Stony Brook University

(Ideas for a) Design of a New Synchronisation Scheme for MPI RMA

Joseph Schuchart, Institute for Advanced Computational Science, Stony Brook University

Joseph.Schuchart@stonybrook.edu

Thomas Gillis, NVIDIA (frmly Argonne National Lab)

EuroMPI/Australia, September 26, 2024, Perth, Australia

FAR BEYOND

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

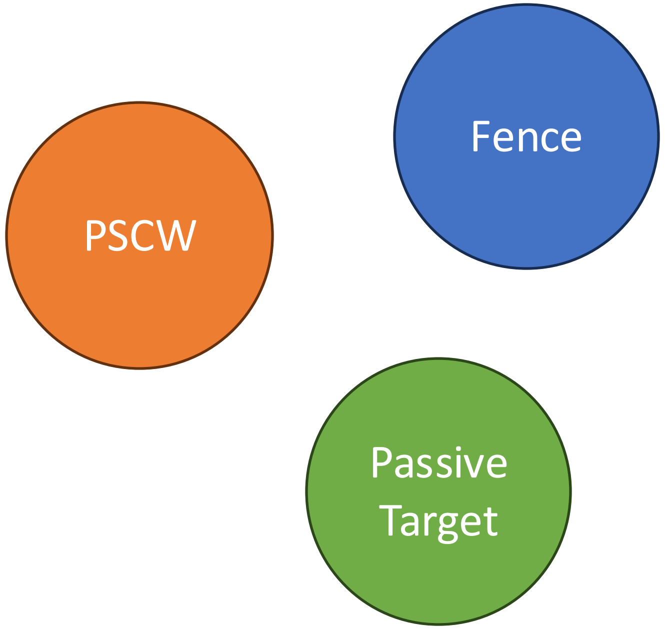
Disclaimer

- **Work in progress**
- Started in 2023
- Disrupted by changes in employment and funding
- Prelim discussions in the RMA WG

Joseph
UTK → SBU

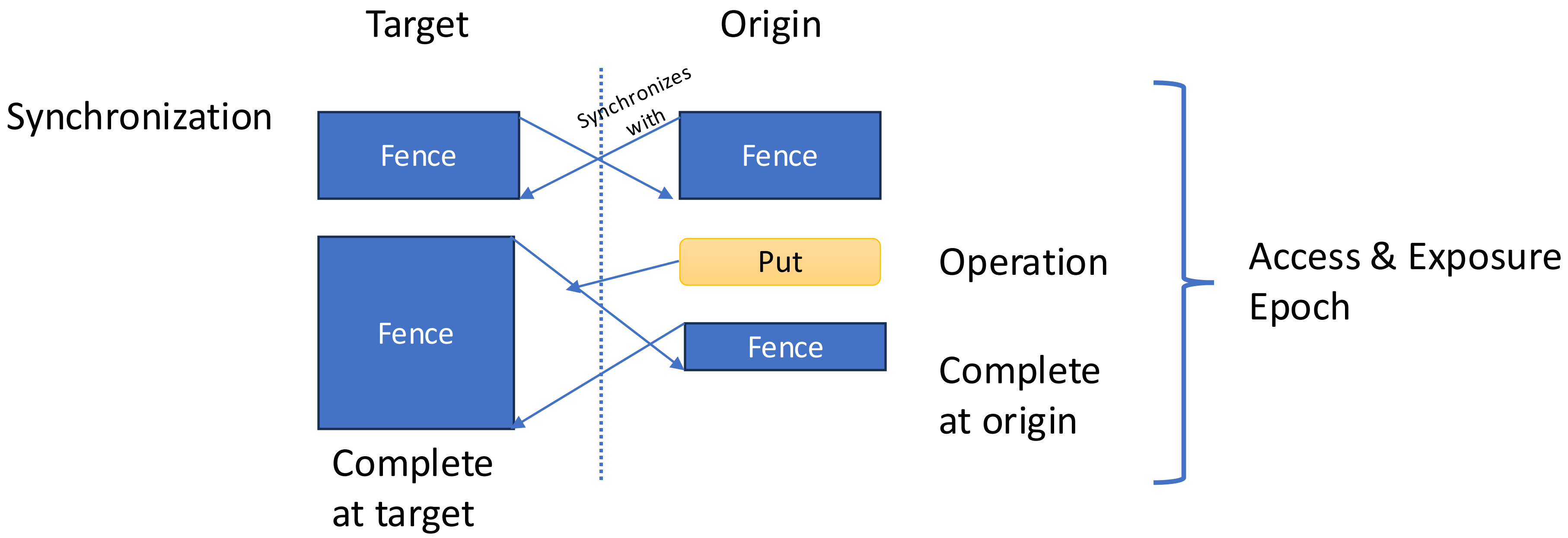
Thomas
ANL → Nvidia

20 Years of MPI RMA



Version	Date	Features
MPI 1	May 5, 1994	
MPI 2	Nov 15, 2003	Fence PSCW Locks
MPI 3.0	Sept 21, 2012	Lockall Flush Sync Allocated Win Shared Win Dynamic Win
MPI 3.1	June 4, 2015	
MPI 4.0	June 9, 2021	
MPI 4.1	Nov 2, 2023	Flexible Shared Memory Atomic operation splitting

RMA Terms


Target
Origin
Synchronization
Fence
Synchronizes with
Fence
Put
Operation
Fence
Fence
Complete at origin
Complete at target
Access & Exposure Epoch

Motivation

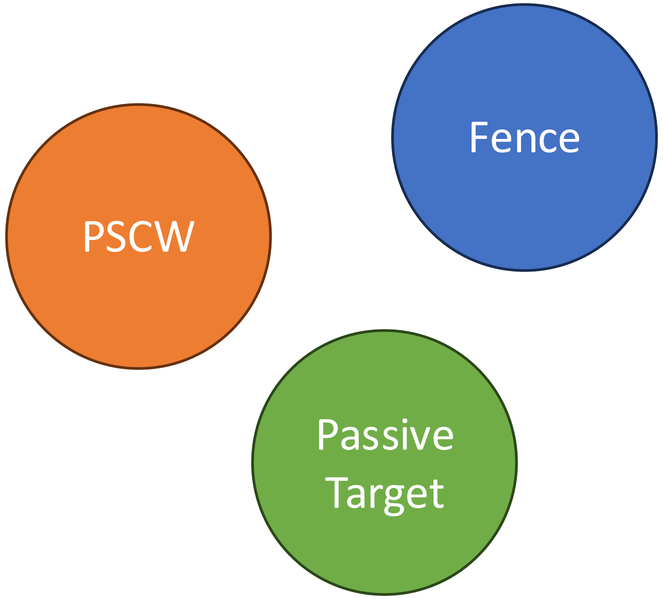
Three synchronization methods in MPI RMA

- Confusing rules
- Mutually exclusive usage

Data movement is easy, **synchronization** is hard

Synchronization has **process-scope**

What would a clean-slate approach look like?



Stony Brook University

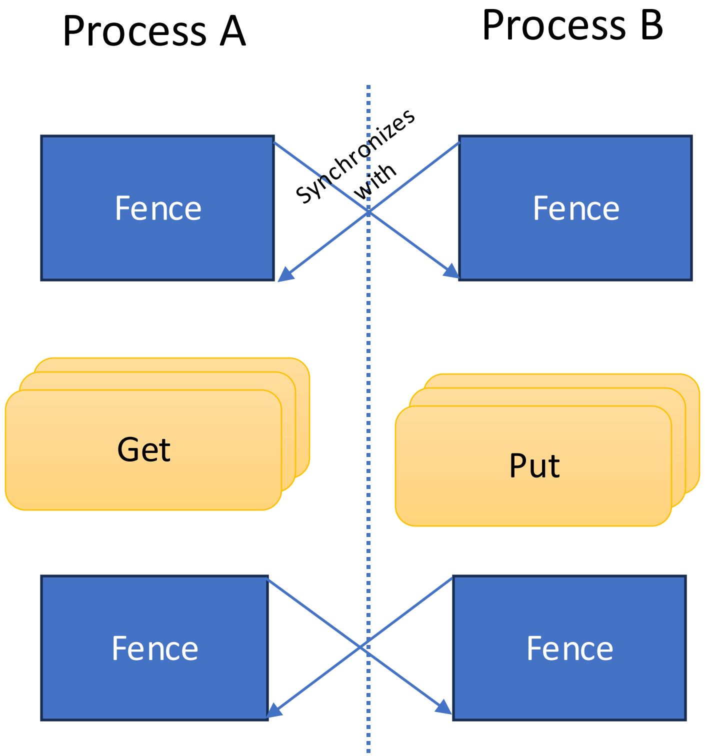
Review: MPI_Fence

Collective synchronization

Upon return on Process A:

- Operations for which Process A is the **target** will have completed at Process A ("remote completion")
- Operations for which Process A is the **origin** will have completed at Process A ("local completion)"

Fast on some networks

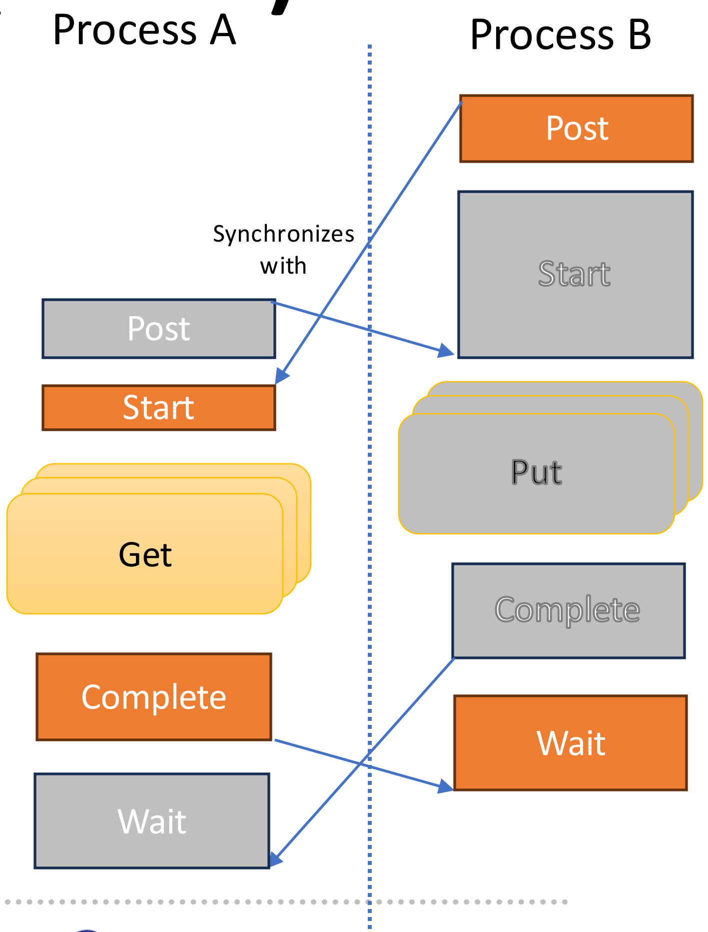


Review: Generalized Active Target (PSCW)

Post: open exposure epoch
Start: open access epoch
Complete: close access epoch
Wait: close exposure epoch

P2P synchronization in flexible peer groups

One signal per peer



Review: Passive Target Synchronization

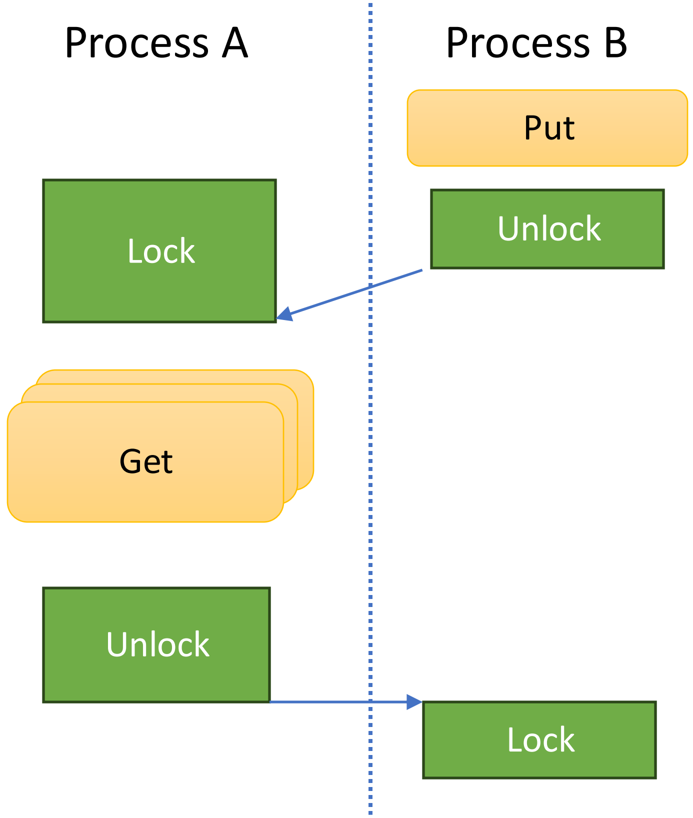
Mutually exclusive access epochs

- Lock: waits for other access epochs to complete
- Unlock: ensures completion at target & origin

No exposure epochs

P2P synchronization

Reader/Writer synchronization through shared & exclusive locks



Review: Passive Target Synchronization

Mutually exclusive access epochs

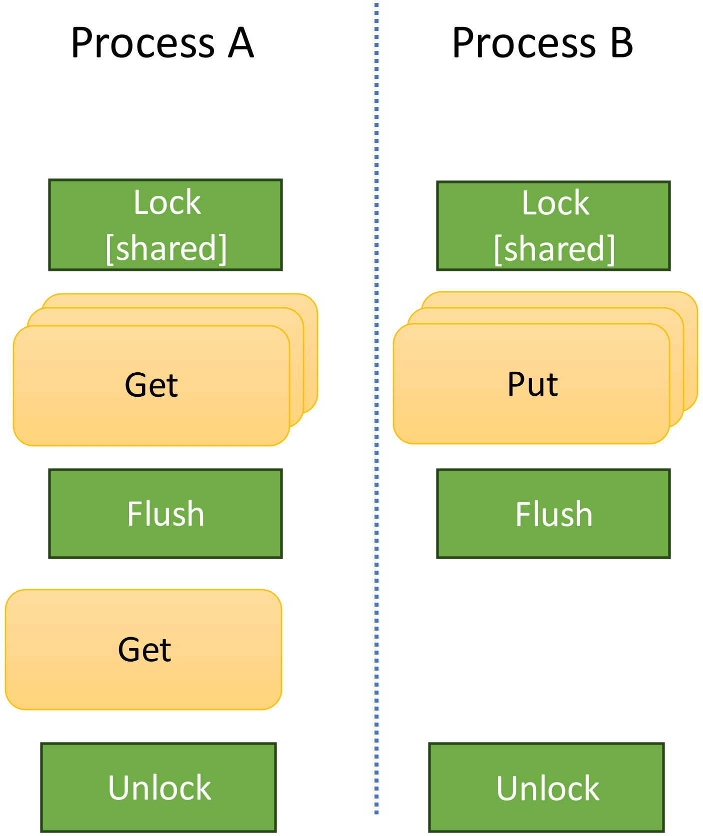
- Lock: waits for other access epochs to complete
- Unlock: ensures completion at target & origin

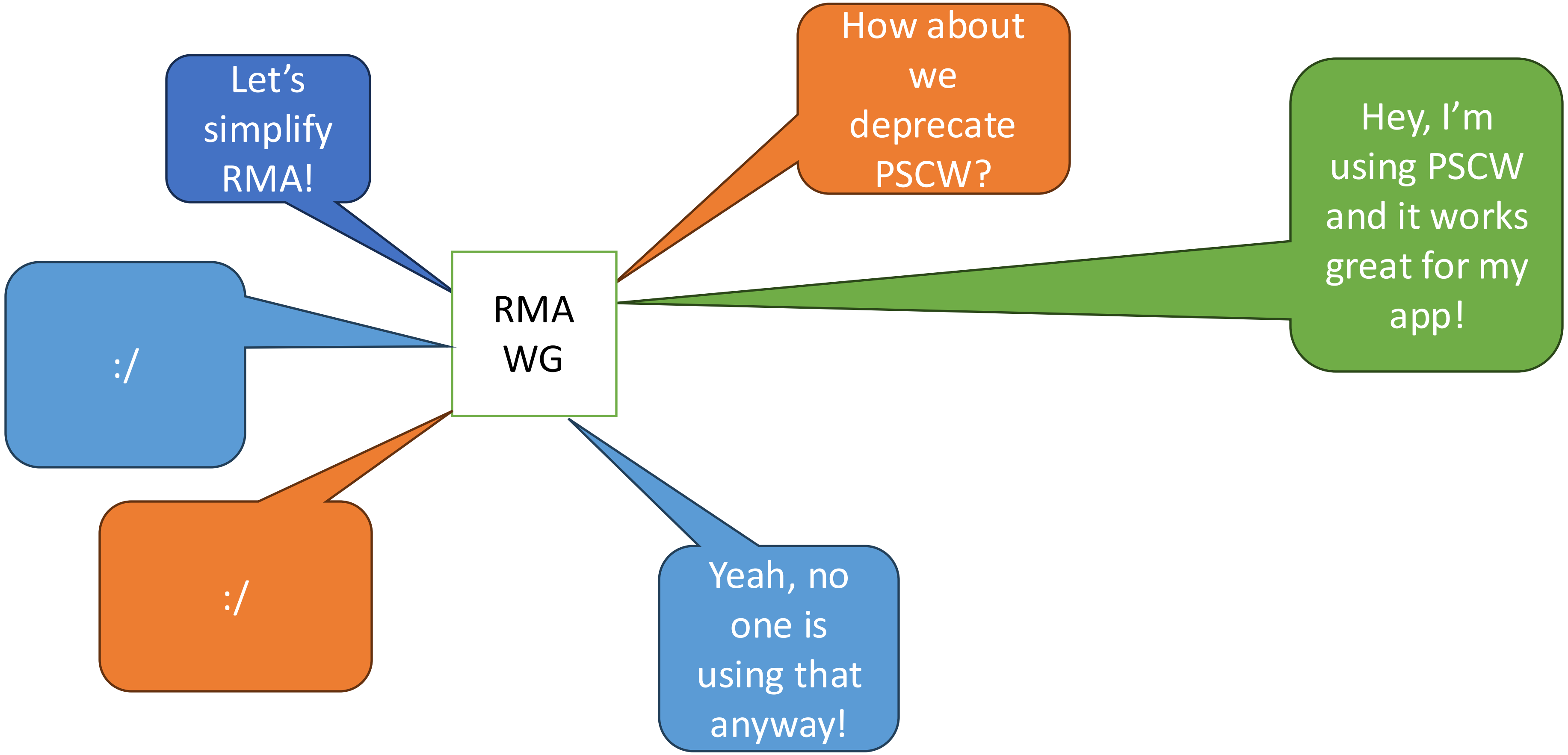
No exposure epochs

P2P synchronization

Reader/Writer synchronization through shared & exclusive locks

“Bulletin-style” communication **without** synchronization
Closest to shared memory & OpenSHMEM

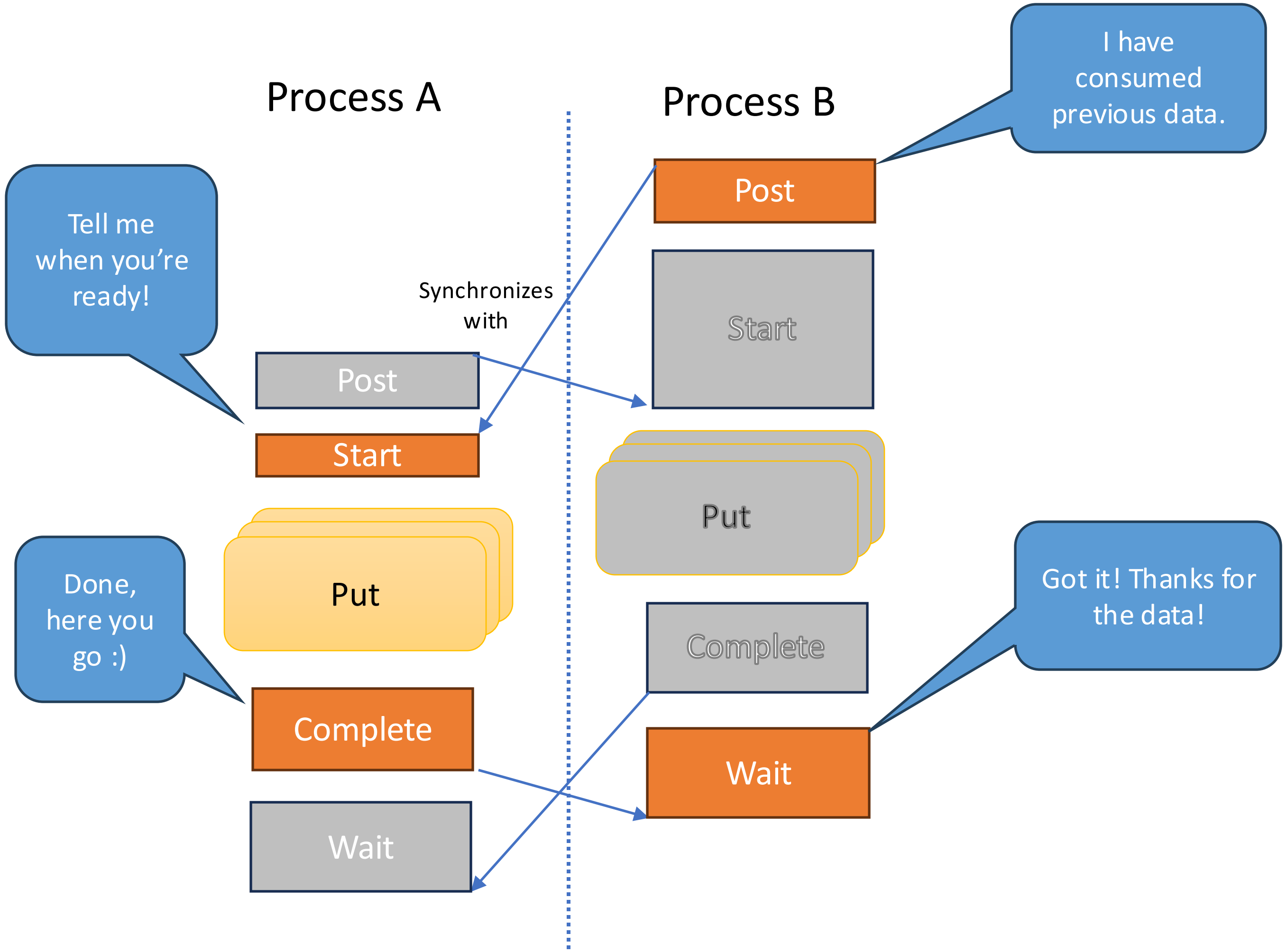



Let's simplify RMA!
How about we deprecate PSCW?
Hey, I'm using PSCW and it works great for my app!
:/
RMA WG
:/
Yeah, no one is using that anyway!

Stony Brook University

Why PSCW?

Flexible Bidirectional Synchronization



But: Multi-Threading Challenges

Multiple threads may initiate RMA operations

Only one thread must synchronize

Threads must join before synchronizing the window

Or

Application must roll their own synchronization scheme


ONE SYNCHRONIZATION MODEL
TO RULE THEM ALL
imgflip.com

Stony Brook University

Enter: Signals

Bi-directional synchronization mechanism

Combine exposure and access epochs

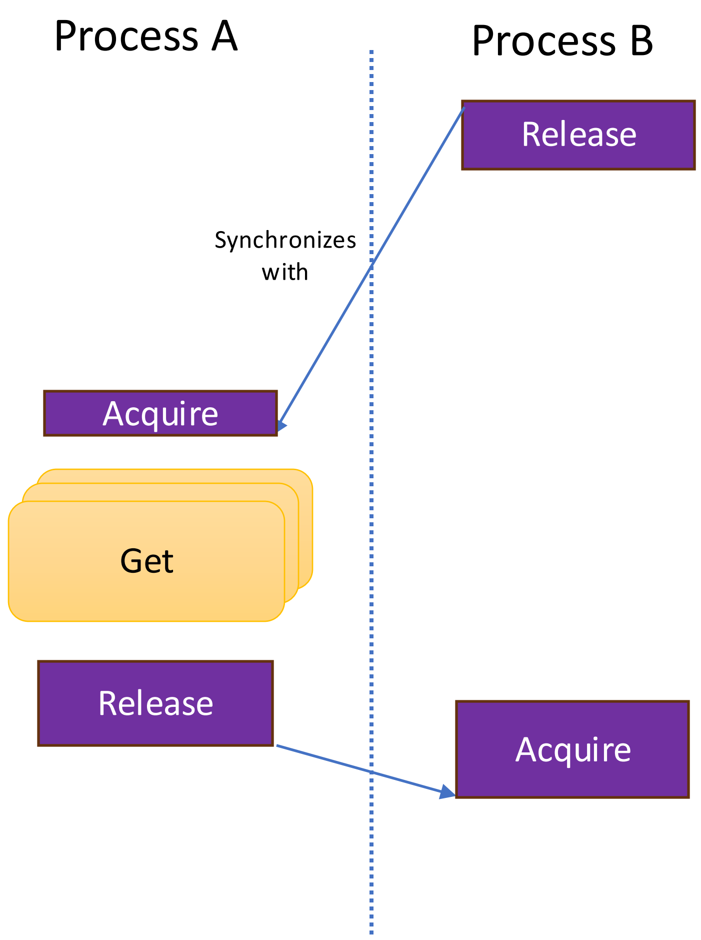
Acquisition: wait for prior use to complete

- Memory availability

Release: completes operations and notifies target

- Data avilability

PSCW: one signal for all peers



From One To Many Signals

Signals should be identifiable

Max number of signals known up front (e.g., number of threads)

Number of signals specified during window creation

Global naming

Aggregating Operations: Batches

Map **sets of operations** to Signals

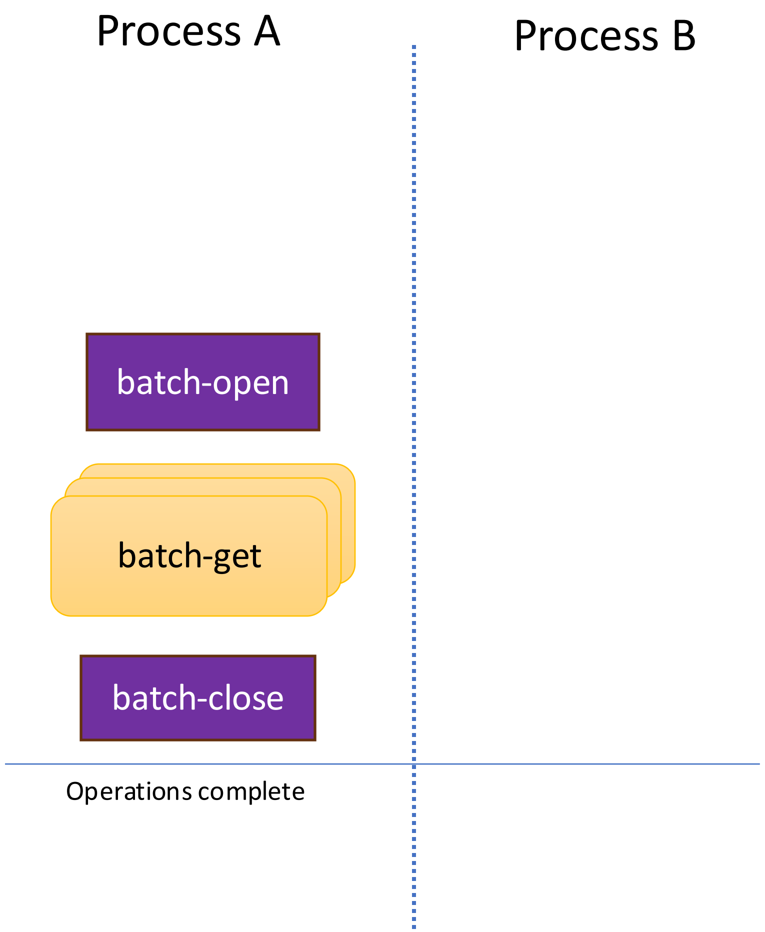
Completion of a batch releases the signal at the target

Arbitrary number of batches

Batches without signals: **thread-scope passive target**

Allows aggregation of small operations

Windows are always exposed

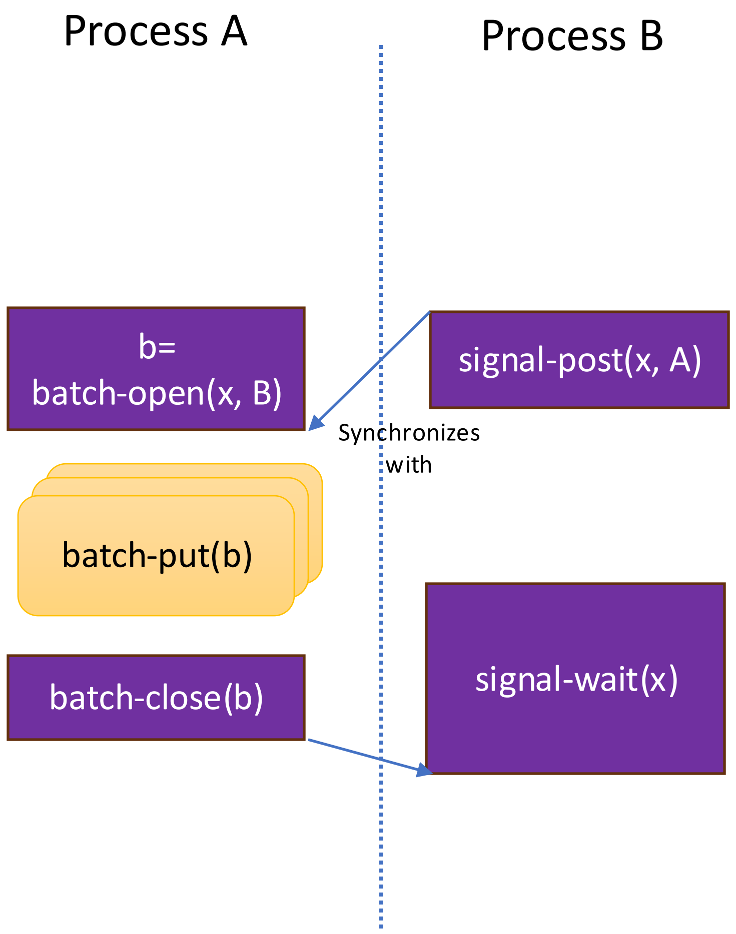


Stony Brook University

Batches & Signals

Batches may synchronize with a signal

Simplest case: P2P synchronization



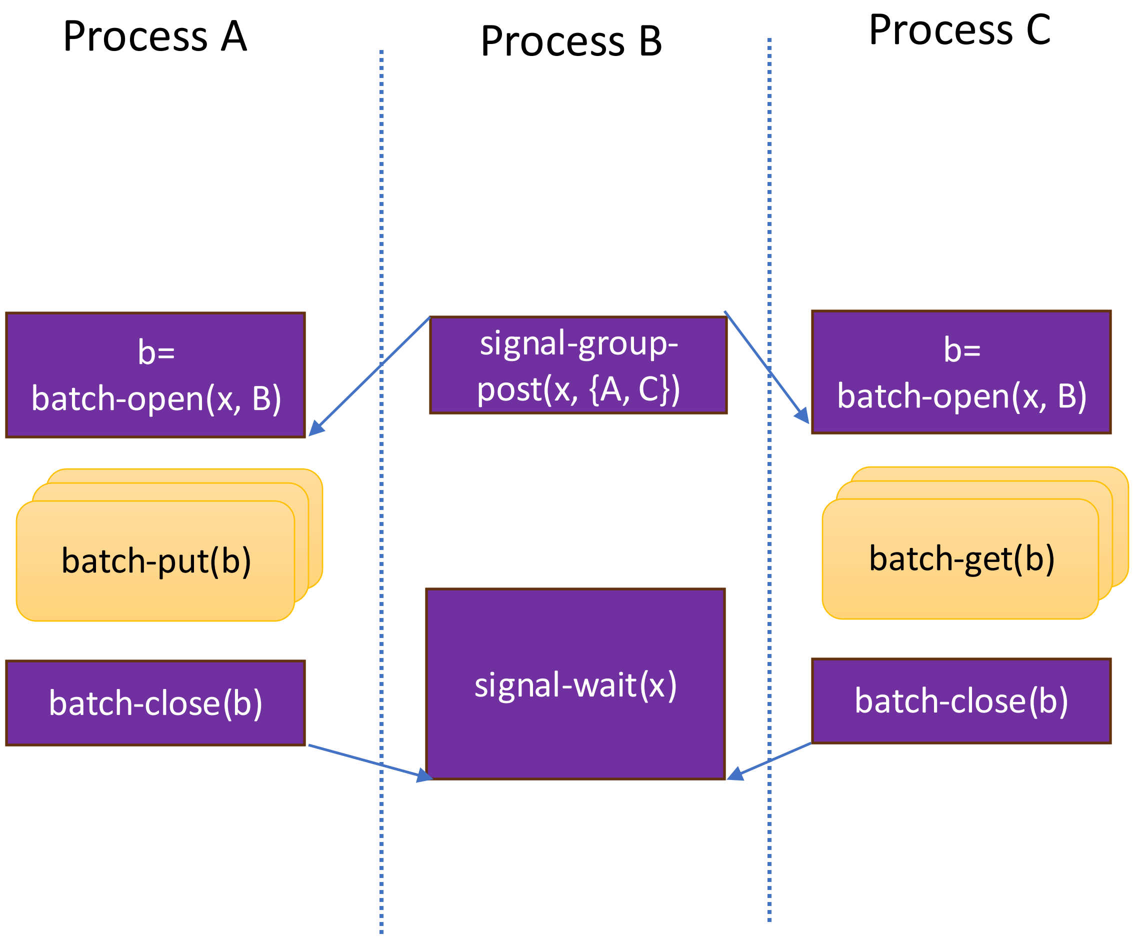
FAR BEYOND

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Batches, Signal & Groups

Batches may synchronize with a signal

Signal release may depend on multiple peers



Stony Brook University

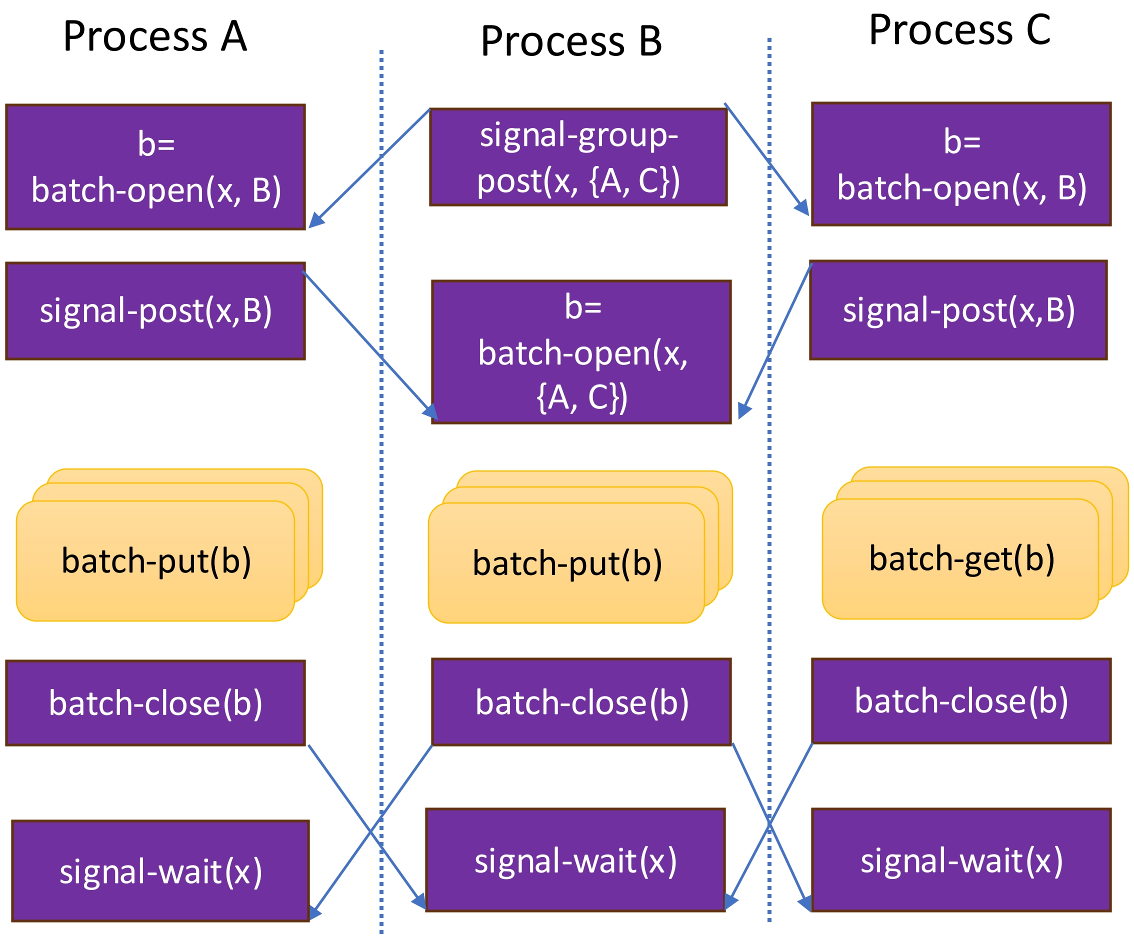
Batches, Signal & Groups

Batches may synchronize with a signal

Signal release may depend on multiple peers

Batches may release signals on multiple peers

Single Signal replaces PSCW



Stony Brook University

Batches, Signal & Groups

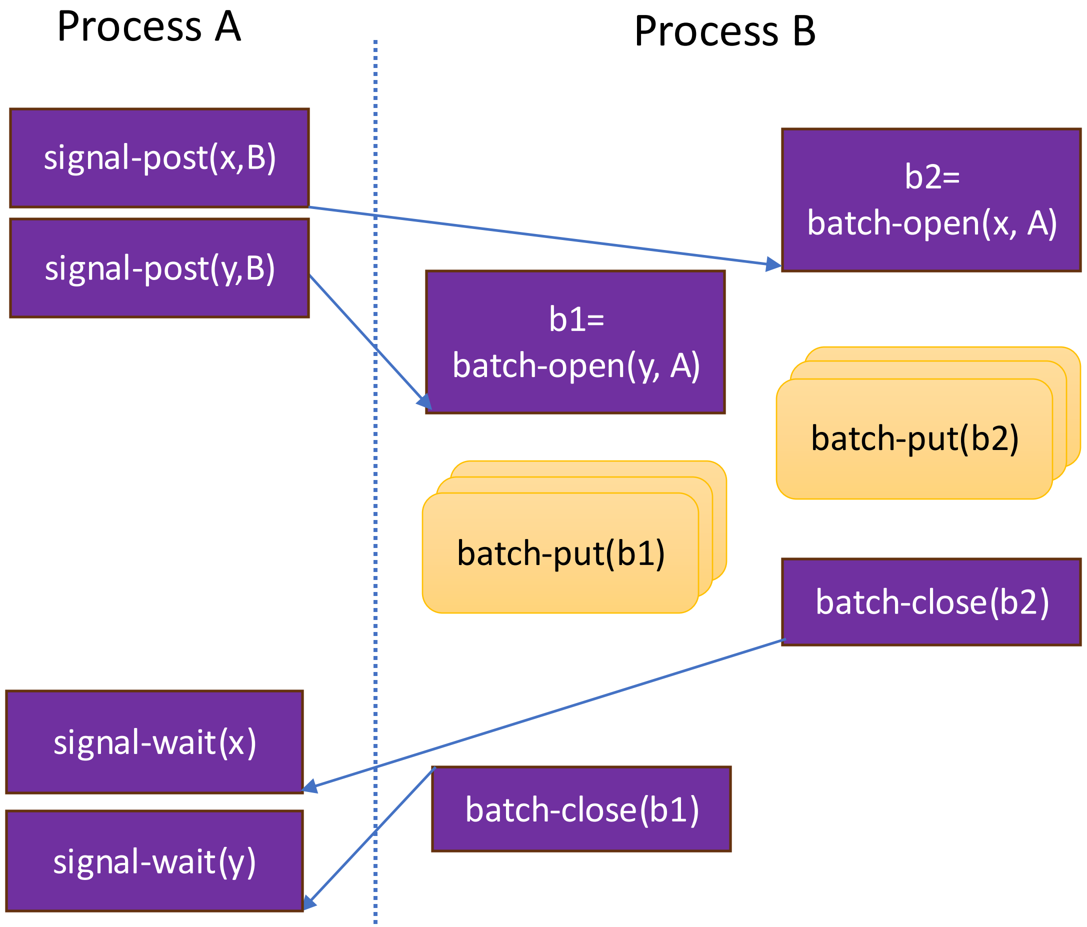
Batches may synchronize with a signal

Signal release may depend on multiple peers

Batches may release signals on multiple peers

Single Signal replaces PSCW

Multiple signals & batches provide thread-scope synchronization



Memory Semantics

Acquire: batch-open & signal-wait acquire the signal

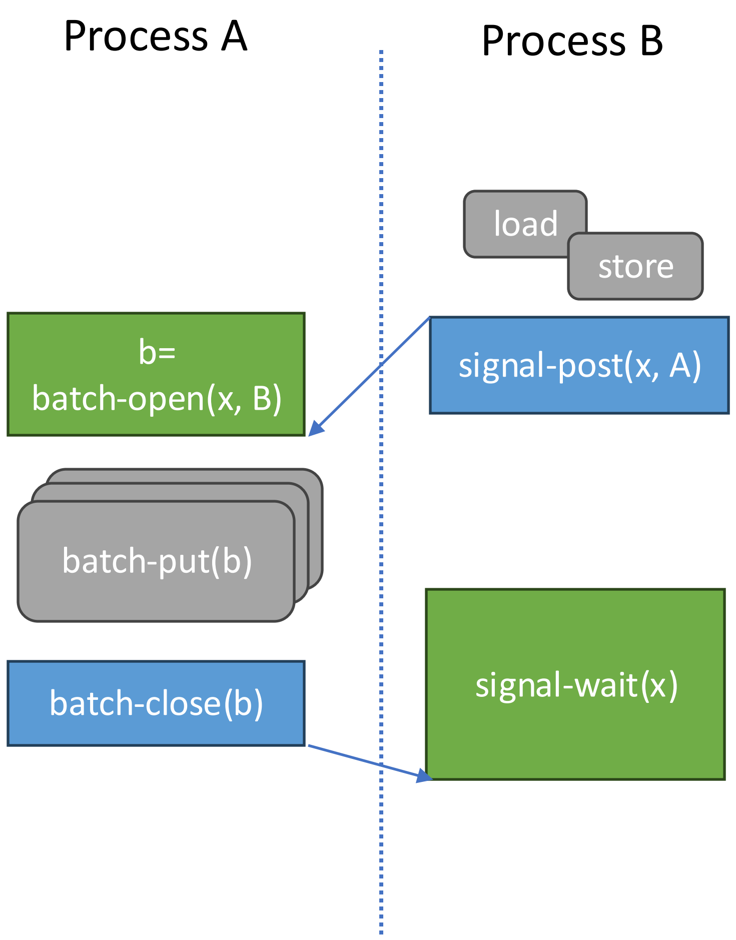
Release: signal-post & batch-close release the signal

Relaxed: put/get & load/store operations have no ordering guarantee

Release
All put/get & load/store operations complete before the release.

Acquire
Put/get & load/store operations must occur after the acquire

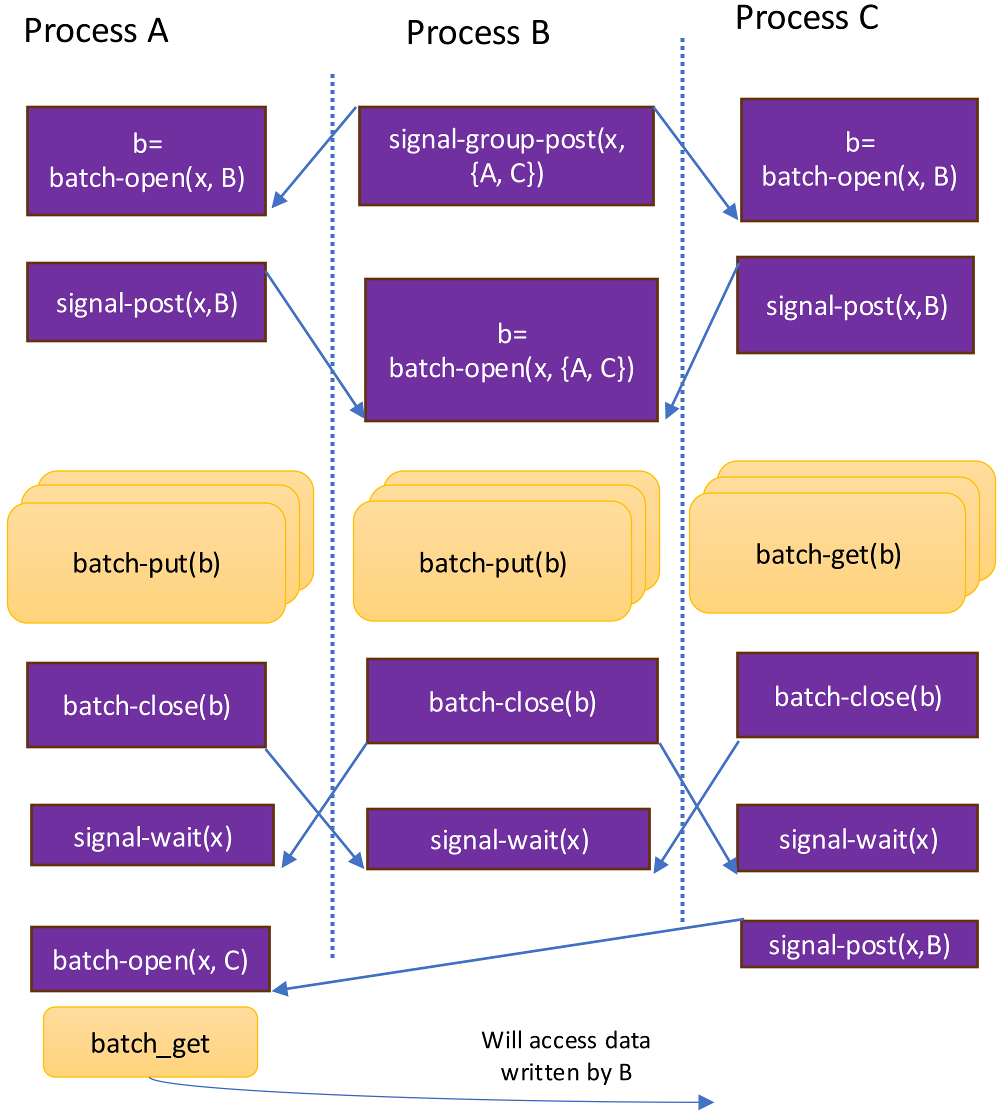
Relaxed
Unordered with respect to other relaxed operations



Semantics: Local completion

Batch-close guarantees **local completion**

- Allows reuse of buffers
- Potentially avoids network latency
- Signal acquisition is ordered with signal release so there is **no race** between Process B and Process A in memory of Process C



Collective Synchronization

All-to-all communication pattern

Signal-fence combines **signal release and acquisition** in a collective operation

- group(comm) ⊆ group(window)

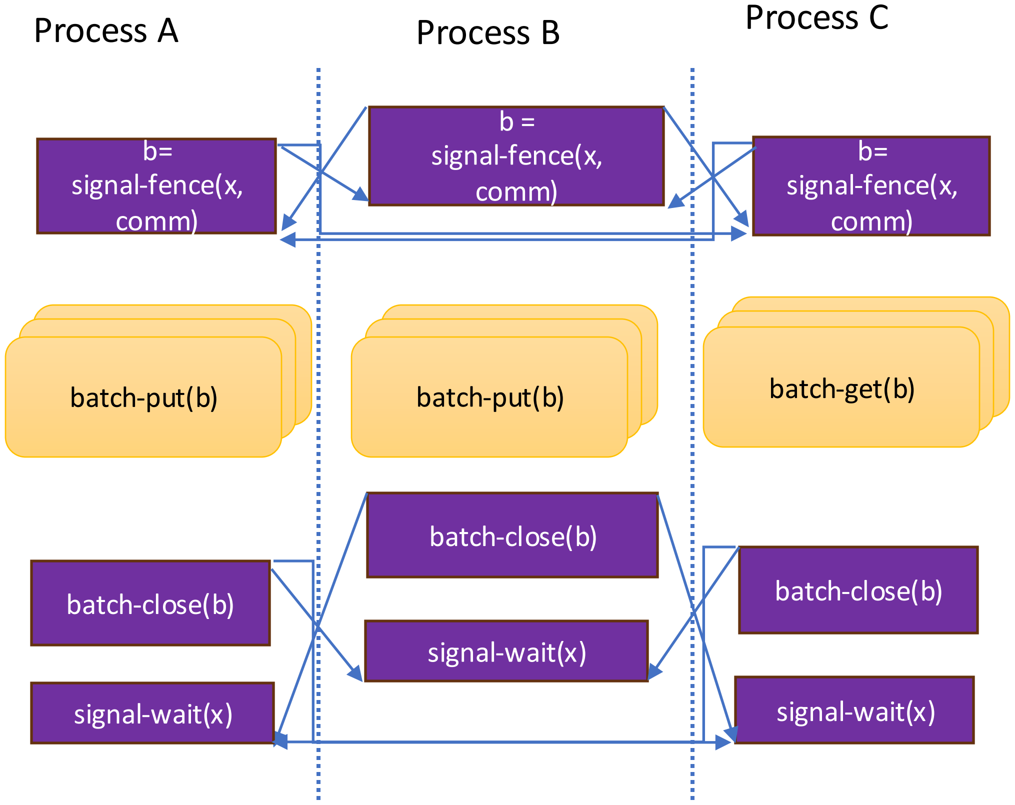
Signal-fence and batch-close operations **nonblocking** to avoid deadlocks

- Potentially nonblocking signal-iwait

Only one epoch per communicator at a time

- Multiple epochs on different communicators

Coexist with P2P & group signals

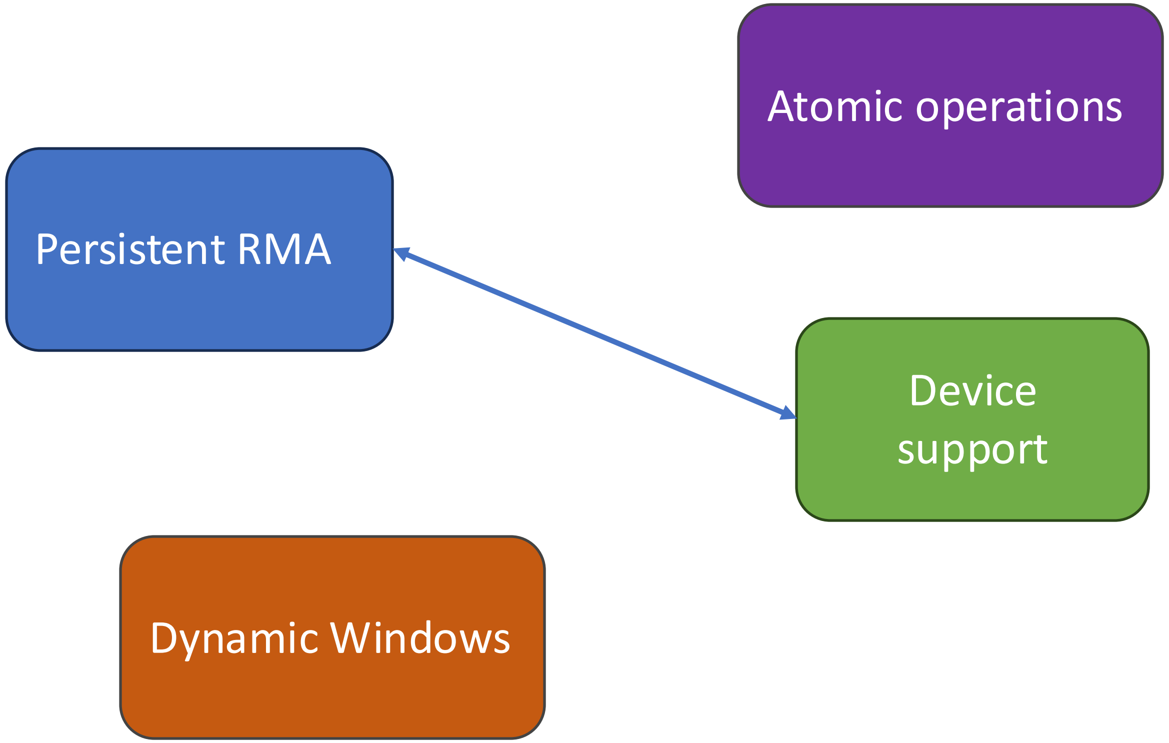


Stony Brook University

Implementation & Evaluation

TBD

Also Under Consideration



Summary

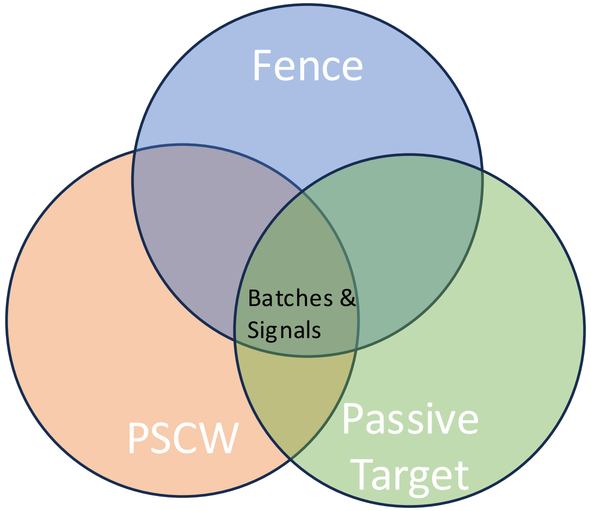
Data movement is easy, synchronization is hard

Signals & Batches provide flexible synchronization mechanism

- Combine all three existing models into one

Separation of concerns

- Windows holds memory
- Batches & Signals provide synchronization



Feedback welcome ☺

Joseph.Schuchart@stonybrook.edu

https://github.com/mpiwg-rma/rma-issues/

Stony Brook University

Acknowledgements

This research was supported partly by NSF awards #1931347 and #1931384, and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. We gratefully acknowledge the provision of computational resources by the Oak Ridge National Laboratory (ORNL) and the High-Performance Computing Center (HLRS) at the University of Stuttgart, Germany.

ECP EXASCALE COMPUTING PROJECT

NSF

OAK RIDGE National Laboratory

HLRS