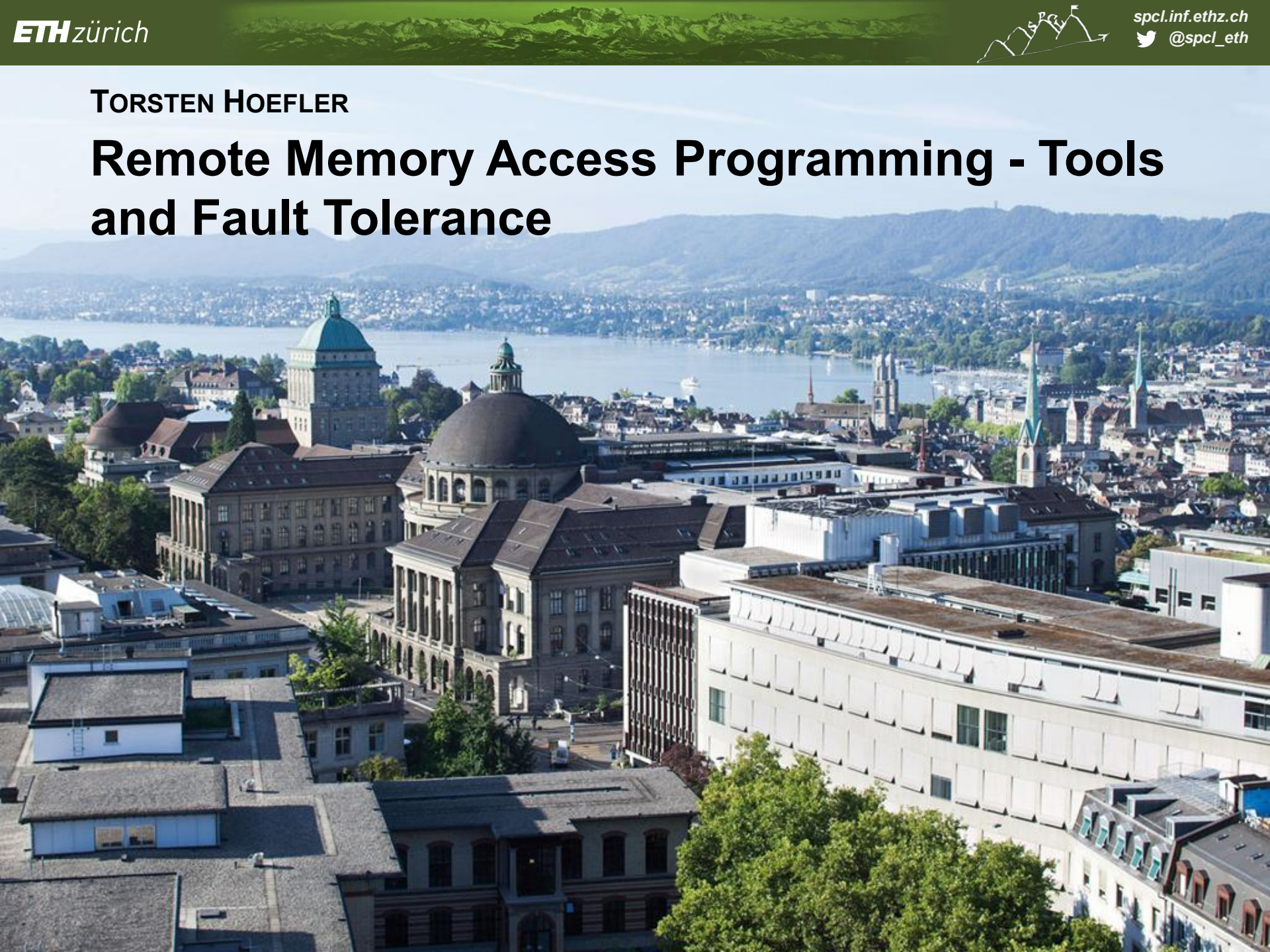
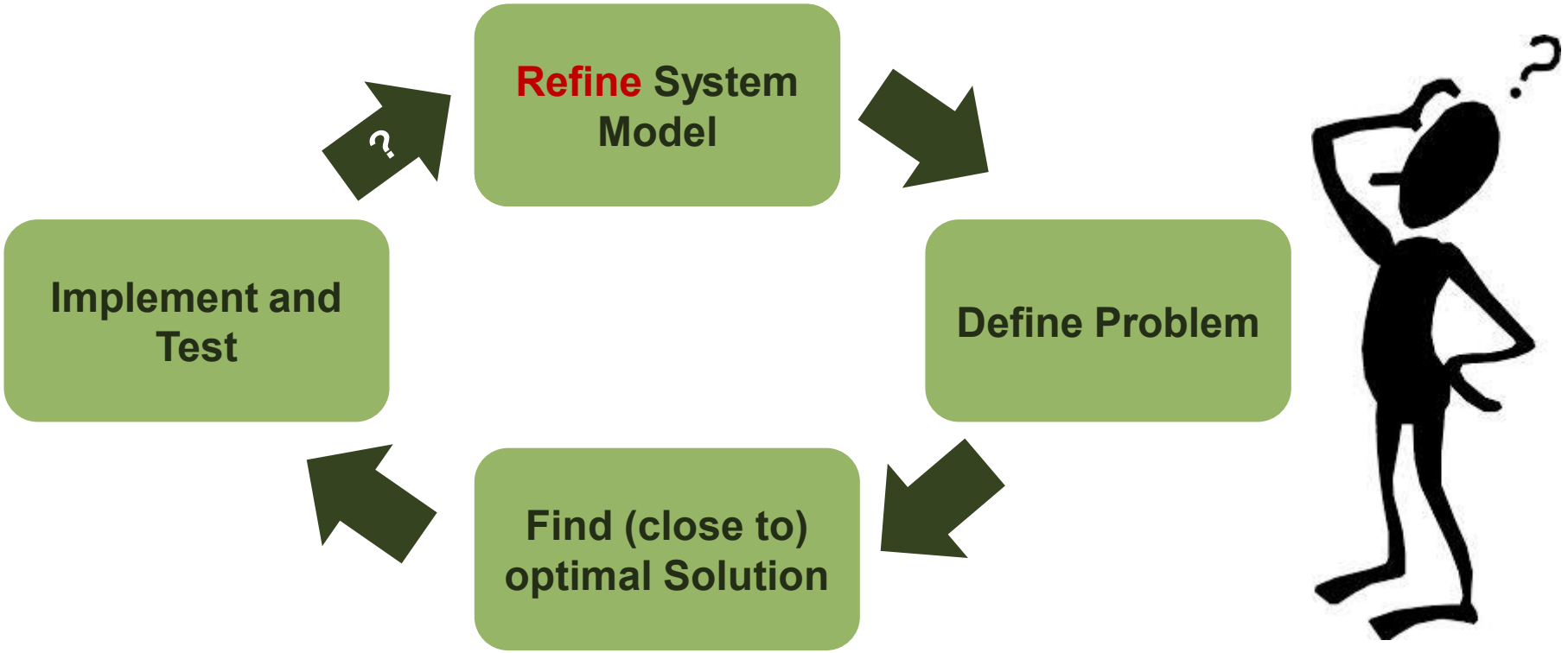
TORSTEN HOEFLER

Remote Memory Access Programming - Tools and Fault Tolerance



Model-based Performance Engineering

- **My dream: provably optimal performance (time and energy)**
  - From problem to machine code

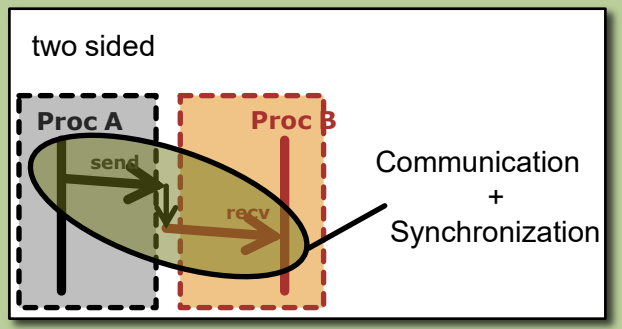


- **A philosophy for system and application design**
  - At different levels of course

State of the Art – Parallel Programming

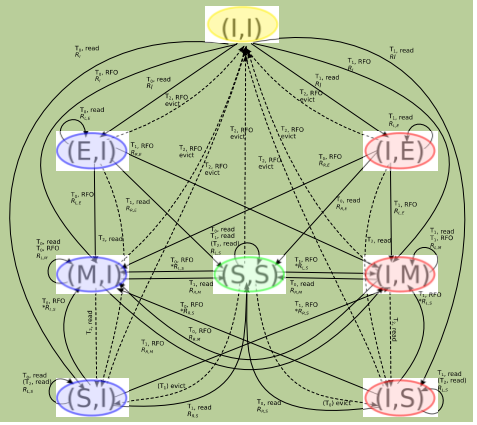
Message Passing (MPI-1)

- De-facto programming model



Coherent Shared Memory

- Hardware support ☺
- Very very hard to optimize [1]

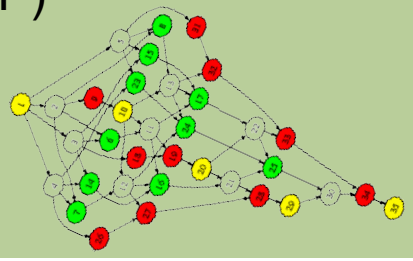


Requirements for a new low-level programming model

1. Messaging needs 100% hardware support (offload) – it's simple after all!
2. Minimal overheads (**tiny**) layer between user and hardware
3. Offer a simple abstract performance model (e.g., LogGP)

DPLASMA

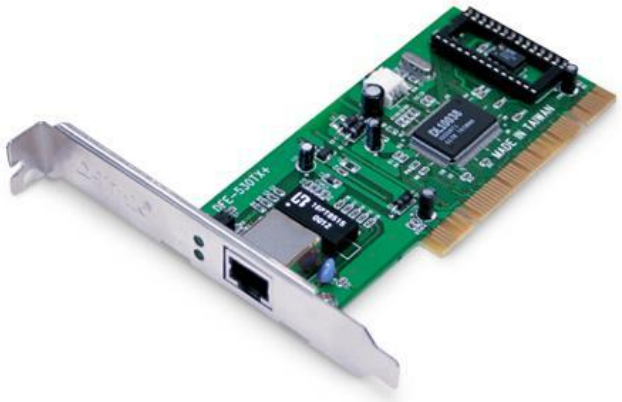
CONSORTIUM FOR SMALL SCALE MODELING
COSMO



Ramos, TH: "Modeling Communication in Cache-Coherent SMP Systems - A Case-Study with Xeon Phi", HPDC'13

MPI-3.0 RMA

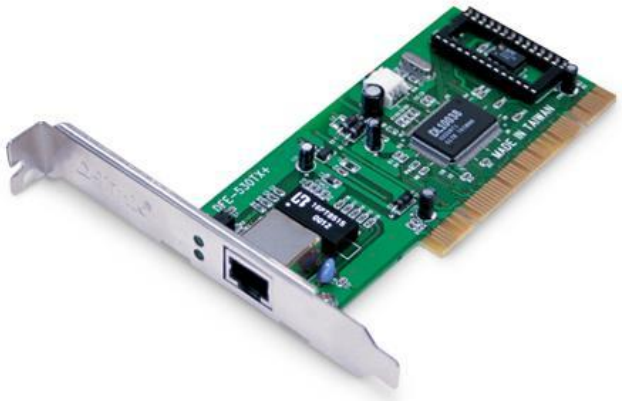
- MPI-3.0 supports RMA ("MPI One Sided")
  - Designed to react to hardware trends
  - Majority of HPC networks support RDMA



MPI Forum, TH: „MPI: A Message-Passing Interface Standard Version 3.0" http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

MPI-3.0 RMA

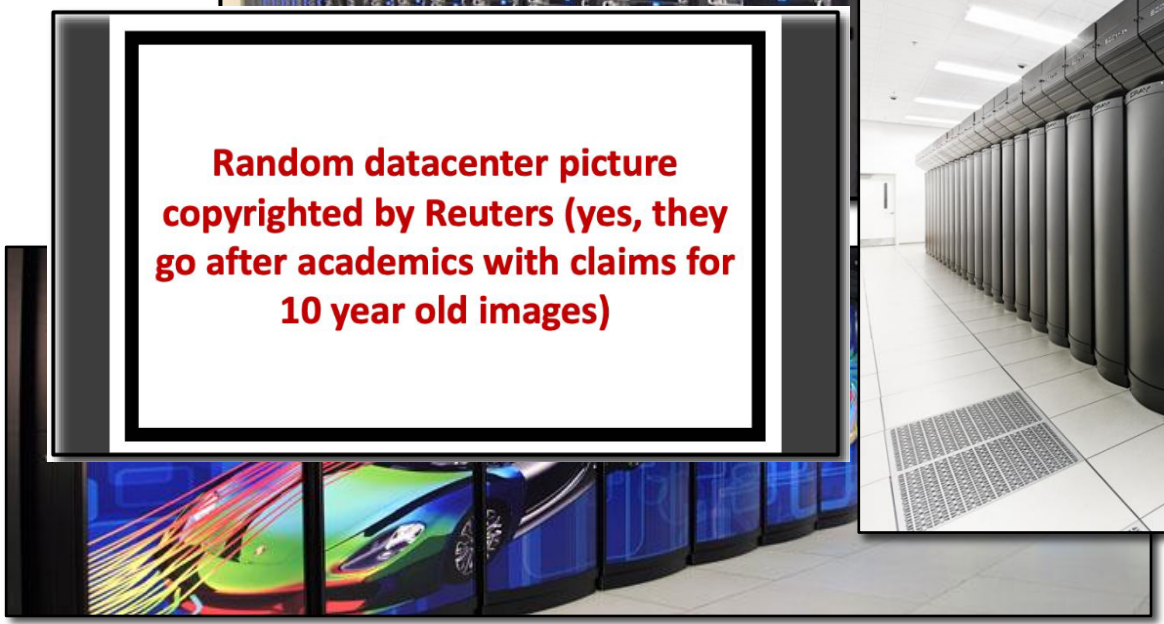
- MPI-3.0 supports RMA ("MPI One Sided")
  - Designed to react to hardware trends
  - Majority of HPC networks support RDMA



MPI Forum, TH: „MPI: A Message-Passing Interface Standard Version 3.0" http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

MPI-3.0 RMA

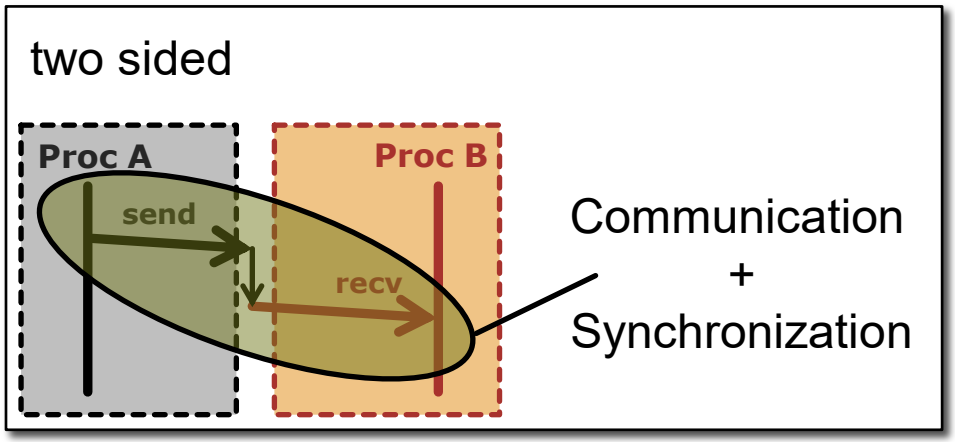
- MPI-3.0 supports RMA ("MPI One Sided")
  - Designed to react to hardware trends
  - Majority of HPC networks support RDMA

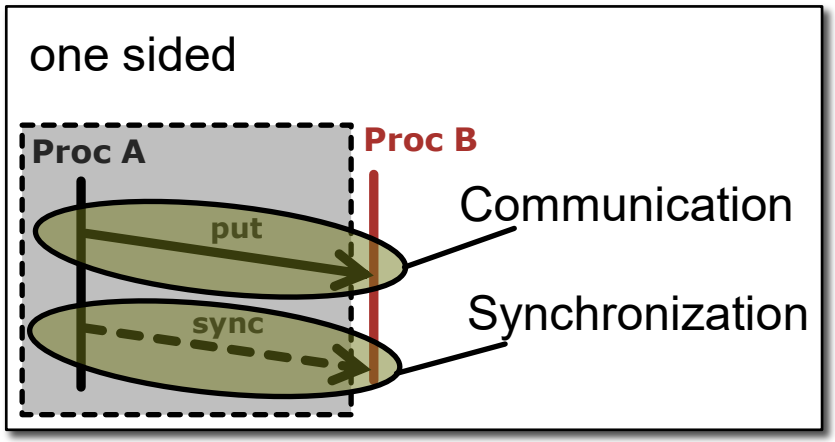


MPI-3.0 RMA

Random datacenter picture copyrighted by Reuters (yes, they go after academics with claims for 10 year old images)

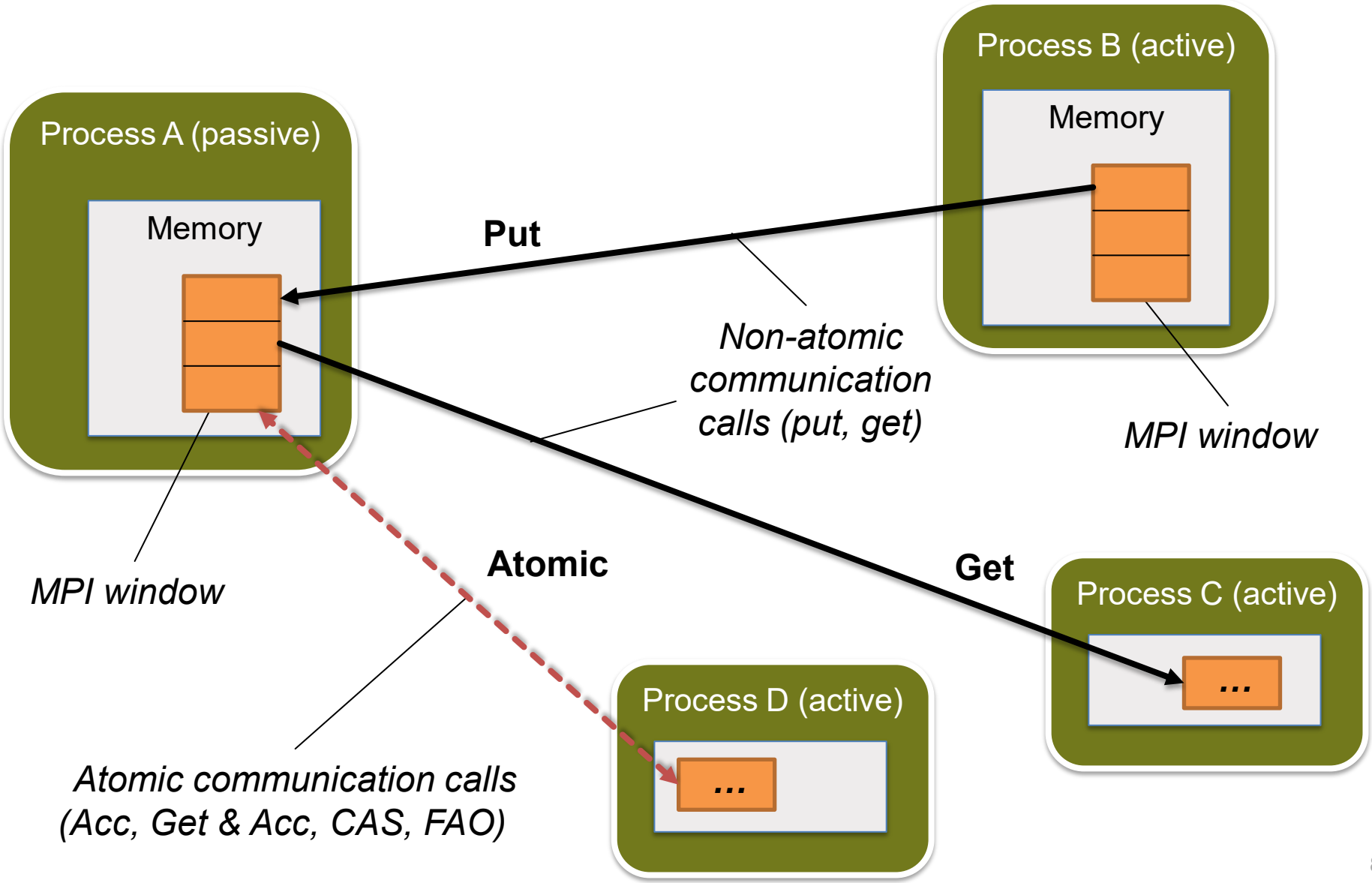
- MPI-3.0 supports RMA ("MPI One Sided")
  - Designed to react to hardware trends
  - Majority of HPC networks support RDMA
- Communication is „one sided" (no involvement of destination)
- RMA decouples communication & synchronization
  - Different from message passing



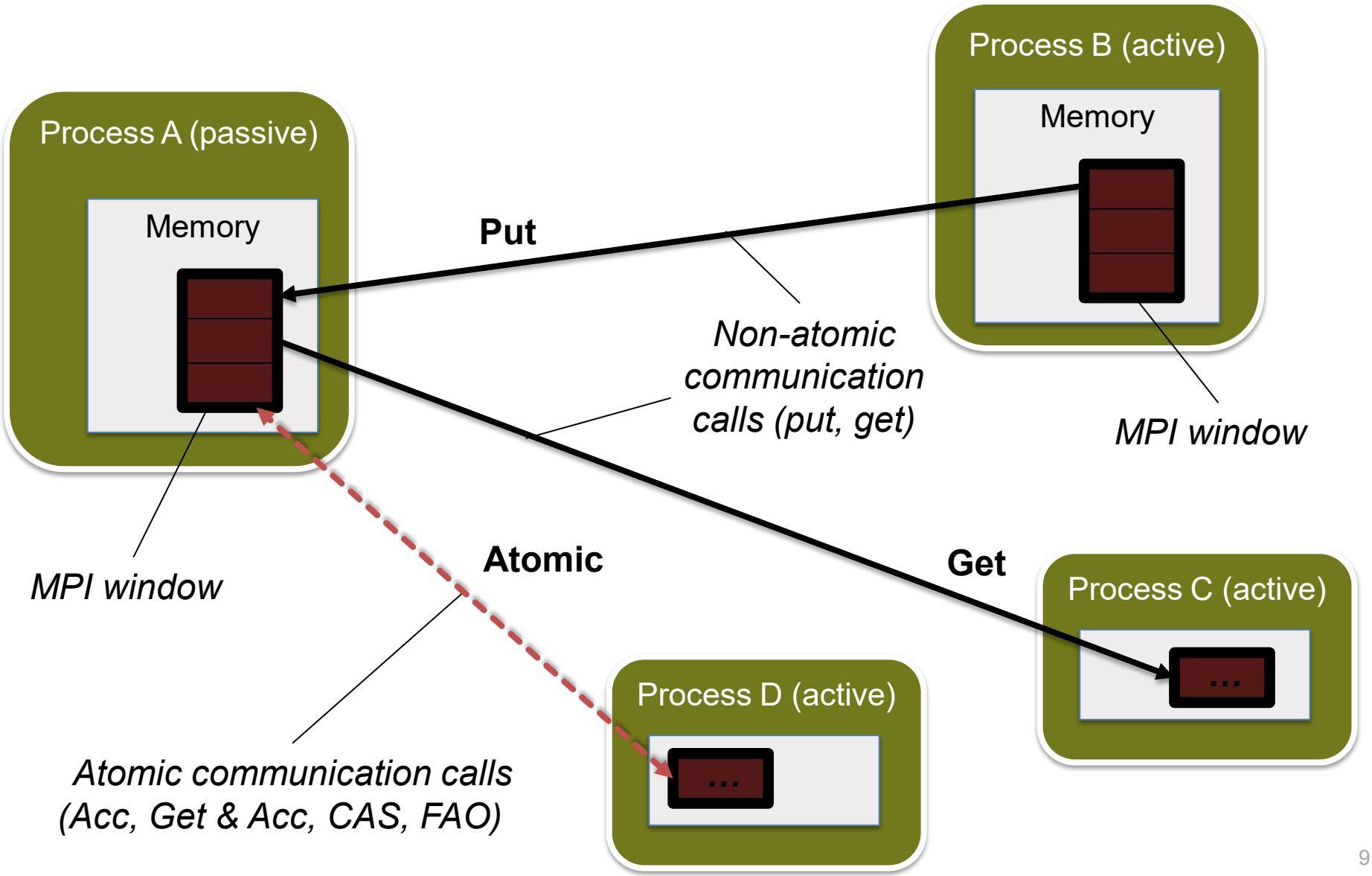


MPI Forum, TH: „MPI: A Message-Passing Interface Standard Version 3.0" http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

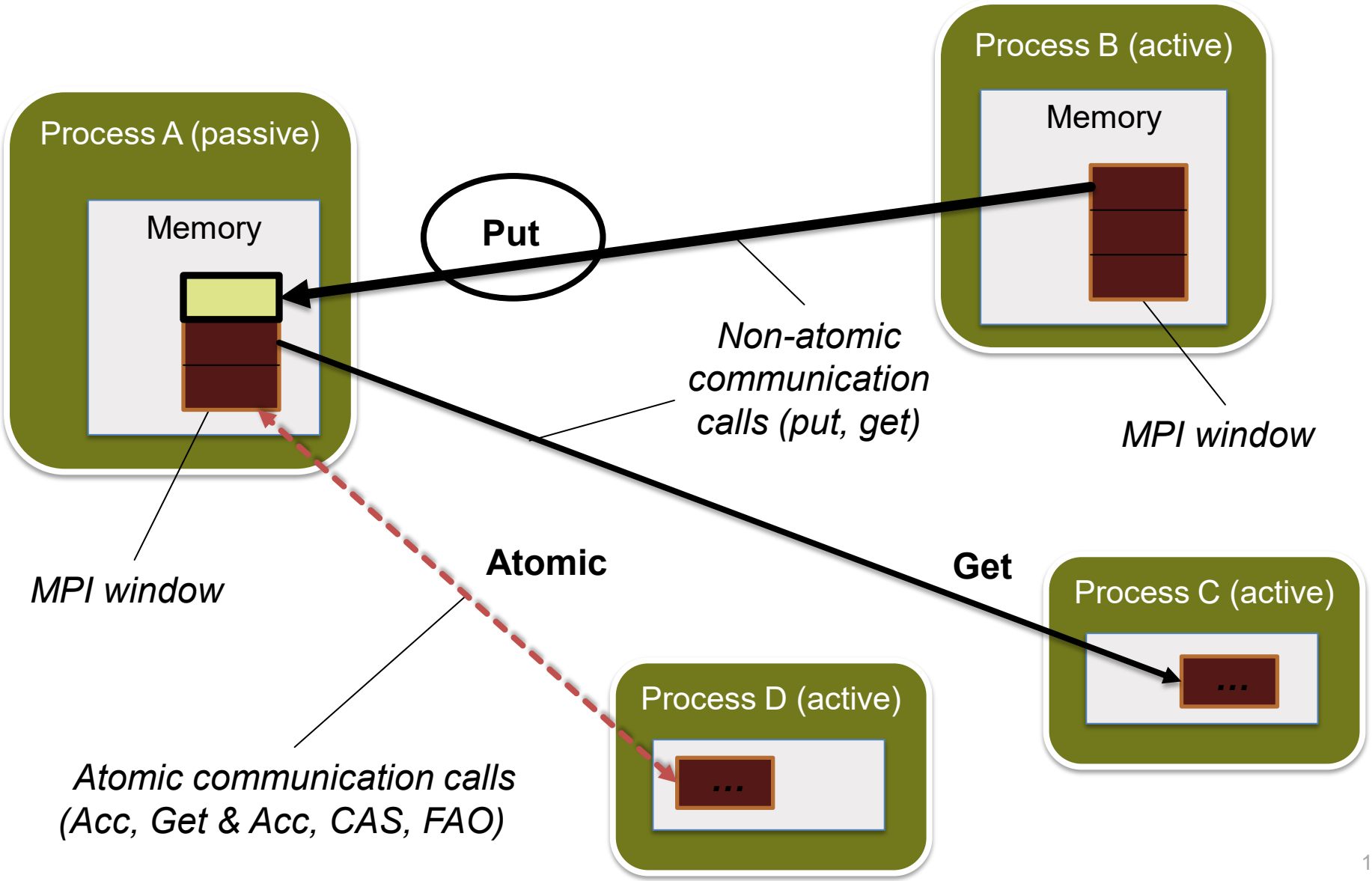
MPI-3 RMA Communication Overview



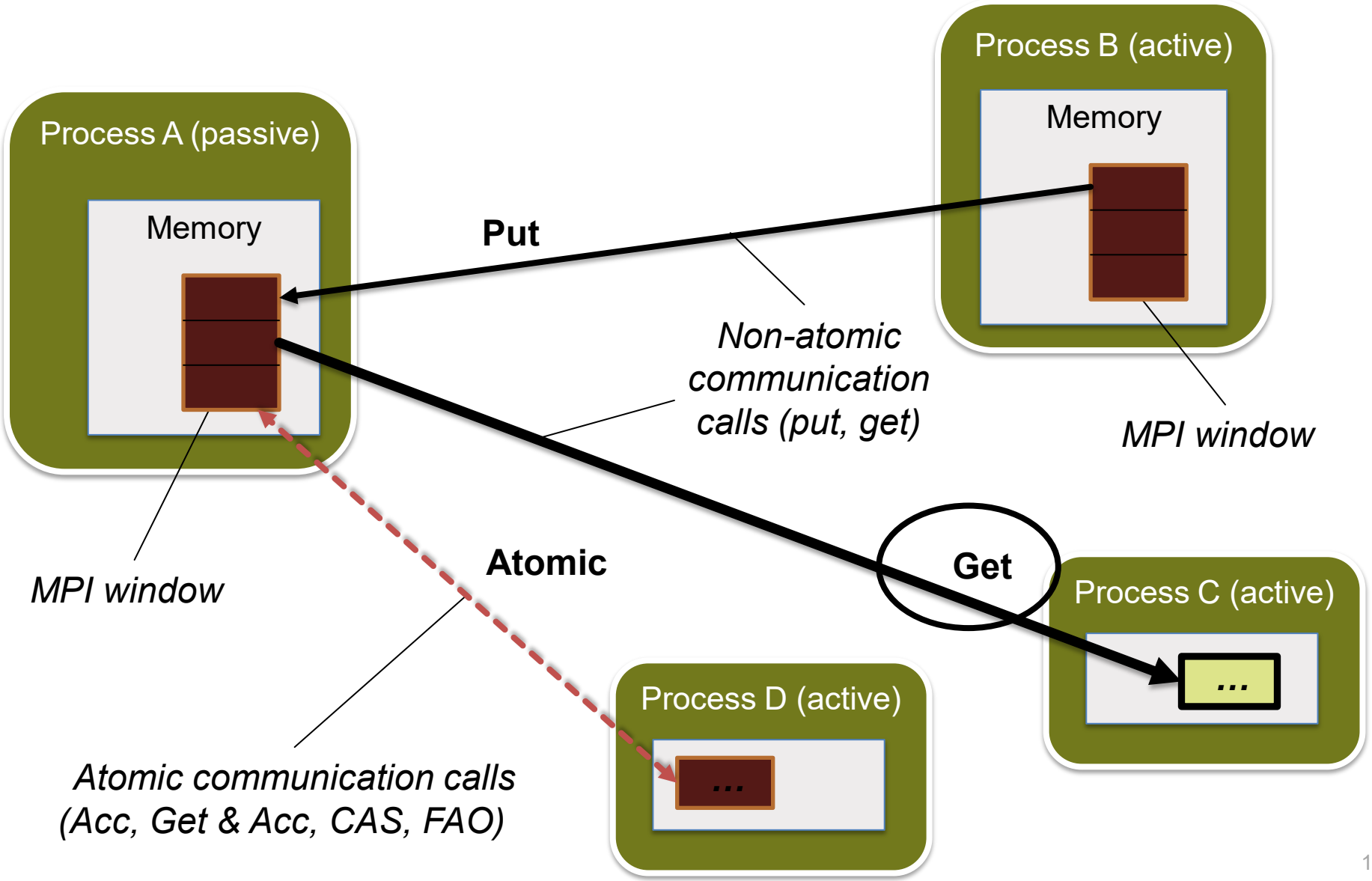
MPI-3 RMA Communication Overview



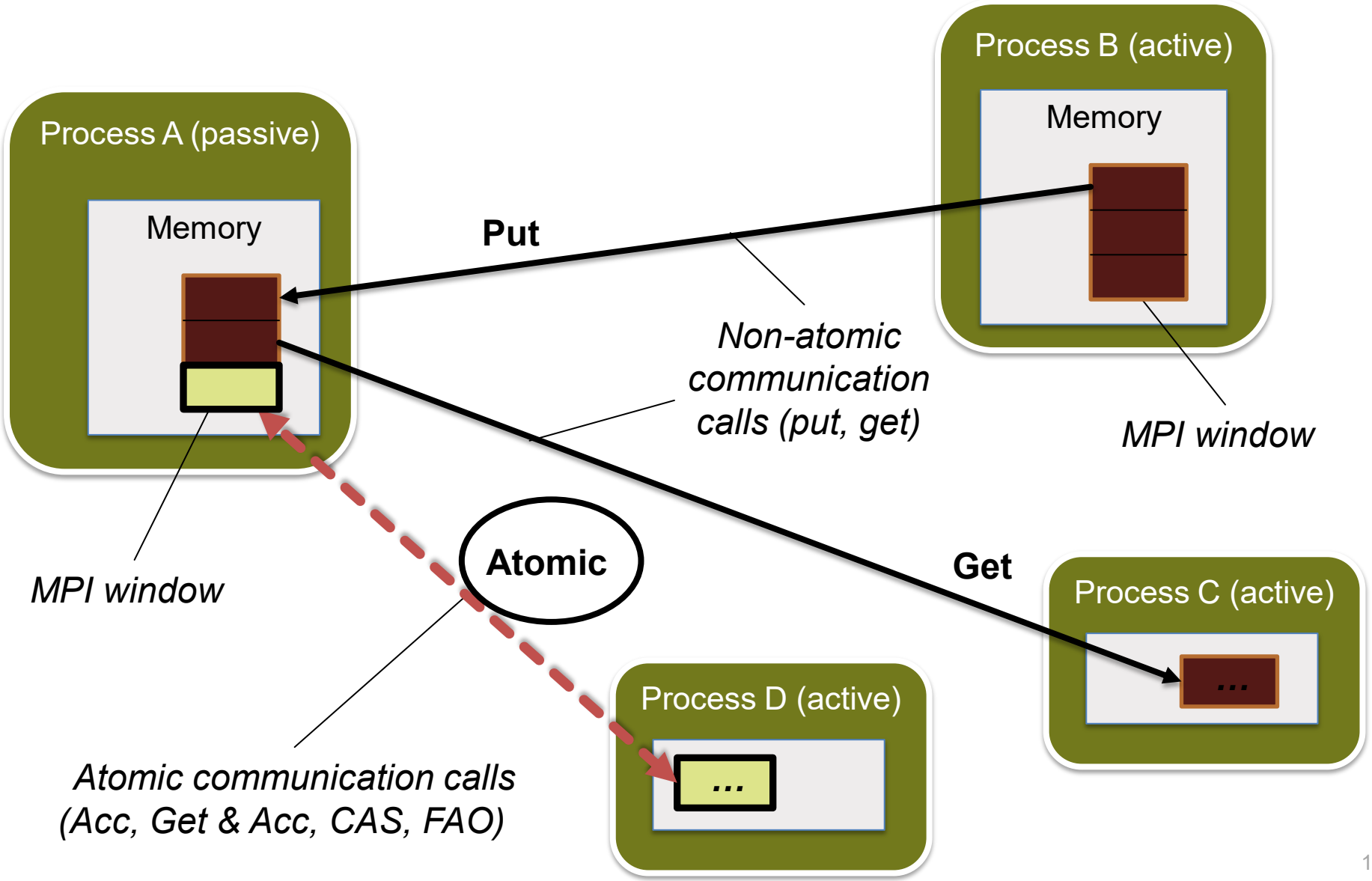
MPI-3 RMA Communication Overview



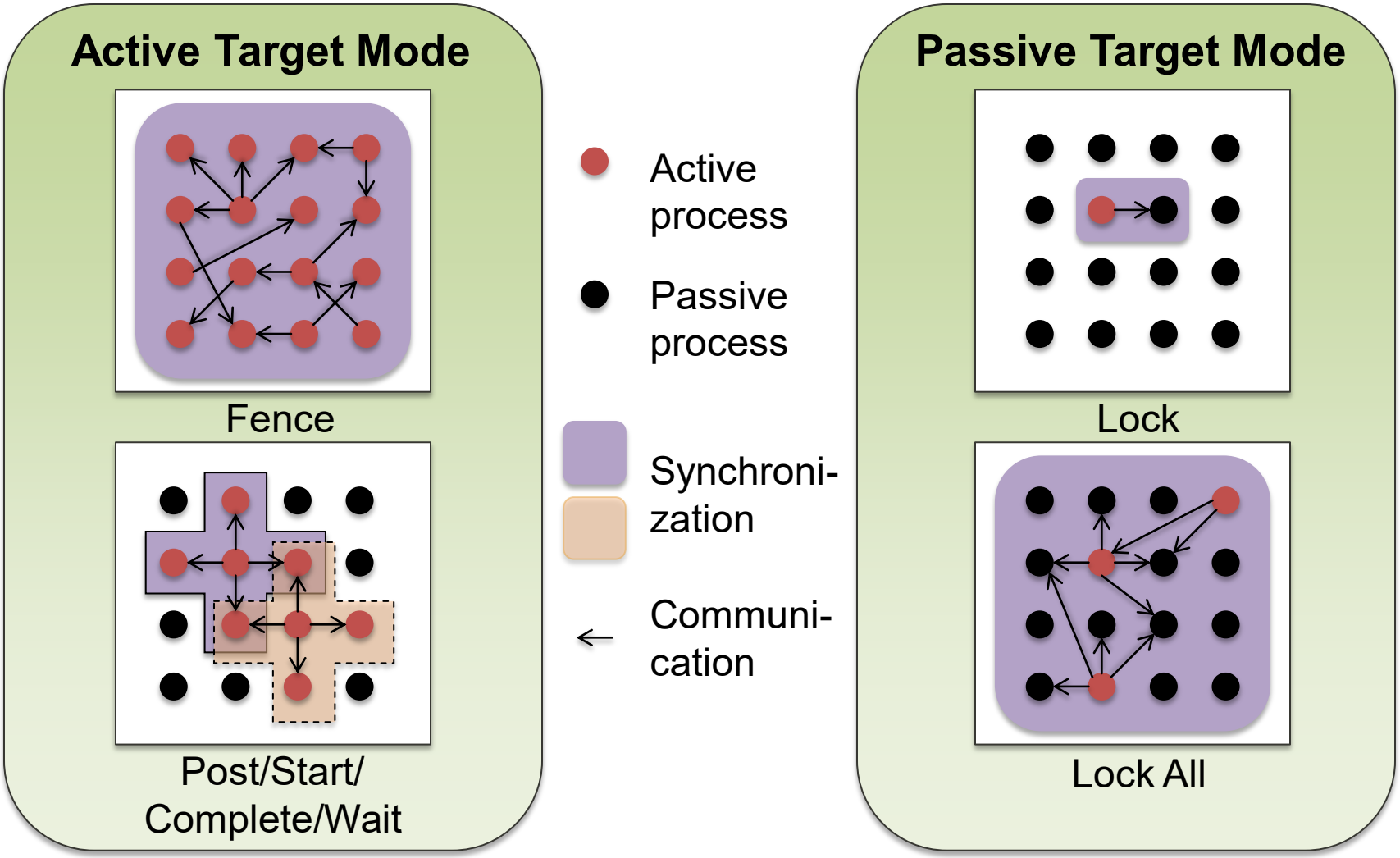
MPI-3 RMA Communication Overview



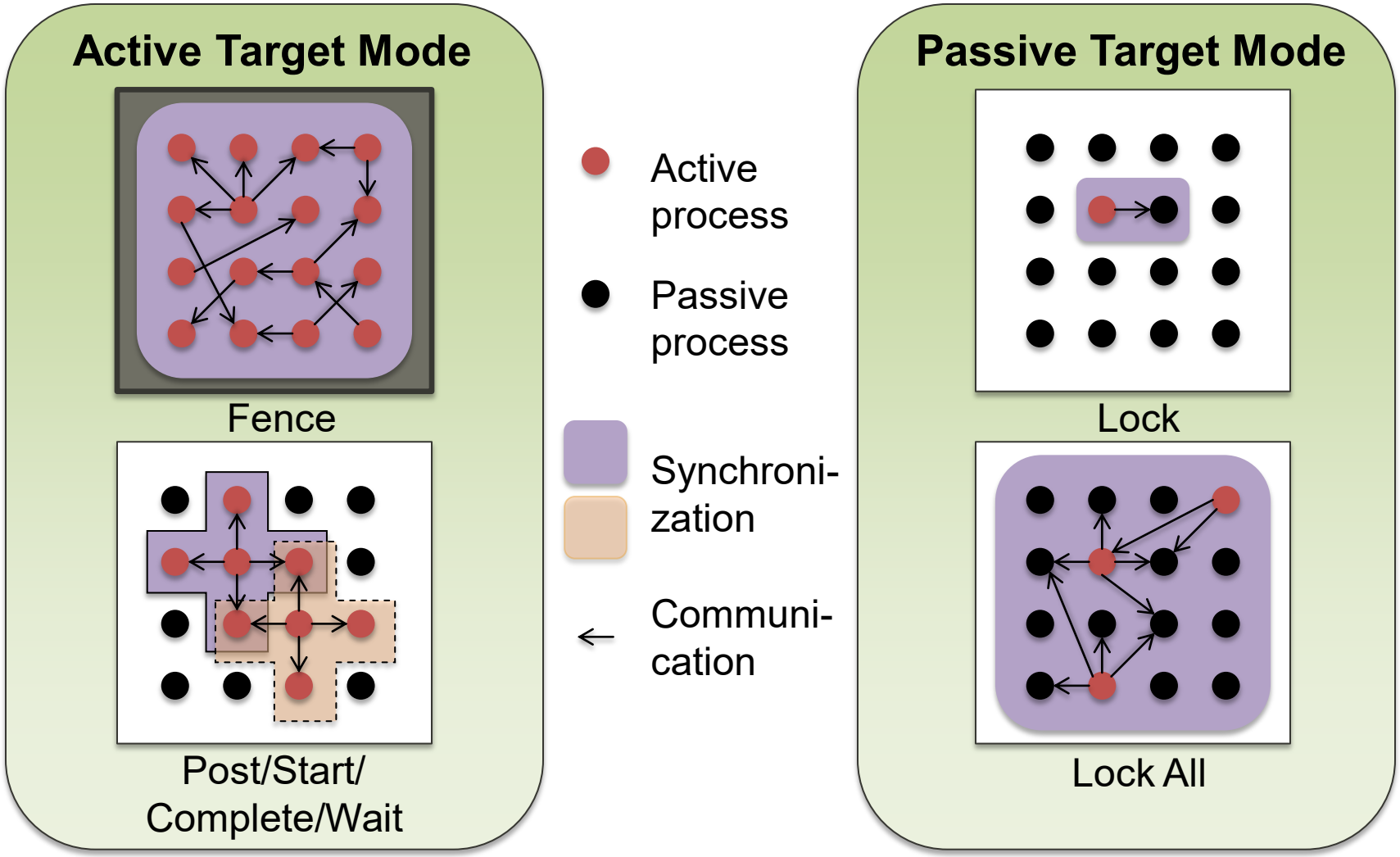
MPI-3 RMA Communication Overview



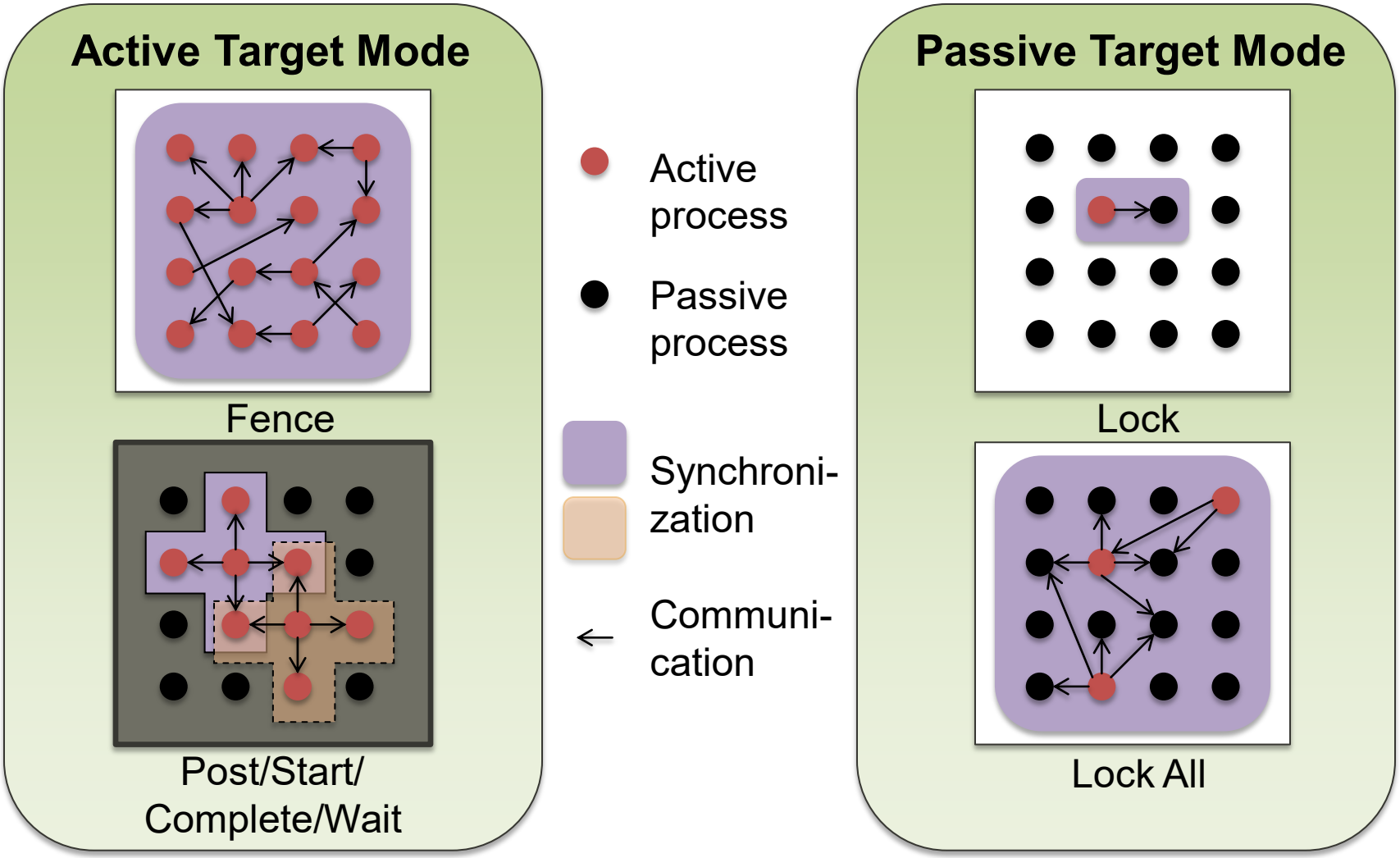
MPI-3.0 RMA Synchronization Overview



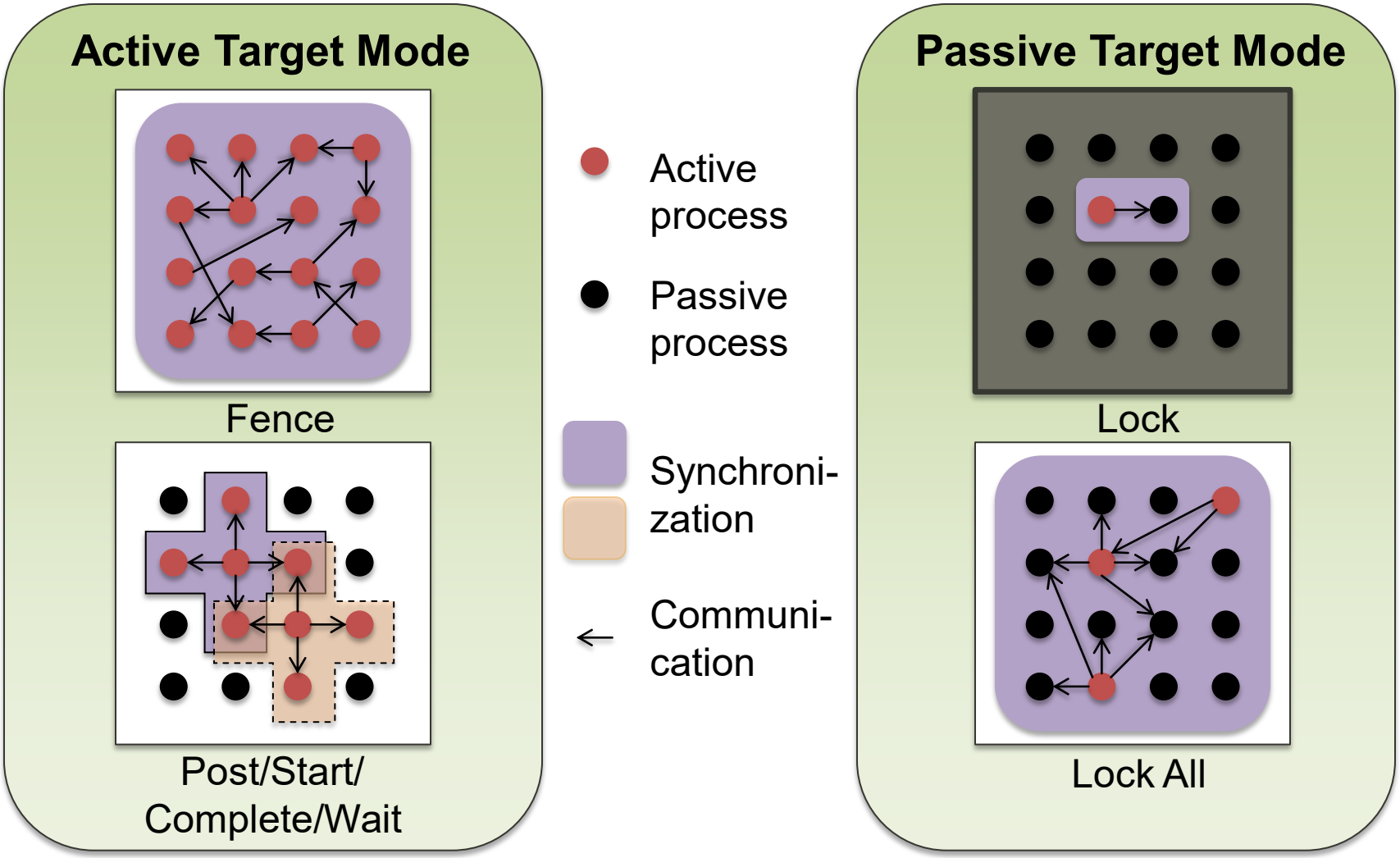
MPI-3.0 RMA Synchronization Overview



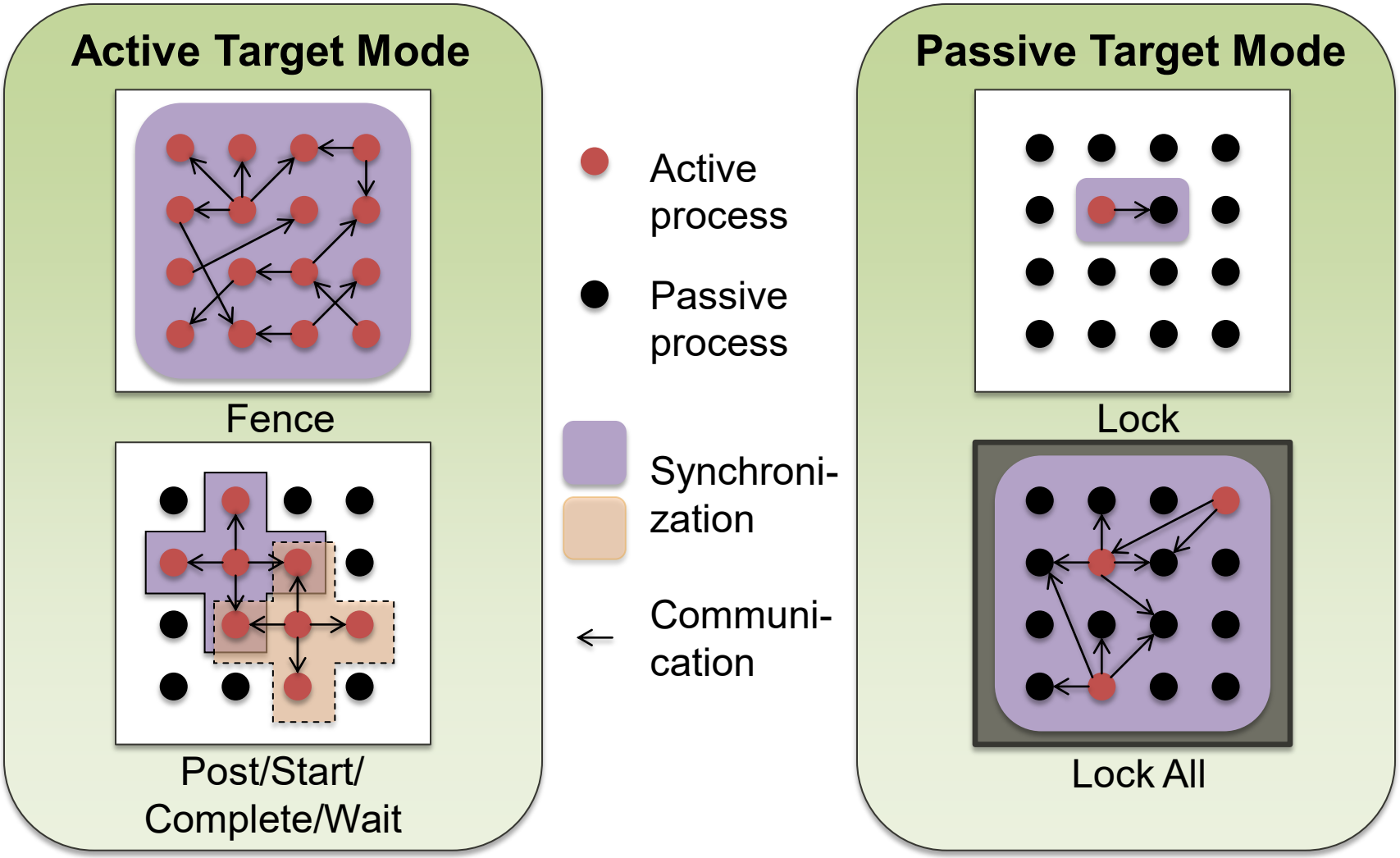
MPI-3.0 RMA Synchronization Overview



MPI-3.0 RMA Synchronization Overview



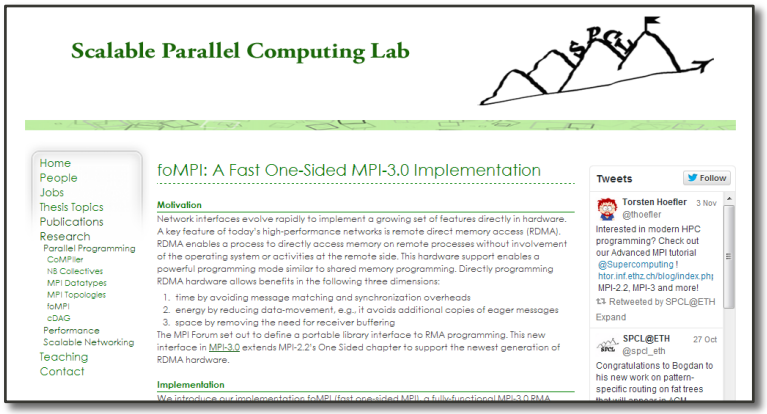
MPI-3.0 RMA Synchronization Overview

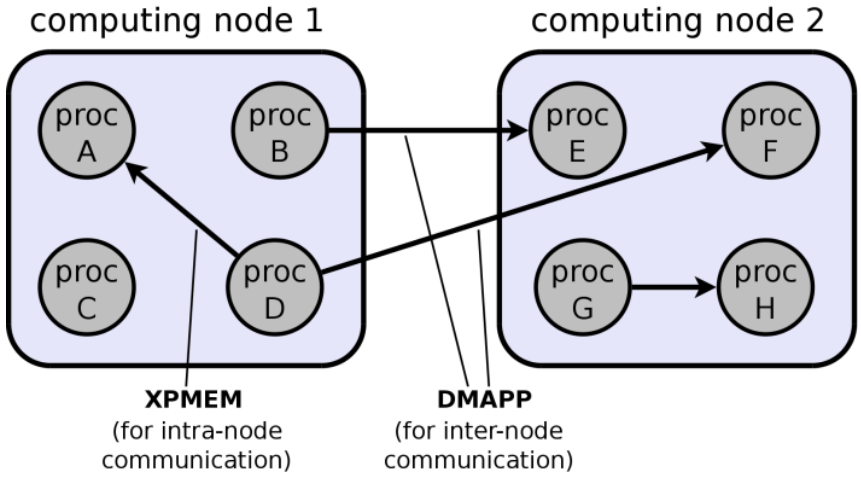


Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization

- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM, a portable Linux kernel module



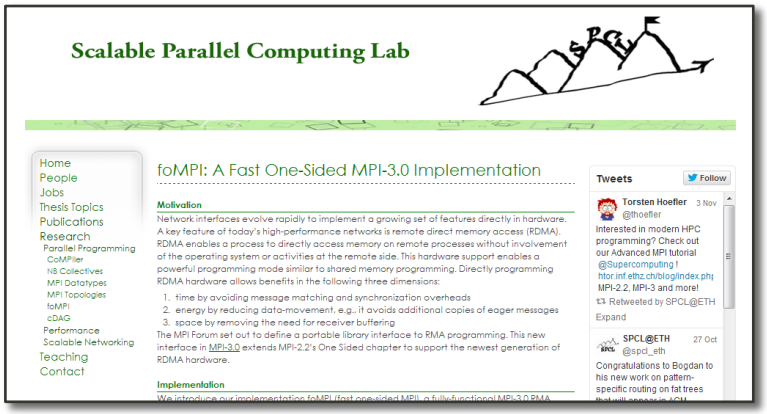


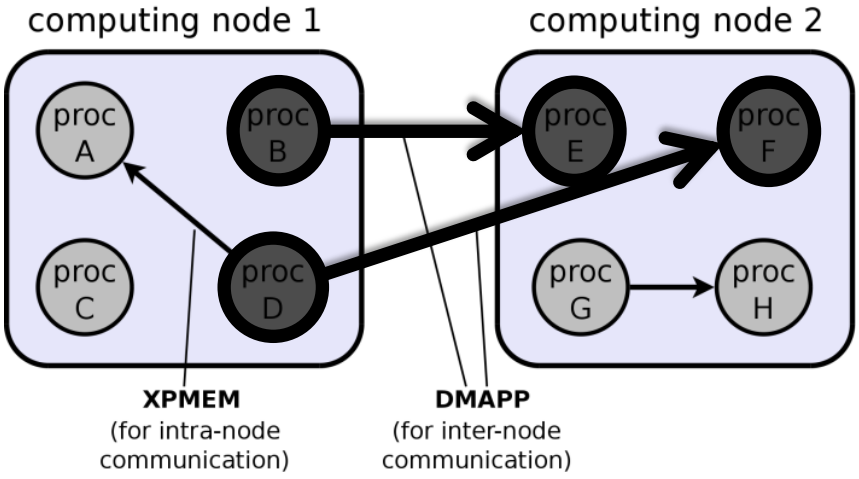
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization

- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM, a portable Linux kernel module



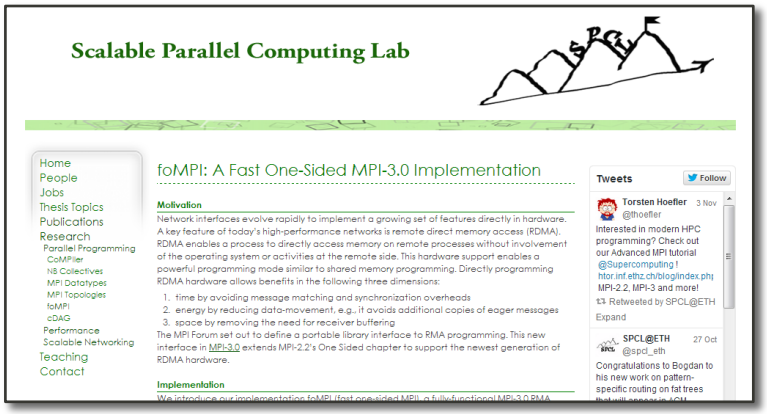


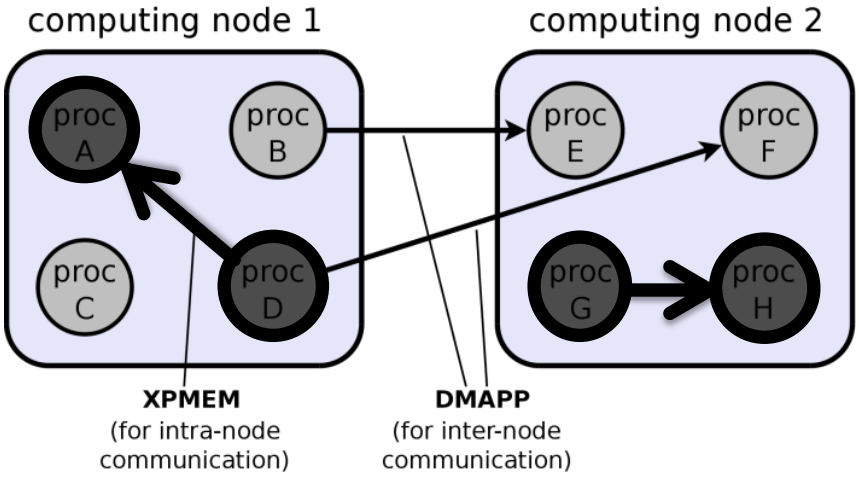
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

Scalable Protocols & Reference Implementation

- Scalable & generic protocols
  - Can be used on any RDMA network (e.g., OFED/IB)
  - Window creation, communication and synchronization

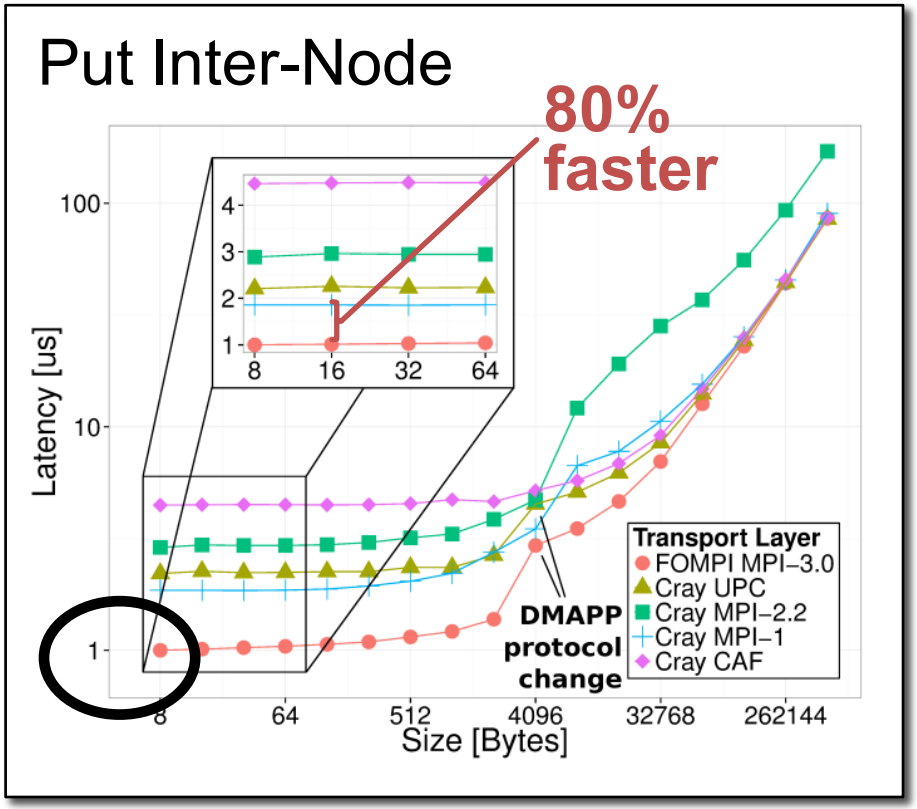
- foMPI, a fully functional MPI-3 RMA implementation
  - DMAPP: lowest-level networking API for Cray Gemini/Aries systems
  - XPMEM: a portable Linux kernel module

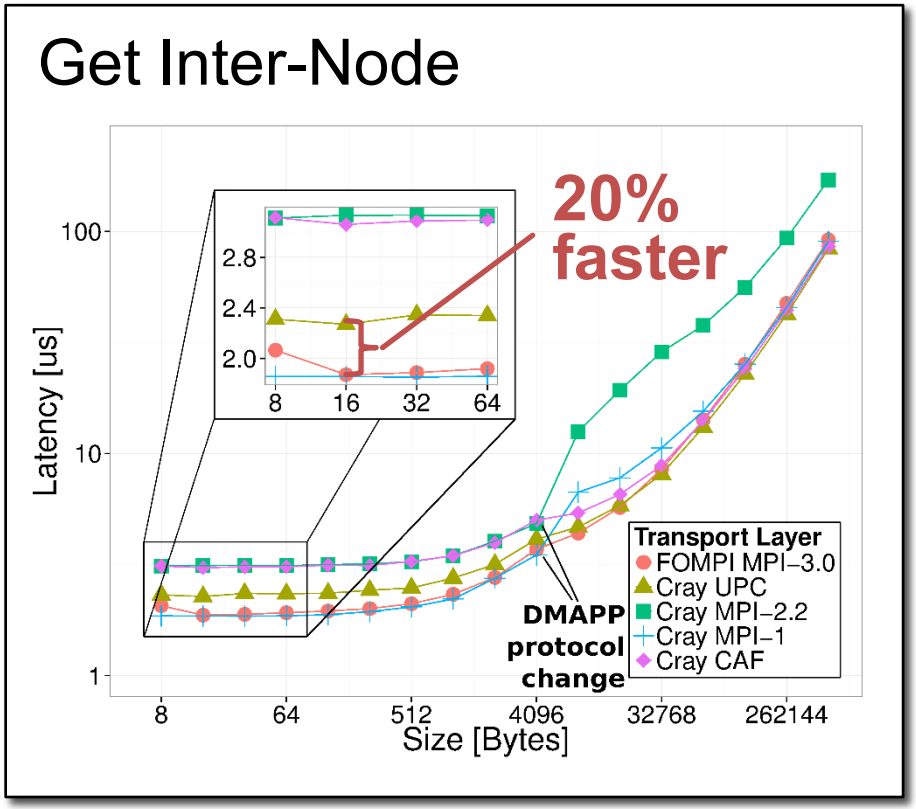


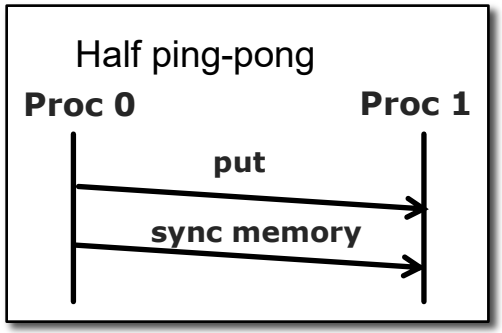


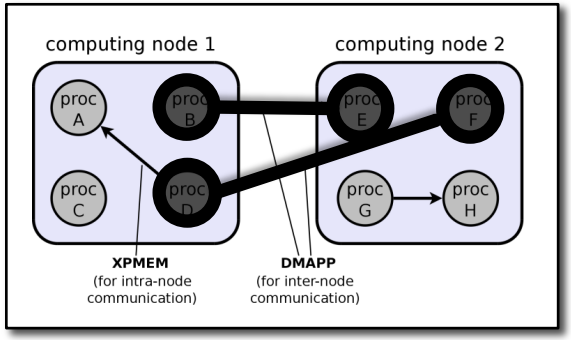
http://spcl.inf.ethz.ch/Research/Parallel_Programming/foMPI

PERFORMANCE INTER-NODE: LATENCY



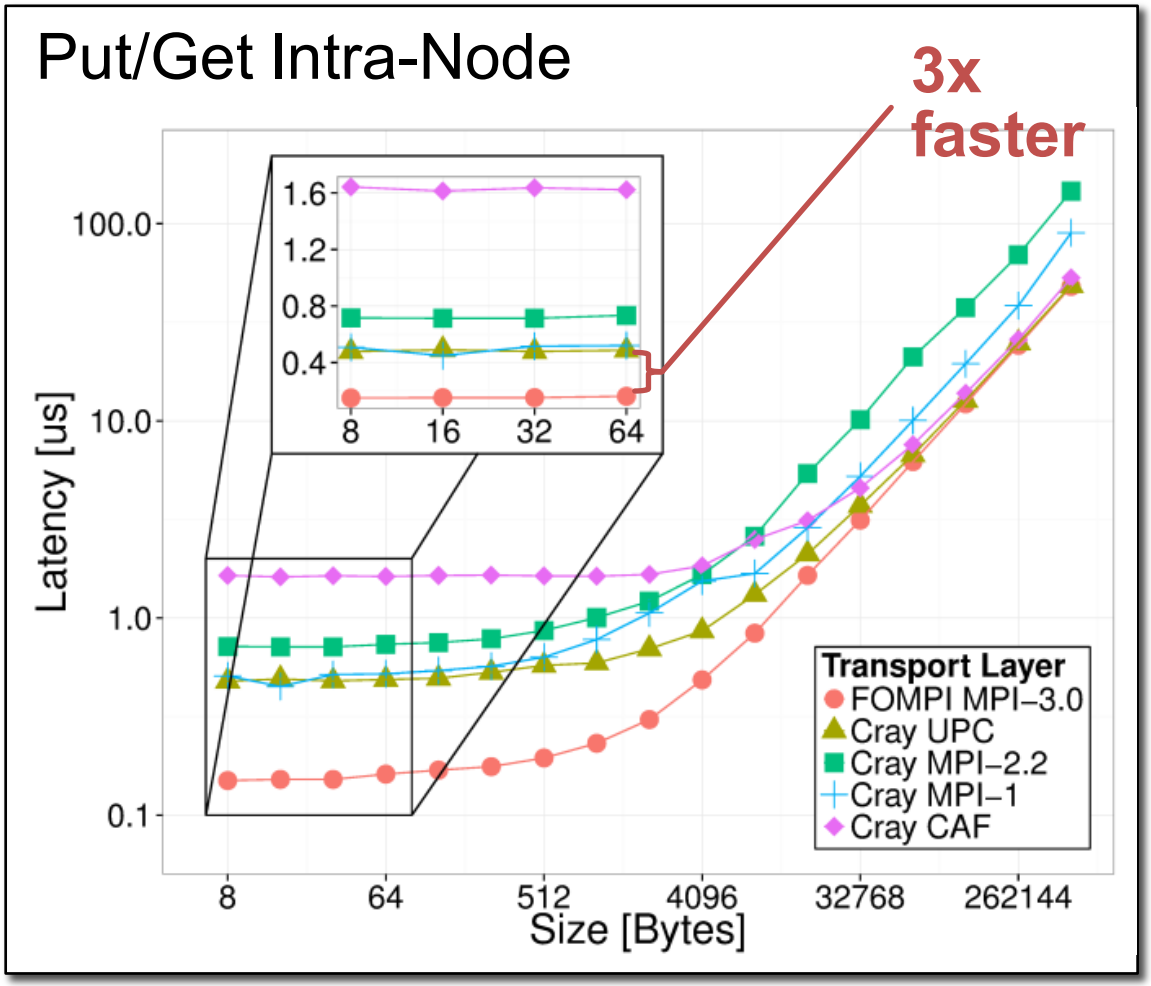


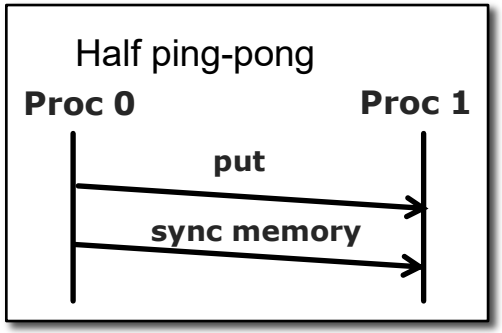


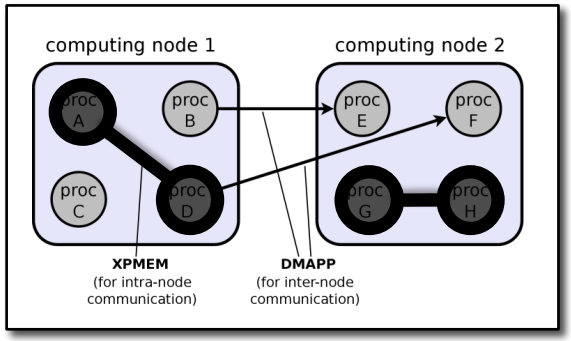


Gerstenberger, Besta, TH: „ Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided“, SC13

PERFORMANCE INTRA-NODE: LATENCY

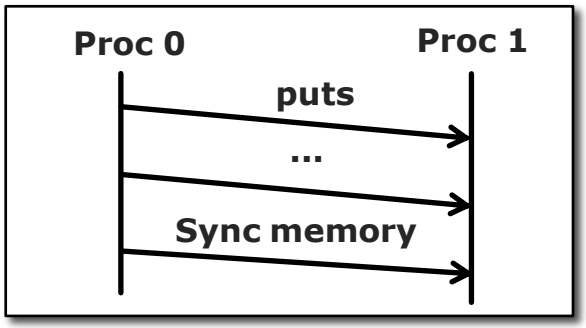




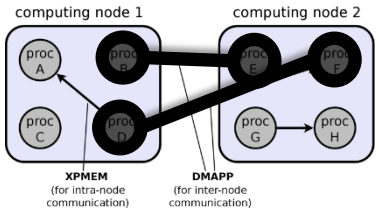


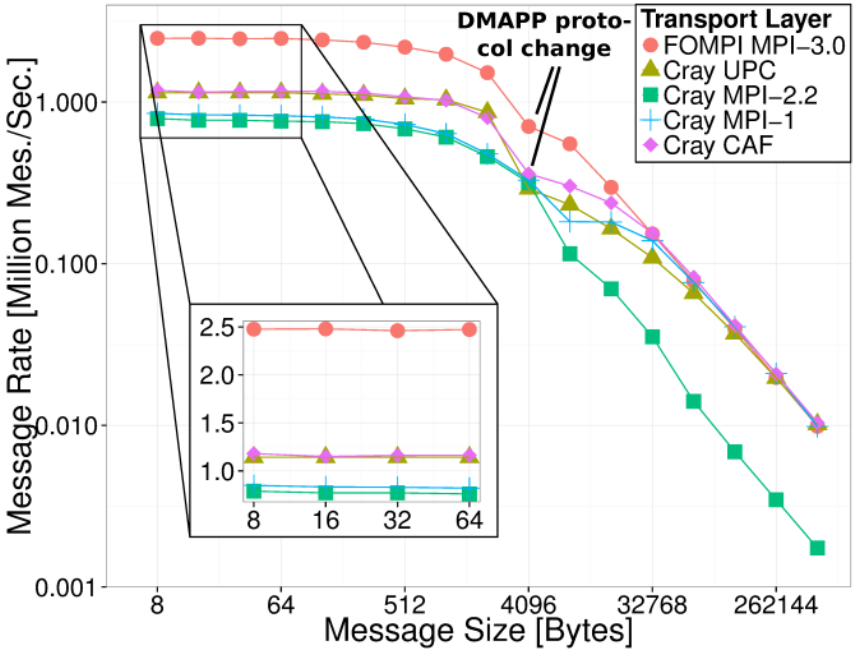
Gerstenberger, Besta, TH: „ Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided“, SC13

PERFORMANCE: MESSAGE RATE

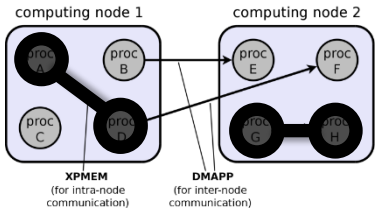


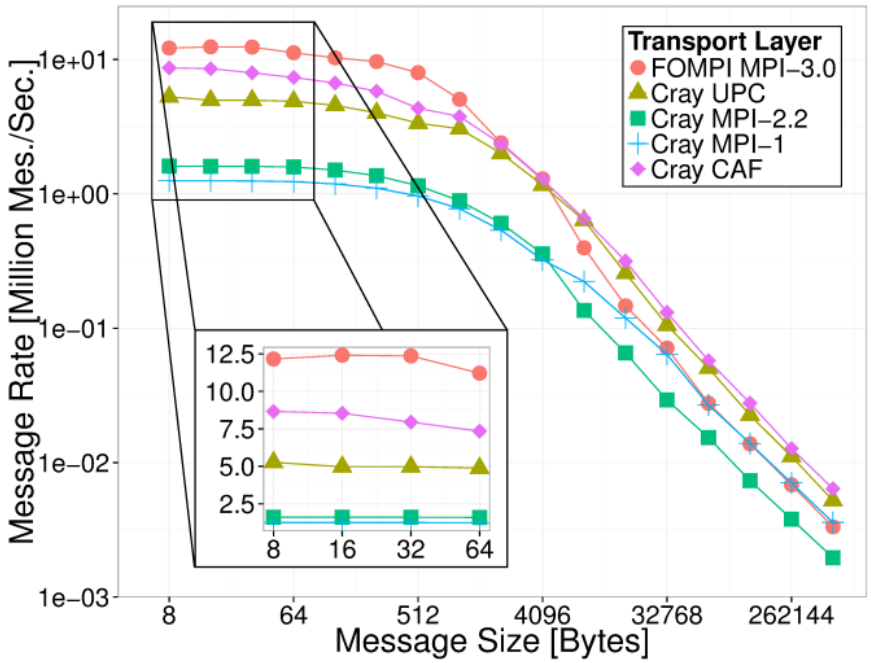
Inter-Node





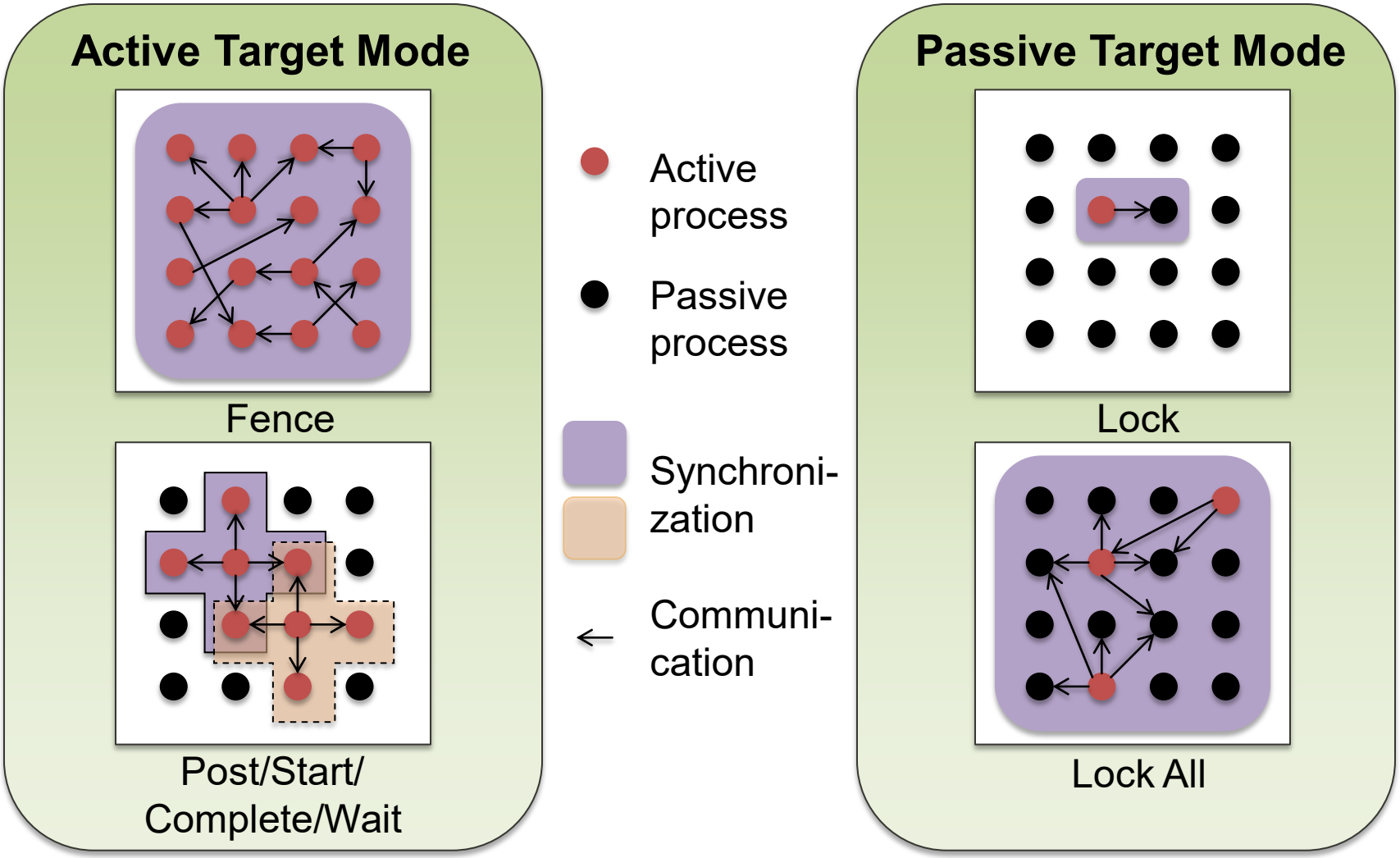
Intra-Node





Gerstenberger, Besta, TH: „ Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided“, SC13

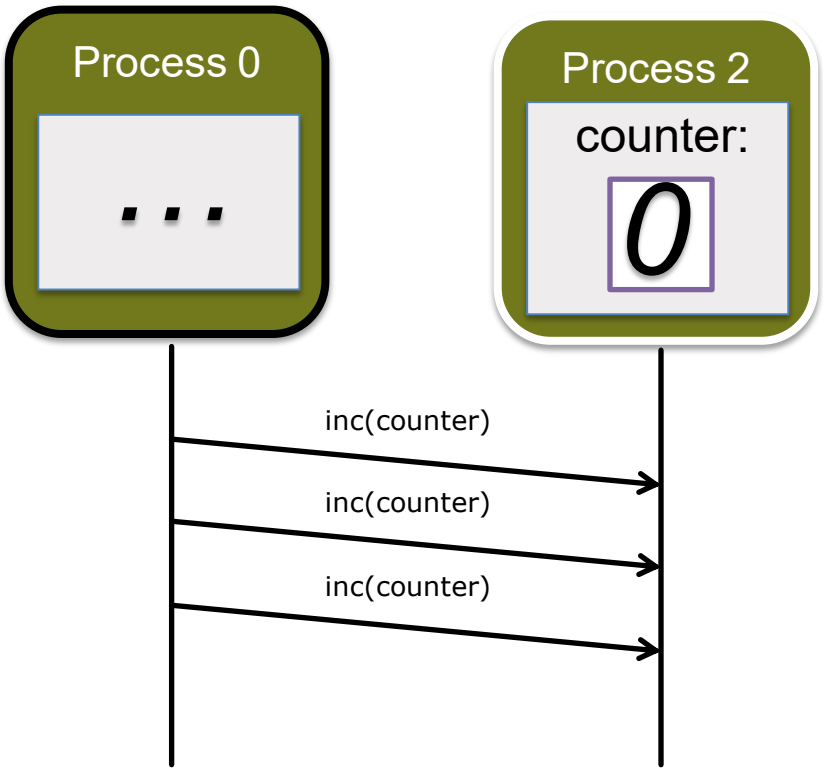
PART 3: SYNCHRONIZATION



FLUSH SYNCHRONIZATION

Time bound	$\mathcal{O}(1)$
Memory bound	$\mathcal{O}(1)$

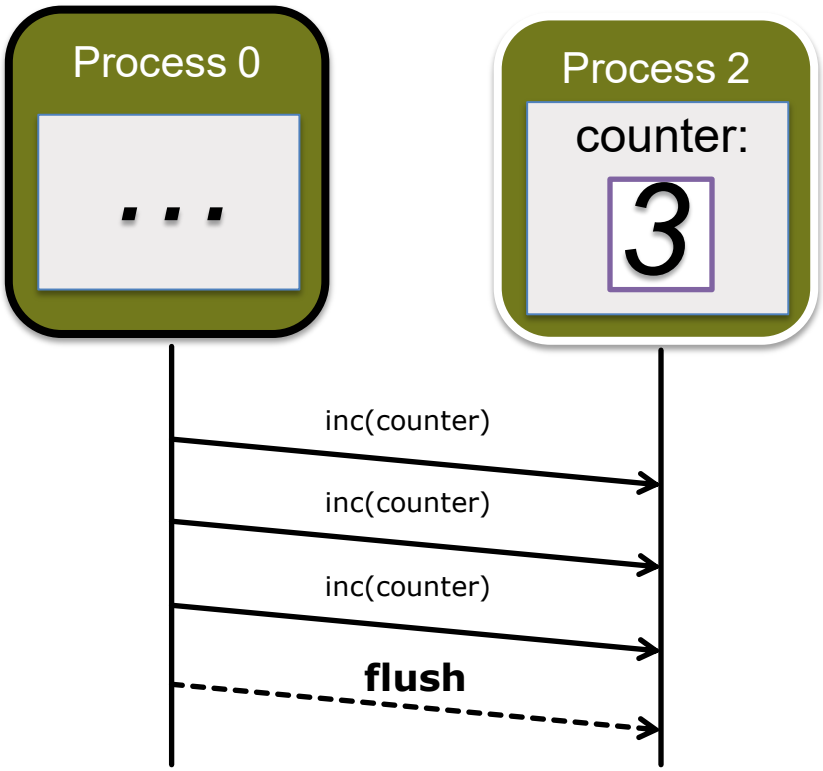
- Guarantees remote completion
- Issues a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only **78 x86** CPU instructions to the critical path



FLUSH SYNCHRONIZATION

Time bound	$\mathcal{O}(1)$
Memory bound	$\mathcal{O}(1)$

- Guarantees remote completion
- Issues a remote bulk synchronization and an x86 `mfence`
- One of the most performance critical functions, we add only **78 x86** CPU instructions to the critical path

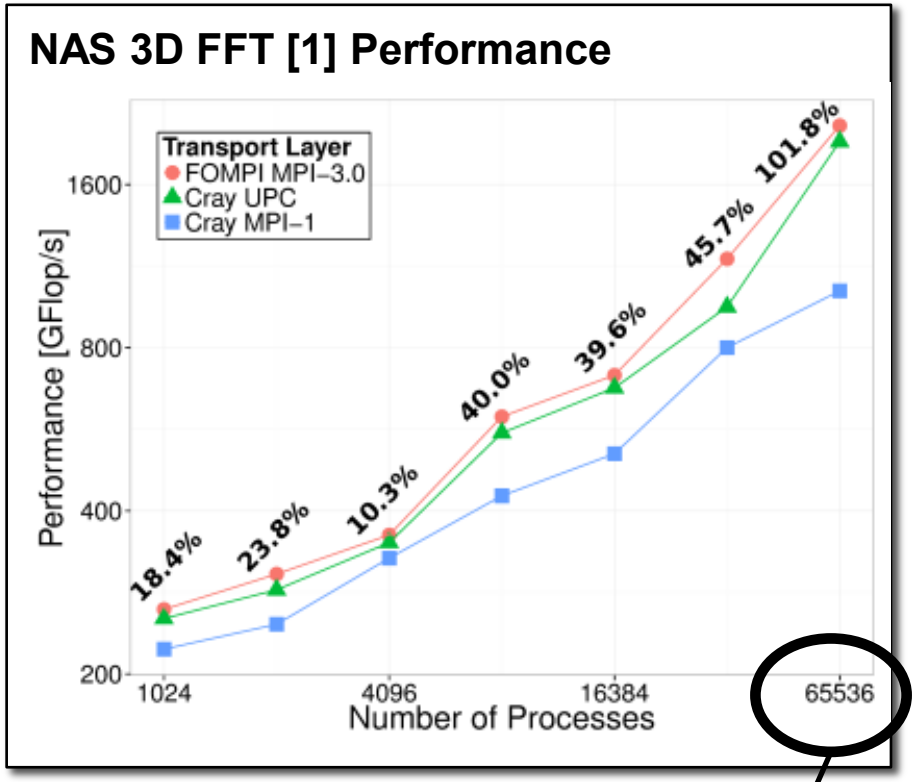


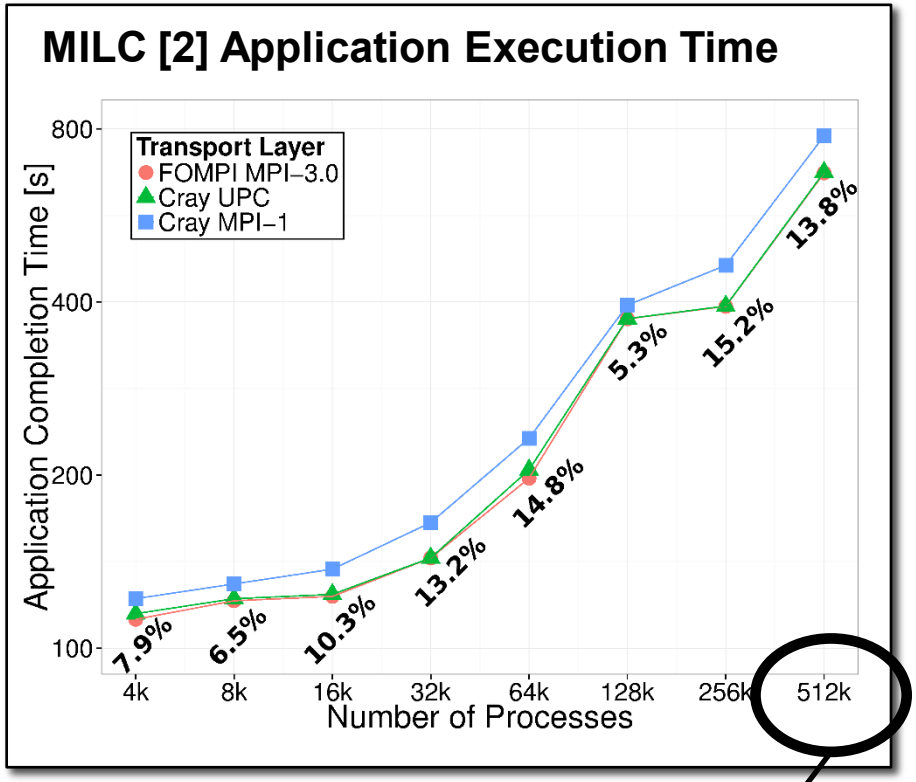
PERFORMANCE

- Evaluation on Blue Waters System
  - 22,640 computing Cray XE6 nodes
  - 724,480 schedulable cores
- All microbenchmarks
- 4 applications
- One nearly full-scale run ☺

PERFORMANCE: APPLICATIONS

Annotations represent performance gain of foMPI over Cray MPI-1.





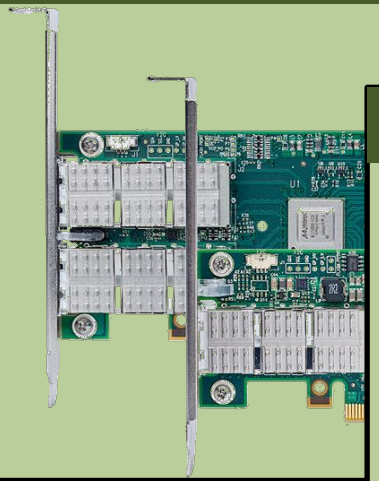
[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap. IPDPS'09
[2] Shan et al. Accelerating applications at scale using one-sided communication. PGAS'12

INTERMEZZO CONCLUSIONS

Requirements for a new low-level programming model

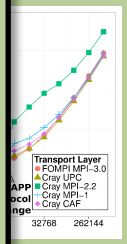
1. Messaging needs 100% hardware support (offload) – it's simple after all!
2. Minimal overheads (**tiny**) layer between user and hardware
3. Offer a simple abstract performance model (e.g., LogGP)

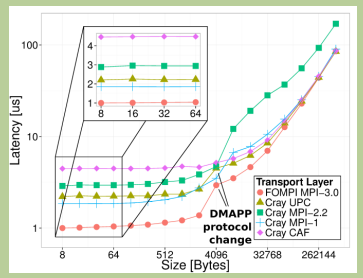
1) Messaging needs hardware support



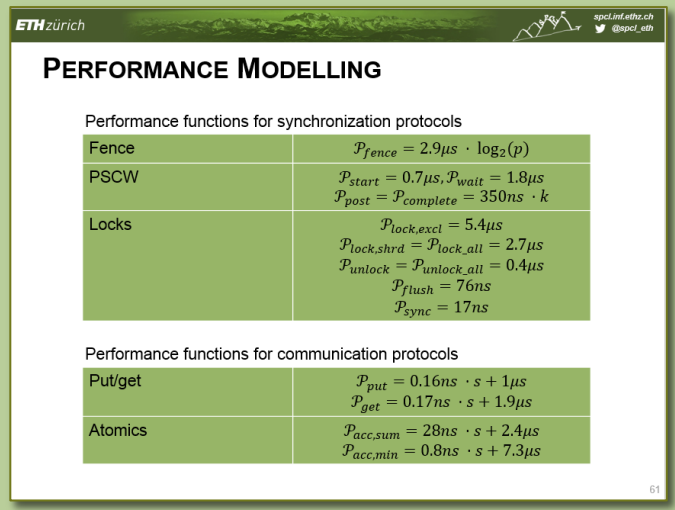
2) Minimal overheads

1. Only 78 x86 CPU instructions
optimal collective
ication





3) Simple performance model



ETH zürich — spcl.inf.ethz.ch @spcl_eth

PERFORMANCE MODELLING

Performance functions for synchronization protocols

Fence	$\mathcal{P}_{fence} = 2.9\mu s \cdot \log_2(p)$
PSCW	$\mathcal{P}_{start} = 0.7\mu s, \mathcal{P}_{wait} = 1.8\mu s$ $\mathcal{P}_{post} = \mathcal{P}_{complete} = 350ns \cdot k$
Locks	$\mathcal{P}_{lock,excl} = 5.4\mu s$ $\mathcal{P}_{lock,shrd} = \mathcal{P}_{lock\_all} = 2.7\mu s$ $\mathcal{P}_{unlock} = \mathcal{P}_{unlock\_all} = 0.4\mu s$ $\mathcal{P}_{flush} = 76ns$ $\mathcal{P}_{sync} = 17ns$

Performance functions for communication protocols

Put/get	$\mathcal{P}_{put} = 0.16ns \cdot s + 1\mu s$ $\mathcal{P}_{get} = 0.17ns \cdot s + 1.9\mu s$
Atomics	$\mathcal{P}_{acc,sum} = 28ns \cdot s + 2.4\mu s$ $\mathcal{P}_{acc,min} = 0.8ns \cdot s + 7.3\mu s$

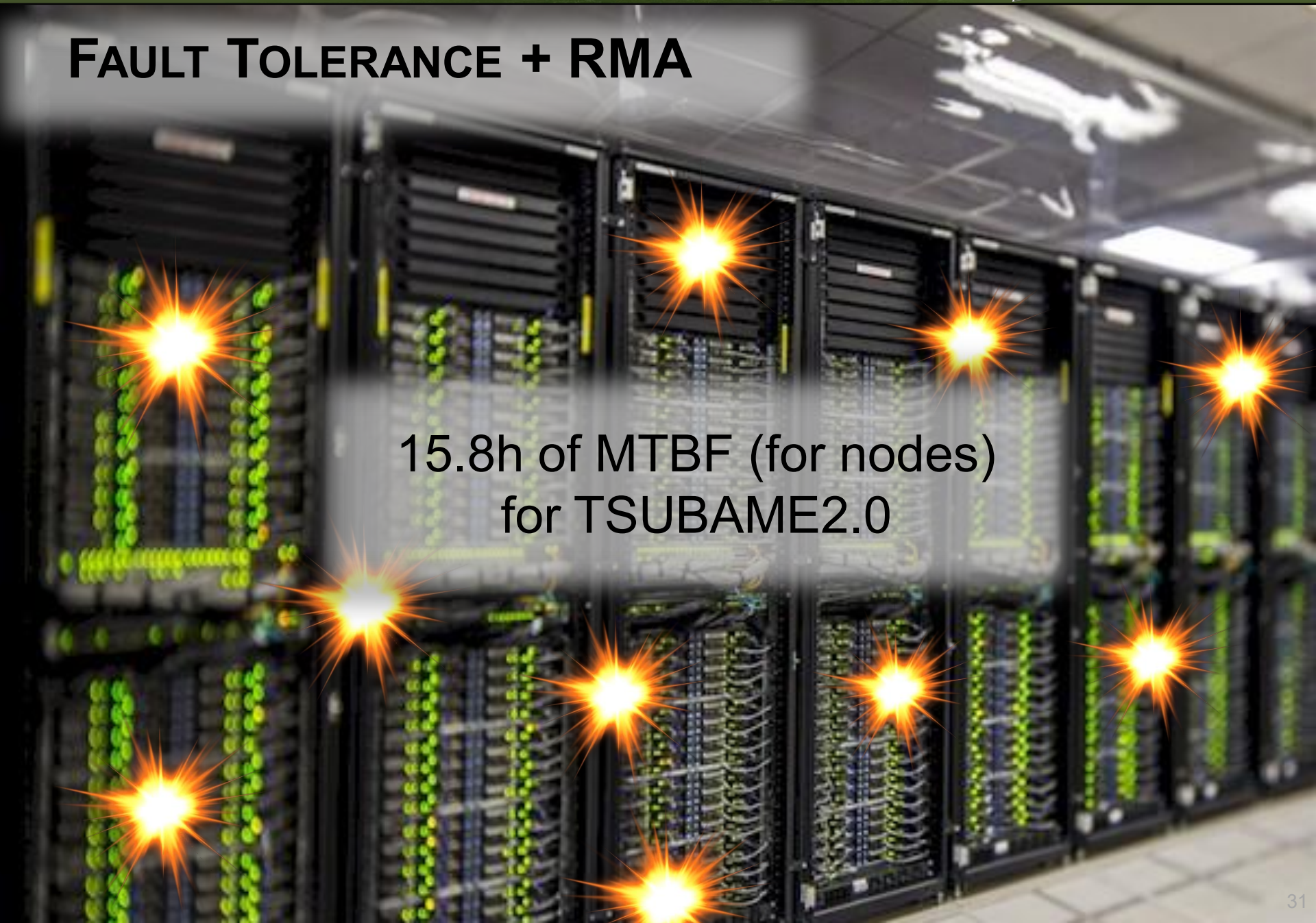
61

Gerstenberger, Besta, TH: „ Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One Sided“, SC13

WHAT ARE THE CHALLENGES?

- RMA is supported by many HPC libraries and languages

FAULT TOLERANCE + RMA



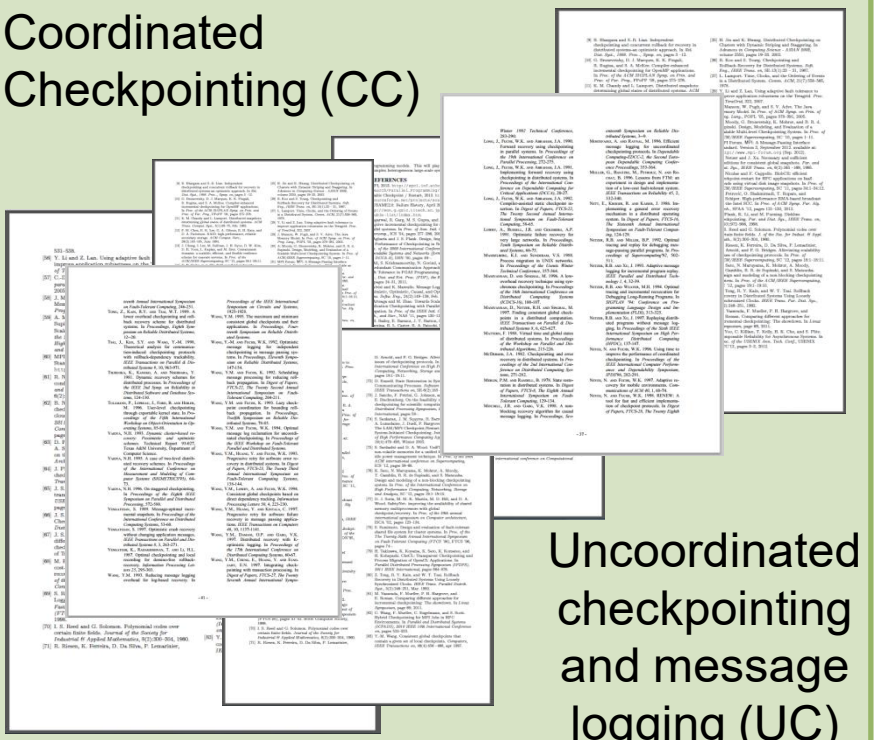
15.8h of MTBF (for nodes)
for TSUBAME2.0

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA
  - Most mechanisms from MP are not applicable
  - Applicable schemes (e.g., simple checkpointing) may be highly inefficient

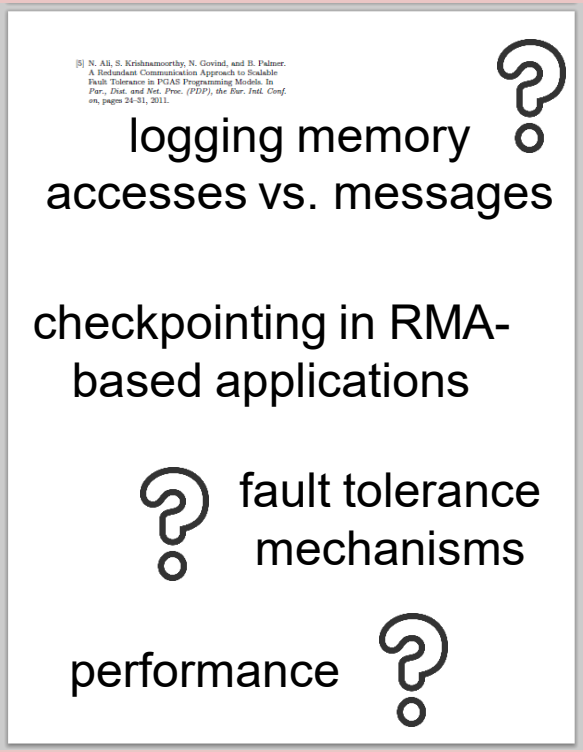
Message Passing

Coordinated Checkpointing (CC)



Uncoordinated checkpointing and message logging (UC)

RMA



logging memory accesses vs. messages

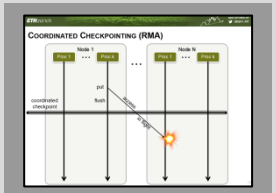
checkpointing in RMA-based applications

fault tolerance mechanisms

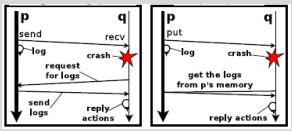
performance

OVERVIEW

MP vs. RMA



Coord. checkpt. (CC)



Uncoord. checkpt. and logging (UC)

Generic model

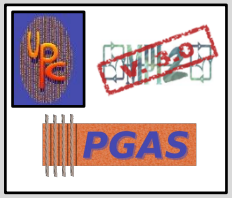
$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, ... \rangle$, $\xrightarrow{hb}$ $\xrightarrow{co}$

$a \xrightarrow{cohb} \equiv a \xrightarrow{co} b \wedge a \xrightarrow{hb} b$

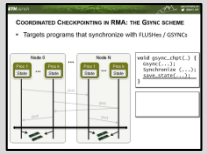
$\xrightarrow{po}$ $\xrightarrow{so}$

$\text{PUT}(p \rightrightarrows q)$

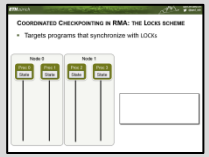
$\text{GET}(p \leftrightarrows q)$



CC in RMA

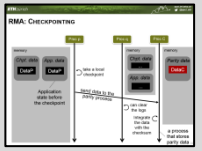


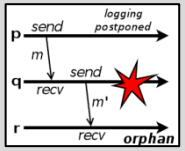
„Gsync" Scheme



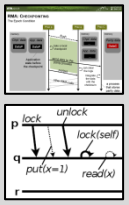
„Locks" Scheme

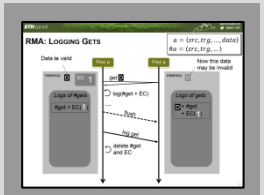
UC in RMA





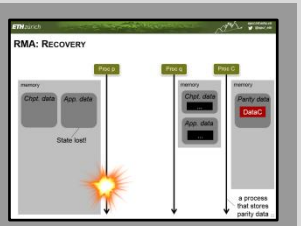
Checkpointing





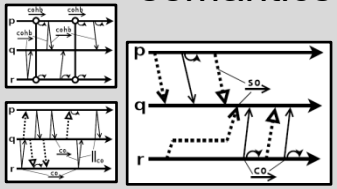
Logging

Recovery in RMA



Basic scheme

Extended RMA semantics



Theory

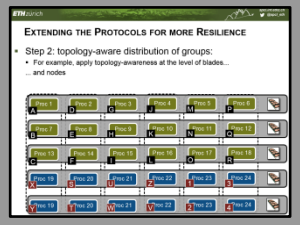
THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

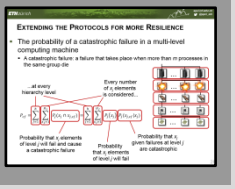
$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions



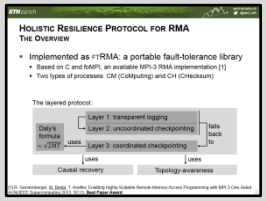
Distribution of processes

Decreasing failure prob.



$$\frac{D_j \cdot \binom{|G|}{2} \cdot \binom{H_j - 2}{x_j - 2}}{\binom{H_j}{x_j}}.$$

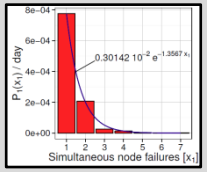
Holistic fault-tolerance library

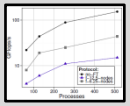


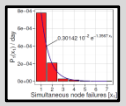
Design and optimizations

Checkpoints on demand

Optimum CC intervals



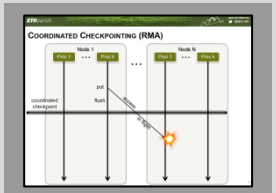




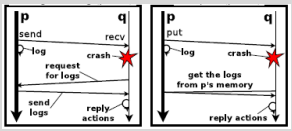
Performance

OVERVIEW

MP vs. RMA



Coord. checkpt. (CC)



Uncoord. checkpt. and logging (UC)

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \ldots \rangle$, $\xrightarrow{hb}$, $\xrightarrow{co}$

$a \xrightarrow{cohb} \equiv a \xrightarrow{co} b \wedge a \xrightarrow{hb} b$

$\xrightarrow{po}$ $\xrightarrow{so}$

$\mathrm{PUT}(p \rightrightarrows q)$

$\mathrm{GET}(p \leftrightarrows q)$

CC in RMA

„Gsync" Scheme

„Locks" Scheme

UC in RMA

Checkpointing

Logging

Recovery in RMA

Basic scheme

Extended RMA semantics

Theory

THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Decreasing failure prob.

Distribution of processes

$$\frac{D_j \cdot \binom{|C|}{2} \cdot \binom{H_j - 2}{x_j - 2}}{\binom{H_j}{x_j}}$$

Holistic fault-tolerance library

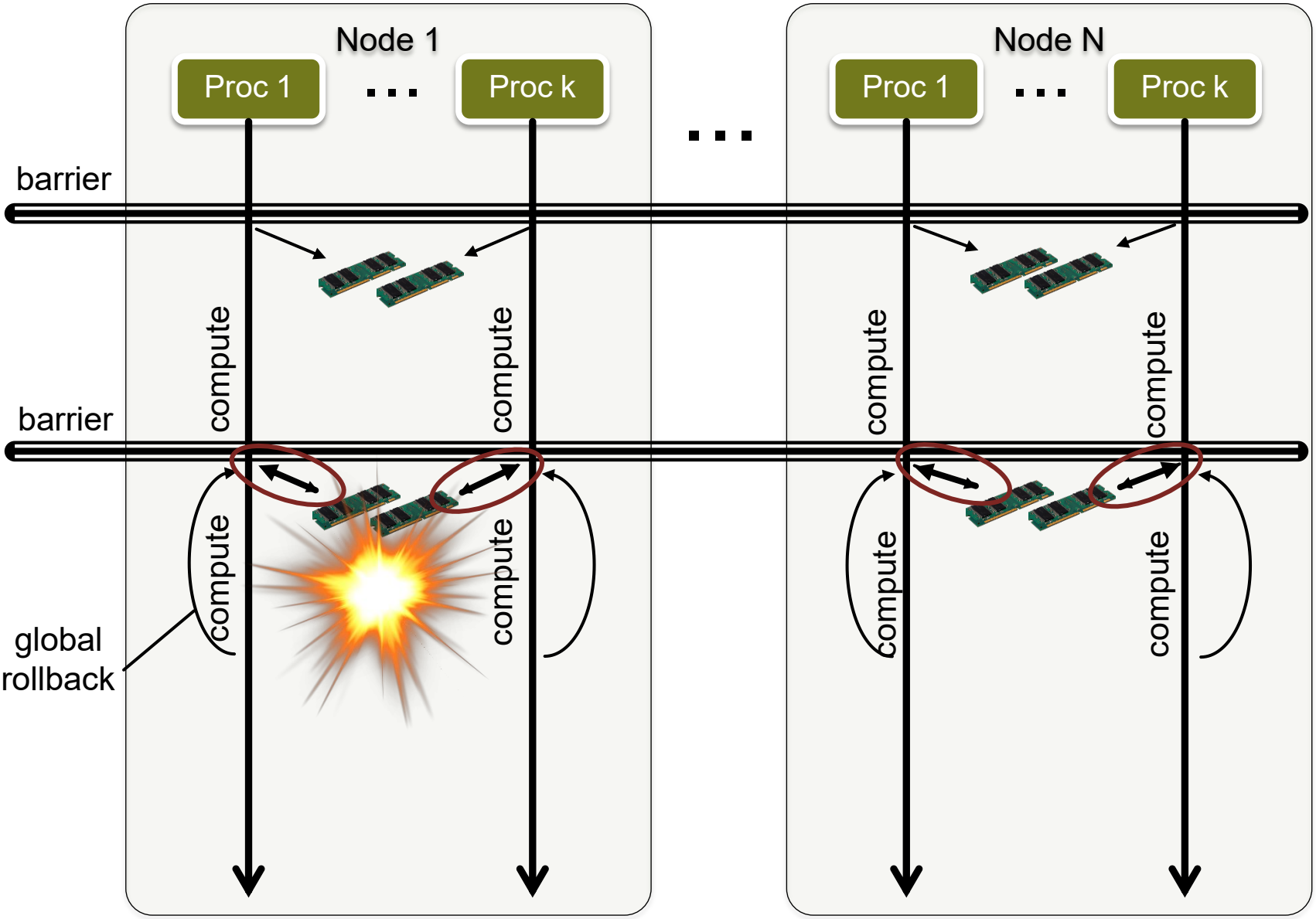
Design and optimizations

Checkpoints on demand

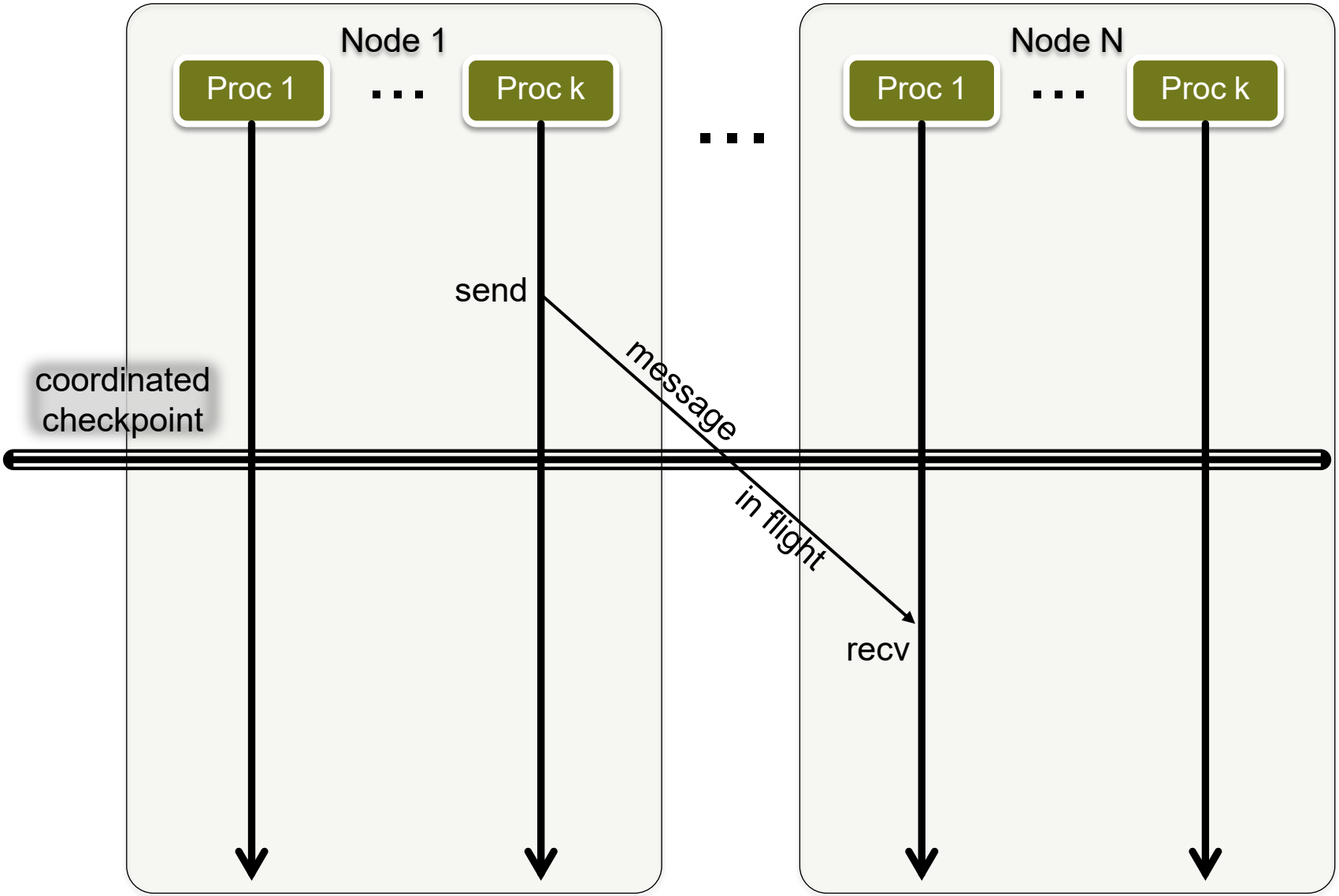
Optimum CC intervals

Performance

COORDINATED CHECKPOINTING (MP)

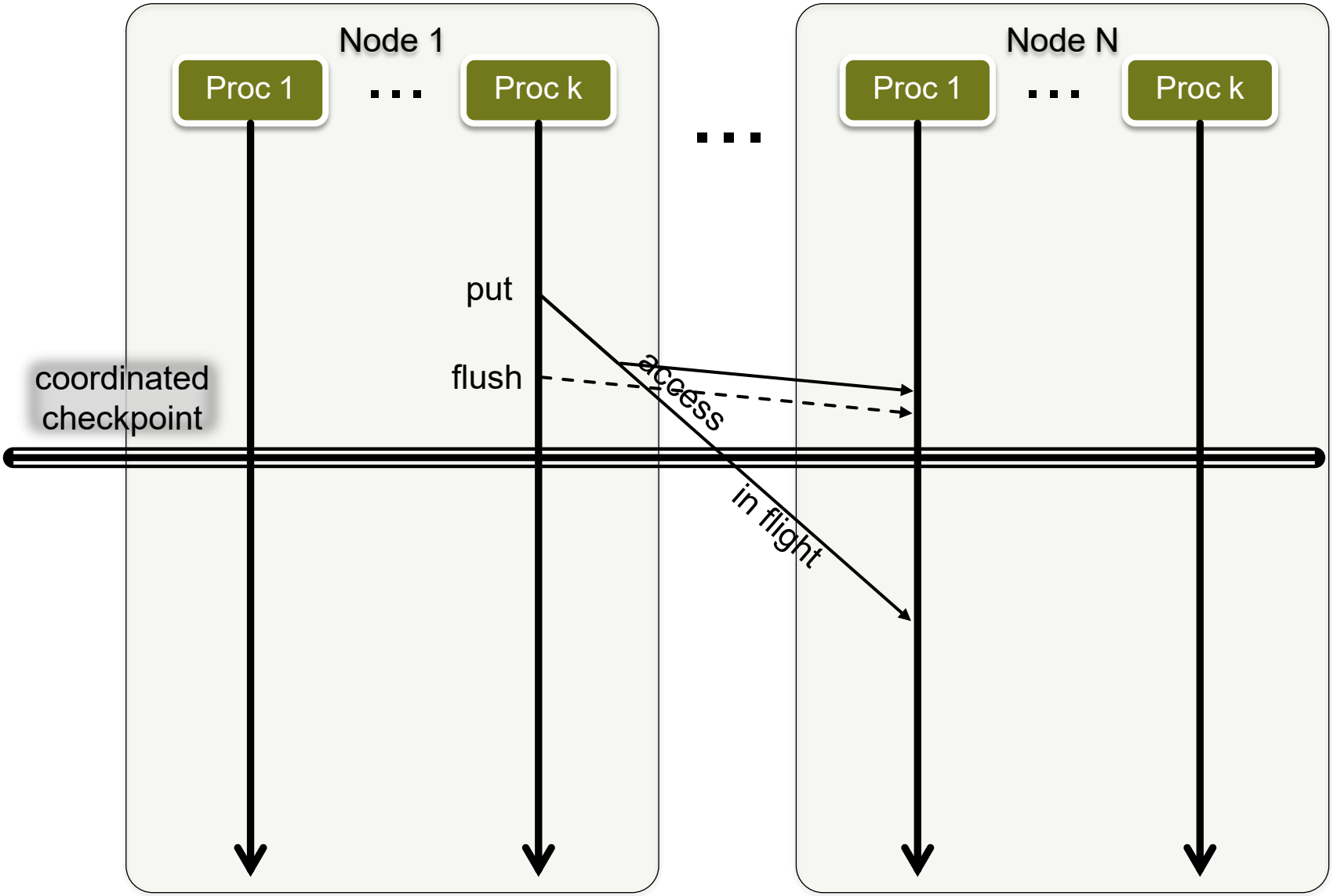


COORDINATED CHECKPOINTING (MP)



[1] Elnozahy et al. A survey of rollback-recovery protocols in message-passing systems. Journal ACM Computing Surveys, 2002

COORDINATED CHECKPOINTING (RMA)



Besta, TH: „ Fault Tolerance for Remote Memory Access Programming Models“, HPDC‘14

CONTRIBUTIONS

MP vs. RMA

Coord. checkpt. (CC)

Uncoord. checkpt. and logging (UC)

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \ldots \rangle, \quad \xrightarrow{hb} \quad \xrightarrow{co}$$

$$a \xrightarrow{cohb} \equiv a \xrightarrow{co} b \wedge a \xrightarrow{hb} b$$

$$\xrightarrow{po} \quad \xrightarrow{so}$$

$$\mathrm{PUT}(p \rightrightarrows q)$$
$$\mathrm{GET}(p \leftrightarrows q)$$

CC in RMA

„Gsync" Scheme

„Locks" Scheme

UC in RMA

Checkpointing

Logging

Recovery in RMA

Basic scheme

Extended RMA semantics

Theory

THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

Decreasing failure prob.

Distribution of processes

$$\frac{D_j \cdot \binom{|G|}{2} \cdot \binom{H_j - 2}{x_j - 2}}{\binom{H_j}{x_j}}$$

Holistic fault-tolerance library

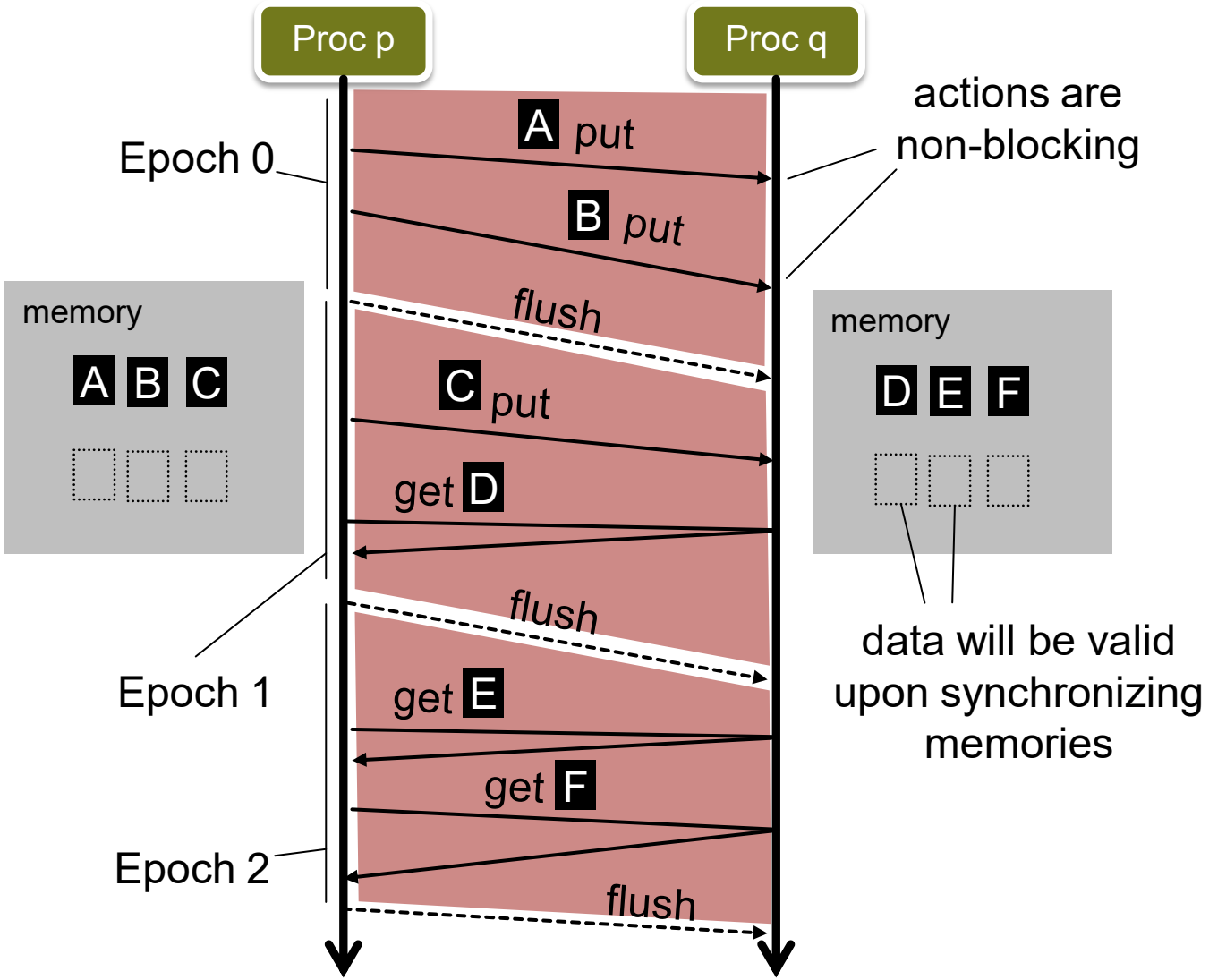
Design and optimizations

Checkpoints on demand

Optimum CC intervals

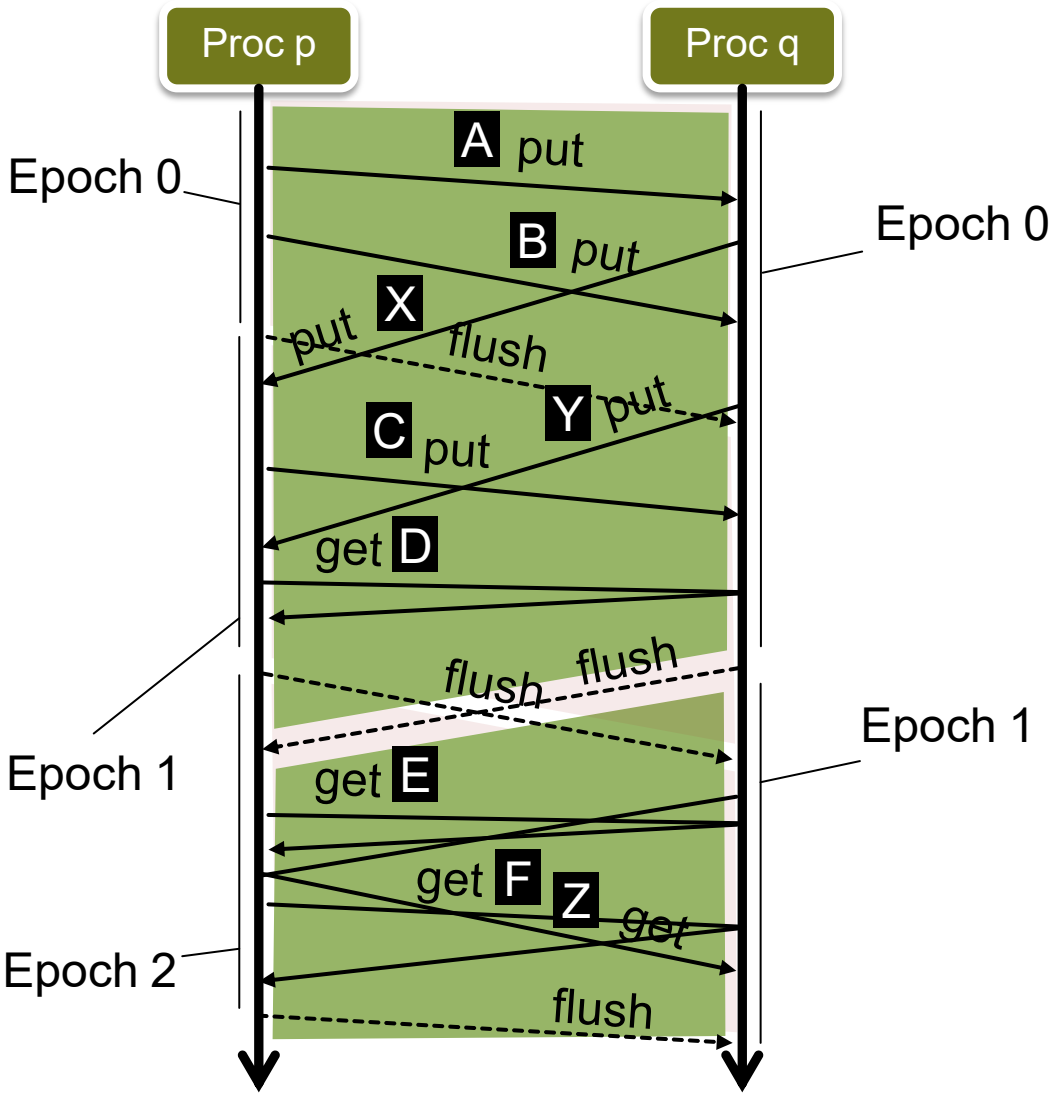
Performance

RMA: EPOCHS



TH et al.: Remote memory access programming in MPI-3.0“, submitted

RMA: EPOCHS

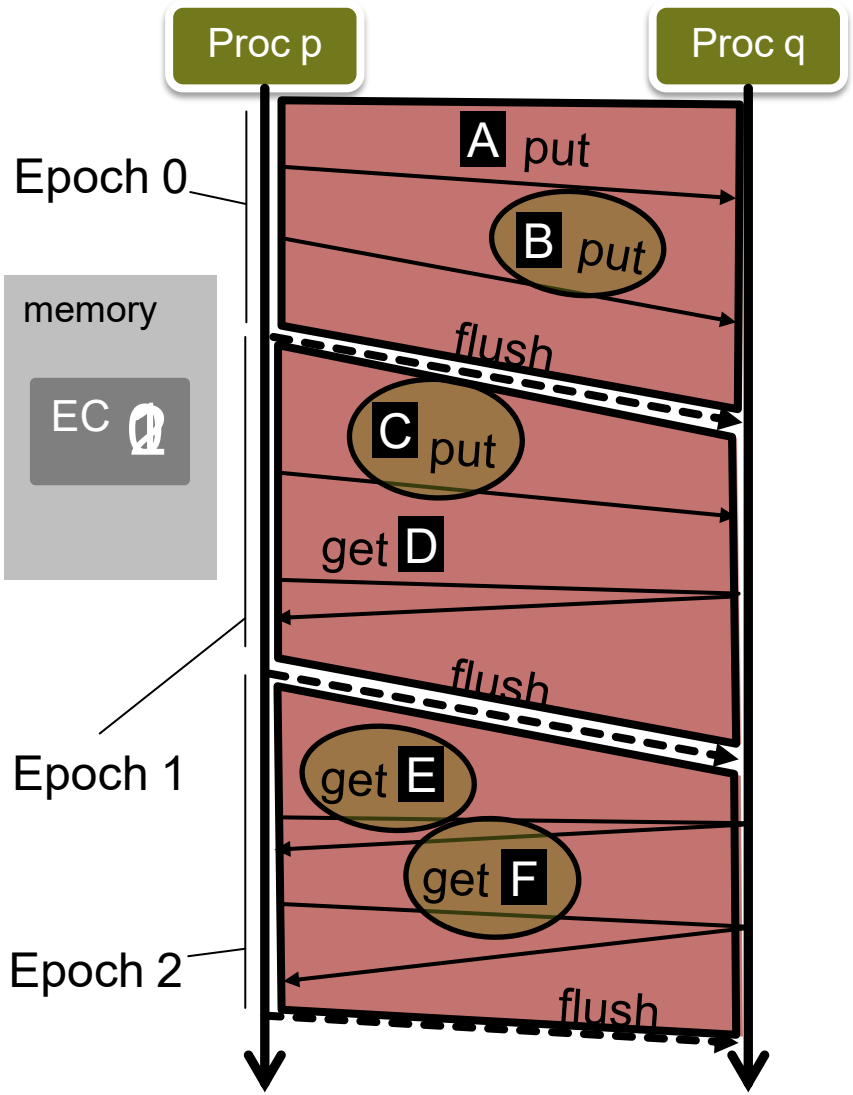


TH et al.: Remote memory access programming in MPI-3.0“, submitted

RMA: The Consistency Order $\xrightarrow{co}$

For recovery, a process has to replay actions in the correct $\xrightarrow{co}$ order!

For this, we use **epoch counters (EC)**



B put $\xrightarrow{co}$ C put

get E $\|_{co}$ get F

TH et al.: Remote memory access programming in MPI-3.0“, submitted

CONTRIBUTIONS

MP vs. RMA

Coord. checkpt. (CC)

Uncoord. checkpt. and logging (UC)

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \ldots \rangle,$ $\xrightarrow{hb}$ $\xrightarrow{co}$

$a \xrightarrow{cohb} \equiv a \xrightarrow{co} b \wedge a \xrightarrow{hb} b$

$\xrightarrow{po}$ $\xrightarrow{so}$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftrightarrows q)$

CC in RMA

„Gsync" Scheme

„Locks" Scheme

UC in RMA

Checkpointing

Logging

Recovery in RMA

Basic scheme

Extended RMA semantics

Theory

THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Decreasing failure prob.

$$\frac{D_j \cdot \binom{|C|}{2} \cdot \binom{H_j - 2}{x_j - 2}}{\binom{H_j}{x_j}}.$$

Distribution of processes

Holistic fault-tolerance library

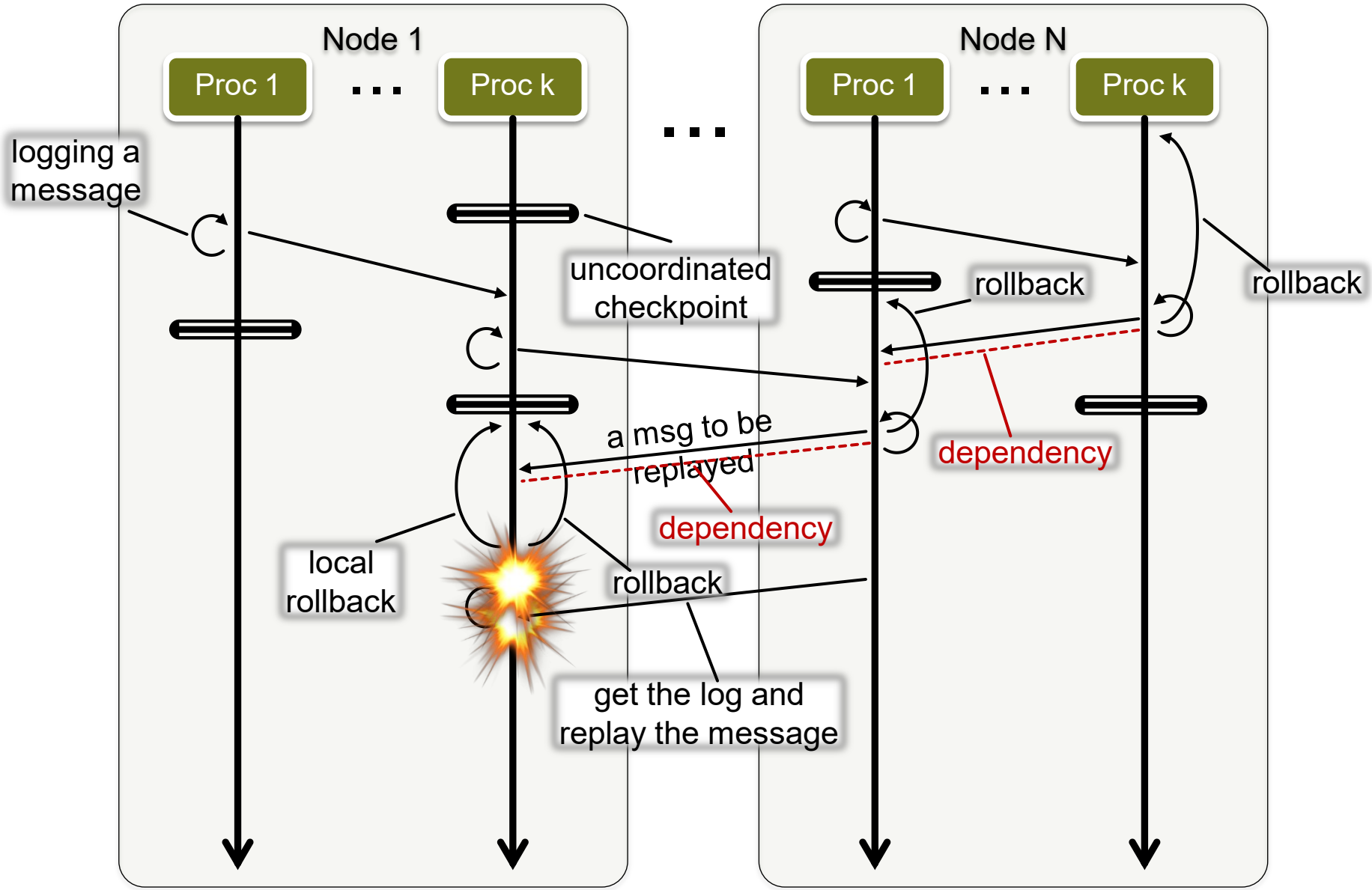
Design and optimizations

Checkpoints on demand

Optimum CC intervals

Performance

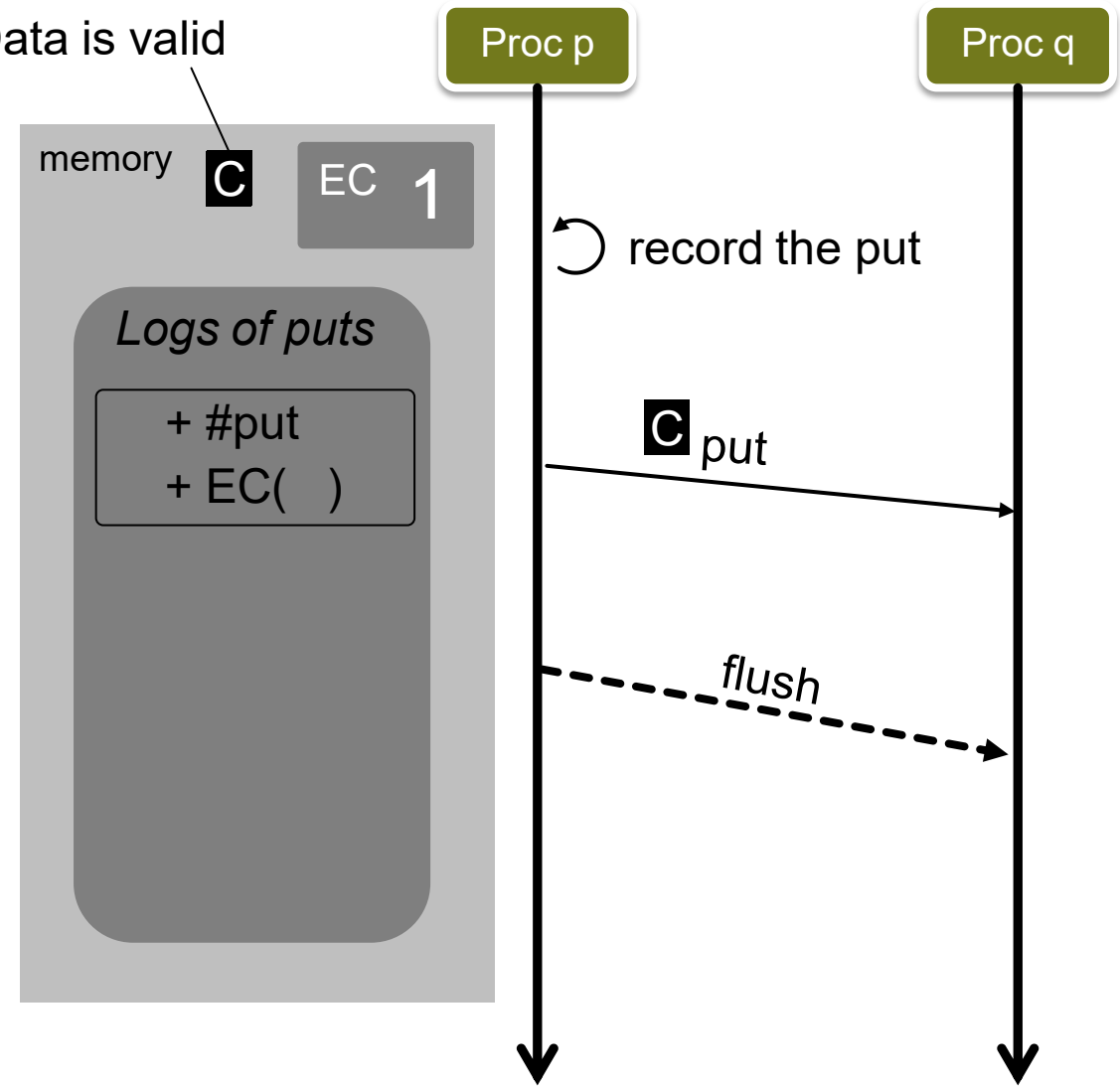
UNCOORDINATED CHECKPOINTING (MP)



Elnozahy, Alvisi, Wang, and Johnson. "A survey of rollback-recovery protocols in message-passing systems." Comput. Surv., 2002.

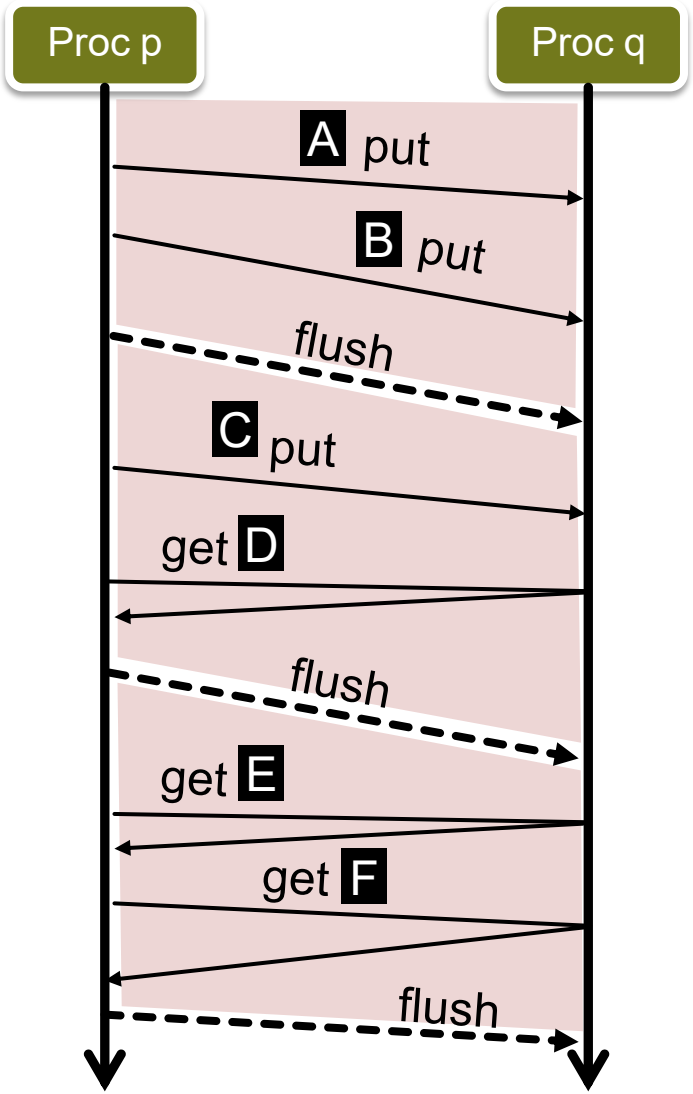
RMA: Logging Puts

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



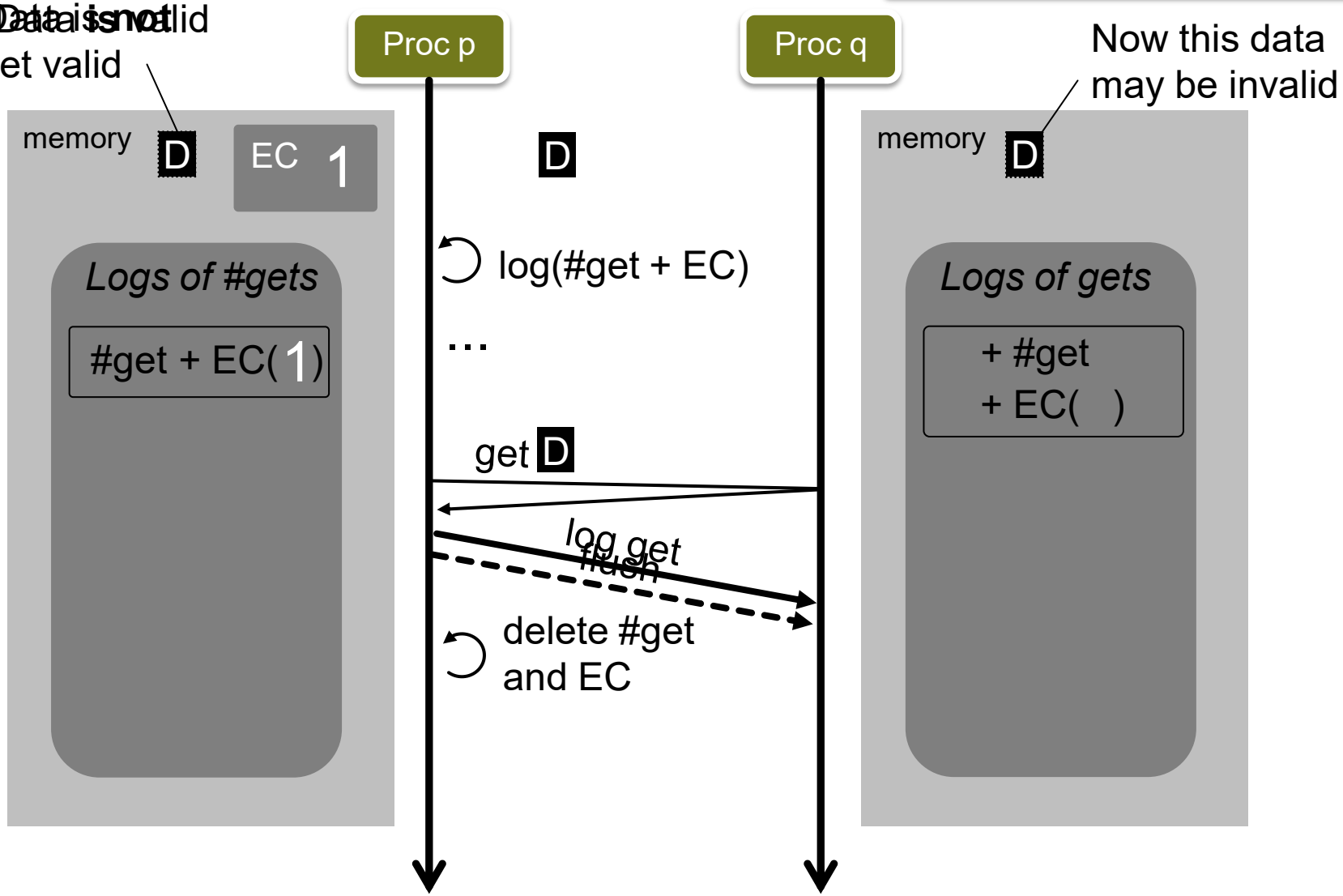
RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



RMA: Logging Gets

$$a = \langle src, trg, \ldots, data \rangle$$
$$\#a = \langle src, trg, \ldots \rangle$$



CONTRIBUTIONS

MP vs. RMA

Coord. checkpt. (CC)

Uncoord. checkpt. and logging (UC)

Generic model

$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \ldots \rangle,$ $\xrightarrow{hb}$ $\xrightarrow{co}$

$a \xrightarrow{cohb} \equiv a \xrightarrow{co} b \wedge a \xrightarrow{hb} b$

$\xrightarrow{po}$ $\xrightarrow{so}$

$\mathrm{PUT}(p \rightrightarrows q)$
$\mathrm{GET}(p \leftleftarrows q)$

CC in RMA

„Gsync" Scheme

„Locks" Scheme

UC in RMA

Checkpointing

Logging

Recovery in RMA

Basic scheme

Extended RMA semantics

Theory

THEOREM 4.2. *The recove rithm 2 preserves the* $\xrightarrow{cohb}$ *or ferred to as the* gsync order*).*

Deadlock freedom
Correct recovery

Topology-awareness

$\langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$

Model extensions

Distribution of processes

Decreasing failure prob.

$$\frac{D_j \cdot \binom{|C|}{2} \cdot \binom{H_j - 2}{x_j - 2}}{\binom{H_j}{x_j}}$$

Holistic fault-tolerance library

Design and optimizations

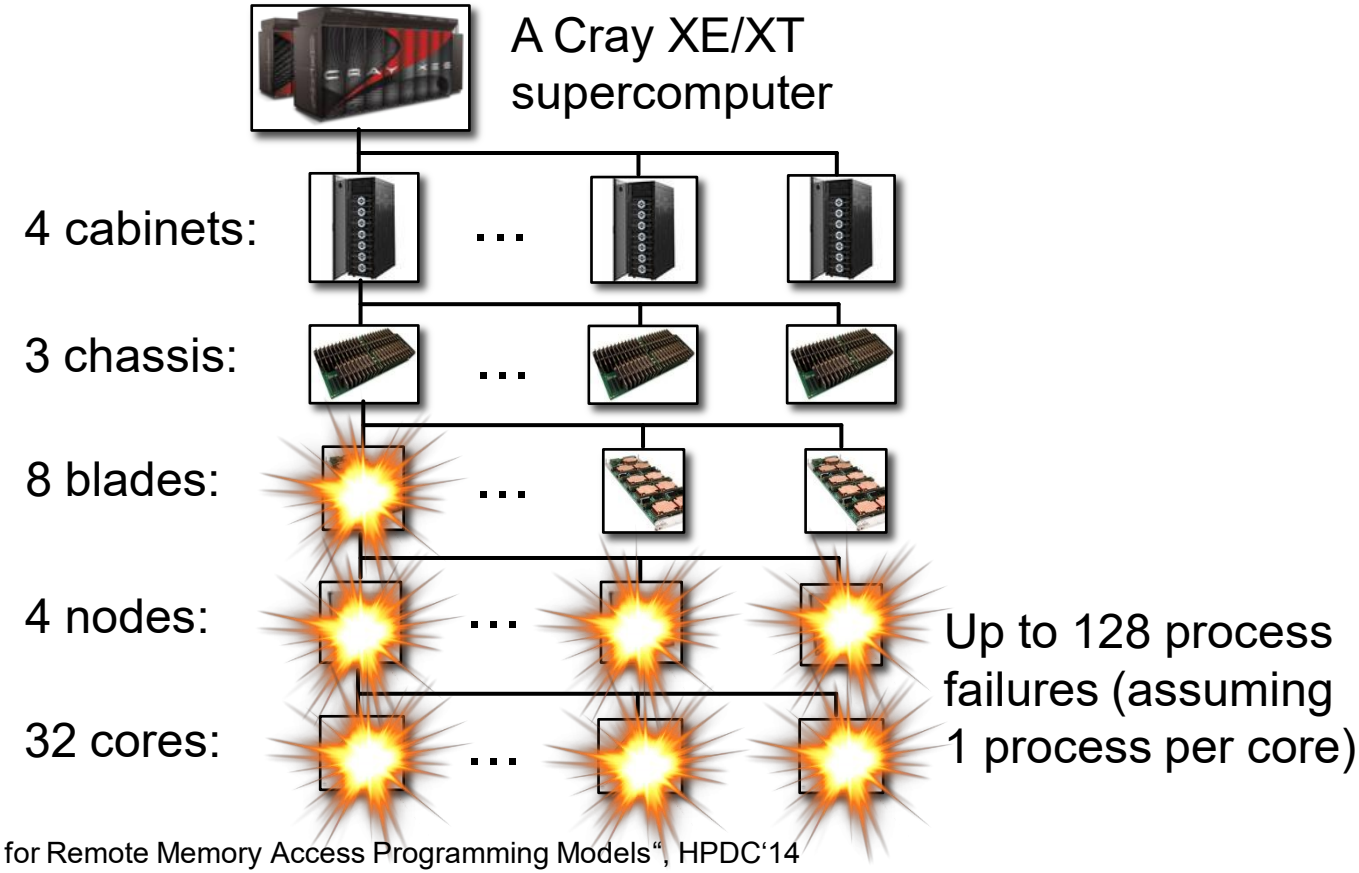
Checkpoints on demand

Optimum CC intervals

Performance

TOPOLOGY-AWARENESS FOR INCREASING RESILIENCE

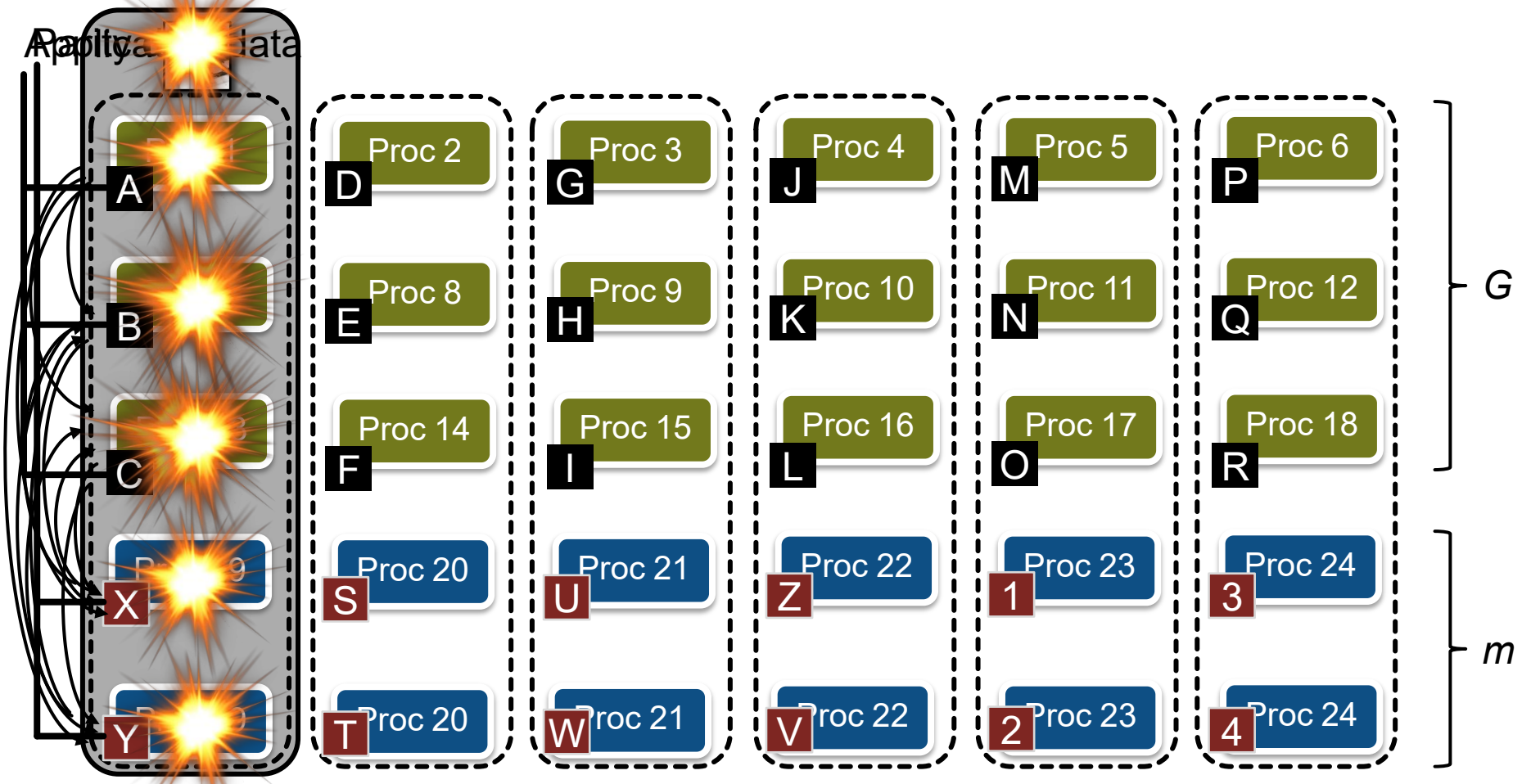
- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes...
- ...protocols will not work and the whole computation is lost ☹



Besta, TH: „ Fault Tolerance for Remote Memory Access Programming Models“, HPDC'14

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

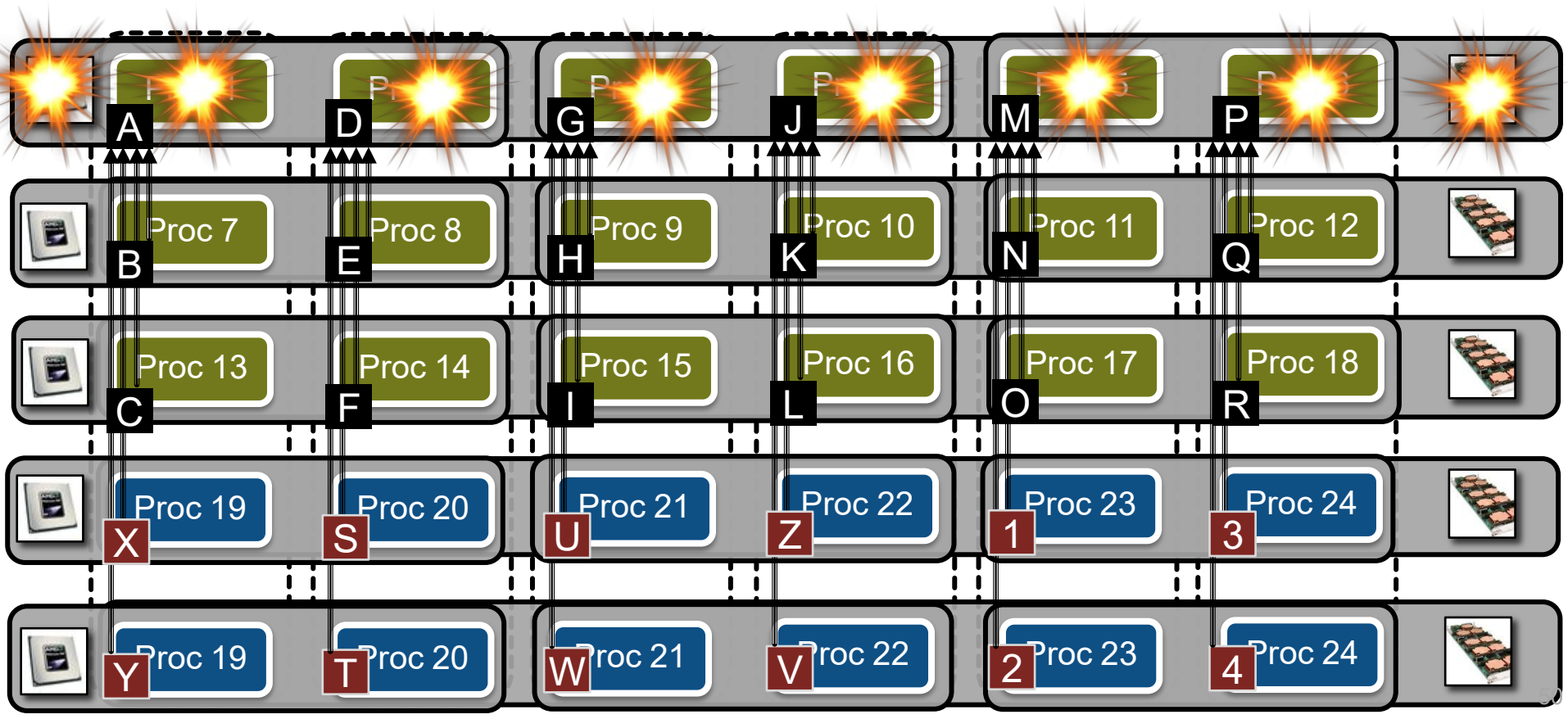
- Step 1: groups of processes:
  - Divide processes into groups of size *G* each
  - Add *m* parity processes to each group to store the parity data



TOPOLOGY-AWARENESS FOR INCREASING RESILIENCE

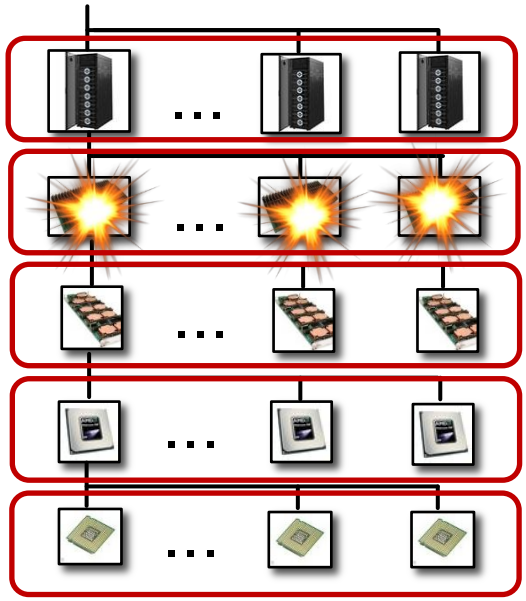
- Step 2: topology-aware distribution of groups:
  - For example, apply topology-awareness at the level of blades...

... and nodes



TOPOLOGY-AWARENESS FOR INCREASING RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine (generalizing [1])
  - A catastrophic failure: a failure that takes place if > *m* processes in the same group die



...at every hierarchy level

Every number of x_j elements is considered...

$$P_{cf} = P_j(x_j \cap x_{j,cf}) =$$

Probability that x_j elements of level *j* will fail and cause a catastrophic failure

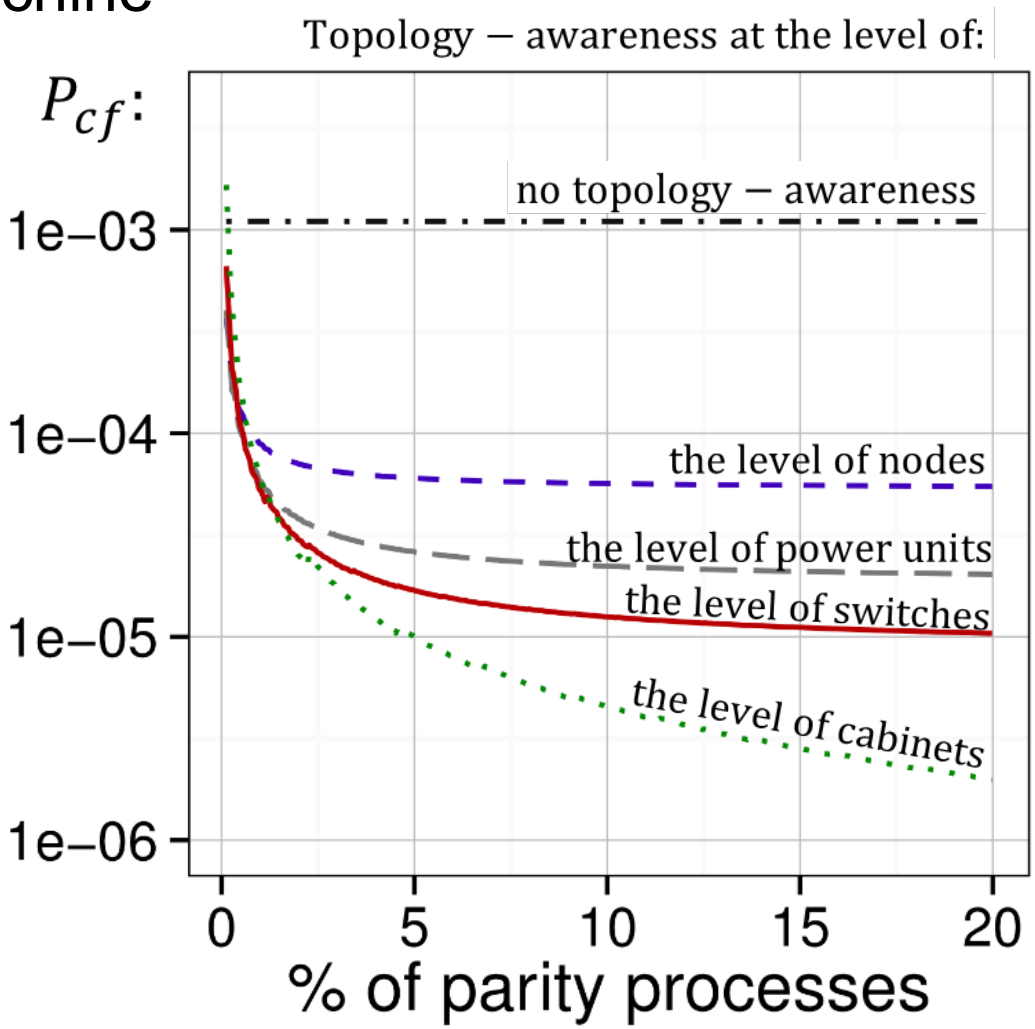
Probability that x_j elements of level *j* will fail

Probability that x_j given failures at level *j* are catastrophic

[1] Bautista-Gomez et al. FTI: high performance Fault Tolerance Interface for hybrid systems. SC11

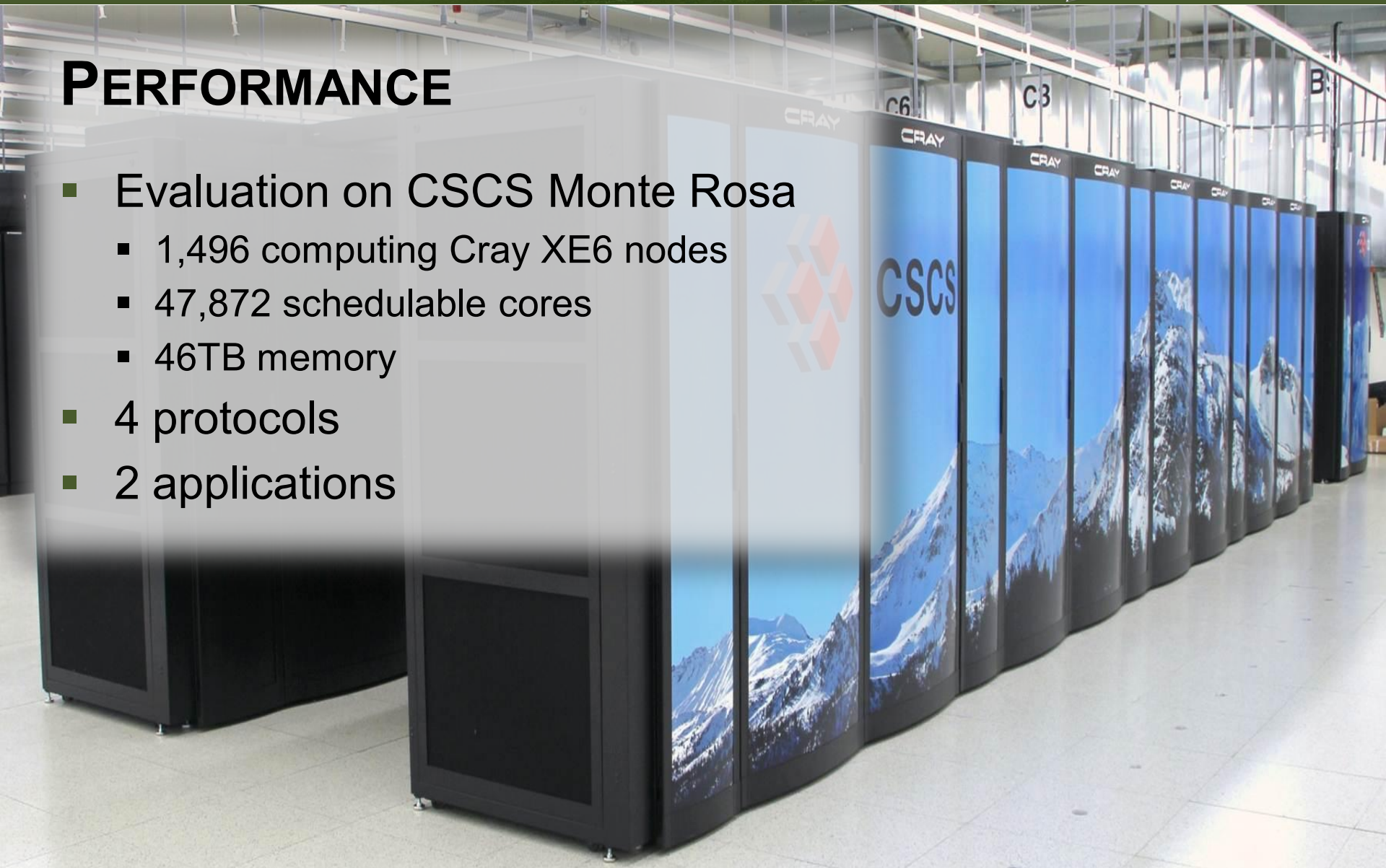
TOPOLOGY-AWARENESS FOR INCREASING RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine

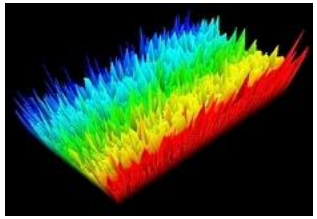


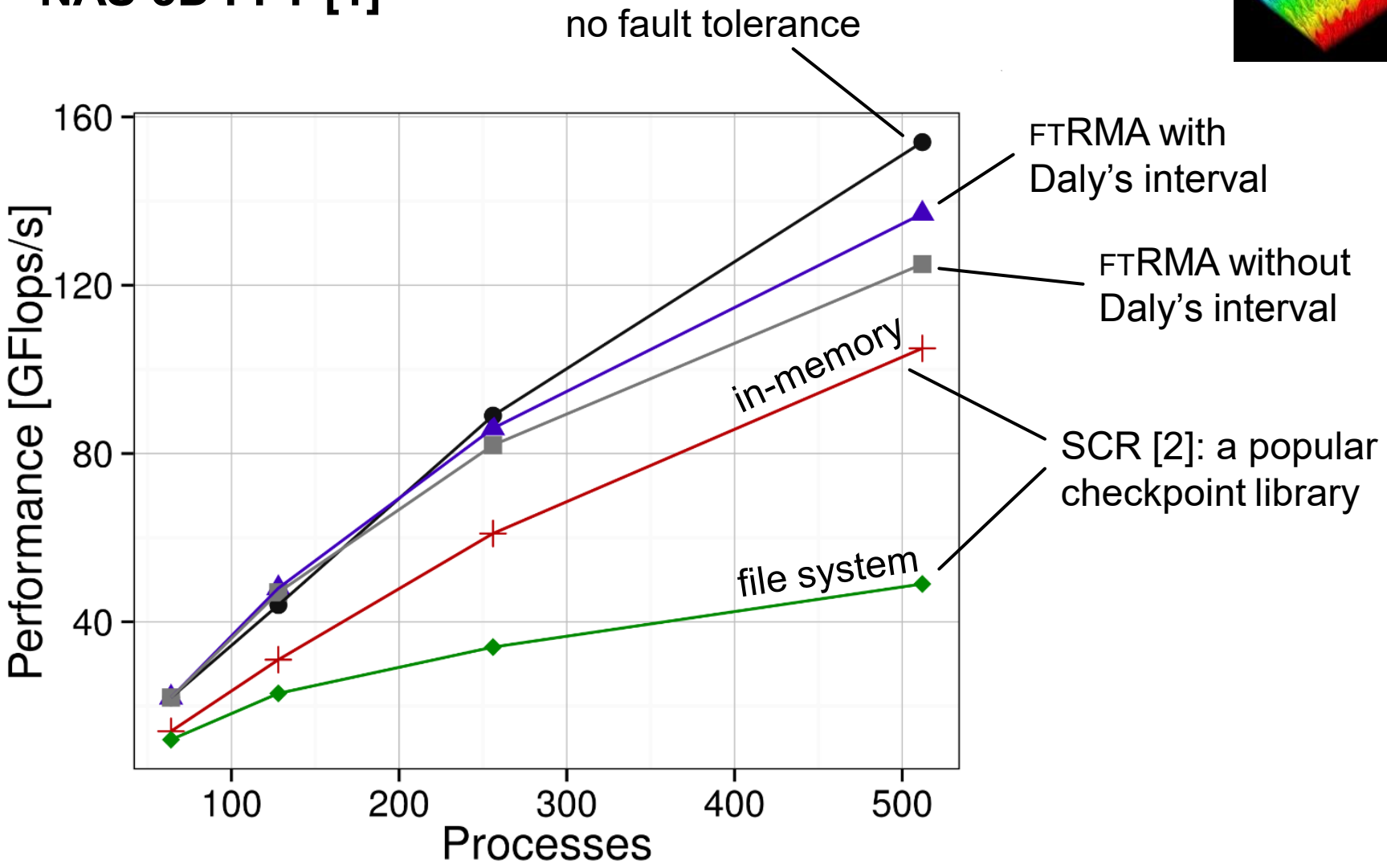
PERFORMANCE

- Evaluation on CSCS Monte Rosa
  - 1,496 computing Cray XE6 nodes
  - 47,872 schedulable cores
  - 46TB memory
- 4 protocols
- 2 applications



PERFORMANCE: COORDINATED CHECKPOINTING
NAS 3D FFT [1]

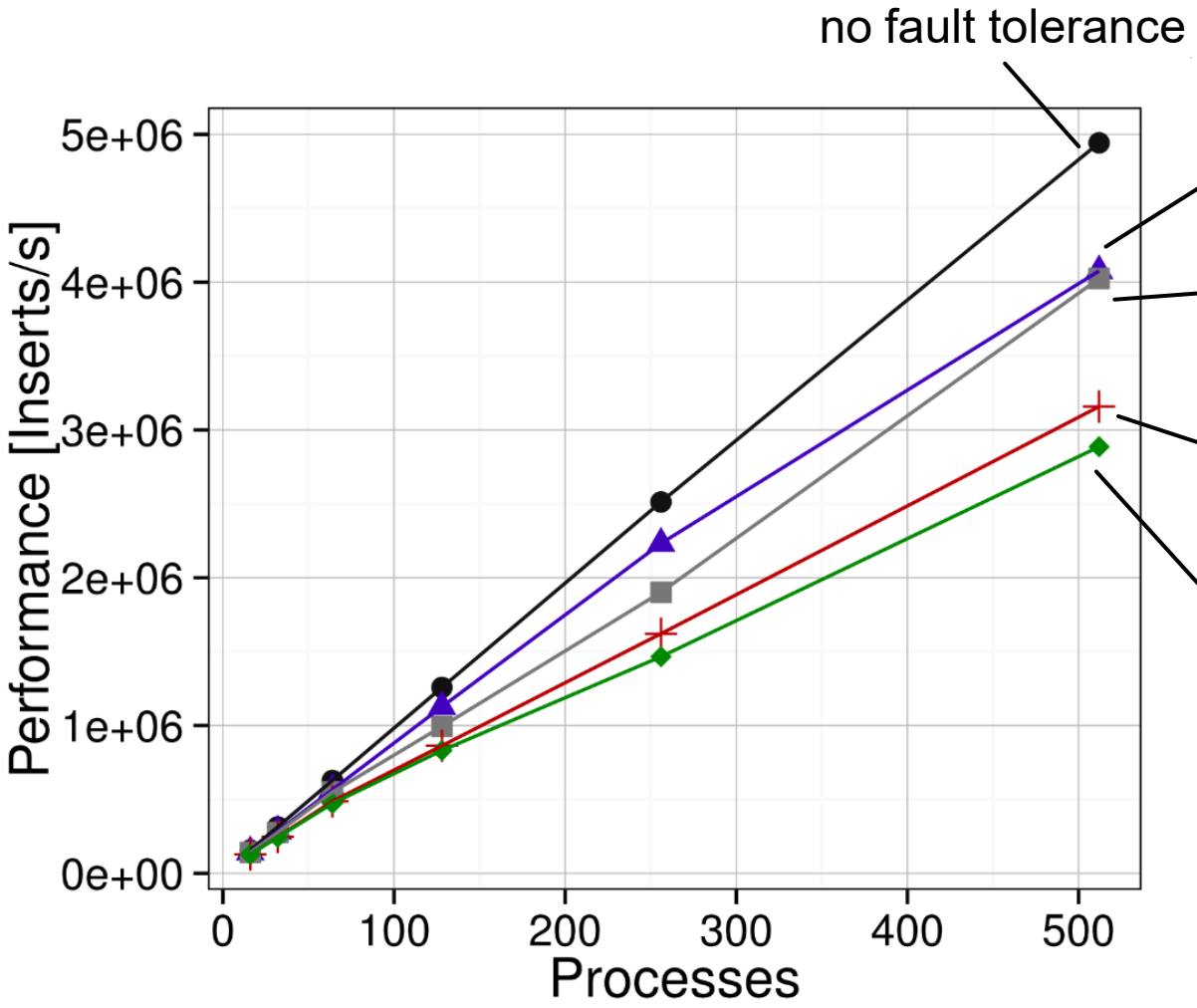




[1] Nishtala et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap. IPDPS'09
[2] Moody et al. Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. SC10

PERFORMANCE: LOGGING OF PUTS & GETS

DISTRIBUTED HASHTABLE



An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

Other Recent Results & Open Challenges

- **Networking (topology, routing)**
  - Besta, TH: "Slim Fly: A Cost Effective Low-Diameter Network Topology", SC14, best student paper finalist
  - Domke, TH, Matsuoka: "Fail-in-Place Network Design: Interaction between Topology, Routing Algorithm and Failures", SC14
  - Prisacari, TH, et al.: "Efficient Task Placement and Routing in Dragonfly Networks", HPDC'14
  - ➢ ***… optimal routing? Topology mapping? On-chip topologies? …***
- **Resilience**
  - Ferreira, et al., TH: "Understanding the Effects of Communication and Coordination on Checkpointing at Scale", SC14
  - ➢ ***… optimal checkpointing? Lowest storage vs. runtime overhead? …***
- **Performance modeling for complex applications**
  - Bhatacharyya TH: "PEMOGEN: Automatic Adaptive Performance Modeling during Program Runtime", PACT'14
  - TH, Kwasniewski: "Automatic Complexity Analysis of Explicitly Parallel Programs", SPAA'14
  - ➢ ***… designing optimal implementations? Automated modeling for co-design? ...***

ACKNOWLEDGMENTS

Paul Hargrove (and the whole UPC team), the MPI Forum RMA WG, many others, …

… and

NCSA

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

CSCS

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

NSF

Google™

Thanks for your attention!