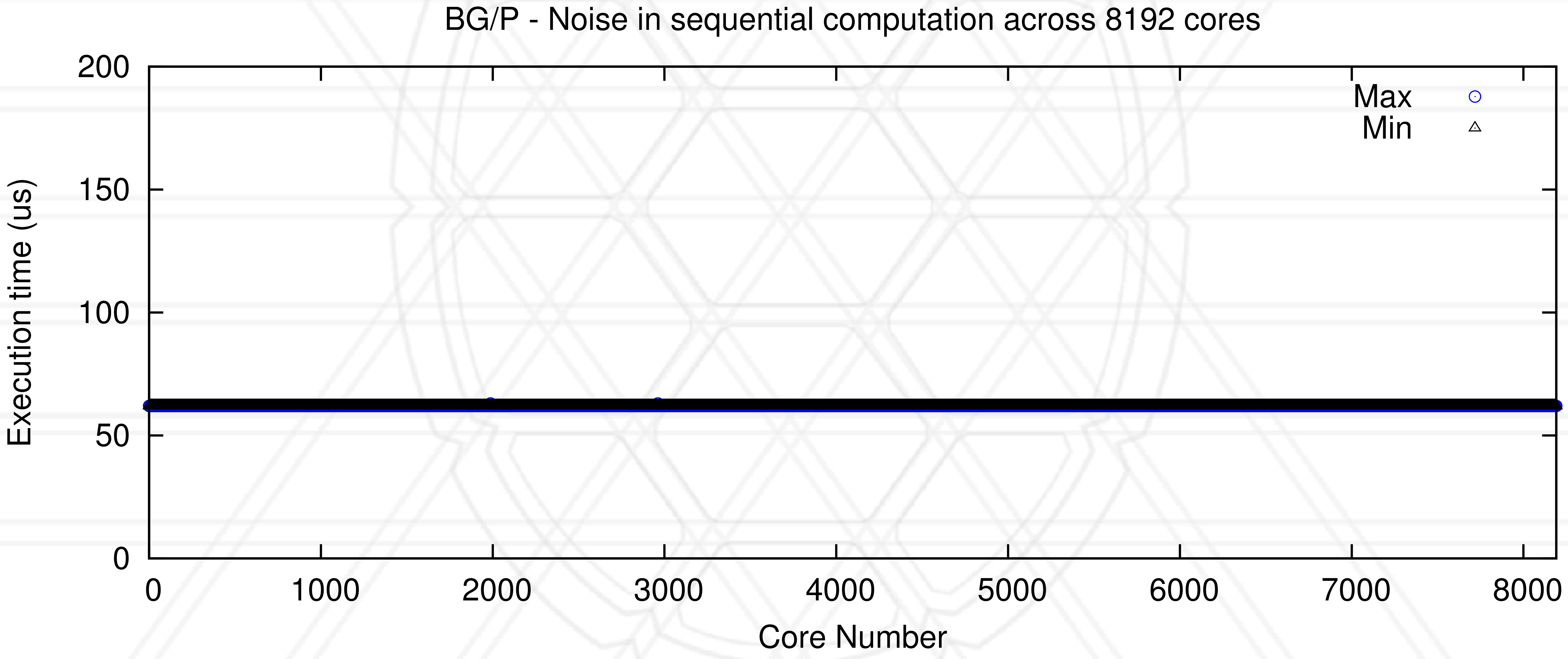
Operating System (OS) Noise

- Also called "jitter"
- Impacts computation due to interrupts by OS



Measuring OS Noise

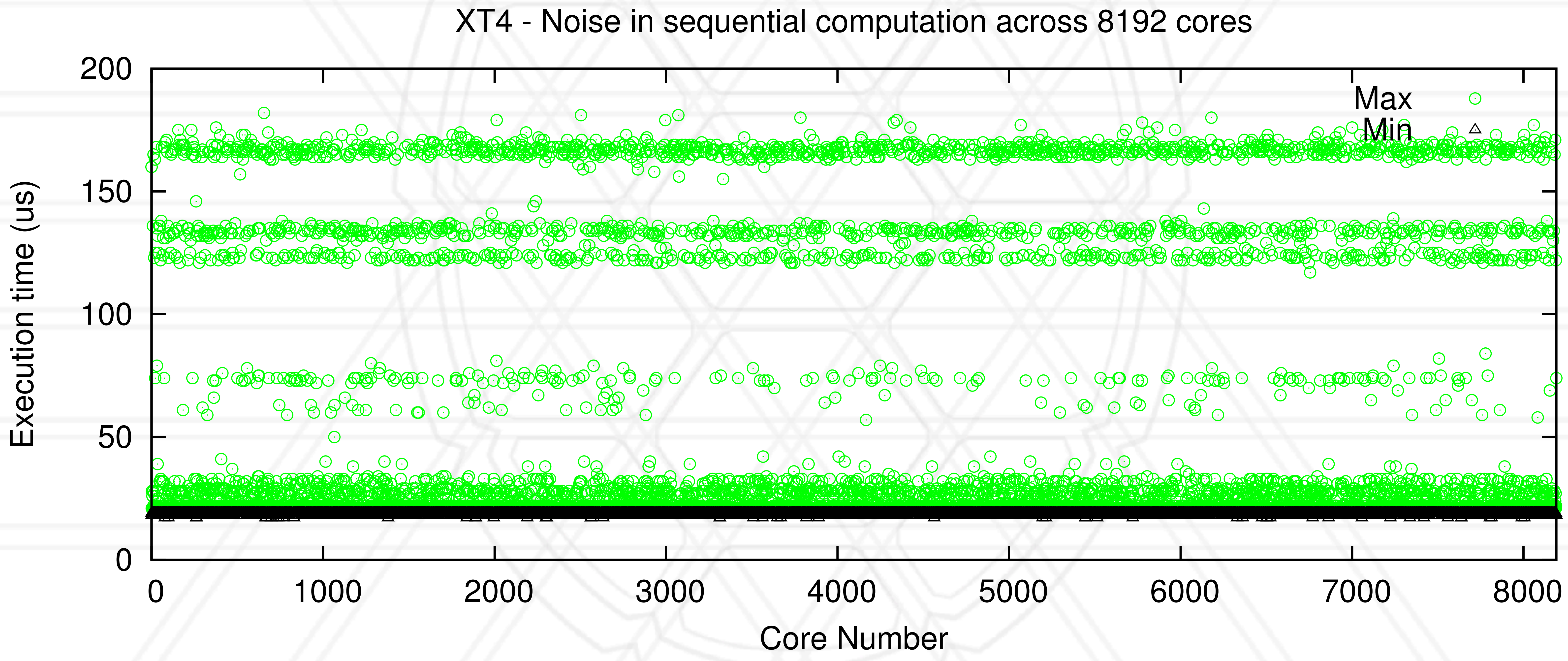
- Fixed Work Quanta (FTW) and Fixed Time Quanta (FTQ)



Benchmarks: https://asc.llnl.gov/sites/asc/files/2020-06/FTQFTW_Summary_v1.1.pdf

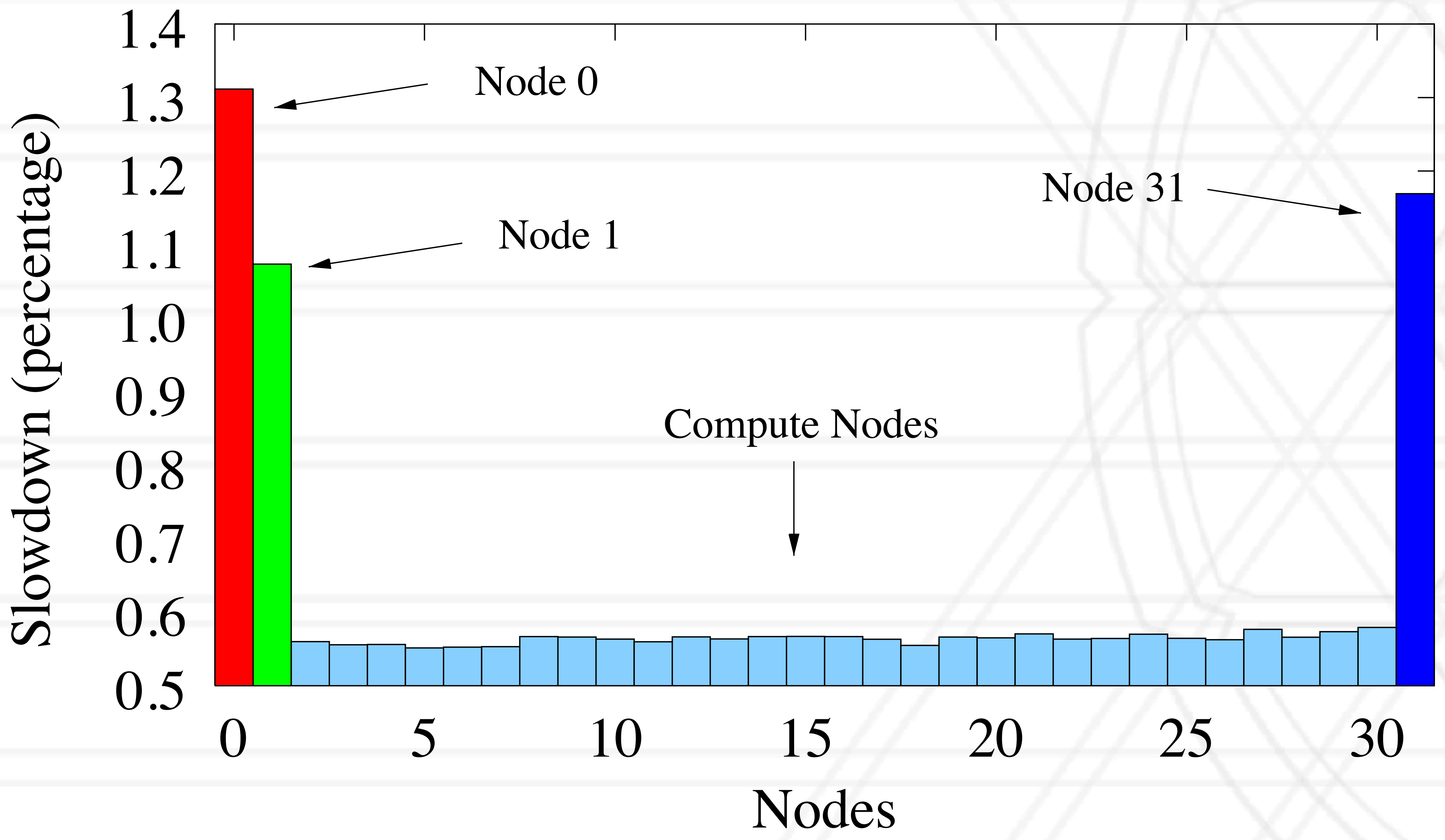
Measuring OS Noise

- Fixed Work Quanta (FTW) and Fixed Time Quanta (FTQ)



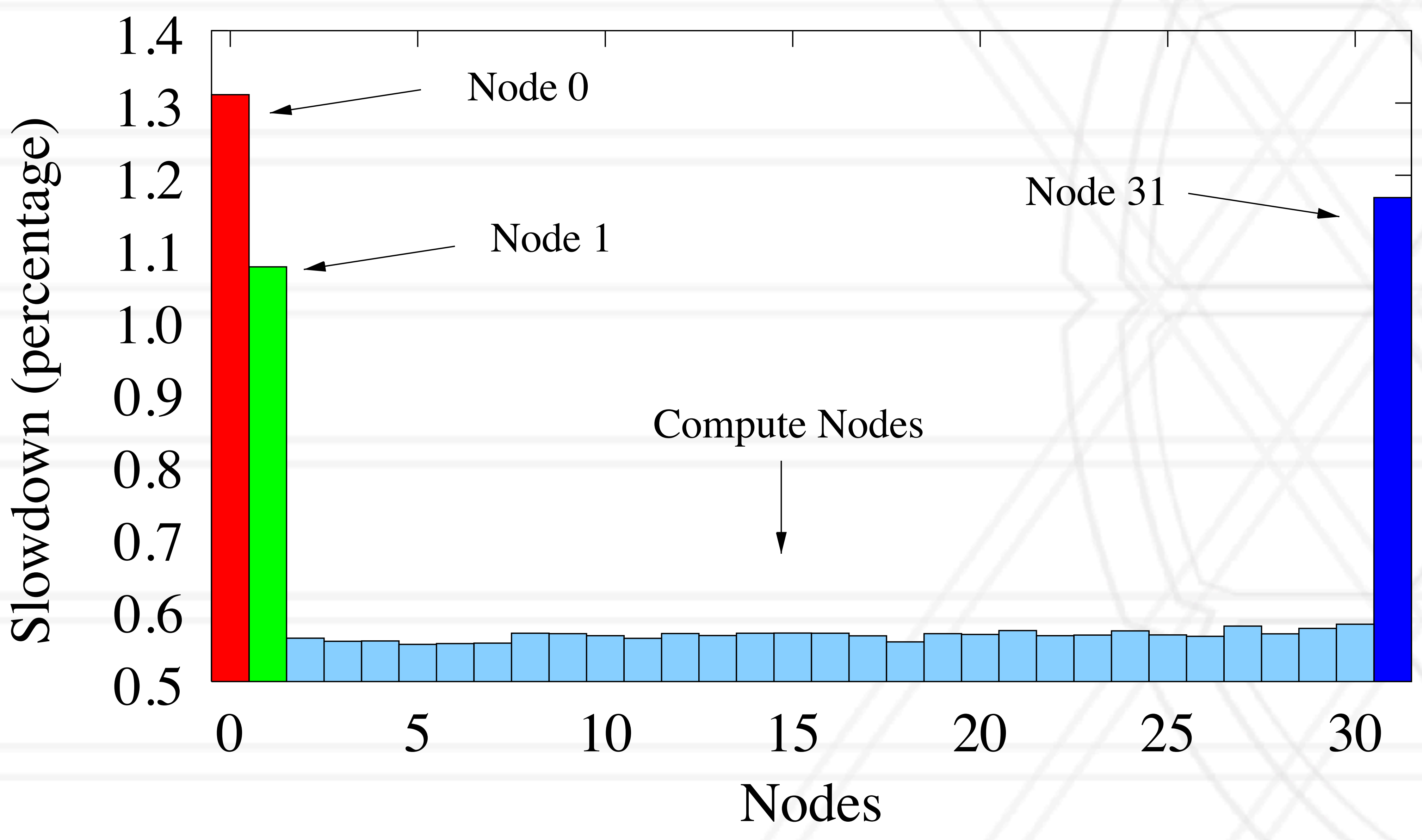
Benchmarks: https://asc.llnl.gov/sites/asc/files/2020-06/FTQFTW_Summary_v1.1.pdf

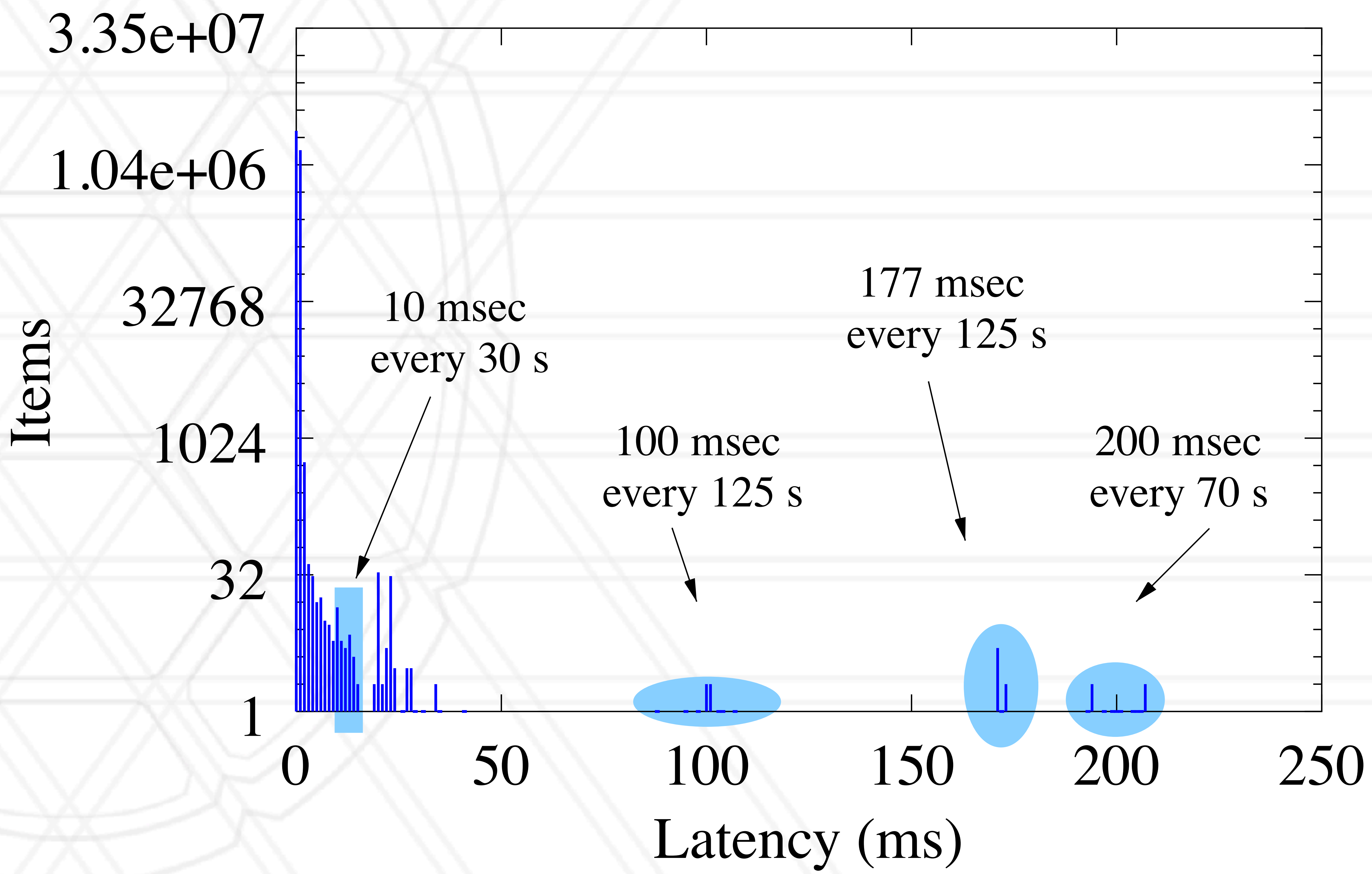
The Case of the Missing Supercomputer Performance



Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. 2003. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC '03). Association for Computing Machinery, New York, NY, USA, 55. DOI:https://doi.org/10.1145/1048935.1050204

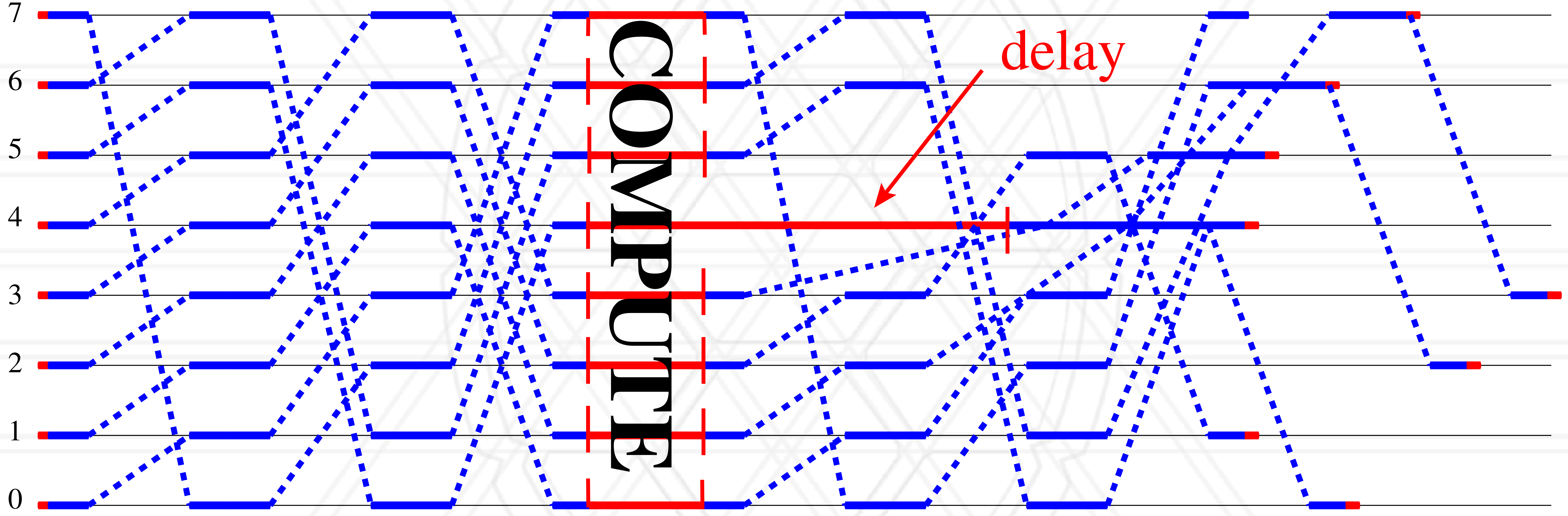
The Case of the Missing Supercomputer Performance





Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. 2003. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC '03). Association for Computing Machinery, New York, NY, USA, 55. DOI:https://doi.org/10.1145/1048935.1050204

Impact on communication



Hoefler et al.: https://htor.inf.ethz.ch/publications/img/hoefler-noise-sim.pdf

Mitigating OS noise

- Running a light-weight OS
- Turn off unnecessary daemons
- Reduce the frequency of daemons
- Dedicated cores for OS daemons
- User programs can avoid using certain cores

UNIVERSITY OF MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu