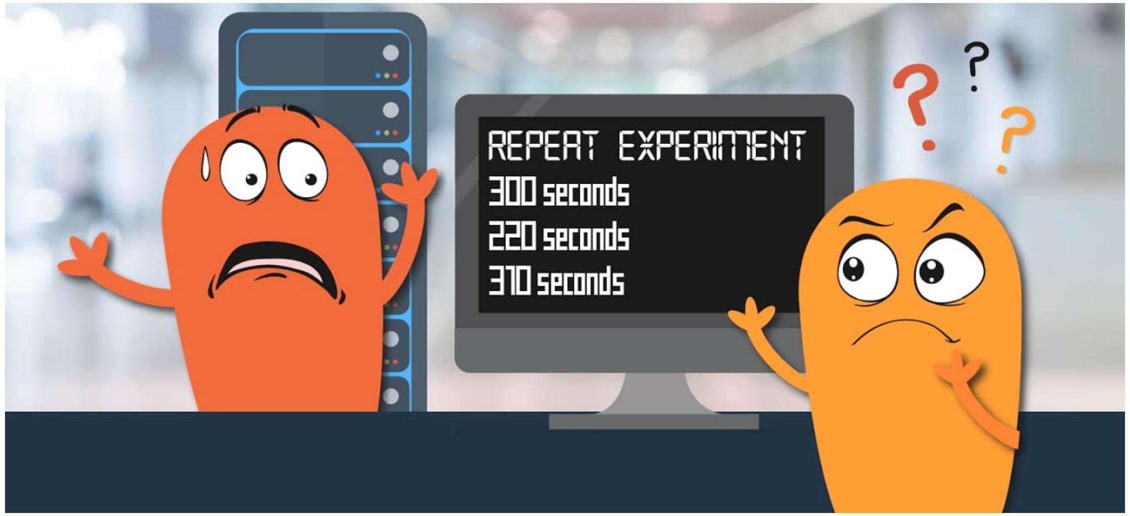
Run-to-run Variability on Theta and Best Practices for Performance Benchmarking

ALCF Developer Session – September 26th 2018

Sudheer Chunduri
sudheer@anl.gov

Run-to-run Variability

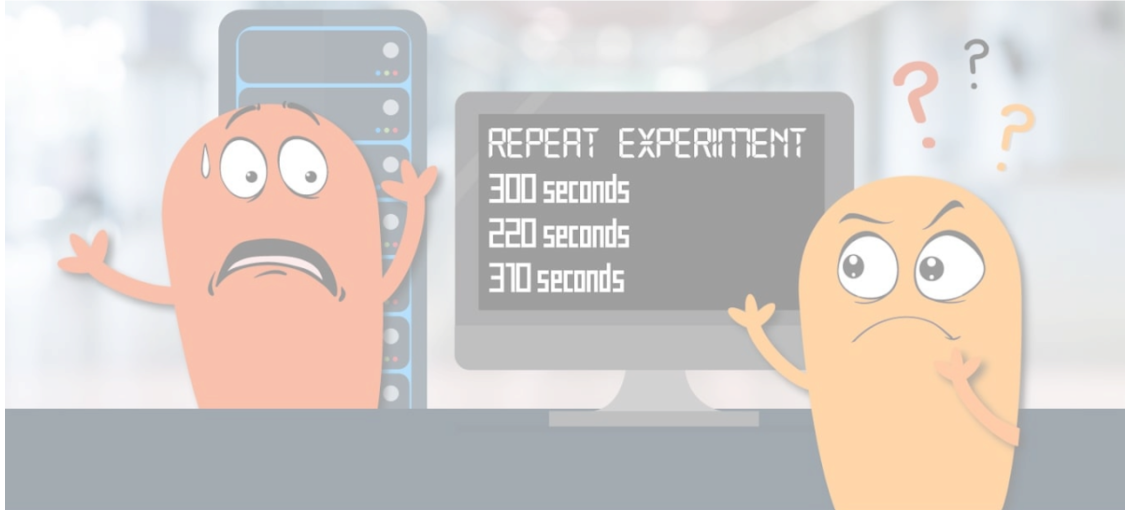


Equal work is not Equal time

Image courtesy: https://concertio.com/2018/07/02/dealing-with-variability/

Equal work is not Equal time

- **Sources of Variability**
  - Core-level
    - OS noise effects
    - Dynamic frequency scaling
    - Manufacturing variability
  - Node level
    - Shared cache contention on a multi-core
  - System level
    - Network congestion due to inter-job interference



Equal work is not Equal time

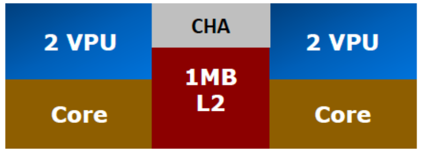
- **Challenges**
  - Less reliable performance measures (multiple repetitions with statistical significance analysis is required)
  - Performance tuning – quantifying the impact of a code change is difficult
  - Difficult to predict job duration
    - Less user productivity
    - Inefficient system utilization
    - Complicates job scheduling

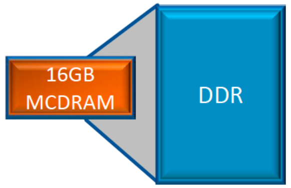
Outline

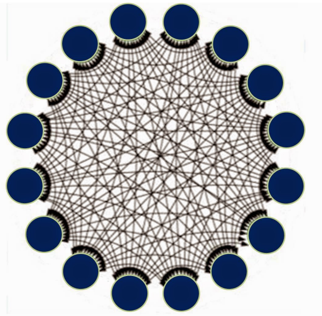
- Overview of Theta Architecture
- Evaluation of run-to-run variability on Theta
  - Classify and quantify sources of variability
  - Present ways to mitigate *wherever possible*
- Recommended Best practices for performance benchmarking

Theta System Overview

- **System:**
  - Cray XC40 system (#21 in Top500 in June 2018)
  - 14 similar systems in top 50 supercomputers
  - 4,392 compute nodes/281,088 cores, 11.69 PF peak performance
- **Processor:**
  - 2nd Generation Intel Xeon Phi (Knights Landing) 7230
  - 64 cores - 2 cores on one tile with shared L2
  - 1.3 base frequency, can turbo up to 1.5 GHz
- **Node:**
  - Single socket KNL
  - 192 GB DDR4-2400 per node
  - 16 GB MCDRAM per node (Cache mode/Flat mode)
- **Network:**
  - Cray Aries interconnect with Dragonfly network topology
  - Adaptive routing

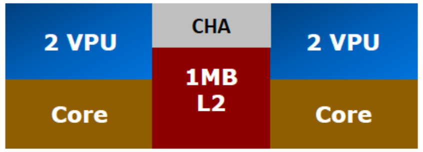




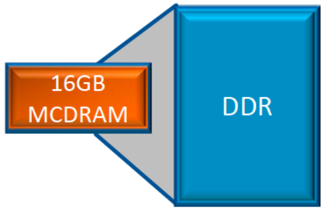


Figures source: Intel, Cray

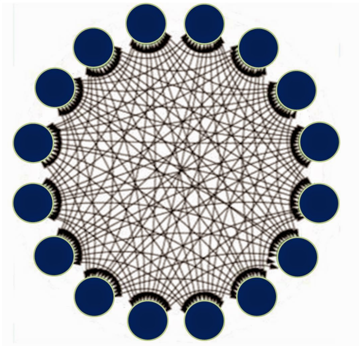
Aspects of Variability Examined



- Core level
  - OS noise effects
  - Core to core variability
  - Cores within a tile



- Node level
  - MCDRAM memory mode effects



- System level
  - Network congestion
  - Node placement and routing mode effects

Micro-benchmarks

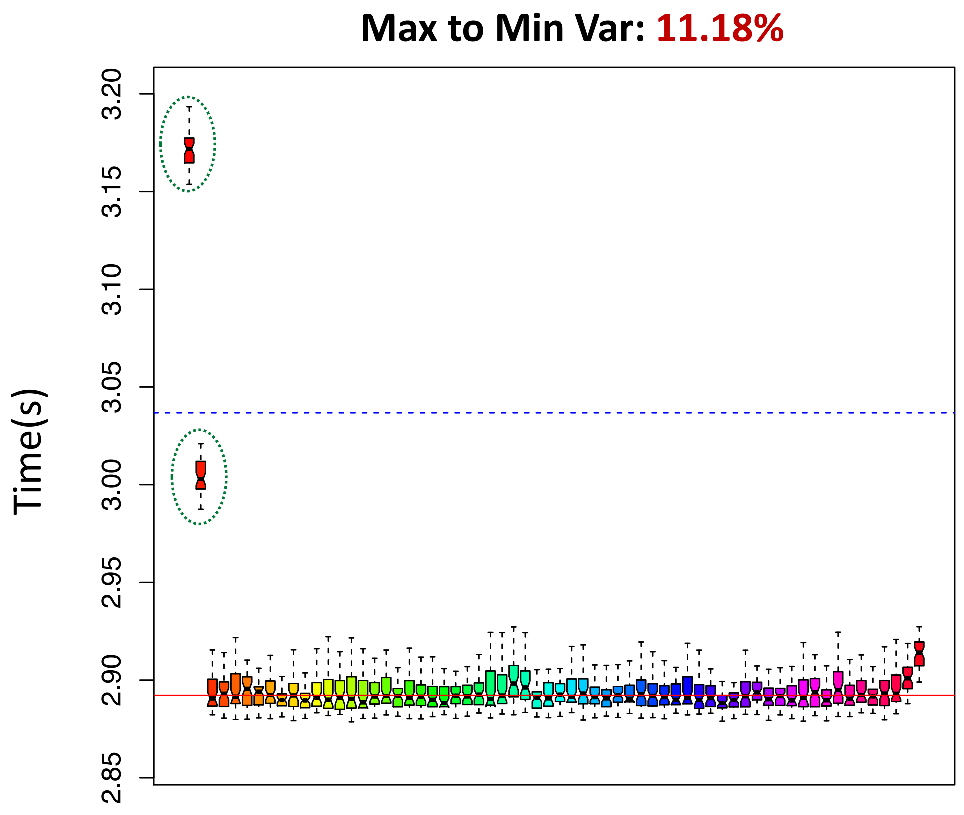
Mini-apps

Applications

Figures source: Intel, Cray

Core-level Variability

- Each core runs the **MKL DGEMM** benchmark
- Matrix size chosen so as to fit within L1 cache

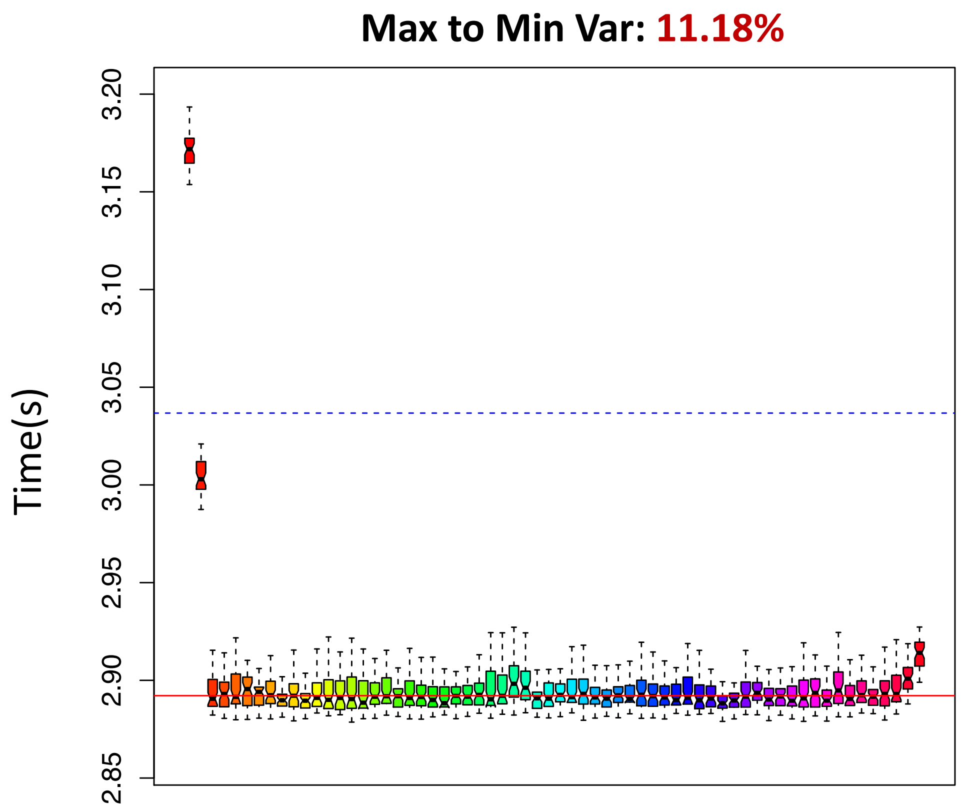


DGEMM on 64 cores

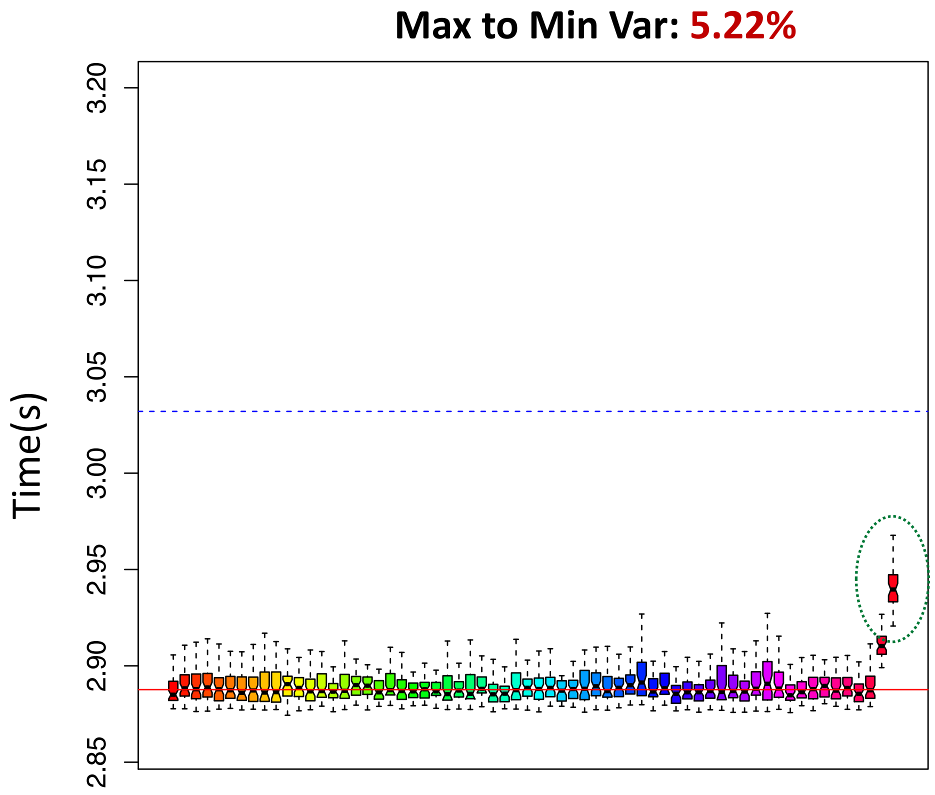
Core-level Variability

- Each core runs the **MKL DGEMM** benchmark
- Matrix size chosen so as to fit within L1 cache

- **Core specialization** – A Cray OS feature allowing users to reserve cores for handling system services



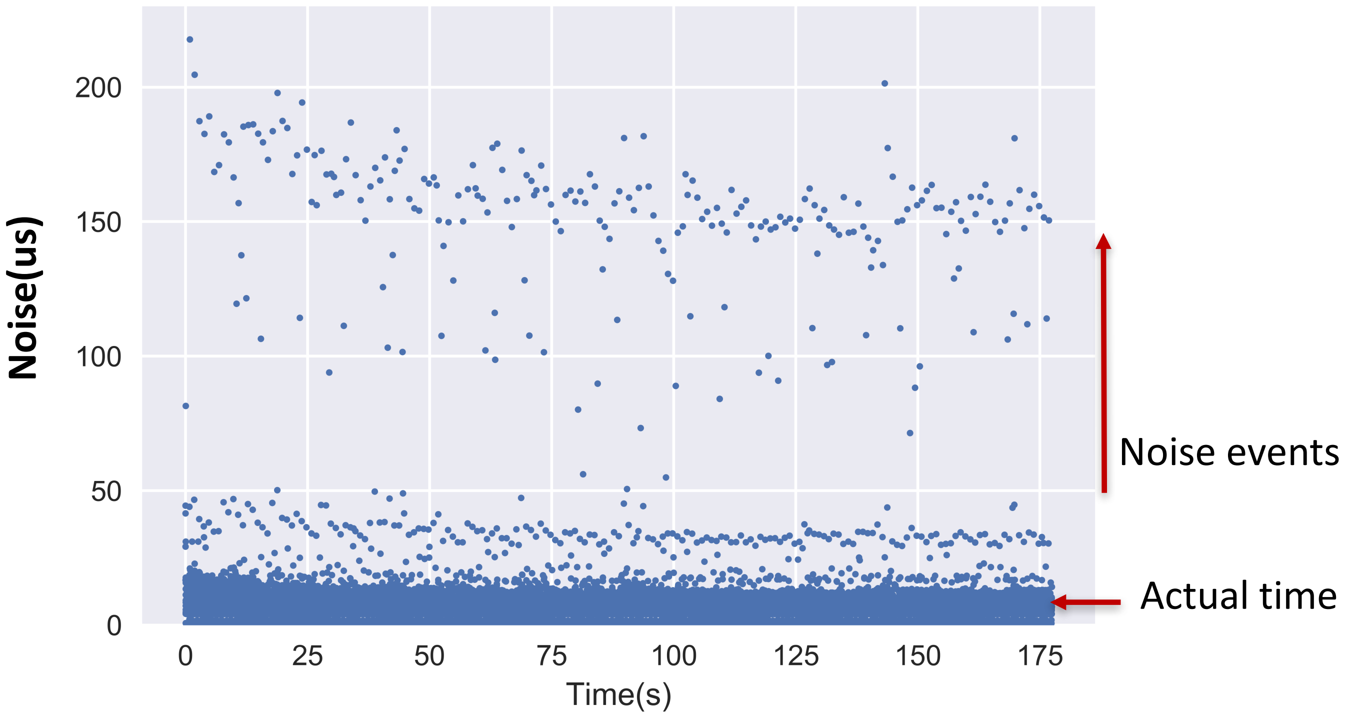
DGEMM on 64 cores



DGEMM on 64 cores with Core Specialization

Core-level Variability

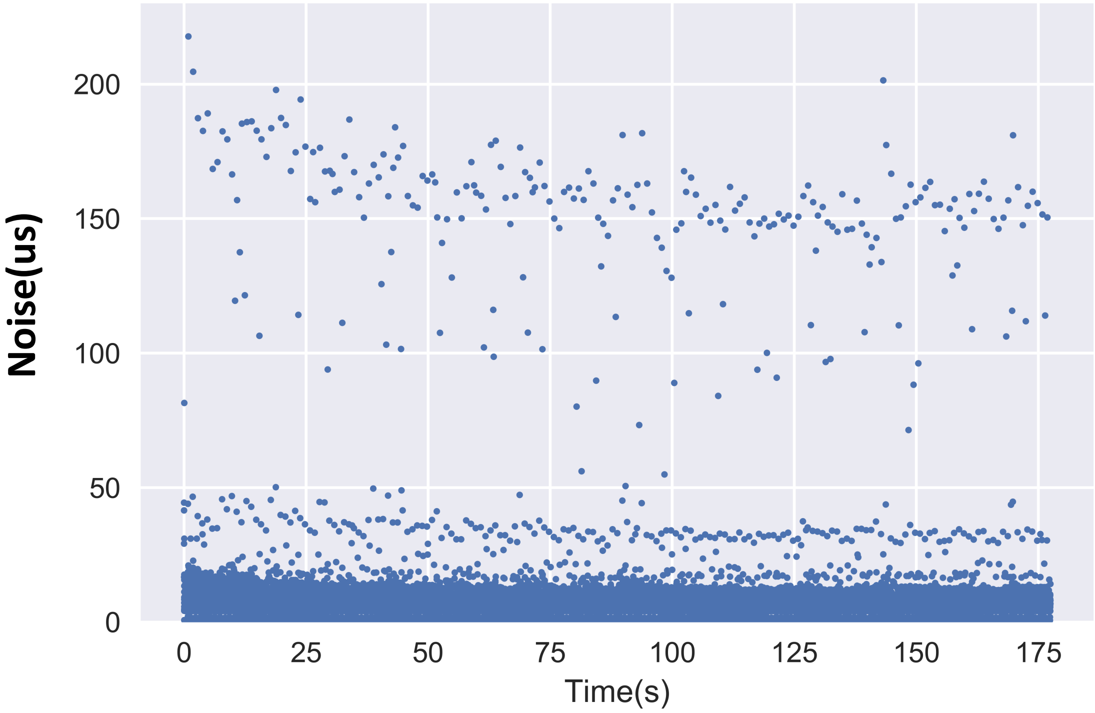
- Benchmark: **Selfish**
- Runs in a tight loop and measures the time for each iteration.
- If an iteration takes longer than a particular threshold, then the timestamp (Noise) is recorded.



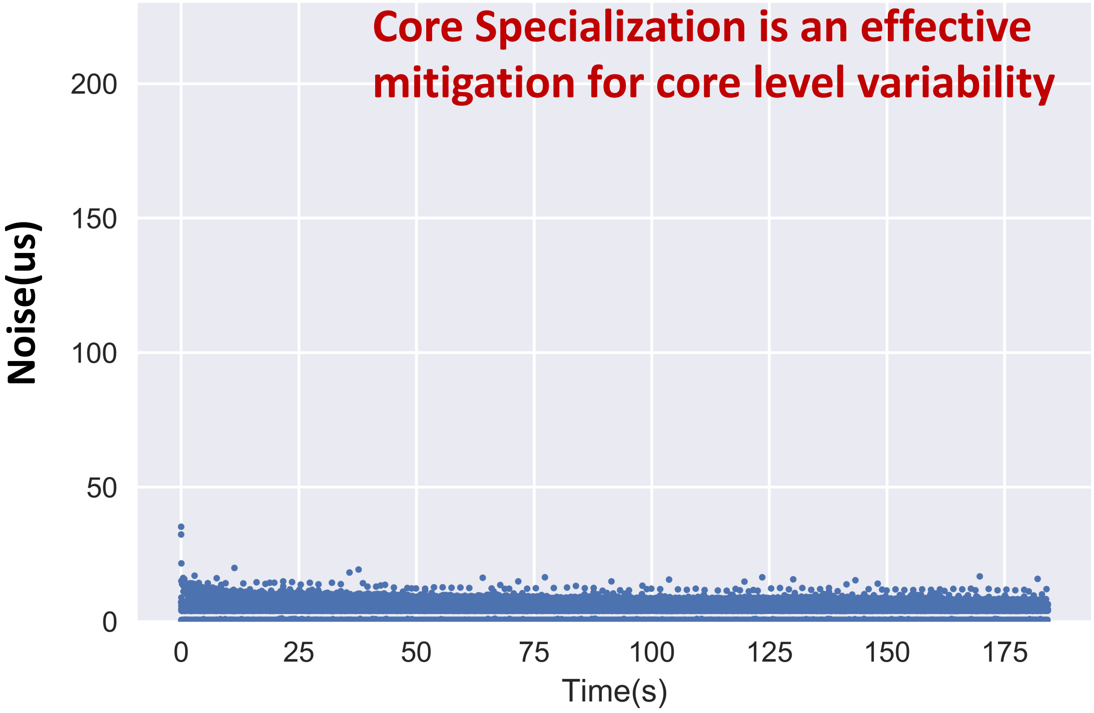
OS noise effects on a core **without Core Specialization**

Core-level Variability

- Benchmark: **Selfish**
- Runs in a tight loop and measures the time for each iteration.
- If an iteration takes longer than a particular threshold, then the timestamp (Noise) is recorded.



OS noise effects on a core **without Core Specialization**

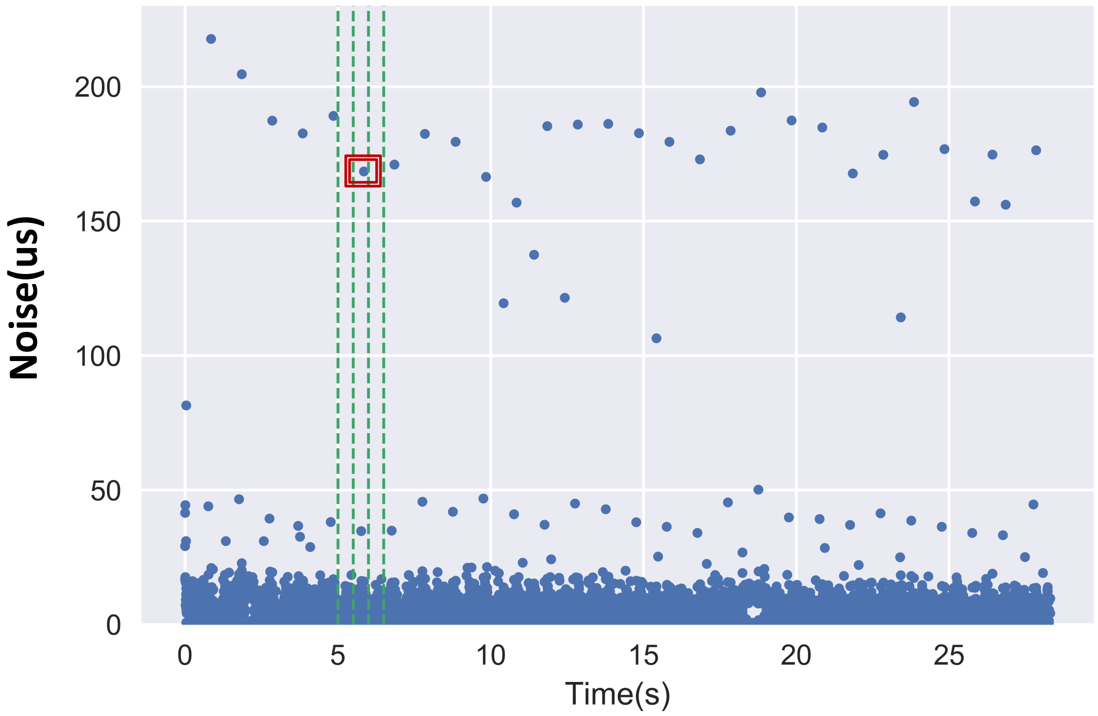


OS noise effects on a core **with Core Specialization**

Core-level Variability

Benchmark: **Selfish**

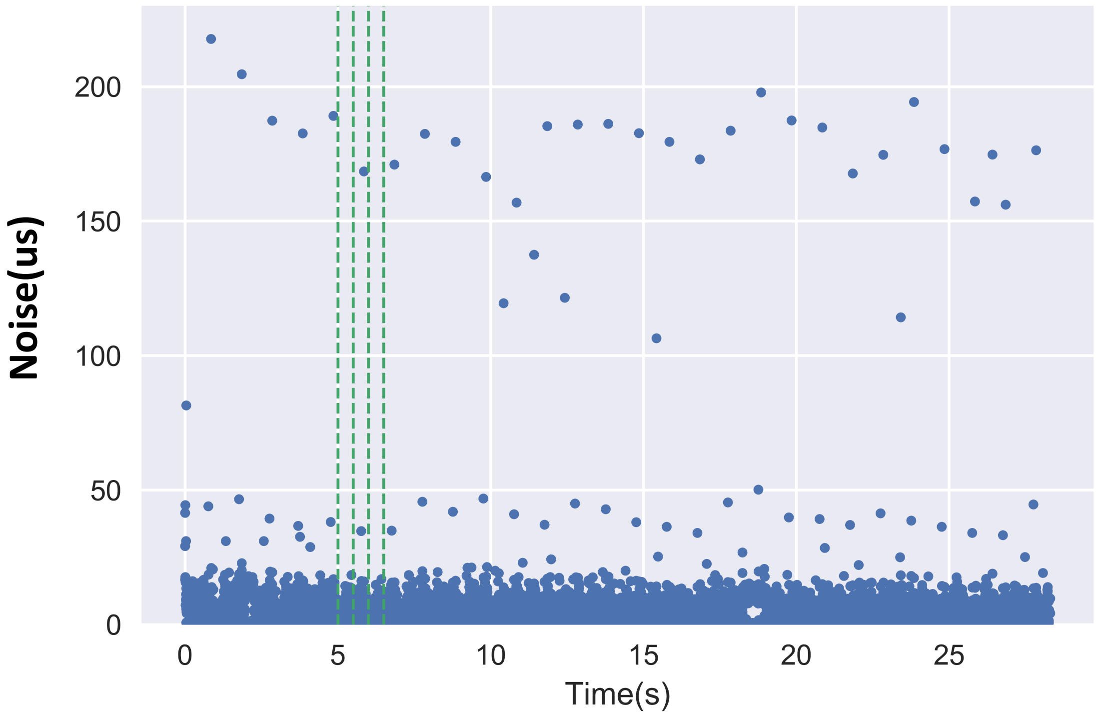
- Small micro-benchmark in the milliseconds range
- Noise is significant



Core-level Variability

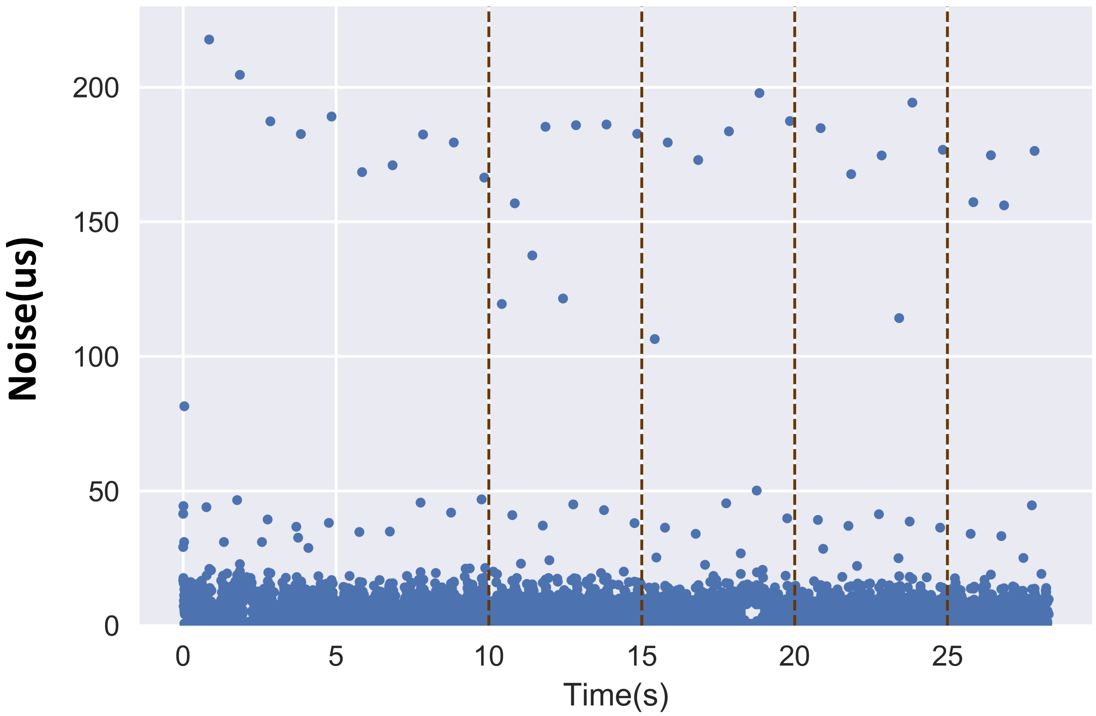
Benchmark: **Selfish**

- Small micro-benchmark in the milliseconds range
- Noise is significant



Micro-benchmark in the seconds range

Time scale matters – runtimes greater than seconds don't see the impact



Node-level Variability

Variability due to memory mode

KNL Has two types of memory

DRAM - 192 GB capacity
~ 90 GB/s effective bandwidth

MCDRAM - 16 GB capacity
~ 480 GB/s effective bandwidth

Node-level Variability

Variability due to memory mode

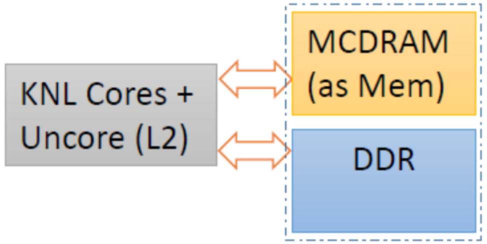
KNL Has two types of memory

DRAM - 192 GB capacity
~ 90 GB/s effective bandwidth

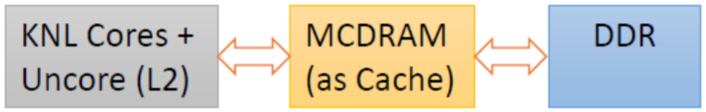
MCDRAM - 16 GB capacity
~ 480 GB/s effective bandwidth

MCDRAM can be operated in two modes

Flat Mode



Cache Mode



Node-level Variability

Variability due to memory mode

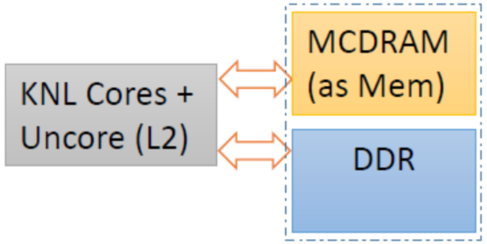
KNL Has two types of memory

DRAM - 192 GB capacity
~ 90 GB/s effective bandwidth

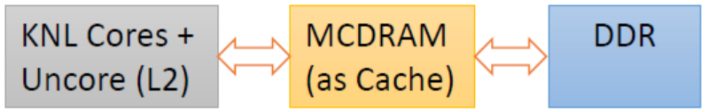
MCDRAM - 16 GB capacity
~ 480 GB/s effective bandwidth

MCDRAM can be operated in two modes

Flat Mode



Cache Mode



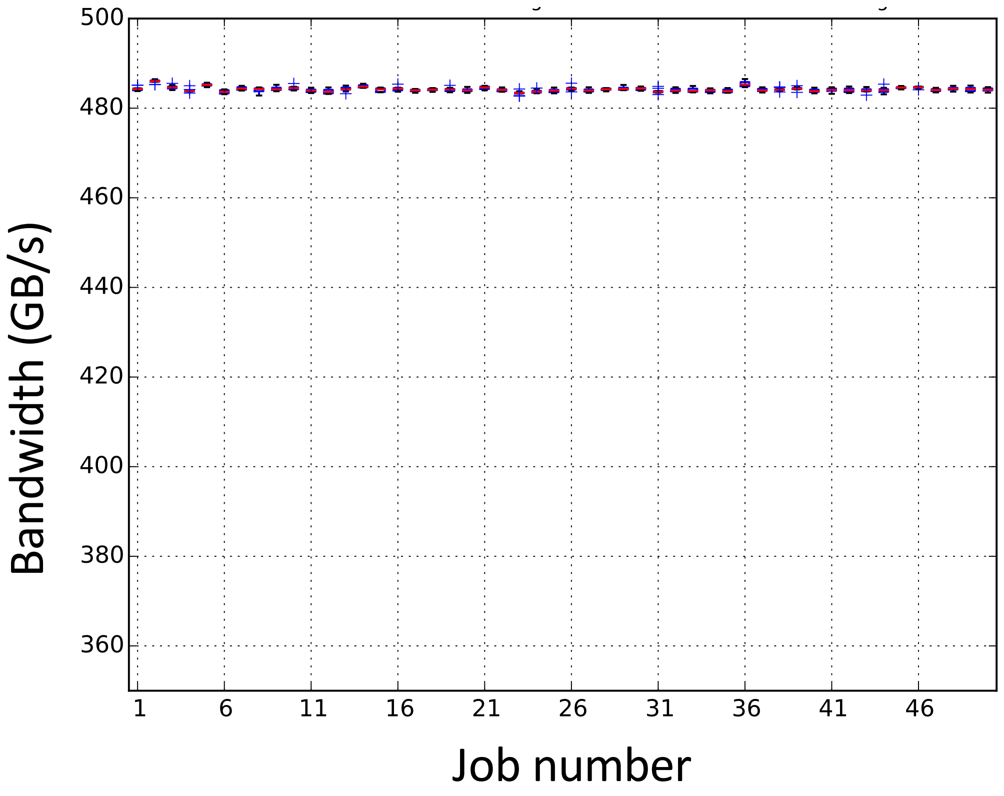
Source of Variability:

- In cache mode, MCDRAM operated as direct-mapped cache to DRAM
- Potential conflicts because of the direct mapping

Node-level variability

Stream TRIAD in flat mode

STREAM benchmark using 63 cores with one core for core specialization & working set of 7.5 GB



Less than **1%** variability: **480 GB/s** effective bandwidth

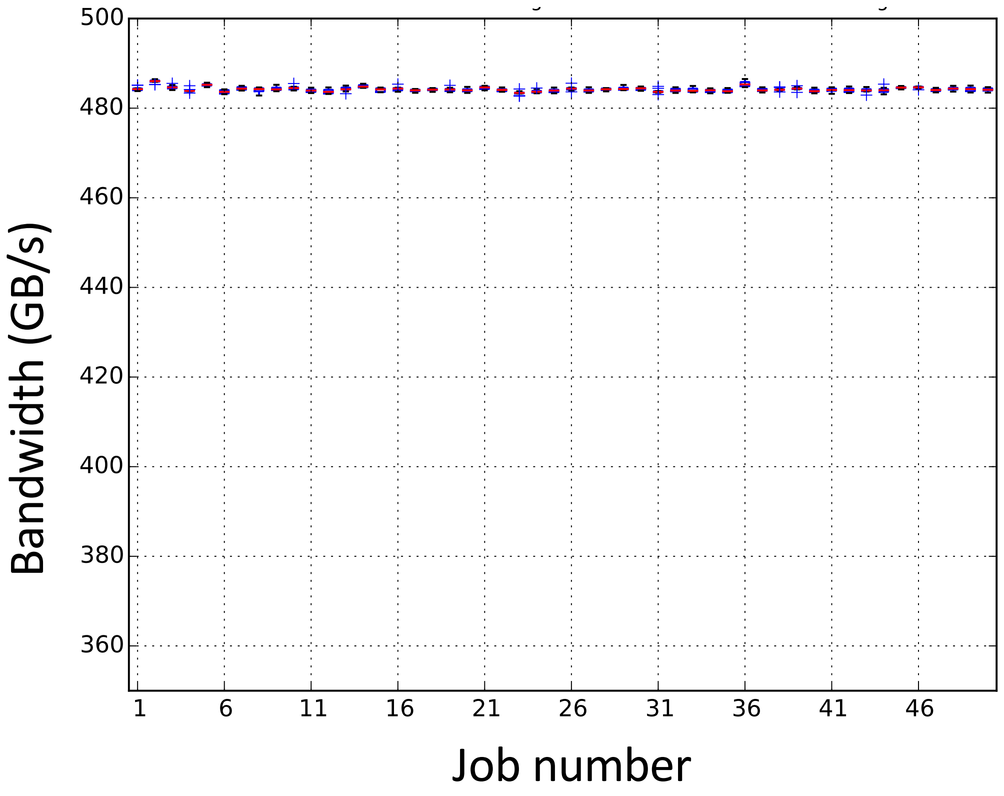
STREAM TRIAD benchmark used to measure memory bandwidth with

A(i) = B(i) + s * C(i)

Node-level variability

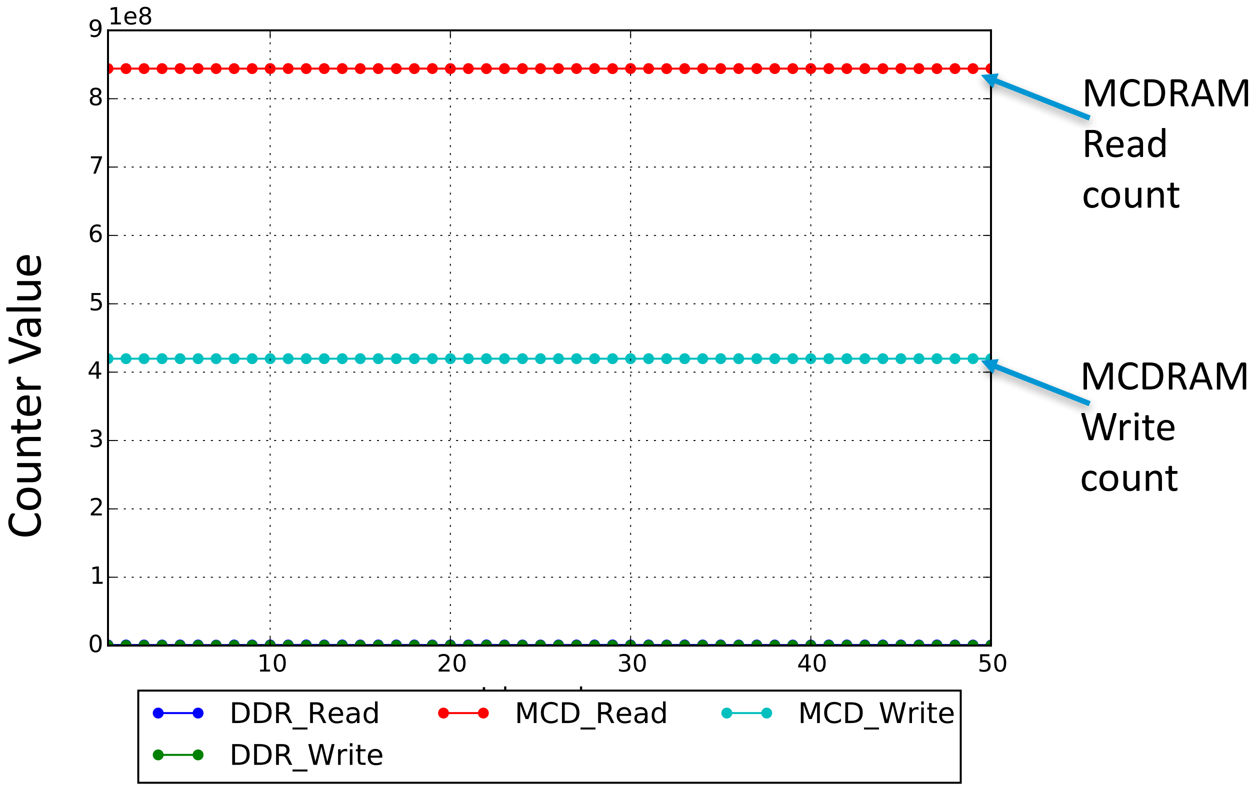
Stream TRIAD in flat mode

STREAM benchmark using 63 cores with one core for core specialization & working set of 7.5 GB



Less than **1%** variability: **480 GB/s** effective bandwidth

DRAM Reads & Writes
MCDRAM Reads & Writes



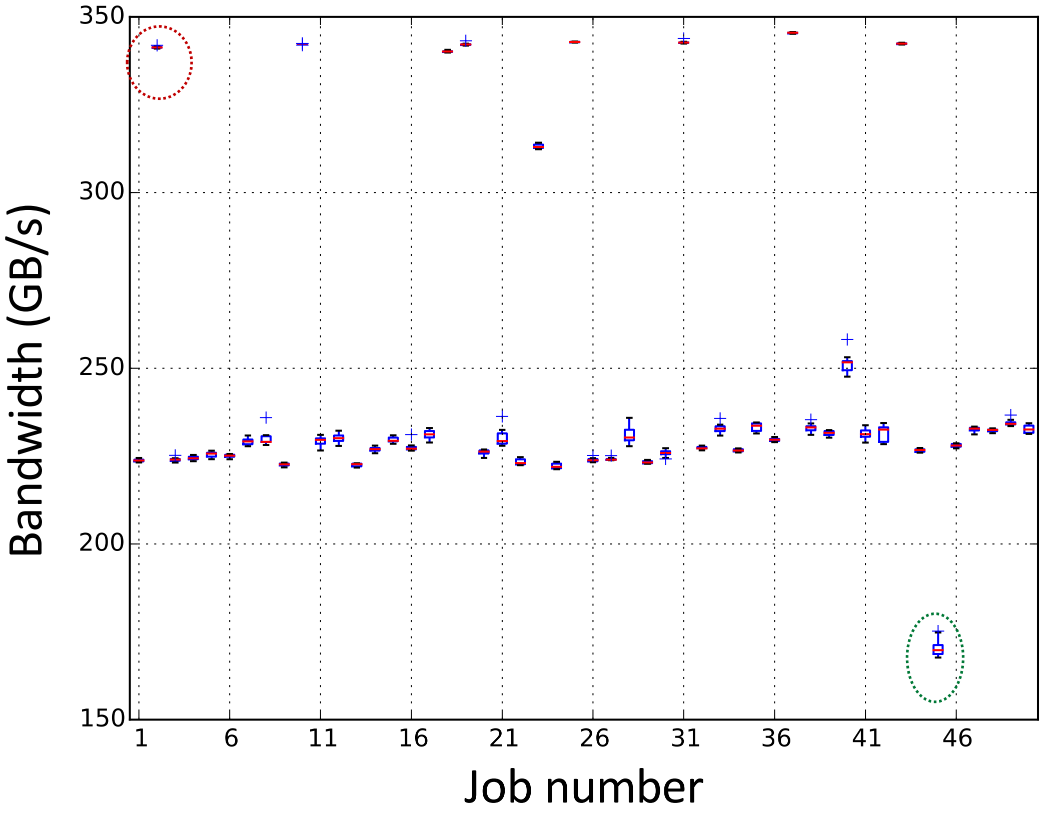
MCDRAM writes are consistent across all the nodes

Argonne NATIONAL LABORATORY

Node-level variability

Stream TRIAD in cache mode

STREAM benchmark using 63 cores with one core for core specialization & working set of 7.5 GB



Max. **4.5%** run-to-run, **2X** job-to-job variability
350 GB/s effective bandwidth

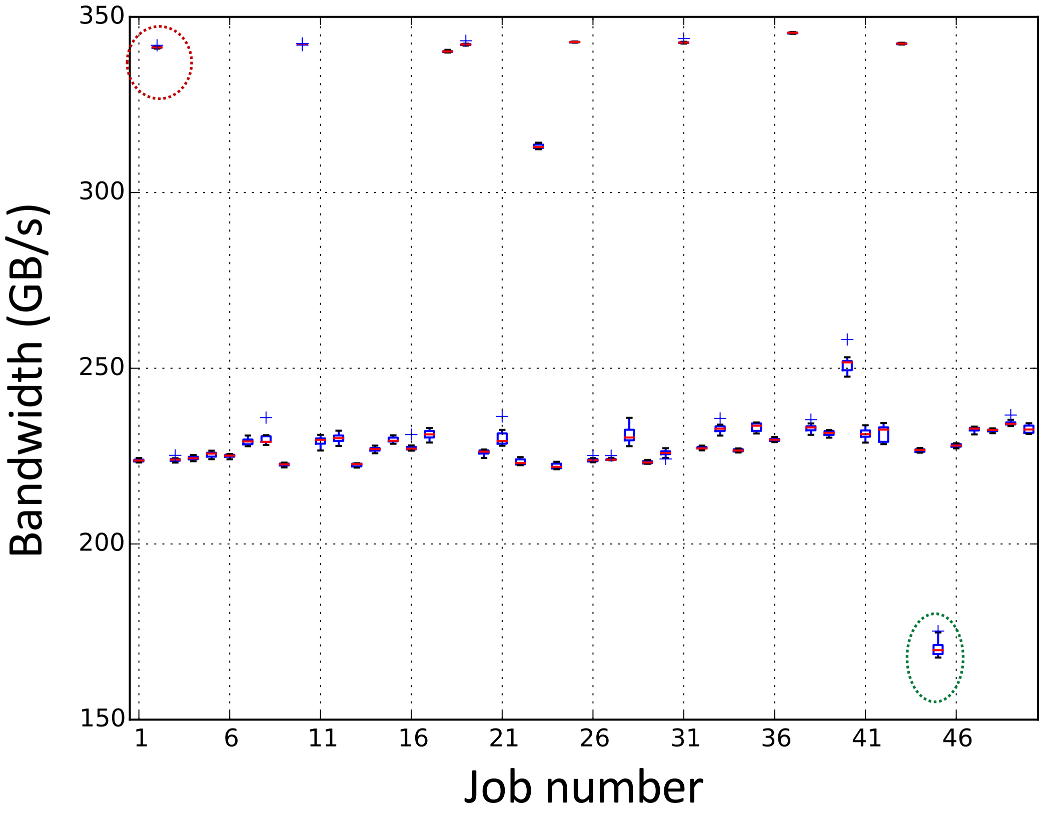
Argonne NATIONAL LABORATORY

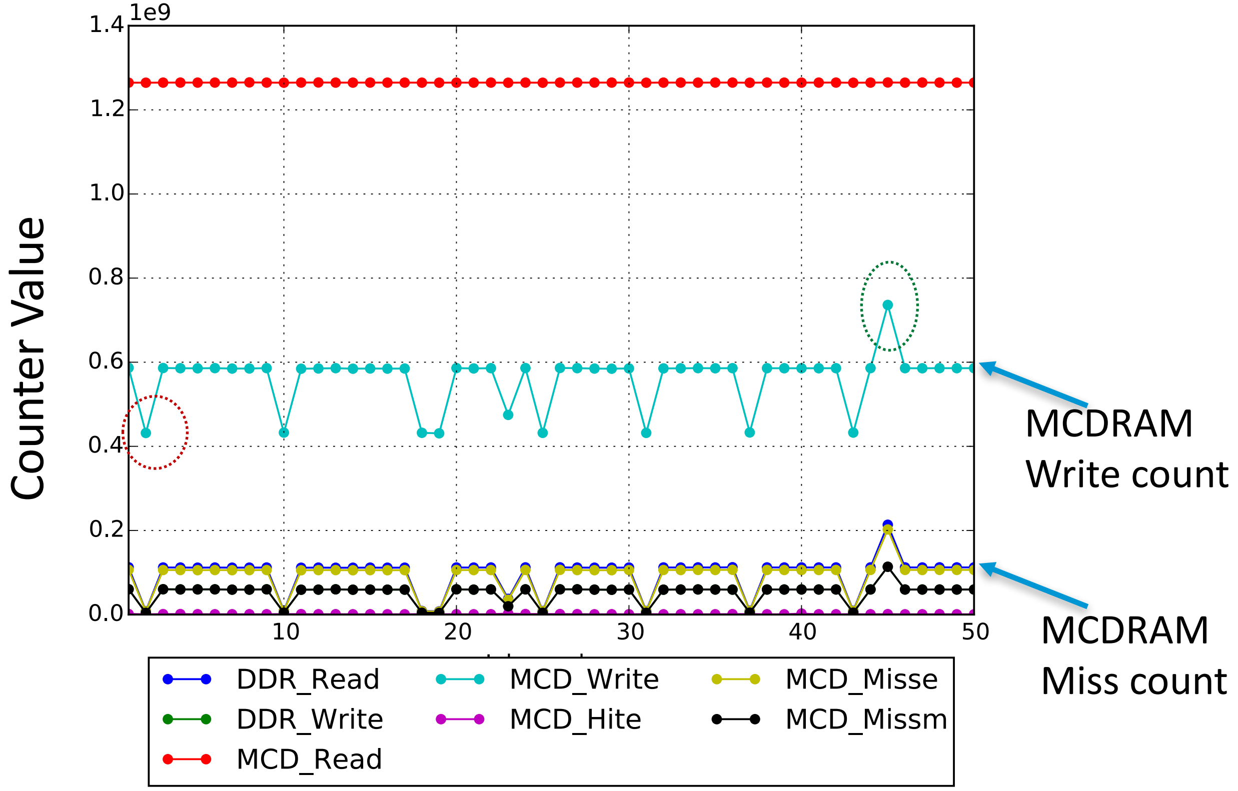
Node-level variability

Stream TRIAD in cache mode

STREAM benchmark using 63 cores with one core for core specialization & working set of 7.5 GB

DRAM Reads & Writes
MCDRAM Hits & Misses, Reads & Writes

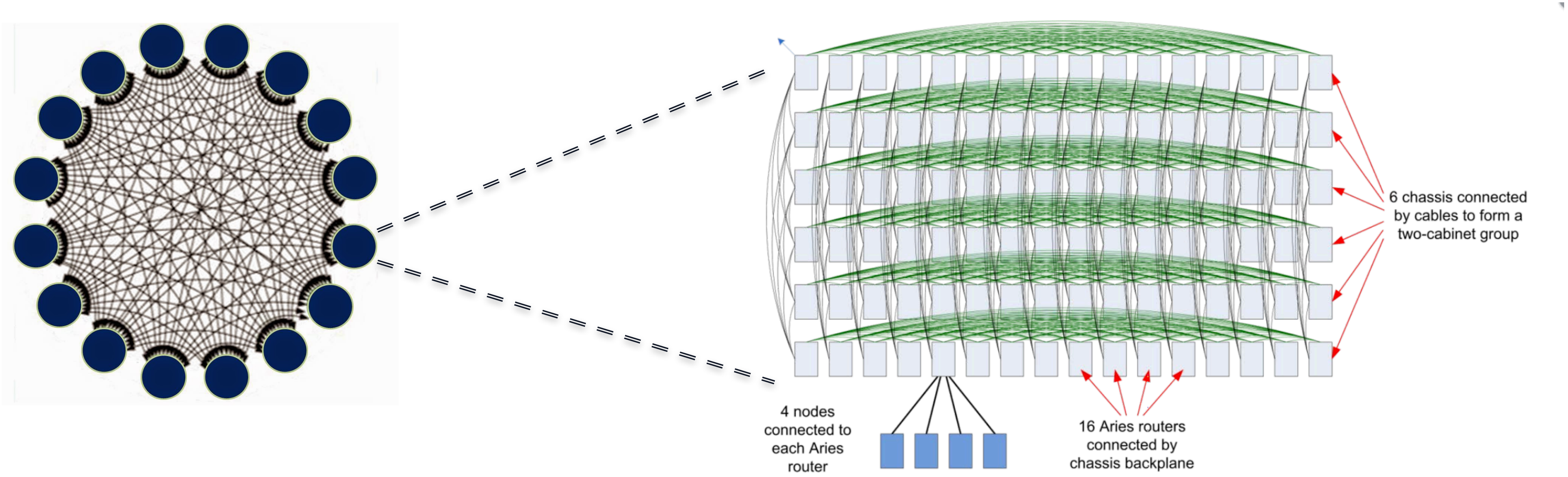




Max. **4.5%** run-to-run, **2X** job-to-job variability
350 GB/s effective bandwidth

Higher bandwidth correlates with lower MCDRAM miss ratio (More MCDRAM writes due to conflicts!)

Network-level variability



- Cray XC Dragonfly topology
  - Potential links sharing between the user jobs
  - High chances for inter-job contention
- Sources of variability -> Inter-job contention
  - Size of the job, Node placement , Workload characteristics , Co-located job mix

Network-level variability

MPI Collectives

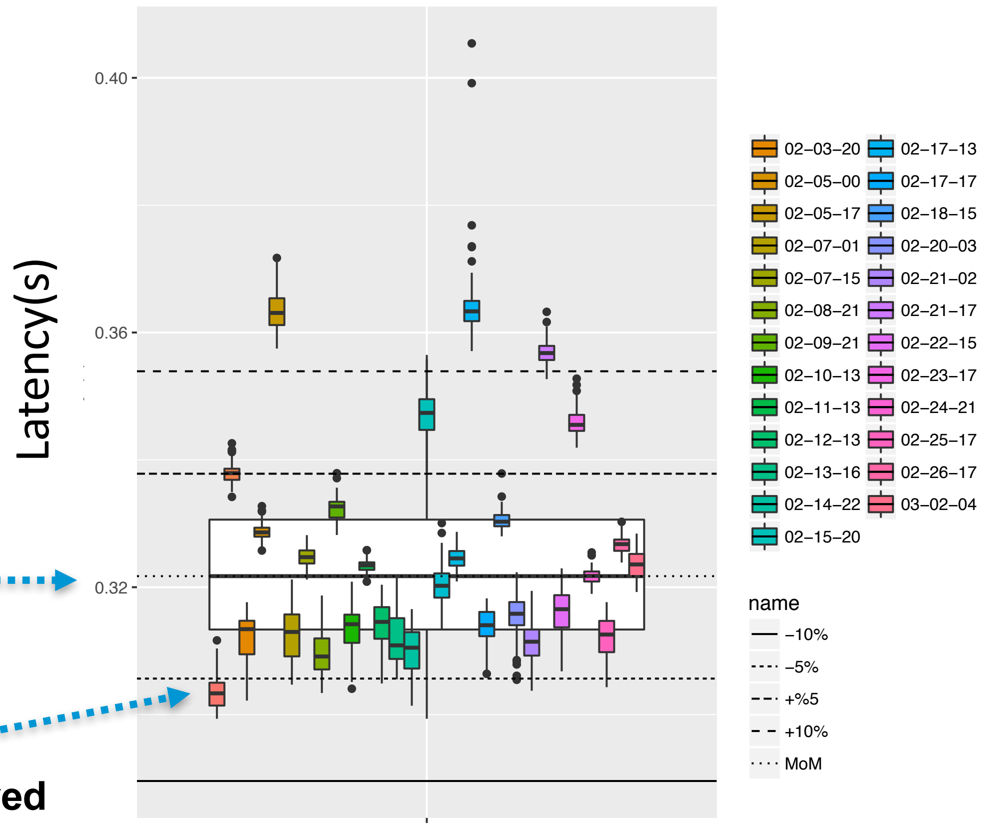
- **MPI_Allreduce** using 64 processes with 8 MB message
- Repeated 100 times within a job
- Measured on several days
  - Changes in node placement and Job mix
- Isolated system run:
  - < **1%** variability
  - Best observed

Network-level variability

MPI Collectives

- **MPI_Allreduce** using 64 processes with 8 MB message
- Repeated 100 times within a job
- Measured on several days
  - Changes in node placement and Job mix
- Isolated system run:
  - < **1%** variability
  - Best observed
- Variability is around **35%**
  - Much higher variability with smaller message sizes (not shown here)
- Each box shows the median, IQR (Inter-Quartile Range) and the outliers

MoM



Best observed

Different jobs

128 nodes Allreduce 8MB 64 PPN

Argonne NATIONAL LABORATORY

Summary on Variability

- Core-to-core level variability due to OS noise
  - Core 0 is slow compared to rest of the cores
  - Crucial for low-latency MPI benchmarking and for micro-kernel benchmarking
  - Longer time scales don't see the effect
  - **Core specialization helps reduce the overhead**
  - Frequency scaling effects are not dominant enough to induce variability
- Node level variability due to MCDRAM cache page conflicts
  - Around 2X variability on STREAM benchmark
  - Linux Zone sort helps improve average performance and reduce variability to some extent
  - Example miniapps that are sensitive: Nekbone, MiniFE
  - For applications with working sets that fits within MCDRAM, using **Flat mode is the mitigation**
- Network level variability due to inter-job contention
  - Up to 35% for large message sized MPI collectives
  - Even higher variability for latency bound small sized collectives
  - No obvious mitigation

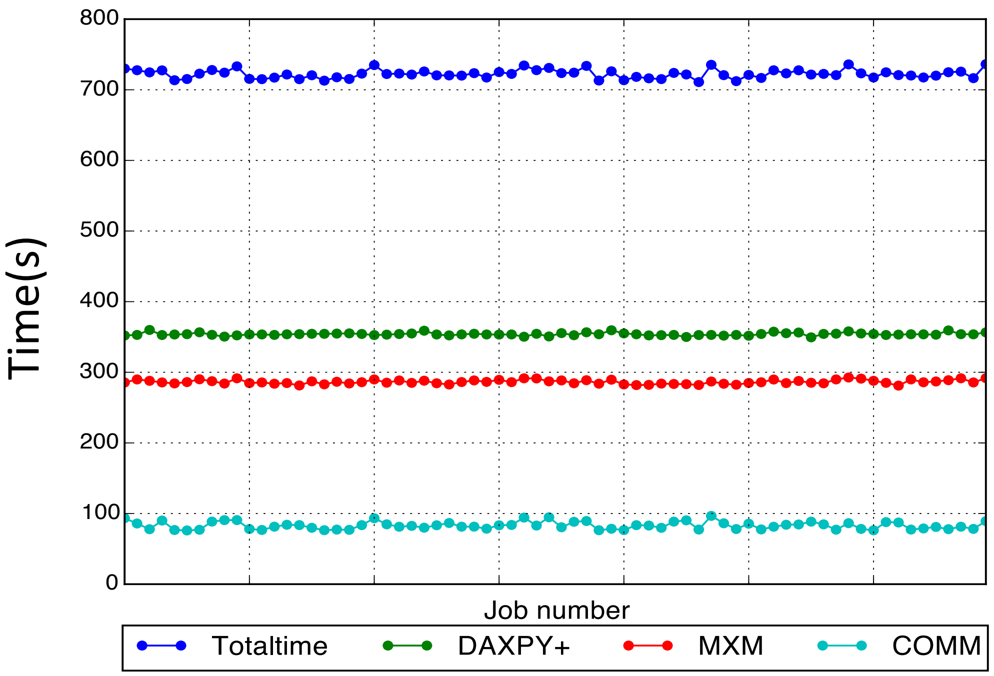
Application Level Variability

Nekbone variability at the node level

Nekbone: Nekbone mini-app derived from Nek5000

- Streaming kernels – BW bound – **DAXPY+**
- Matrix multiply – Compute bound – **MXM**
- Communication bound – **COMM**

Max. to Min. ratio = **3.5%**



Flat mode on Theta

Application Level Variability

Nekbone variability at the node level

Nekbone: Nekbone mini-app derived from Nek5000

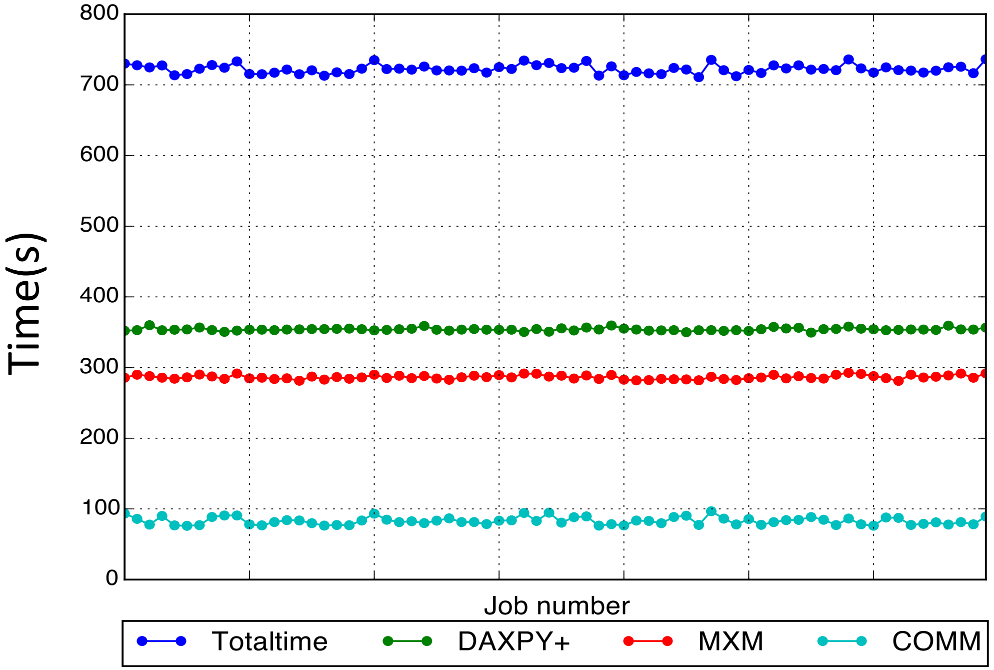
- Streaming kernels – BW bound – **DAXPY+**
- Matrix multiply – Compute bound – **MXM**
- Communication bound – **COMM**

Problem is memory bandwidth intensive

3.57% Max-to-Min variability in **Flat mode**

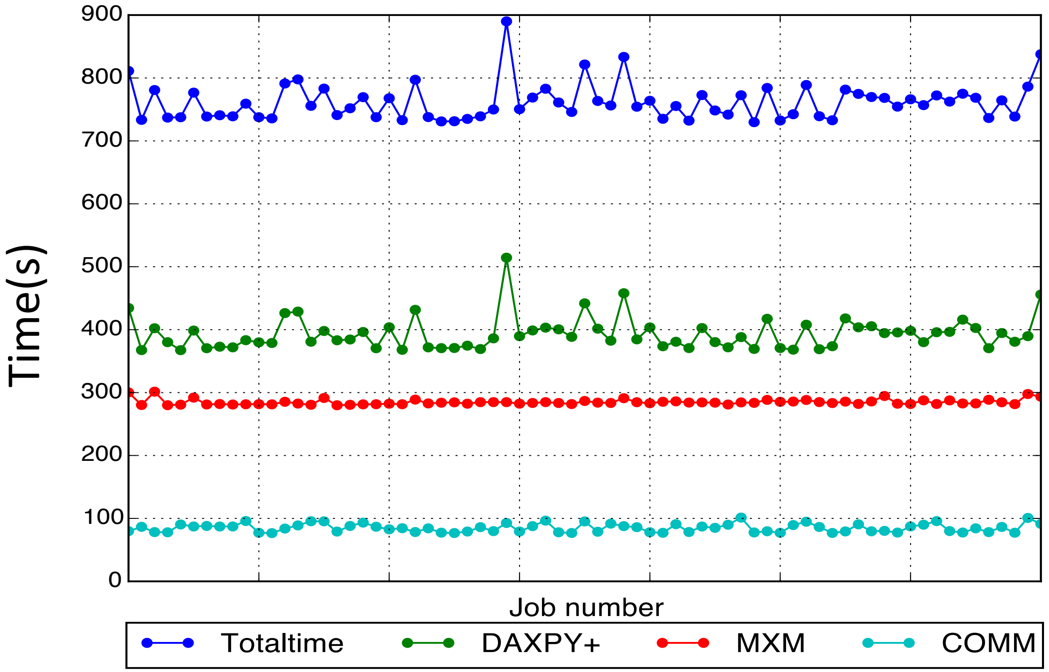
22% Max-to-Min variability in **Cache-mode**

Max. to Min. ratio = **3.5%**



Flat mode on Theta

Max. to Min. ratio = **22%**

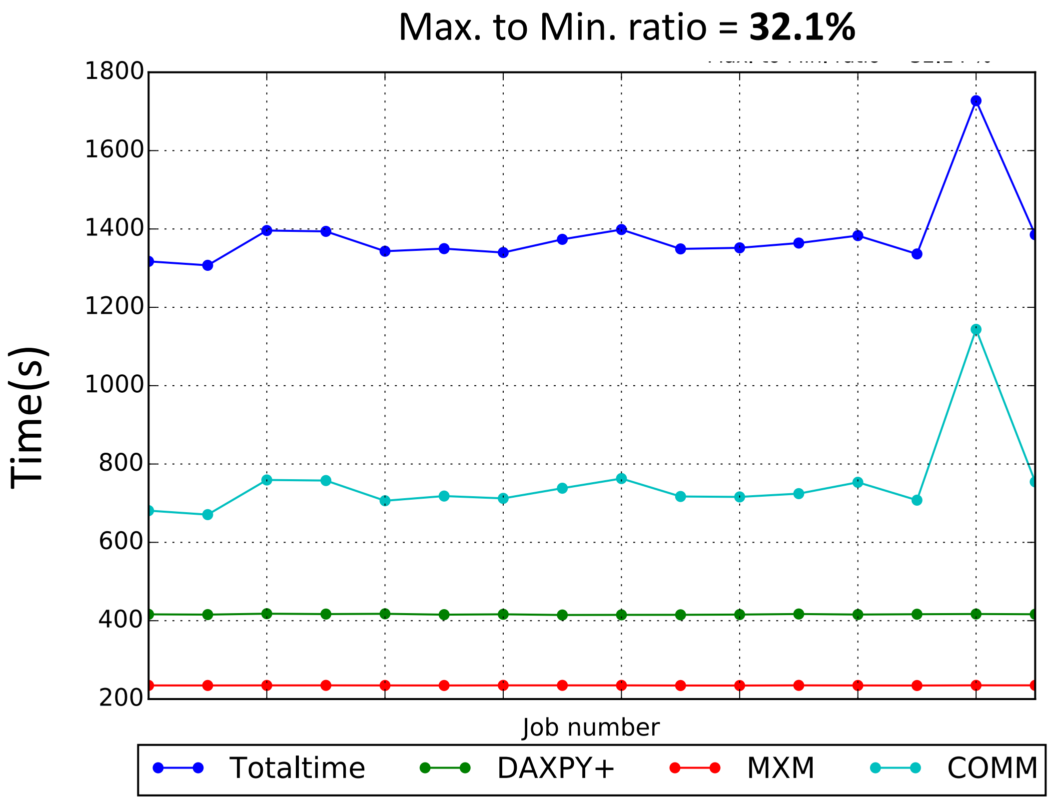


Cache mode on Theta

Application Level Variability

Nekbone variability at the network level

With a different input, Nekbone is communication bound
32.14% variability on 128 node jobs on Theta
Variability in Total time ~ variability in COMM time



128 nodes on Theta

Argonne NATIONAL LABORATORY

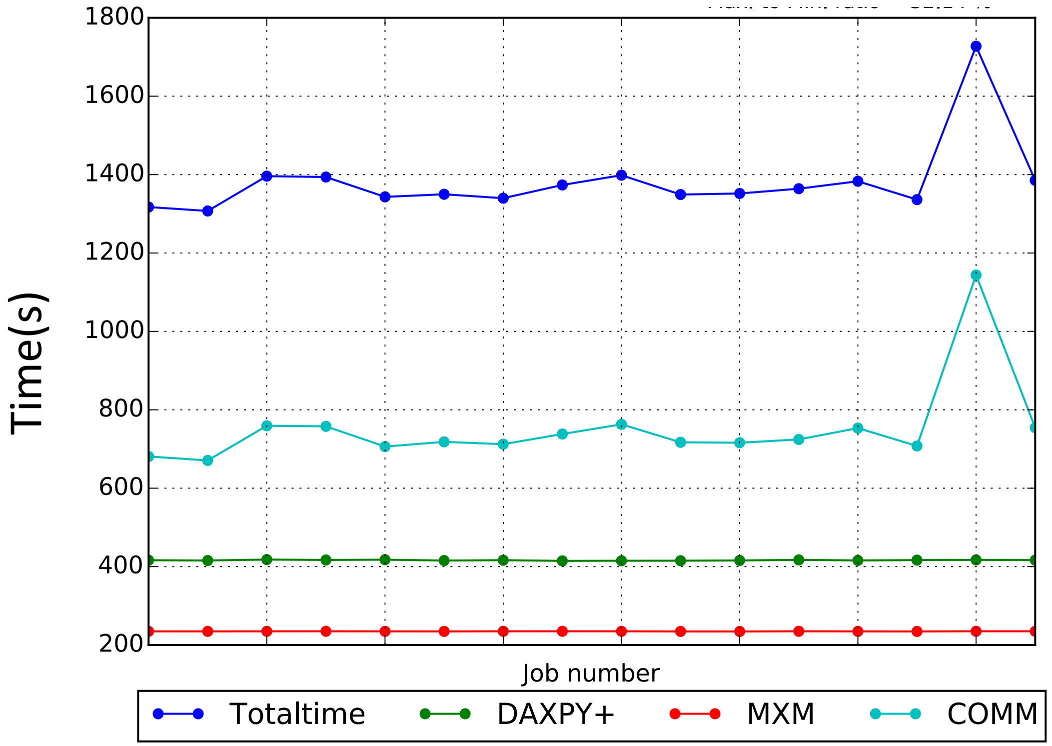
Application Level Variability

Nekbone variability at the network level

With a different input, Nekbone is communication bound
32.14% variability on 128 node jobs on Theta
Variability in Total time ~ variability in COMM time

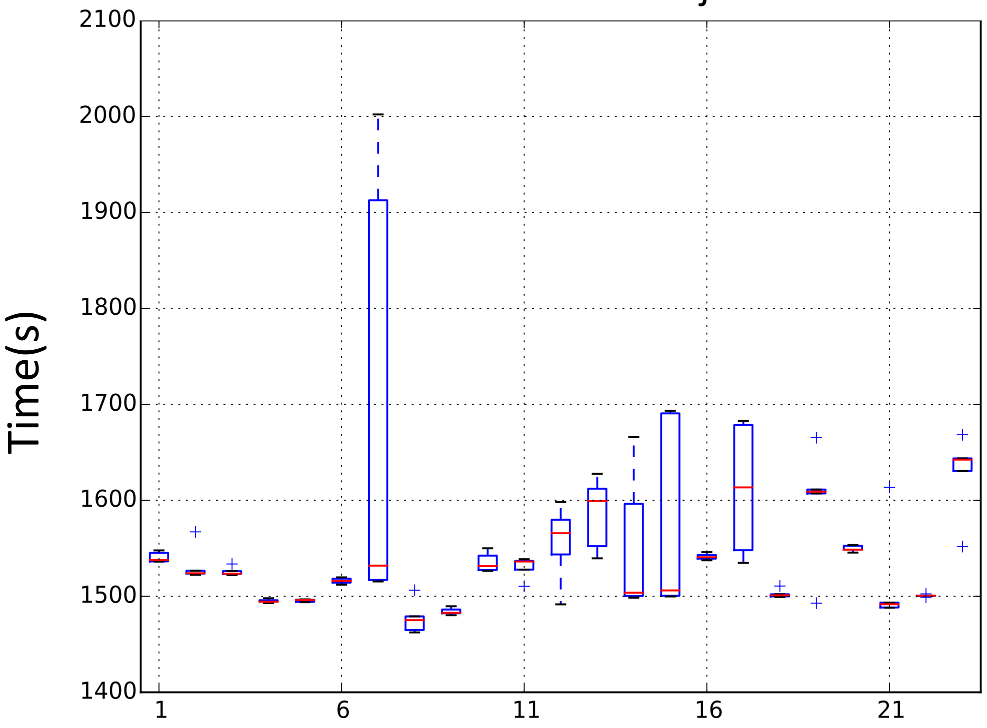
5 repetitions within a job
All use the same **node allocation** in a job

Max. to Min. ratio = **32.1%**



128 nodes on Theta

Run to Run ratio= 32.1% Job to job ratio = **36.9%**



256 nodes on Theta

Application Level Variability

MILC variability at the network level

- **MILC**
  - MIMD Lattice Computation QCD Code simulating 4D SU(3) lattice gauge theory
  - Performs large scale numerical simulations to study quantum chromodynamics (QCD)
  - Compute intensive per one lattice site with low memory footprint per compute node

Application Level Variability

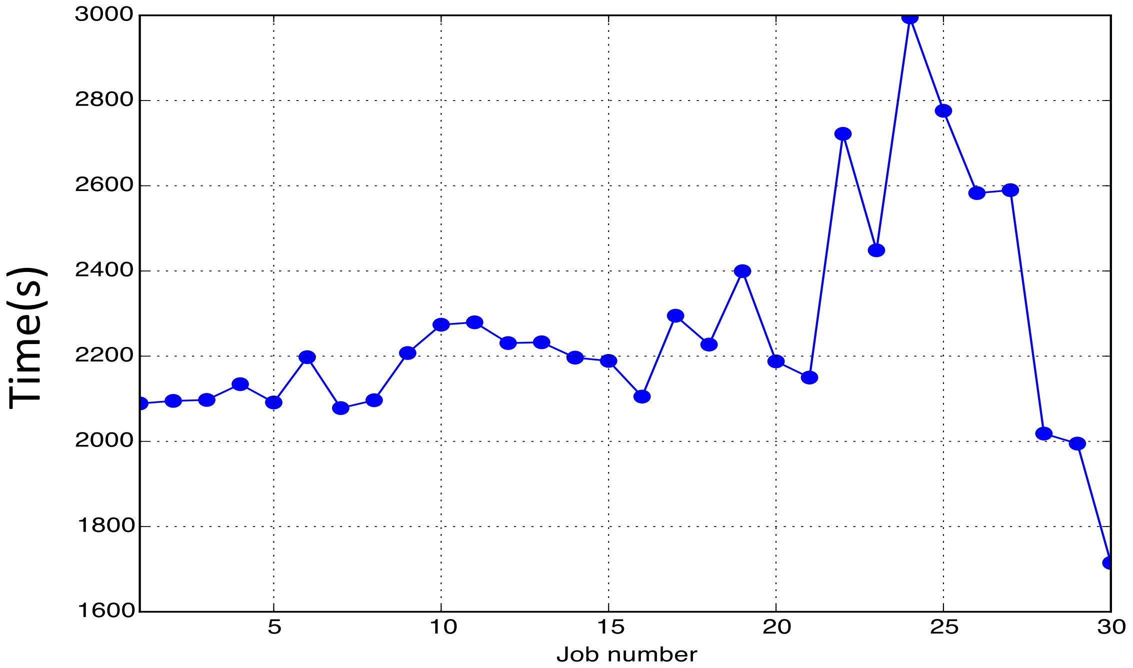
MILC variability at the network level

- **MILC**
  - MIMD Lattice Computation QCD Code simulating 4D SU(3) lattice gauge theory
  - Performs large scale numerical simulations to study quantum chromodynamics (QCD)
  - Compute intensive per one lattice site with low memory footprint per compute node

- Job-to-job variability:
  - **74%** on 128 node jobs on Theta
  - **41%** on 256 node jobs on Theta

- Higher the time has a corresponding higher time in the communication (MPI) part – Cray PAT MPI profiling

Job to job ratio = **74.6%**



128 nodes on Theta

Impact of Variability on Performance Tuning

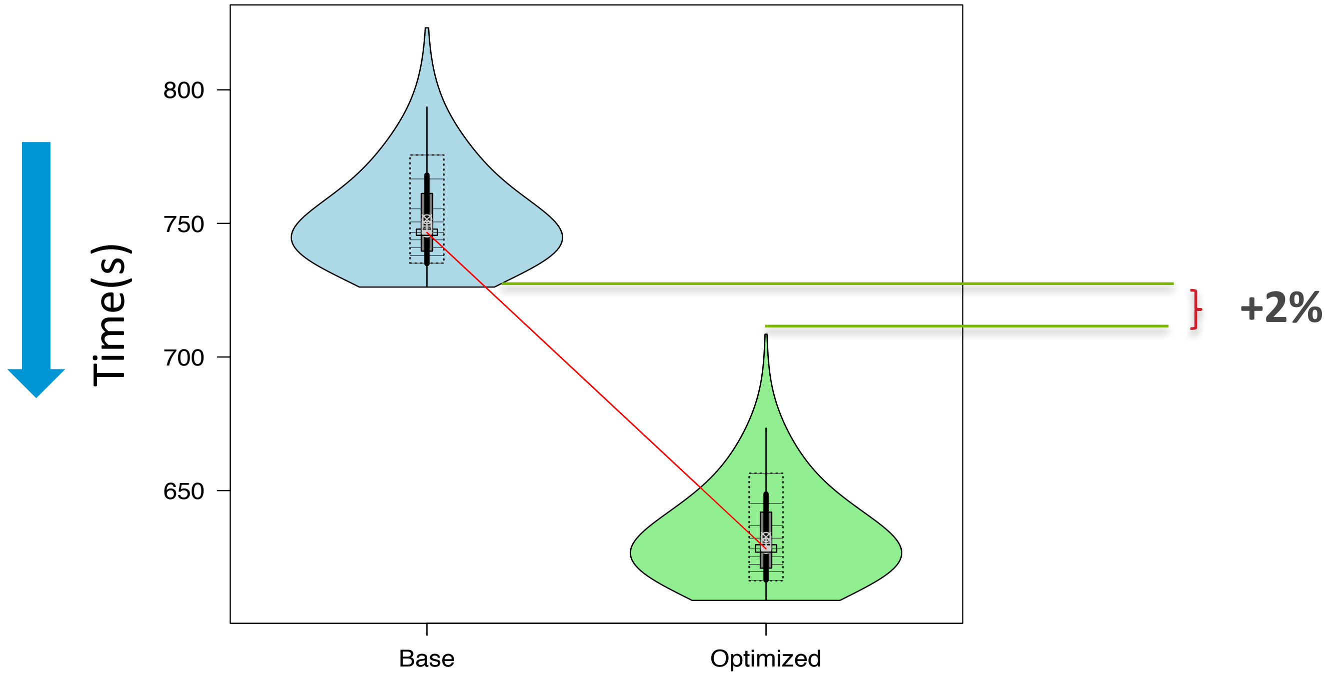
Nekbone:

Optimization: **libxsmm** to optimize small matmul

Impact of optimization in Flat mode: 20.7% (no variability)

Cache mode Avg. performance improvement: 18.8%(95%CI)

- Variability: ~10%
- Performance improvement range [+2% +35%]



Argonne NATIONAL LABORATORY

Impact of Variability on Performance Tuning

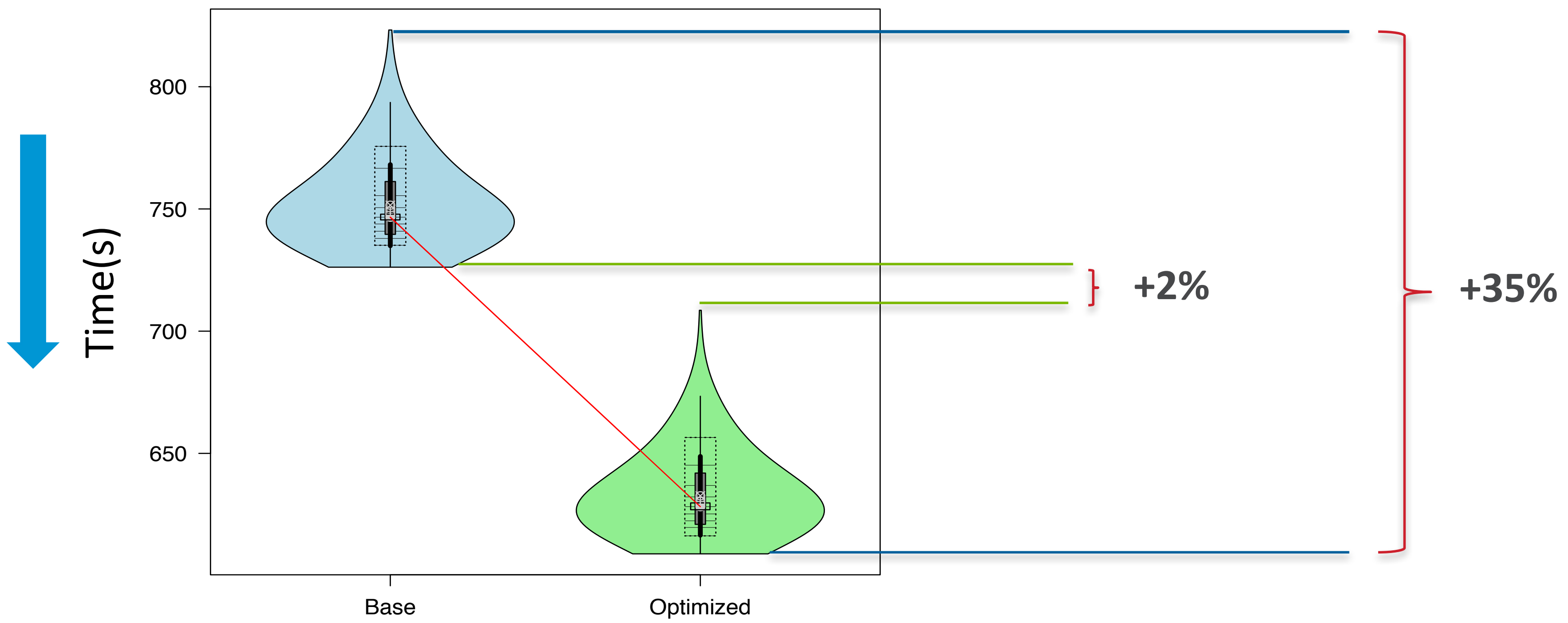
Nekbone:

Optimization: **libxsmm** to optimize small matmul

Impact of optimization in Flat mode: 20.7% (no variability)

Cache mode Avg. performance improvement: 18.8%(95%CI)

- Variability: ~10%
- Performance improvement range [+2% +35%]



Impact of Variability on Performance Tuning

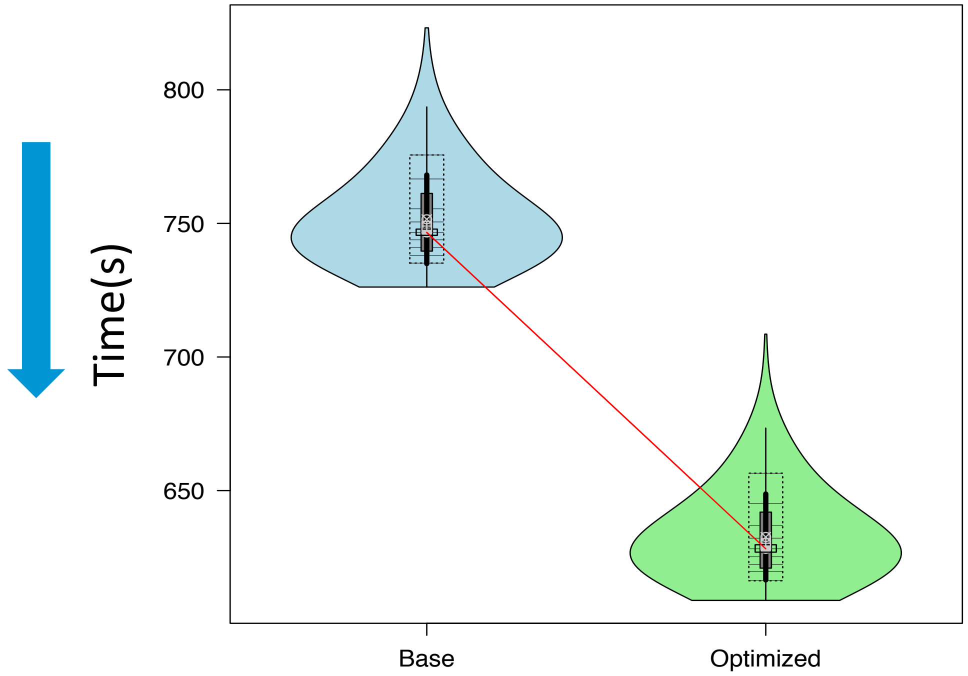
Nekbone:

Optimization: **libxsmm** to optimize small matmul

Impact of optimization in Flat mode: 20.7% (no variability)

Cache mode Avg. performance improvement: 18.8%(95%CI)

- Variability: ~10%
- Performance improvement range [+2% +35%]



MILC:

Optimization: **Rank reorder** to minimize inter-node traffic

Impact of Optimization in less variable environment: **22%**

Impact of Variability on Performance Tuning

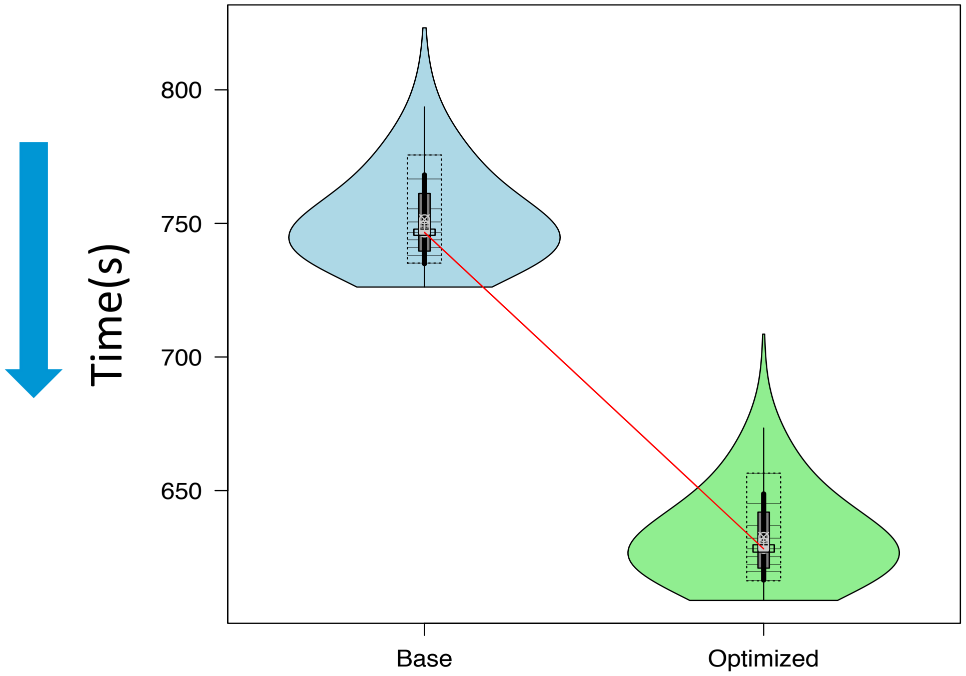
Nekbone:

Optimization: **libxsmm** to optimize small matmul

Impact of optimization in Flat mode: 20.7% (no variability)

Cache mode Avg. performance improvement: 18.8%(95%CI)

- Variability: ~10%
- Performance improvement range [+2% +35%]



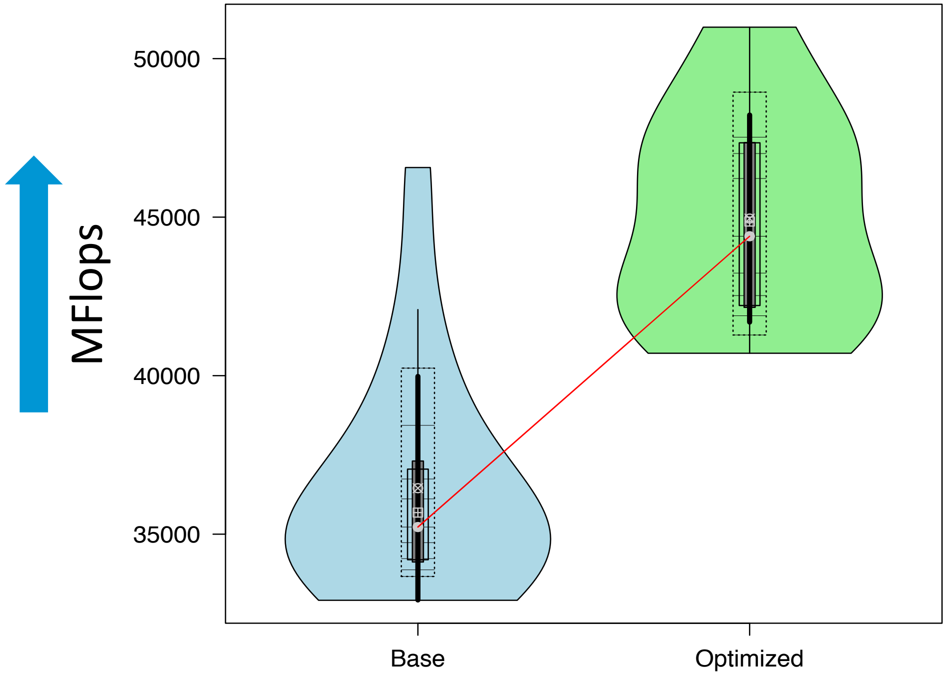
MILC:

Optimization: **Rank reorder** to minimize inter-node traffic

Impact of Optimization in less variable environment: **22%**

Production mode Avg. performance improvement: **23.3%**

- Variability: 25% in Opt. case & 41% in base case
- Performance improvement range **[-14% +55%]**



Conclusions

- Classified and quantified sources of variability on Xeon Phi based Cray XC
  - Core level variability due to OS noise
    - Available mitigations: Use core spec (mechanism to reduce OS noise), exclude tile 0 & 32
  - Memory mode variability due to cache mode page conflicts
    - Available mitigations: run in flat mode
    - Potential mitigations: improved zone sort (part of Cray software stack)
  - Network variability due to shared network resources
    - Available mitigations: run without other jobs present on system
    - Potential mitigations: A compact job placement with static routing
- Characterized impact on the Applications – up to 70% for MILC; up to 35% for Nekbone
- Guidelines on performance tuning in the presence of variability:
  - Be aware of the network level congestion that does not have a clear mitigation strategy, this could potentially influence the communication intensive applications.
  - Incorporate statistical analysis in the performance benchmarking and analysis (refer https://htor.inf.ethz.ch/publications/img/hoefler-scientific-benchmarking.pdf for more details on statistics)

Conclusions

Questions?

- Classified and quantified sources of variability on Xeon Phi based Cray XC
  - Core level variability due to OS noise
    - Available mitigations: Use core spec (mechanism to reduce OS noise), exclude tile 0 & 32
  - Memory mode variability due to cache mode page conflicts
    - Available mitigations: run in flat mode
    - Potential mitigations: improved zone sort (part of Cray software stack)
  - Network variability due to shared network resources
    - Available mitigations: run without other jobs present on system
    - Potential mitigations: A compact job placement with static routing
- Characterized impact on the Applications – up to 70% for MILC; up to 35% for Nekbone
- Guidelines on performance tuning in the presence of variability:
  - Be aware of the network level congestion that does not have a clear mitigation strategy, this could potentially influence the communication intensive applications.
  - Incorporate statistical analysis in the performance benchmarking and analysis (refer https://htor.inf.ethz.ch/publications/img/hoefler-scientific-benchmarking.pdf for more details on statistics)