TECHNISCHE UNIVERSITÄT DRESDEN

ZIB

The Hebrew University of Jerusalem

A MICROKERNEL-BASED OPERATING SYSTEM FOR EXASCALE COMPUTING

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

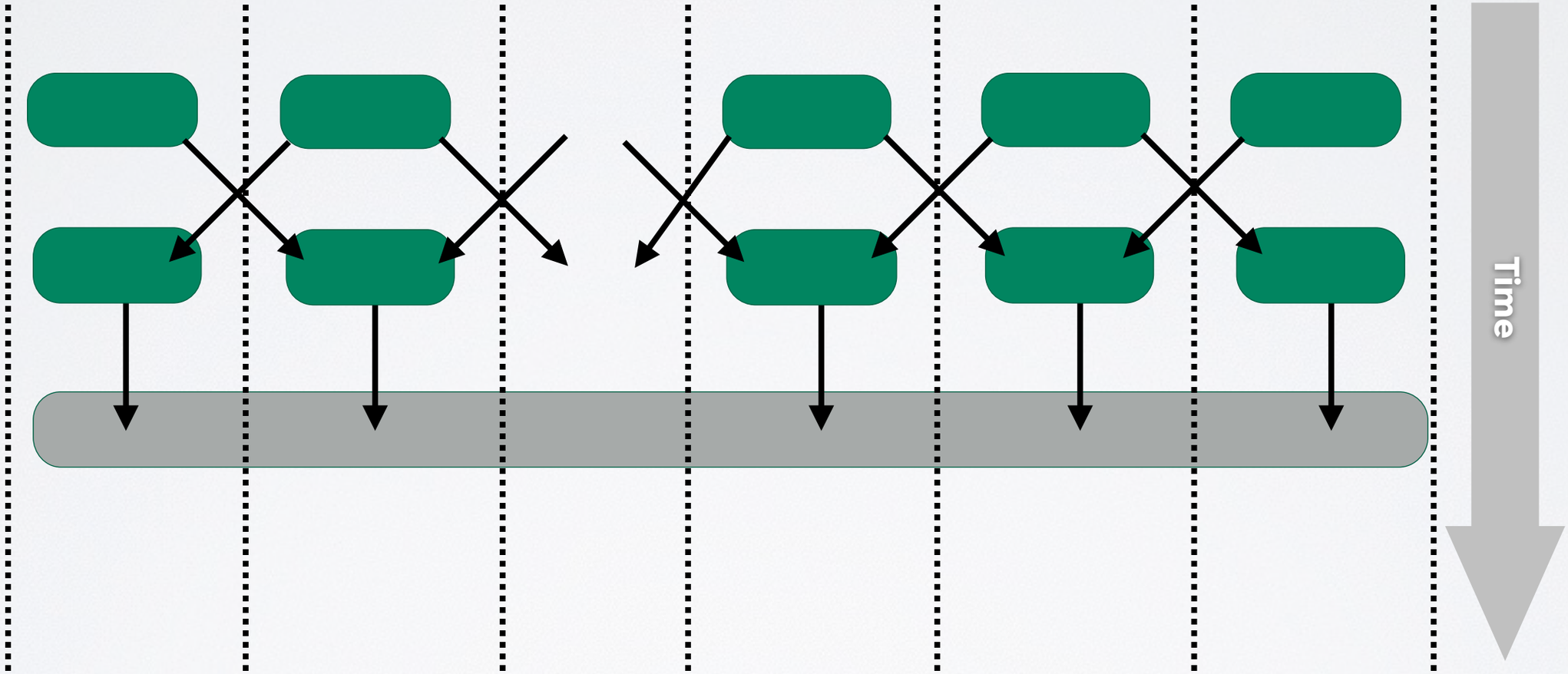
CARSTEN WEINHOLD, TU DRESDEN

TRADITIONAL HPC

The ideal world assumption:

- Identical, predictable, and reliable nodes
- Fast and reliable interconnect
- Balanced applications
- Isolated partitions of fixed size

TRADITIONAL HPC



Fixed-size chunks of work

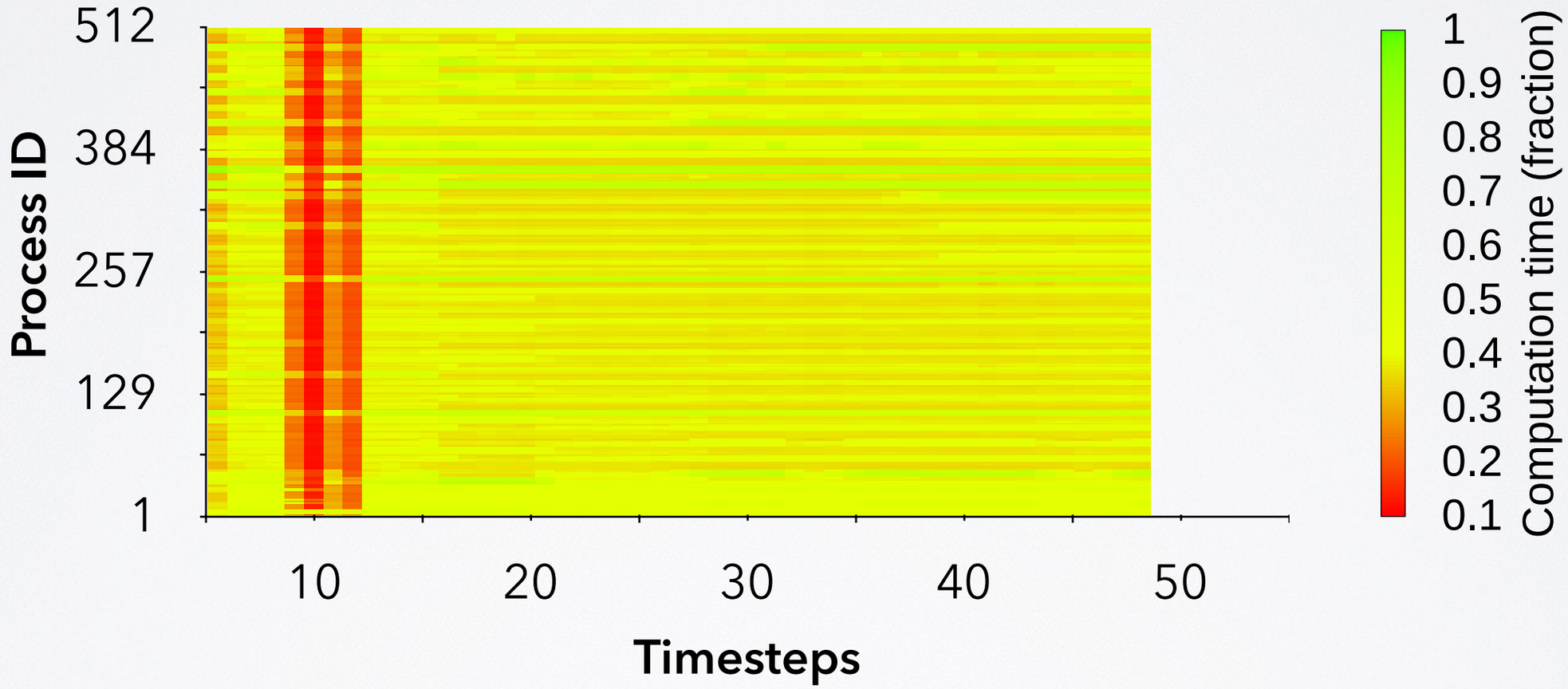
One thread per core

Systems software: optimize communication latency

- Message passing uses polling
- Batch scheduler for start / stop
- Separate servers for I/O
- Small OS on each node

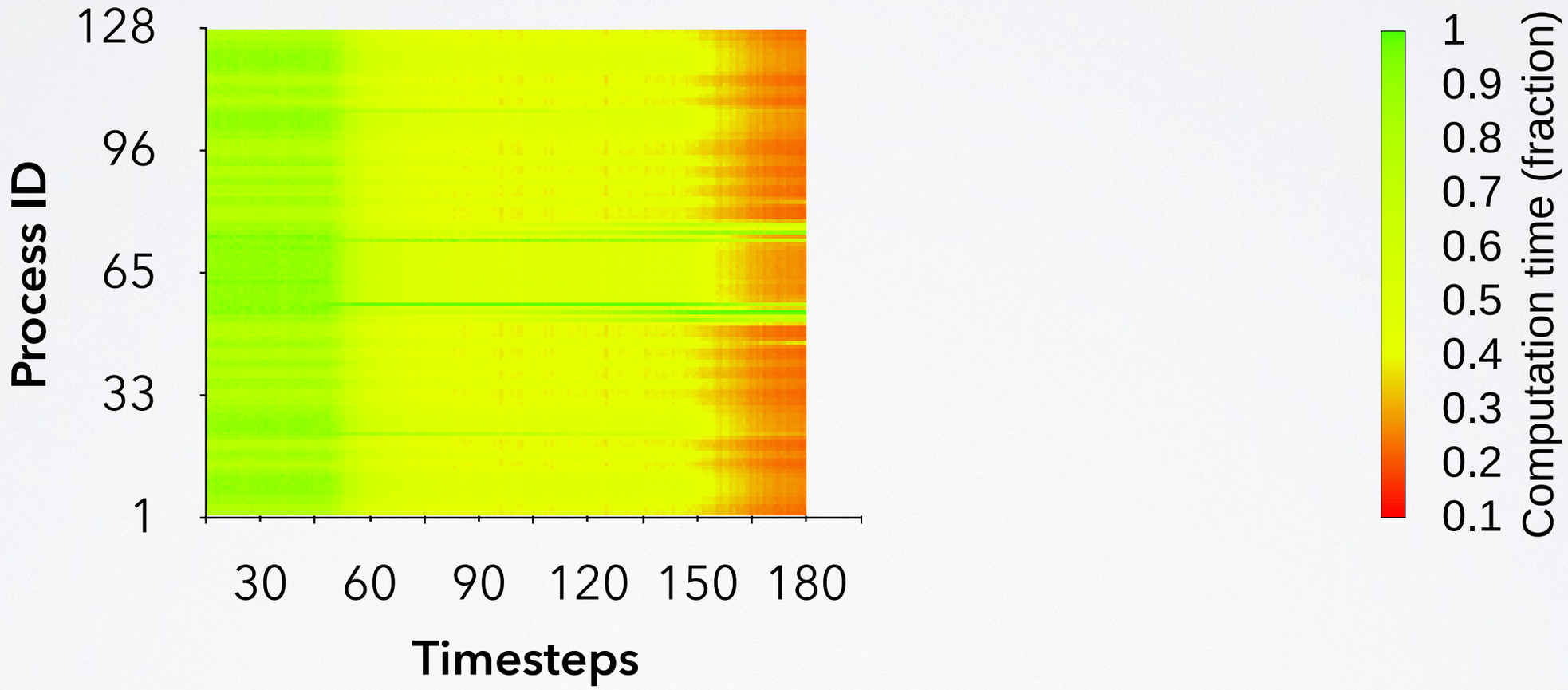
No OS on critical path

Application: CP2K



Computation–communication ratio of CP2K on 512 cores

Application: COSMO-SPECS+FD4



Computation–communication ratio of COSMO-SPECS+FD4 on 128 cores

Application: COSMO-SPECS+FD4

Unbalanced compute times of ranks per time step

Hand balanced compute times of ranks per time step

Computation–communication ratio of COSMO-SPECS+FD4 on 128 cores

Application: PRIME

Unbalanced compute times of ranks per time step

Now think of:

- Composite applications
- In-situ visualization, etc.

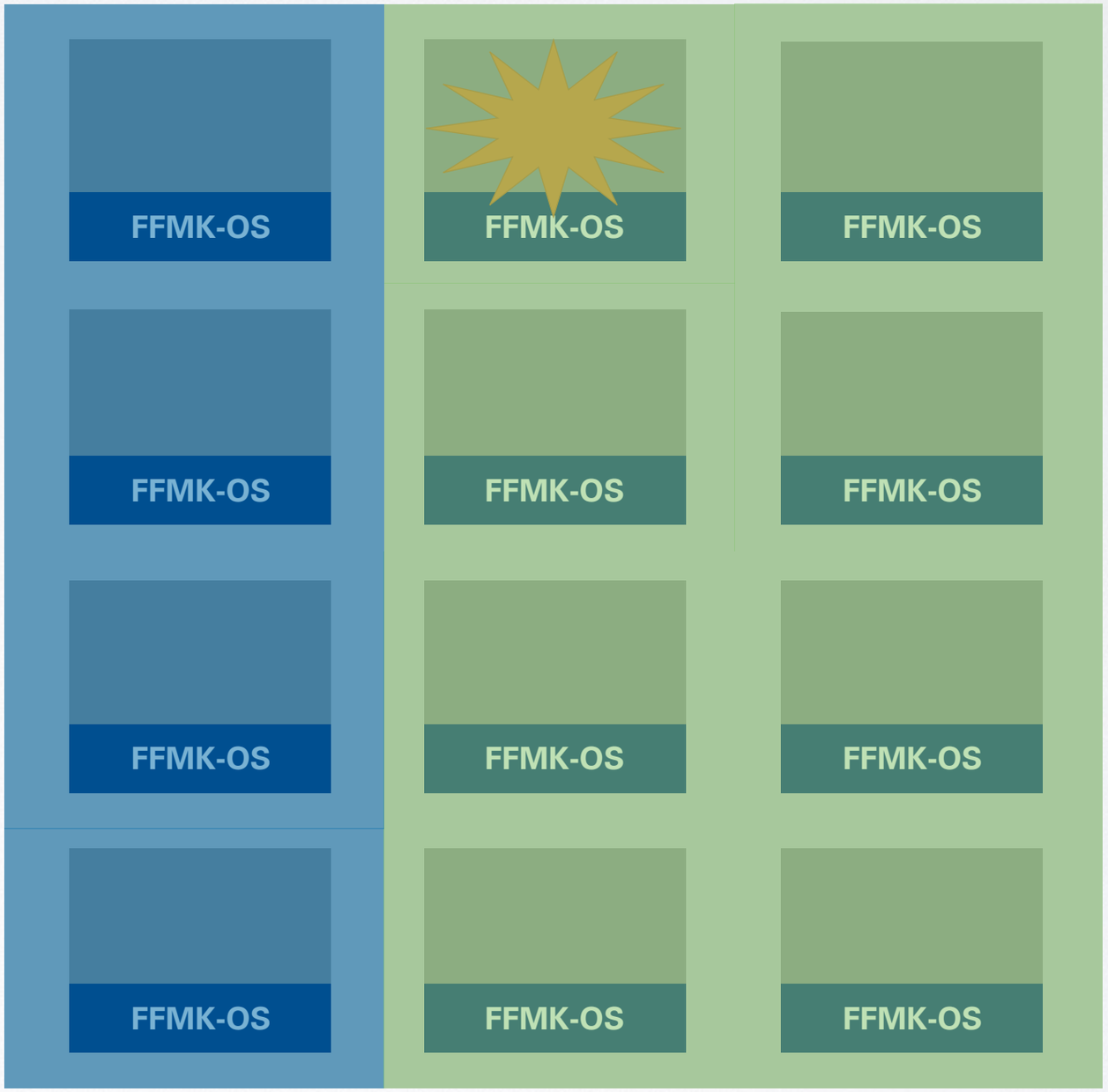
FFMK: A Fast and Fault-Tolerant Microkernel-Based Operating System for Exascale Computing

DFG Deutsche Forschungsgemeinschaft

SPPEXA German Priority Programme 1648
Software for Exascale Computing

CHALLENGES

- Dynamic applications & platforms
- Increased fault rates
- Power / Dark silicon
- Heterogeneity (cores, memory, …)

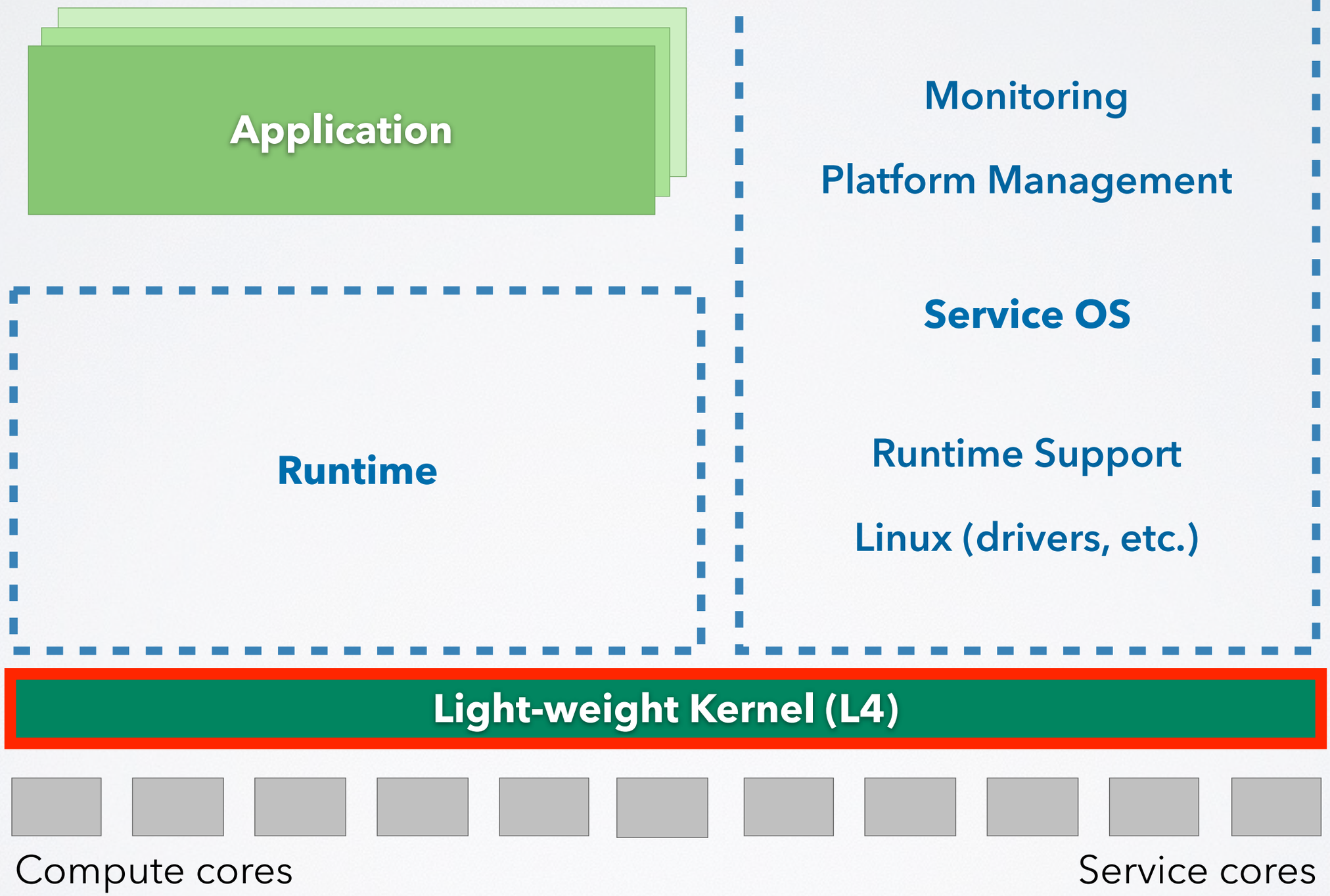


TECHNISCHE UNIVERSITÄT DRESDEN

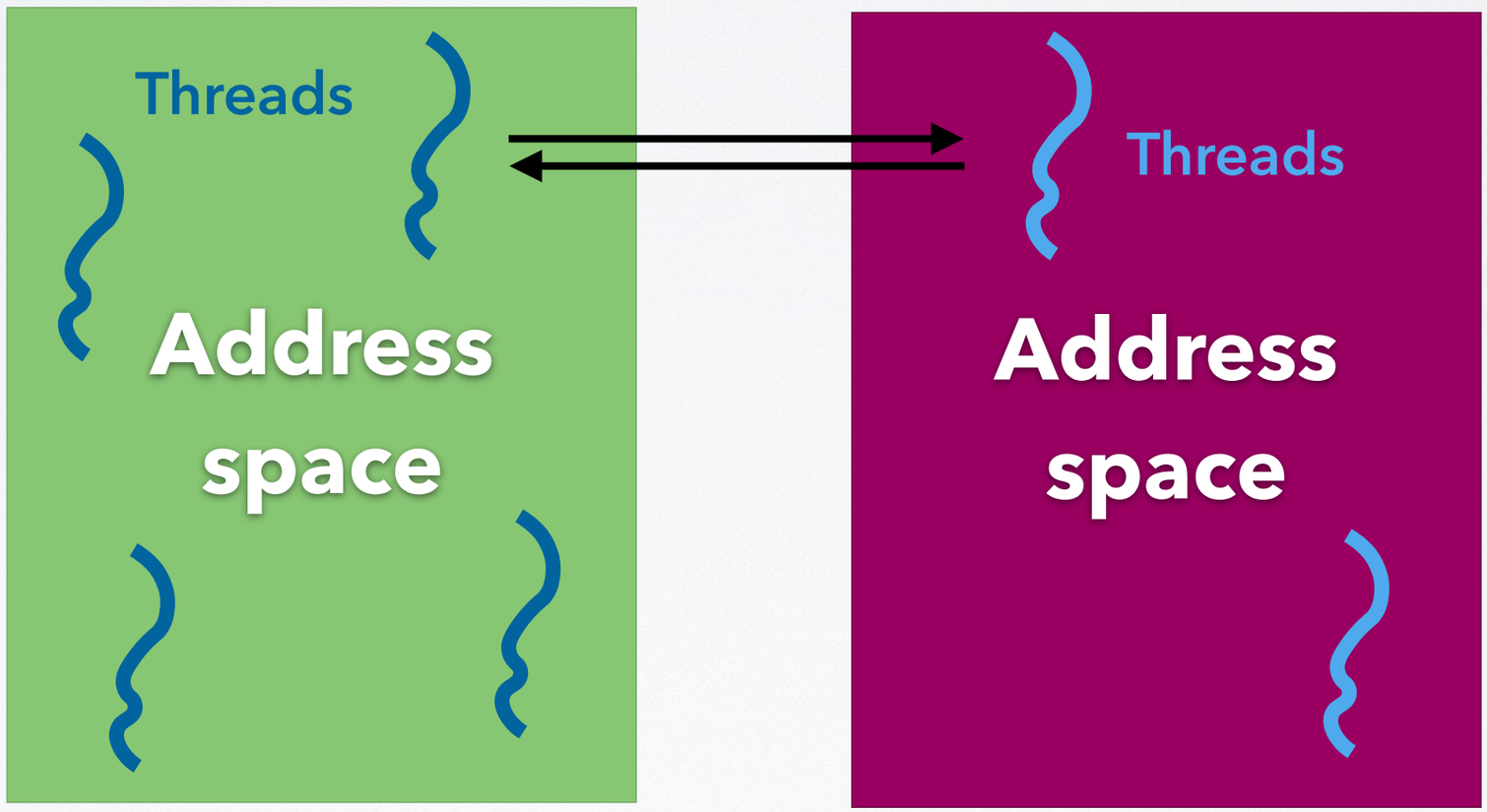
ZIB

The Hebrew University of Jerusalem

NODE ARCHITECTURE

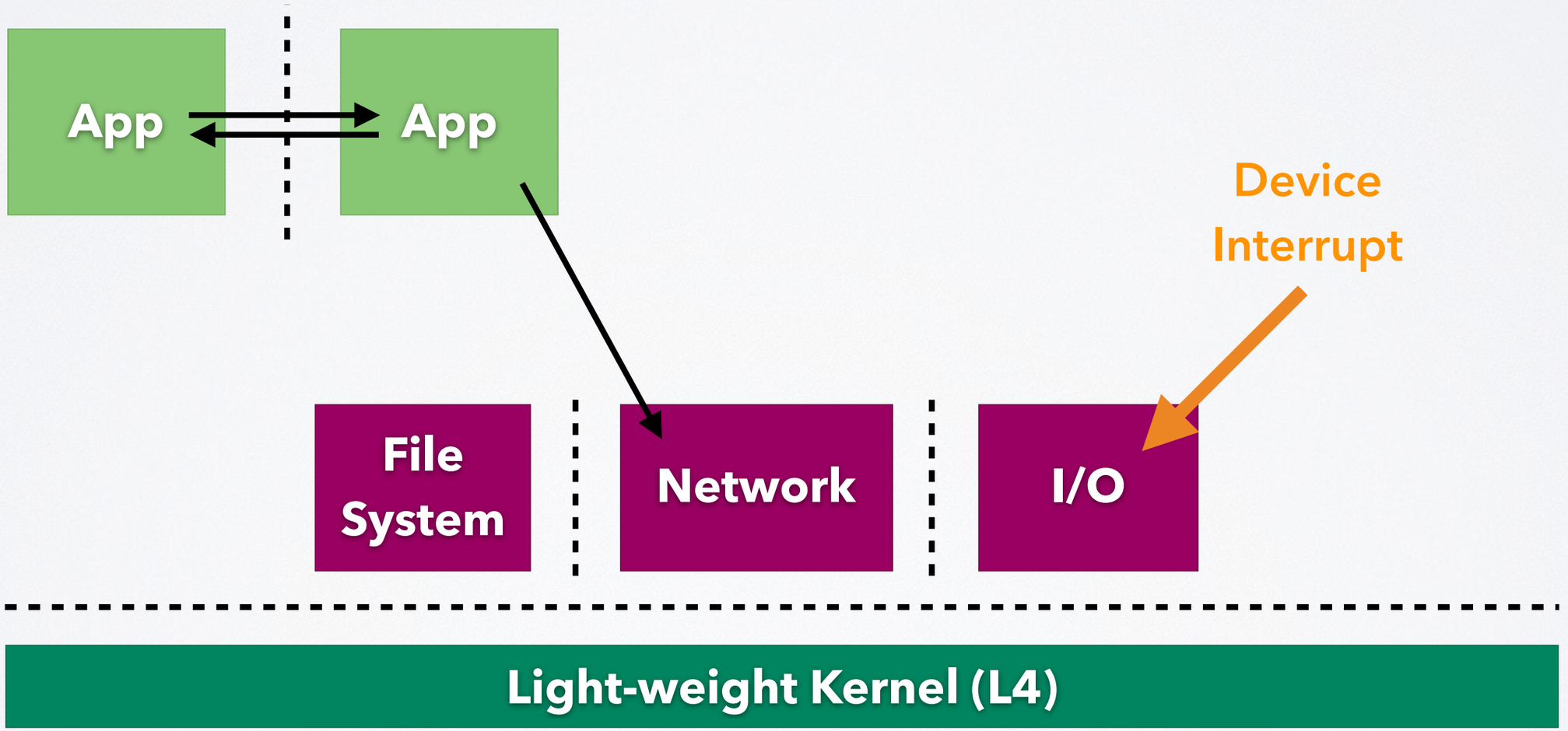


3 ABSTRACTIONS

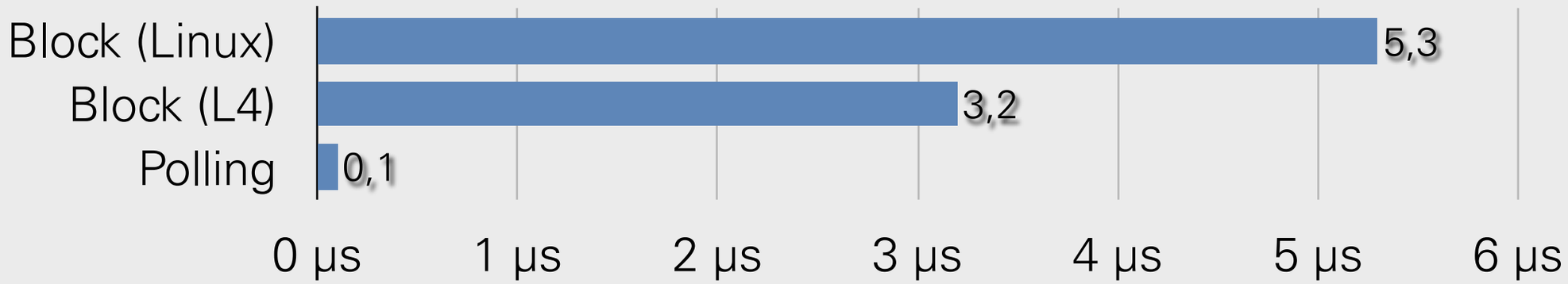


Light-weight Kernel (L4)

MESSAGE PASSING



BLOCK, POLL, IRET

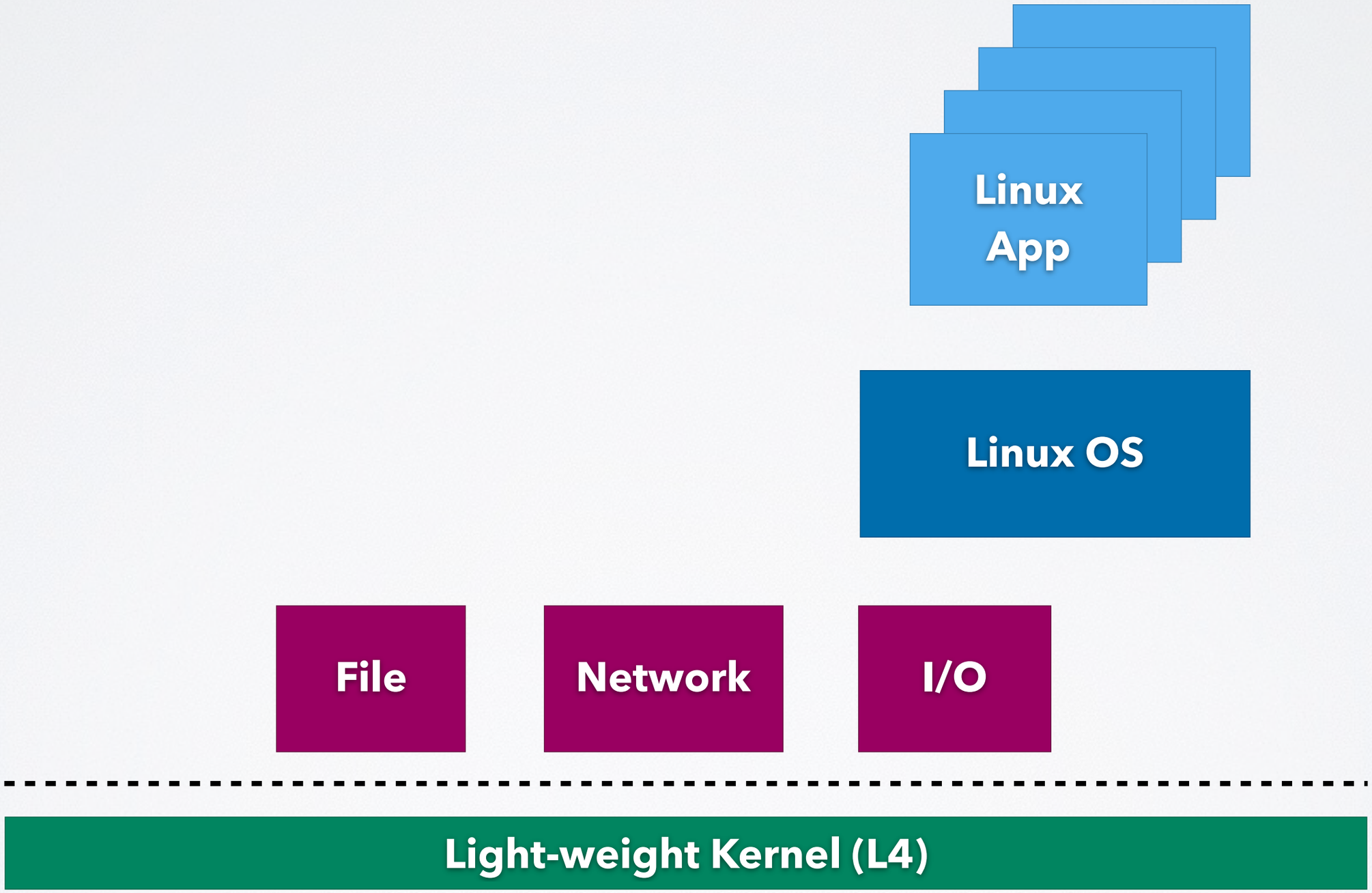


- Intel Core i7 3770S @ 3100 MHz
- No Hyperthreading, no Turboboost
- No dynamic power management
- Same socket
- 64-bit Linux 3.11.6 (cpuidle.off=0, intel_idle.max_cstate=0)
- 64-bit L4/Fiasco.OC

Wake from interrupt on L4: 900 cycles, **0.3 µs**

(best case, on Intel Core i7-4770 CPU @ 3.40GHz)

LINUX ON L4



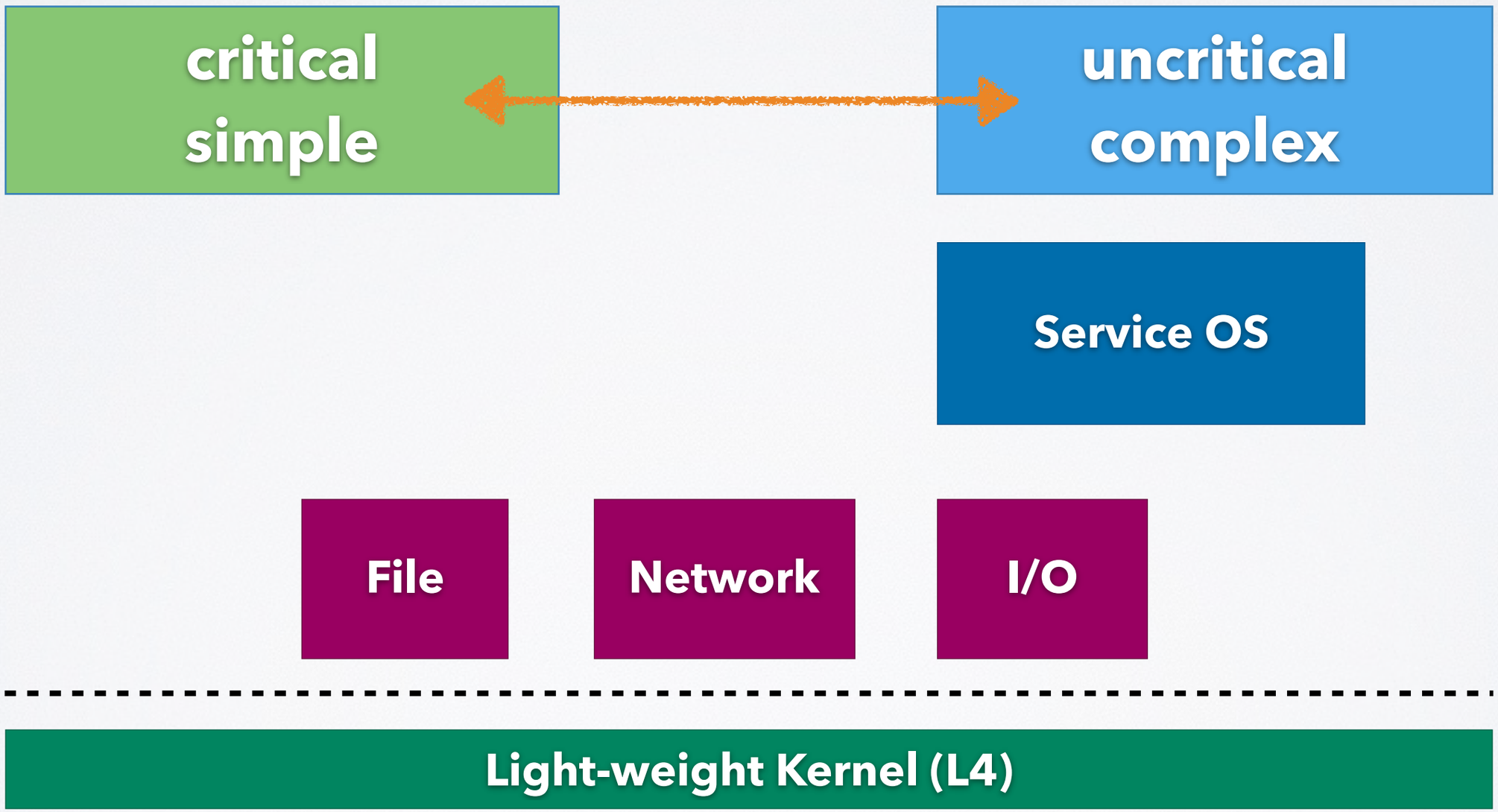
HYBRID SYSTEM

Real-time

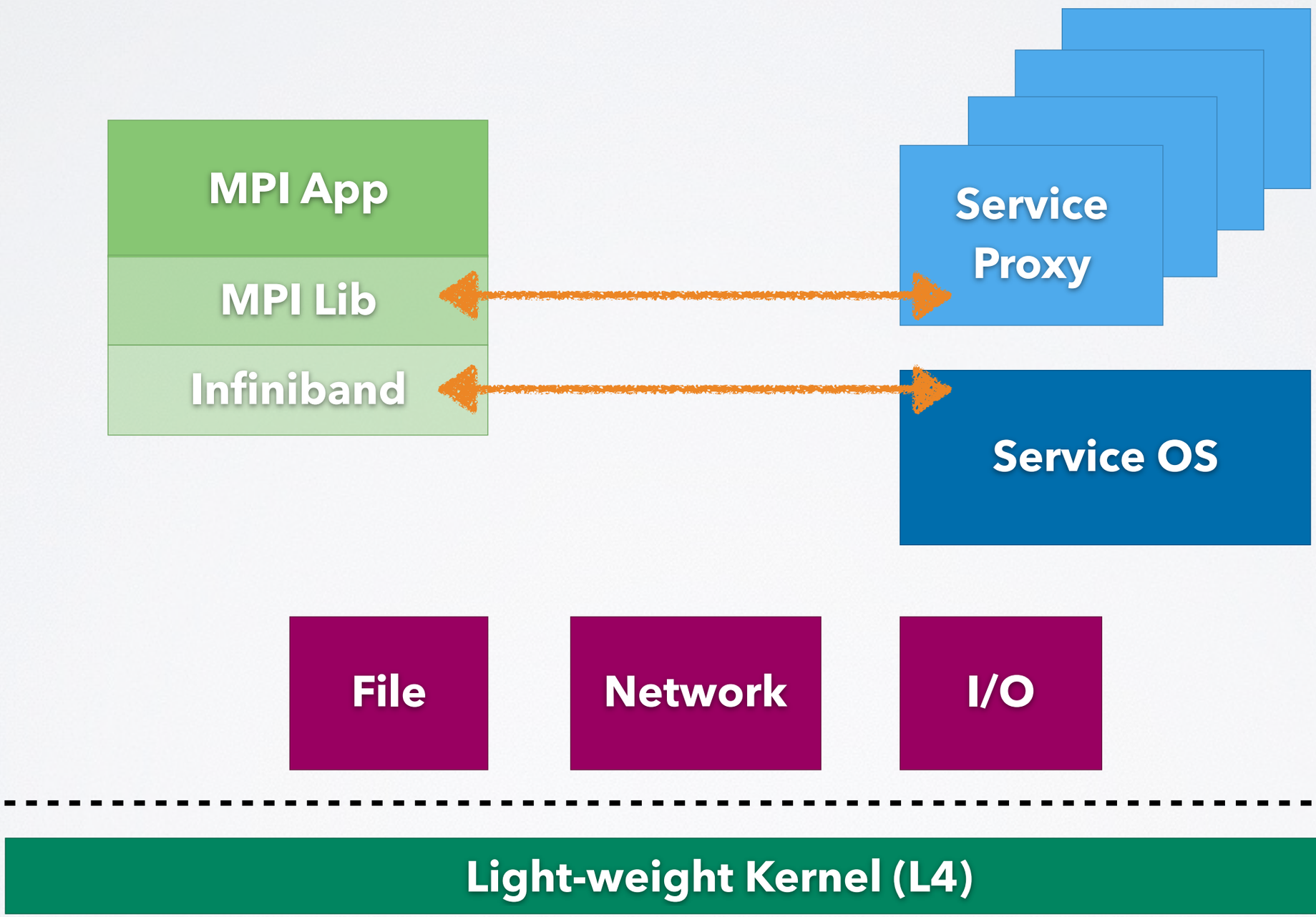
Security: small Trusted Computing Base

Resilience: small Reliable Computing Base

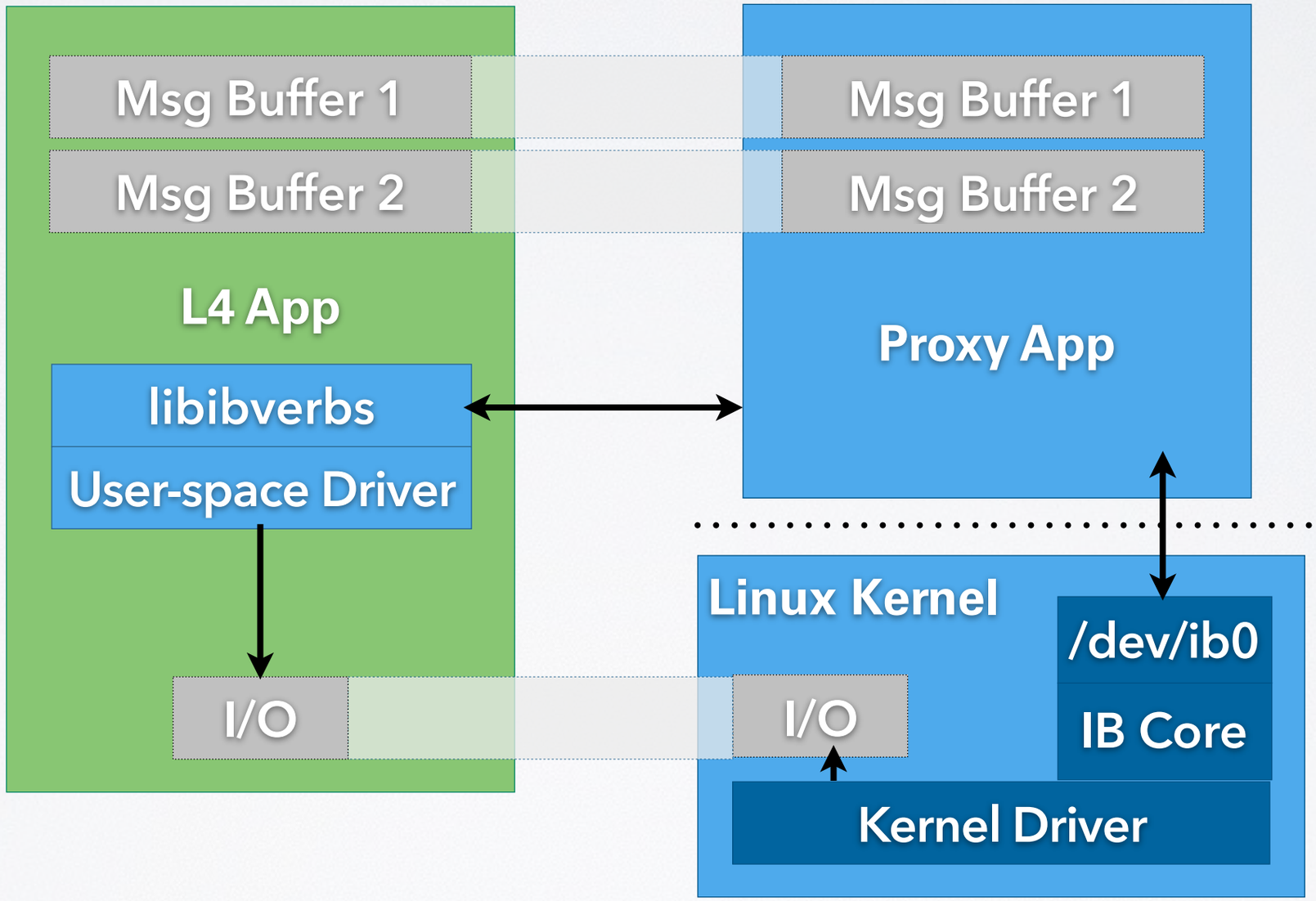
HYBRID SYSTEM



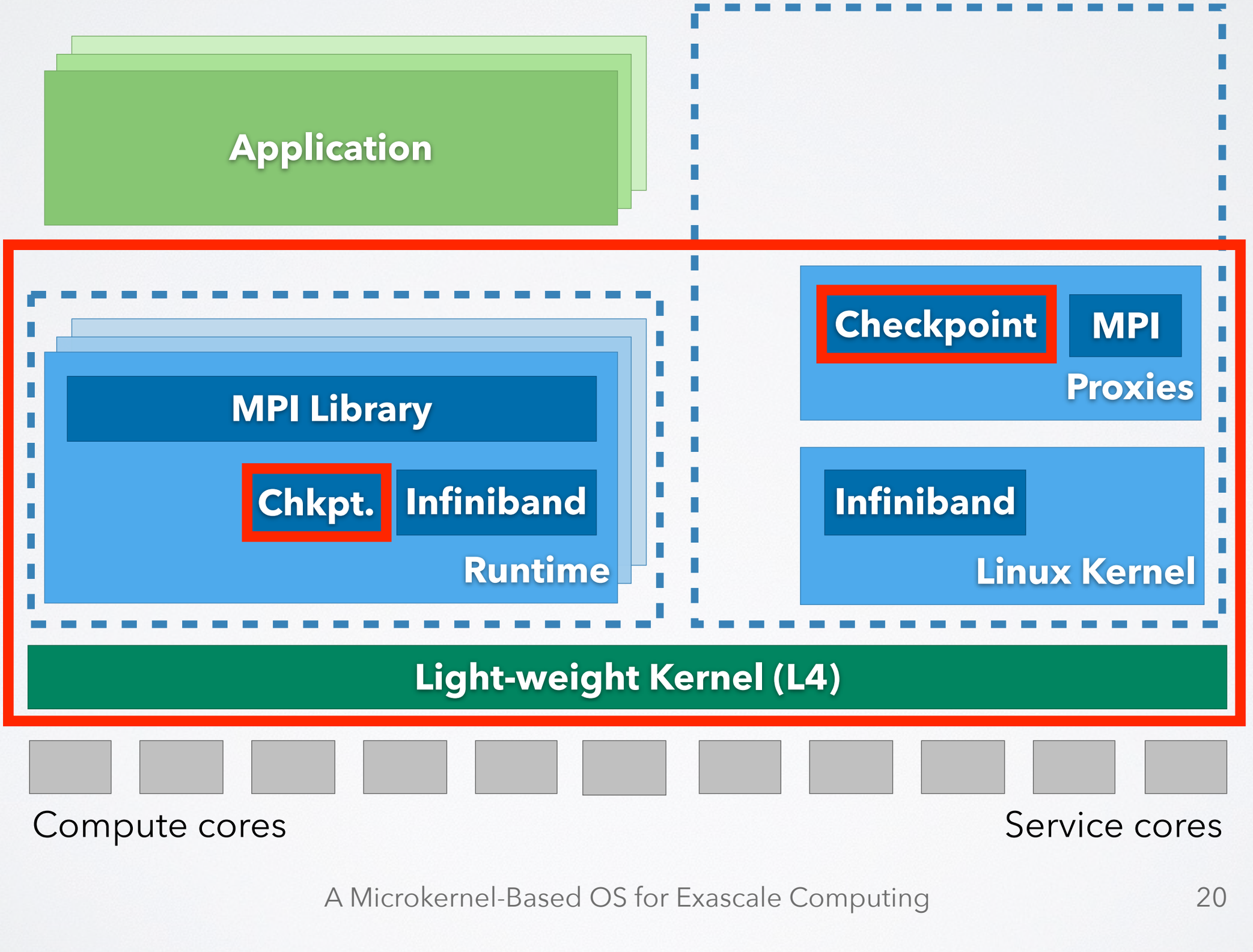
HYBRID SYSTEM



DRIVER REUSE



NODE ARCHITECTURE

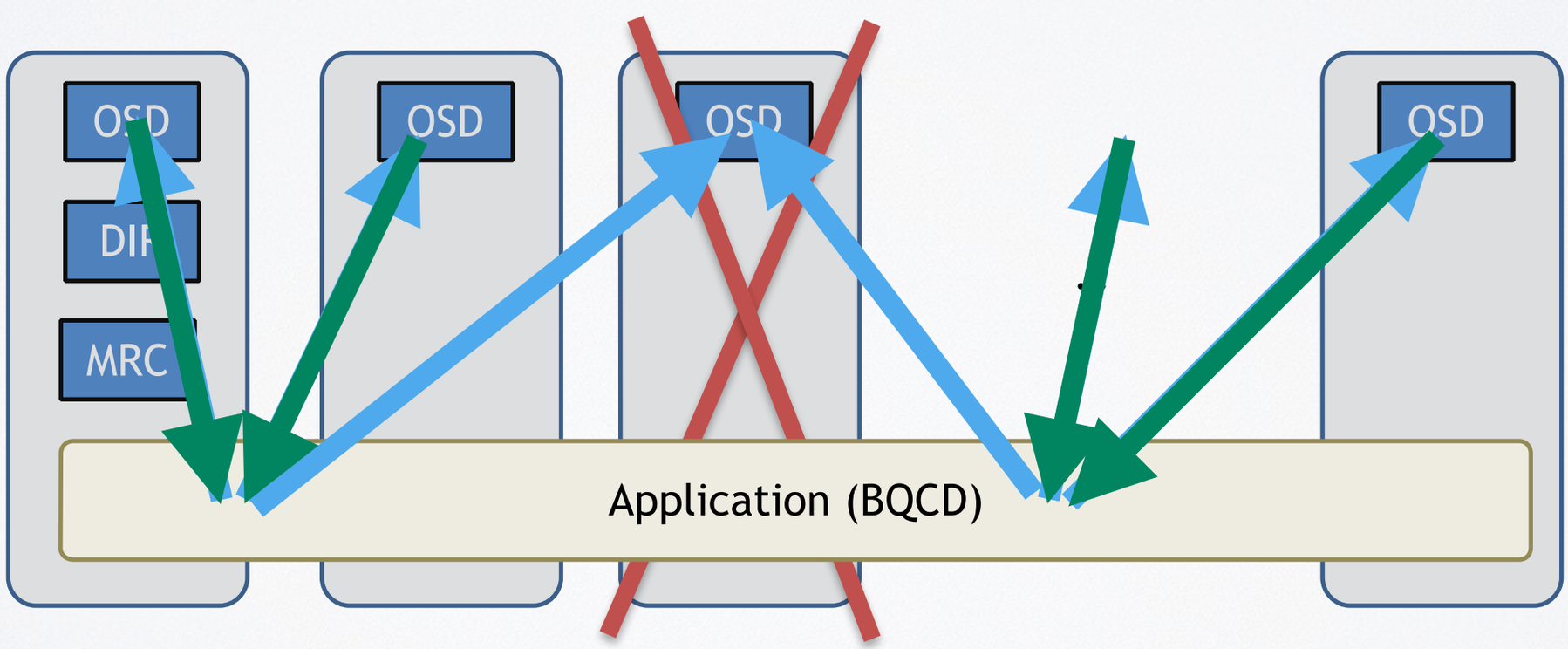


FAILURE RATES

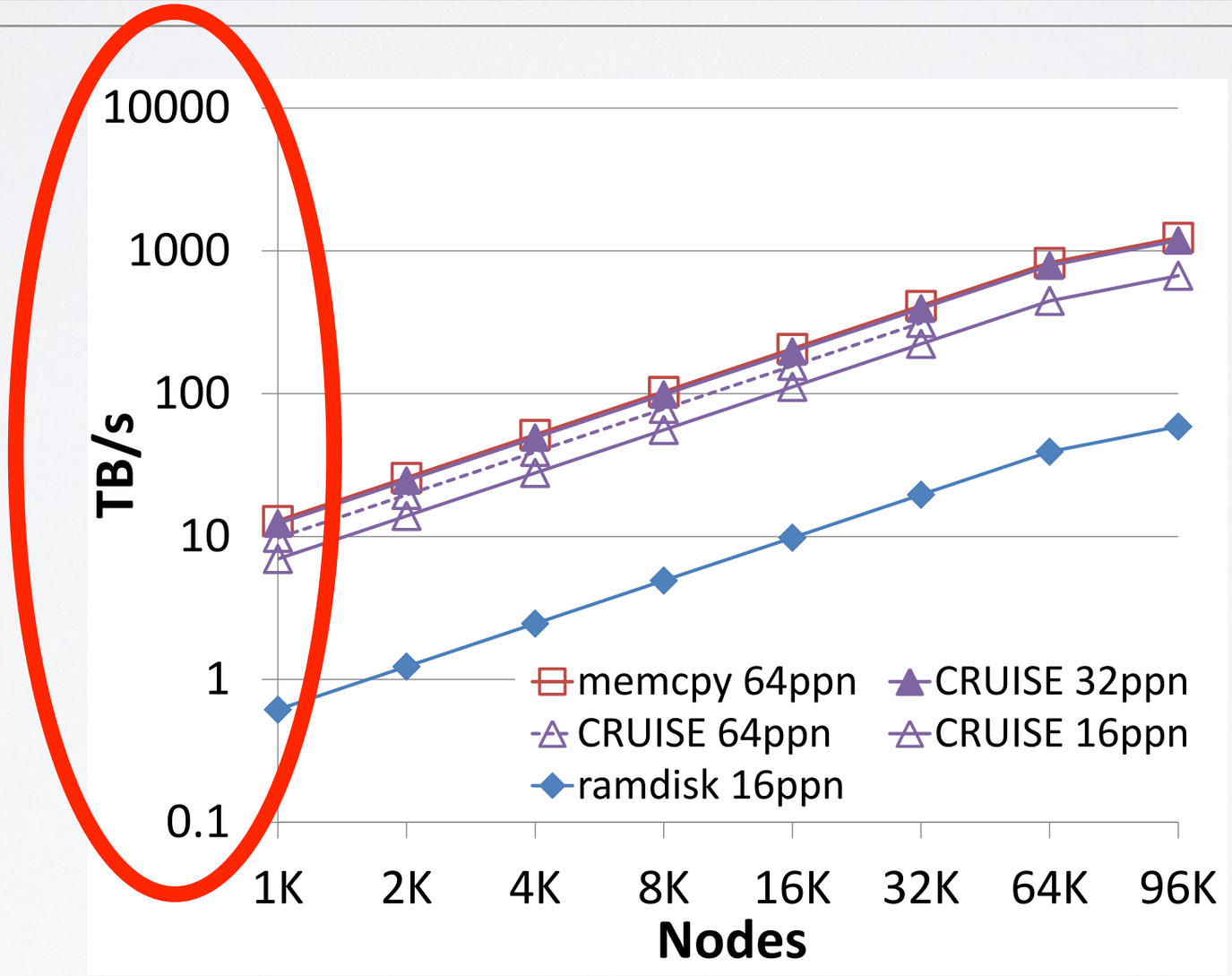
MTBF for Component Failures in an HPC System		
Failed Component	**# of Nodes Affected**	**MTBF**
PFS, core switch	1408	65.10 days
Rack	32	86.90 days
Edge switch	16	17.37 days
PSU	4	28.94 days
Compute nodes	1	15.8 hours

Kento Sato et al., „Design and Modeling of a Non-blocking Checkpointing System“, SC'12

- **In-memory XtreemFS volume** for application-level checkpointing
- **RAID-5 erasure coding:** recovery with 1 failed OSD
- Demonstrator running BQCD code on a Cray XC30



BANDWIDTH



(b) Sequoia Cluster (IBM Blue Gene/Q)

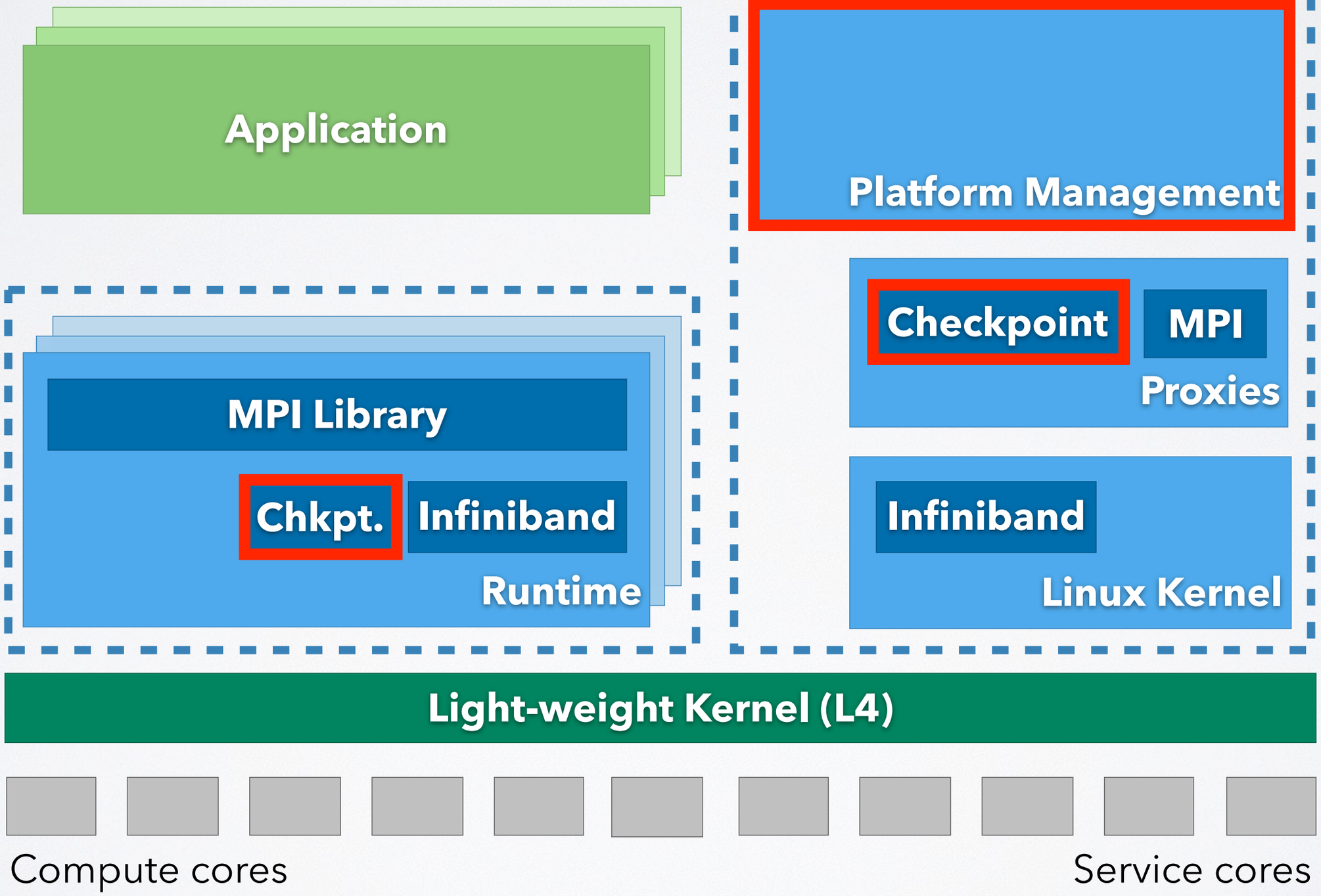
Raghunath Rajachandrasekar et al., A 1 PB/s File System to Checkpoint Three Million MPI Tasks, HPDC'13

TECHNISCHE UNIVERSITÄT DRESDEN

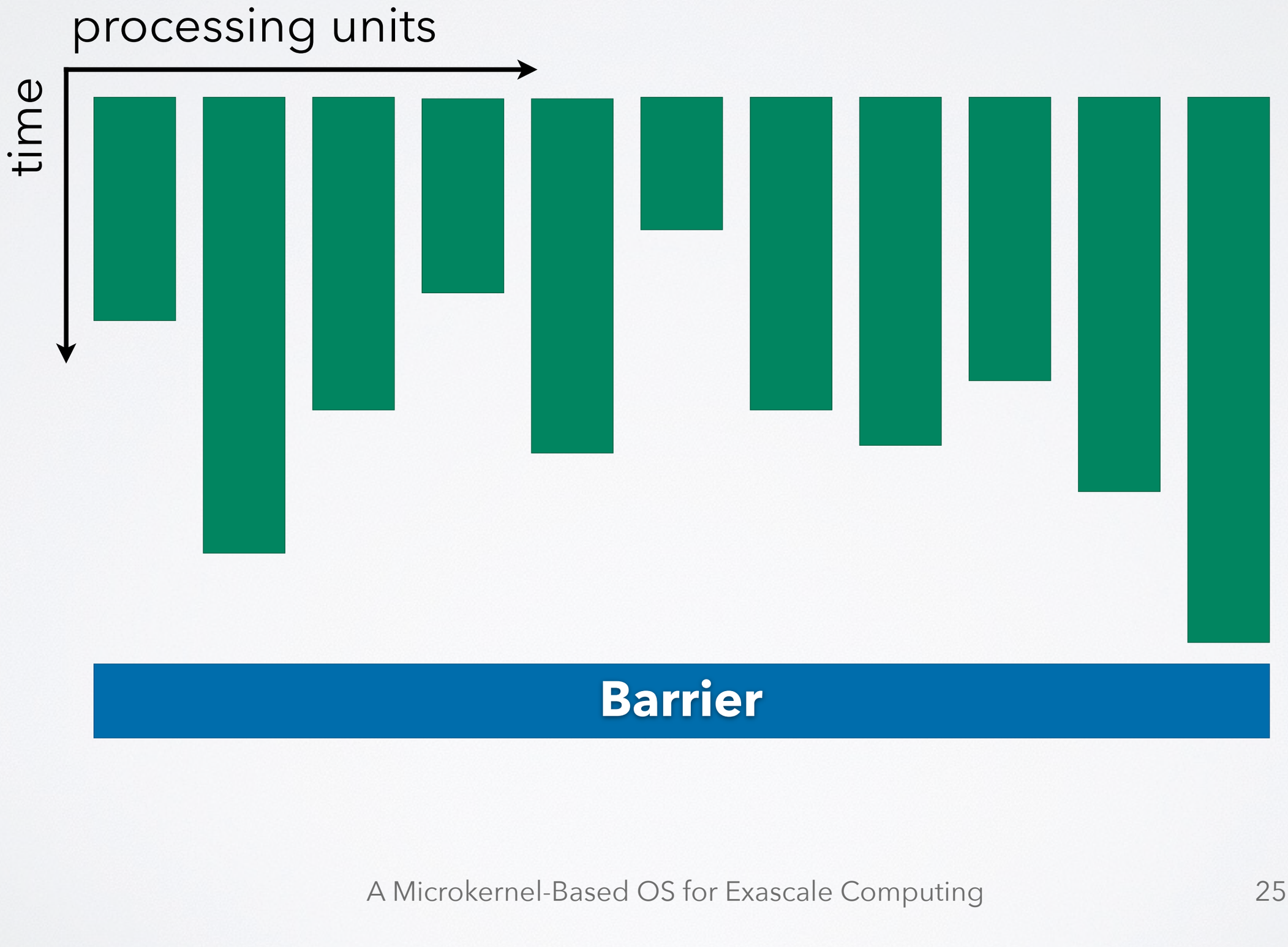
ZIB

The Hebrew University of Jerusalem

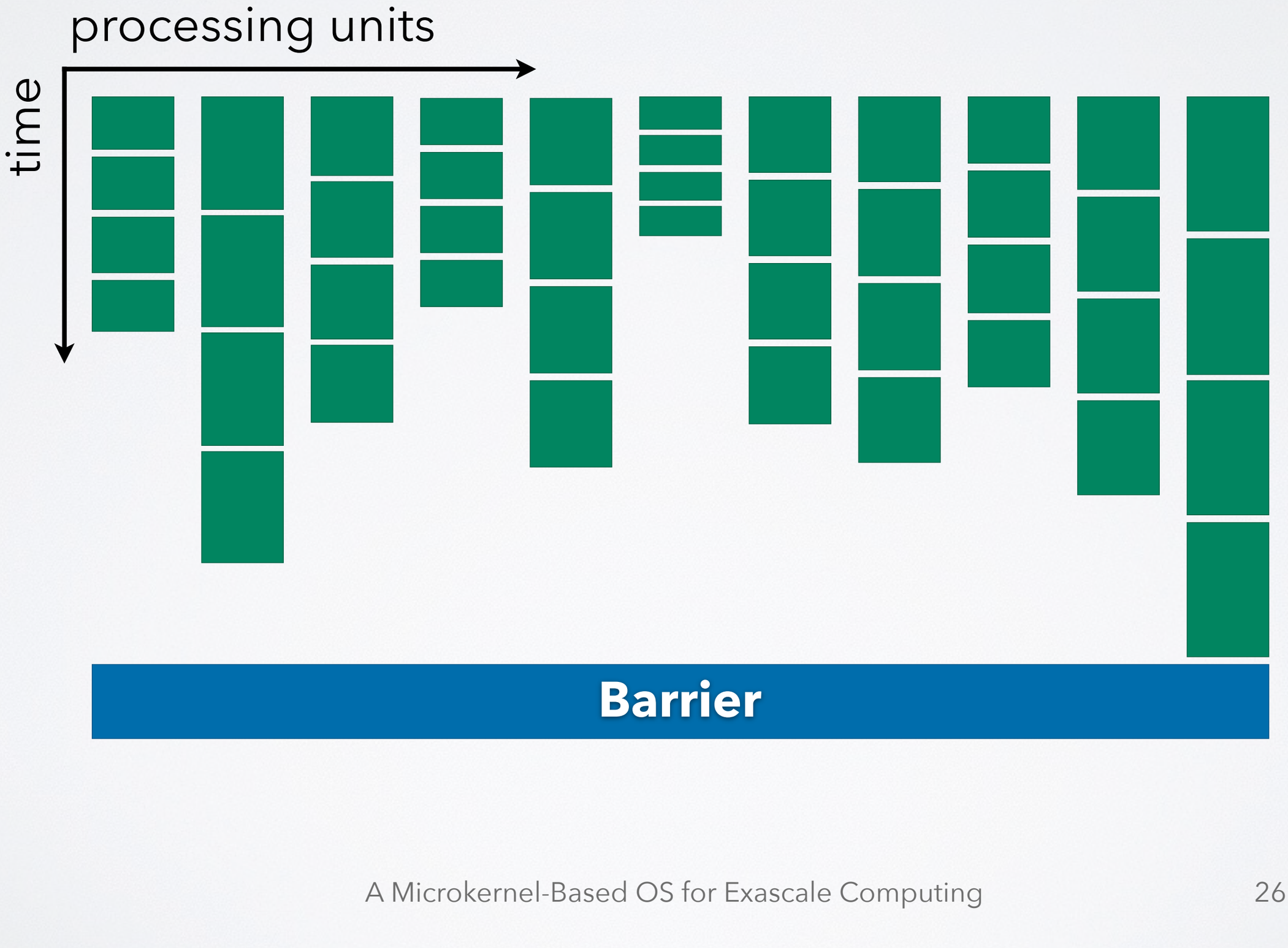
NODE ARCHITECTURE



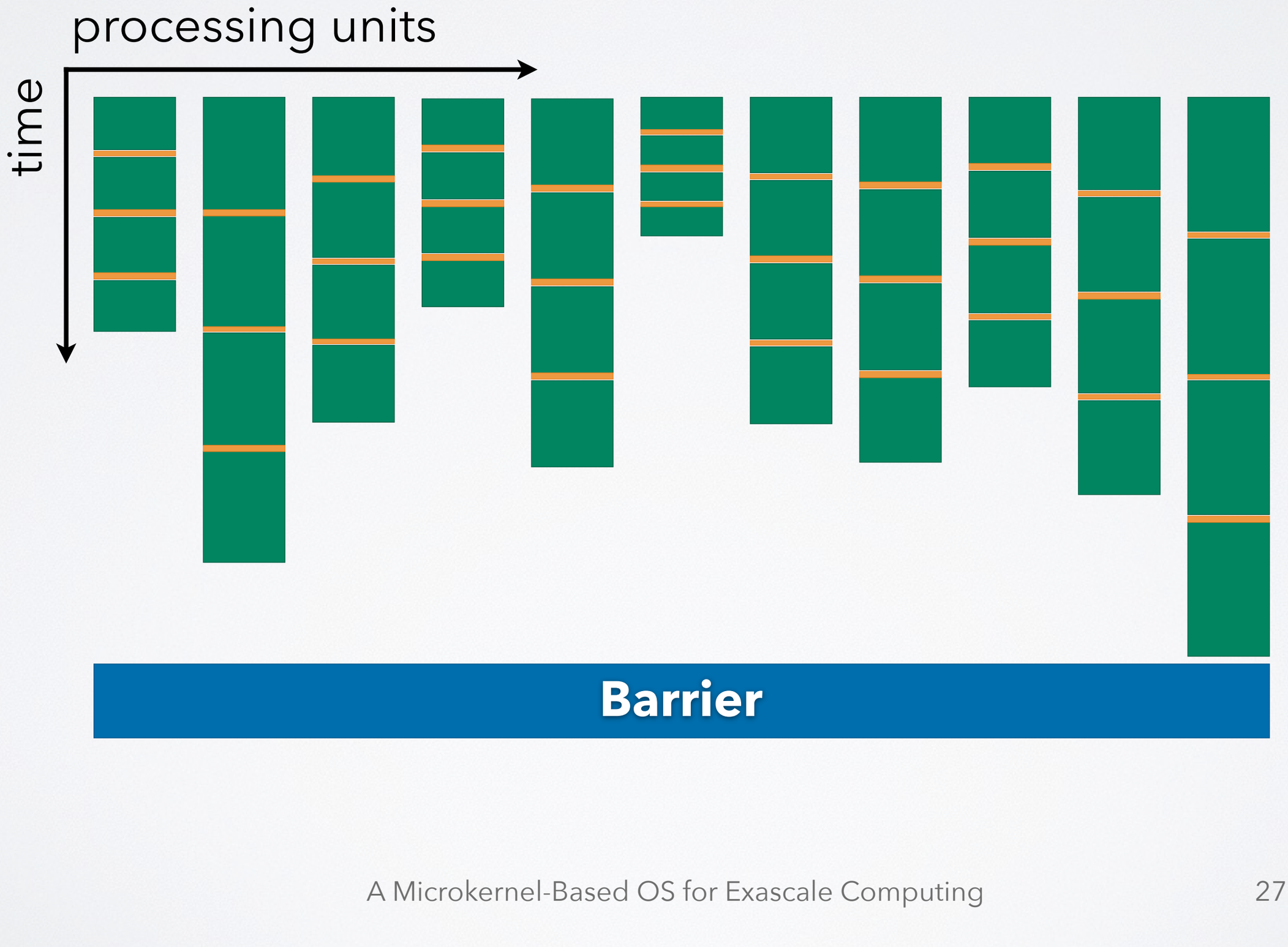
OVERDECOMPOSITION



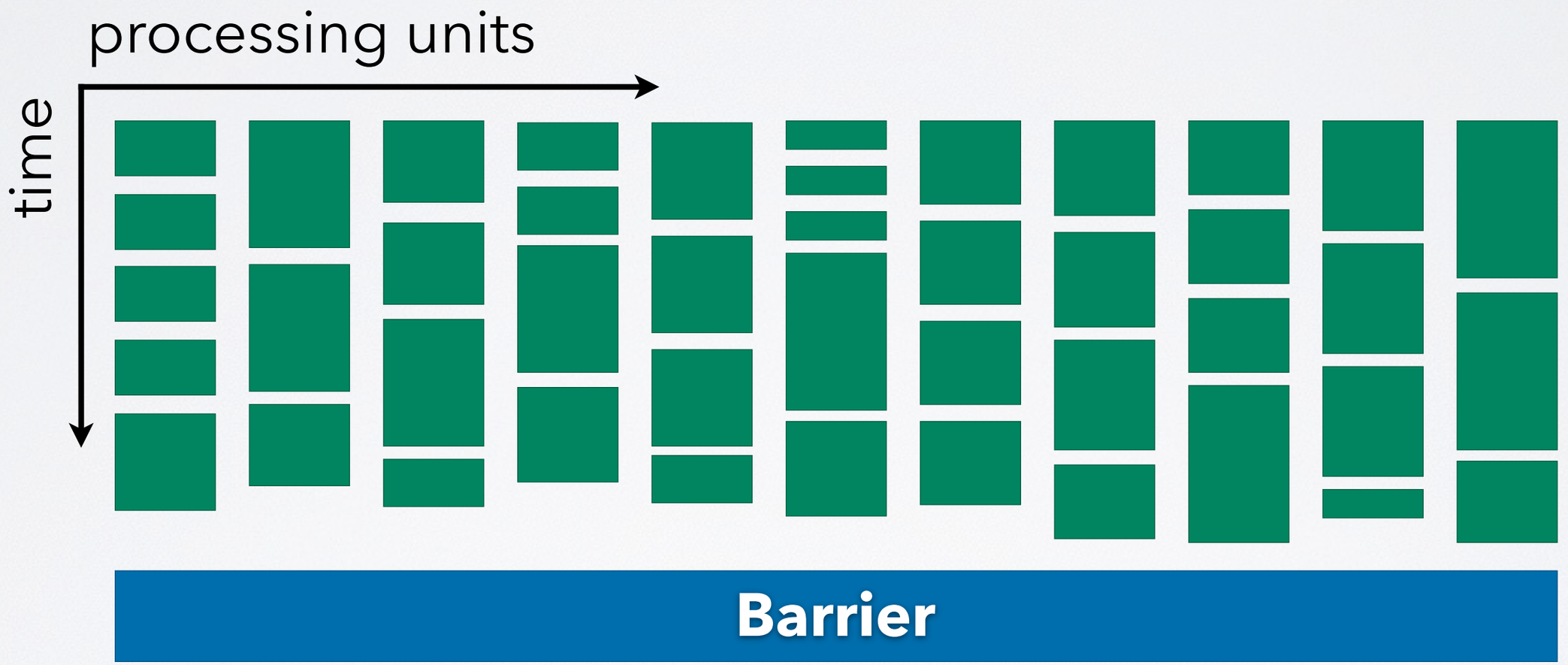
OVERDECOMPOSITION



OVERDECOMPOSITION



OVERDECOMPOSITION

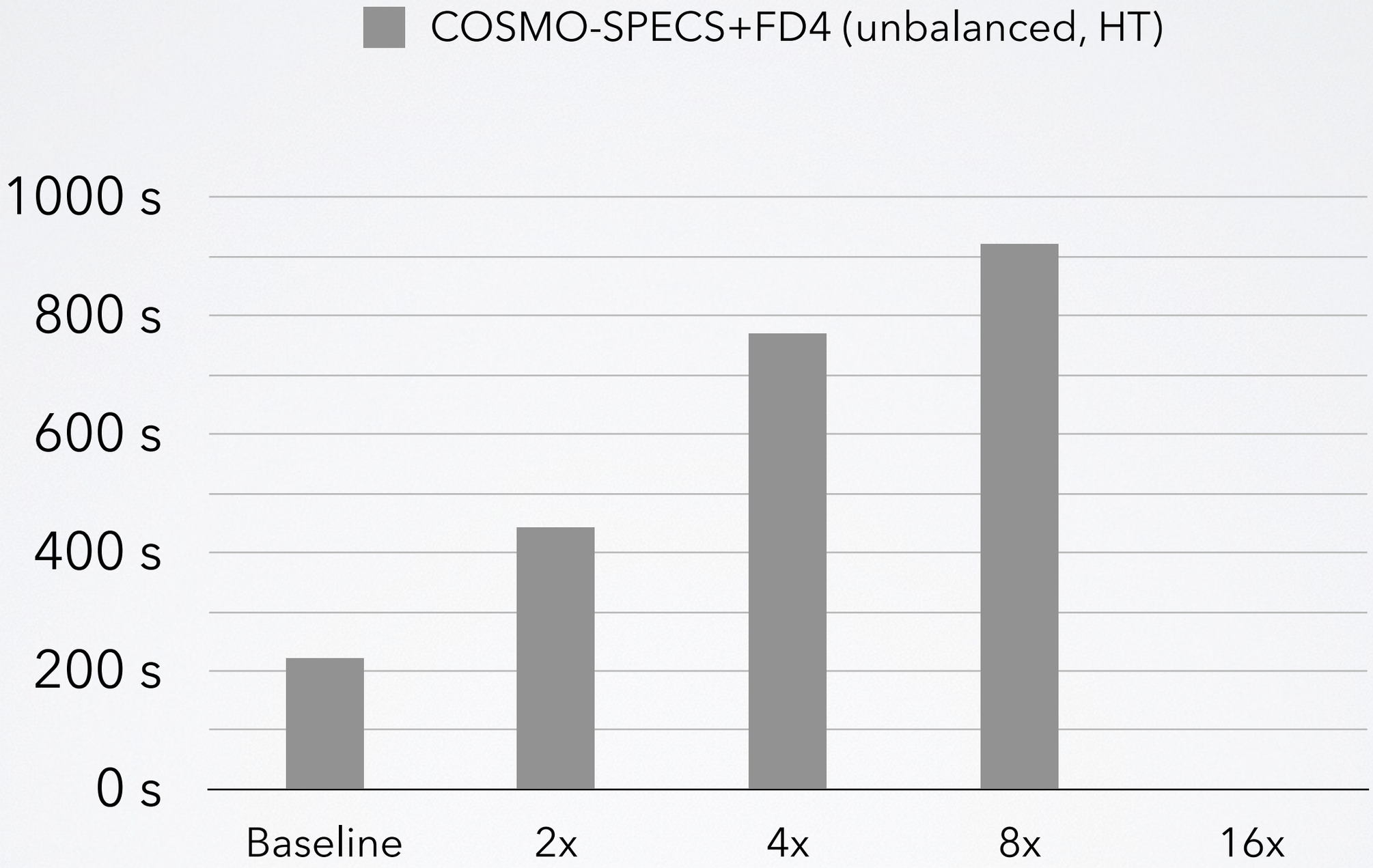


TECHNISCHE UNIVERSITÄT DRESDEN

ZIB

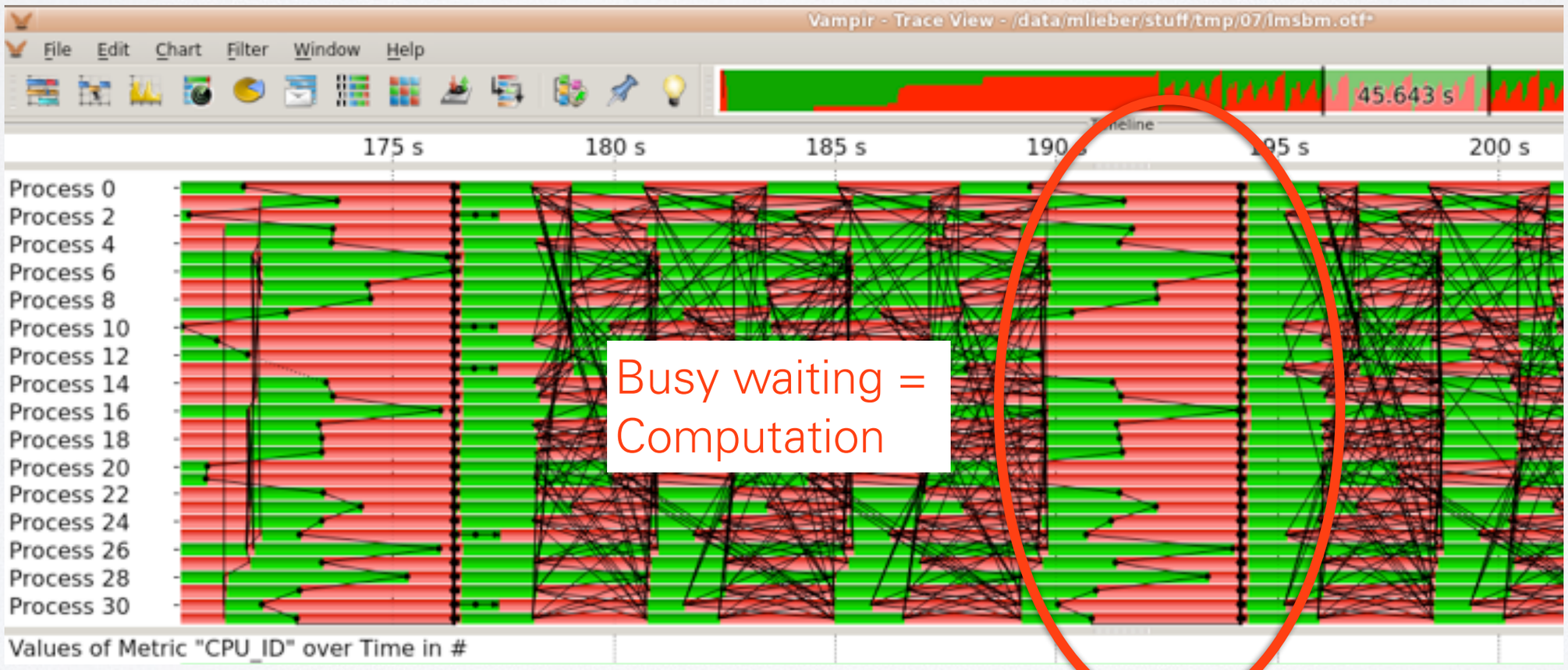
The Hebrew University of Jerusalem

OVERDECOMPOSITION

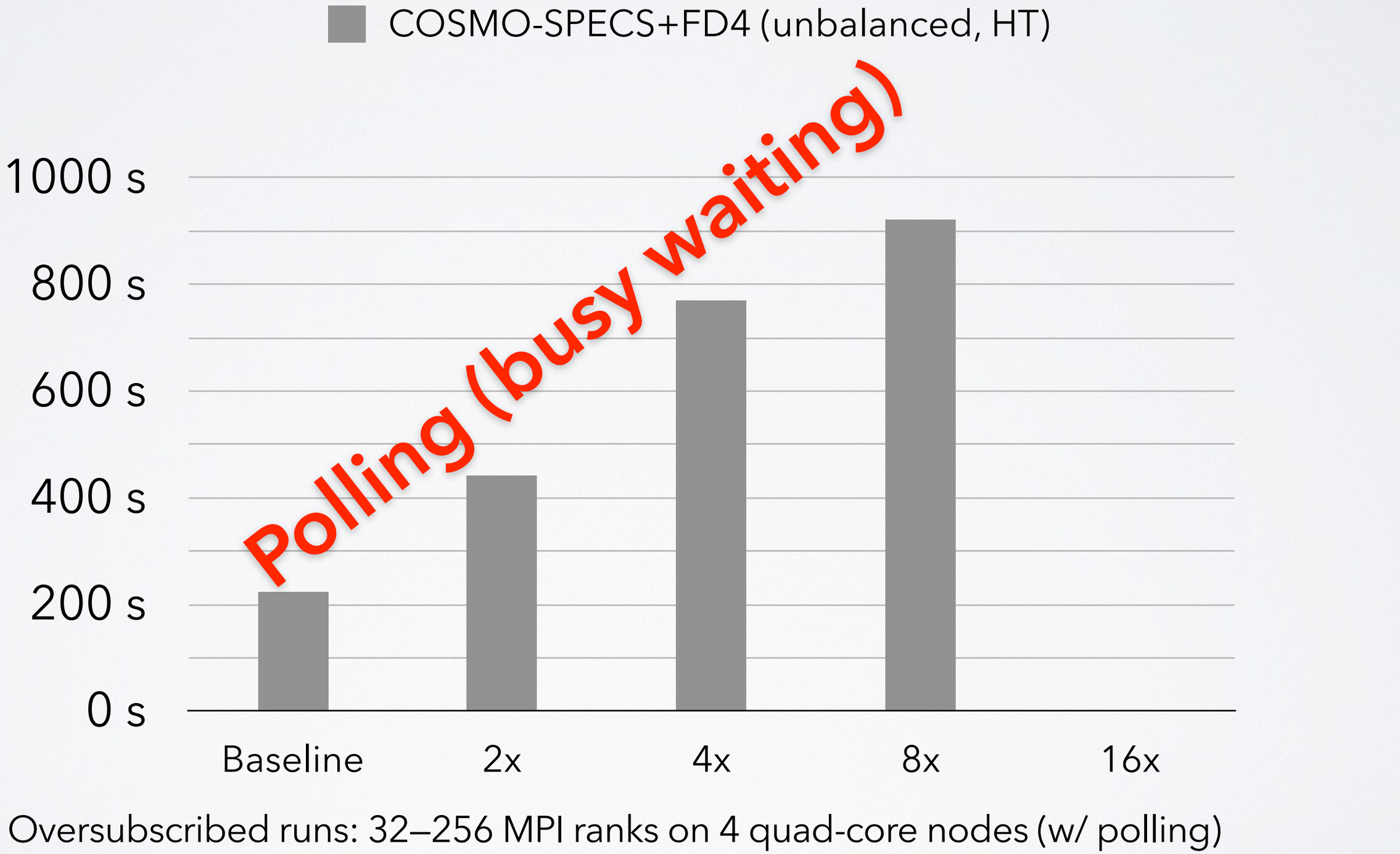


Oversubscribed runs: 32–256 MPI ranks on 4 quad-core nodes (w/ polling)

BUSY WAITING

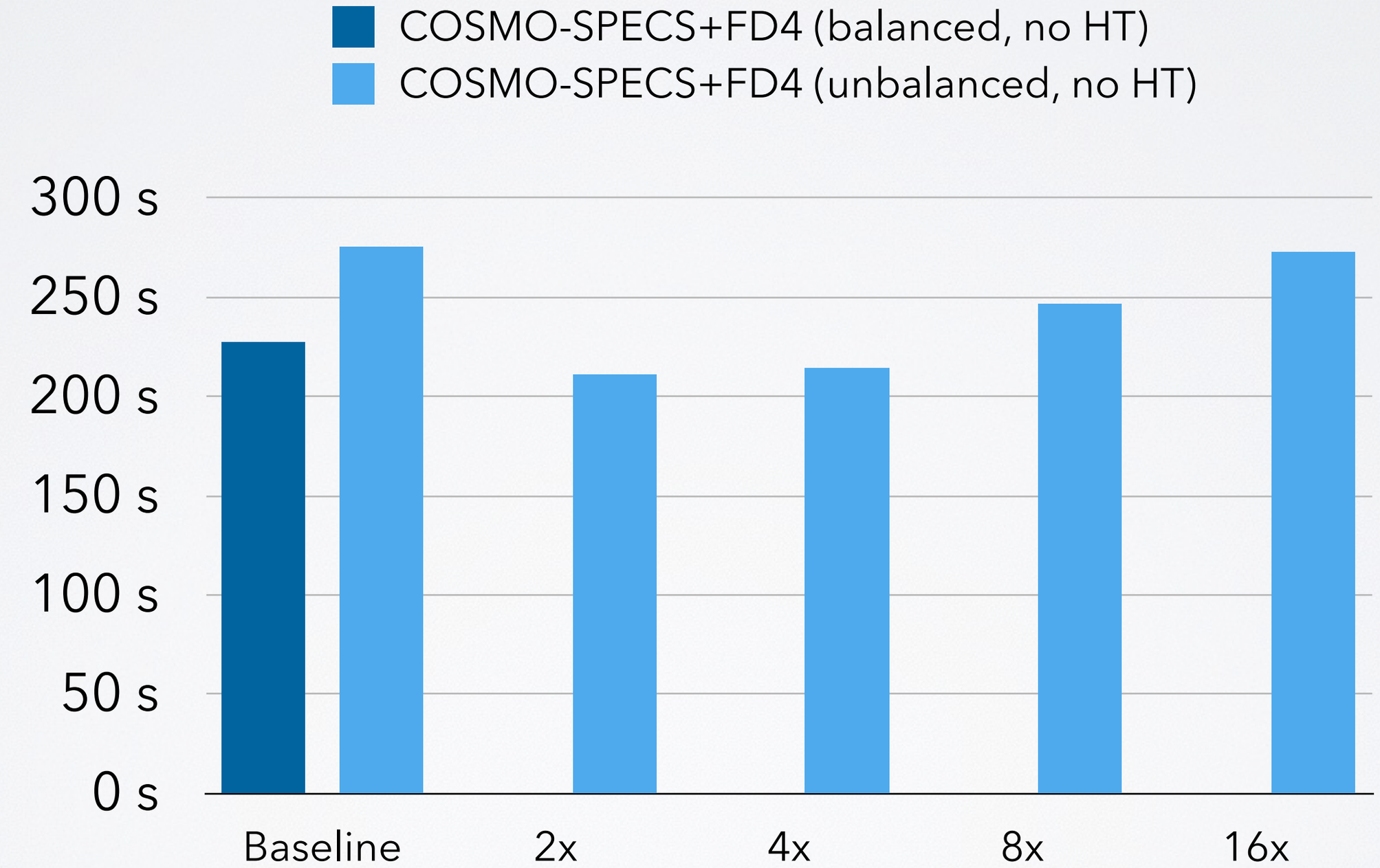


OVERDECOMPOSITION



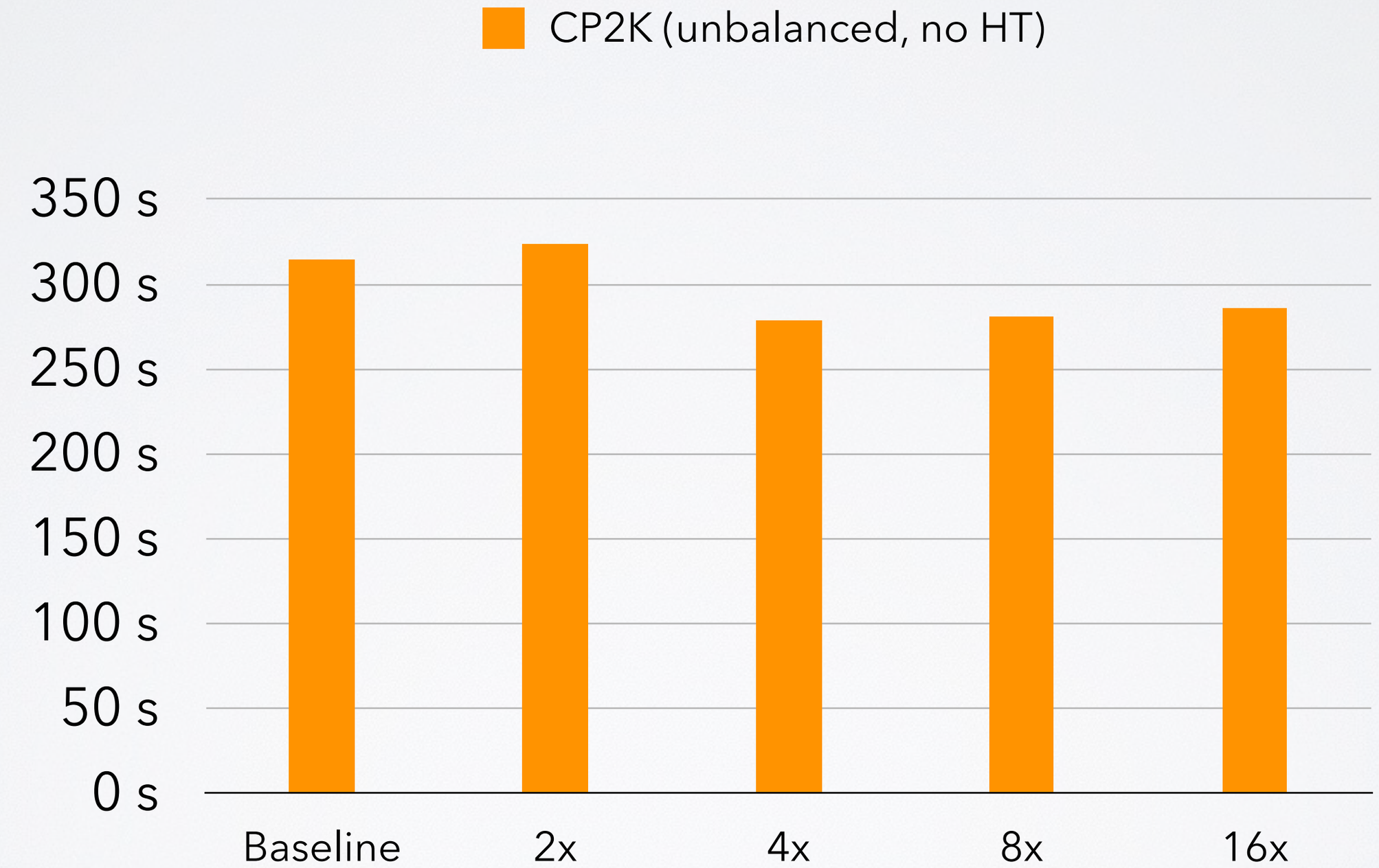
Oversubscribed runs: 32–256 MPI ranks on 4 quad-core nodes (w/ polling)

OVERDECOMPOSITION



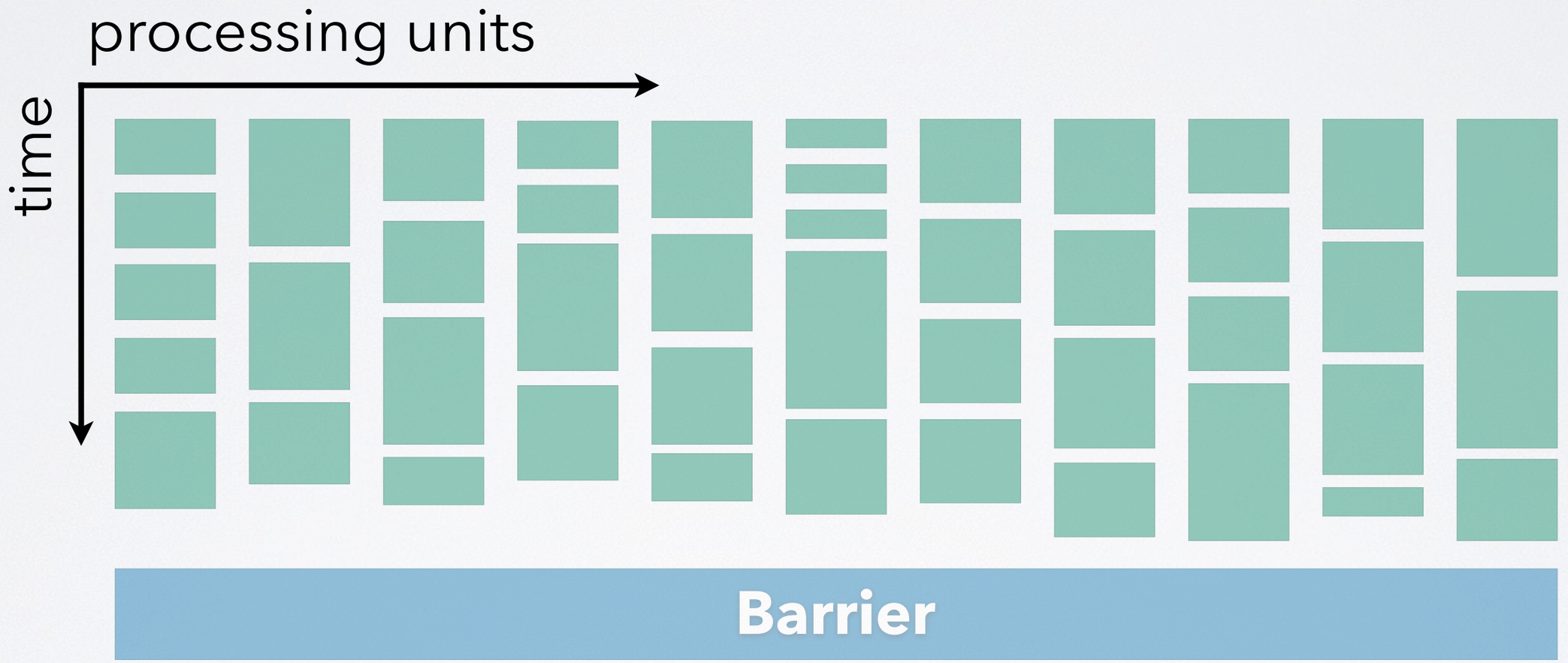
Oversubscribed runs: 16–256 MPI ranks on 4 quad-core nodes (w/o polling)

OVERDECOMPOSITION



Oversubscribed runs: 16–256 MPI ranks on 4 quad-core nodes (w/o polling)

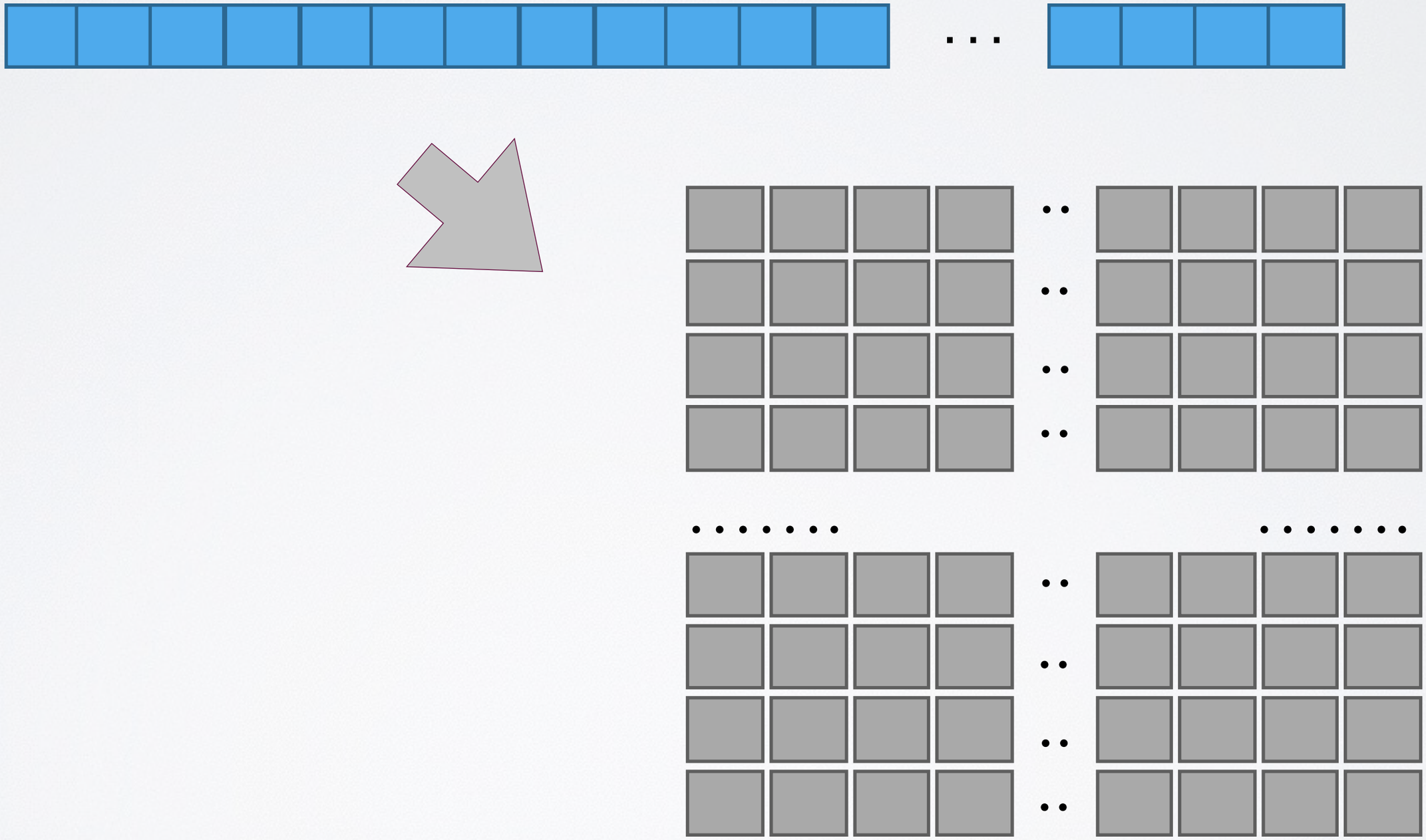
OVERDECOMPOSITION



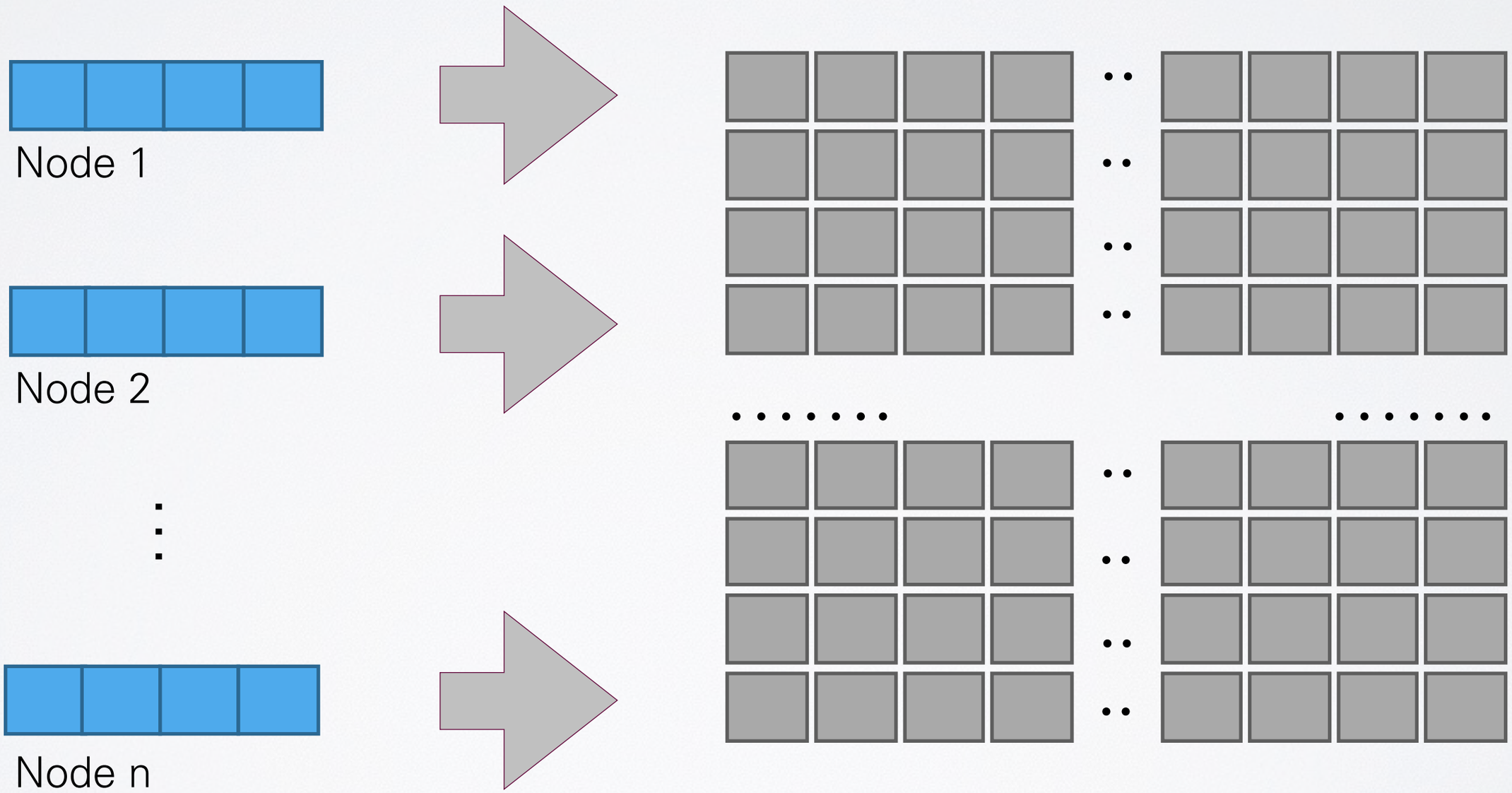
With MPI:

- **Do not:** busy wait (except very shortly)
- **Do:** Block in kernel
- **Needs:** fast unblocking of threads, when message comes in
- **We build:** shortcut from IB driver into MPI threads (no Linux!)

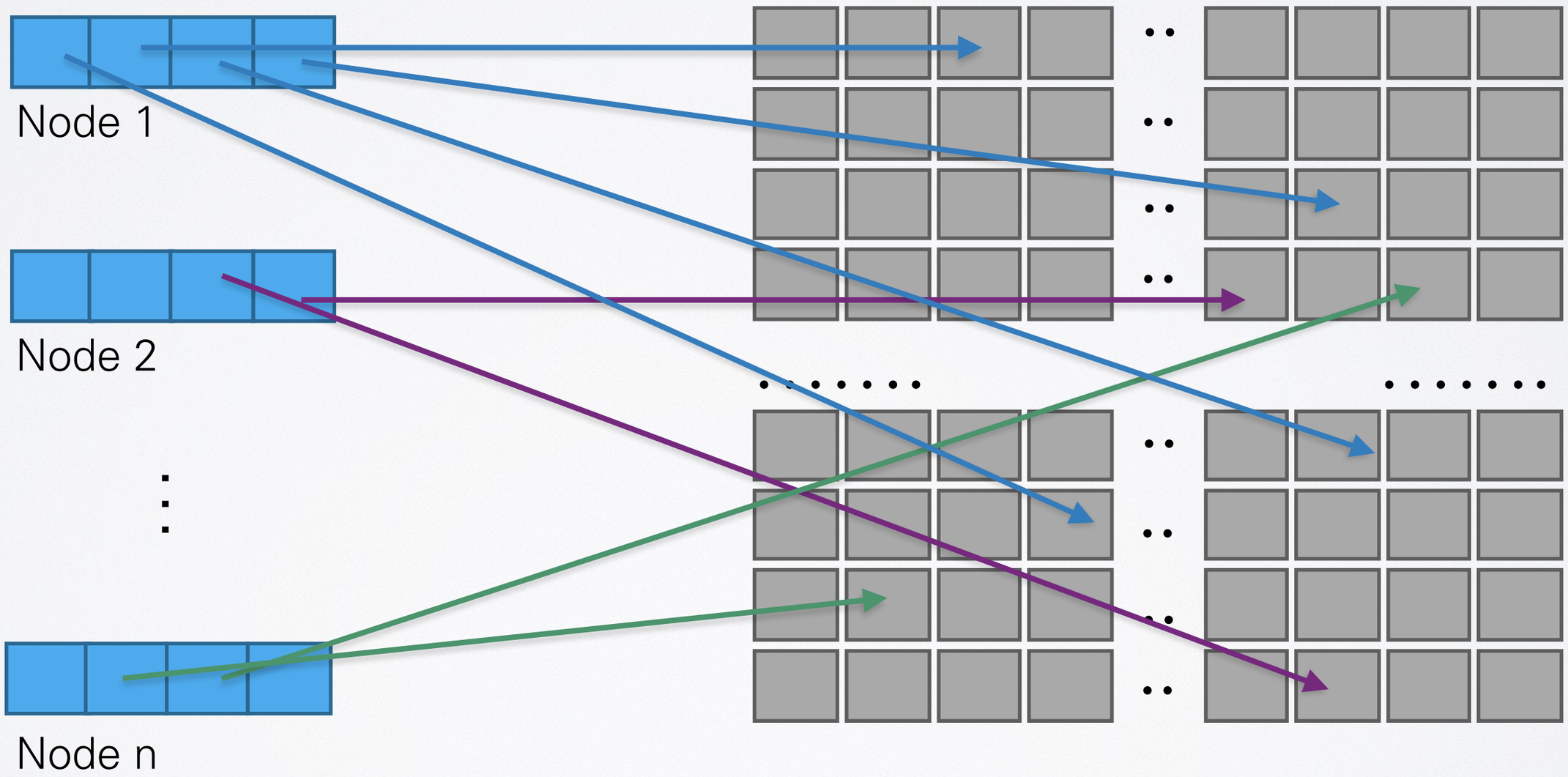
CENTRALIZED



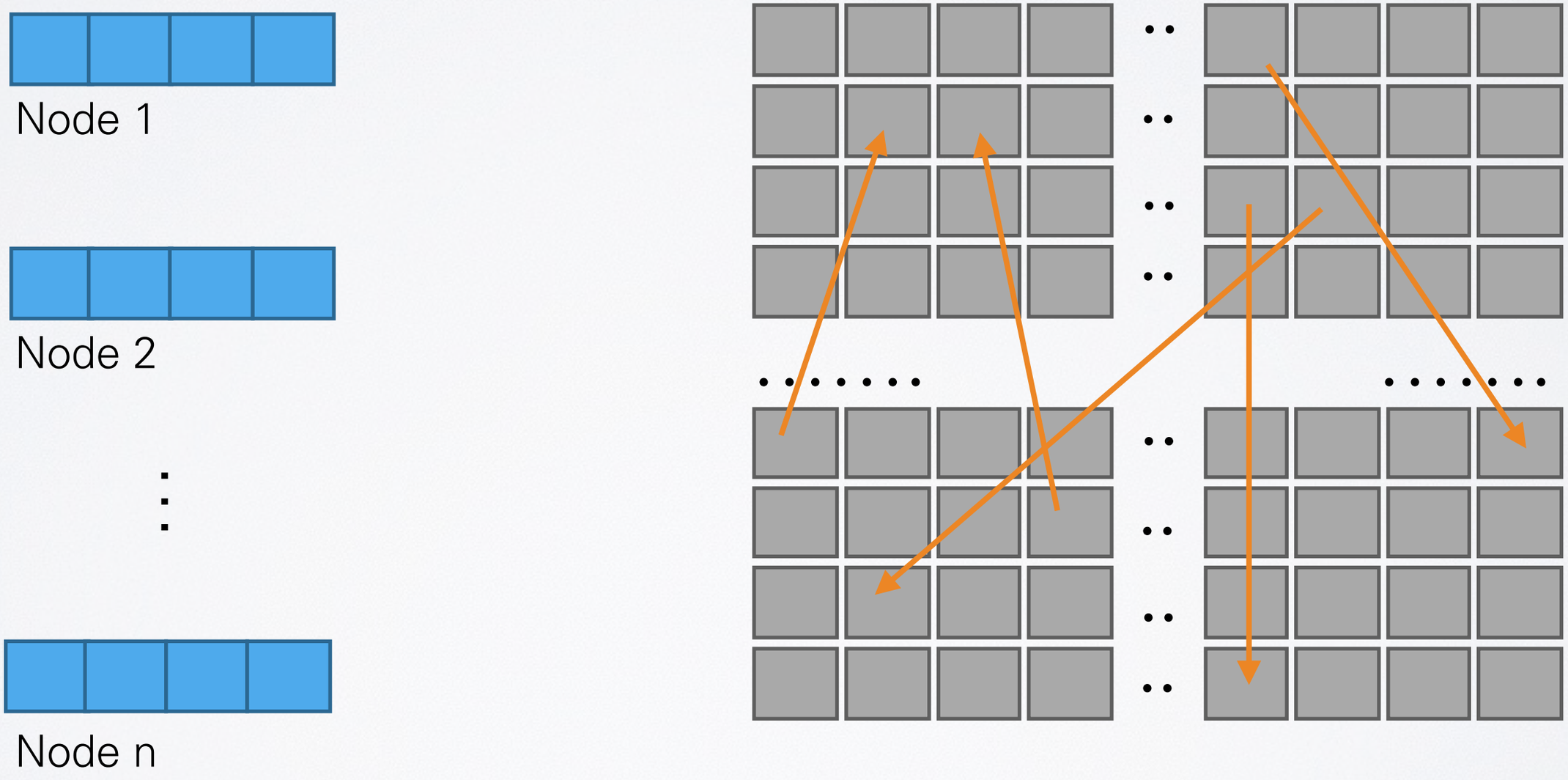
DECENTRALIZED



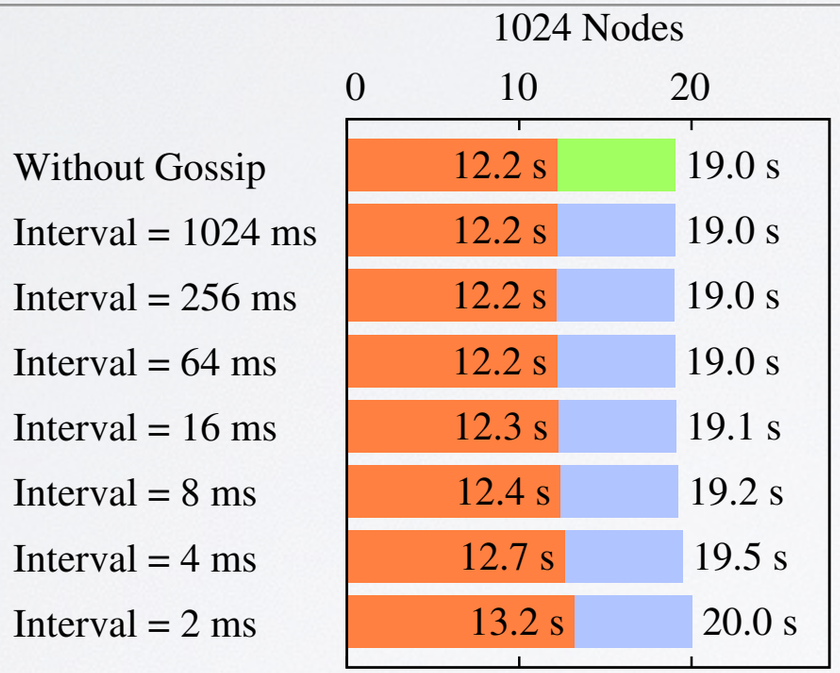
DECENTRALIZED



DECENTRALIZED



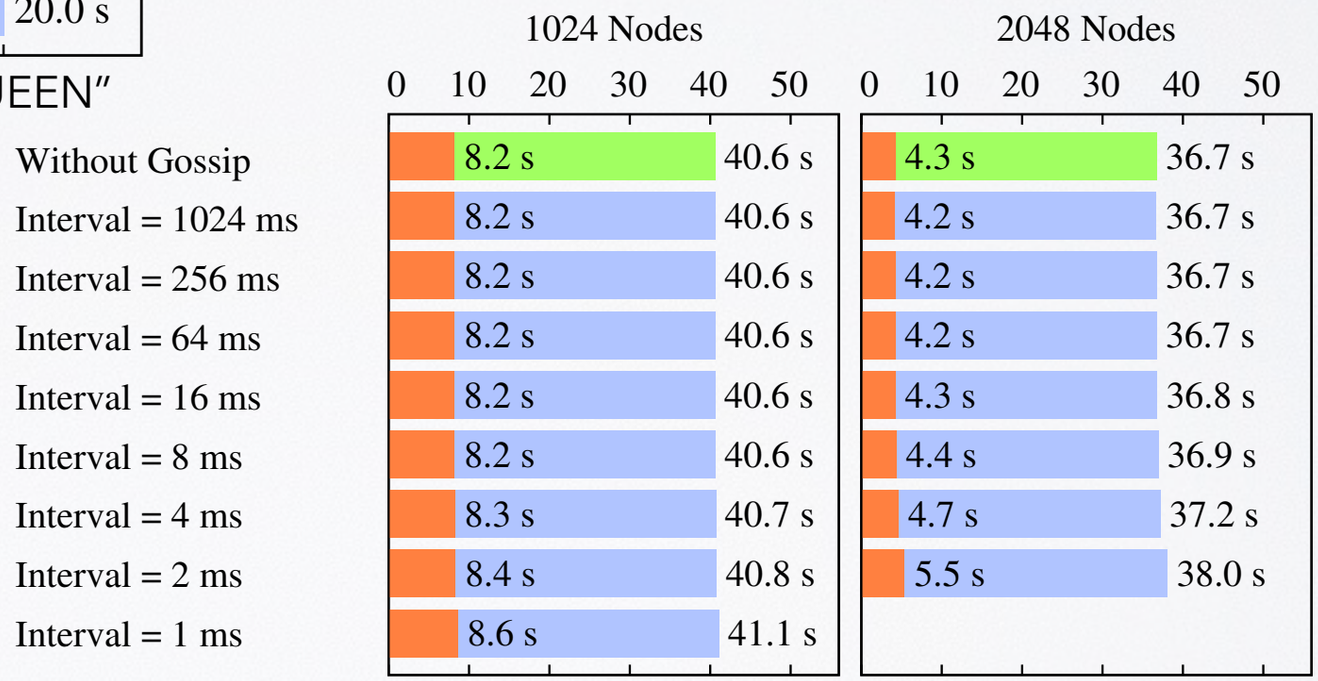
GOSSIP SCALABILITY



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%

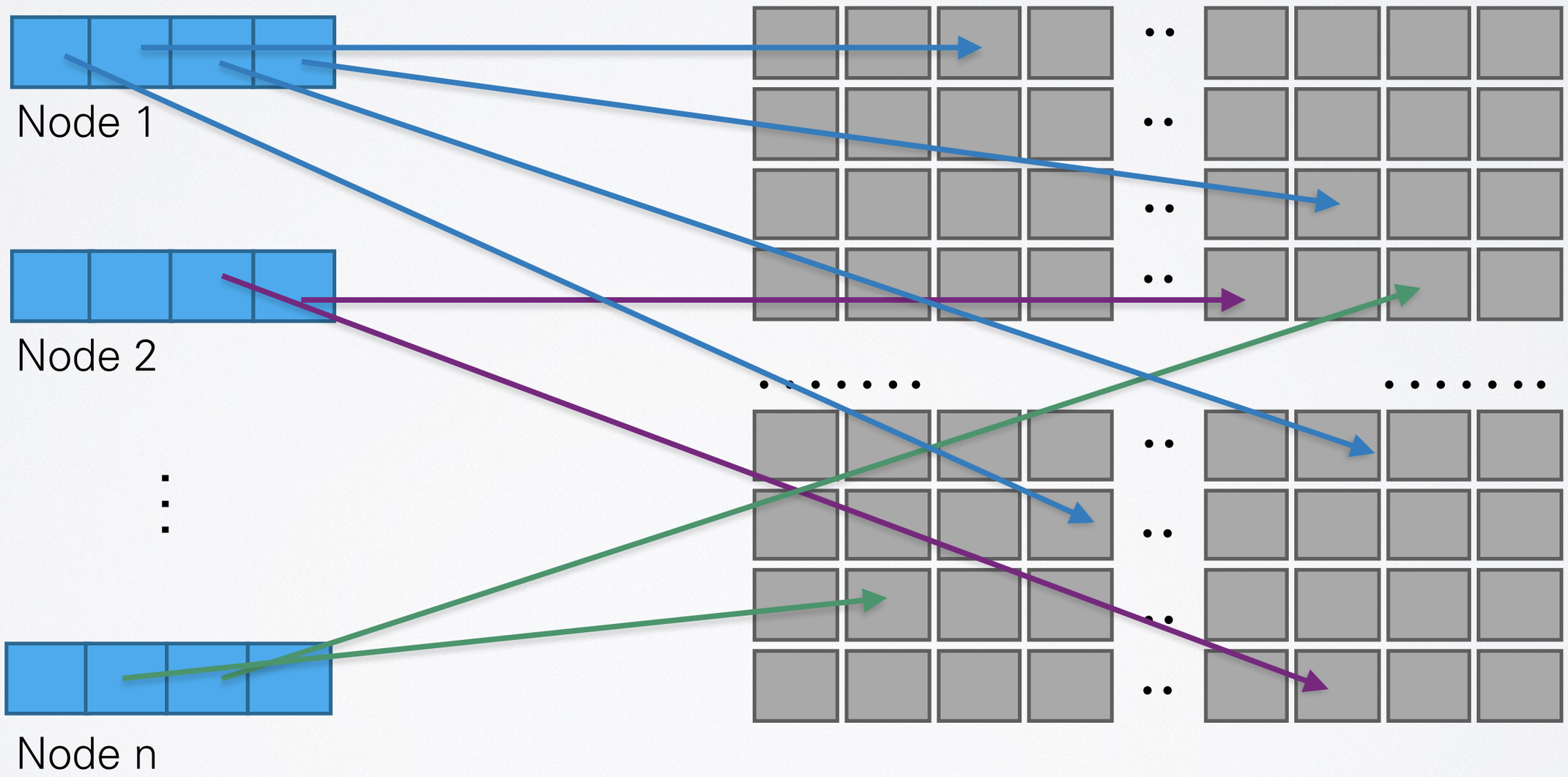
Average age at master (1024 nodes per colony)

Gossip is fault tolerant:

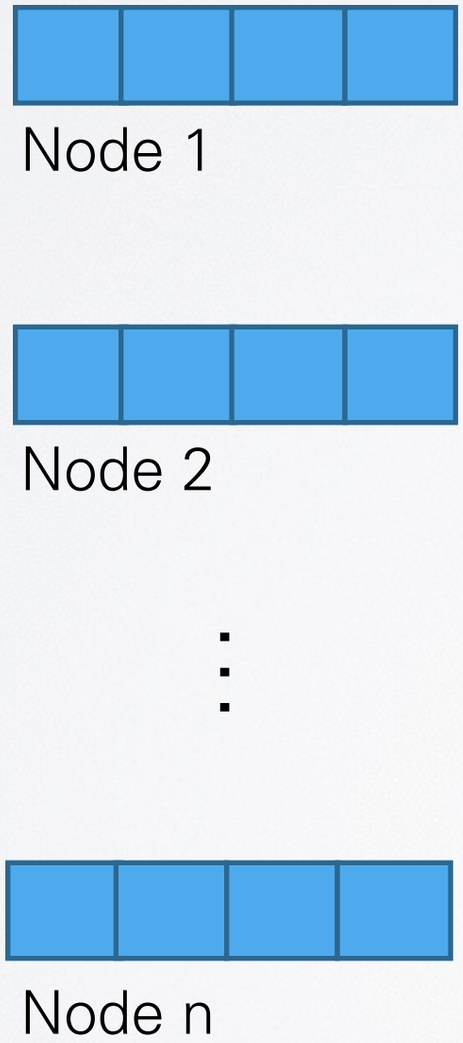
Only slight increase in average age when substantial number of nodes fail (up to 32 of 1024 in each colony)

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

DECENTRALIZED



DECENTRALIZED



When

MOSIX: Load difference discovered

+ Anomaly anticipated

Where

MOSIX: Memory, cycles, comm

+ Topology

Which

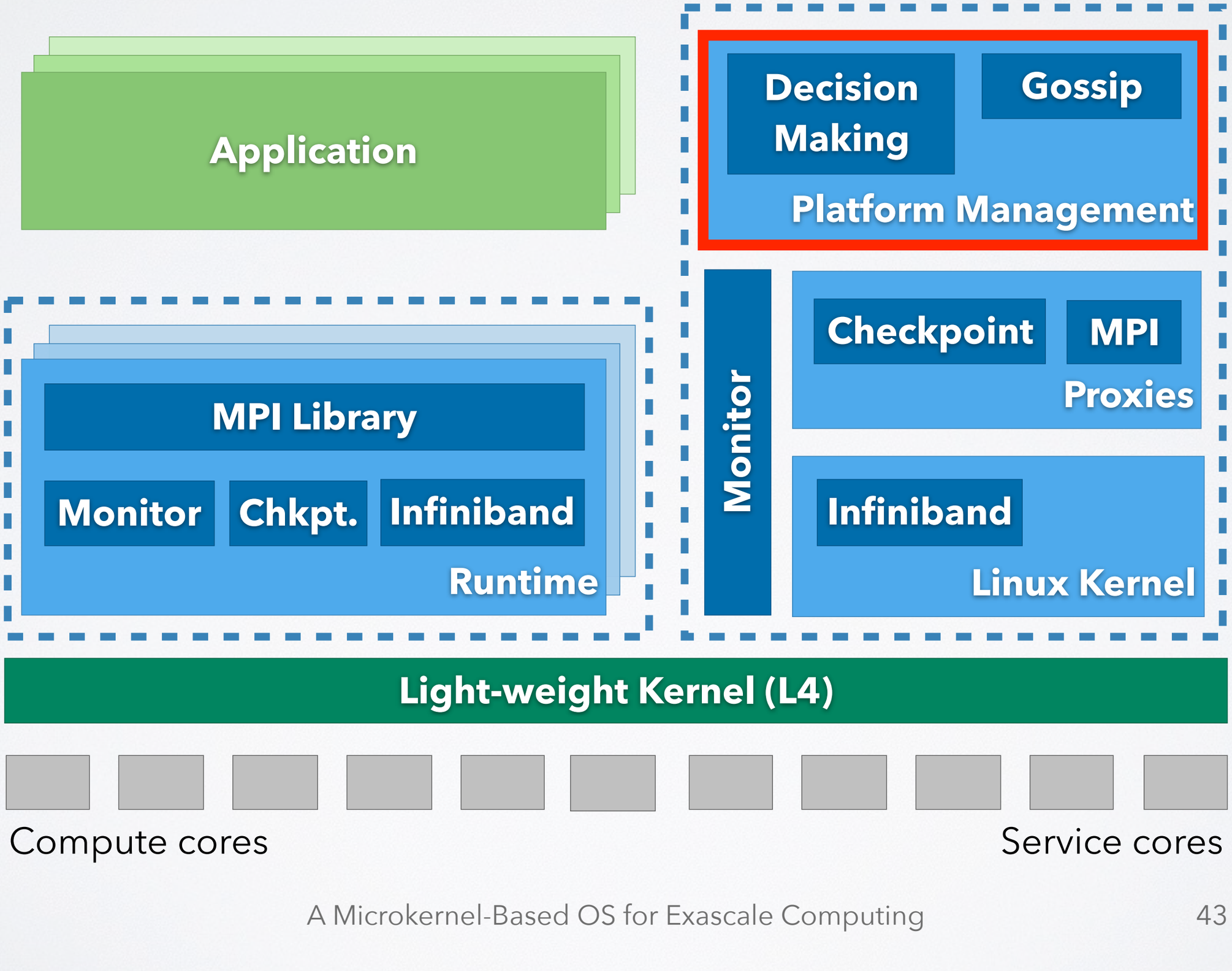
MOSIX: Past predicts future

+ Application knowledge

TECHNISCHE UNIVERSITÄT DRESDEN

The Hebrew University of Jerusalem

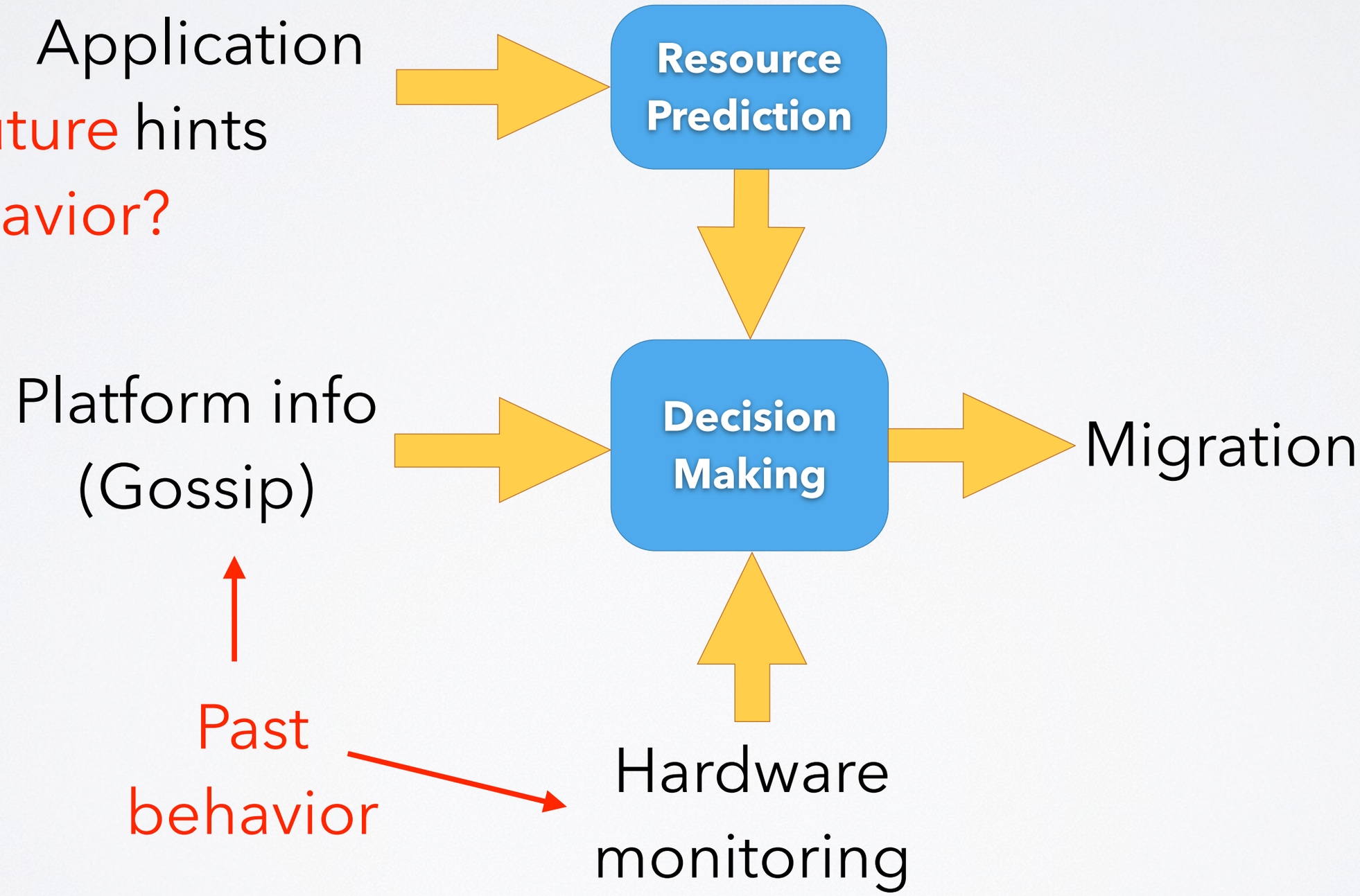
NODE ARCHITECTURE



Application

Future hints

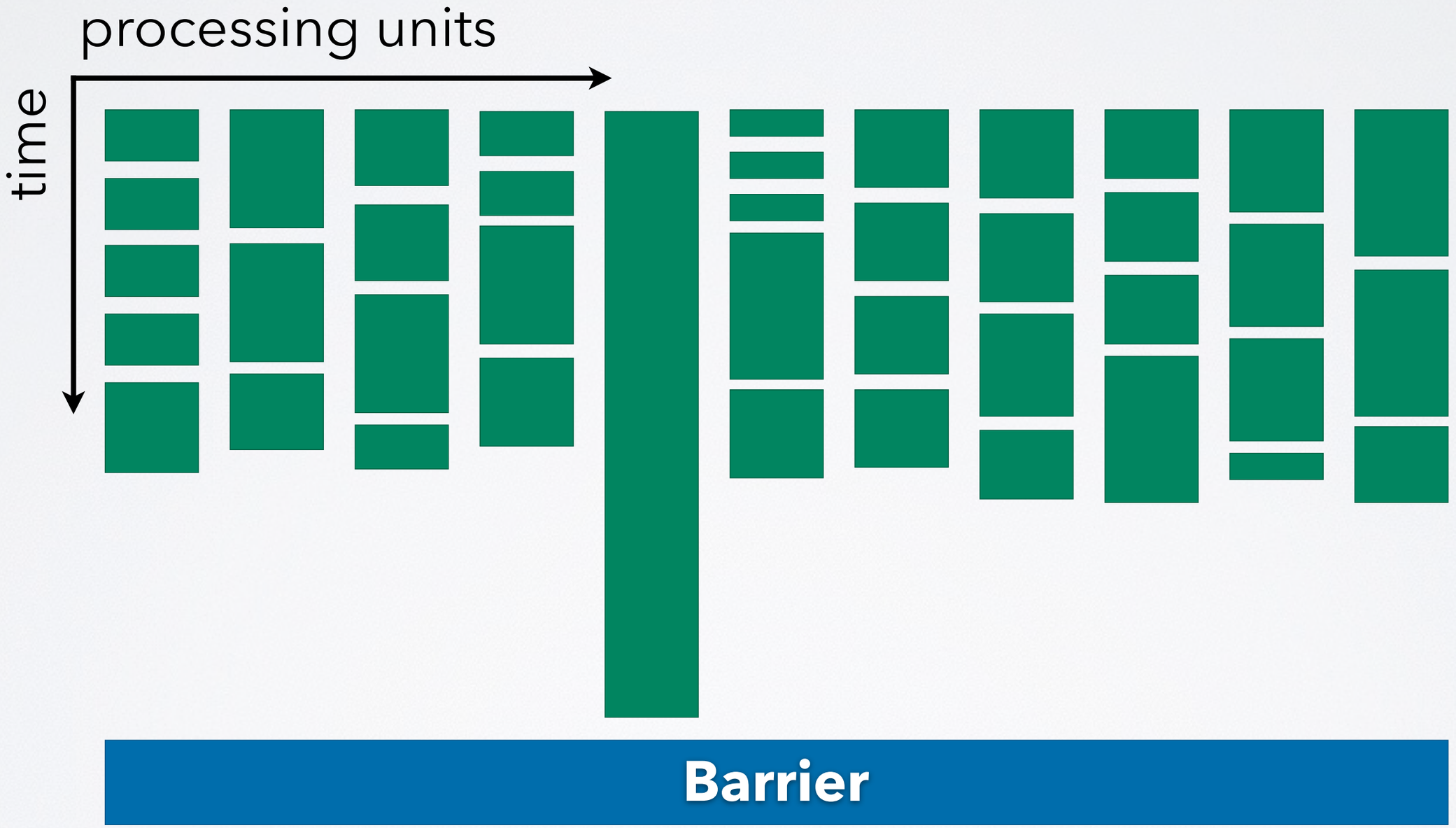
behavior?



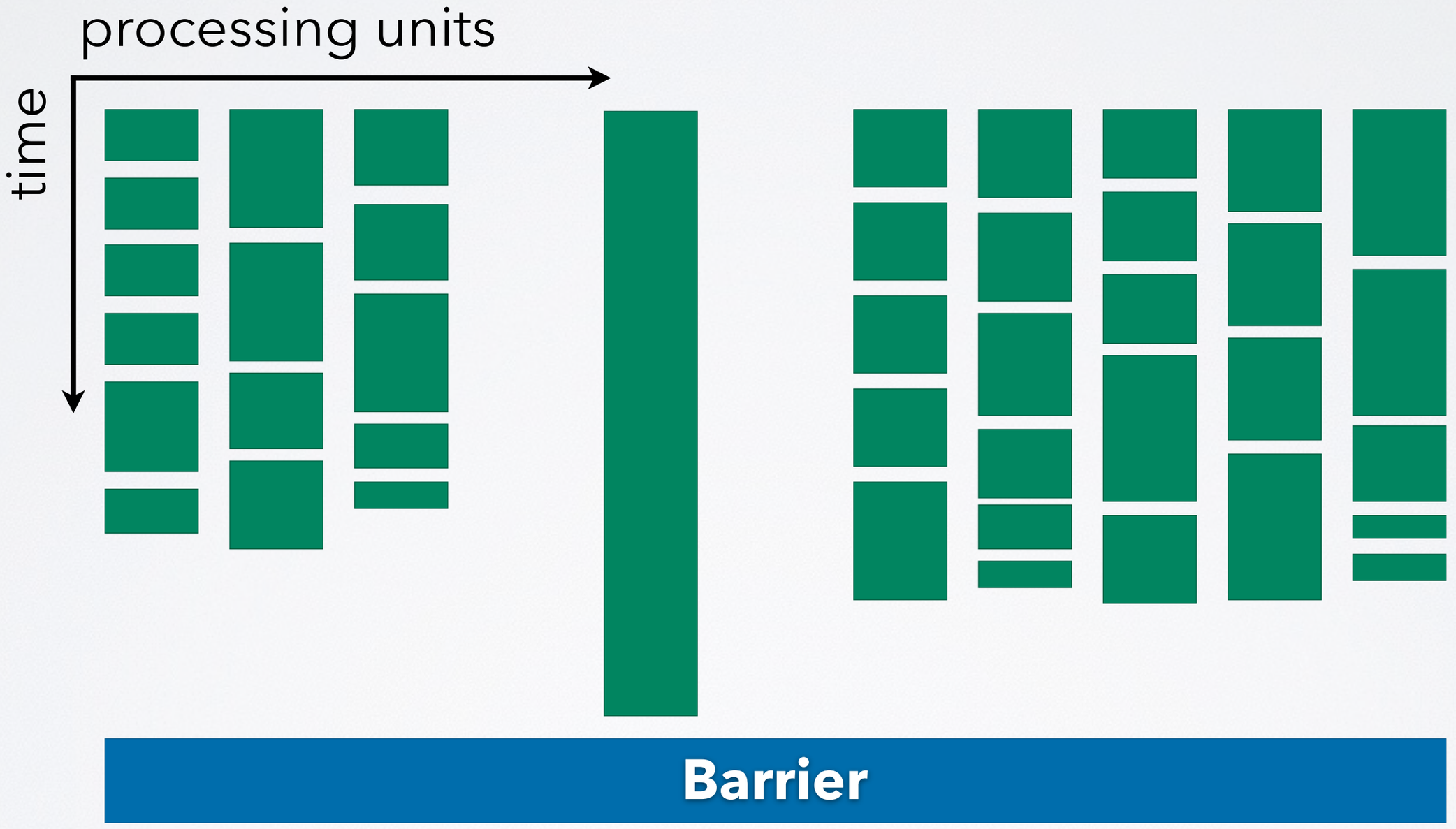
APPLICATION HINTS

- Resource prediction:
  - CPU cycles
  - Cache misses
  - Memory
  - Energy?

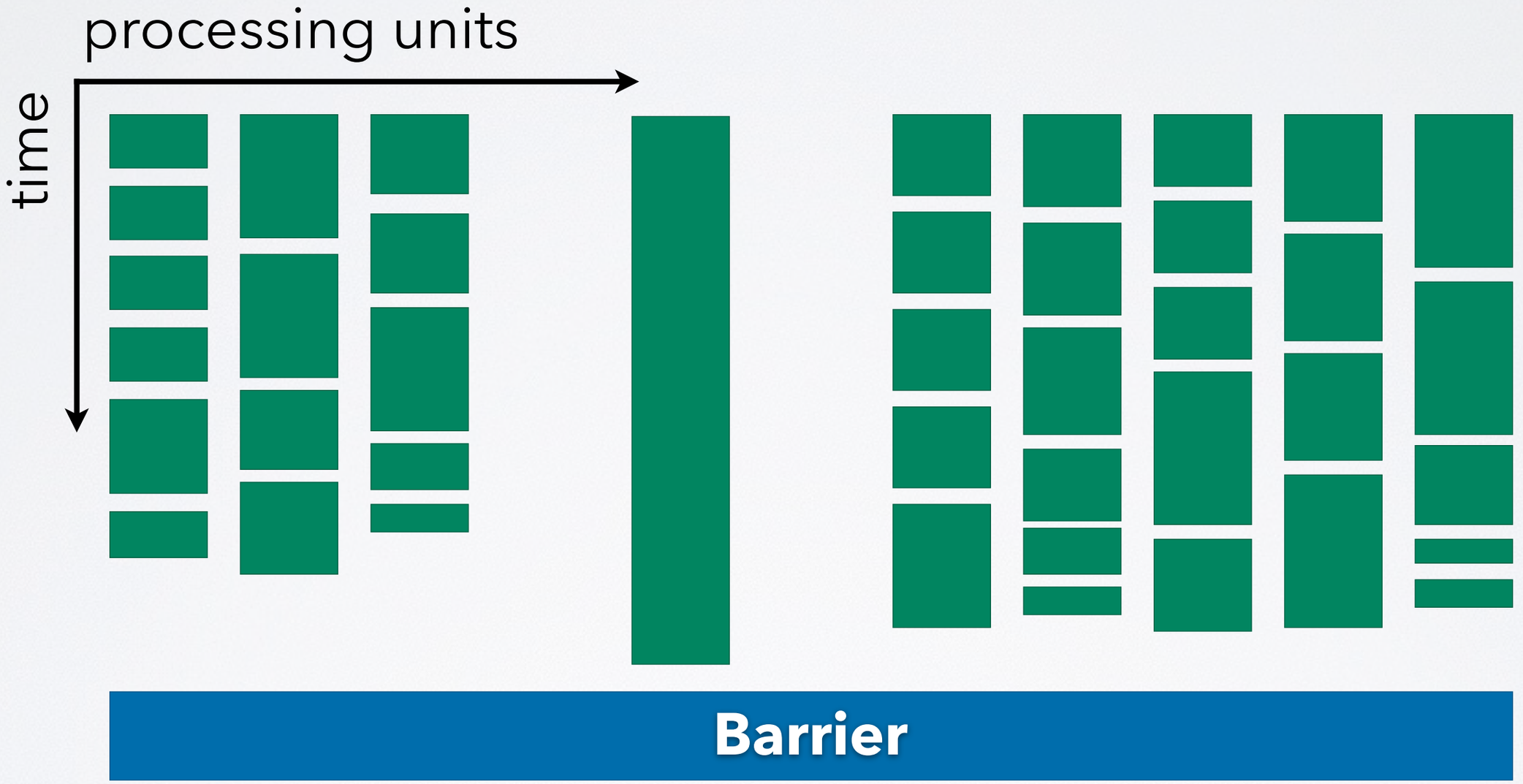
THERMAL PLANNING



THERMAL PLANNING



THERMAL PLANNING



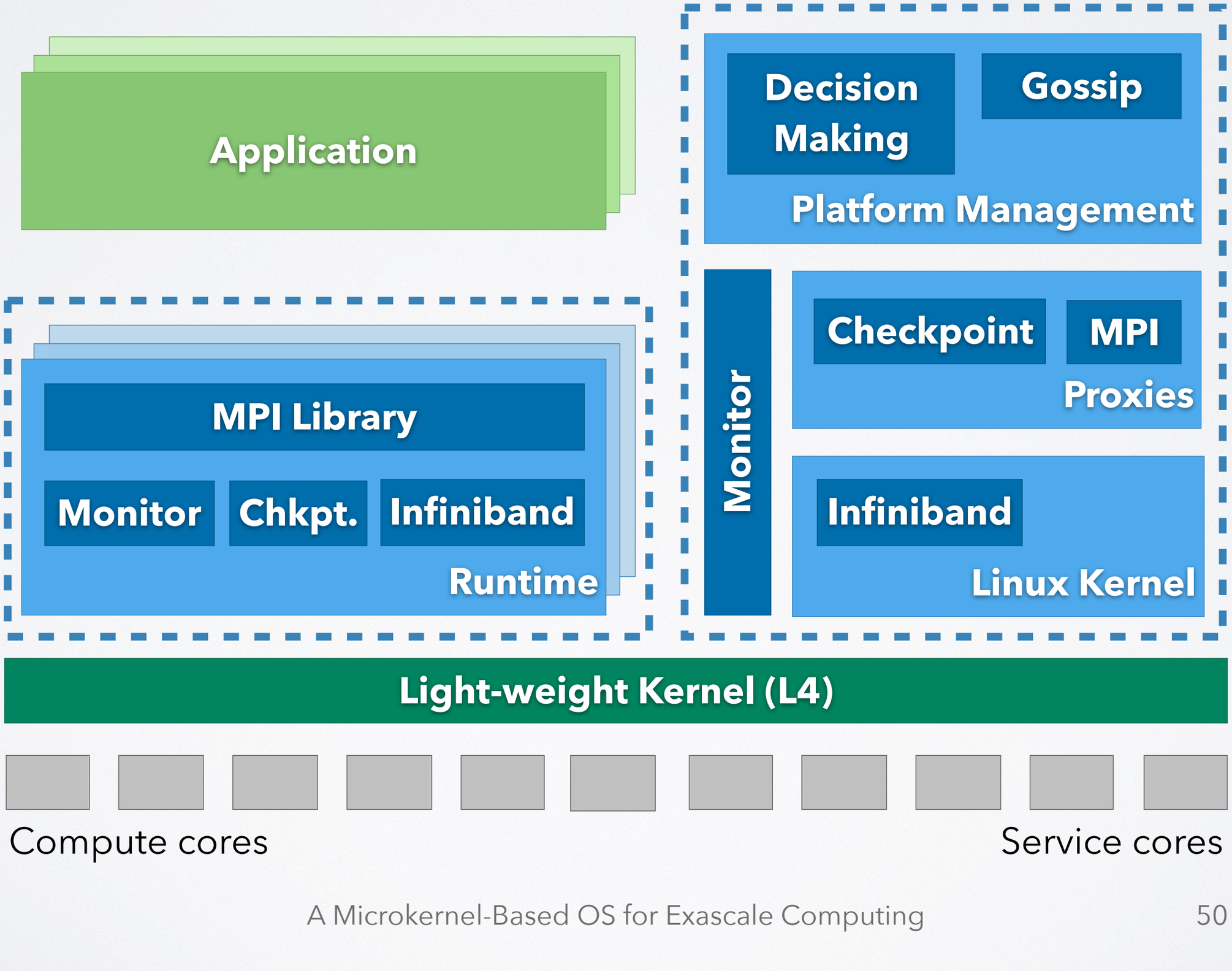
APPLICATION HINTS

- Resource prediction:
  - CPU cycles
  - Cache misses
  - Memory
  - Energy?
- Fault tolerance:
  - Which data to checkpoint (and when)
  - Ability to handle node failures

TECHNISCHE UNIVERSITÄT DRESDEN

The Hebrew University of Jerusalem

NODE ARCHITECTURE



SUMMARY

- FFMK prototype runs on real HPC system
- We build **OS/R for Exascale:**
  - Load management
  - Fault tolerance
- Take burden from application developers

SPPEXA

ffmk.tudos.org