TECHNISCHE UNIVERSITÄT DRESDEN

ZIB

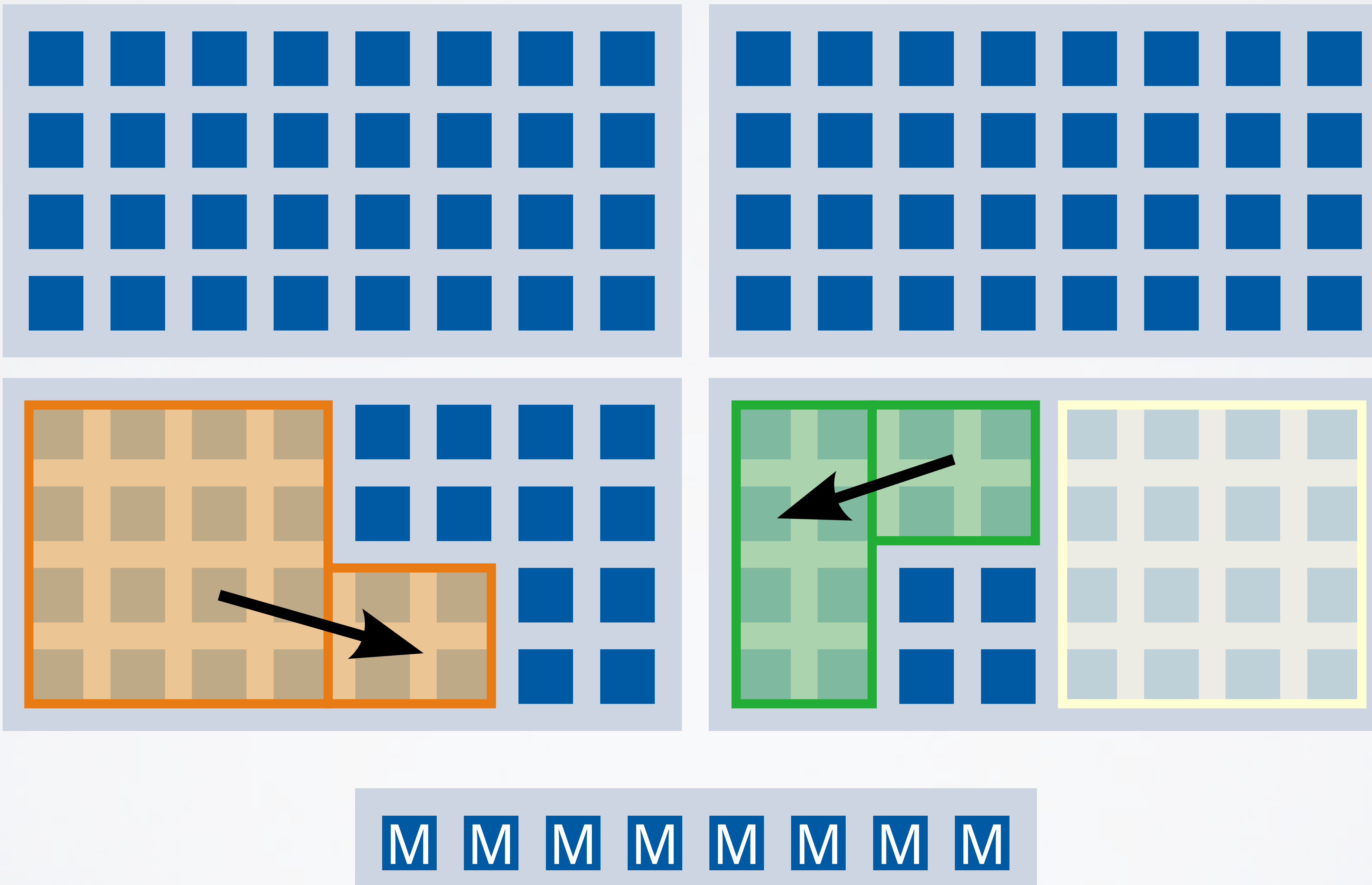
The Hebrew University of Jerusalem

FFMK: A FAST AND FAULT-TOLERANT MICROKERNEL-BASED SYSTEM FOR EXASCALE COMPUTING

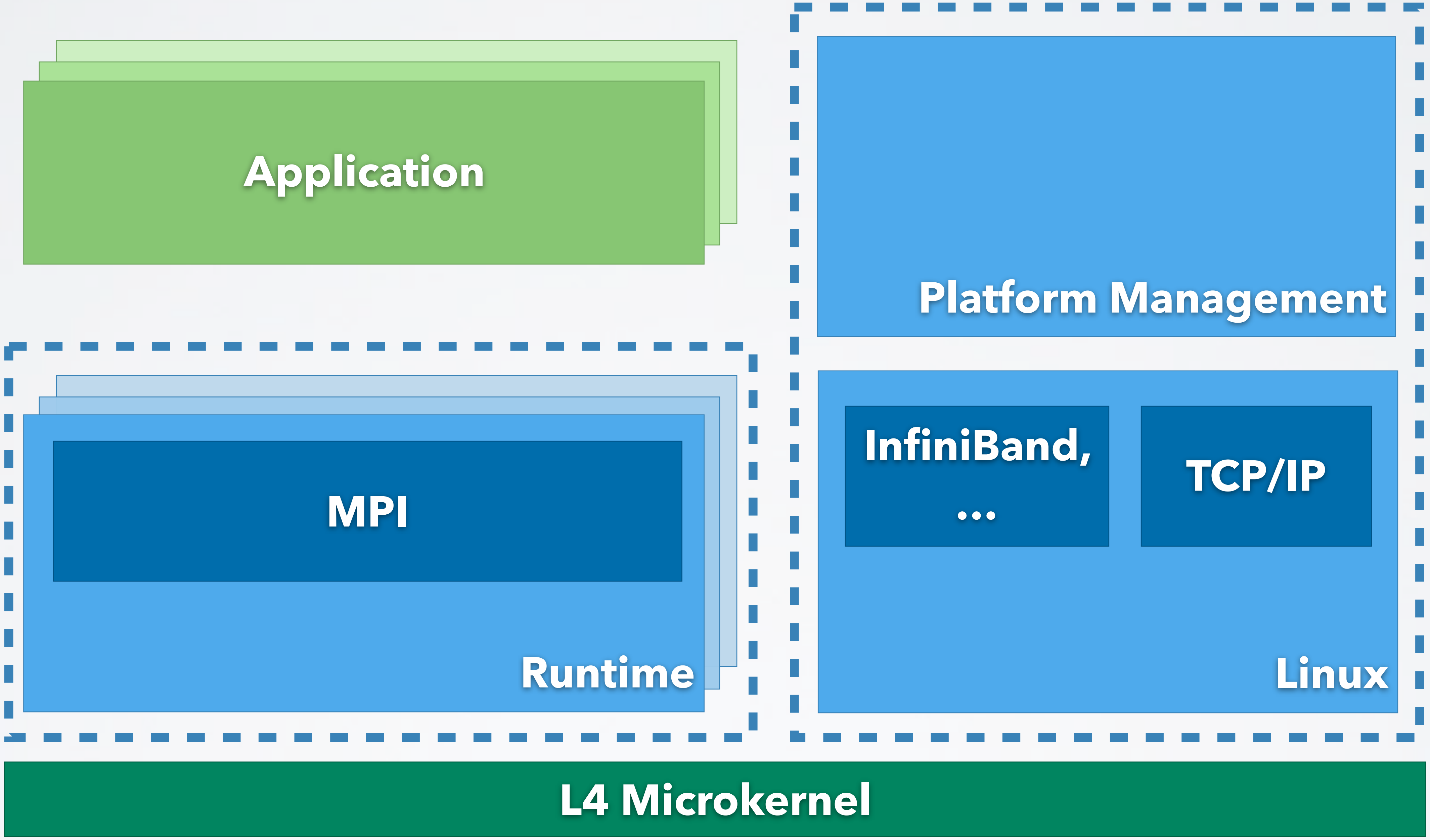
Amnon Barak	Hebrew University Jerusalem (HUJI)
Hermann Härtig	TU Dresden, Operating Systems Group (TUDOS)
Wolfgang E. Nagel	TU Dresden, Center for Information Services and HPC (ZIH)
Alexander Reinefeld	Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)

CARSTEN WEINHOLD, TU DRESDEN

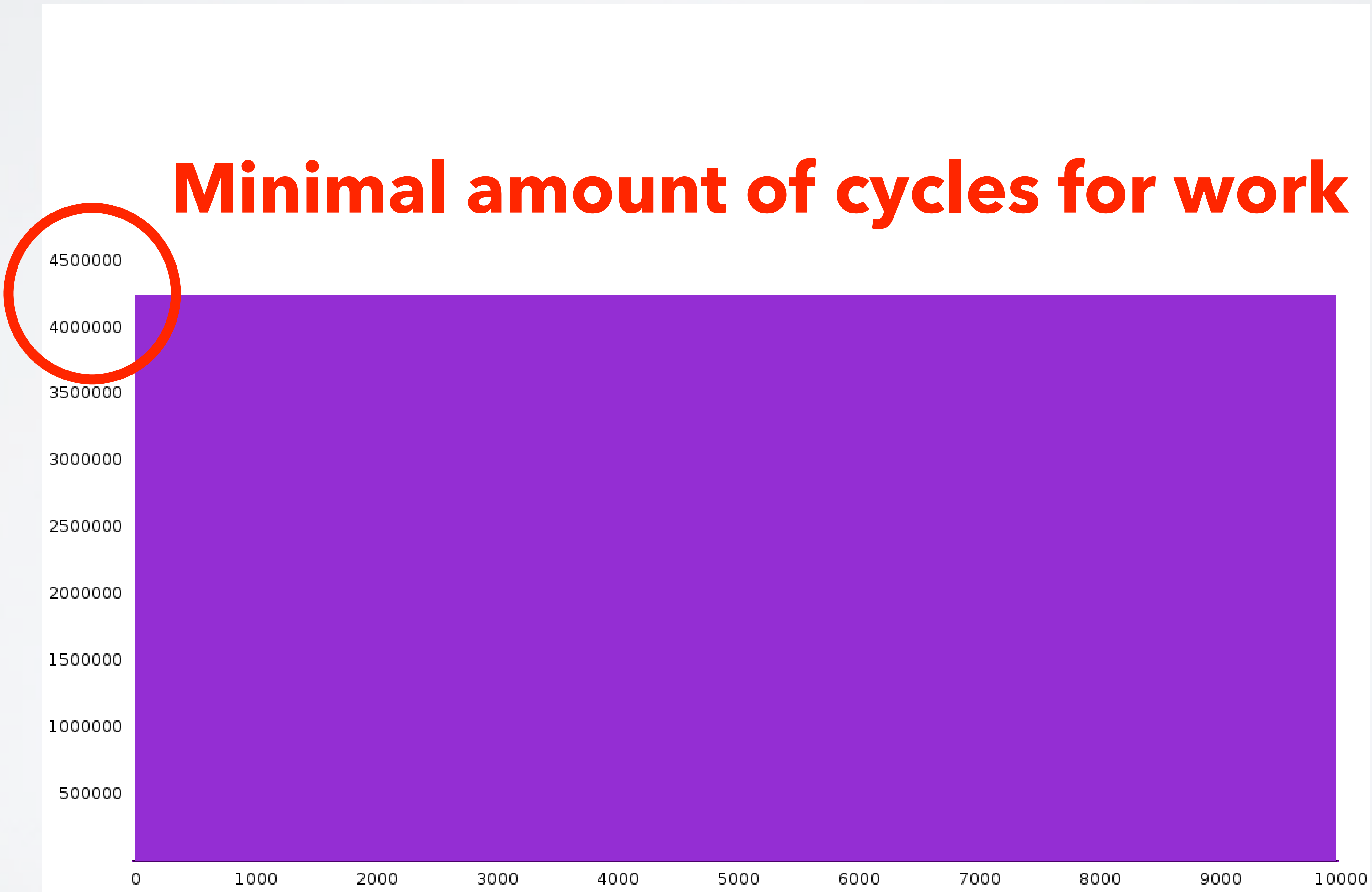
SYSTEM MODEL



NODE ARCHITECTURE

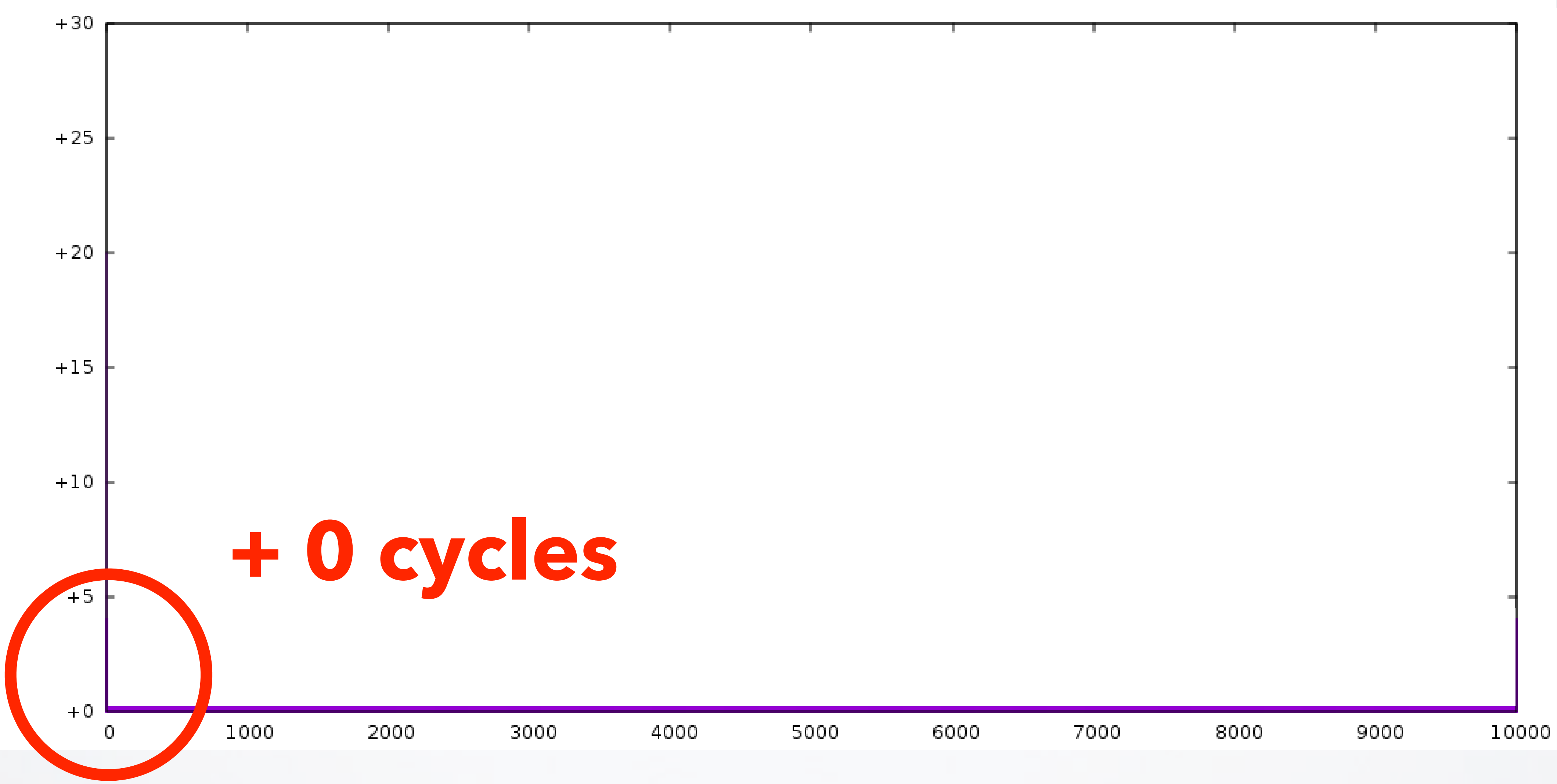


FWQ BENCHMARK



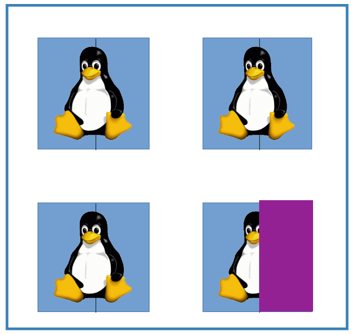
NOISY LINUX?

Ideal: zero extra cycles

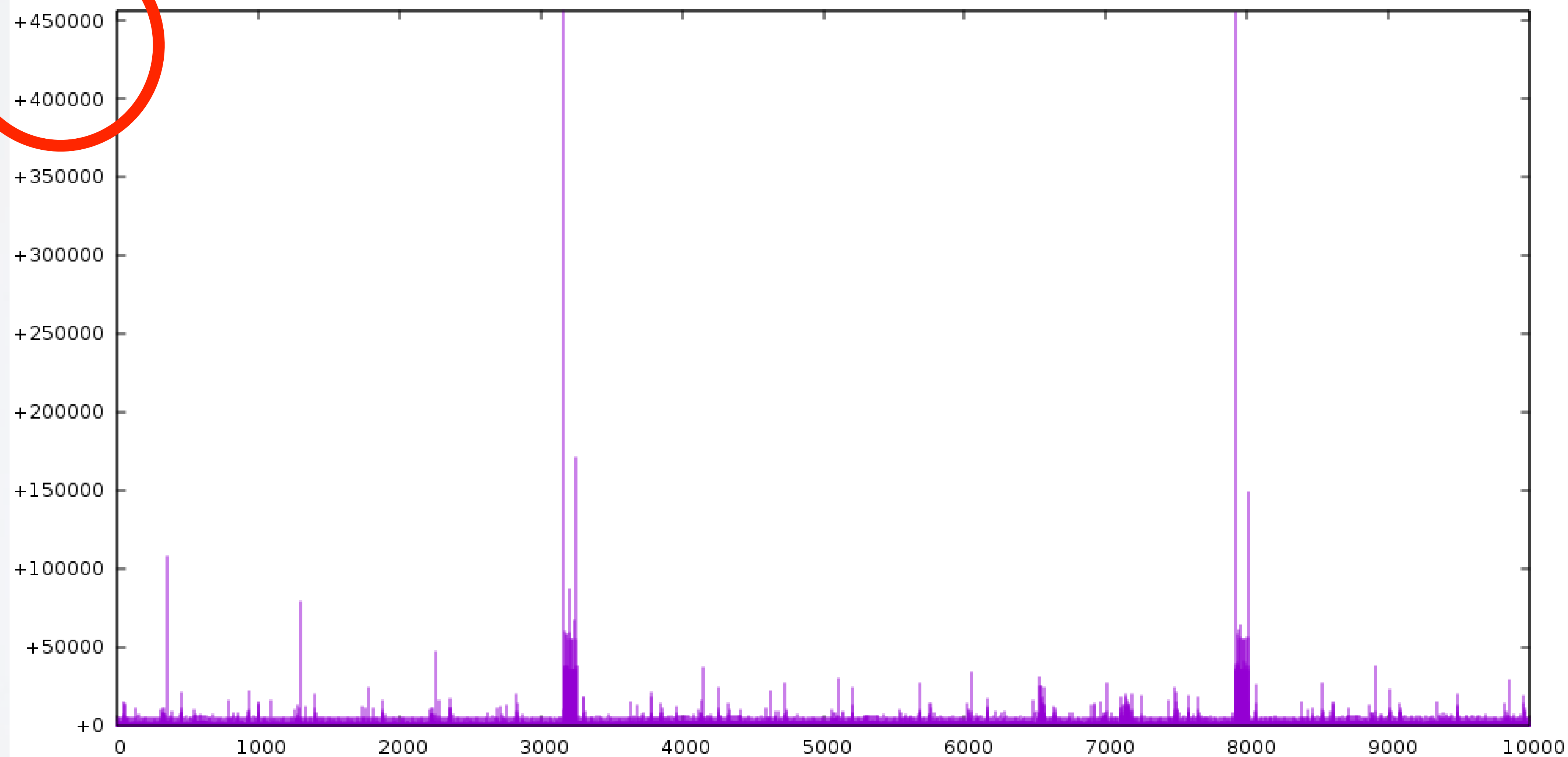


NOISY LINUX?

Real-World HPC Linux

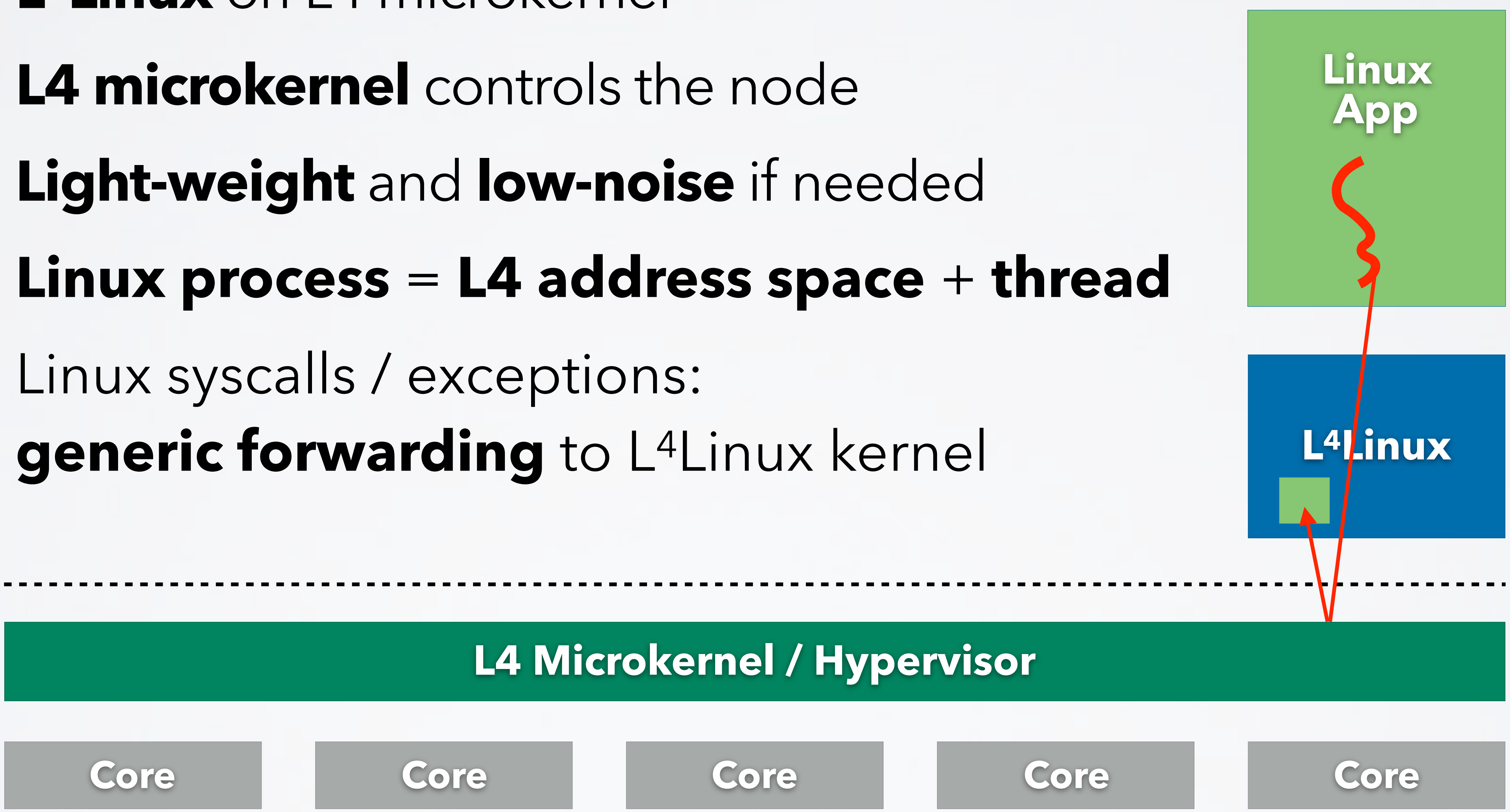


+450,000 cycles ≈ 10%



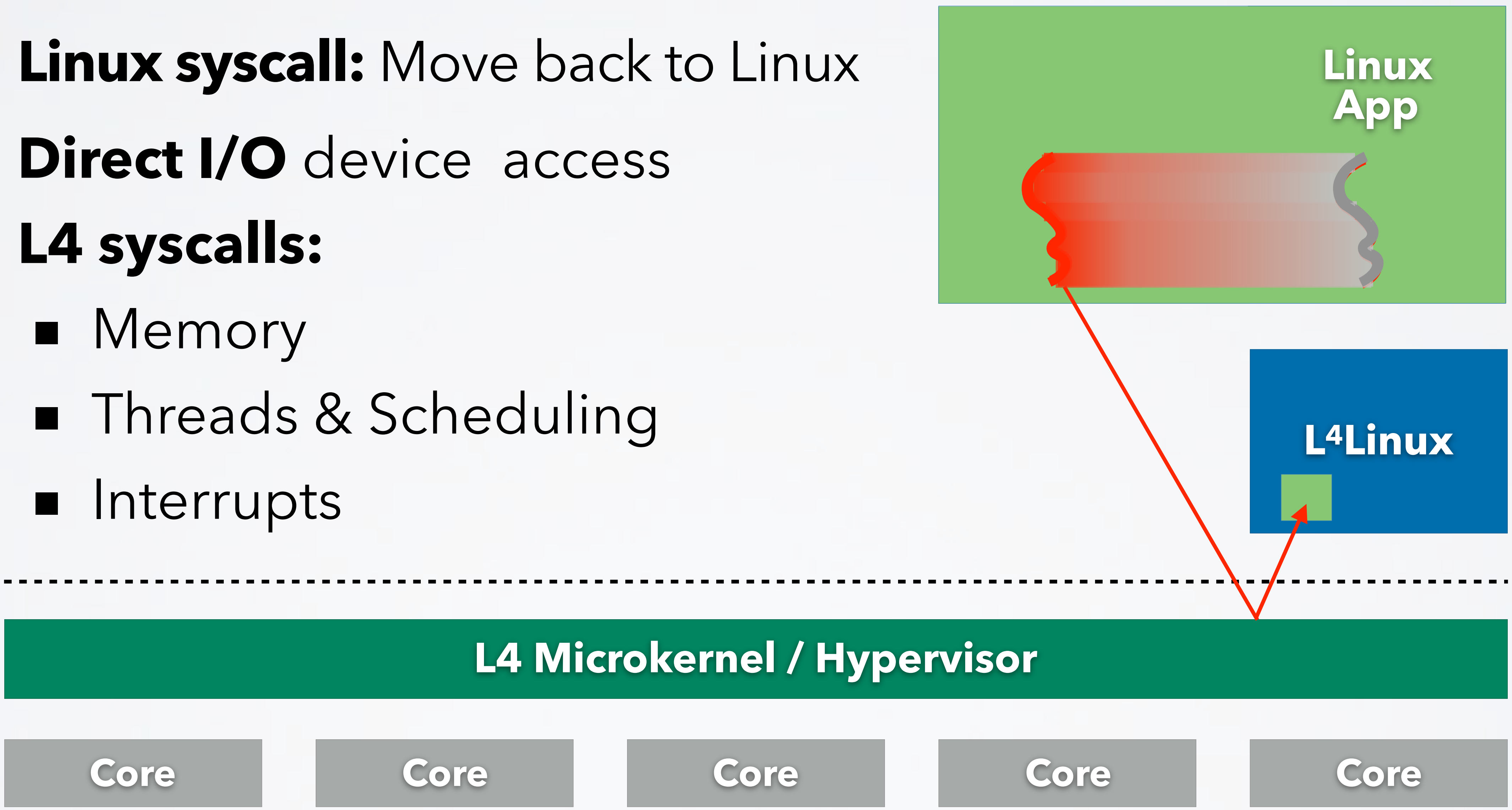
L4 + L^{4}LINUX

- **Unmodified** Linux programs (MPI, …)
- **L^{4}Linux** on L4 microkernel
- **L4 microkernel** controls the node
- **Light-weight** and **low-noise** if needed
- **Linux process** = **L4 address space** + **thread**
- Linux syscalls / exceptions:
  generic forwarding to L^{4}Linux kernel



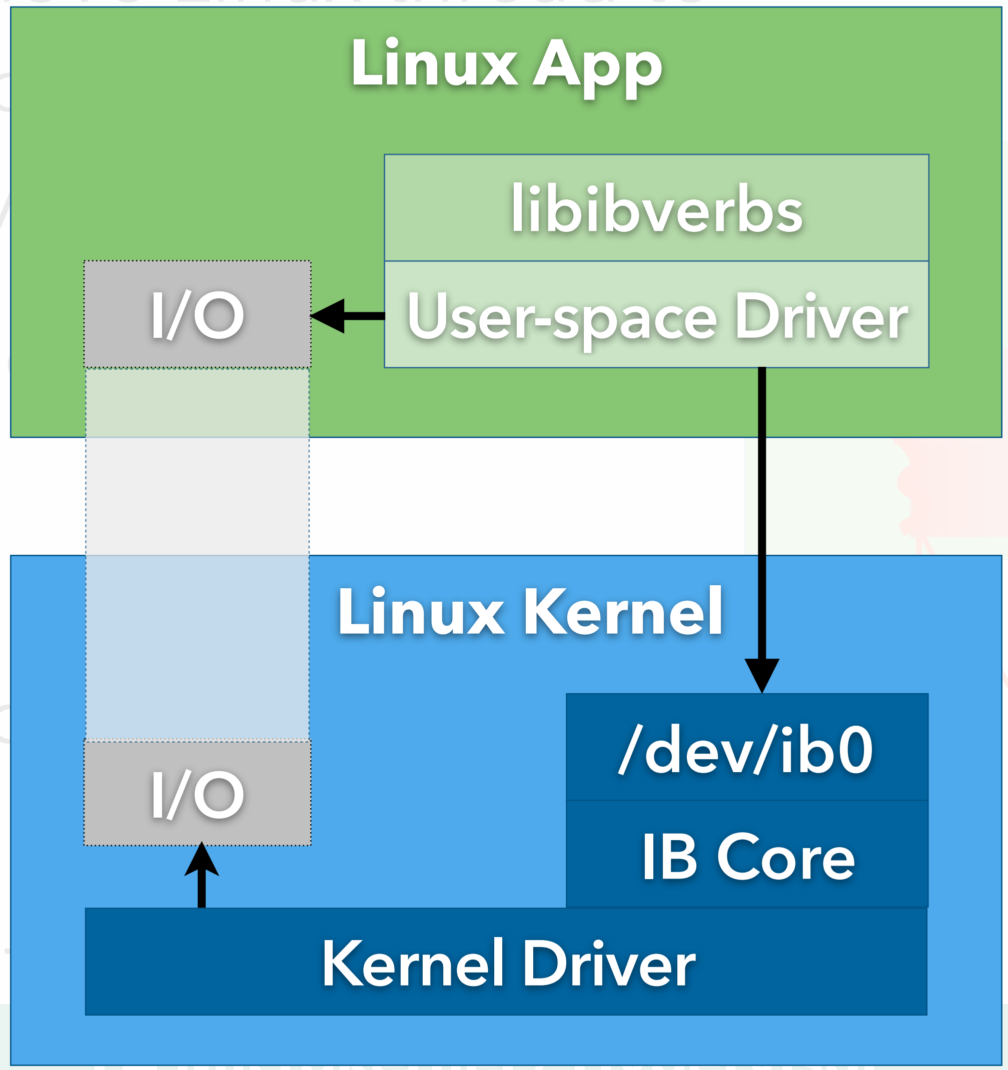
DECOUPLED THREADS

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux
- **Direct I/O** device access
- **L4 syscalls:**
  - Memory
  - Threads & Scheduling
  - Interrupts



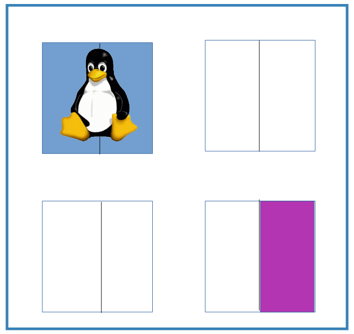
DECOUPLED THREADS

- **Decoupling:** move Linux thread to new L4 thread c
- **Linux syscall:** M
- **Direct I/O** devi
- **L4 syscalls:**
  - Memory
  - Threads & Sc
  - Interrupts

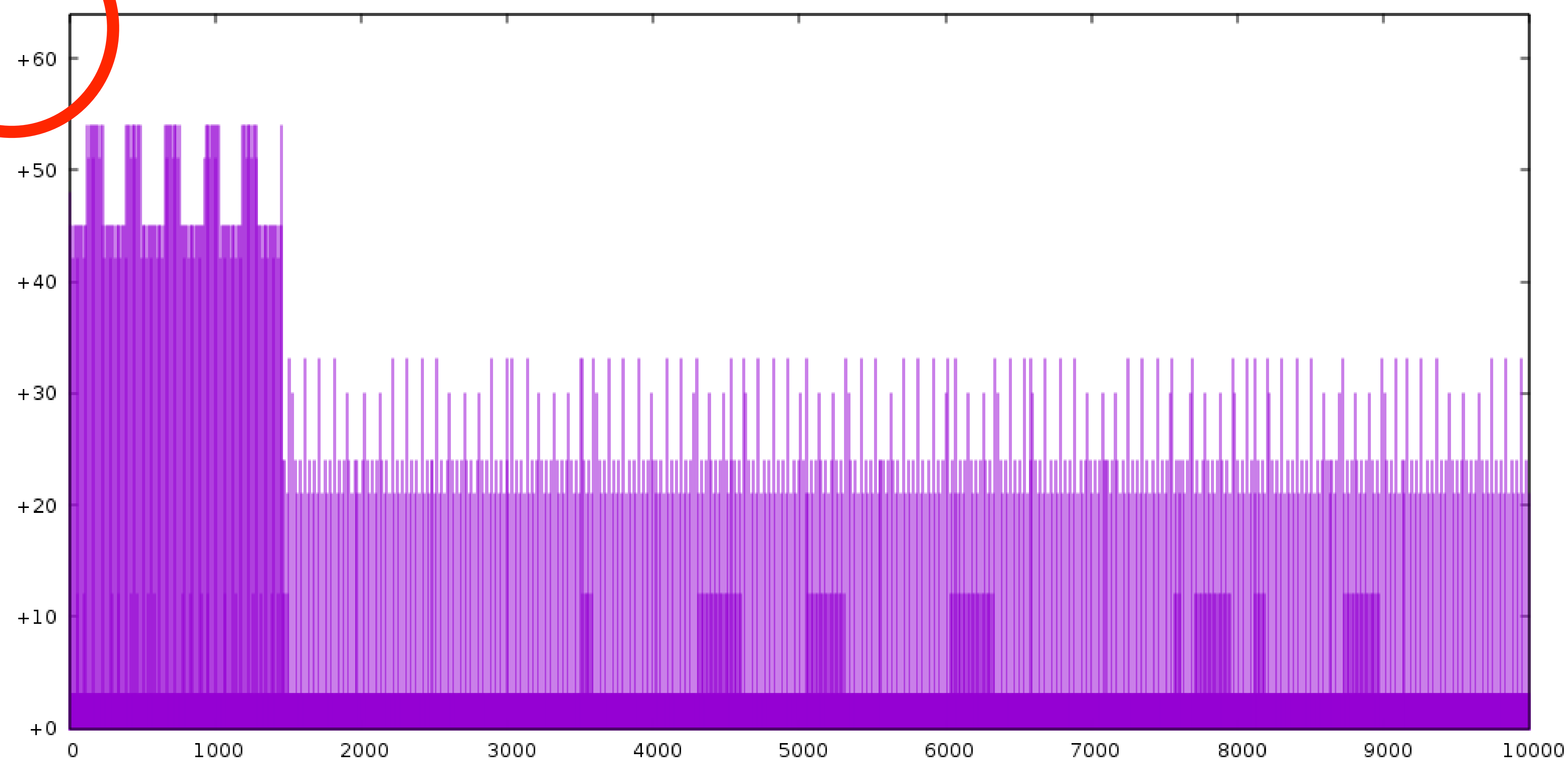


NOISY LINUX?

Decoupled Linux thread

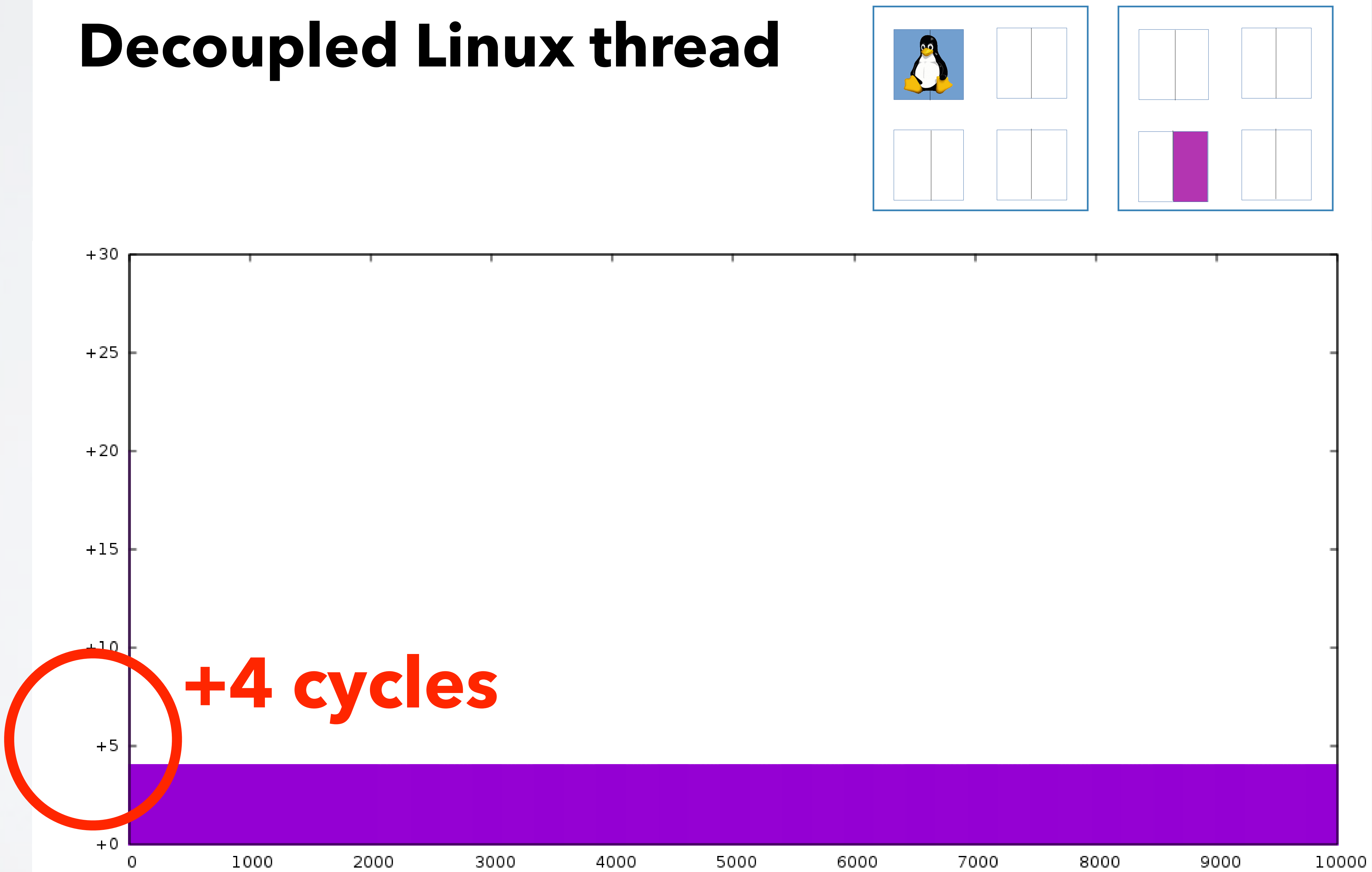


+60 cycles



NOISY LINUX?

Decoupled Linux thread



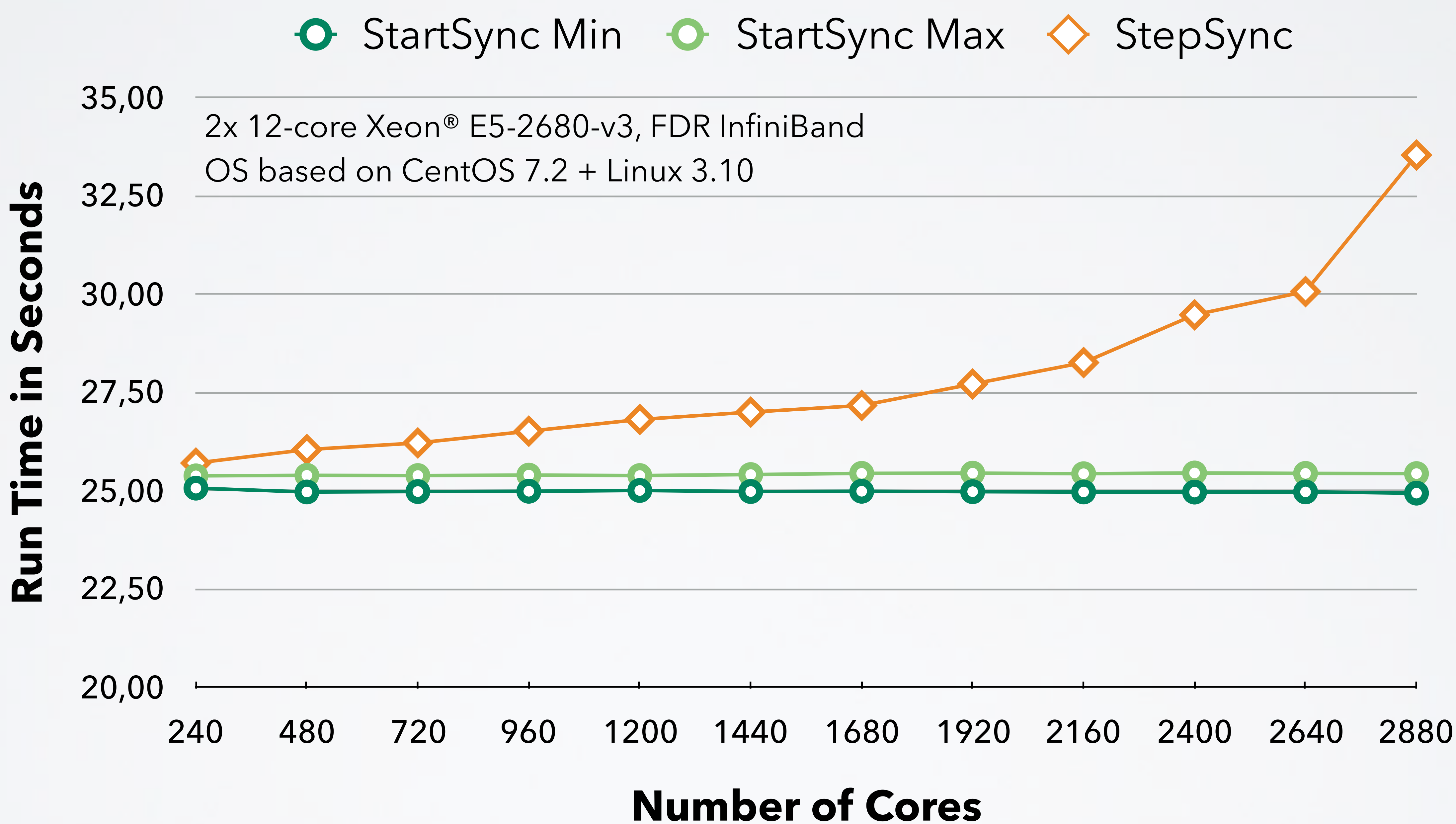
StartSync:

Behavior: **embarrassingly parallel**

StepSync:

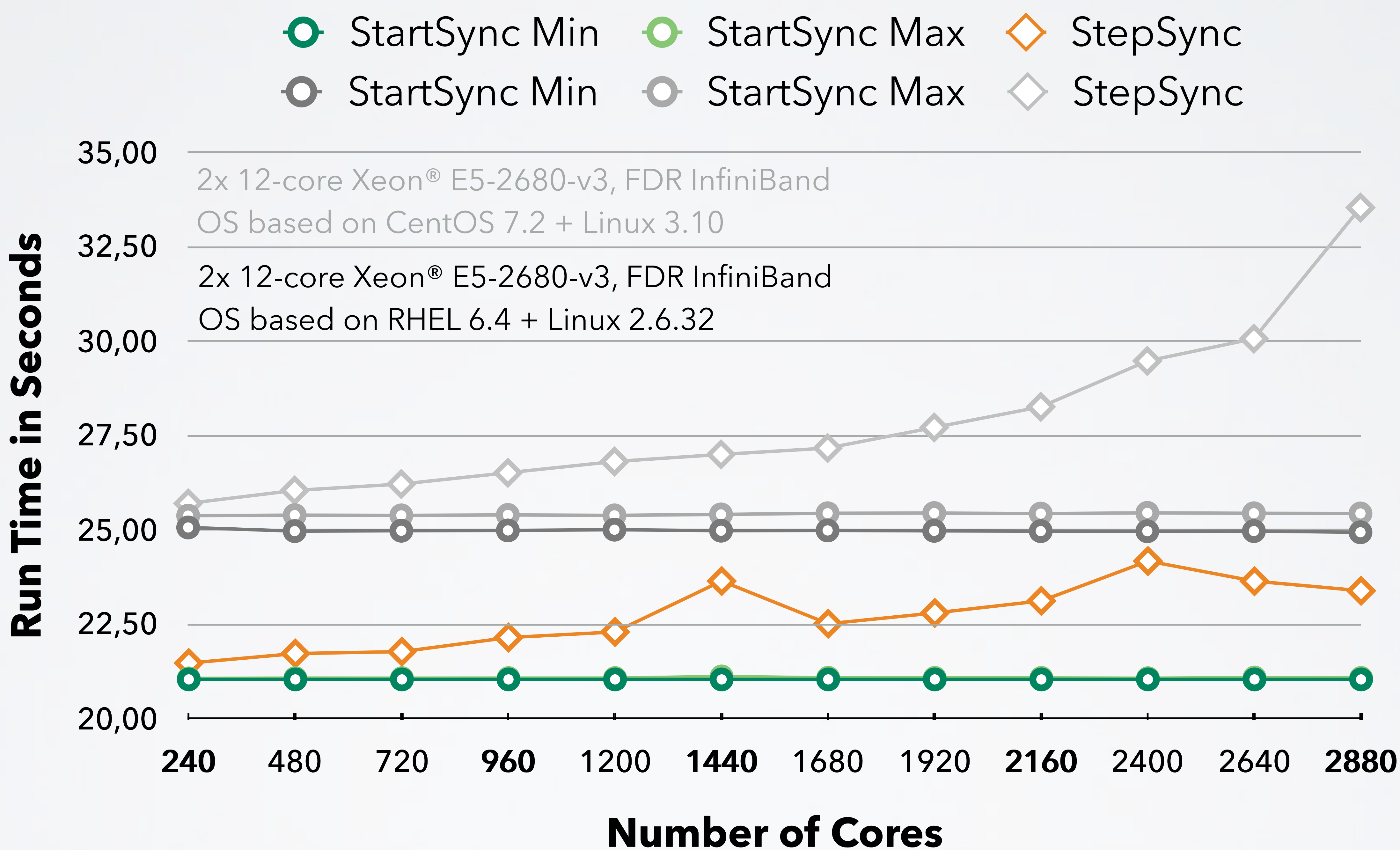
Behavior: **bulk-synchronous**

VENDOR OS [JURECA]



Adam Lackorzynski, Carsten Weinhold, Hermann Härtig, "Decoupled: Low-Effort Noise-Free Execution on Commodity Systems", ROSS 2016, June 2016, Kyoto, Japan

VENDOR OS [TAURUS]

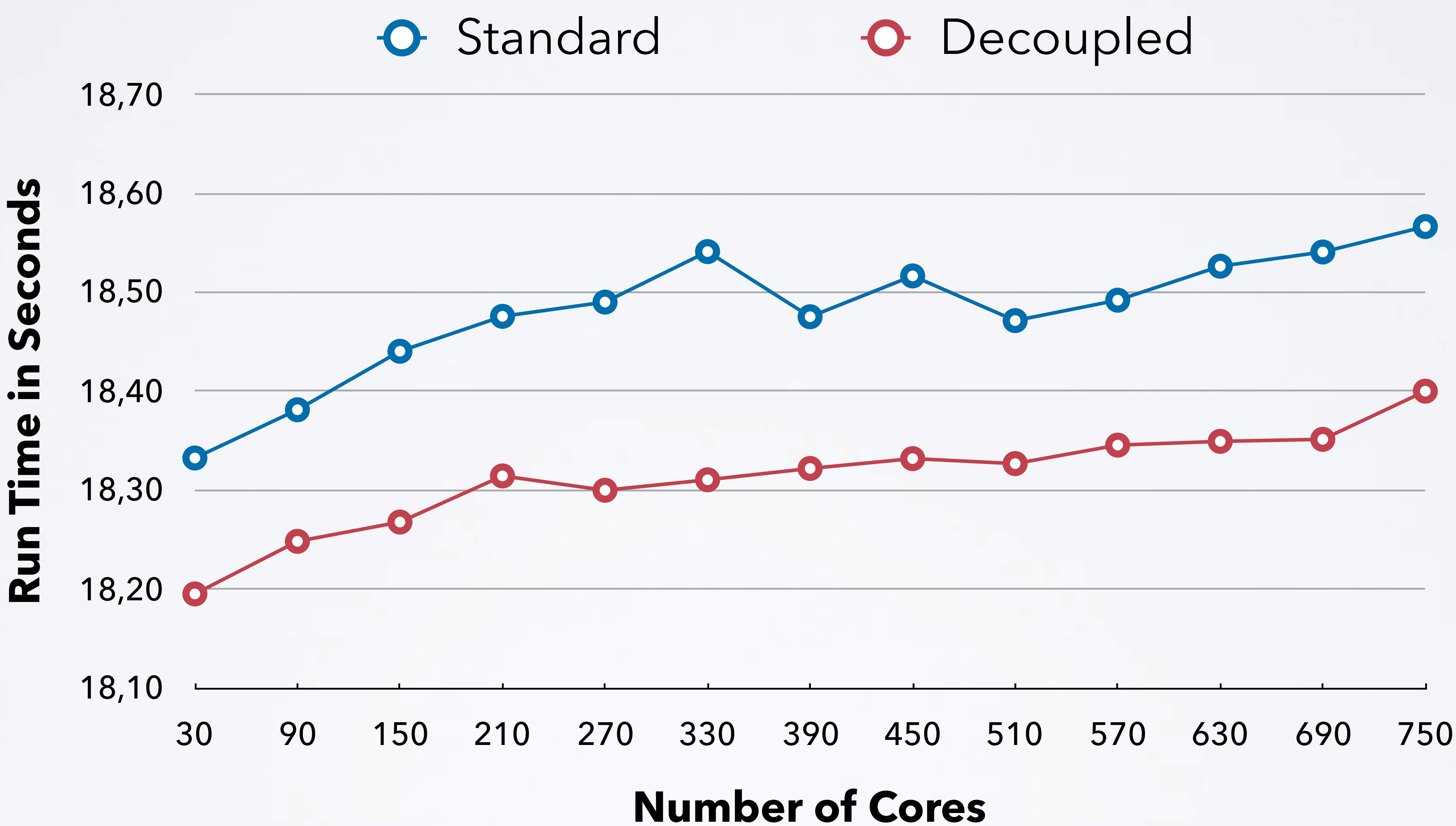


Adam Lackorzynski, Carsten Weinhold, Hermann Härtig, "Decoupled: Low-Effort Noise-Free Execution on Commodity Systems", ROSS 2016, June 2016, Kyoto, Japan

EVALUATION SETUP

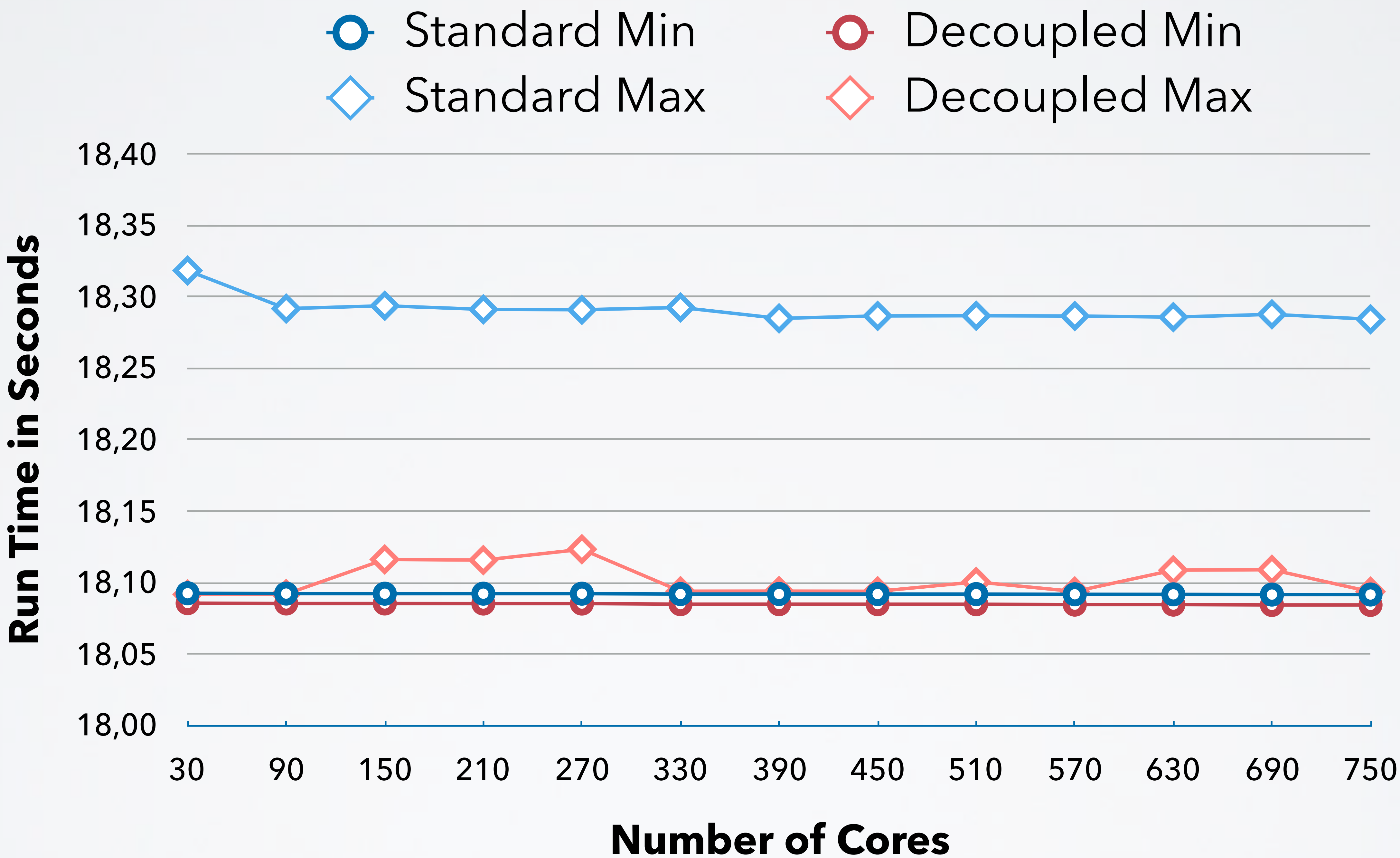
- **Bare-metal access** to Taurus:
  - Little time
  - Fewer cores
  - Different type of nodes
- Vendor OS: **Linux 2.6.32** or **3.10** …
- Decoupled threads: **L^4Linux 4.4**
- Custom Linux distribution

DECOUPLING: BSP



Adam Lackorzynski, Carsten Weinhold, Hermann Härtig, "Decoupled: Low-Effort Noise-Free Execution on Commodity Systems", ROSS 2016, June 2016, Kyoto, Japan

DECOUPLING: EP



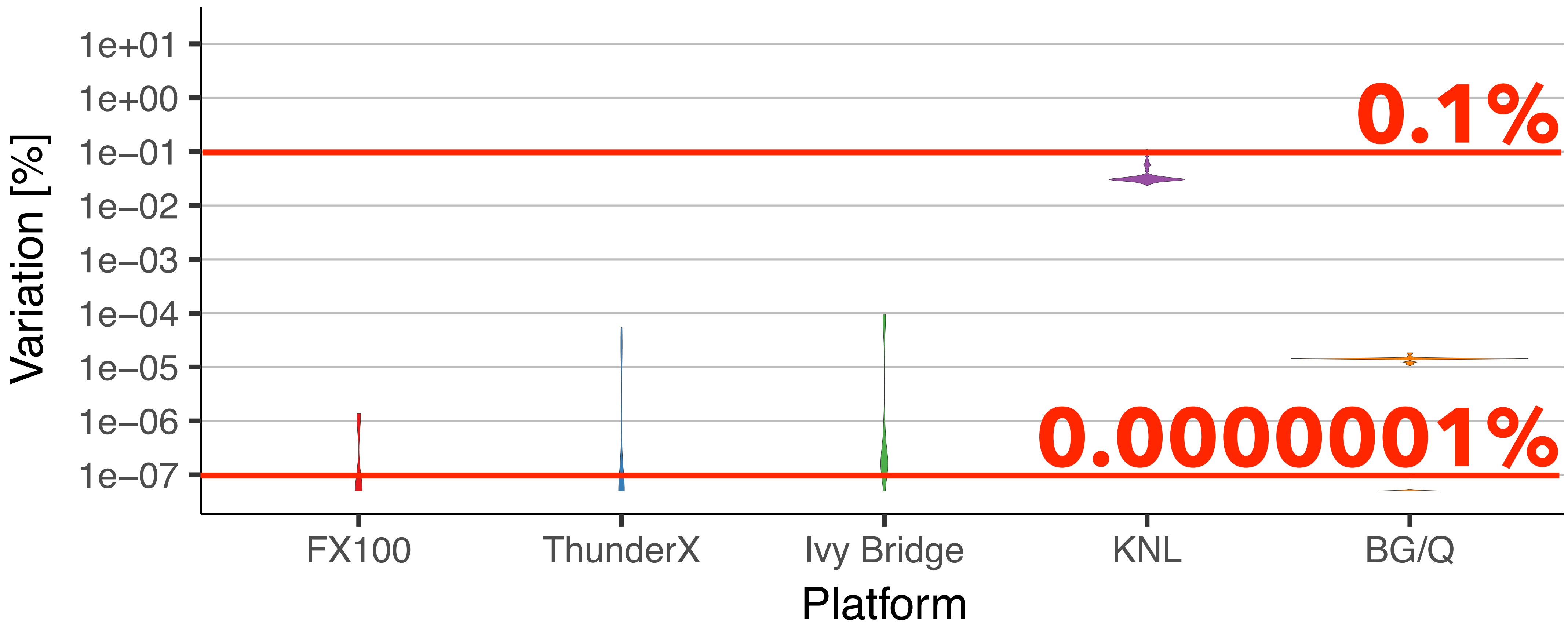
Adam Lackorzynski, Carsten Weinhold, Hermann Härtig, "Decoupled: Low-Effort Noise-Free Execution on Commodity Systems", ROSS 2016, June 2016, Kyoto, Japan

COLLABORATION

- PhD student: internship at RIKEN, Japan
- Comparative study:
  - Hardware performance variation
  - 5 different CPU architectures
  - Light-weight kernel (McKernel)

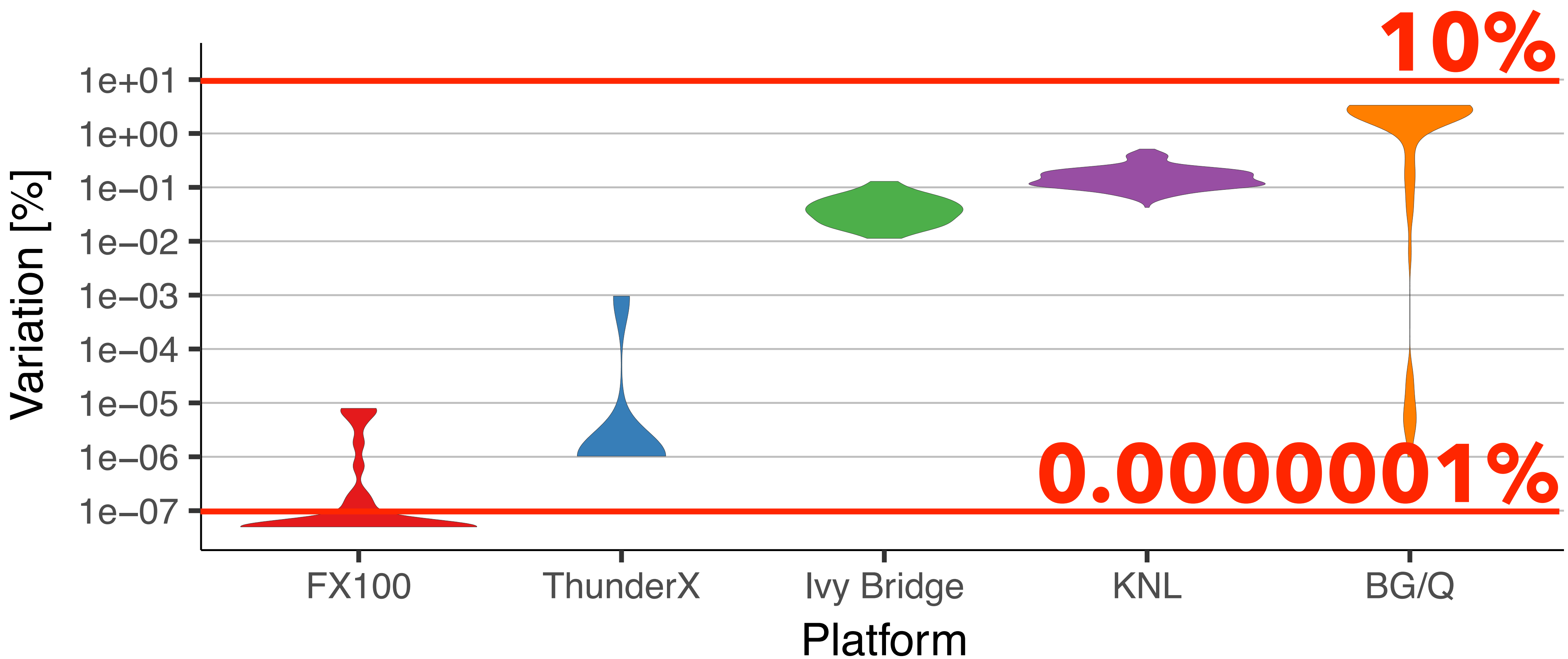
Hannes Weisbach, Brian Kocoloski, Hermann Härtig, Yutaka Ishikawa, Balazs Gerofi, „Hardware Performance Variation: A Comparative Study using Lightweight Kernels“, ISC’18, Frankfurt, Germany, June 2018

FWQ BENCHMARK



Hannes Weisbach, Brian Kocoloski, Hermann Härtig, Yutaka Ishikawa, Balazs Gerofi, „Hardware Performance Variation: A Comparative Study using Lightweight Kernels", ISC'18, Frankfurt, Germany, June 2018

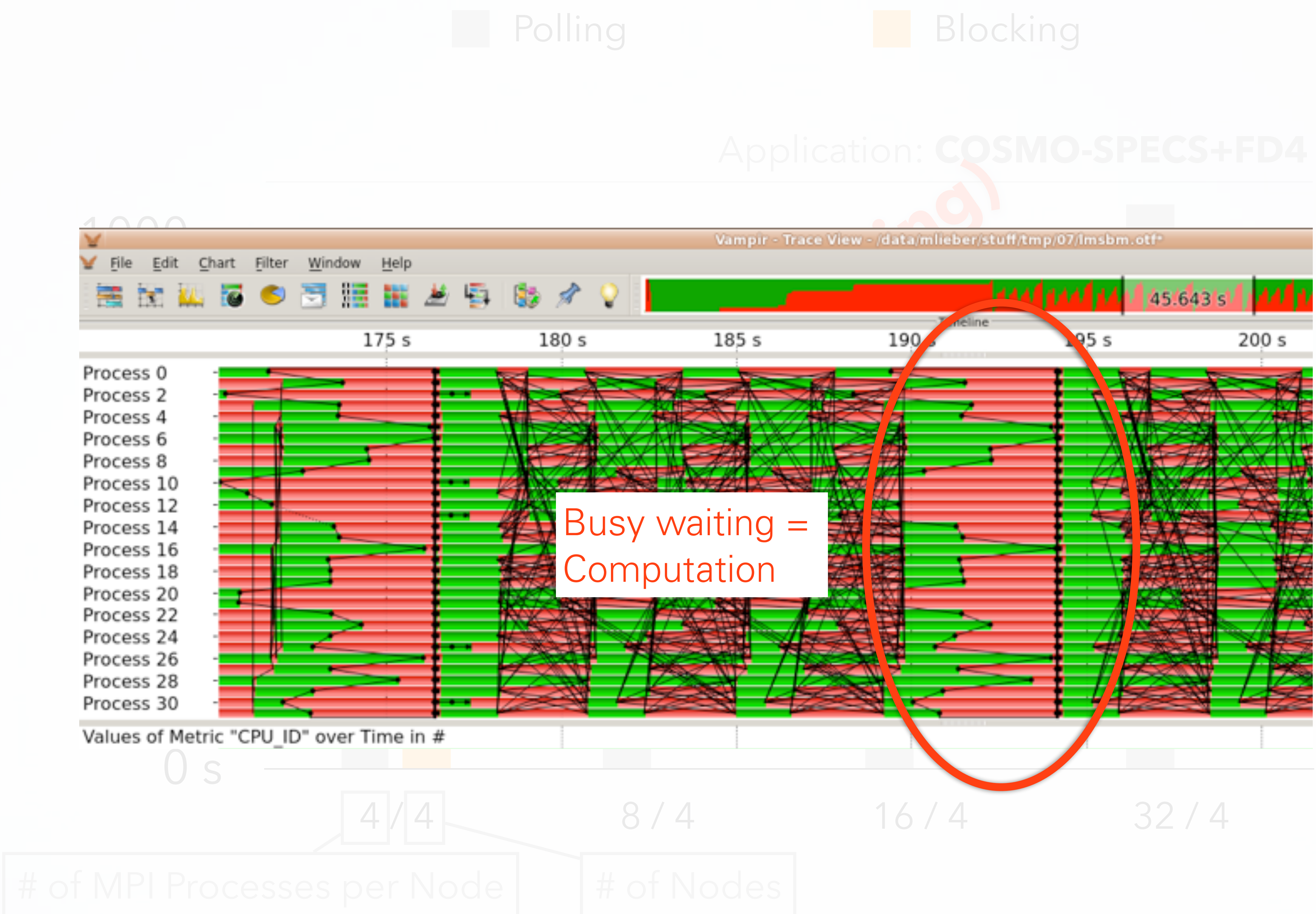
DGEMM BENCHMARK



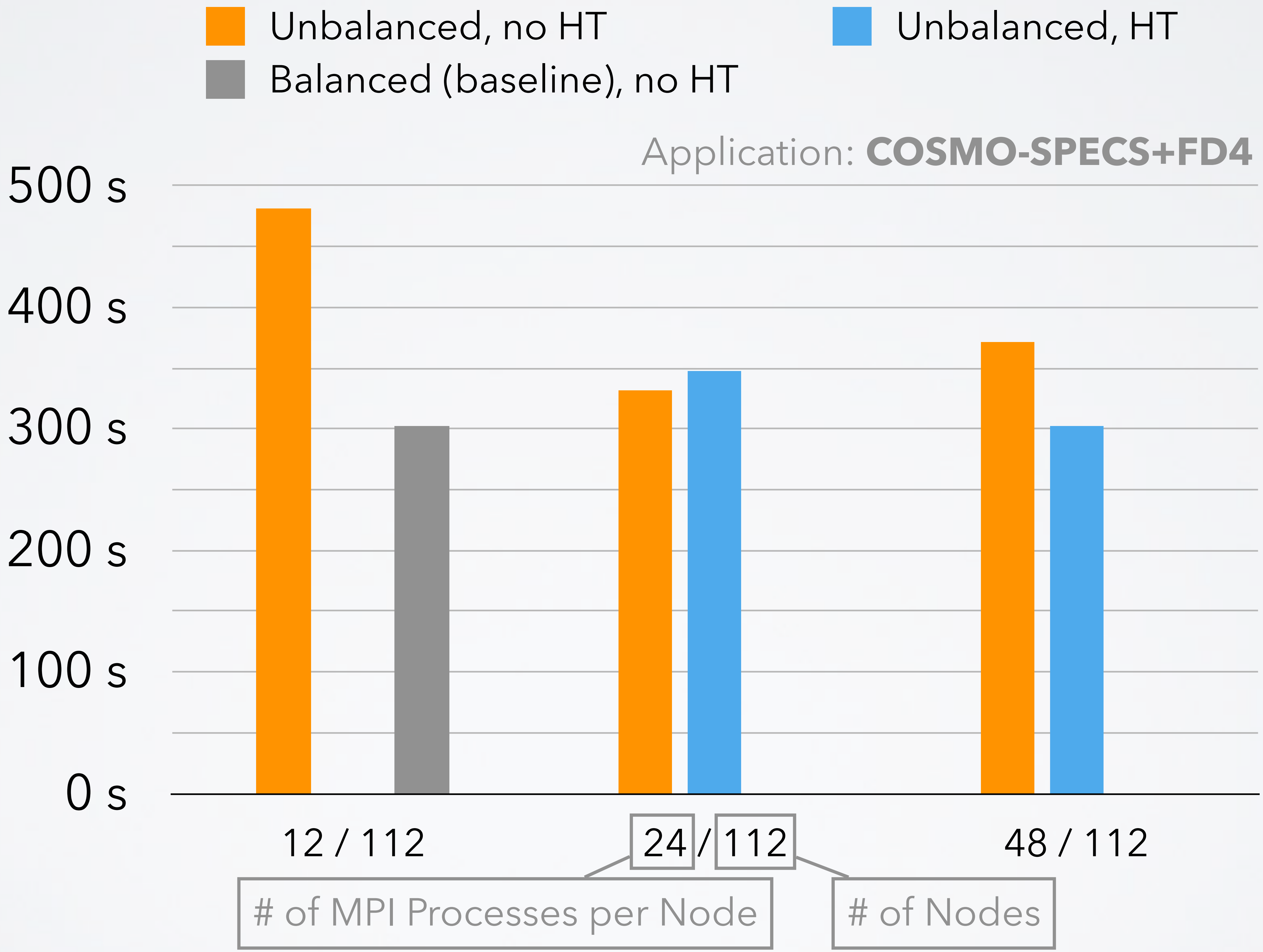
Hannes Weisbach, Brian Kocoloski, Hermann Härtig, Yutaka Ishikawa, Balazs Gerofi, „Hardware Performance Variation: A Comparative Study using Lightweight Kernels“, ISC'18, Frankfurt, Germany, June 2018

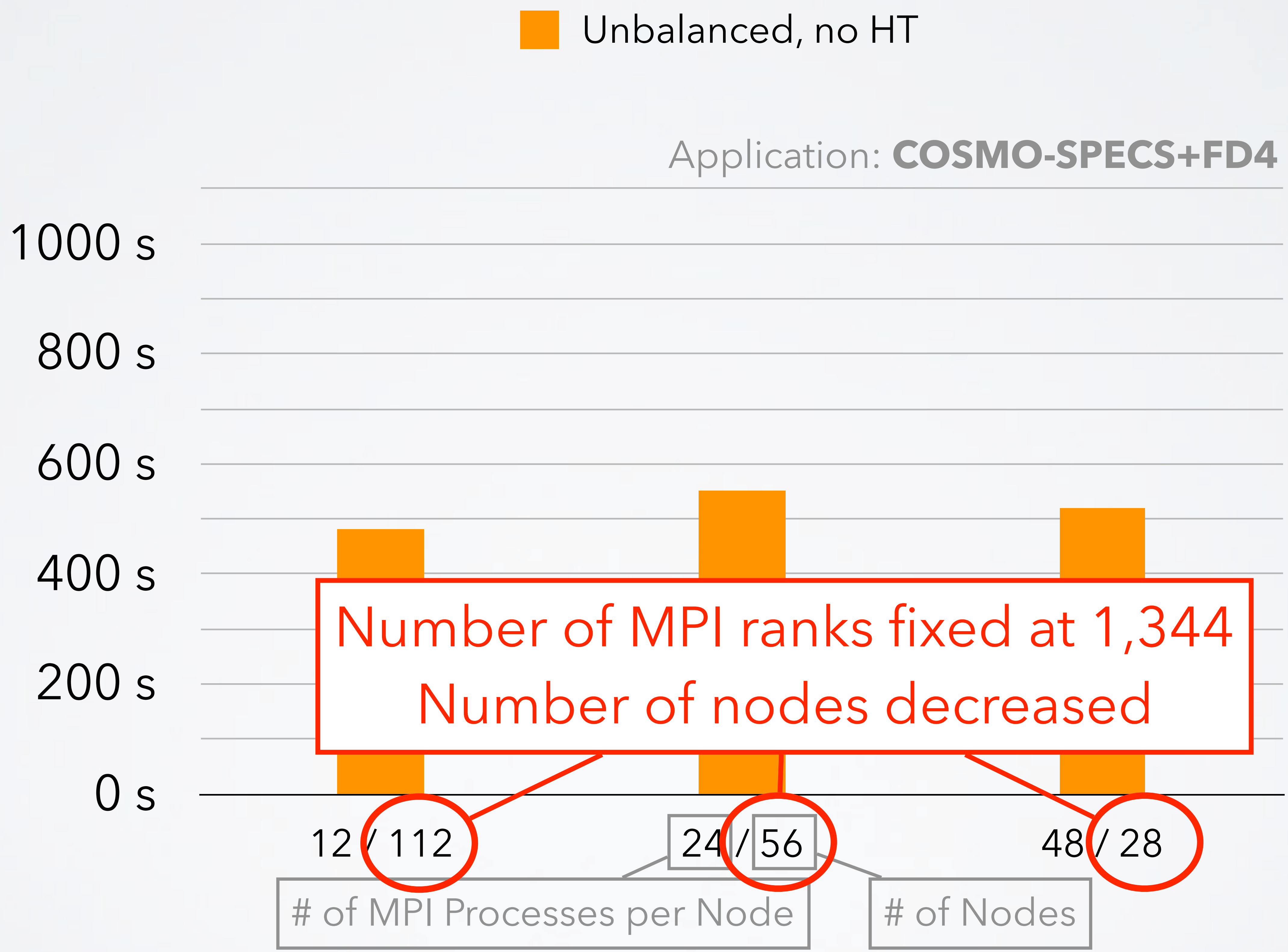
IMBALANCED WORKLOADS

OVERDECOMPOSITION



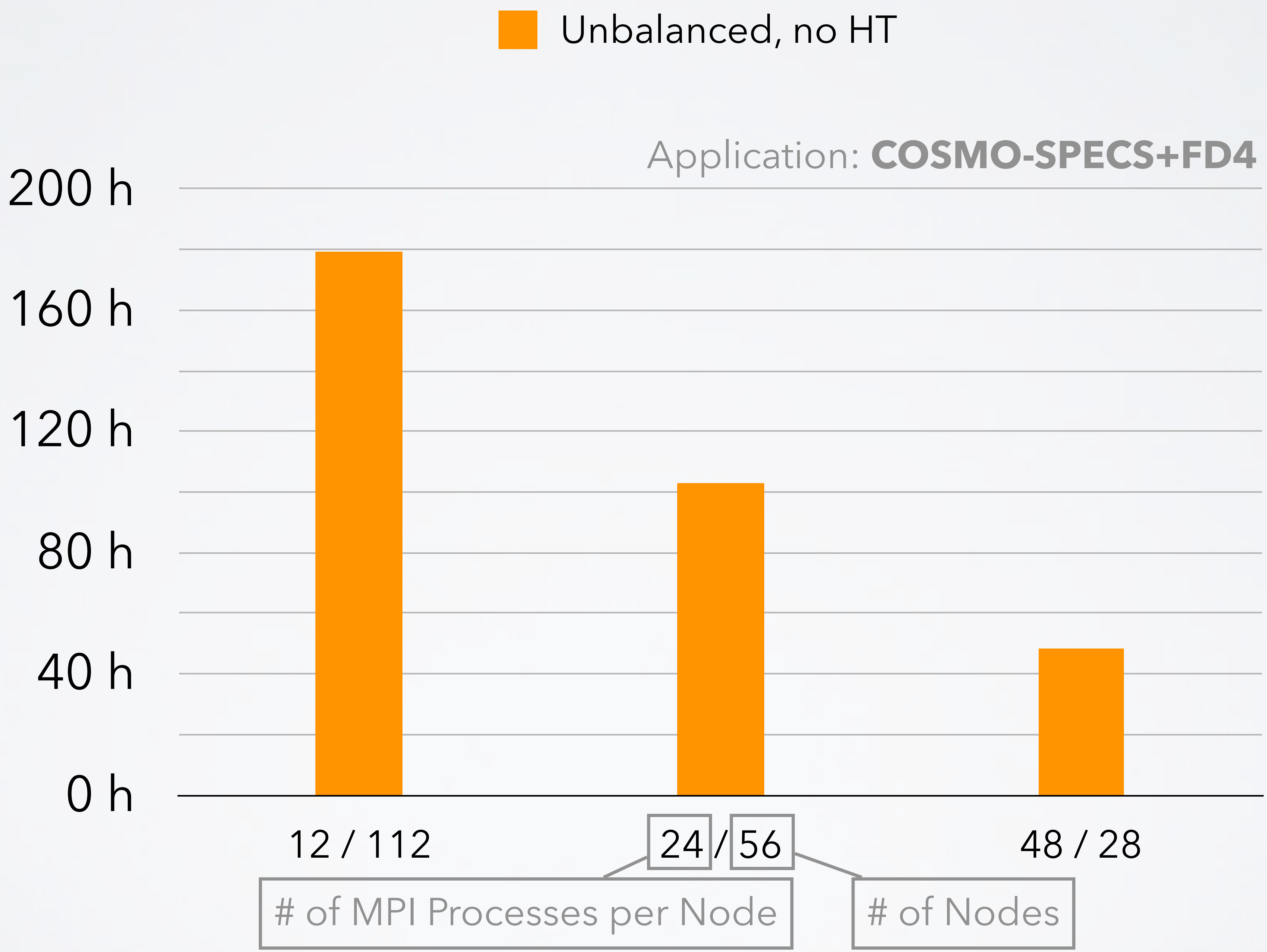
RUN TIME


Unbalanced, no HT
Unbalanced, HT
Balanced (baseline), no HT
Application: **COSMO-SPECS+FD4**
500 s
400 s
300 s
200 s
100 s
0 s
12 / 112
24 / 112
48 / 112
# of MPI Processes per Node
# of Nodes

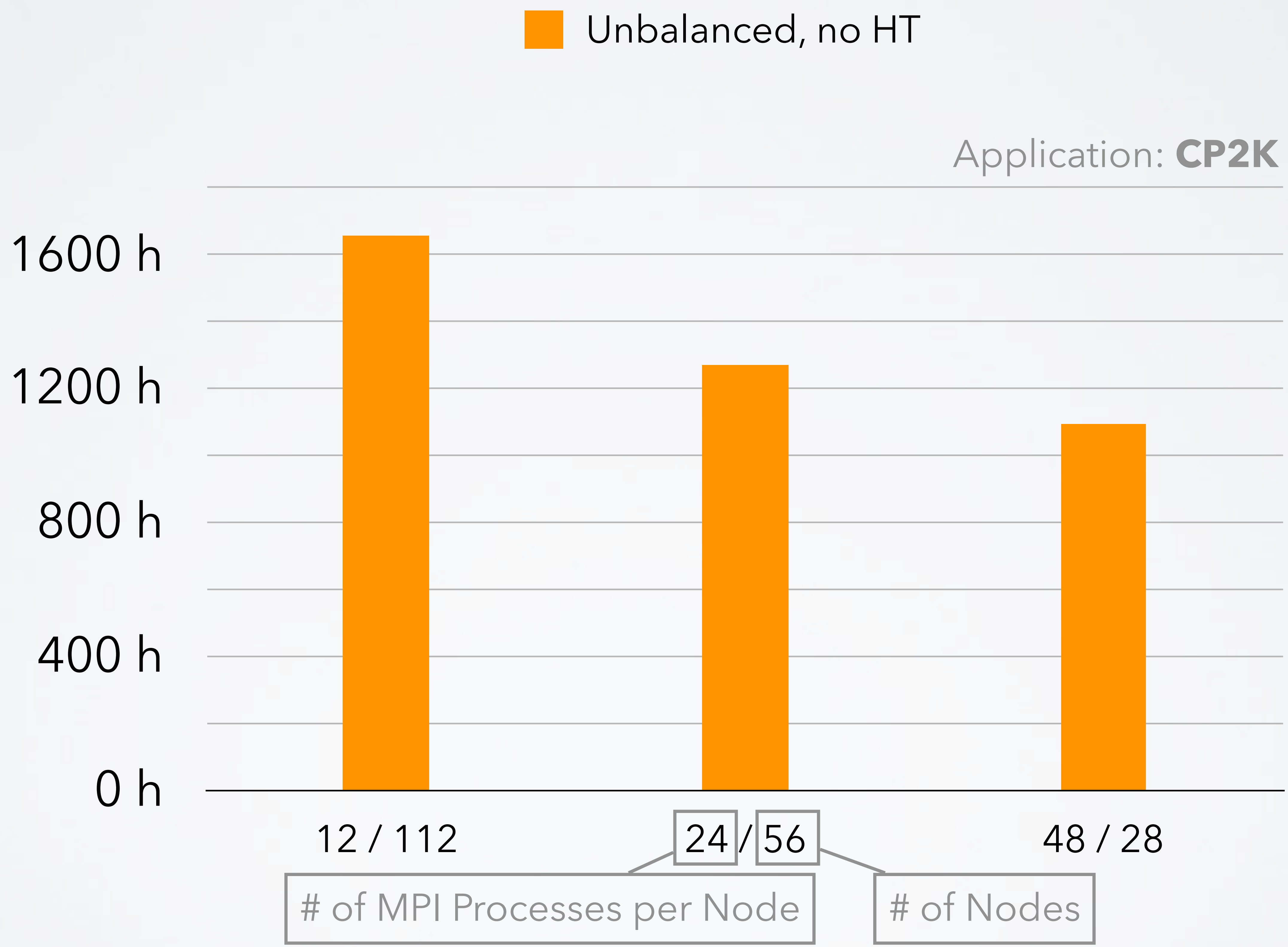

Unbalanced, no HT
Application: **COSMO-SPECS+FD4**
1000 s
800 s
600 s
400 s
200 s
0 s
Number of MPI ranks fixed at 1,344
Number of nodes decreased
12 / 112
24 / 56
48 / 28
# of MPI Processes per Node
# of Nodes

TECHNISCHE UNIVERSITÄT DRESDEN
ZIB
The Hebrew University of Jerusalem

CORE HOURS



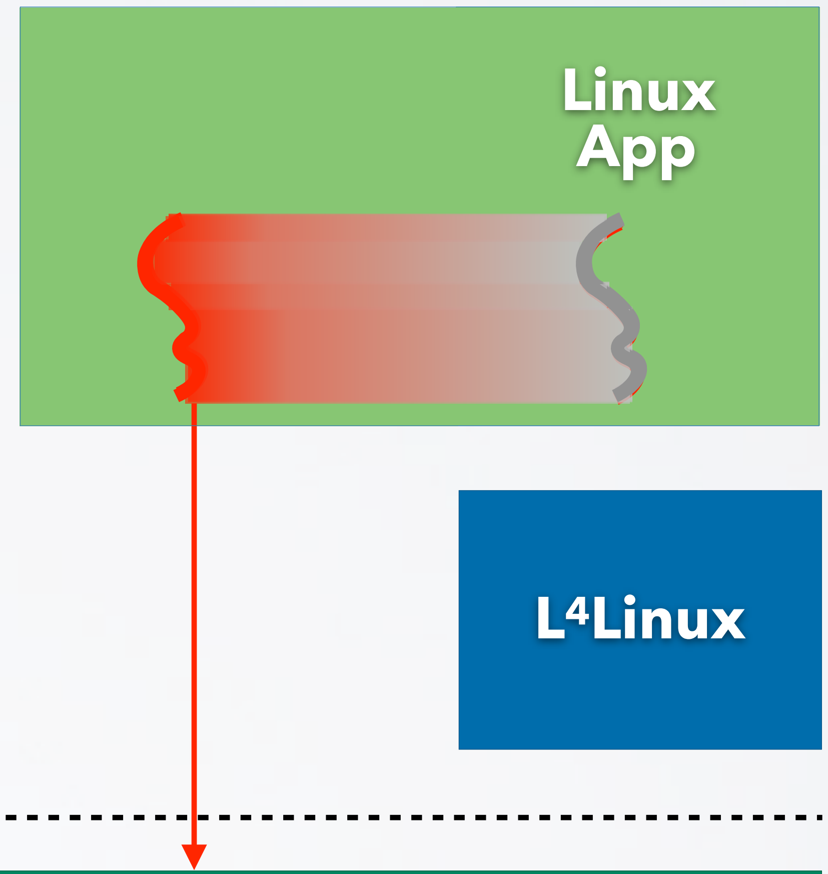
CORE HOURS

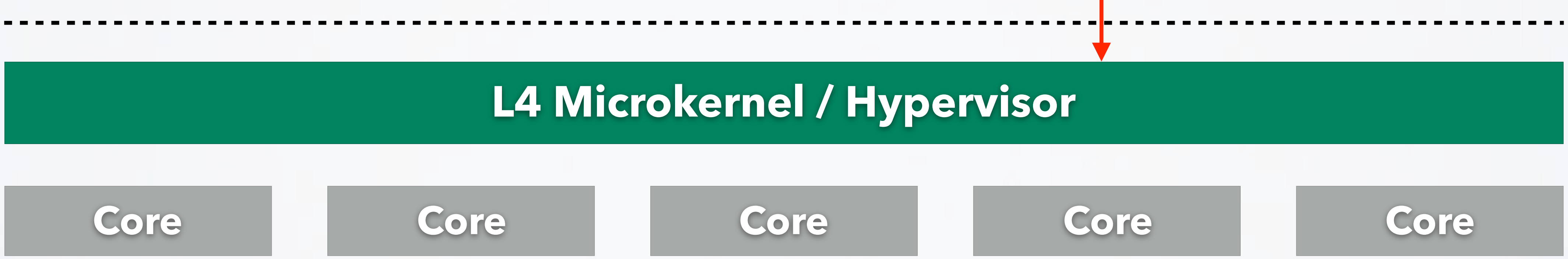

Unbalanced, no HT
Application: **CP2K**
1600 h
1200 h
800 h
400 h
0 h
12 / 112
24 / 56
48 / 28
# of MPI Processes per Node
# of Nodes

WIP: DECOUPLED INTERRUPTS

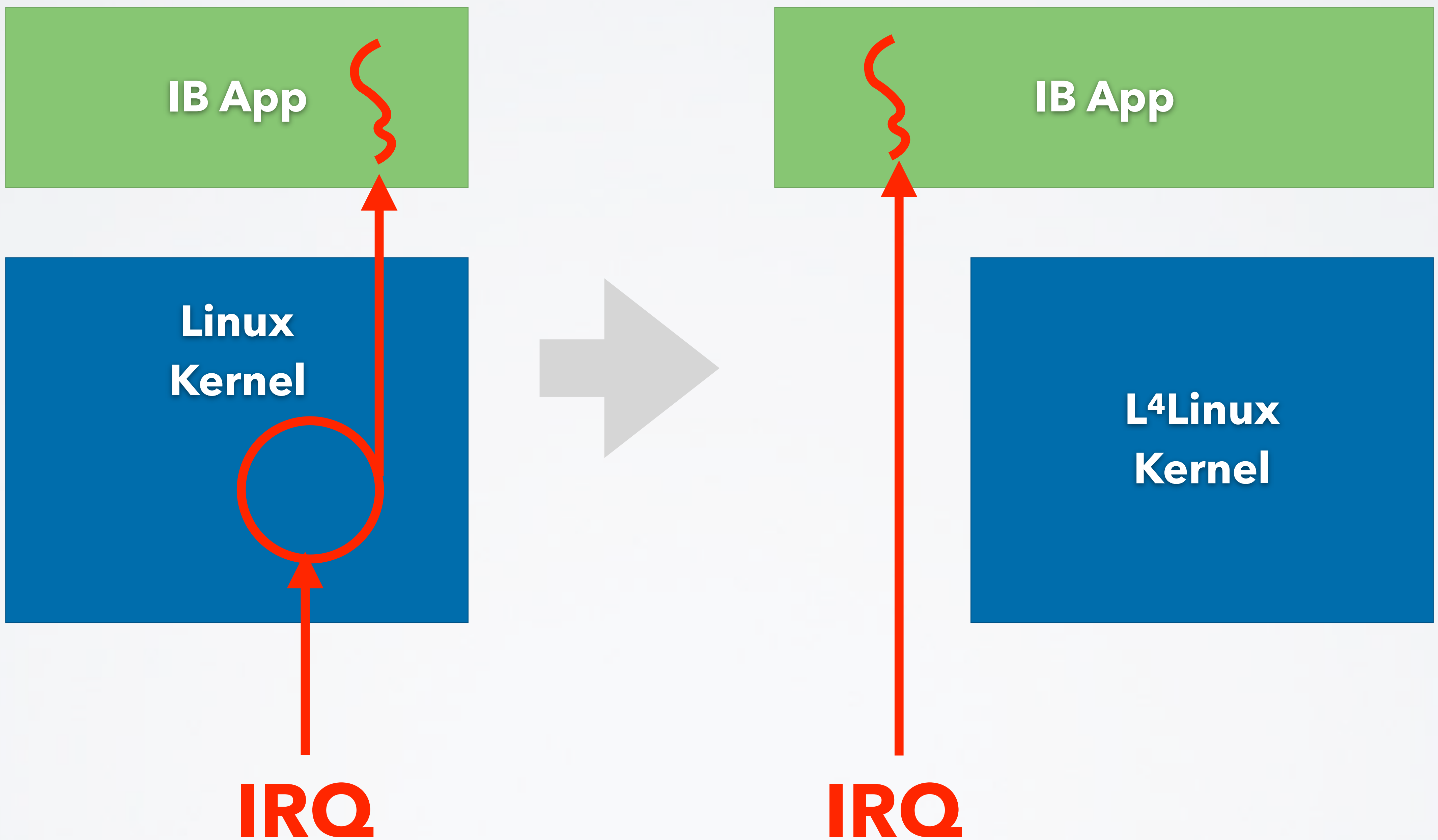
DECOUPLED THREADS

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux
- **Direct I/O** device access
- **L4 syscalls:**
  - Memory
  - Threads & Scheduling
  - Interrupts

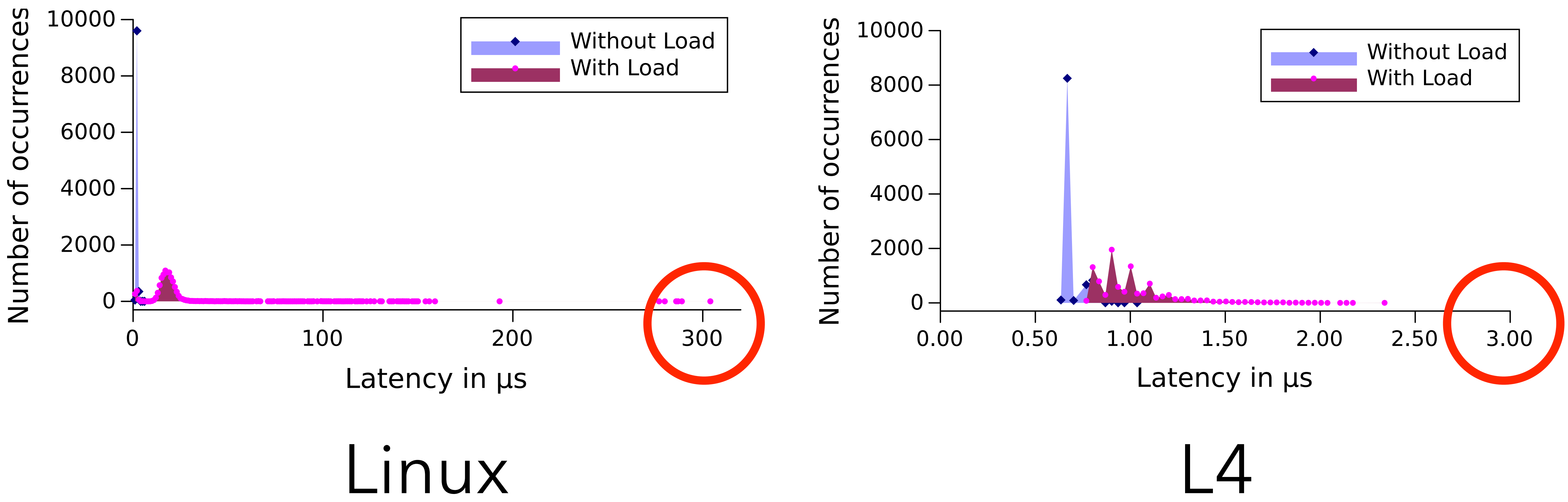




IRQ FASTPATH



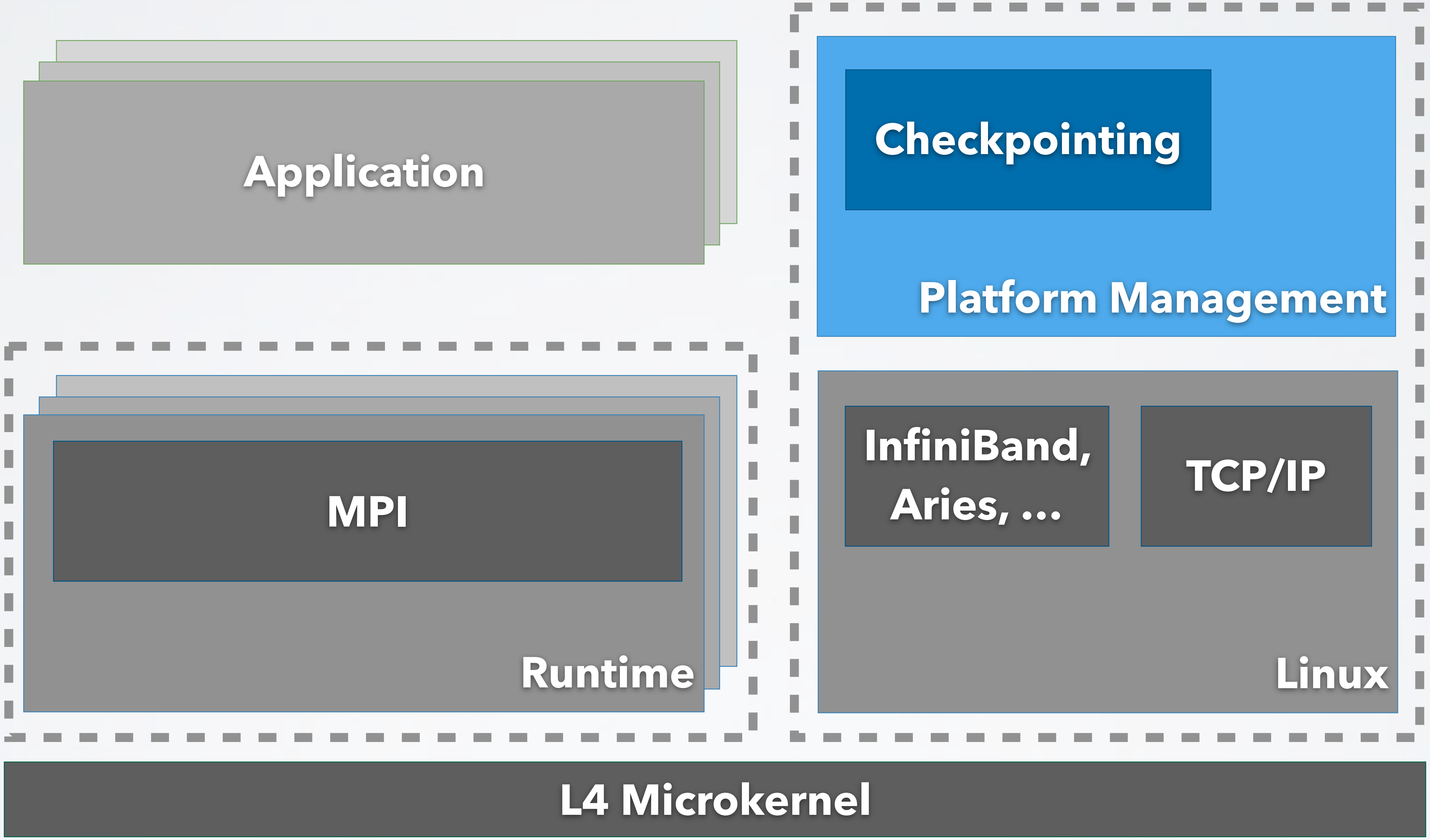
WAKE FROM IRQ



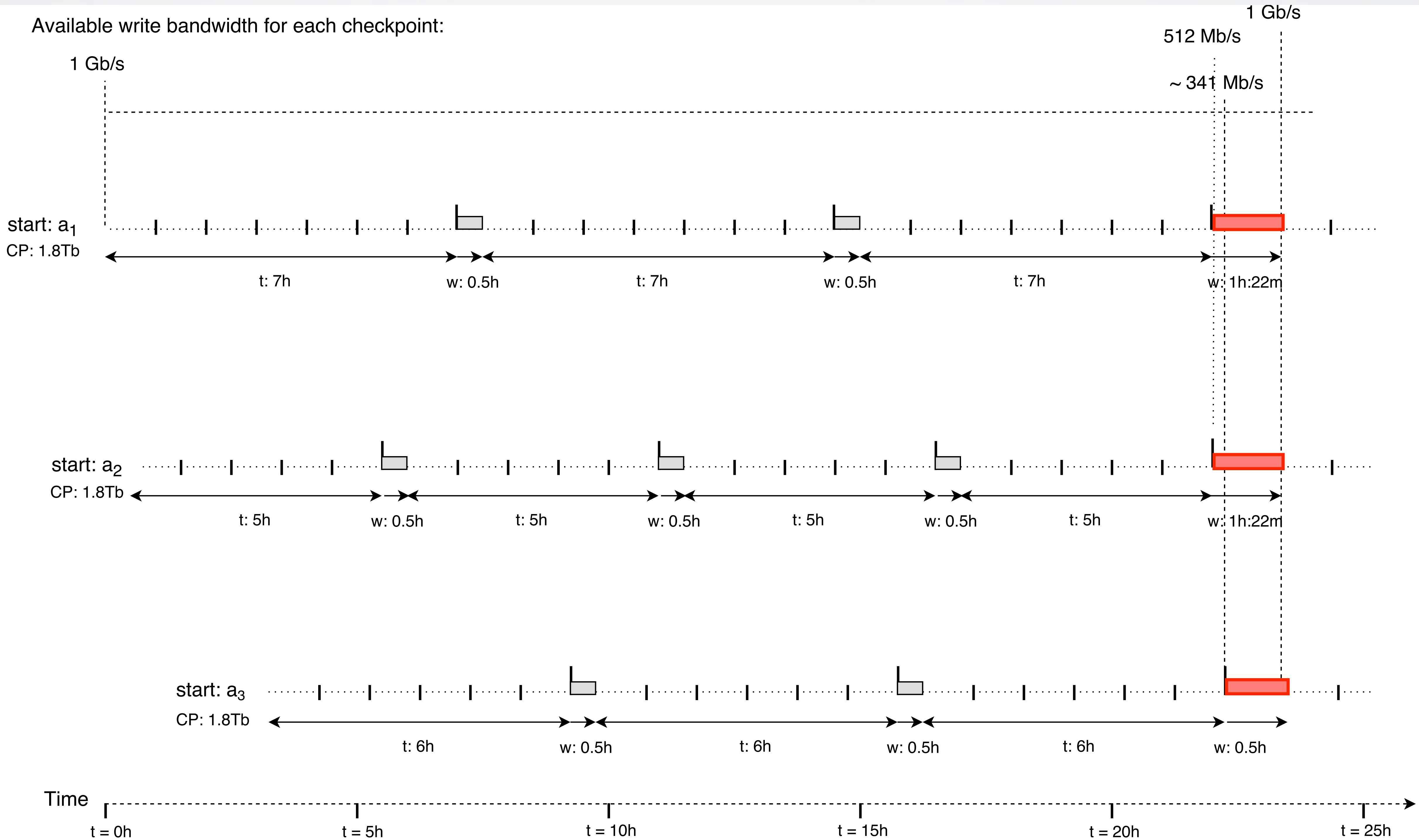
Work in progress: User-space handling of InfiniBand HCA interrupts

Adam Lackorzynski, Carsten Weinhold, Hermann Härtig, „Predictable Low-Latency Interrupt Response with General-Purpose Systems", OSPERT 2017, Dubrovnik, Kroatia, June 2017

NODE ARCHITECTURE

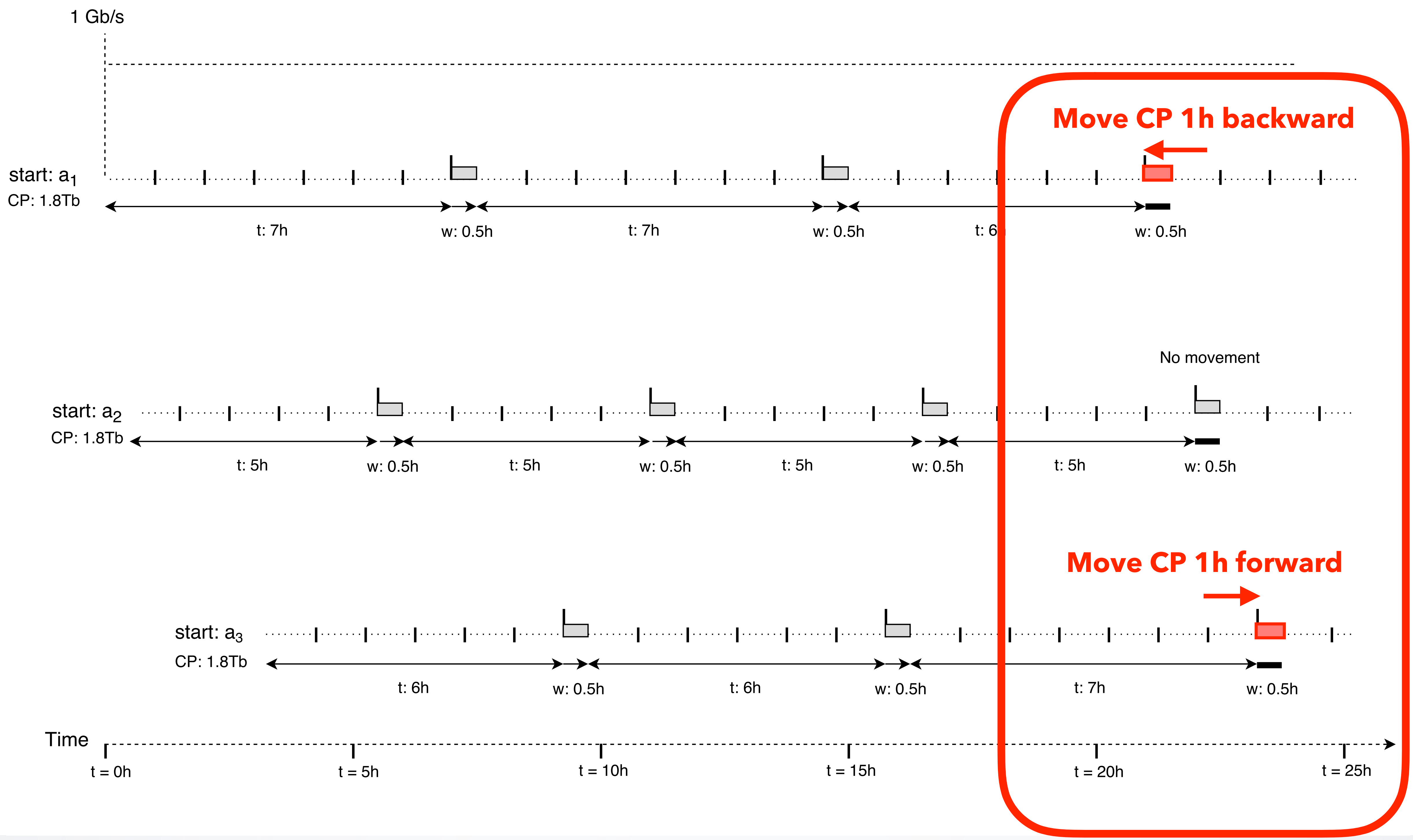


COORDINATED C/R

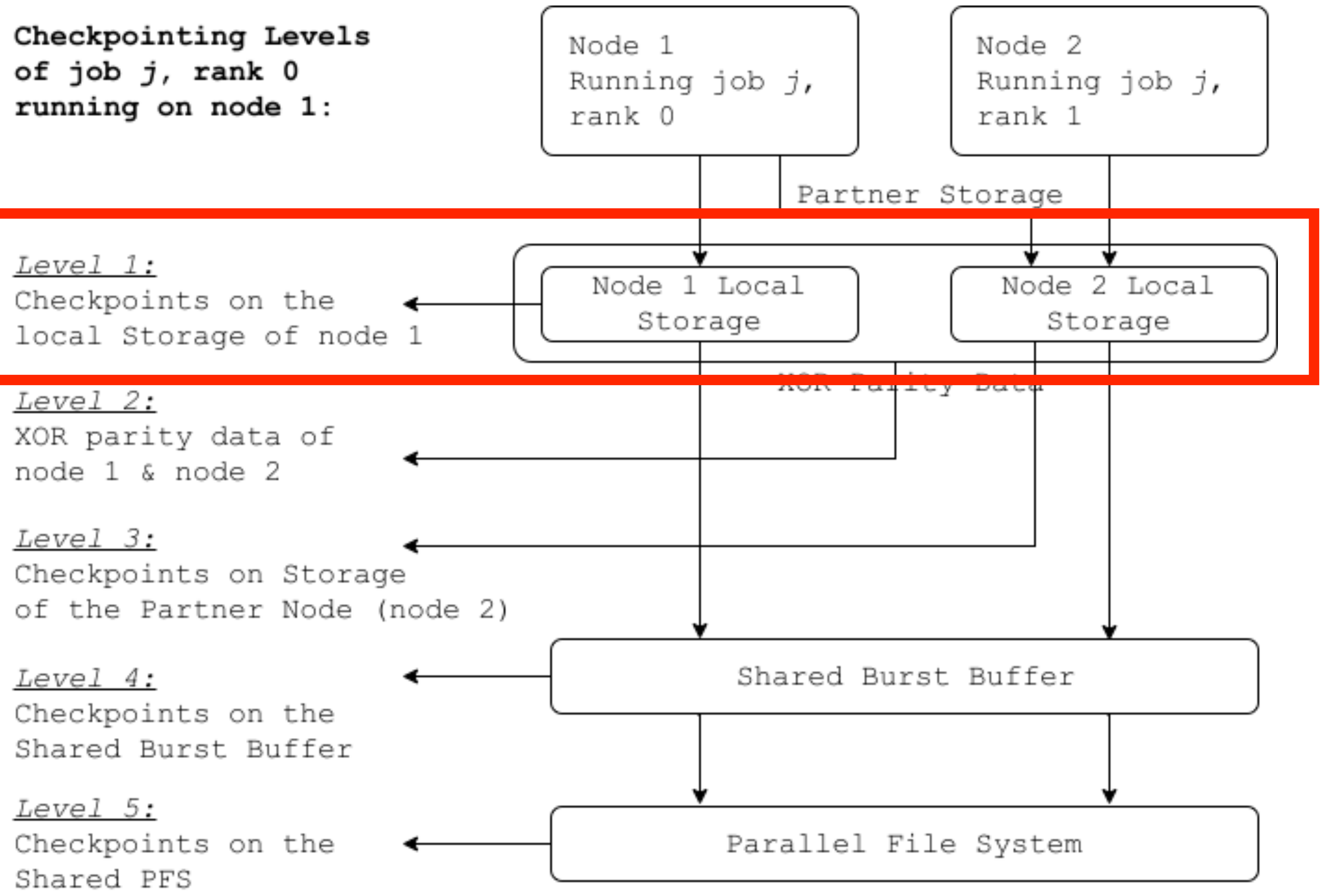


COORDINATED C/R

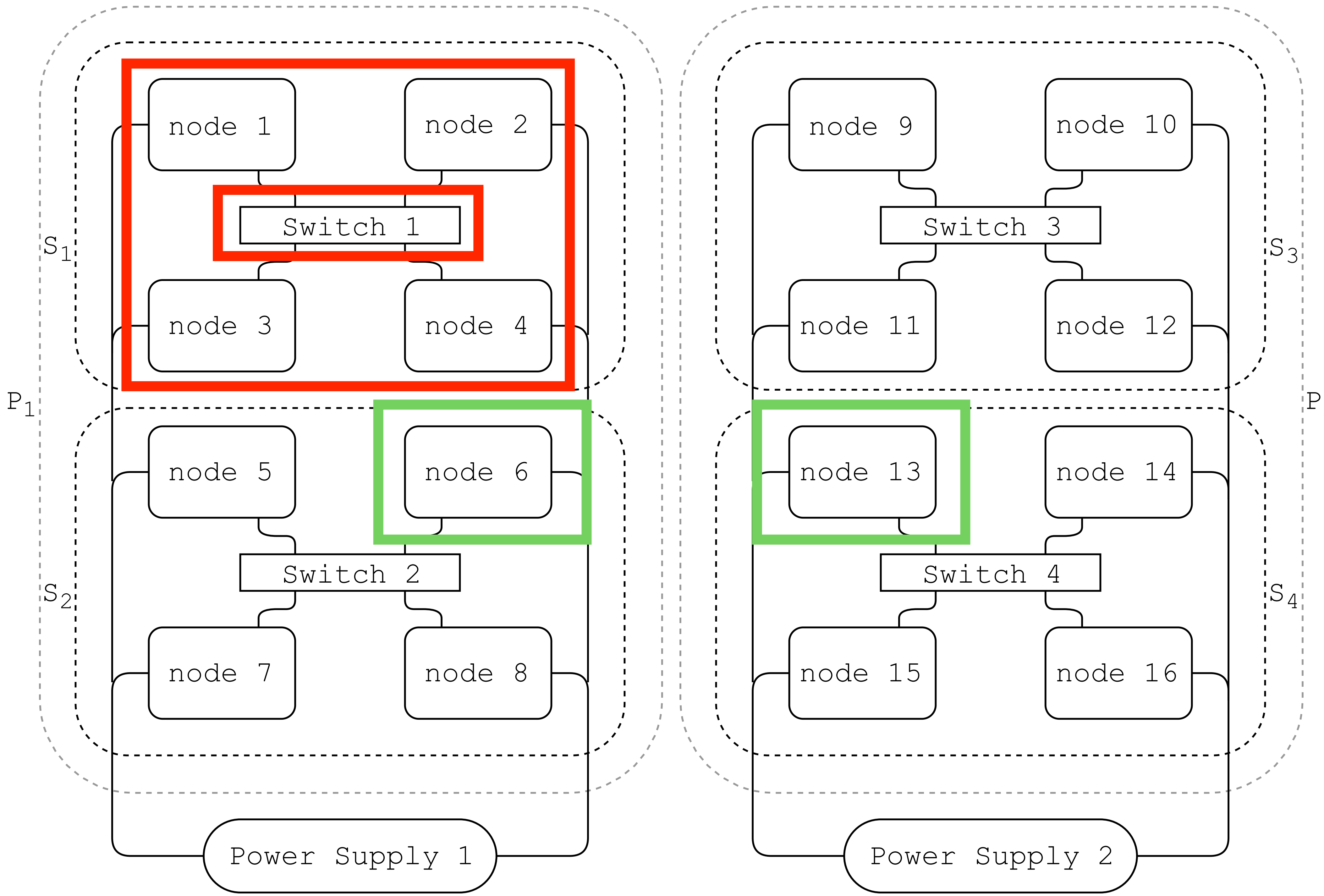
Available write bandwidth for each checkpoint:



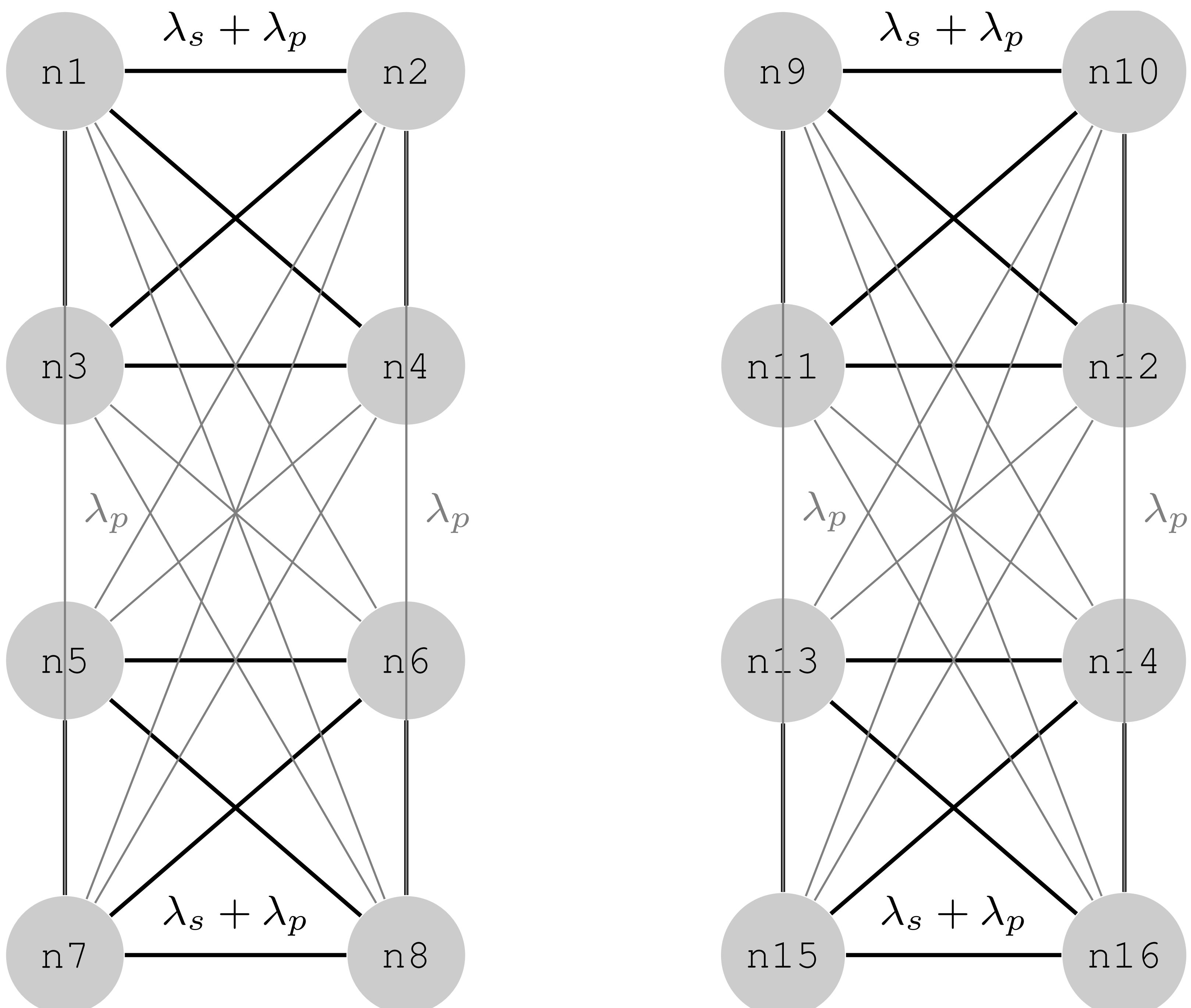
MULTI-LEVEL C/R



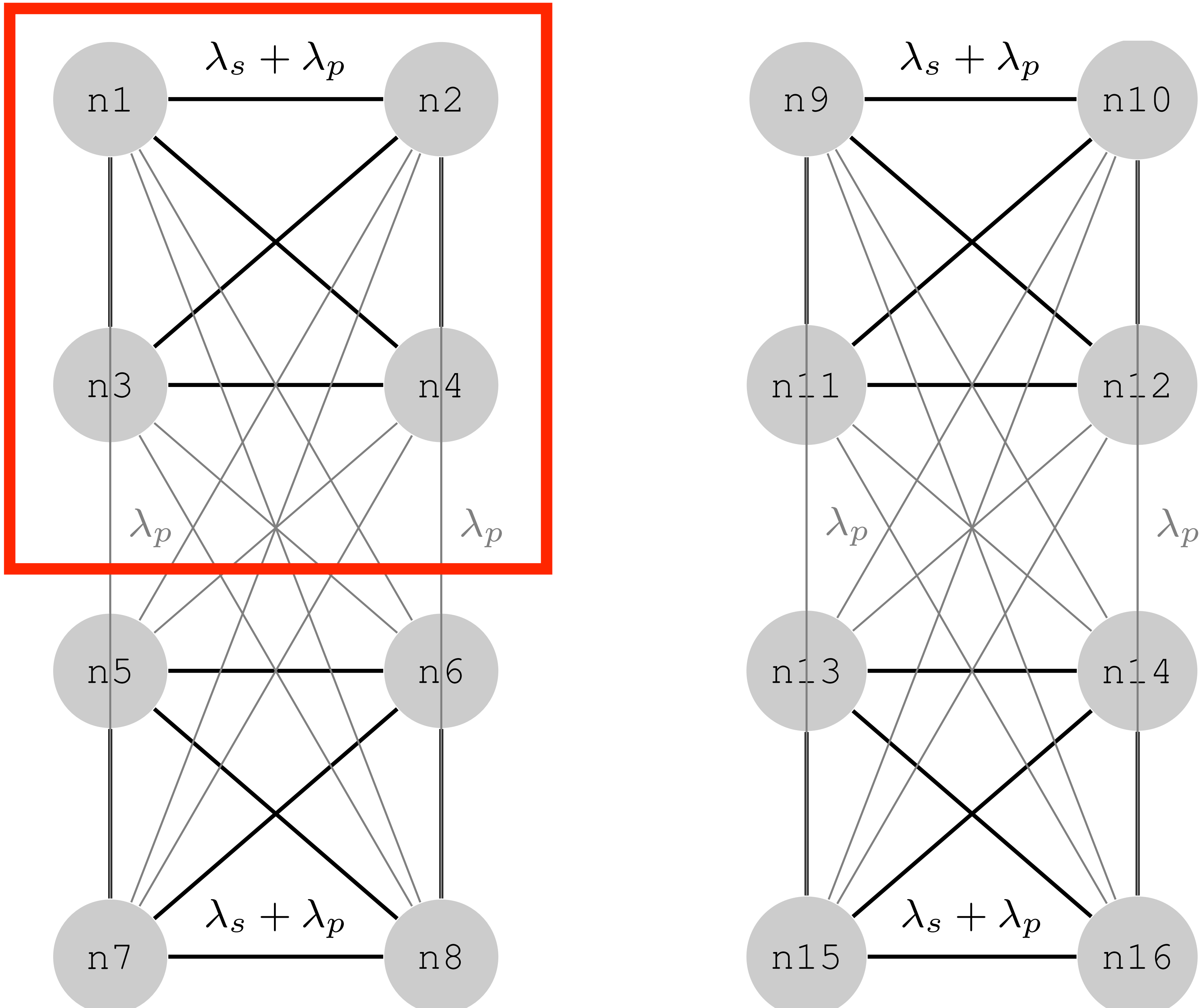
CORRELATED FAILURE



CORRELATED FAILURE



CORRELATED FAILURE



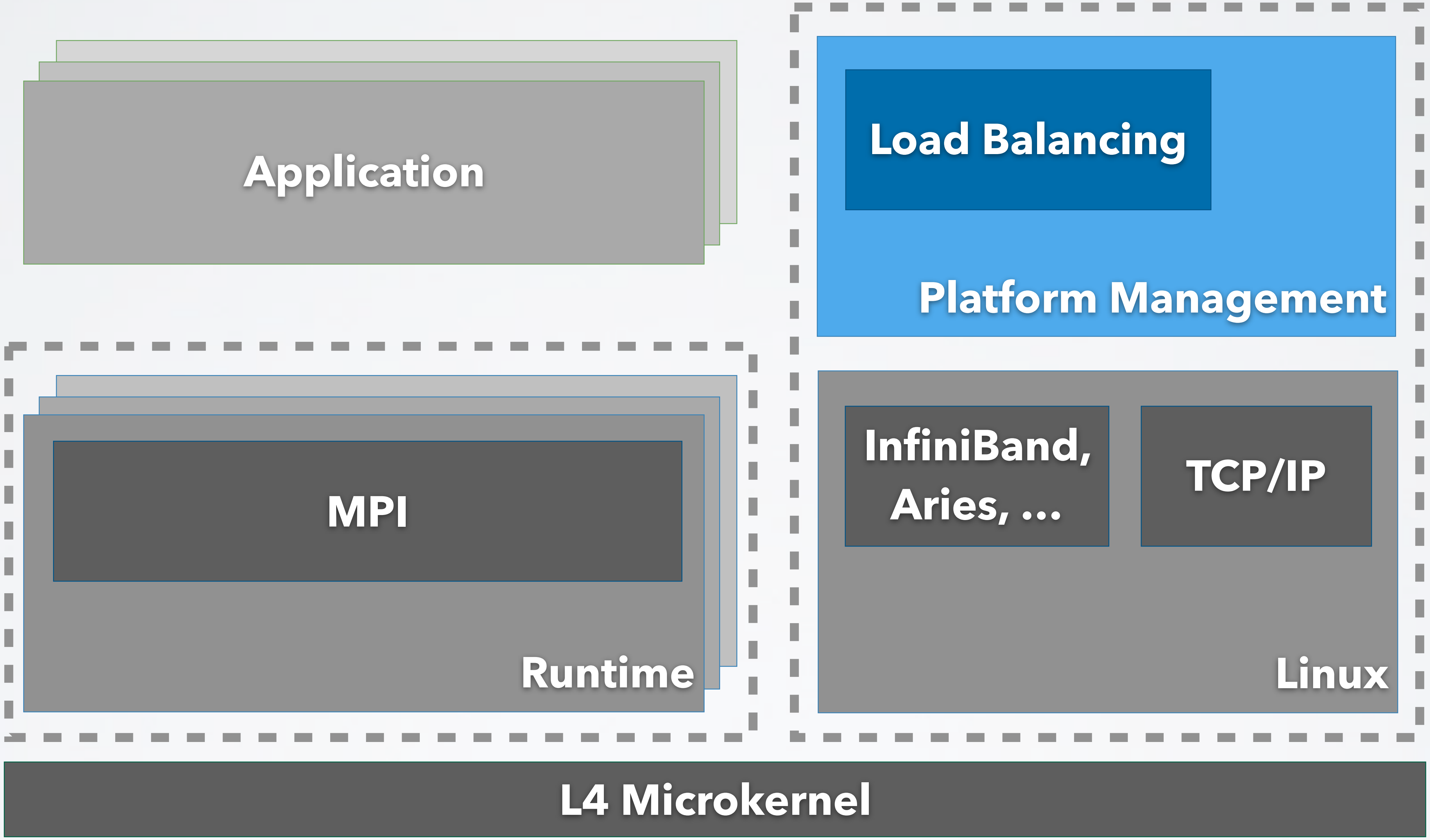
Graph problem:

- Find disjoint independent sets
- Find dominating subgraphs („least correlated nodes“)

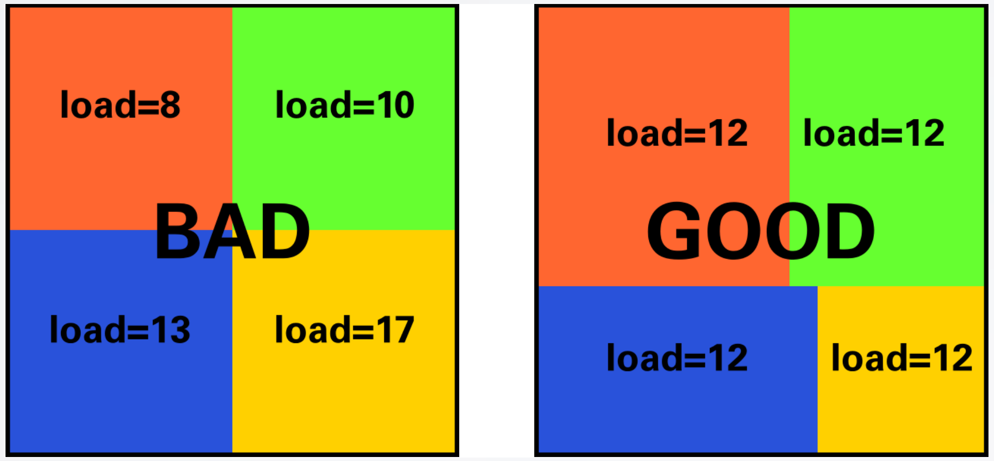
Optimization problem:

- least correlated nodes for checkpoint distribution
- Consider: job run time, C/R cost, MTTI
- Minimize run time

NODE ARCHITECTURE

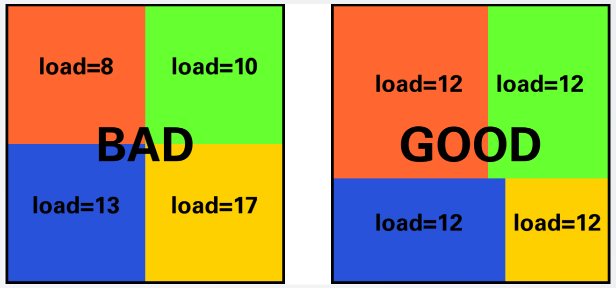


- Balance workload

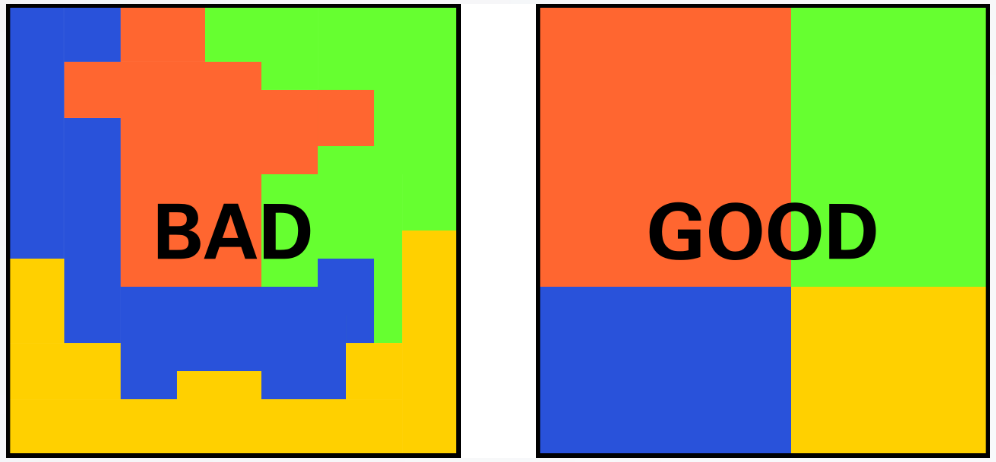


LOAD BALANCING

- Balance workload

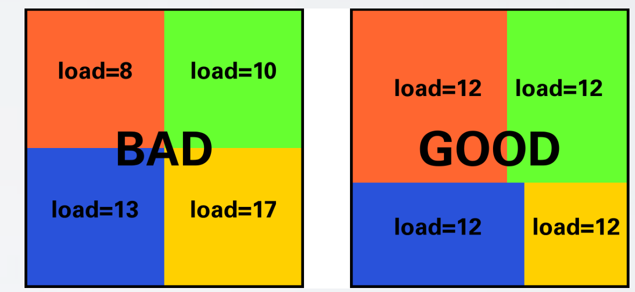


- Minimize communication between partitions

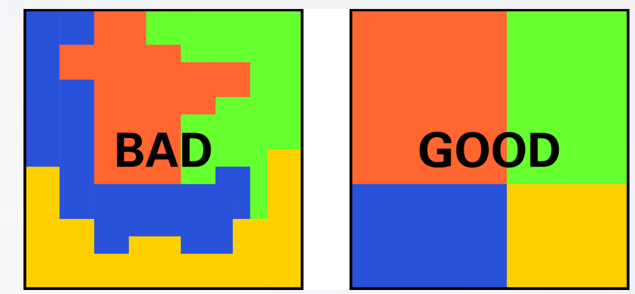


LOAD BALANCING

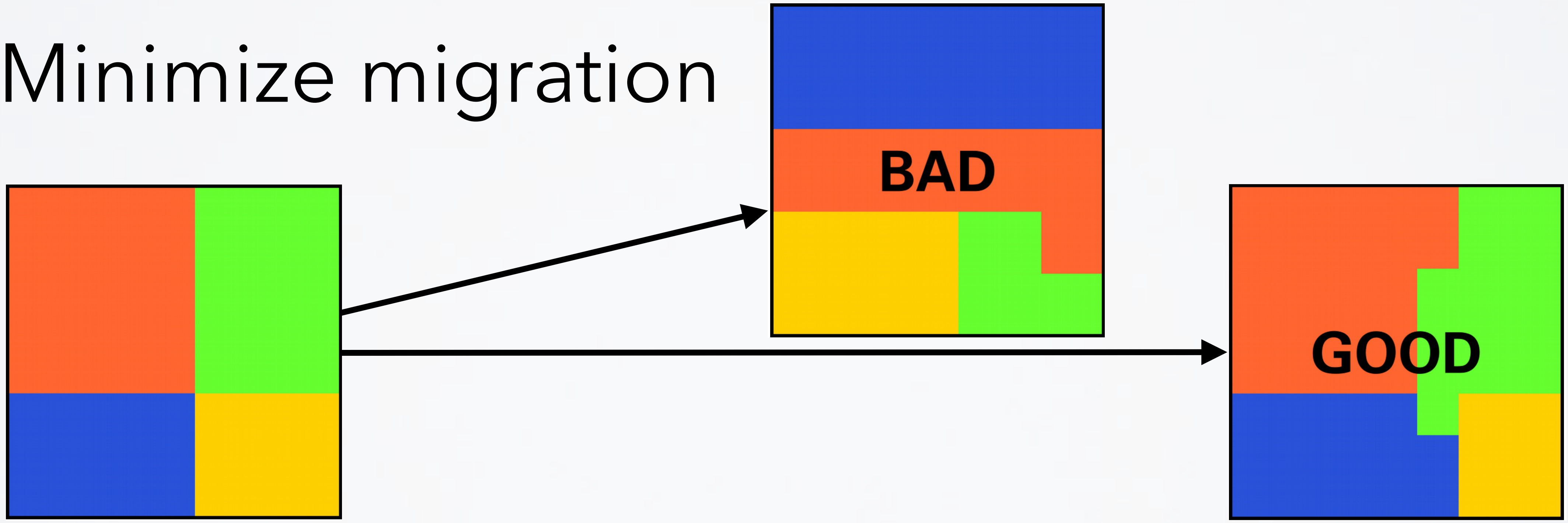
- Balance workload



- Minimize communication between partitions

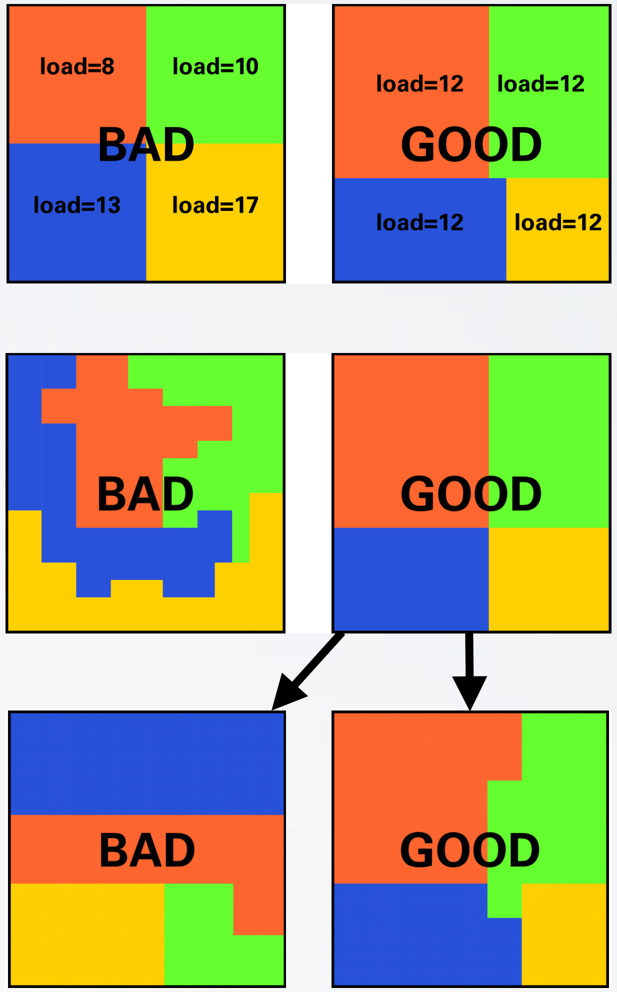


- Minimize migration

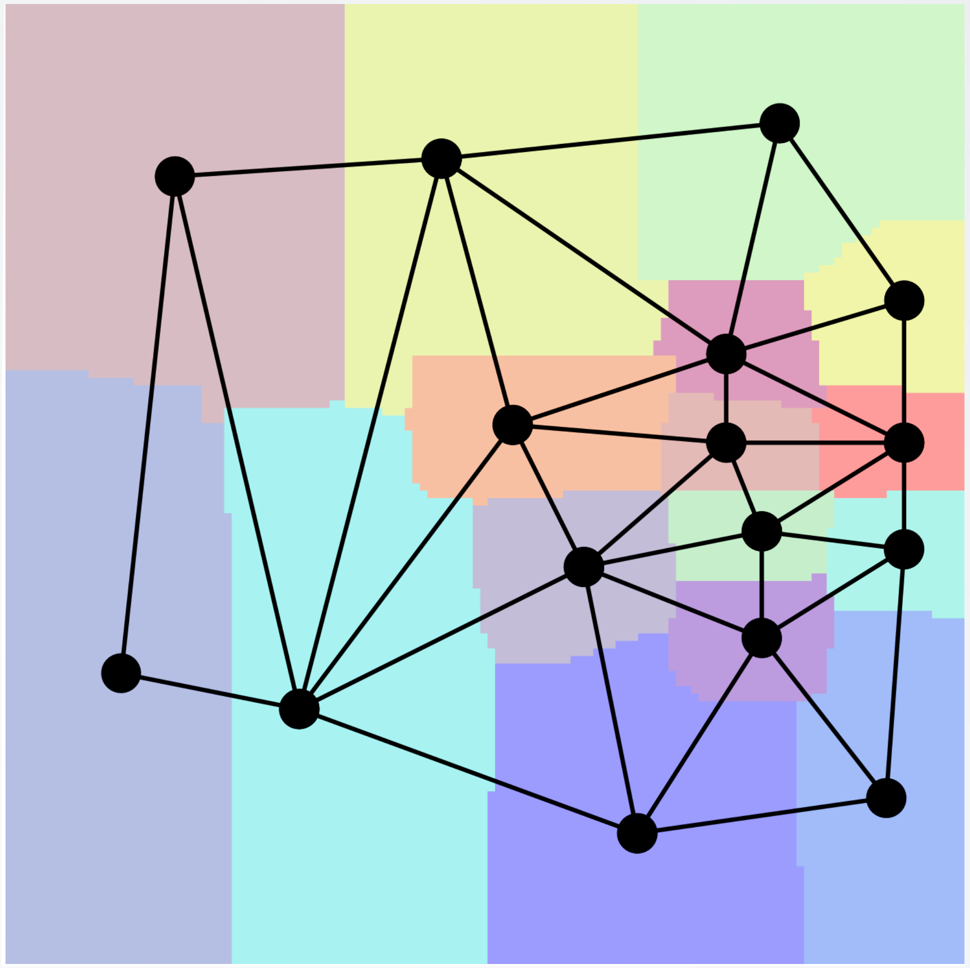


LOAD BALANCING

- Balance workload
- Minimize communication between partitions
- Minimize migration
- Compute new partitions fast



DIFFUSION

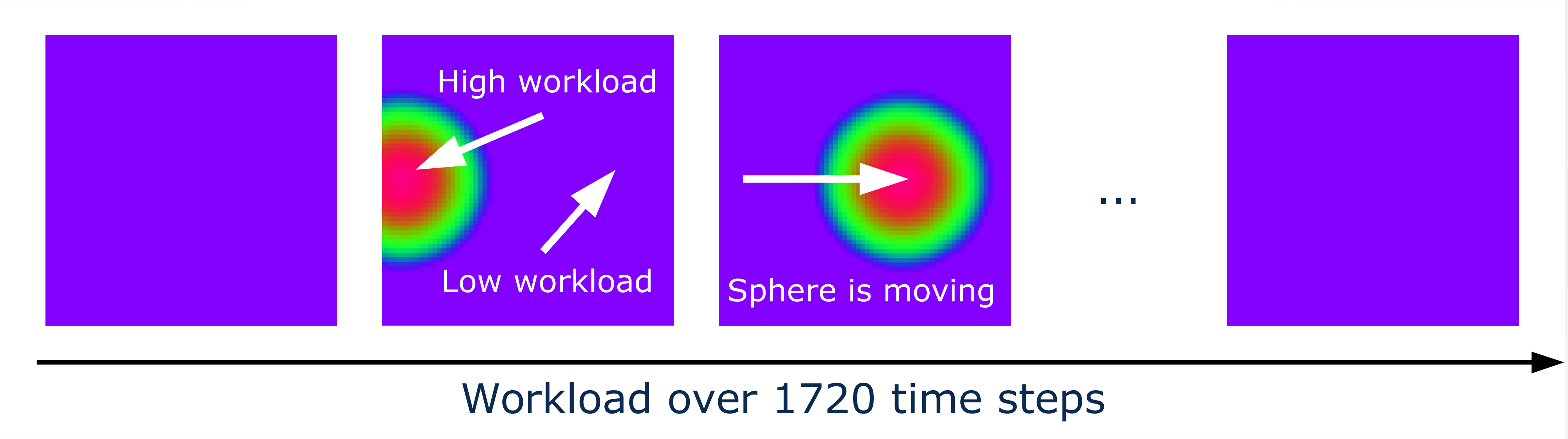


Diffusion graph topology from application topology

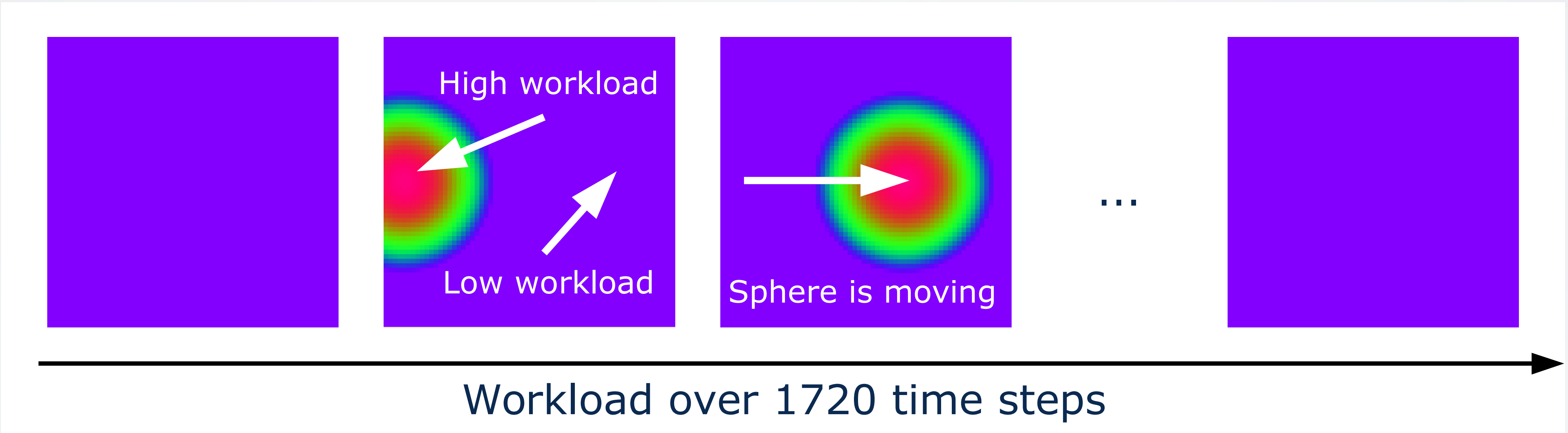
Diffusion coefficient weighted by interface length:

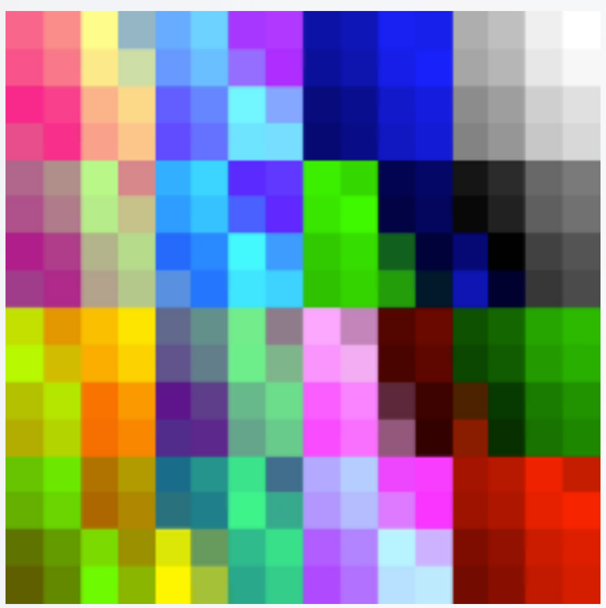
- Tasks migrated between neighbor partitions
- Better partition shape

DIFFUSION EXAMPLE

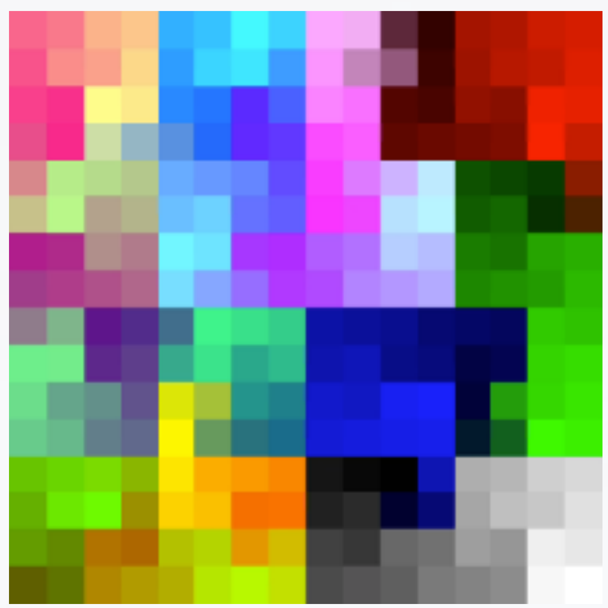


DIFFUSION EXAMPLE

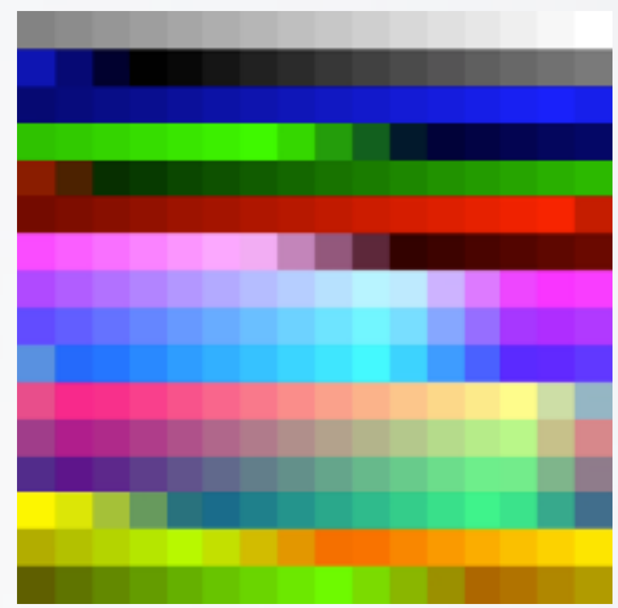




Zoltan

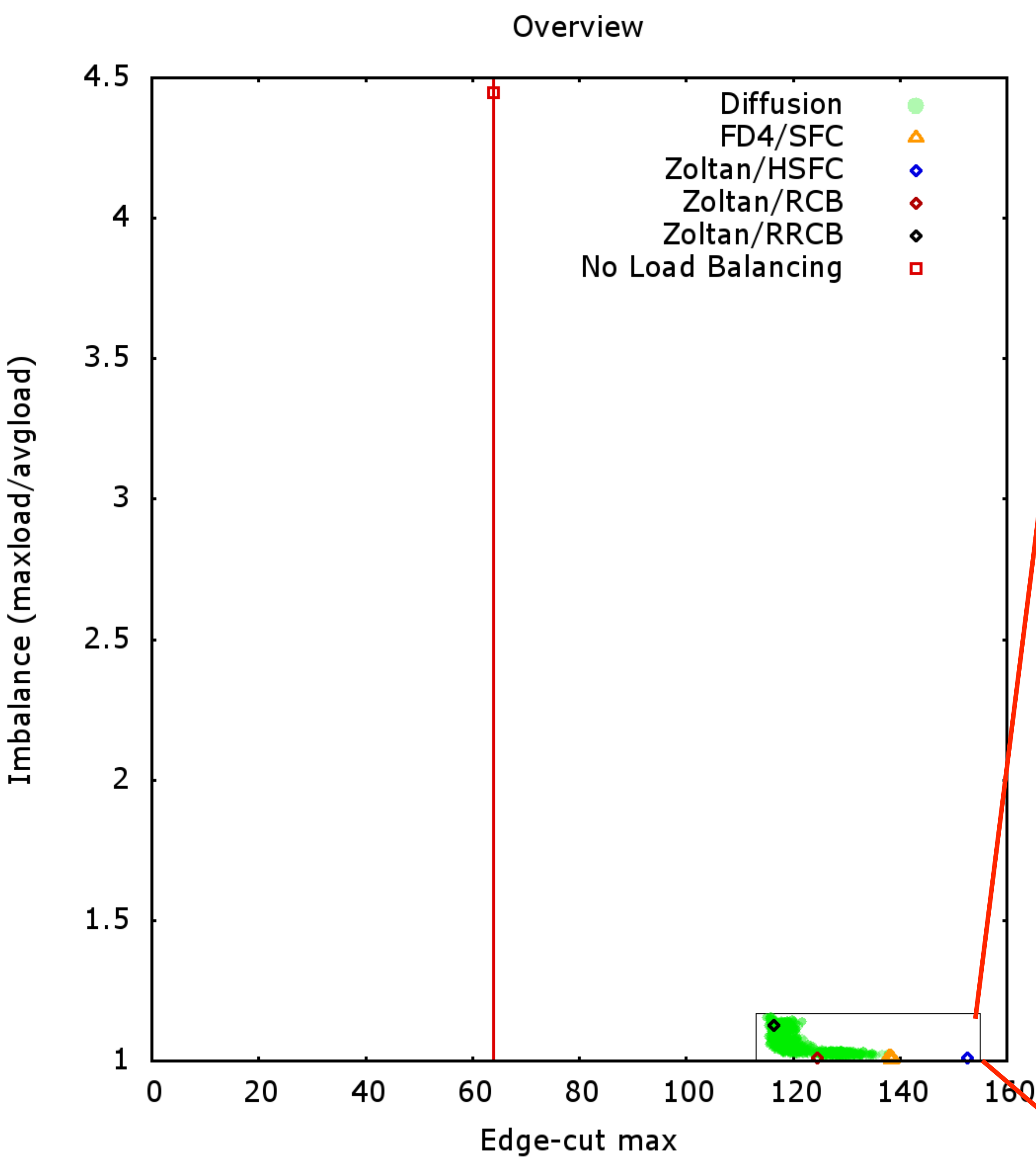


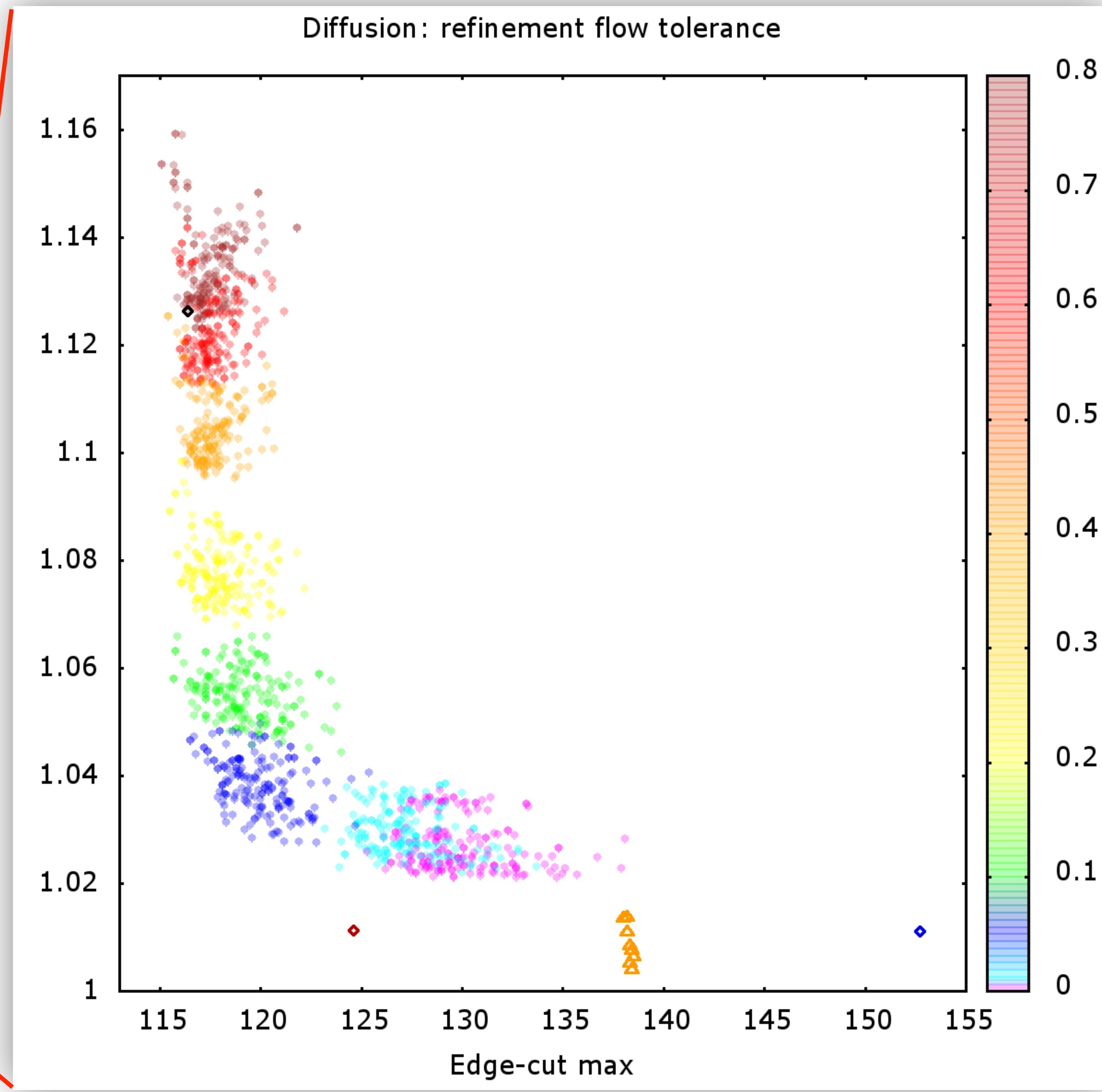
Space-filling Curves



Diffusion

DIFFUSION RESULTS

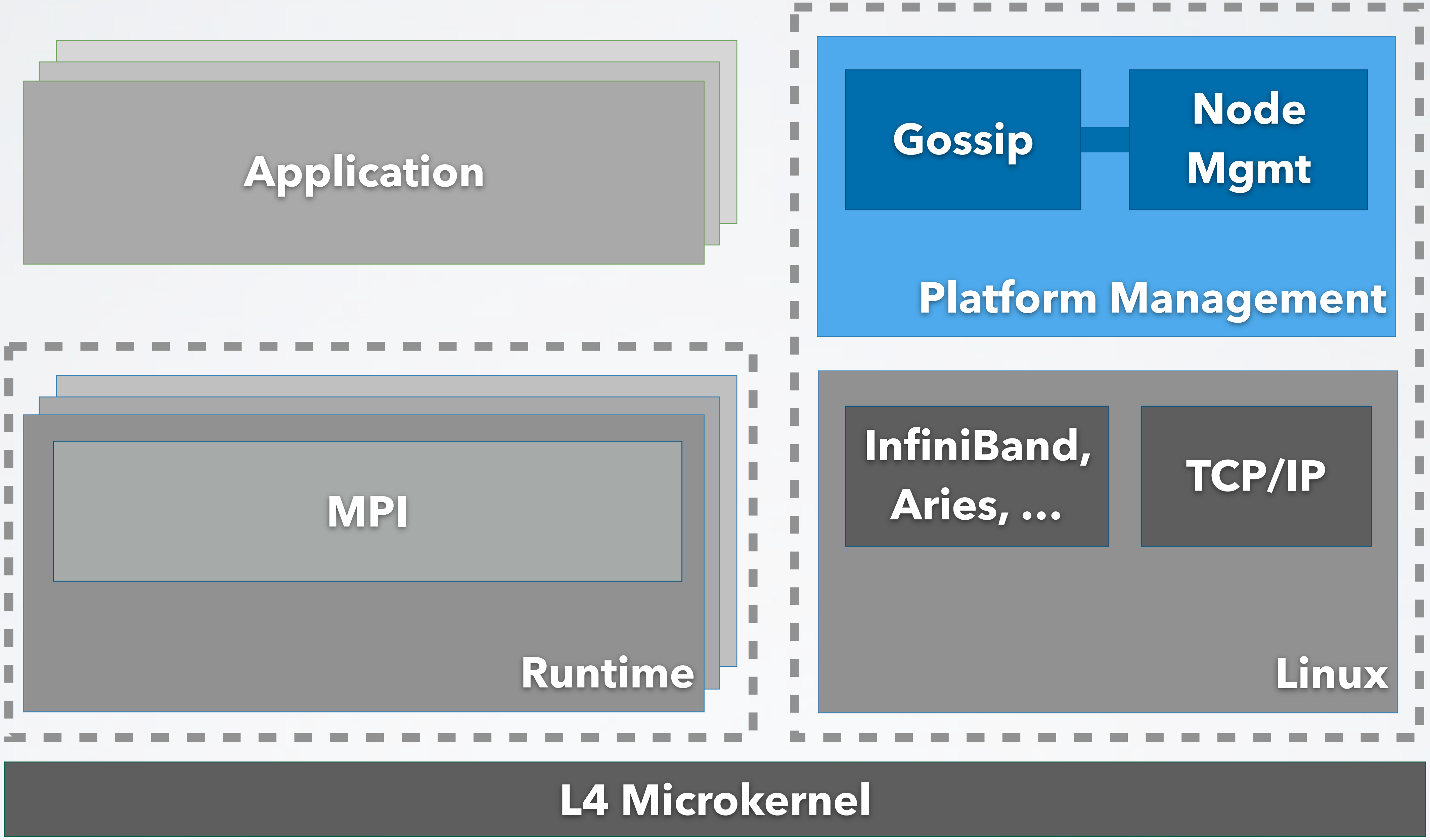




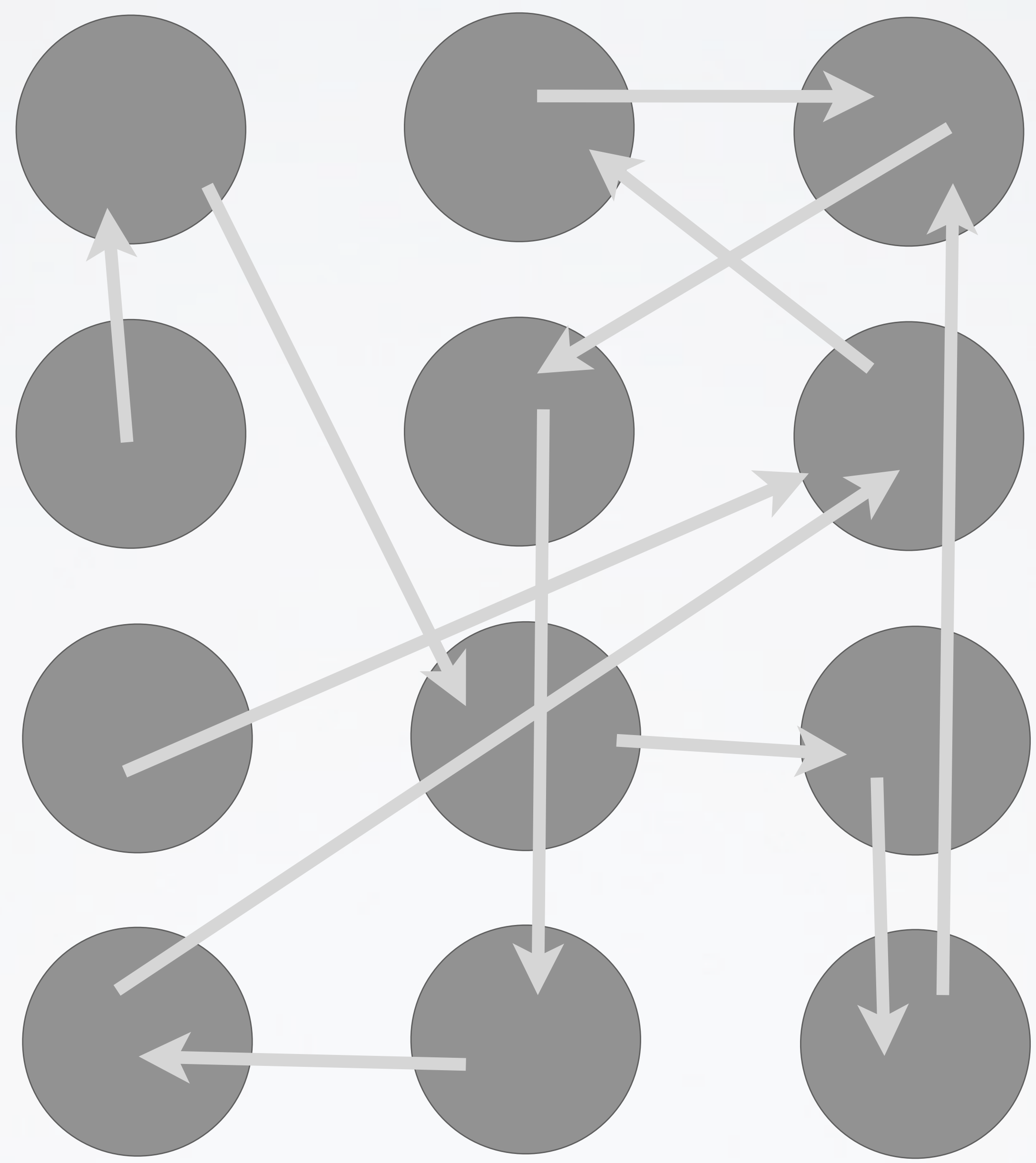
DIFFUSION SUMMARY

- **Best method to reduce:**
  - Migrations (less data movement)
  - Edge cut (less communication)
- **Load balance** good, but not superior
- **Flexible:** uses communication graph specific to application

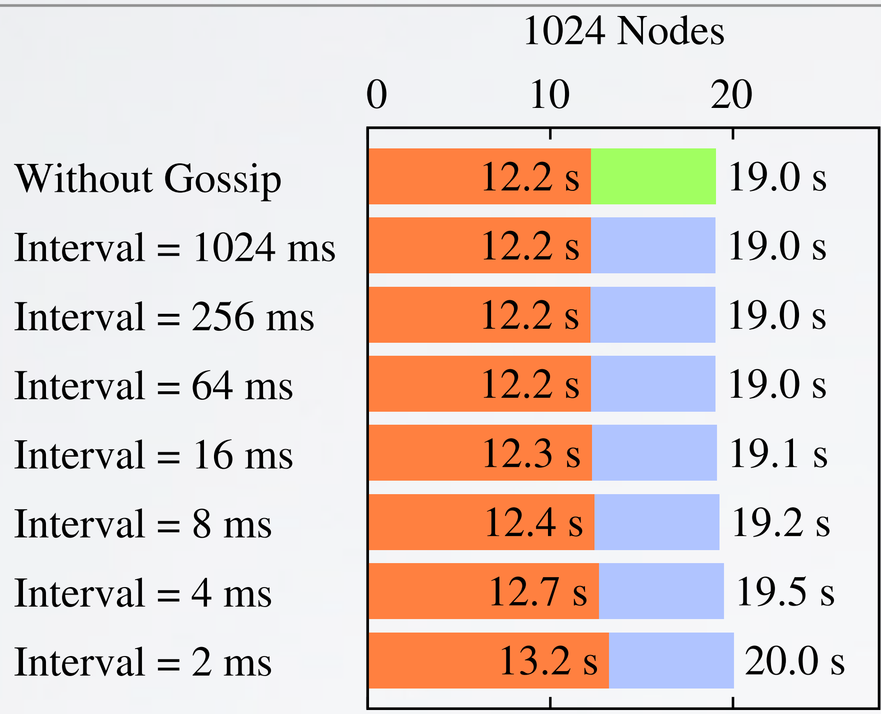
NODE ARCHITECTURE



GOSSIP



GOSSIP SCALABILITY



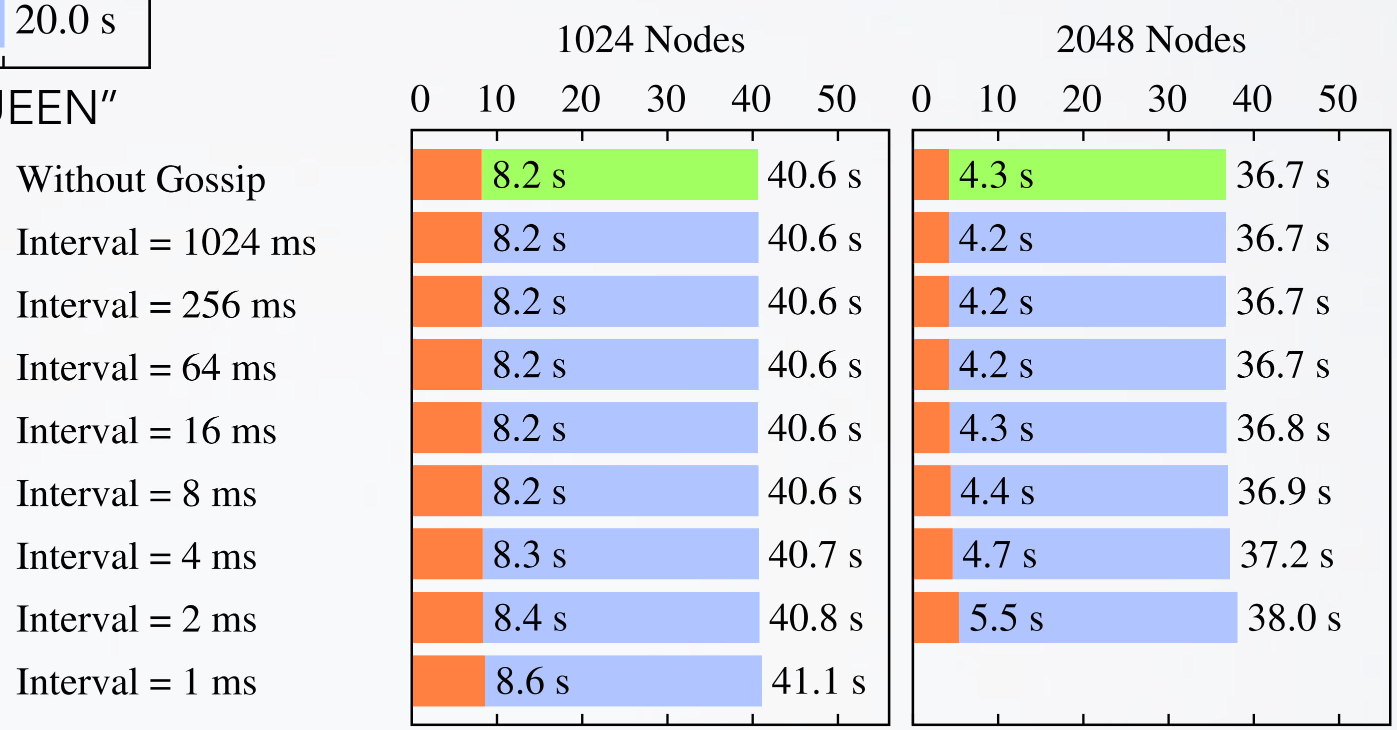
MPI-FFT running on BG/Q "JUQUEEN"

Quality of information:

Average age at nodes in the order of 2–3 s with gossip interval of 256 ms

Low overhead:

No noticeable overhead at gossip interval of 64–256 ms



COSMO-SPECS+FD4 on BG/Q "JUQUEEN"

E. Levy, A. Barak, A. Shiloh, M. Lieber, C. Weinhold, and H. Härtig, „Overhead of a Decentralized Gossip Algorithm on the Performance of HPC Applications", ROSS 2014

GOSSIP VS FAULTS

Number of failed nodes per colony	Circulating *local windows* of size				
	16	**32**	**64**	**128**	**256**
0	11.74	9.67	8.66	8.20	8.07
1	11.71	9.72	8.67	8.21	8.07
2	11.75	9.68	8.70	8.21	8.08
4	11.81	9.73	8.70	8.23	8.11
8	11.83	9.79	8.72	8.28	8.17
16	11.95	9.90	8.79	8.34	8.20
32	12.12	10.05	8.96	8.48	8.36
Standard deviation	0.49	0.42	0.37	0.36	0.36
Increase rate	3.2%	3.9%	3.5%	3.4%	3.6%

Average age at master (1024 nodes per colony)

Gossip is fault tolerant:

Only slight increase in average age when substantial number of nodes fail (up to 32 of 1024 in each colony)

A. Barak, Z. Drezner, E. Levy, M. Lieber, and A. Shiloh, „Resilient gossip algorithms for collecting online management information in exascale clusters“, Concurrency and Computation: Practice and Experience, 2015

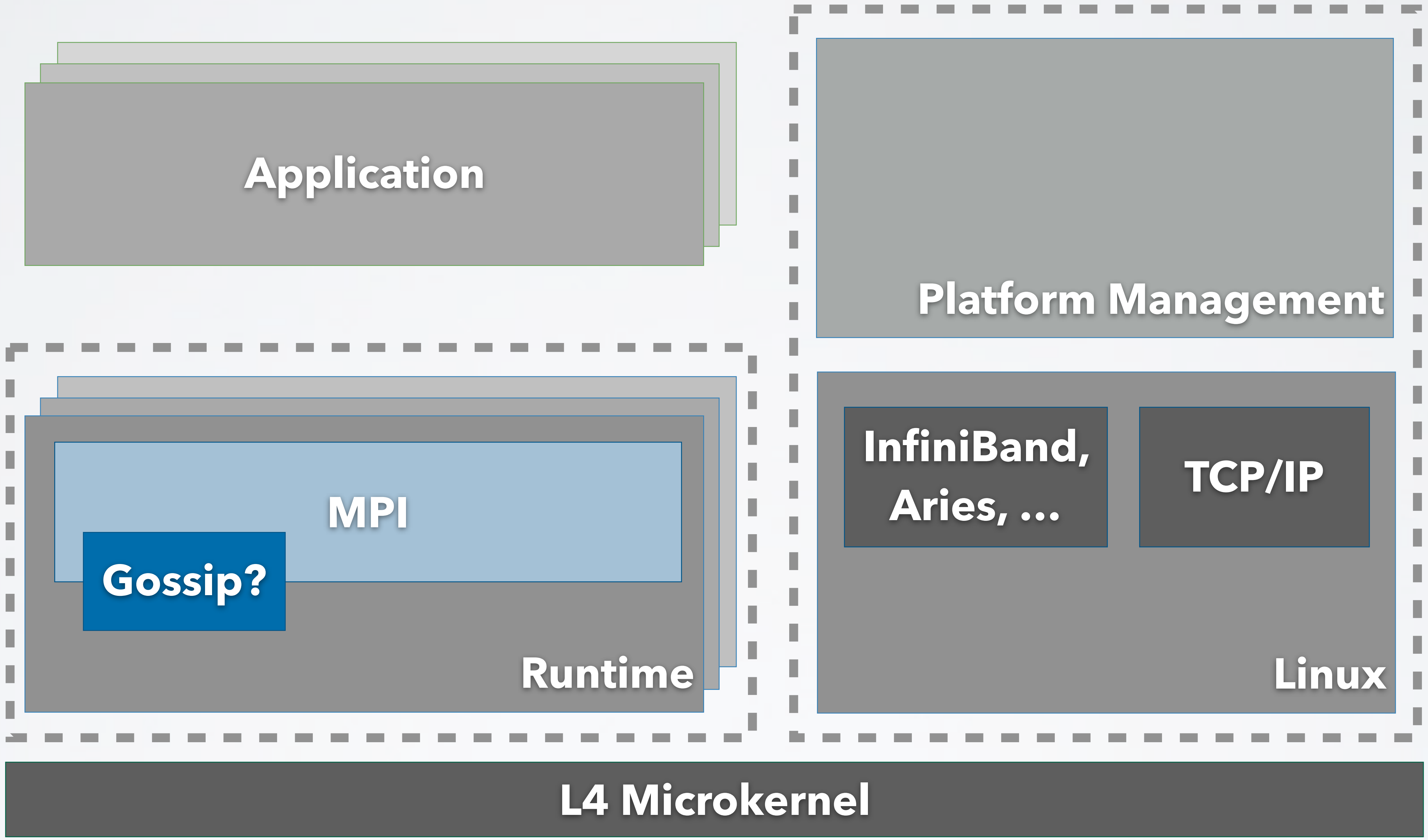
GOSSIP ALGORITHMS

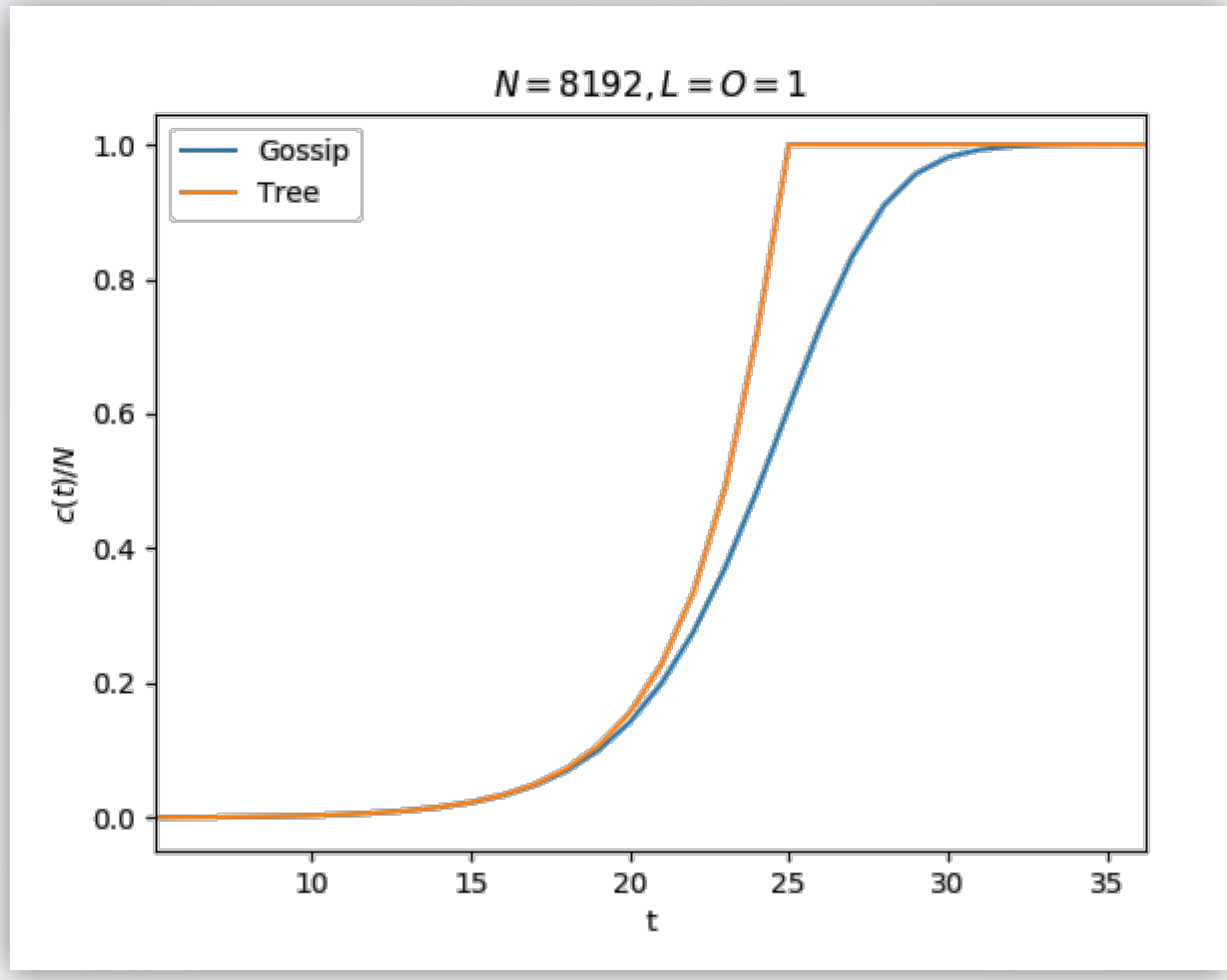
- Spread load+health info among nodes
- Analytic model ~ simulation ~ emulation
- **Negligible overhead** (64–256 ms intervals)
- **Good quality of information** (2–3 s old)
- **Fault tolerant** (simulated for up to 32 of 1024 nodes failing)

E. Levy, A. Barak, A. Shiloh, M. Lieber, C. Weinhold, and H. Härtig, „Overhead of a Decentralized Gossip Algorithm on the Performance of HPC Applications“, ROSS 2014

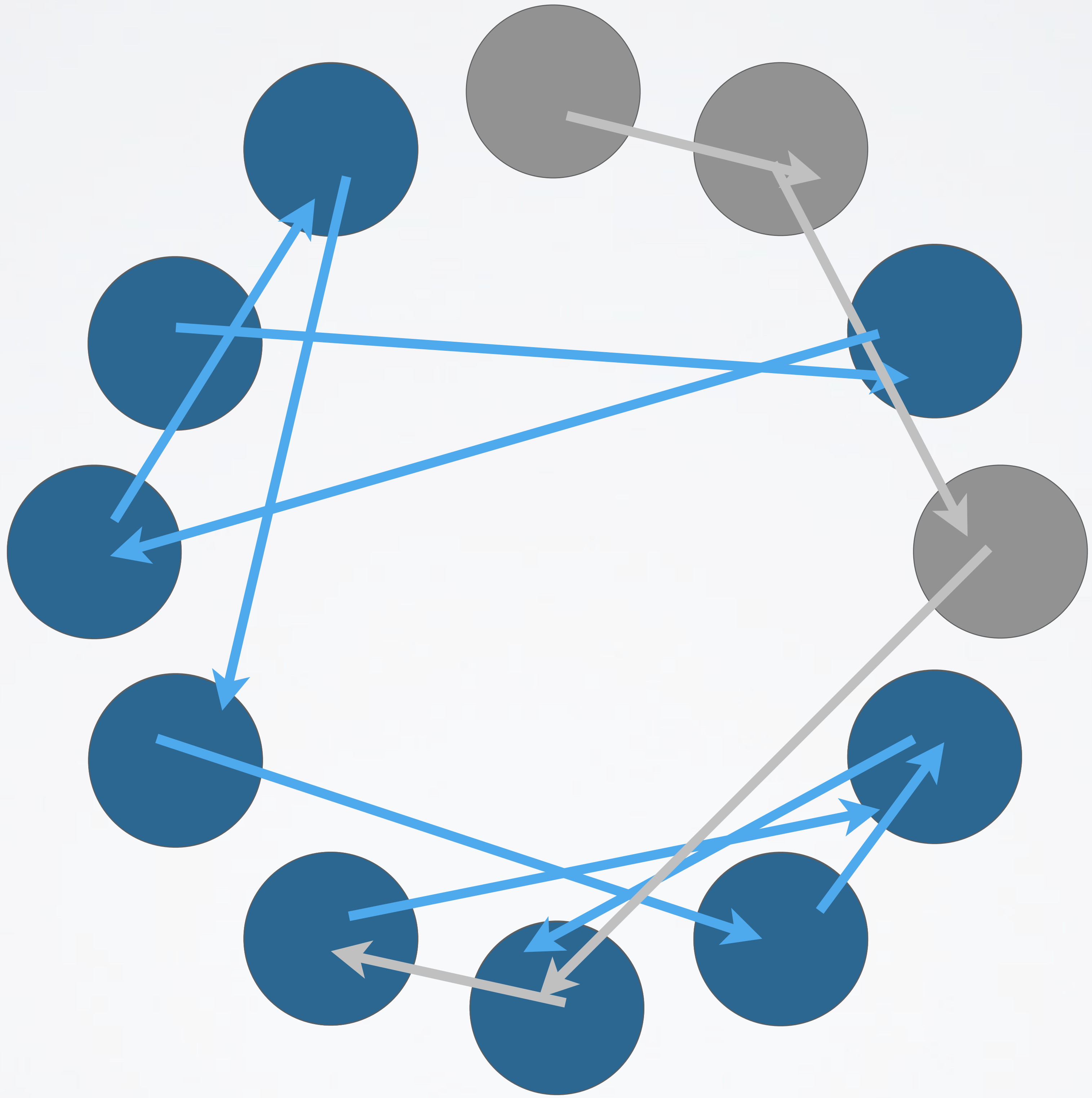
A. Barak, Z. Drezner, E. Levy, M. Lieber, and A. Shiloh, „Resilient Gossip Algorithms for Collecting Online Management information in Exascale Clusters“, Concurrency and Computation: Practice and Experience, 2015

NODE ARCHITECTURE

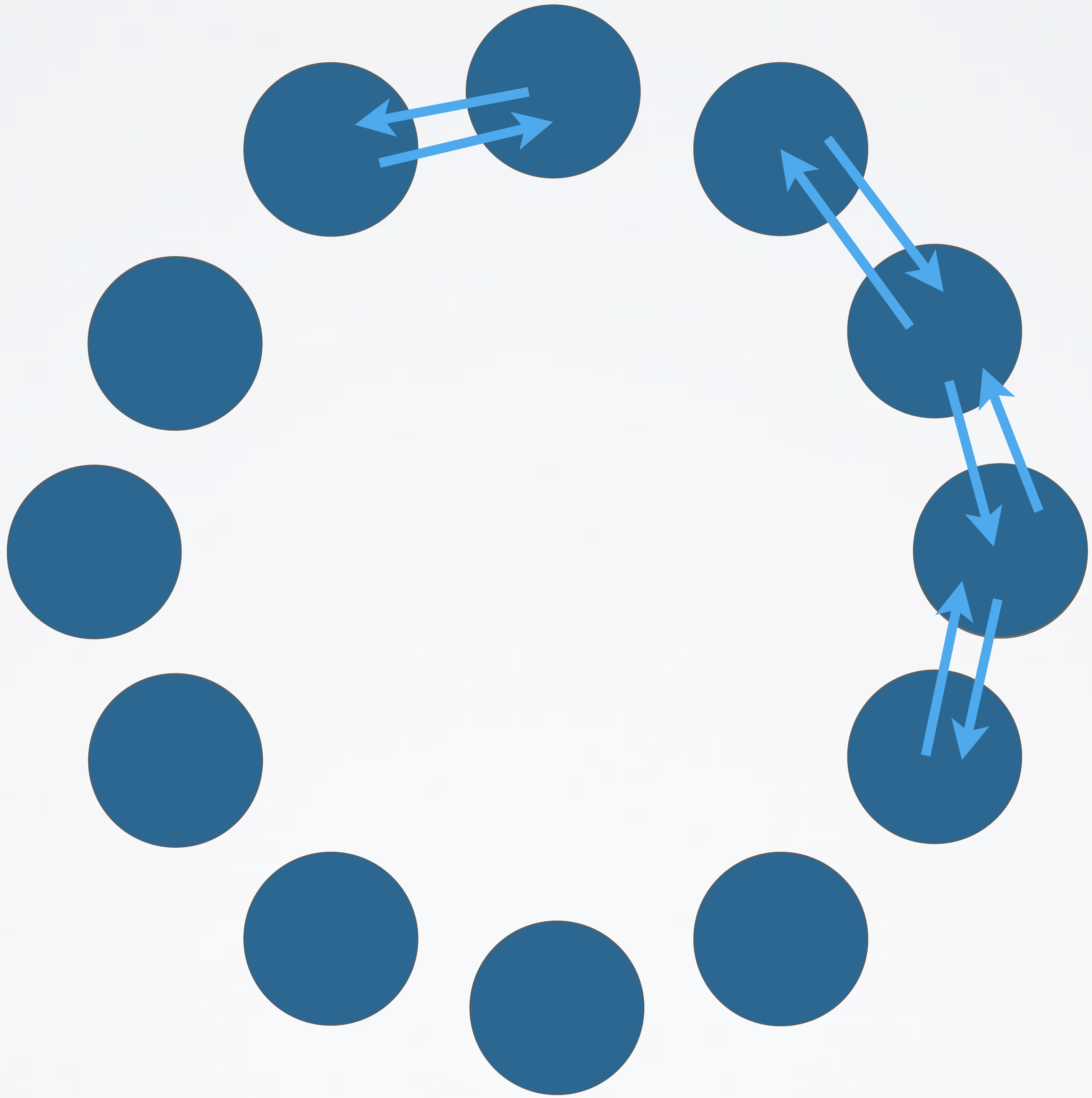



$N = 8192, L = O = 1$
Gossip
Tree
1.0
0.8
0.6
0.4
0.2
0.0
$c(t)/N$
10
15
20
25
30
35
t

STEP 1: GOSSIP



STEP 2: CORRECTION



FT COLLECTIVES

- Fault-tolerant **broadcast:** published[*]
- Fault-tolerant **Reduce** + **Allreduce**, collectives with builtin **fault-detection**
  - Formal analysis, measurements show: log-scalable, sturdy in most cases
- Resiliency for **tree-based collectives:**
  - Succeed / complete with failing nodes
  - Latency comparable to non-ft algorithms

[*] Torsten Hoefler, Amnon Barak, Amnon Shiloh and Zvi Drezner, "Corrected Gossip Algorithms for Fast Reliable Broadcast on Unreliable Systems", IPDPS'17, Orlando, FL, USA

SUMMARY

- **Decoupled execution:** low noise + latency
- **Checkpointing:** Coordinated + optimized
- **Diffusion:** Promising
- **Corrected Gossip & Trees:** fault-tolerant collective operations (maybe for MPI)
- **Integrated:** gossip + decision making
- **WIP:** integrate monitoring + migration

DFG Deutsche Forschungsgemeinschaft
SPPEXA German Priority Programme 1648 Software for Exascale Computing