Adaptive Strategy for One-sided Communication in MPICH2

Xin Zhao, Gopalakrishnan Santhanaraman, and William Gropp

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

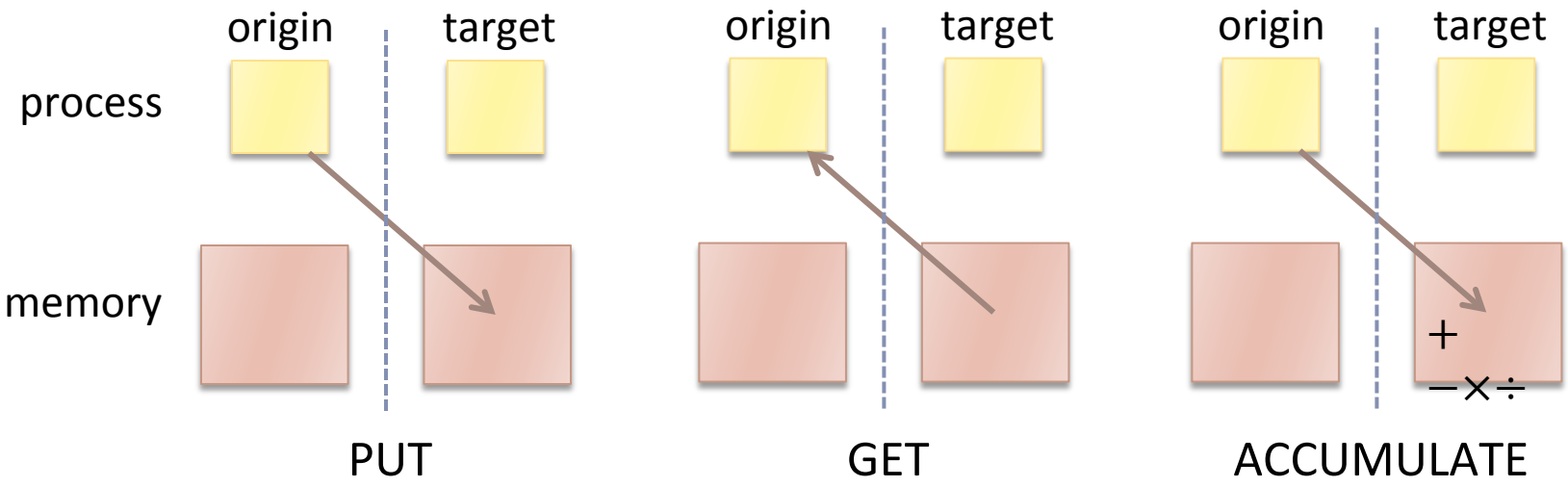
Motivation

- MPI one-sided communication (RMA)
  - supported by MPI-2 since 1997
  - one process specifies all communication parameters
  - more convenient for some computations and has potential for better performance
- When MPI RMA operations can be issued and completed is up to the implementation
  - operations are queued and issued at end in current implementation, reducing network transmissions
    - good for short updates, not for large updates
  - large updates should be issued as early as possible
    - provide maximum communication overlap
  - approach that adaptively handles both cases is necessary to get good performance in both cases

Background

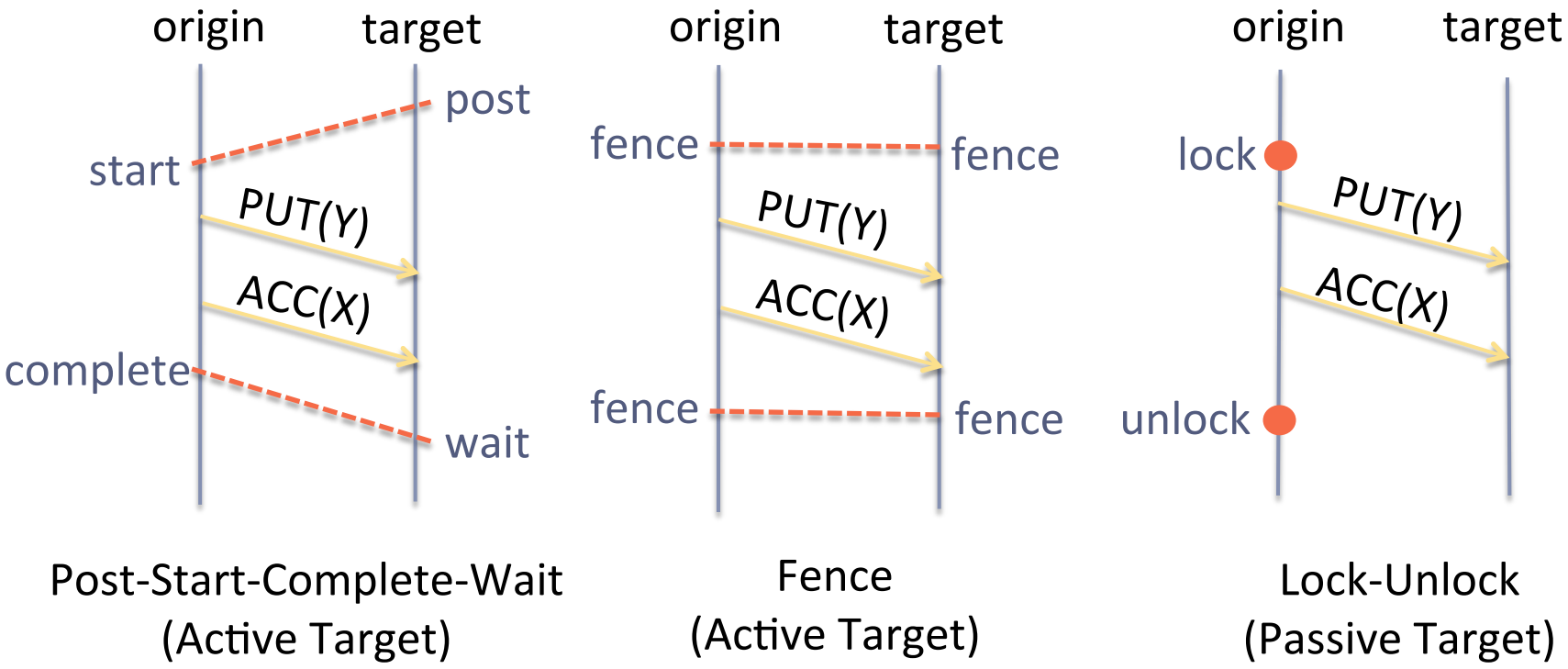
MPI-2 RMA Interface

- Window creation
  - expose target memory accessed by RMA operations
- Three types of one-sided operations



MPI RMA Interface

- Three synchronizations



Post-Start-Complete-Wait
(Active Target)

Fence
(Active Target)

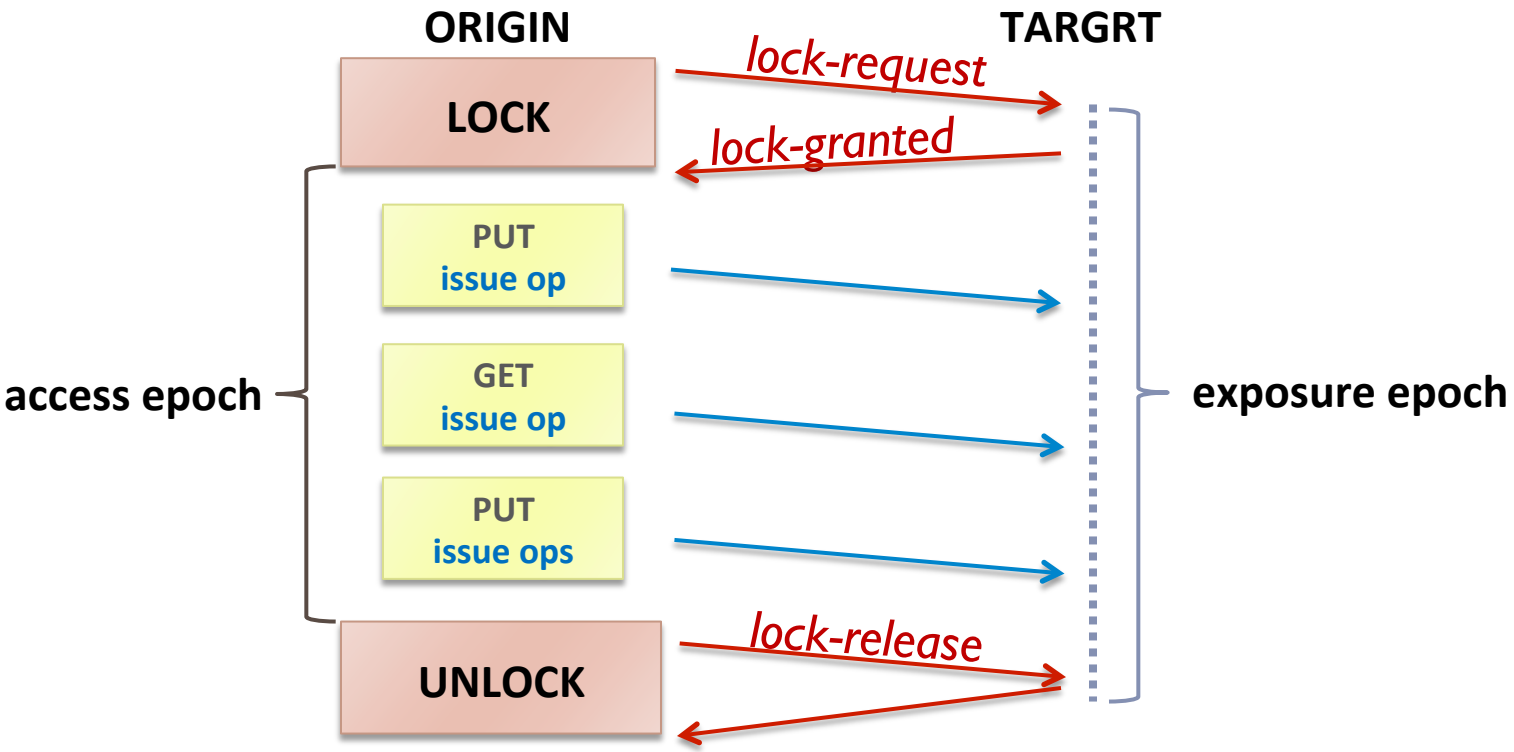
Lock-Unlock
(Passive Target)

- active: target participates in synchronization
- passive: target does not participate in synchronization

Design and Implementation

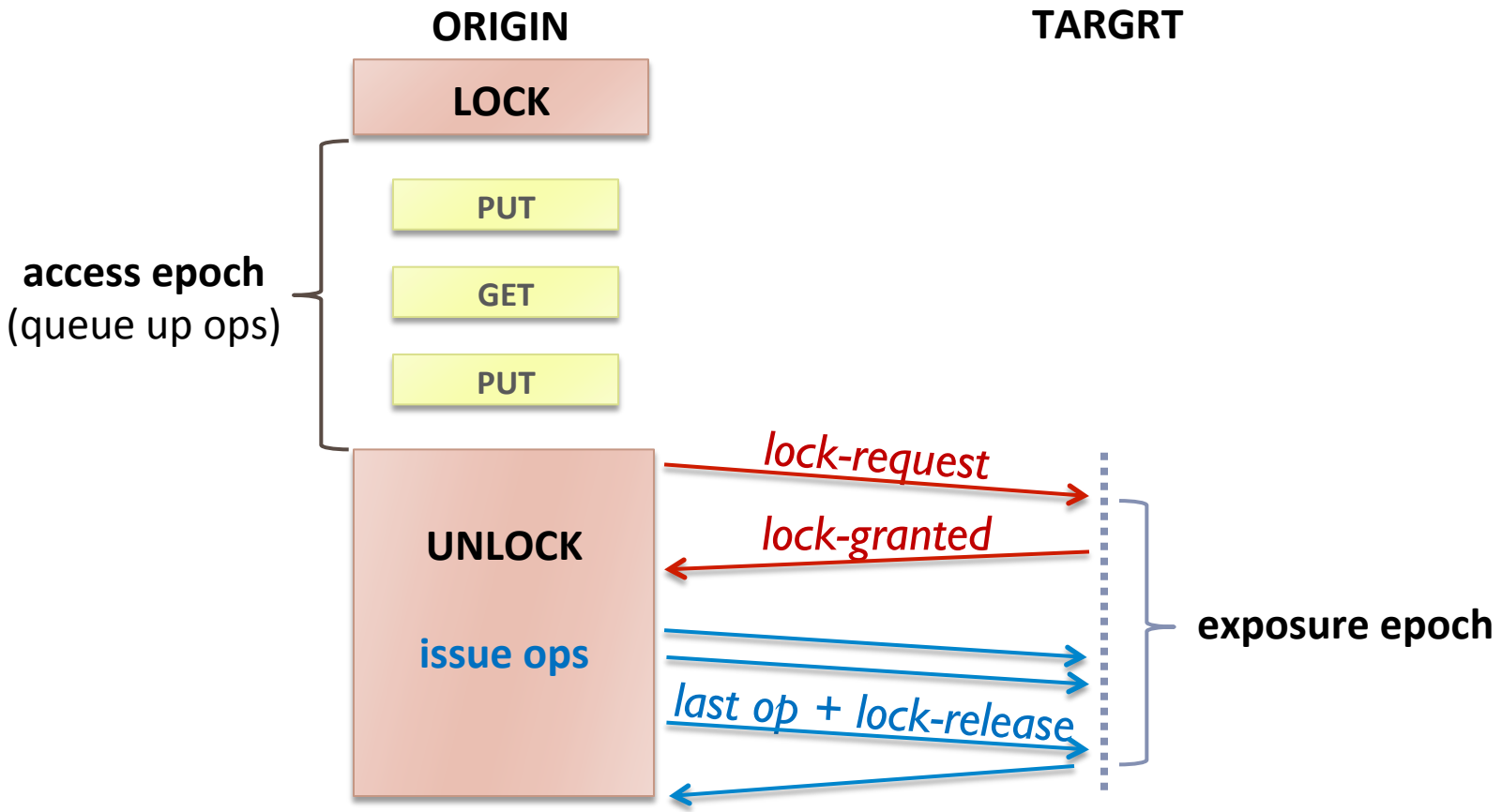
Lock-Unlock

- Basic implementation choice
  - *eager* approach: issue operations immediately
  - two synchronizations



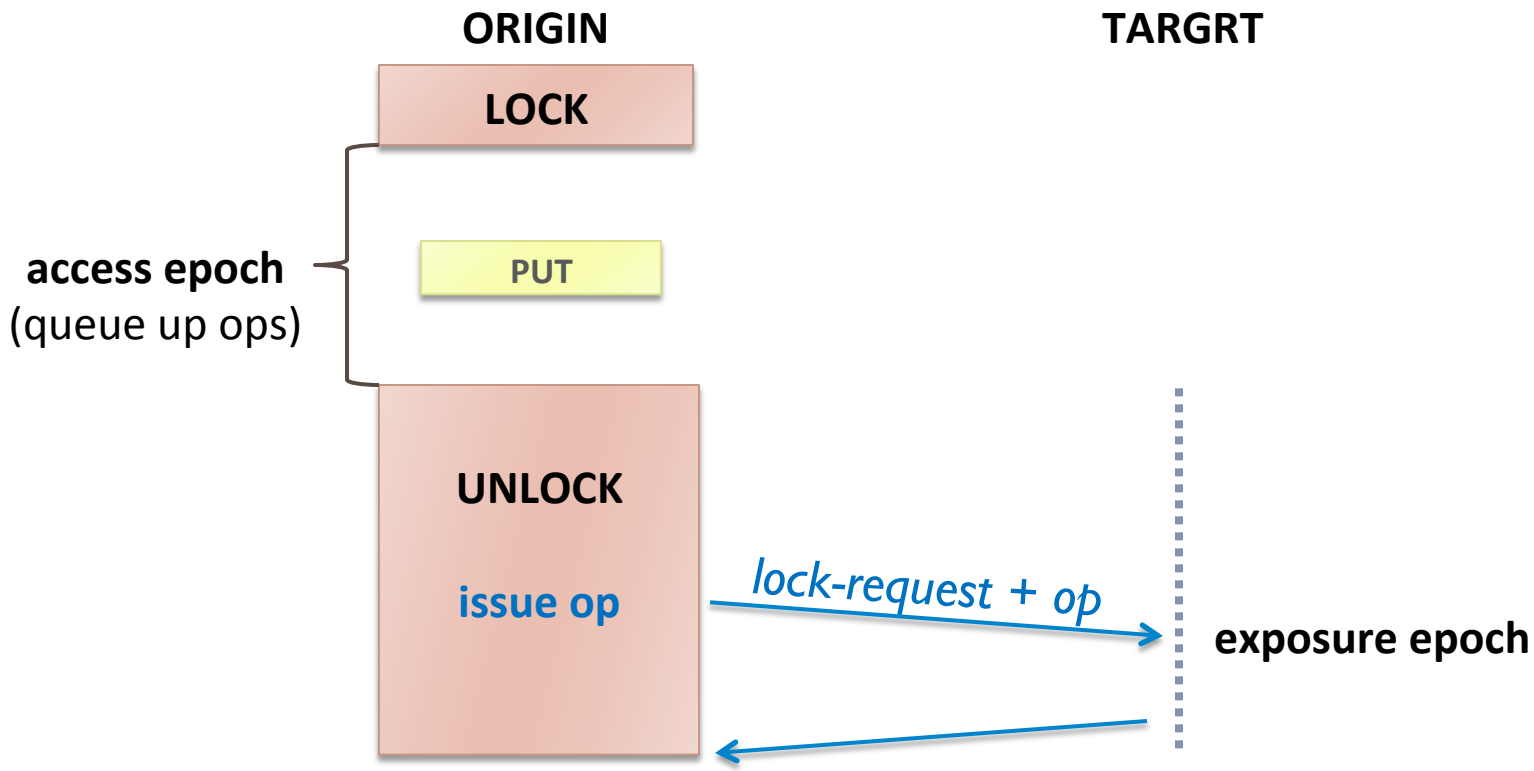
Current Implementation in MPICH2

- *Lazy* approach: do everything in UNLOCK
- Eliminate synchronization message at end



Current Implementation in MPICH2

- Single operation optimization
  - eliminate synchronization messages at beginning
  - cannot be implemented in *eager* approach

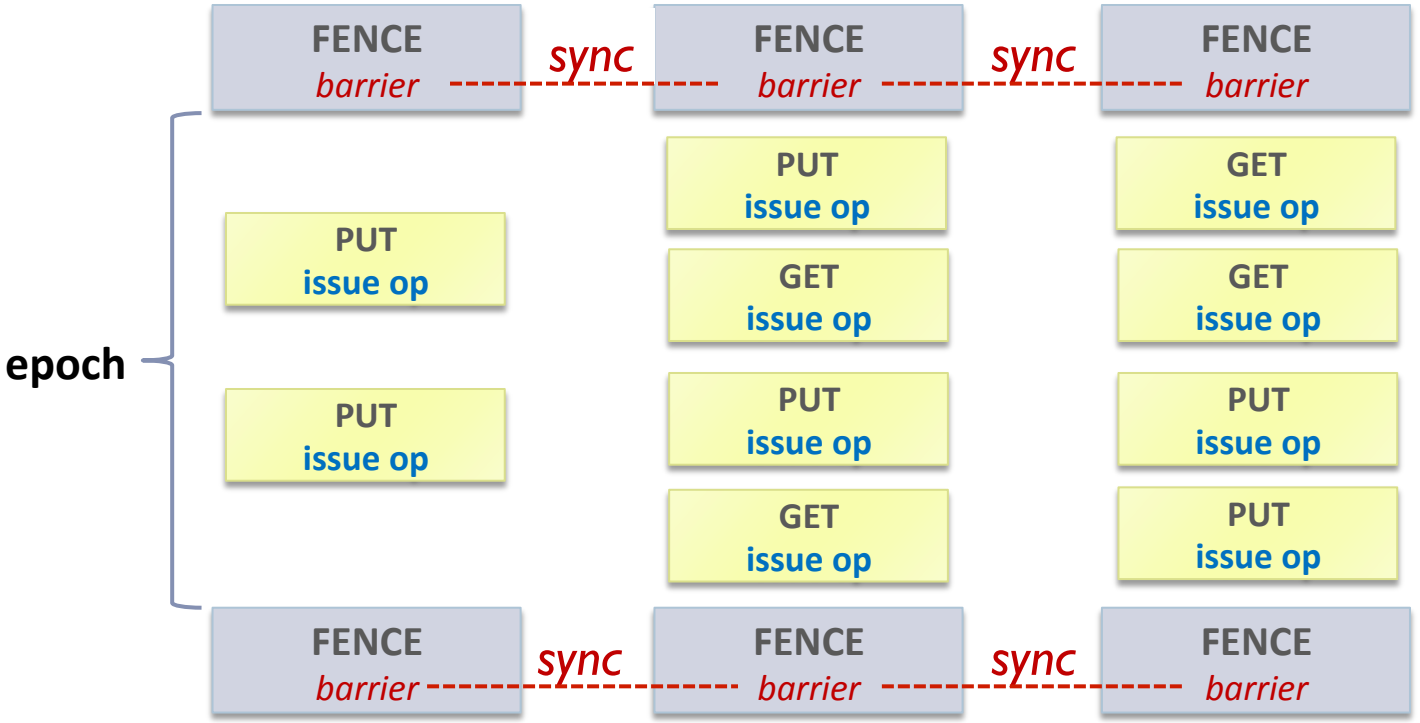


Our Strategy – Adaptive Approach

- Initially performs as LAZY mode
- When encounter large number of operations / large data volume, switch from LAZY mode to EAGER mode
- Gain advantages from both LAZY and EAGER
  - support single operation optimization
  - eliminate the synchronization message at end (when get) – always keep the current last operation

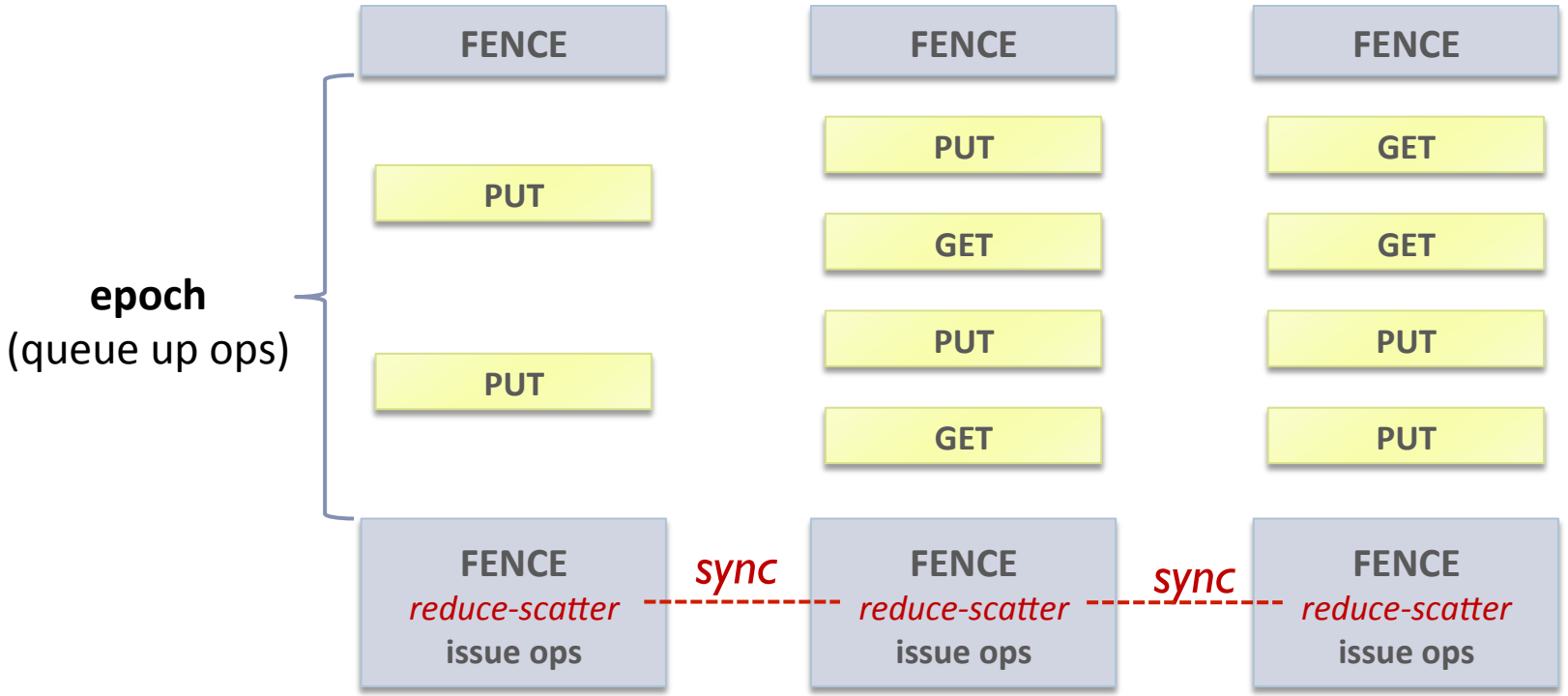
Fence

- Basic implementation choice
  - *eager* approach: issue operations immediately
  - two synchronizations: one at beginning and one at end



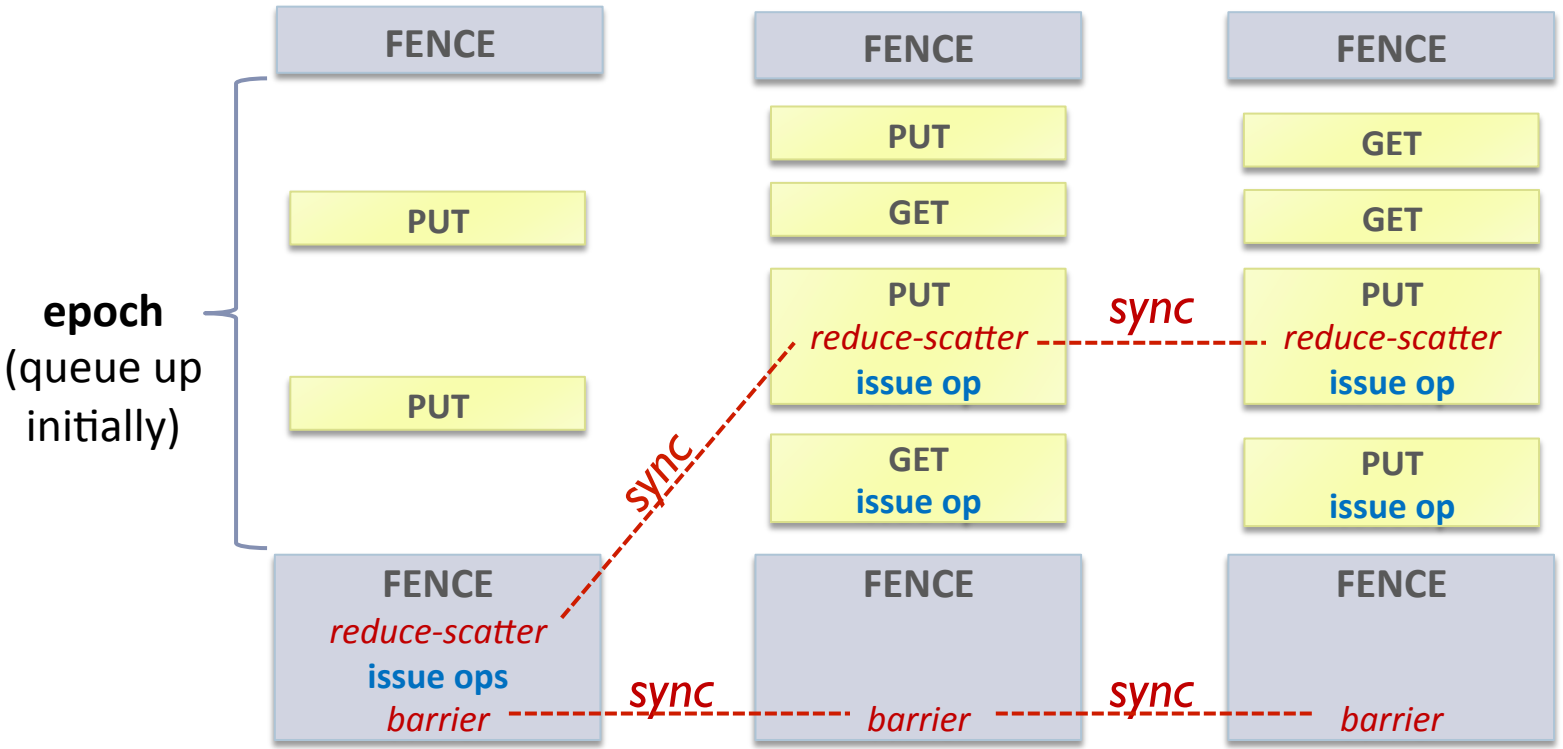
Current implementation in MPICH2

- *Lazy* approach
  - enqueue operations and issue at end; use reducescatter to count number of RMA operations arriving
  - one synchronization



Our strategy

- *Adaptive* Fence
  - switch from *lazy* to *eager* in between
  - one or two synchronizations



Comparison

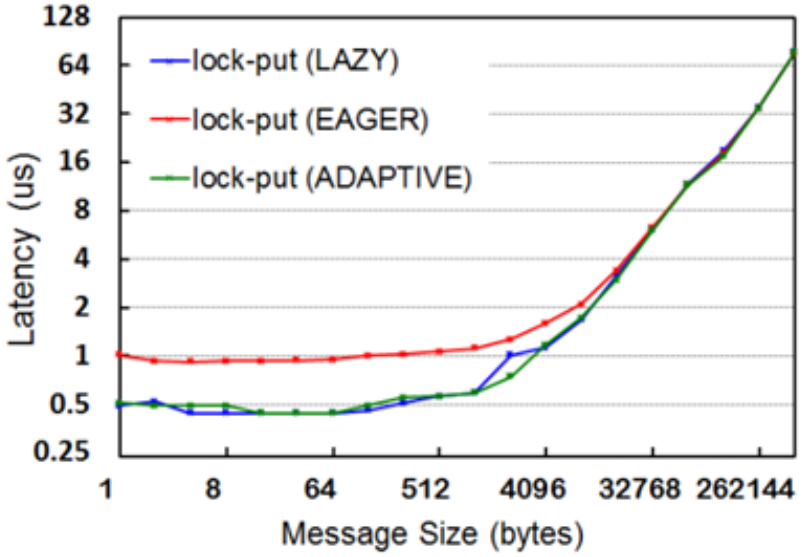
- Lazy approach
  - not synchronize at first, queue up operations and issue them at end
  - less synchronization, but no overlapping opportunity
- Eager approach
  - issue operations as they occur, synchronize at first and end
  - overlapping opportunity, no queuing cost, but more synchronization
- Adaptive approach
  - perform as lazy initially, synchronize and switch to eager if meet large operations/data
  - combines features of both lazy and eager

Evaluation

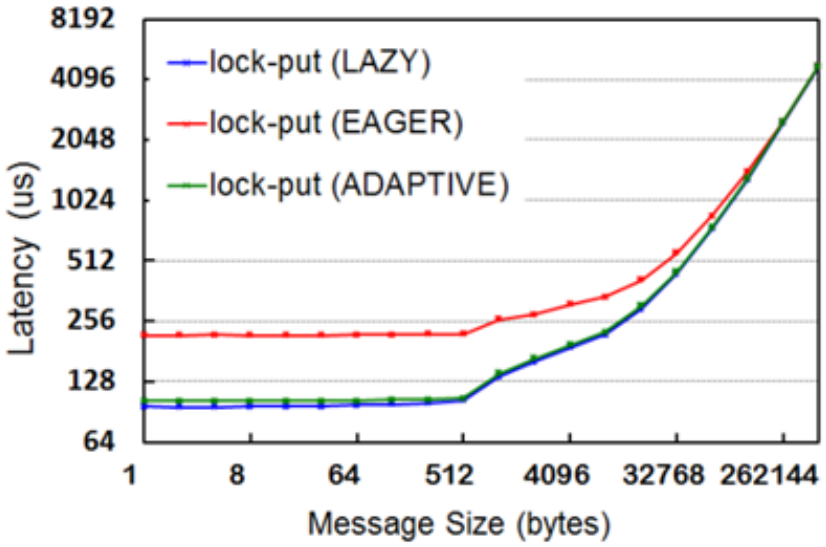
Experimental Setup

- Platforms
  - SMP machine with 4 cores and 8GB memory
  - "breadboard" cluster at ANL, each node has two quad-core processors and 16GB memory, Ethernet interconnect
- Benchmarks
  - Ping-pong latency
  - overlap percentage
  - Graph500, halo exchange
- Comparison
  - *eager / lazy / adaptive*
- Switching threshold
  - $h\downarrow op$ = 10000 or $h\downarrow msg\_sz$ = 400 bytes

Single Operation Latency



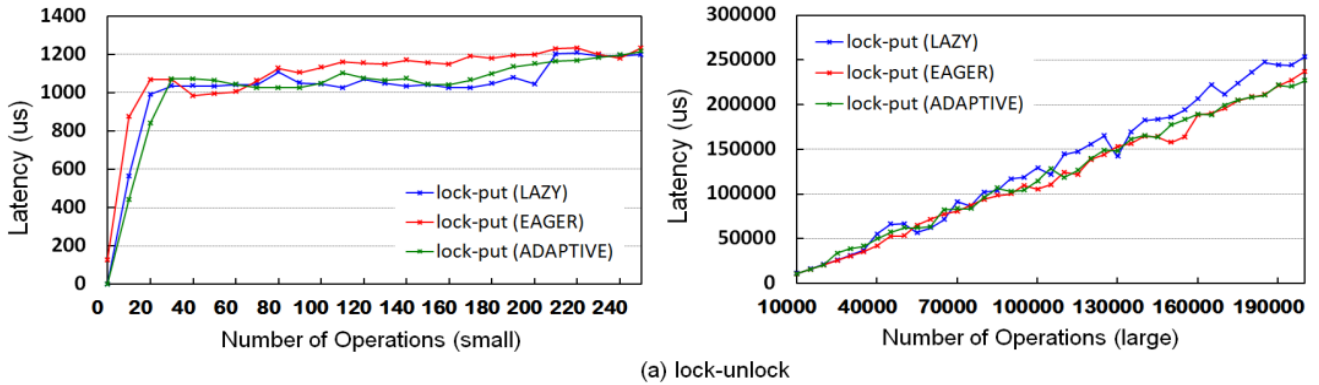
SMP
(Lock-Unlock)



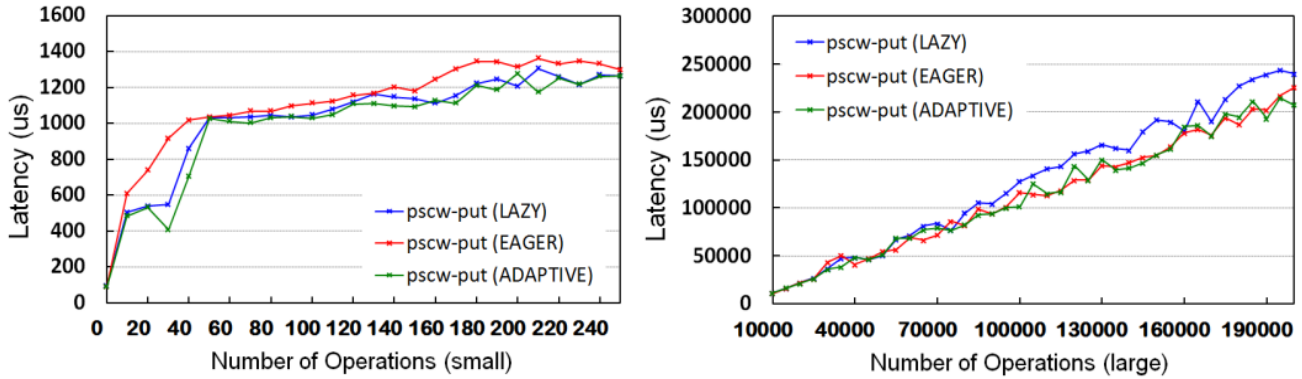
Breadboard
(Lock-Unlock)

- *lazy/adaptive* are better at small message size, due to the optimization for single short operation

Multiple Operations Latency



(a) lock-unlock

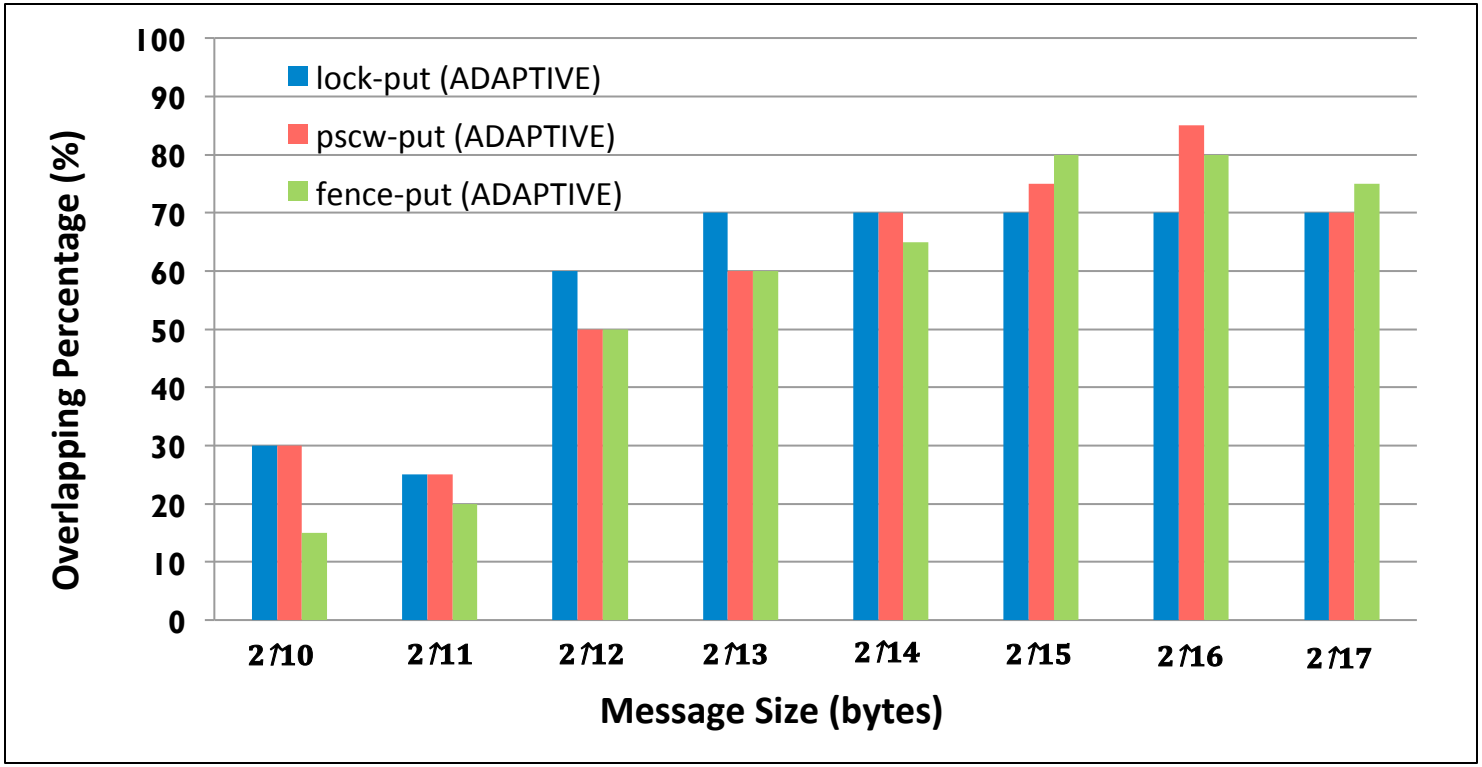


(b) PSCW

breadboard
(Lock-Unlock and PSCW)

- for small operations, *lazy/adaptive* are better; for large operations, *eager/adaptive* are better

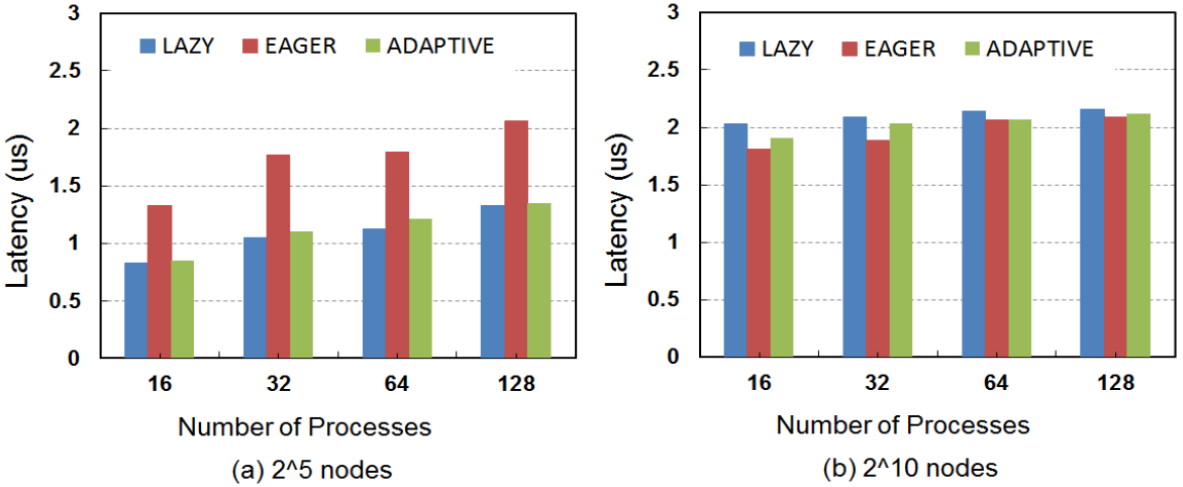
Overlap Percentage



adaptive approach, breadboard

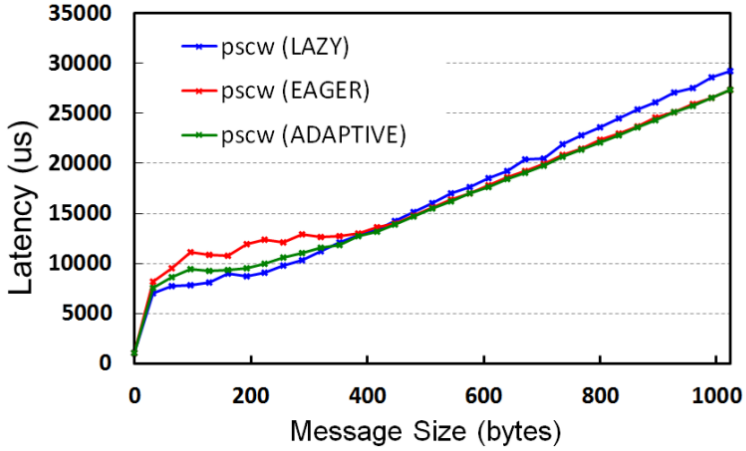
- *eager* has similar results, while *lazy* has no overlapping observed

Performance Benchmarks



adaptive fence

Graph 500, breadboard



adaptive PSCW

Halo exchange, breadboard

Conclusion and Future Work

- Conclusion
  - *Lazy approach* has less synchronization cost and provides opportunities to aggregate or schedule operations
  - *Eager approach* issues operations early, eliminates cost of queuing, and enables the overlap of communication and computation
  - *Adaptive approach* combines features of *lazy* and *eager*, introducing a modest overhead
- Future Work
  - Experiments on other underlying transports (RDMA on InfiniBand, Gemini, etc.) and on large-scale systems
  - Use nonblocking collectives (active target) or nonblocking communication (passive) to continue while performing first sync
  - Support adaptive approach in new synchronization options in MPI-3
    - Fortunately, they are for passive target, where the extensions are natural

Thanks!