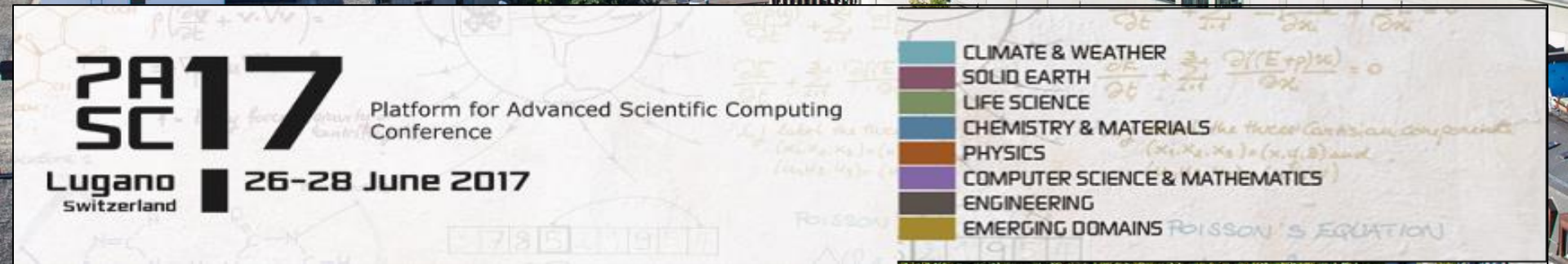
High-Performance Distributed RMA Locks

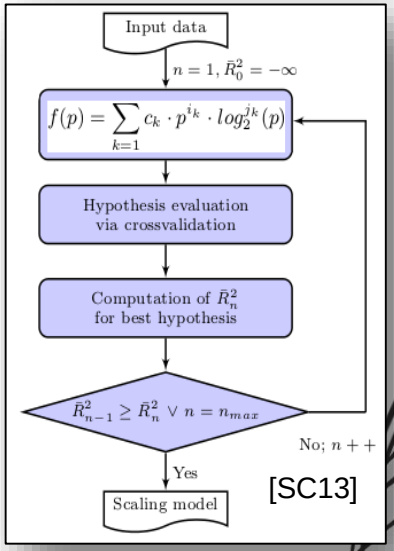
TORSTEN HOEFLER

with support of Patrick Schmid, Maciej Besta @ SPCL
presented at Wuxi, China, Sept. 2016

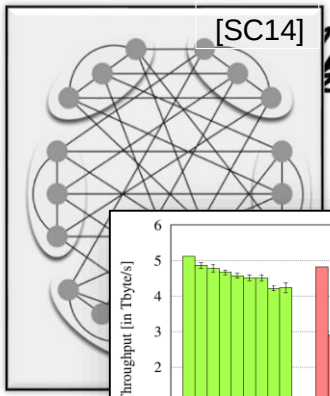


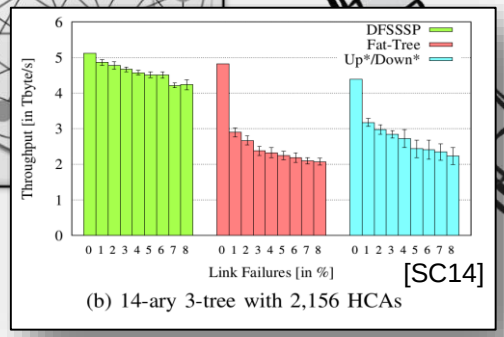
ETH, CS, Systems Group, SPCL

- **ETH Zurich – top university in central Europe**
  - Shanghai ranking ‘15 (Computer Science): #17, best outside North America
  - 16 departments, 1.62 Bn $ federal budget
- **Computer Science department**
  - 28 tenure-track faculty, 1k students
- **Systems group (7 professors)**
  - O. Mutlu, T. Roscoe, G. Alonso, A. Singla, C. Zheng, D. Kossmann, TH
  - Focused on systems research of all kinds (data management, OS, …)
- **SPCL focusses on performance/data/HPC**
  - 1 faculty
  - 3 postdocs
  - 8 PhD students (+2 external)
  - 15+ BSc and MSc students
  - http://spcl.inf.ethz.ch
  - Twitter: @spcl_eth

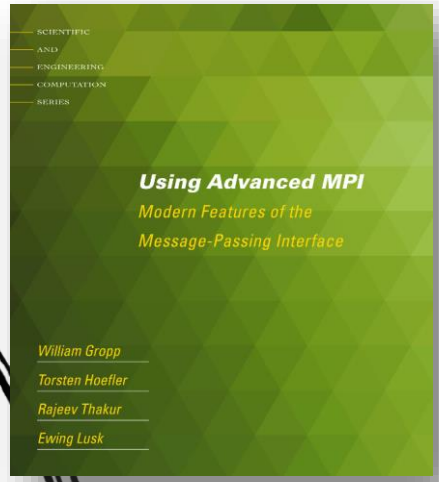


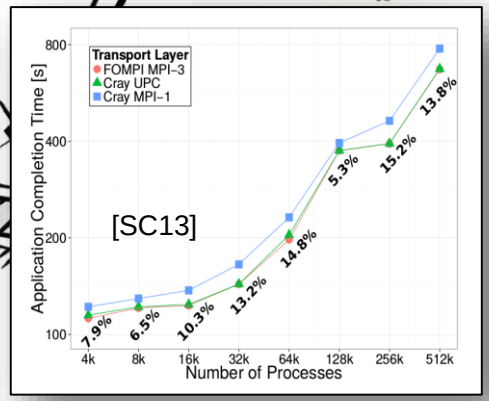
[SC13]


[SC14]



(b) 14-ary 3-tree with 2,156 HCAs
[SC14]





[SC13]

Performance Modeling

Parallel Programming

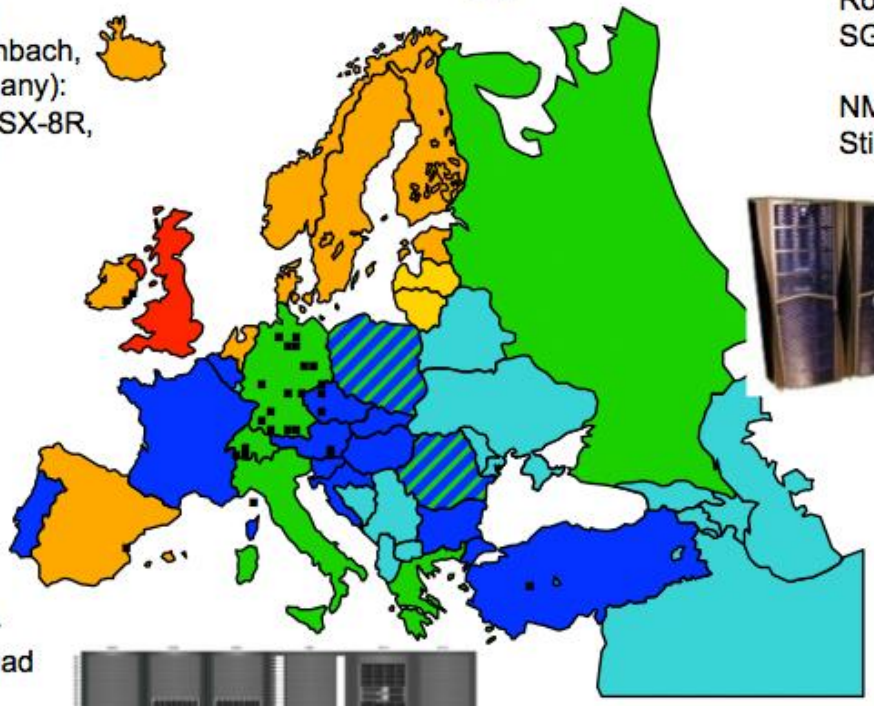
Large-scale Networking

DAPP

crClim – Cloud-resolving Climate Simulations

DWD (Offenbach, Germany): NEC SX-8R, SX-9

COSMO NWP-Applications



Roshydromet (Moscow, Russia), SGI

NMA (Bucharest, Romania): Still in planning / procurement phase

IMGW (Warsawa, Poland): SGI Origin 3800: uses 88 of 100 nodes

MeteoSwiss: Cray XT4: COSMO-7 and COSMO-2 use 980+4 MPI-Tasks on 246 out of 260 quad core AMD nodes

ARPA-SIM (Bologna, Italy): IBM pwr5: up to 160 of 512 nodes at CINECA

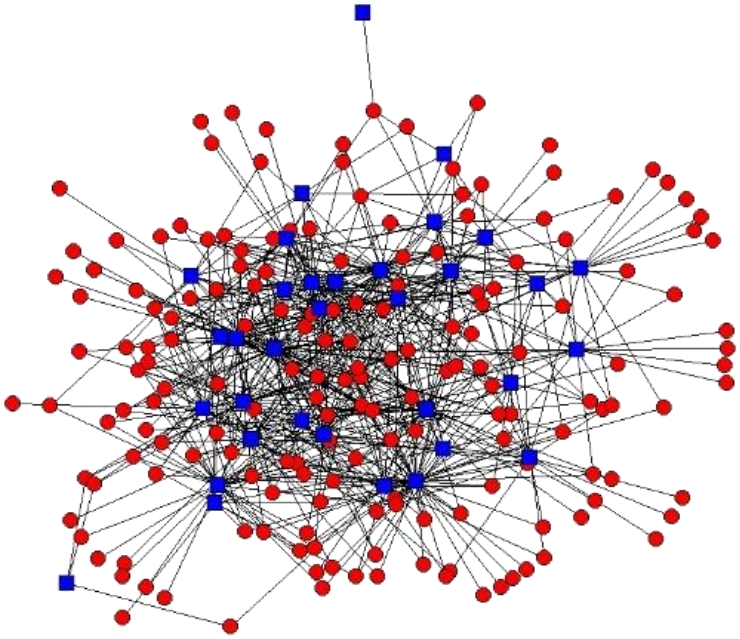
COSMO-LEPS (at ECMWF): running on ECMWF pwr6 as member-state time-critical application

HNMS (Athens, Greece): IBM pwr4: 120 of 256 nodes

USAM (Rome, Italy): HP Linux Cluster XEON biproc quadcore System in preparation

ARPA-SIM (Bologna, Italy): Linux-Intel x86-64 Cluster for testing (uses 56 of 120 cores)

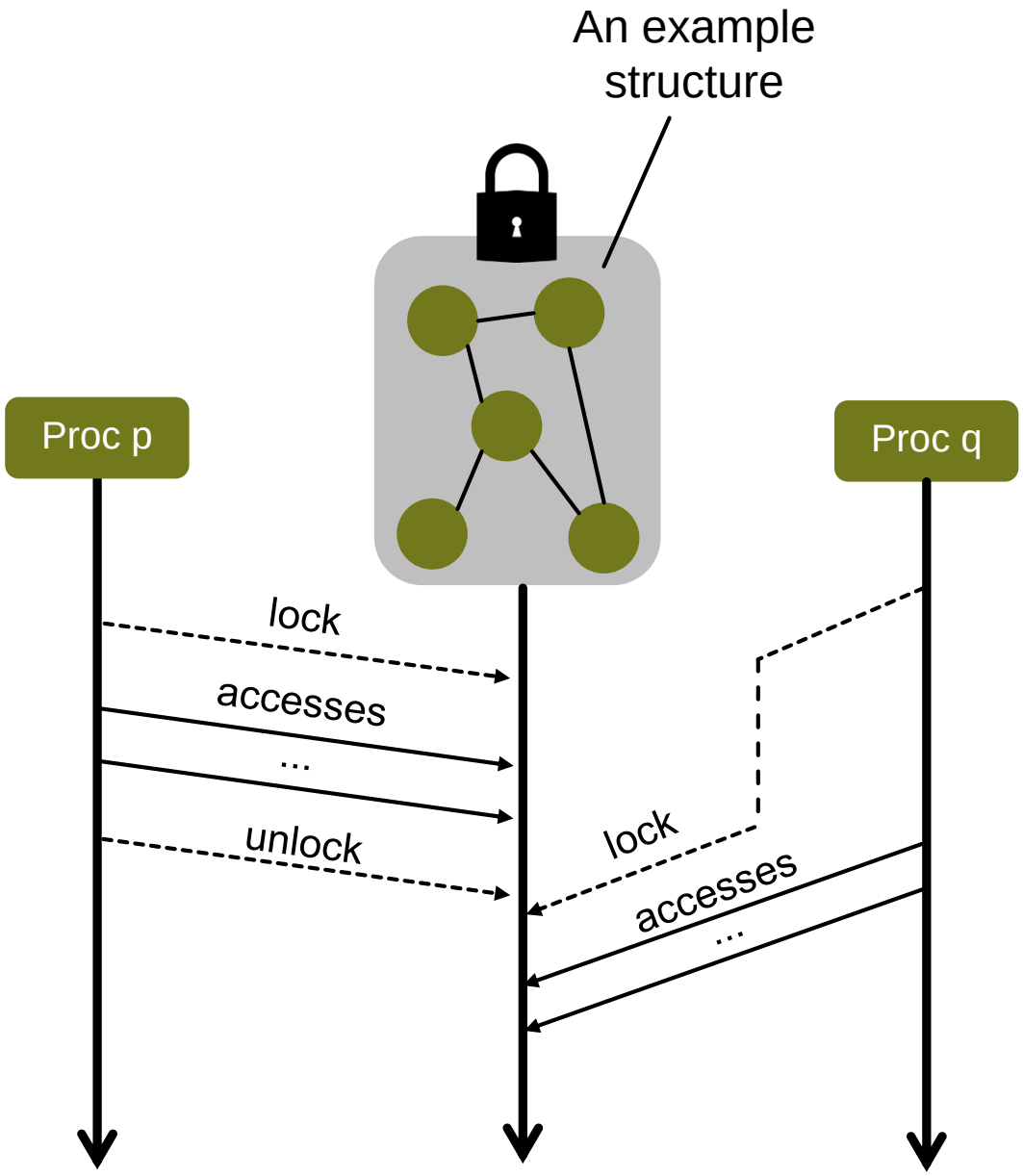
Need for Efficient large-Scale Synchronization



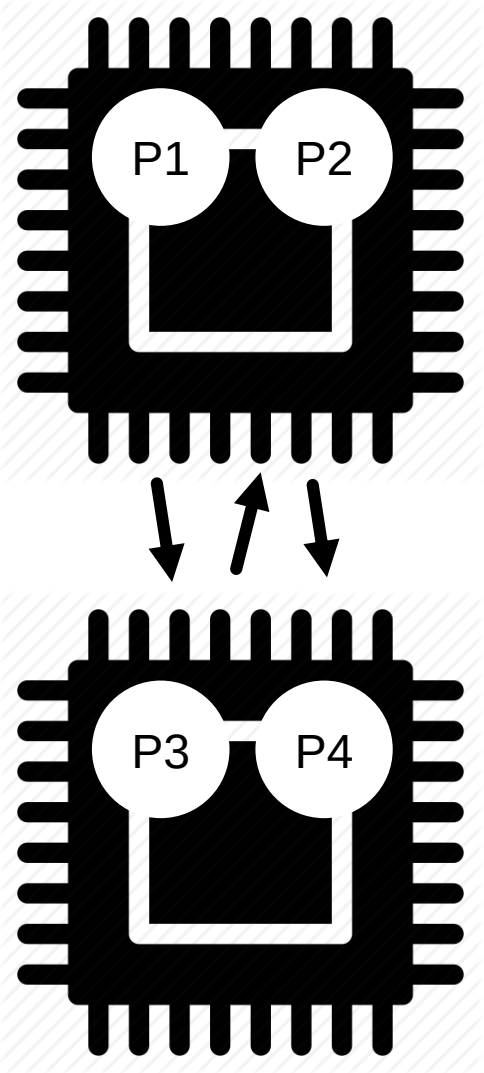
Locks

Inuitive semantics

Various performance penalties



LOCKS: CHALLENGES



LOCKS: CHALLENGES

We need intra- and inter-node topology-awareness

We need to cover arbitrary topologies

Locks: Challenges

We need to distinguish between readers and writers

Reader

Reader

Reader

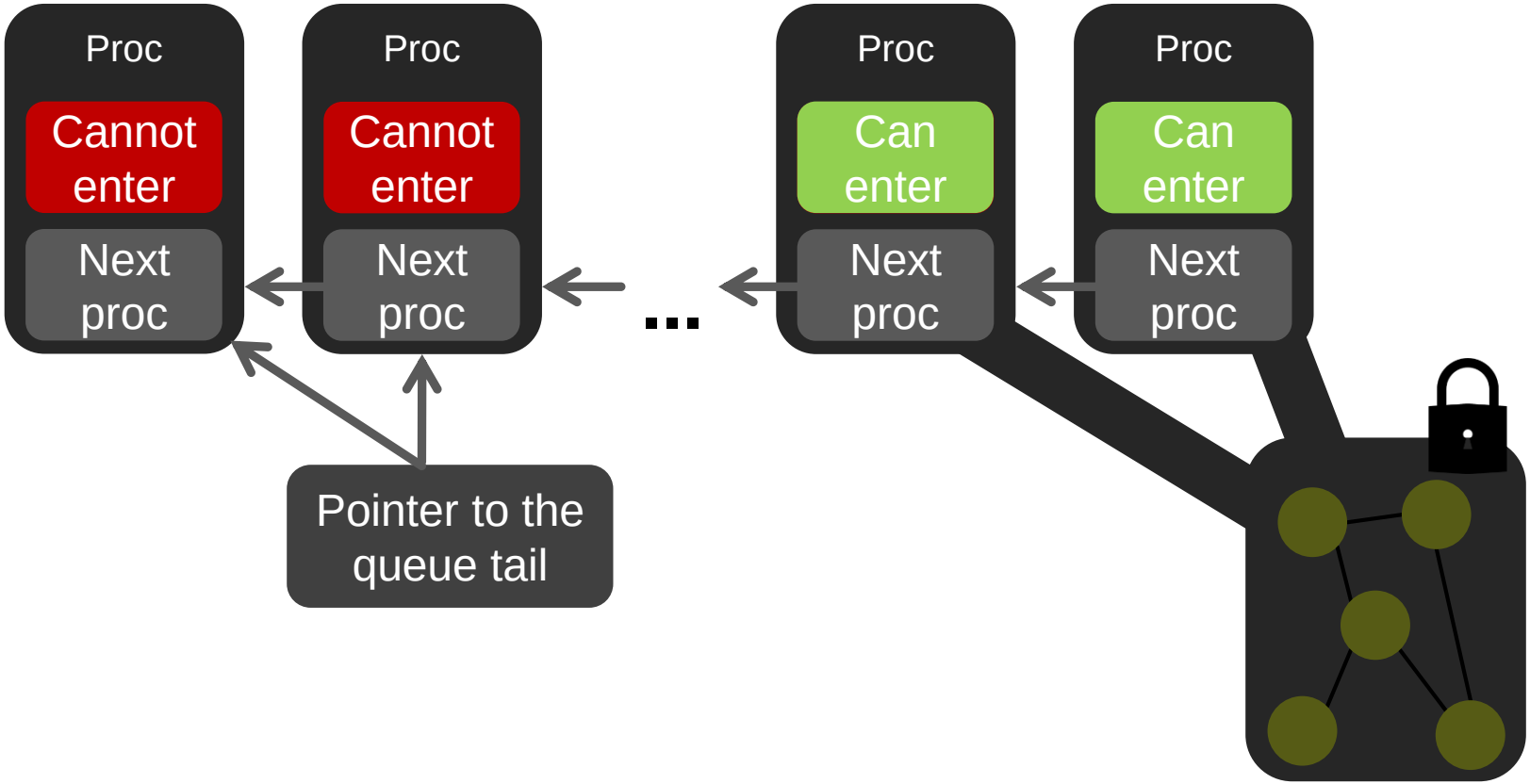
Writer

We need flexible performance for both types of processes

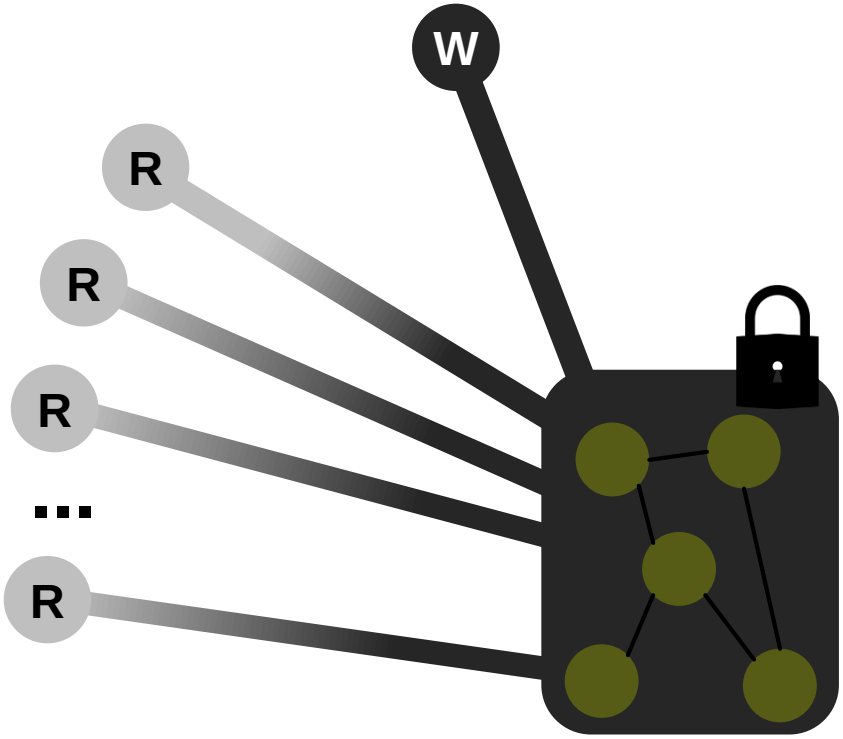
[1] V. Venkataramani et al. Tao: How facebook serves the social graph. SIGMOD'12.

What will we use in the design?

WHAT WE WILL USE
MCS Locks



What we will use
Reader-Writer Locks

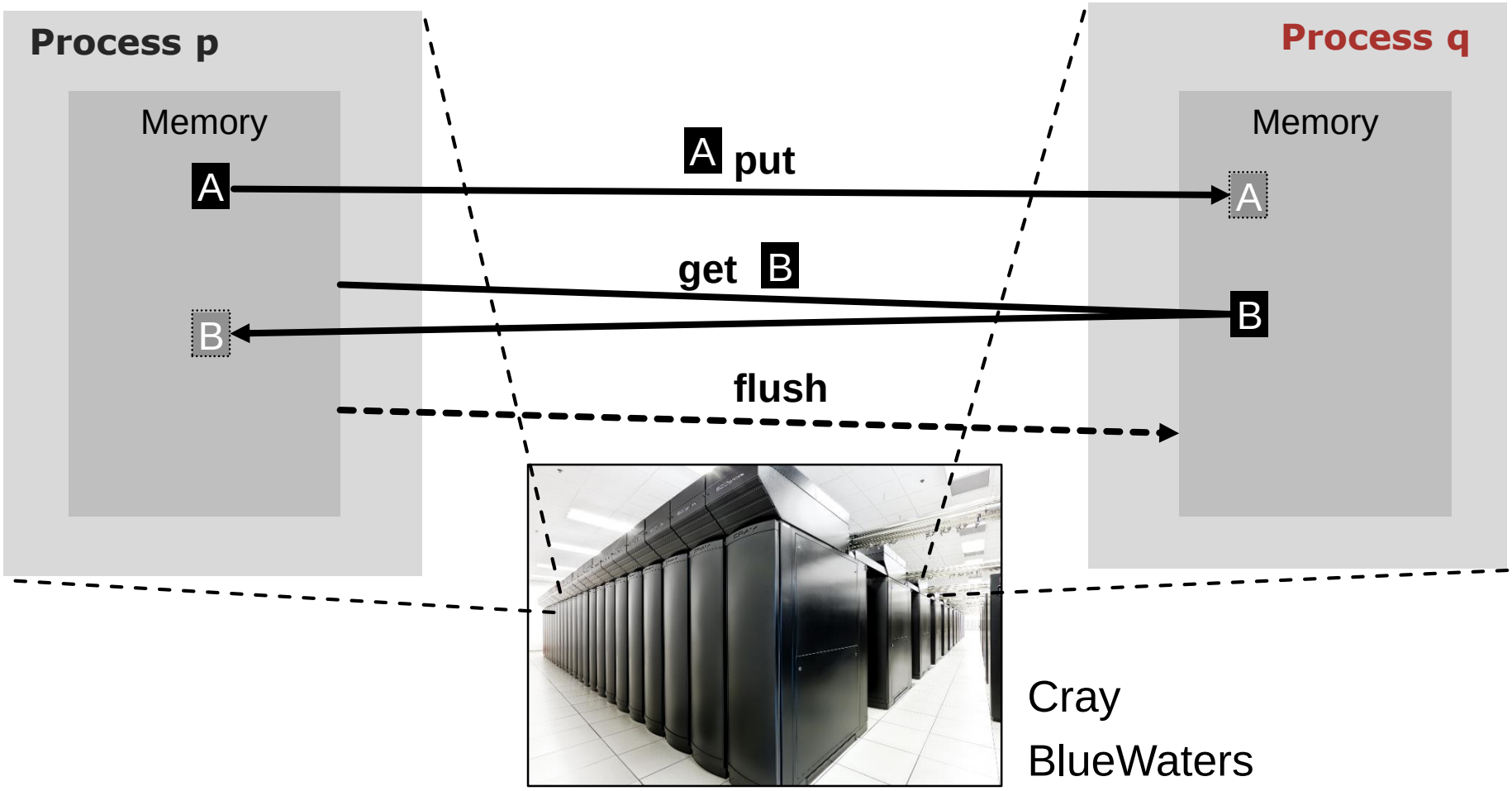


How to manage the design complexity?

How to ensure tunable performance?

What mechanism to use for efficient implementation?

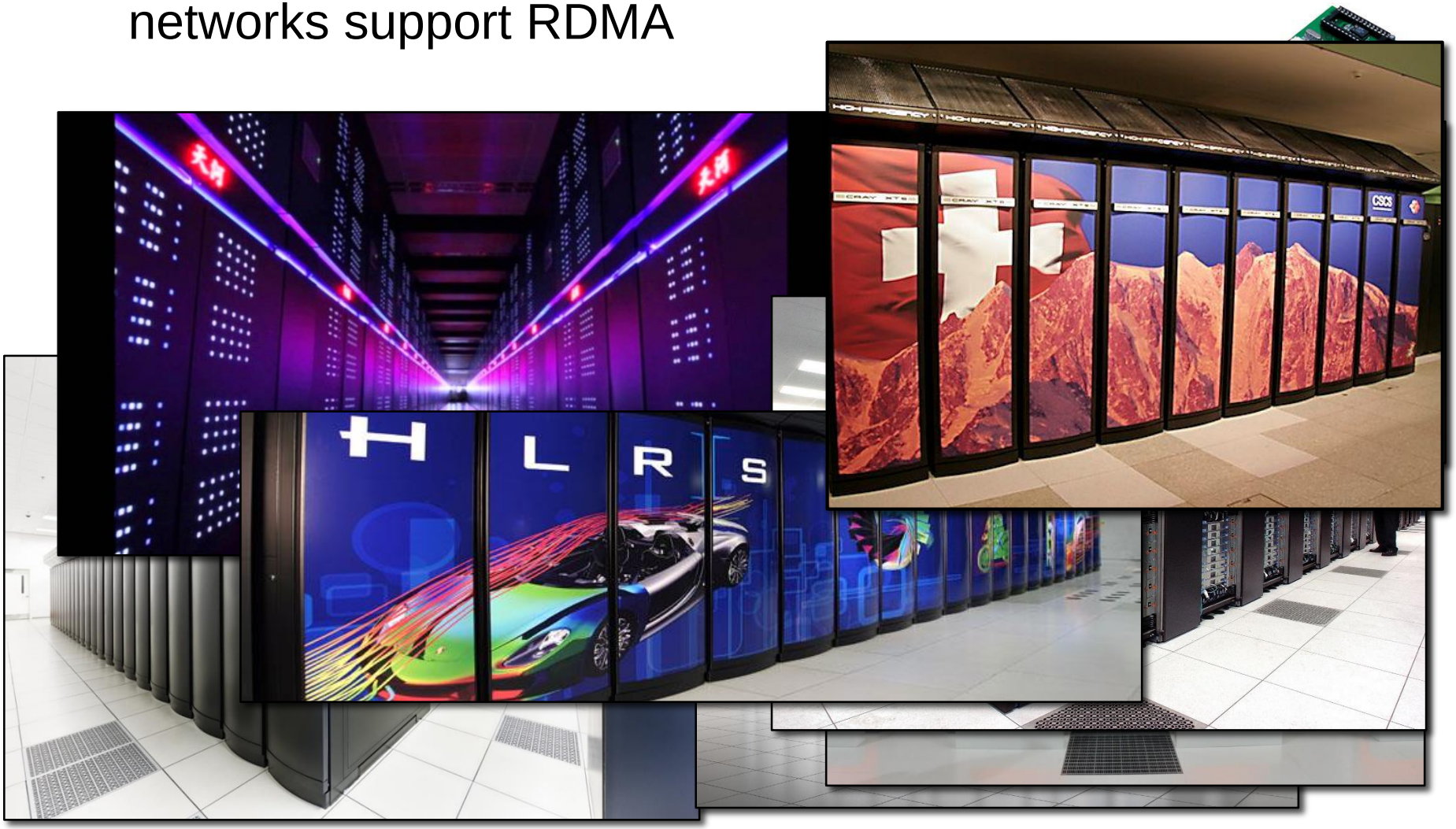
Remote Memory Access (RMA) Programming



Cray BlueWaters

REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



How to manage the design complexity?

How to ensure tunable performance?

What mechanism to use for efficient implementation?

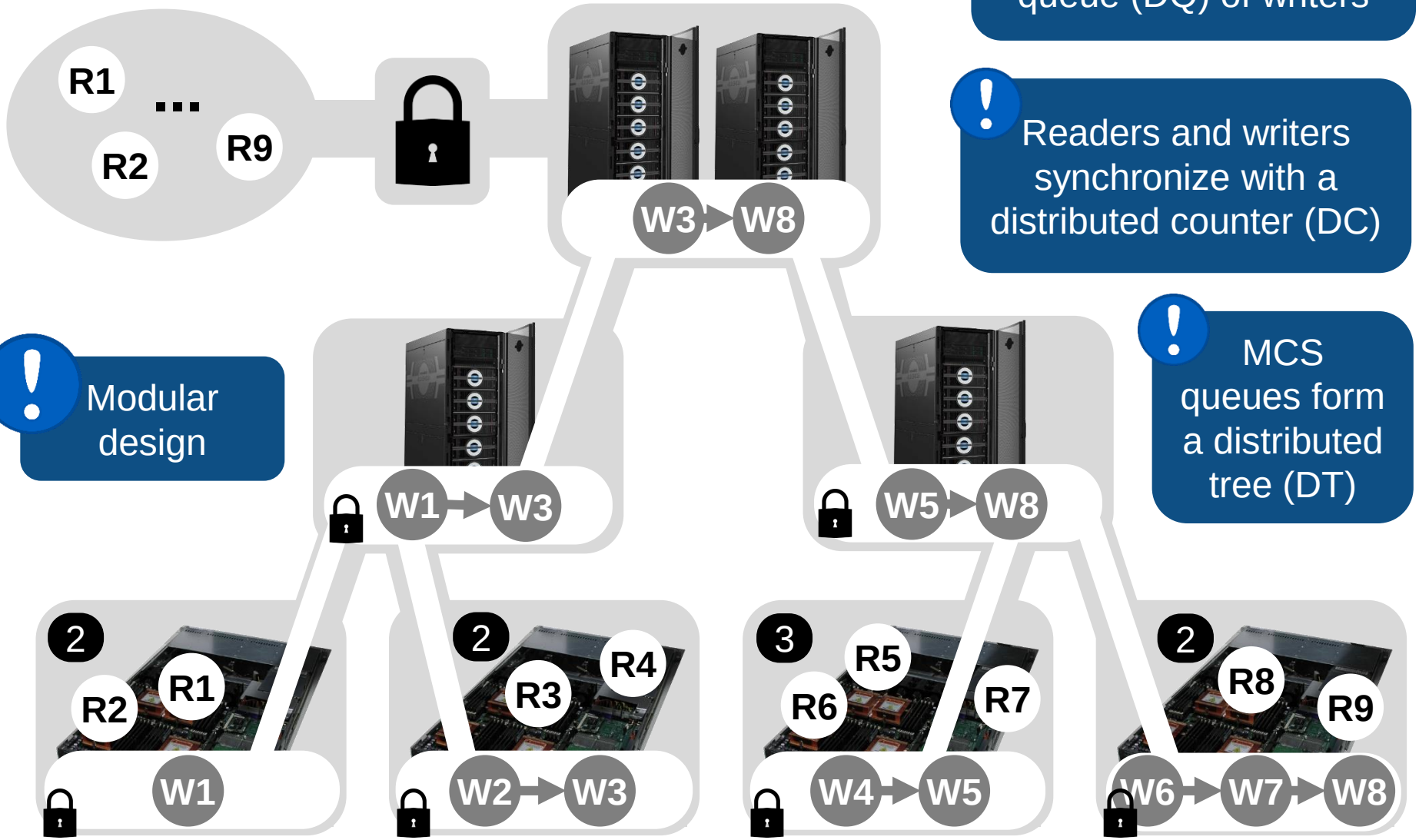
How to manage the design complexity?

Each element has its own distributed MCS queue (DQ) of writers

Readers and writers synchronize with a distributed counter (DC)

MCS queues form a distributed tree (DT)

Modular design



How to ensure tunable performance?

Each DQ: fairness vs throughput of writers

DC: a parameter for the latency of readers vs writers

A tradeoff parameter for every structure

DT: a parameter for the throughput of readers vs writers

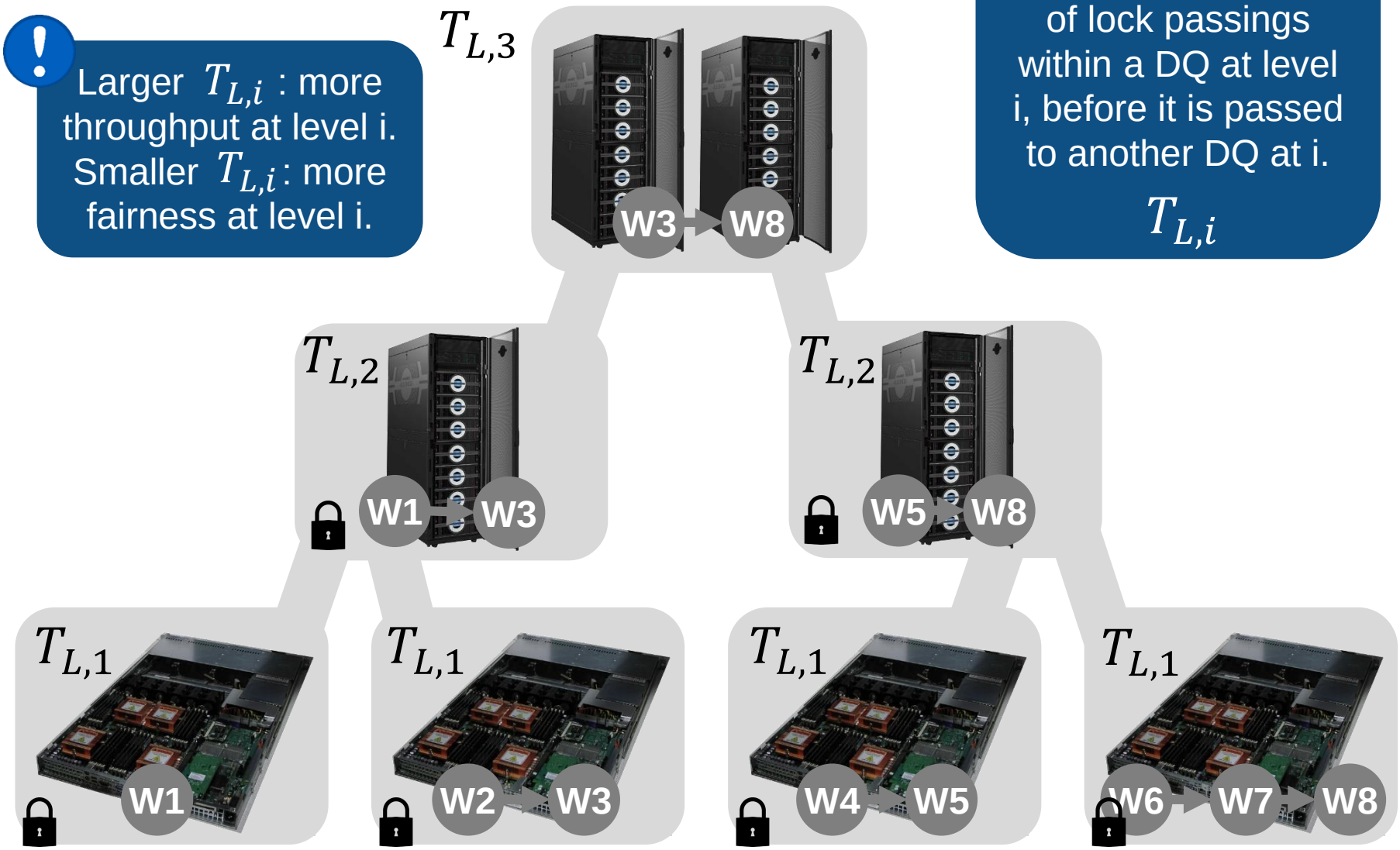
Distributed MCS Queues (DQs)

Throughput vs Fairness

Larger $T_{L,i}$: more throughput at level i.
Smaller $T_{L,i}$: more fairness at level i.

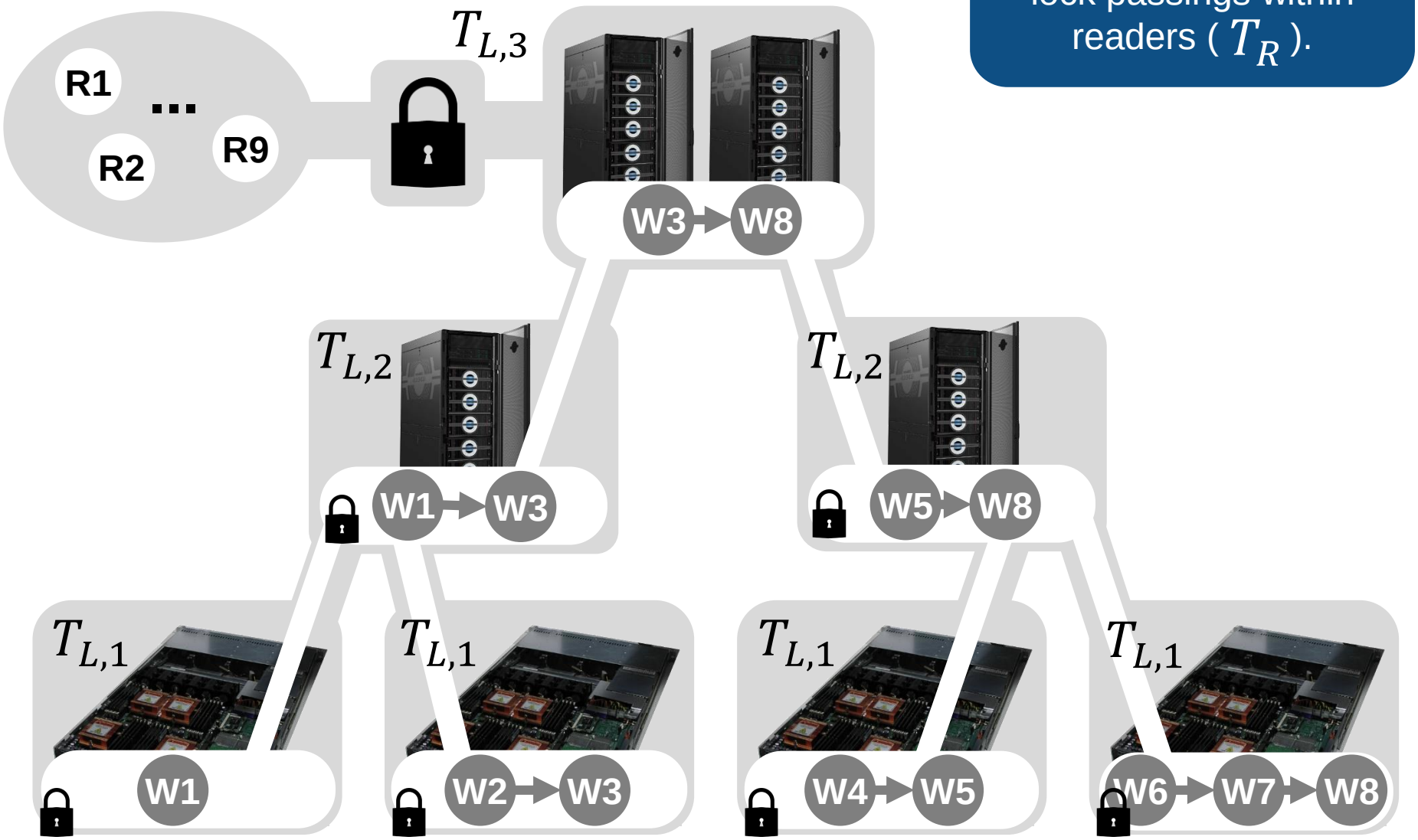
Each DQ: The maximum number of lock passings within a DQ at level i, before it is passed to another DQ at i.

$T_{L,i}$



Distributed Tree of Queues (DT)
Throughput of readers vs writers

DT: The maximum number of consecutive lock passings within readers (T_R).



DISTRIBUTED COUNTER (DC)
Latency of readers vs writers

DC: every *k*th compute node hosts a partial counter, all of which constitute the DC.

$$k = T_{DC}$$

A writer holds the lock

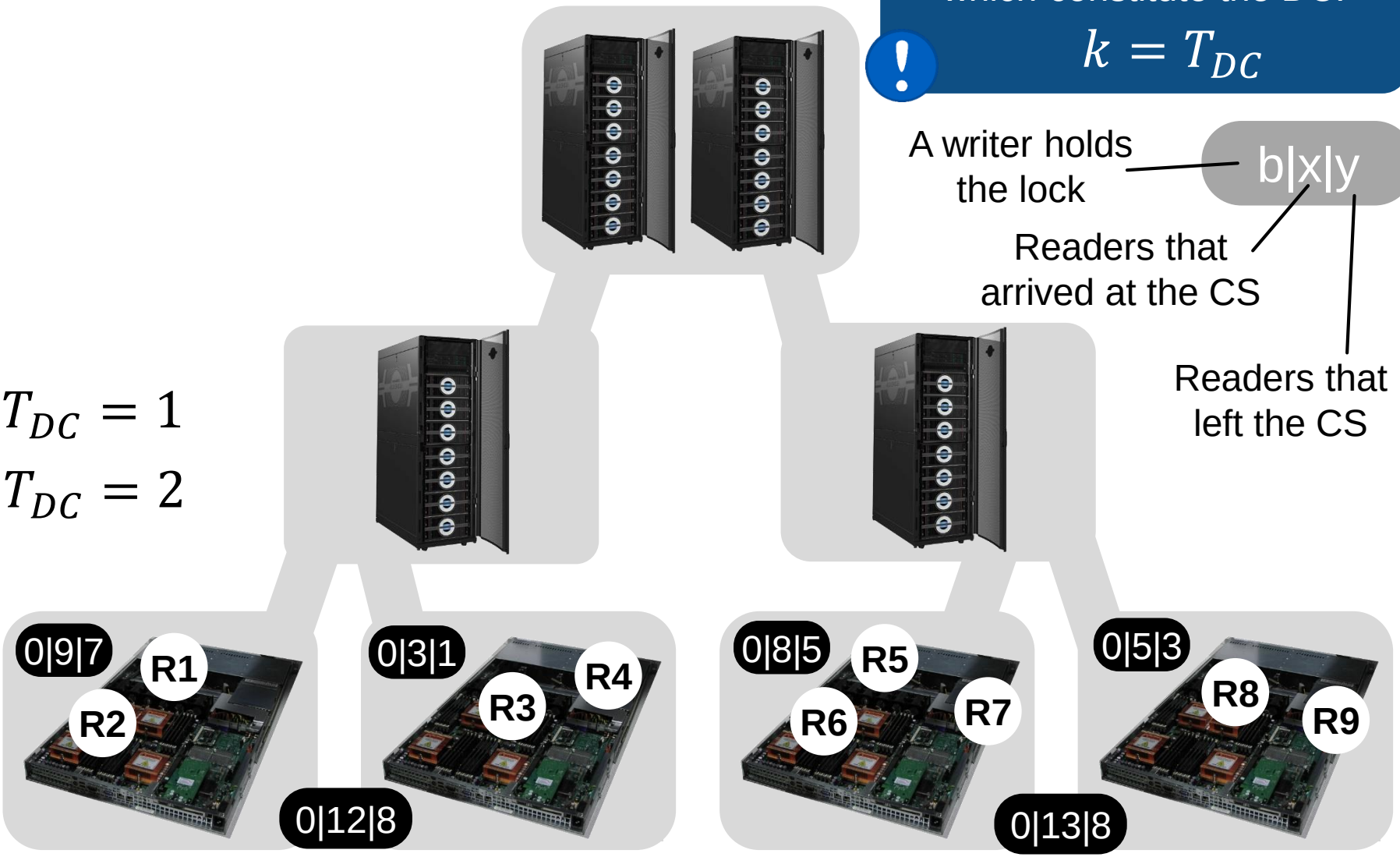
b|x|y

Readers that arrived at the CS

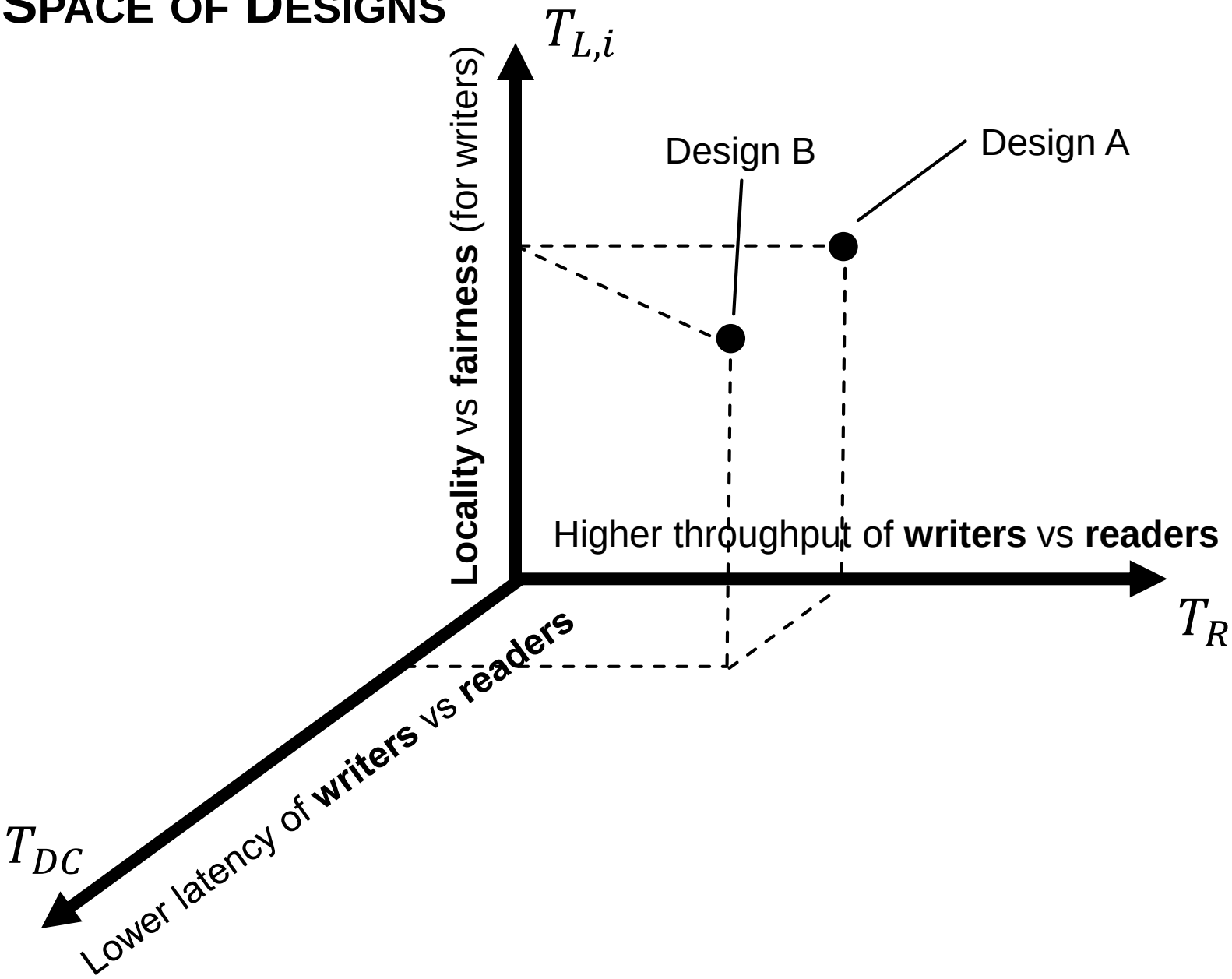
Readers that left the CS

$T_{DC} = 1$

$T_{DC} = 2$

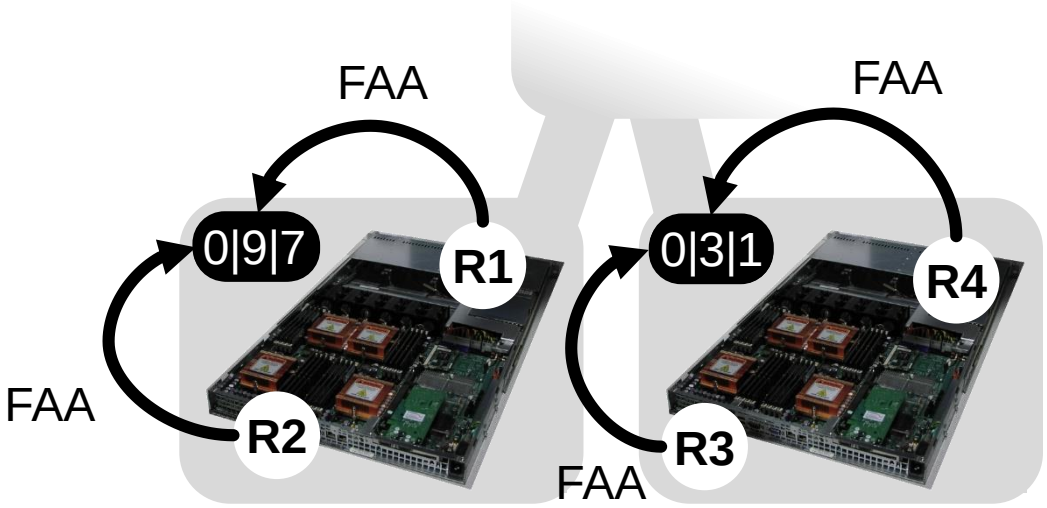


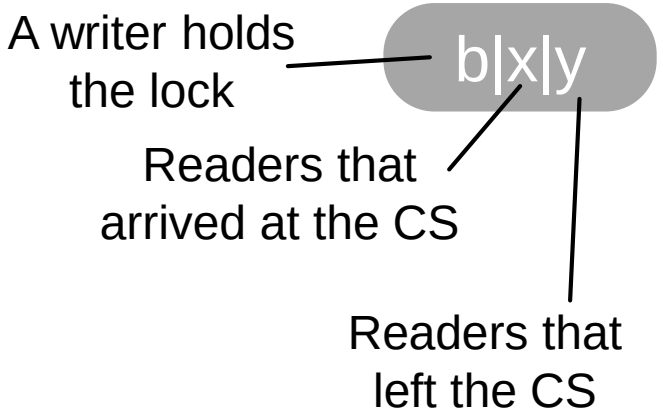
The Space of Designs



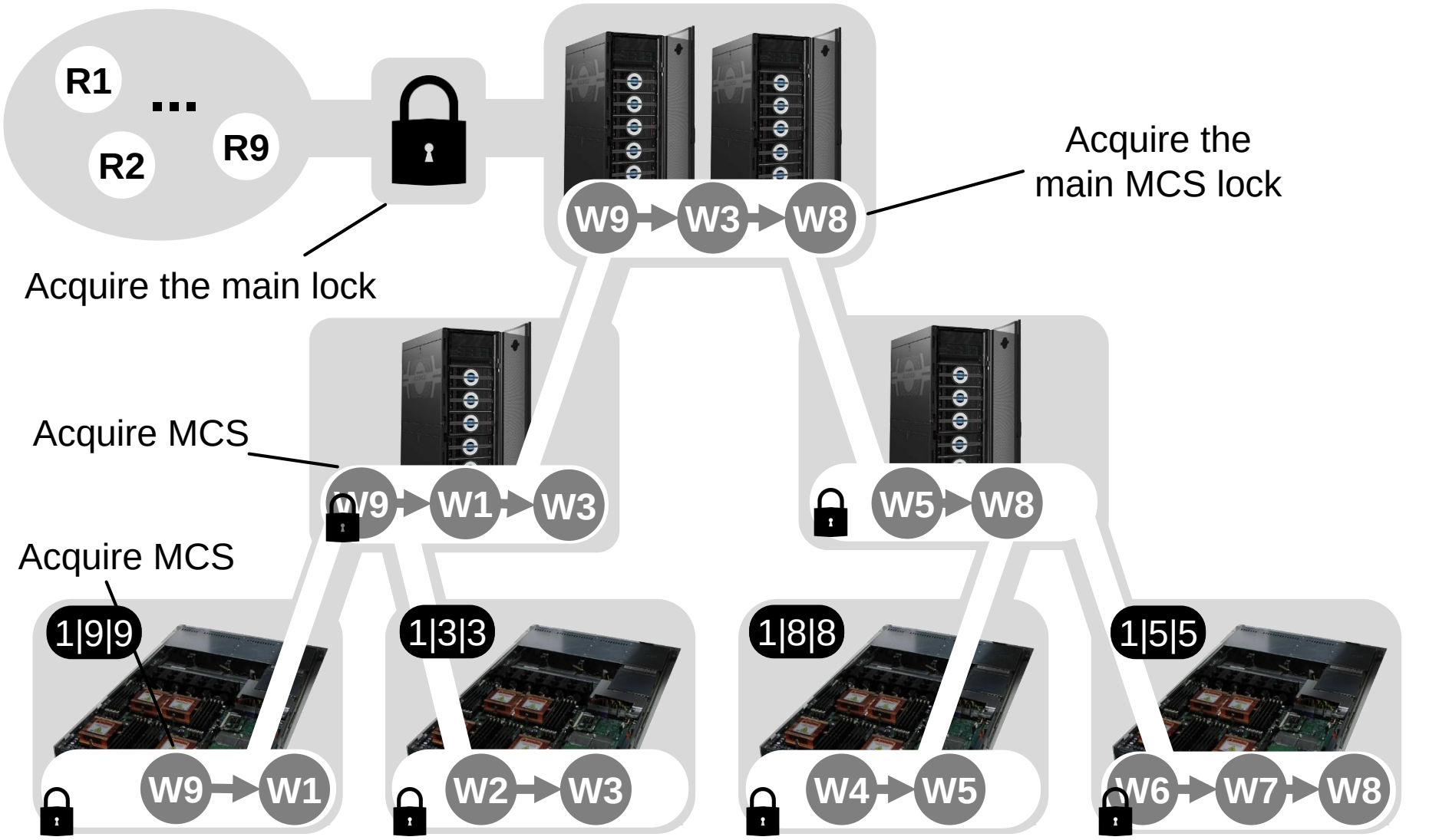
Lock Acquire by Readers

A lightweight acquire protocol for readers: only one atomic fetch-and-add (FAA) operation

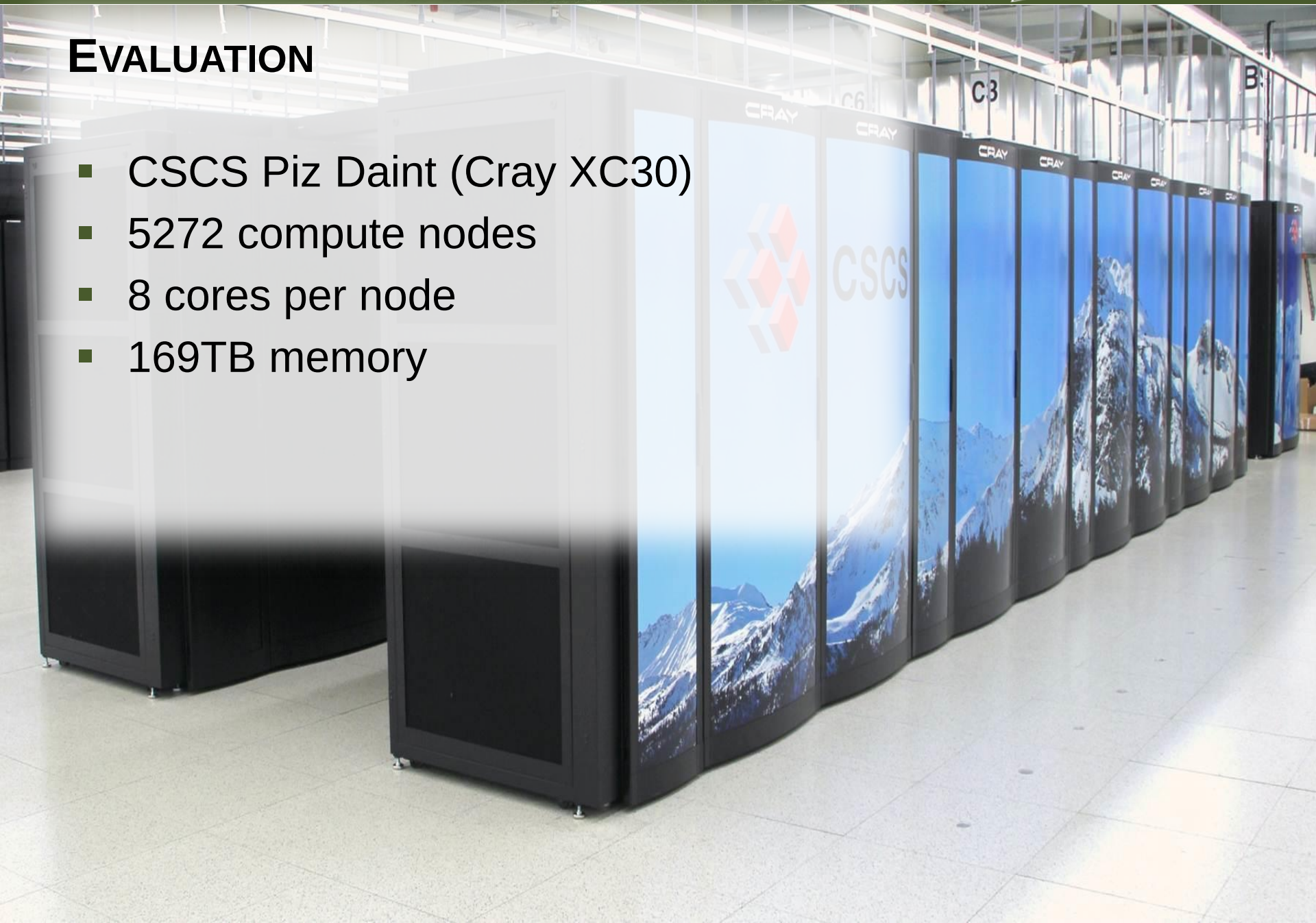




Lock Acquire by Writers



EVALUATION



- CSCS Piz Daint (Cray XC30)
- 5272 compute nodes
- 8 cores per node
- 169TB memory

Evaluation
Considered Benchmarks

The **latency** benchmark

Throughput benchmarks:

- Empty-critical-section
- Single-operation
- Wait-after-release
- Workload-critical-section

DHT

Distributed hashtable evaluation

Evaluation

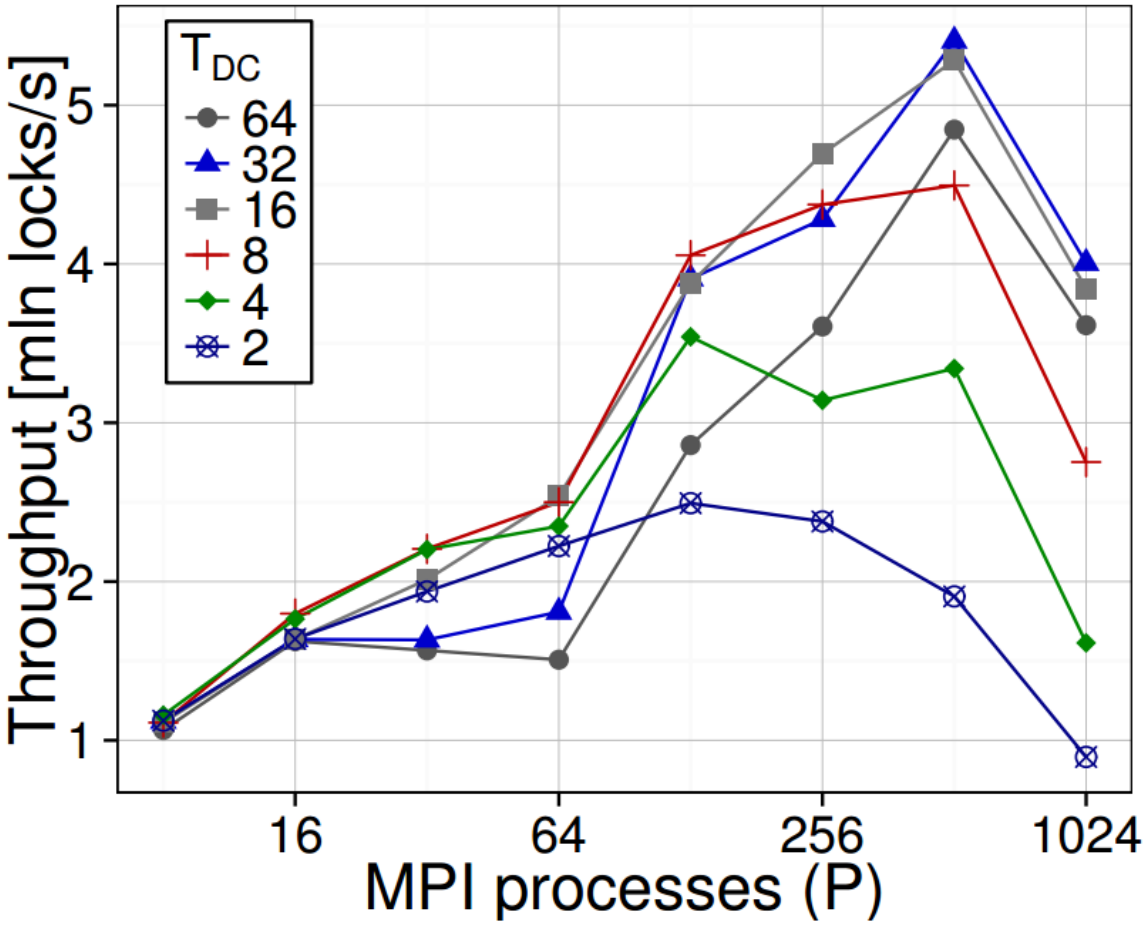
Distributed Counter Analysis

0|9|7 0|3|1

0|12|8

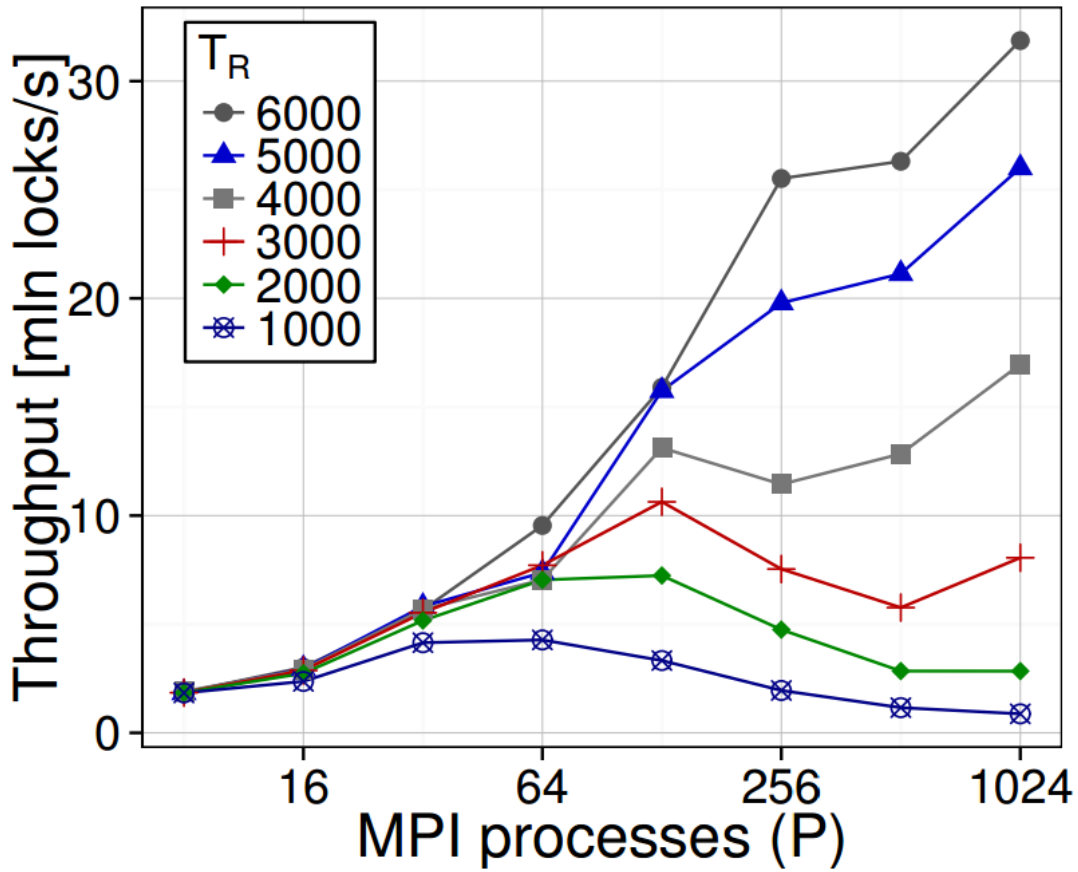
Throughput, 2% writers

Single-operation benchmark



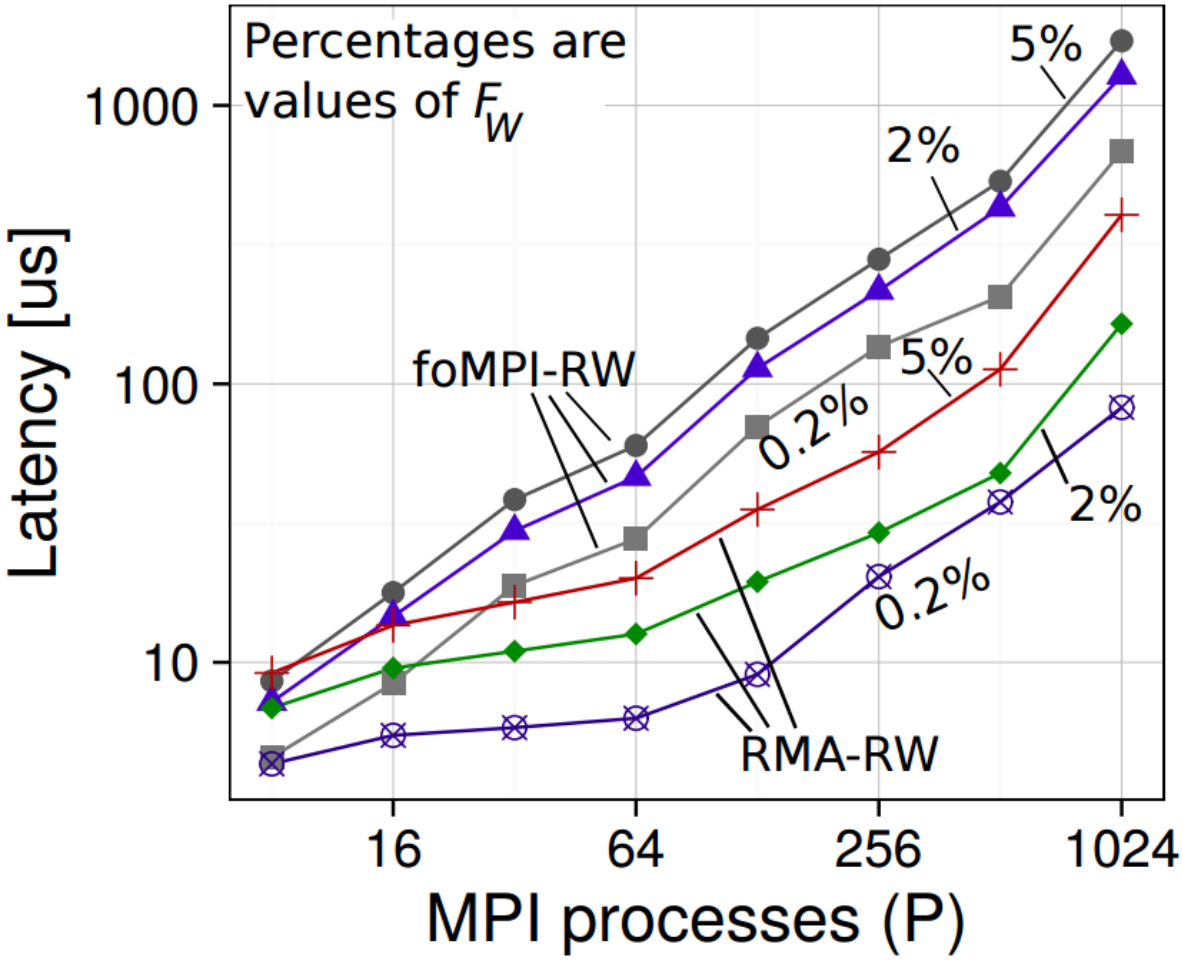
EVALUATION
READER THRESHOLD ANALYSIS

Throughput, 0.2% writers,
Empty-critical-section benchmark



EVALUATION

COMPARISON TO THE STATE-OF-THE-ART

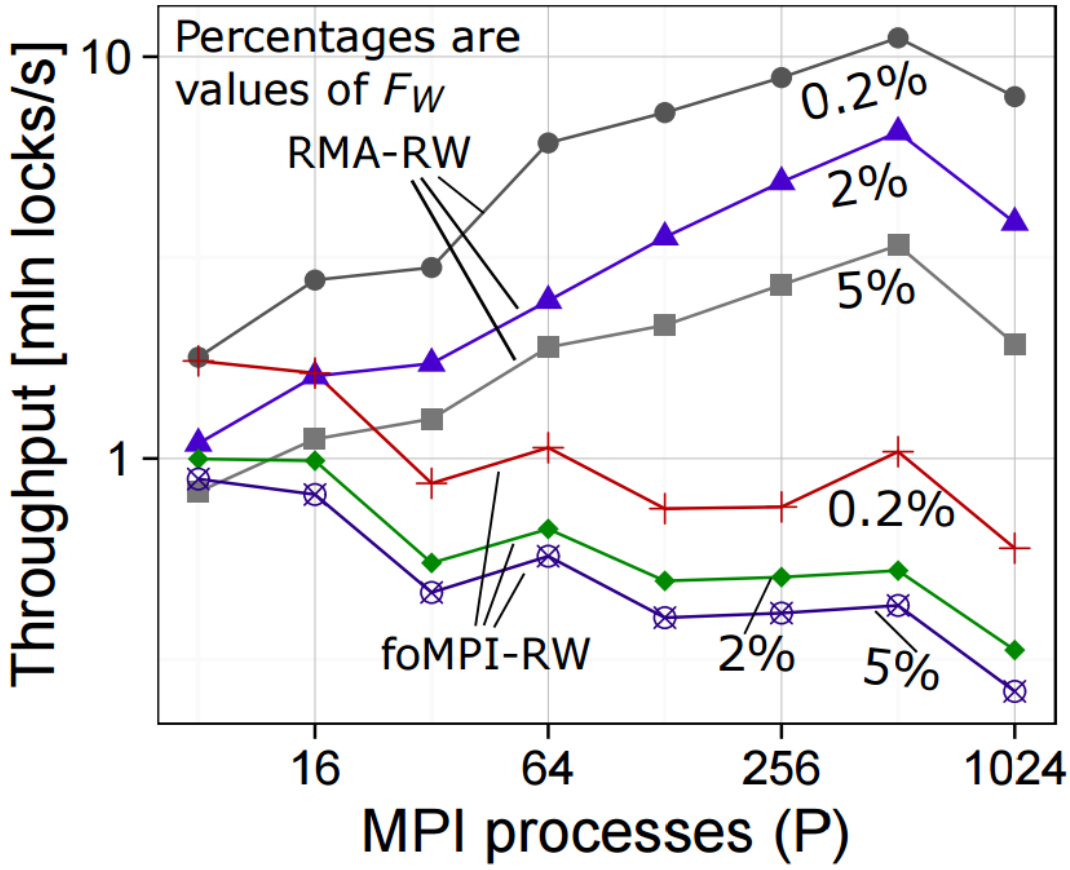


[1] R. Gerstenberger et al. Enabling Highly-scalable Remote Memory Access Programming with MPI-3 One Sided. ACM/IEEE Supercomputing 2013.

EVALUATION

COMPARISON TO THE STATE-OF-THE-ART

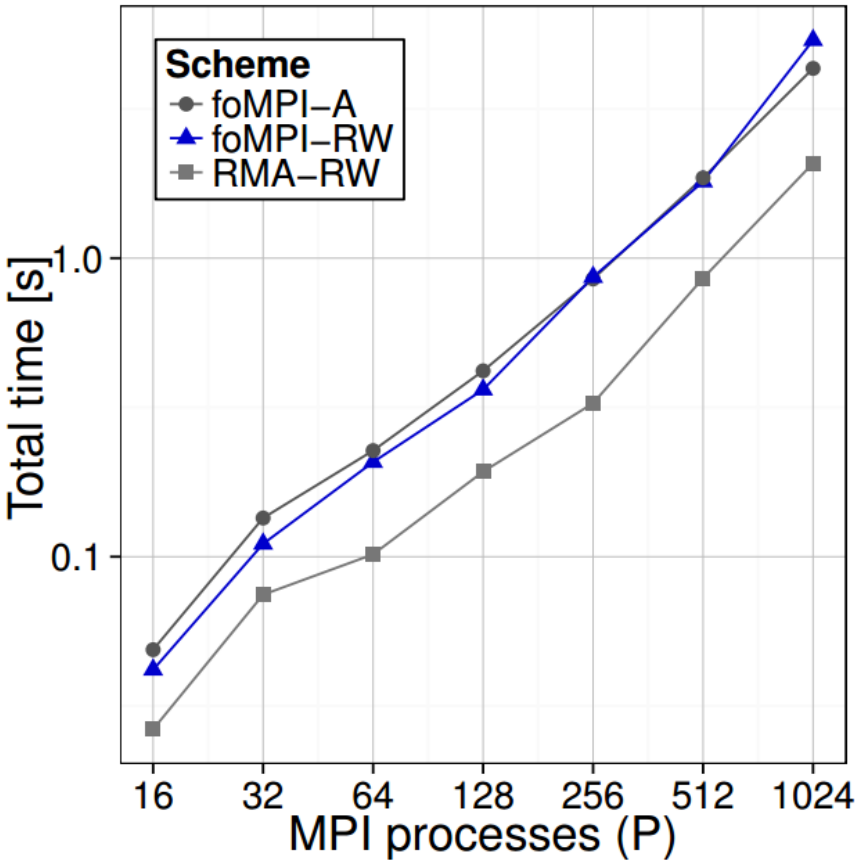
Throughput, single-operation benchmark



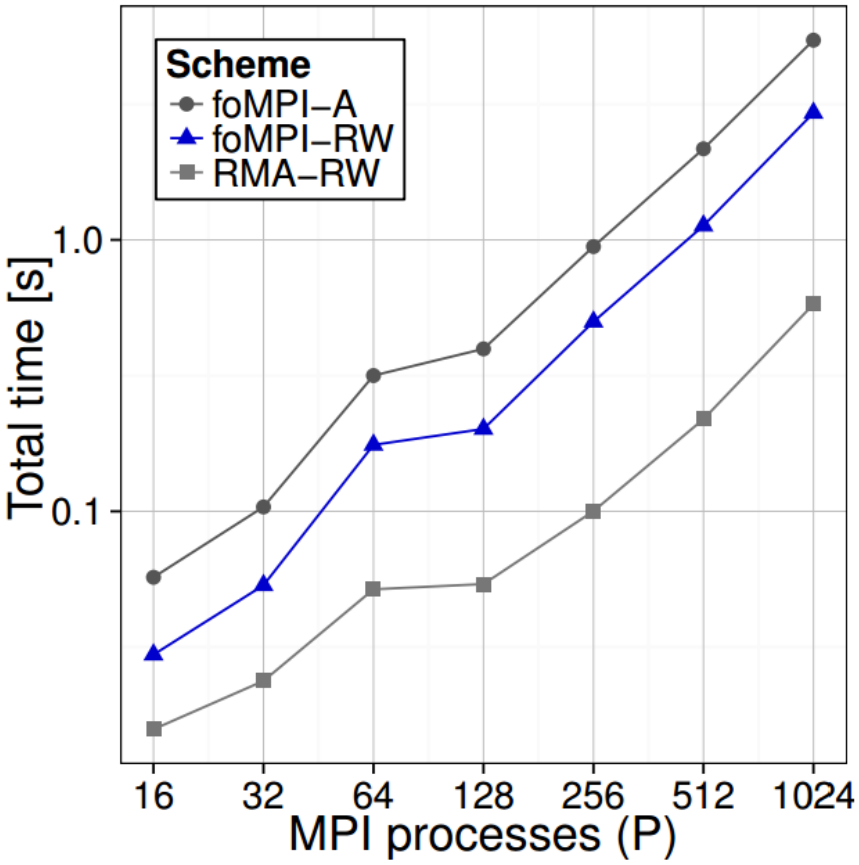
[1] R. Gerstenberger et al. Enabling Highly-scalable Remote Memory Access Programming with MPI-3 One Sided. ACM/IEEE Supercomputing 2013.

EVALUATION
DISTRIBUTED HASHTABLE

20% writers



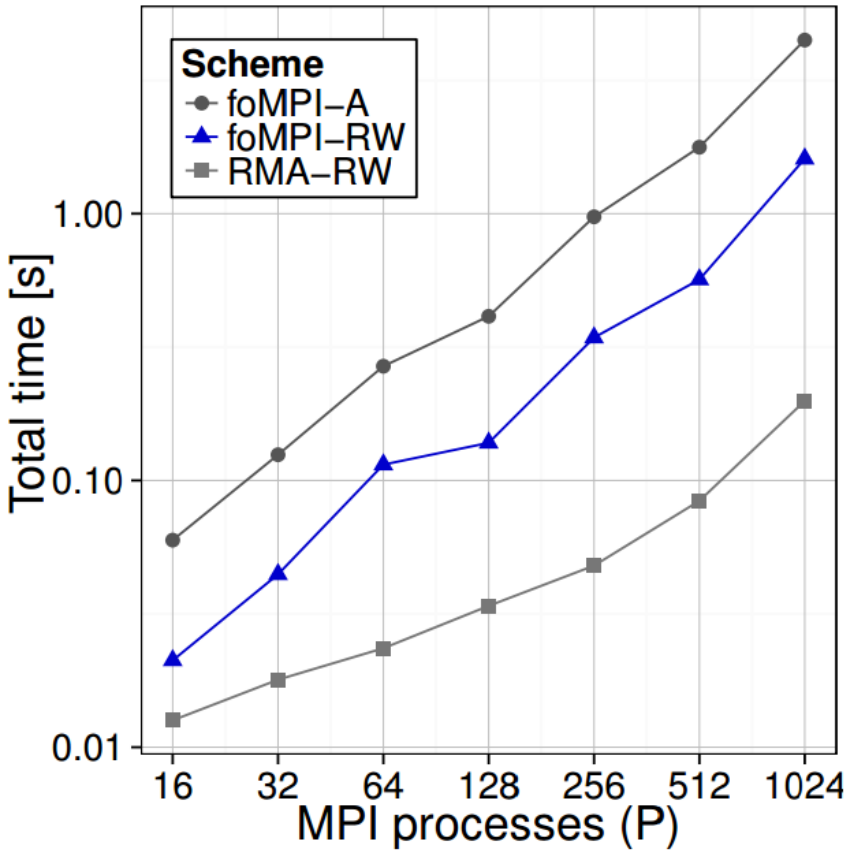
10% writers



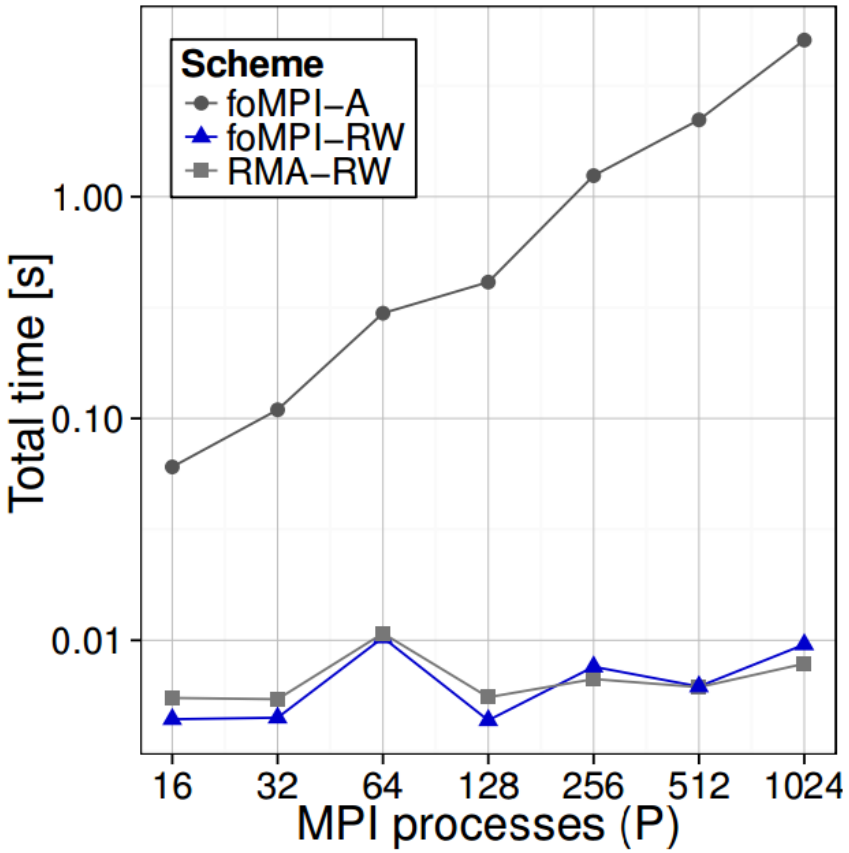
[1] R. Gerstenberger et al. Enabling Highly-scalable Remote Memory Access Programming with MPI-3 One Sided. ACM/IEEE Supercomputing 2013.

EVALUATION
DISTRIBUTED HASHTABLE

2% of writers



0% of writers



[1] R. Gerstenberger et al. Enabling Highly-scalable Remote Memory Access Programming with MPI-3 One Sided. ACM/IEEE Supercomputing 2013.

OTHER ANALYSES