High Performance MPI-2 One-Sided Communication over InfiniBand

W. Jiang J. Liu H. Jin D. K. Panda
W. Gropp R. Thakur

The Ohio State University
Argonne National Laboratory

Presentation Outline

- Introduction
- Background
- Current Send/Receive-Based Design
- Proposed RDMA-Based Design
- Experimental Results
- Conclusions & Future Work

Introduction

- MPI-2
  - One-Sided Communication
  - Process Management
  - MPI-I/O
- One-Sided Communication
  - Send/Receive-Based Implementation
    - High communication overhead
    - Dependency between communication progress and remote process

Motivation

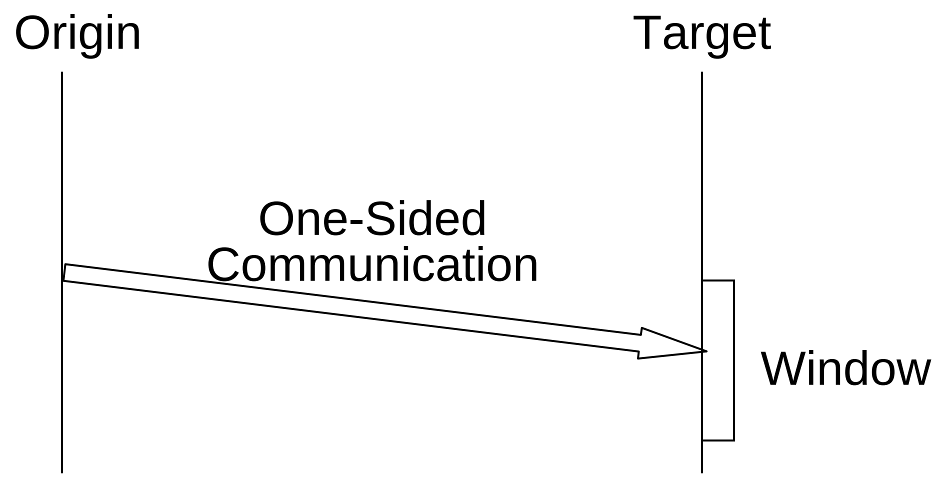
- InfiniBand provides Remote Direct Memory Access (RDMA) operations

- How can we design efficient and scalable MPI-2 one-sided communication by taking advantage of InfiniBand RDMA operations?

Presentation Outline

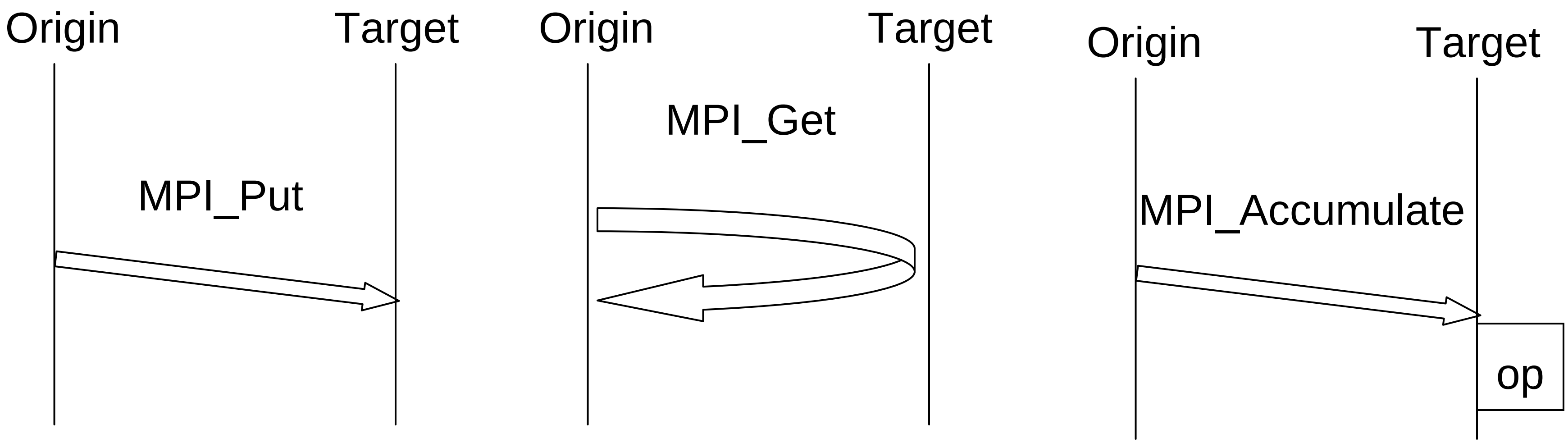
- Introduction
- Background
  - MPI-2 One-Sided Communication
  - InfiniBand
- Current Send/Receive-Based Design
- Proposed RDMA-Based Design
- Experimental Results
- Conclusions & Future Work

MPI-2 One-Sided Communication



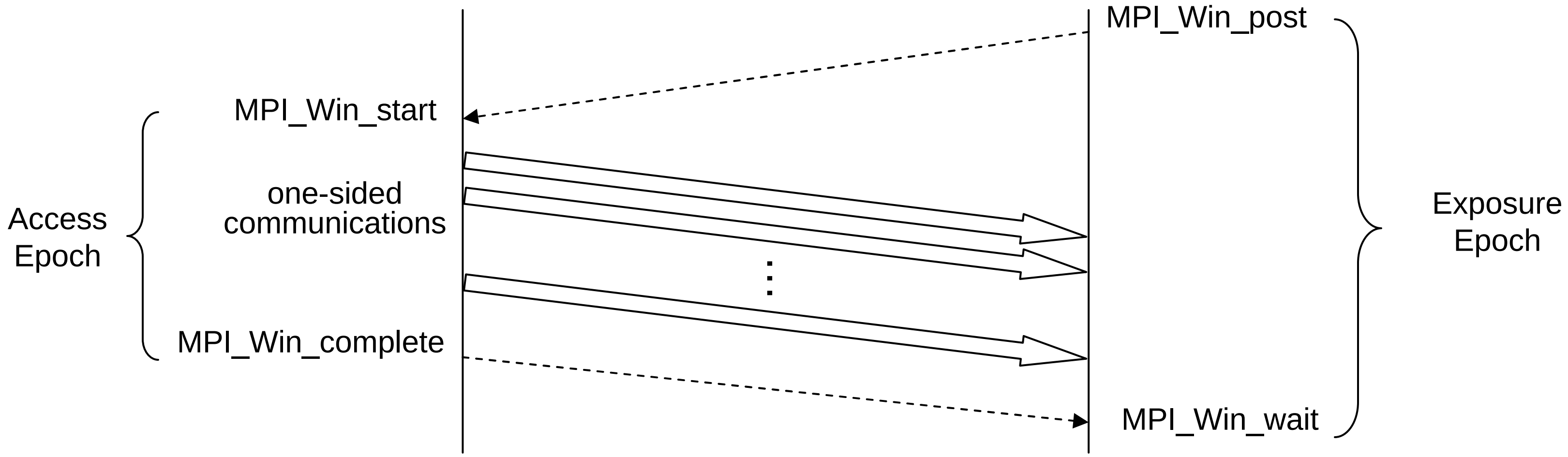
- A process can access another process's memory address space directly
  - **Origin**
  - **Target**
  - **Window**

MPI-2 One-Sided Communication



- Communication functions
  - MPI_Put
  - MPI_Get
  - MPI_Accumulate

MPI-2 One-Sided Communication



- Synchronization functions
  - Active, involves both sides
  - Passive, involves the origin side
- Epochs
  - Access Epoch   MPI_Win_start ~ MPI_Win_complete
  - Exposure Epoch   MPI_Win_post ~ MPI_Win_wait

InfiniBand

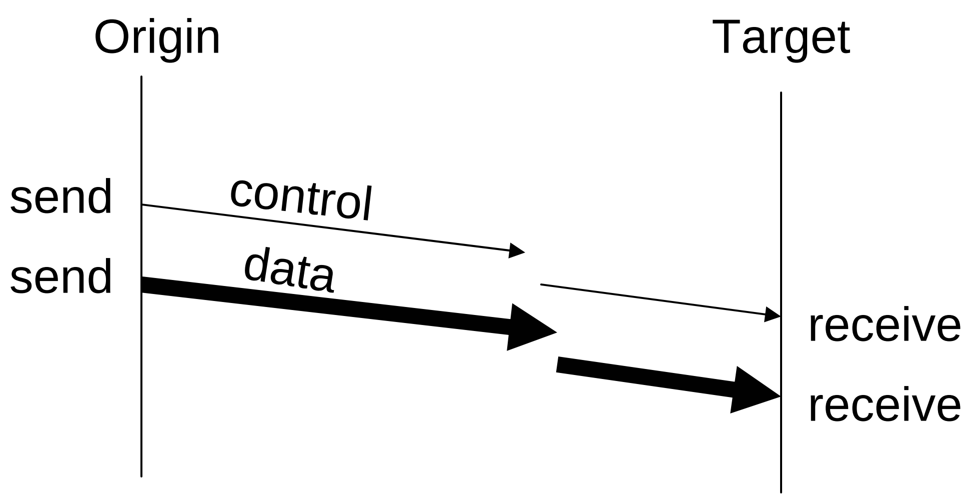
- Open industry standard
- Provides high performance communication (5 us, 10Gbps)
- Advanced features
  - Remote Direct Memory Access (RDMA)
    - RDMA write
    - RDMA read
  - Atomic operations, Multicast, etc.

Presentation Outline

- Introduction
- Background
- Current Send/Receive-Based Design
- Proposed RDMA-Based Design
- Experimental Results
- Conclusions & Future Work

Send/Receive-Based Design

- MPI_Put:
  - Origin:
    - Control message
    - Data message
  - Target:
    - Receive the control message
    - Receive the data
- MPI_Get and MPI_Accumulate are implemented similarly



Performance Issues in Send/Receive-Based Design

- Protocol overhead
  - Handshake in Rendezvous protocol
  - Matching between send and receive functions
    - Unexpected/expected message queue maintenance
    - Tag matching
    - Flow control
- Heavy dependency on the target to make progress
  - Process skew
  - Poor computation/communication overlapping
- Target is actively involved
  - Performance bottleneck

Presentation Outline

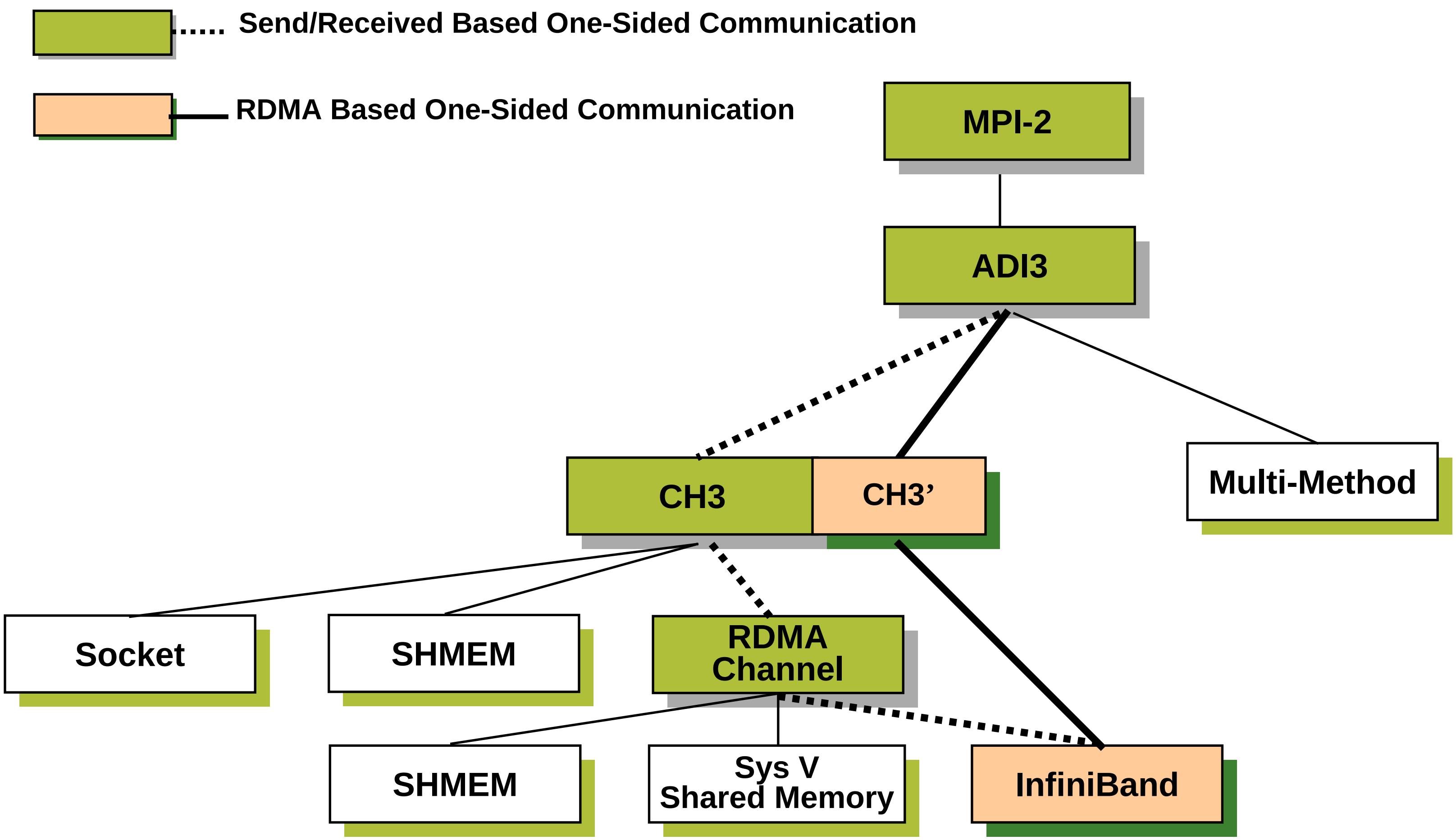
- Introduction
- Background
- Current Send/Receive-Based Design
- Proposed RDMA-Based Design
- Experimental Results
- Conclusions & Future Work

Basic Idea of RDMA-Based Design

- The semantic of InfiniBand RDMA operations is similar to that of MPI-2 one-sided communication.

- **We map MPI-2 one-sided functions directly to InfiniBand RDMA operations.**

Implementation on MPICH2



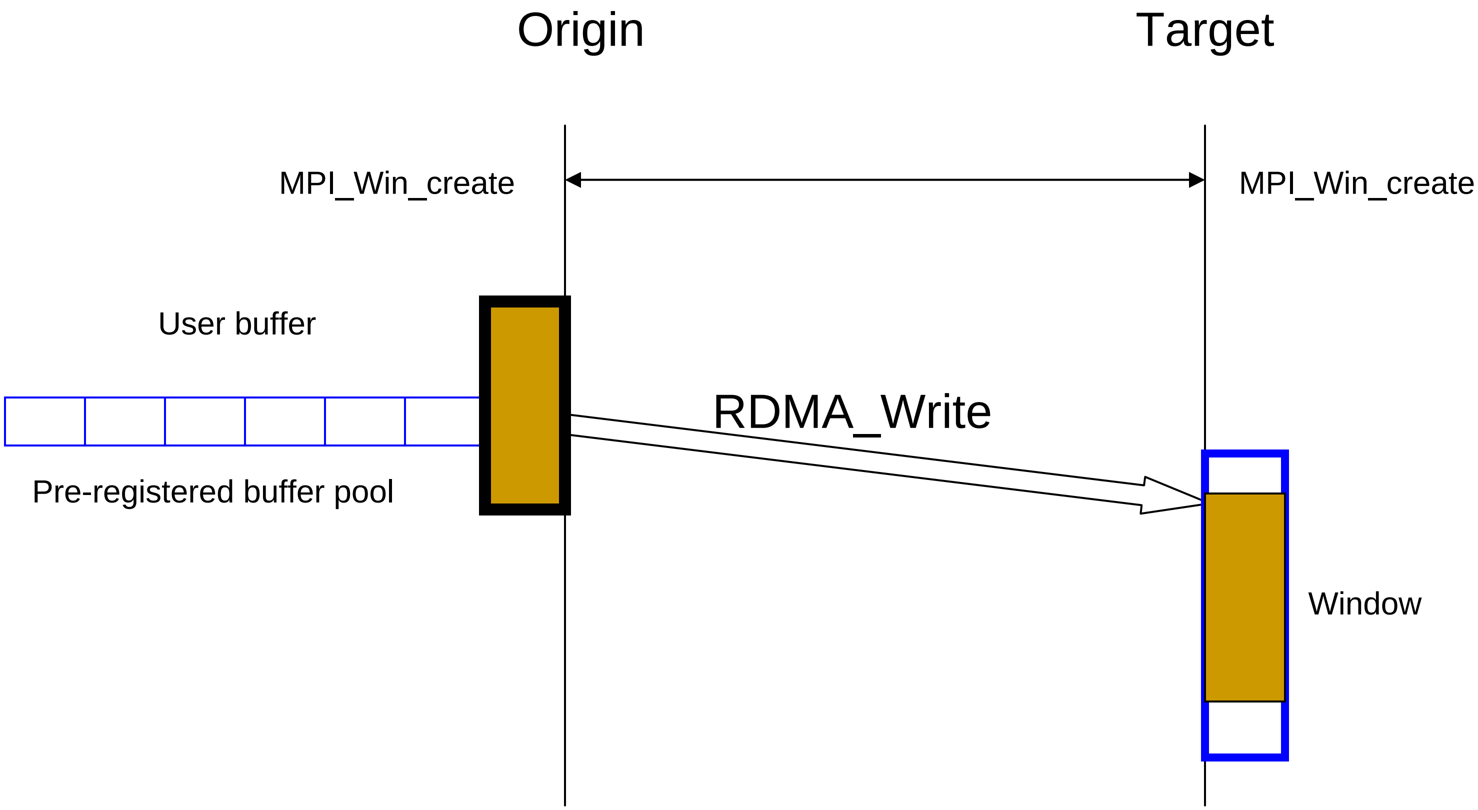
Mapping One-Sided Communication to RDMA

- MPI_Put:
  - RDMA write
- MPI_Get:
  - RDMA read
- MPI_Accumulate:
  - RDMA read/write
  - Atomic operation

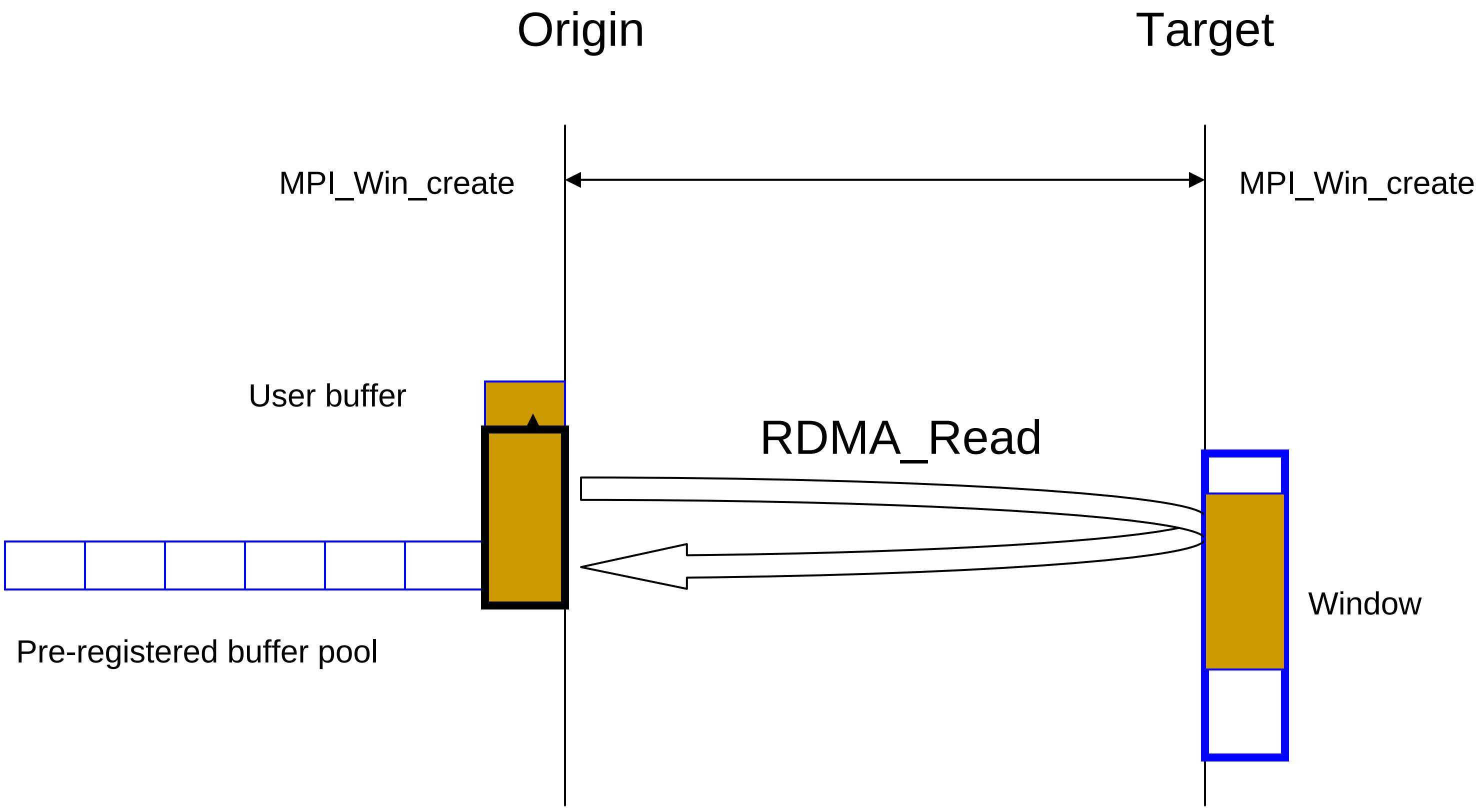
Memory registration

- RDMA need registration – Source and destination memories
- Registration is expensive
- Destination memory during window creation phase
- Source memory
  - Small message
    - Pre-registered buffer pool
  - Large message
    - Pin-down cache

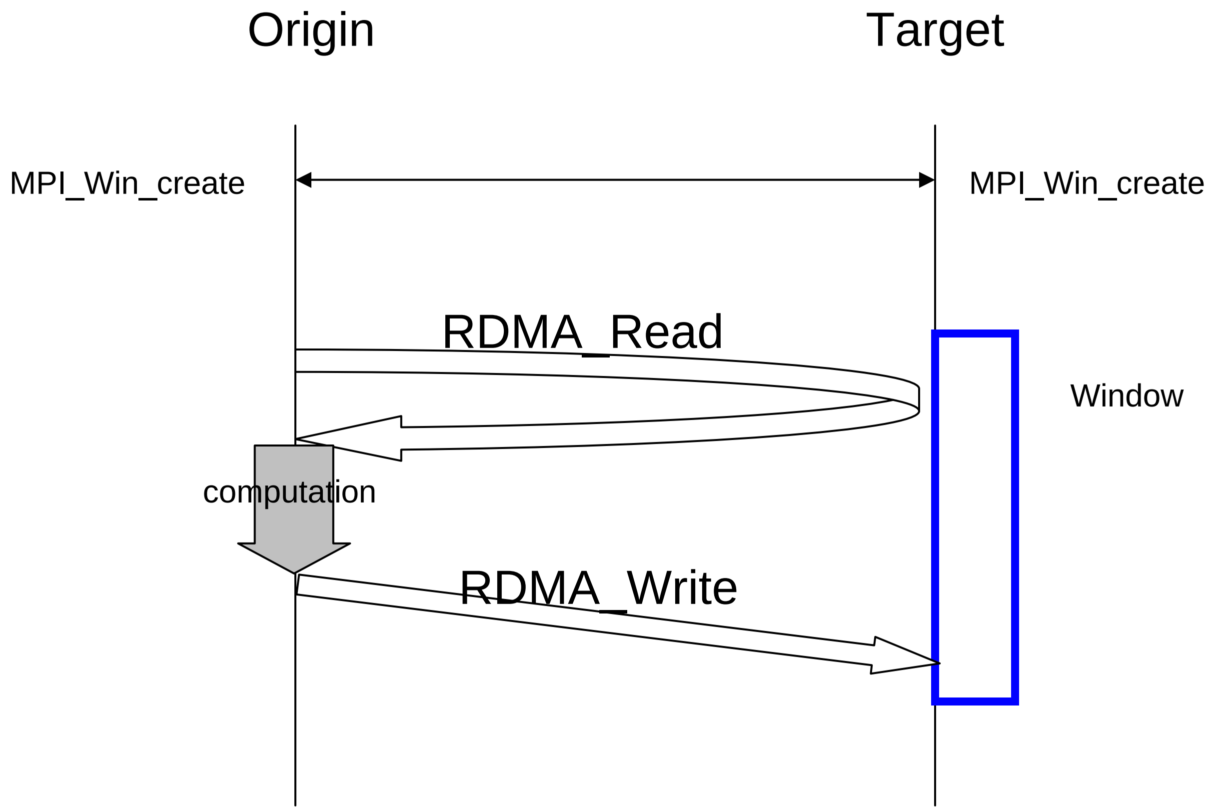
Mapping MPI_Put to RDMA_Write



Mapping MPI_Get to RDMA_Read



Mapping MPI_Accumulate to RDMA operations



Advantages of RDMA-Based Design

- Avoid protocol overhead of two-sided communication.
  - Avoid rendezvous protocol
  - No matching between send and receive functions
- Do not involve the remote process
  - Independent communication progress
    - Suffer much less from process skew
    - Better communication/computation overlapping
  - Target will not be the bottleneck

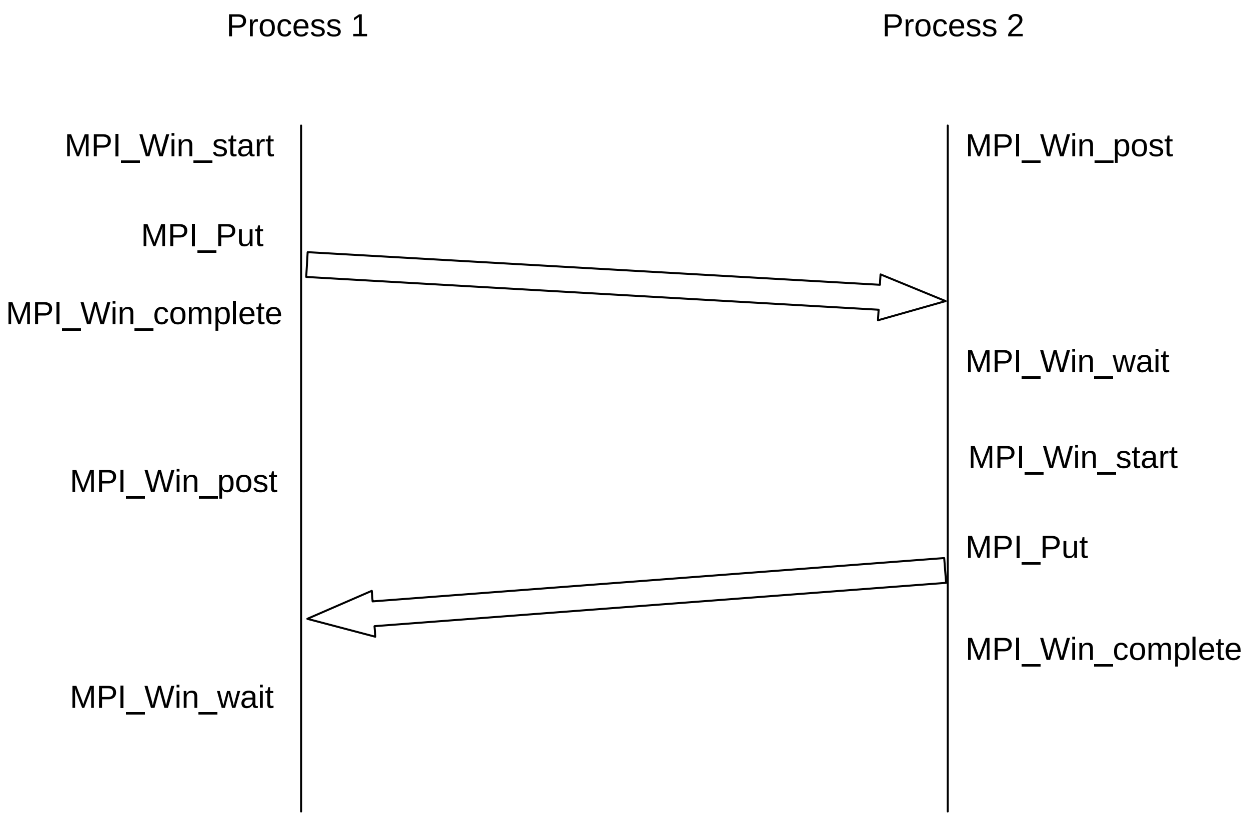
Presentation Outline

- Introduction
- Background
- Current Send/Receive-Based Design
- Proposed RDMA-Based Design
- Experimental Results
  - Ping-pong Test
  - Bi-Directional Test
  - Bandwidth Test
  - Communication/Computation Overlap Test
  - Process Skew Test
  - Scalability Test
- Conclusions & Future Work

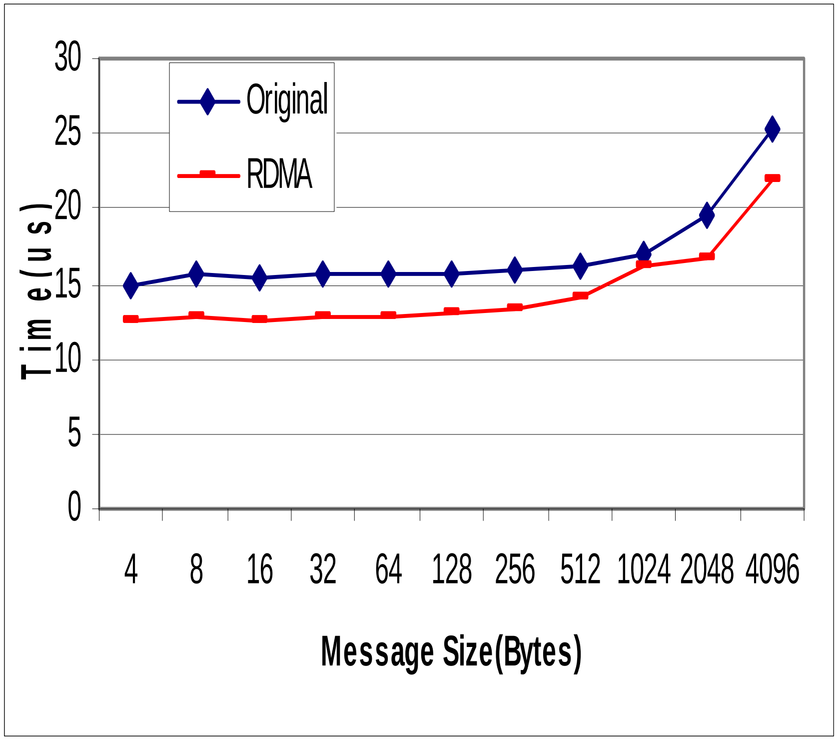
Experimental Testbed

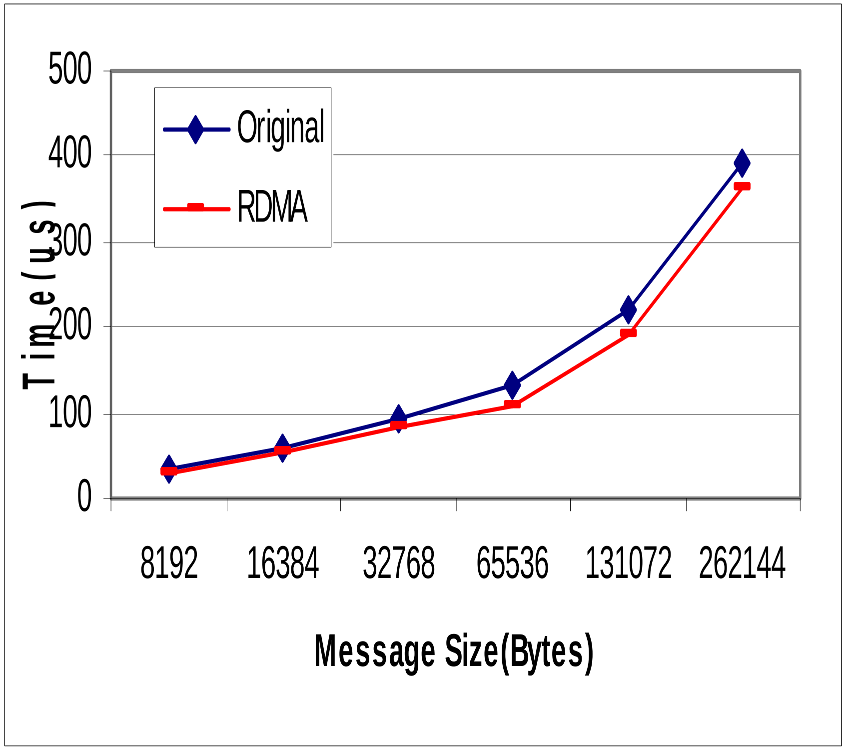
- 8 SuperMicro nodes
  - dual Intel Xeon 2.40 GHz processors
  - PCI-X 64-bit 133MHz interfaces
  - 512K L2 cache and a 400 MHz front side bus
- Mellanox InfiniHost MT23108 DualPort 4X Host Channel Adapter
- InfiniScale MT43132 Eight 4x Port InfiniBand Switch
- Linux Red Hat 7.2 with 2.4.7 kernel, GNU GCC 2.96

Ping-pong Test



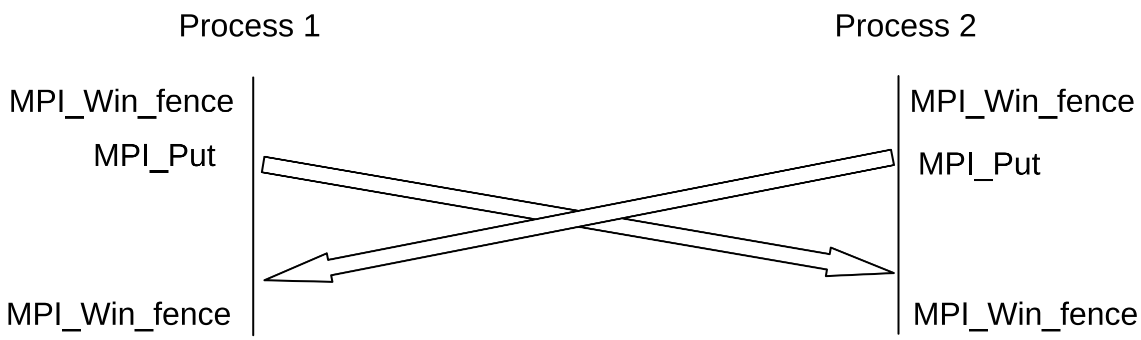
Ping-Pong Latency

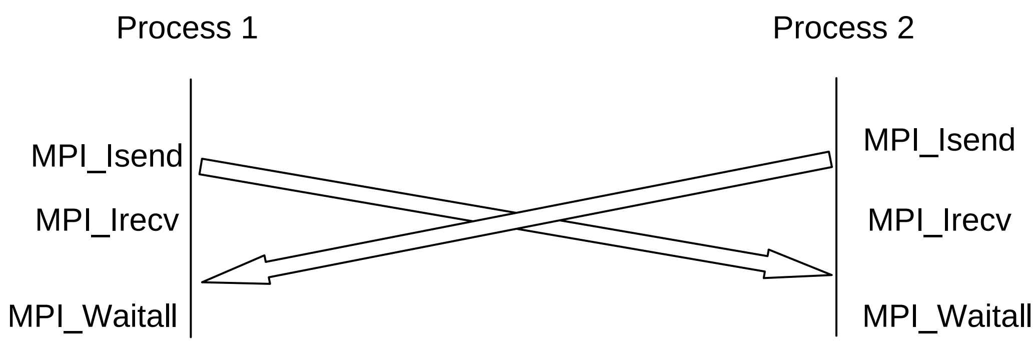




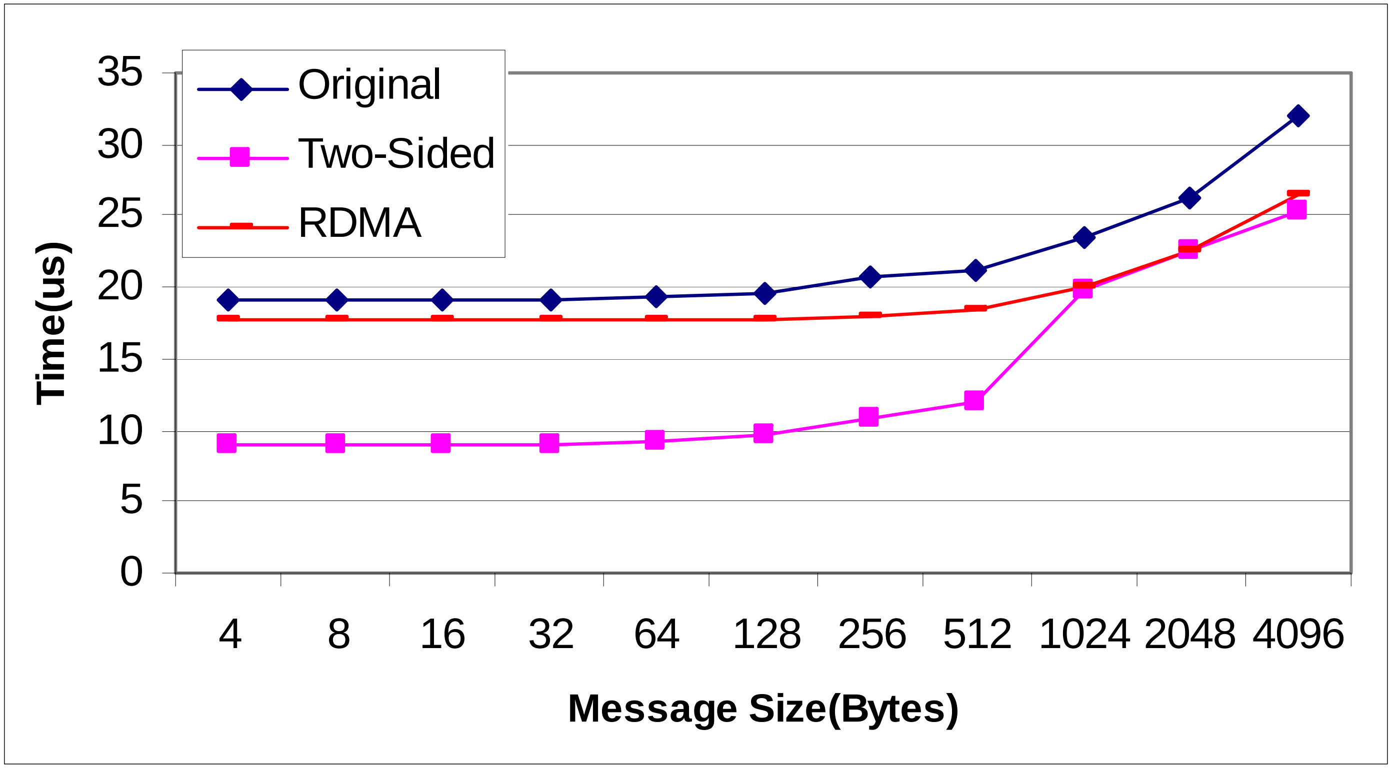
small messages: 15.6 to 12.6 us (19% improvement)
large messages: up to 17 us.

Bi-Directional Test



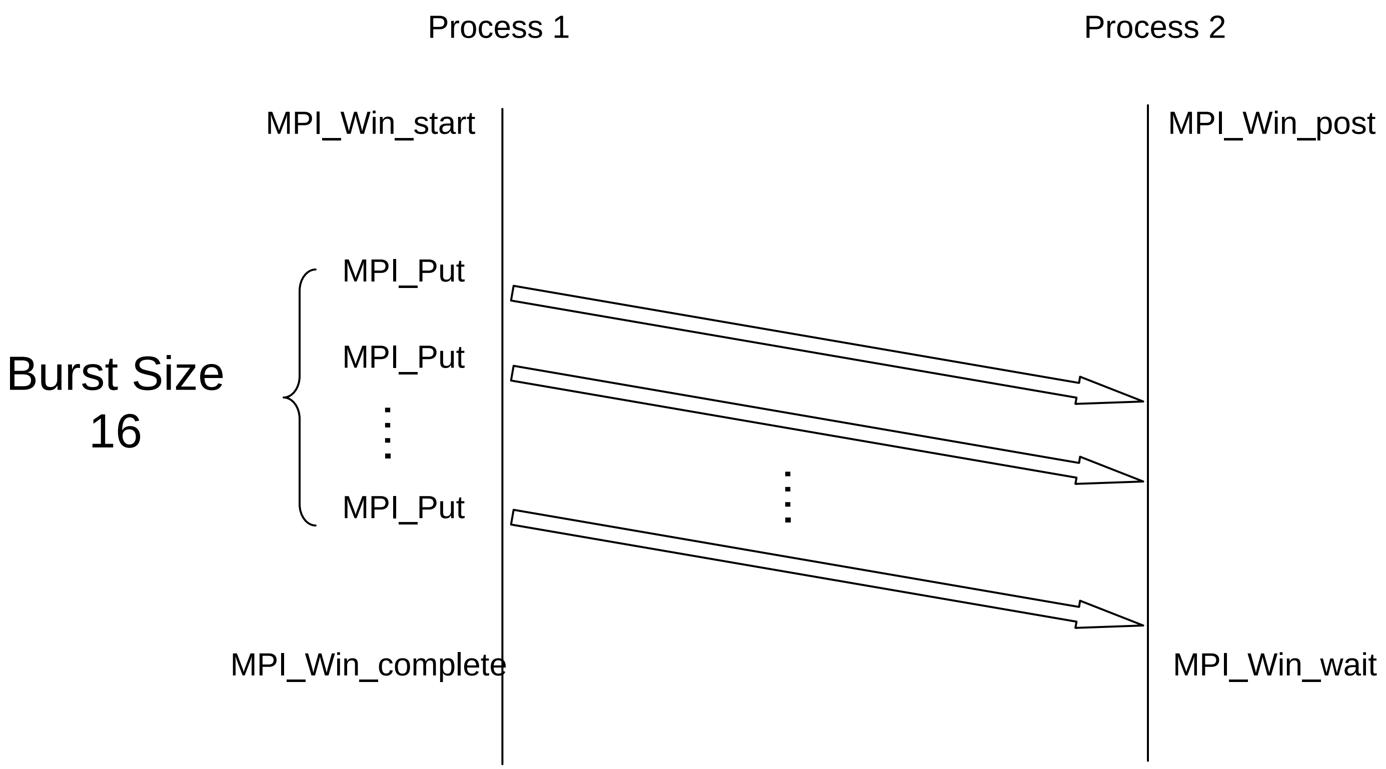


Bi-Directional Latency

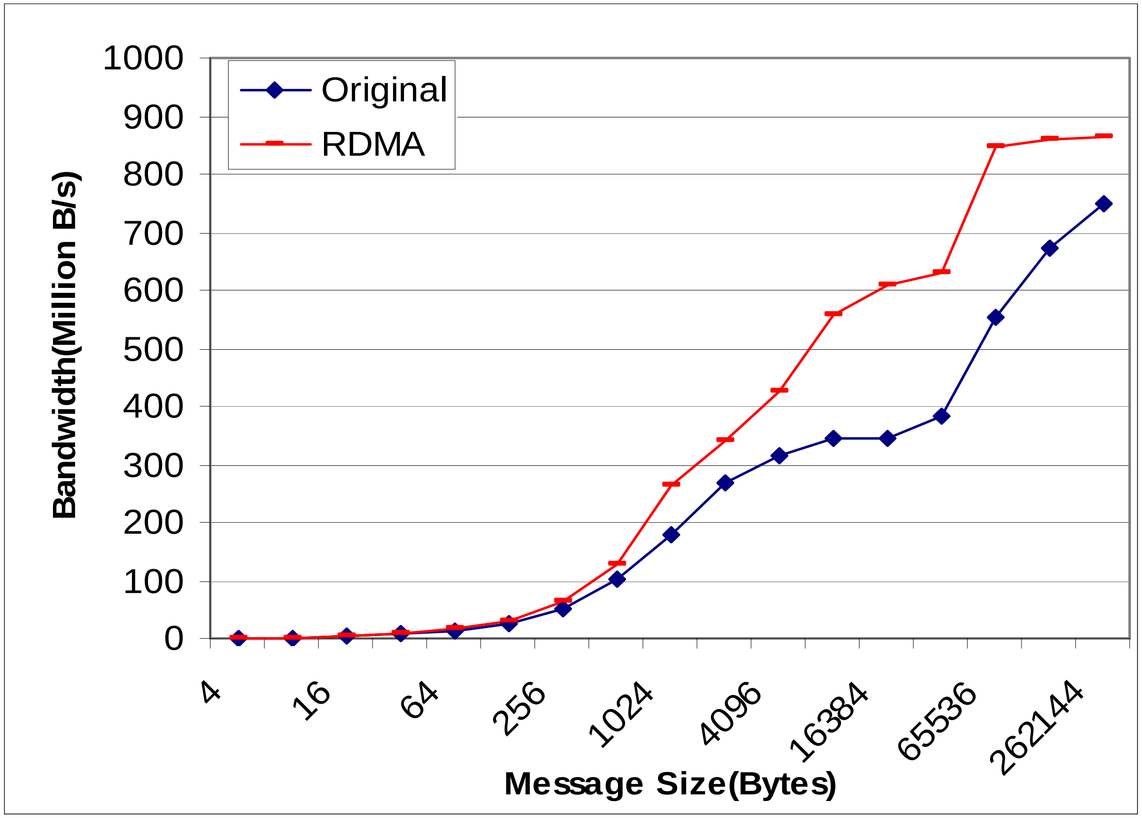


Small messages: two sided > RDMA one-sided > Original one-sided
Large messages: RDMA one-sided > two-sided > Original one-sided

Bandwidth Test

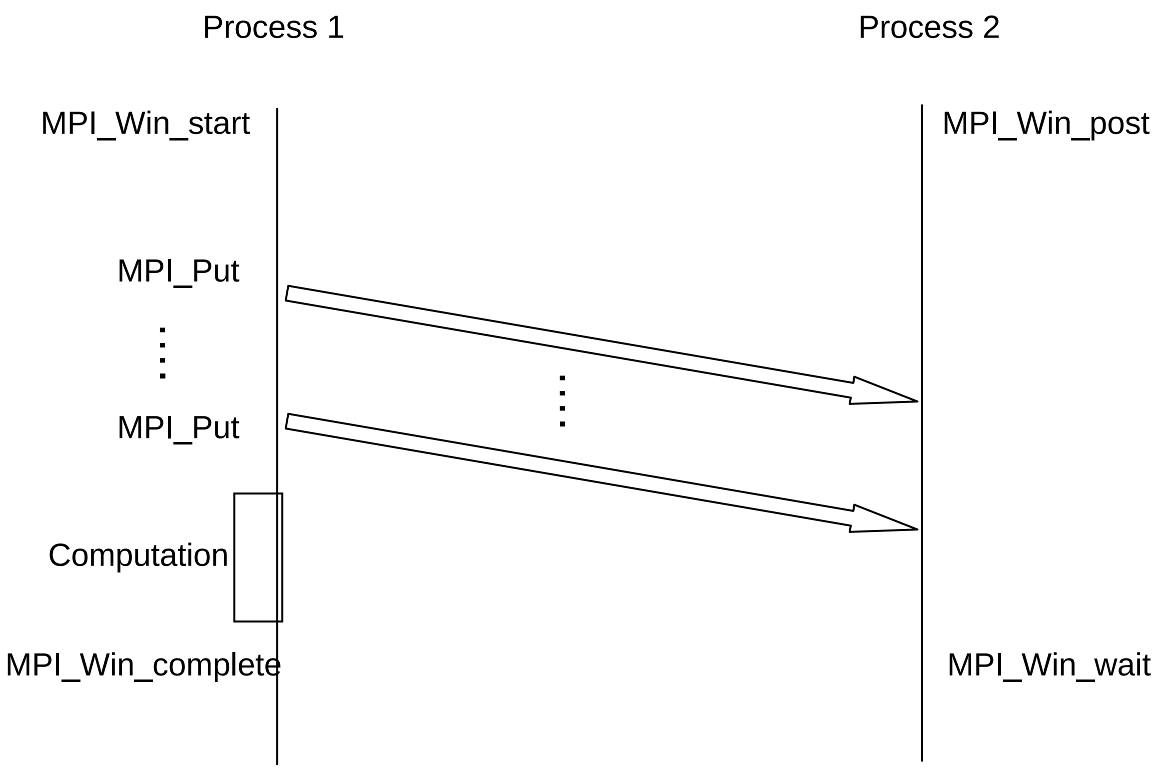


Bandwidth (Put)

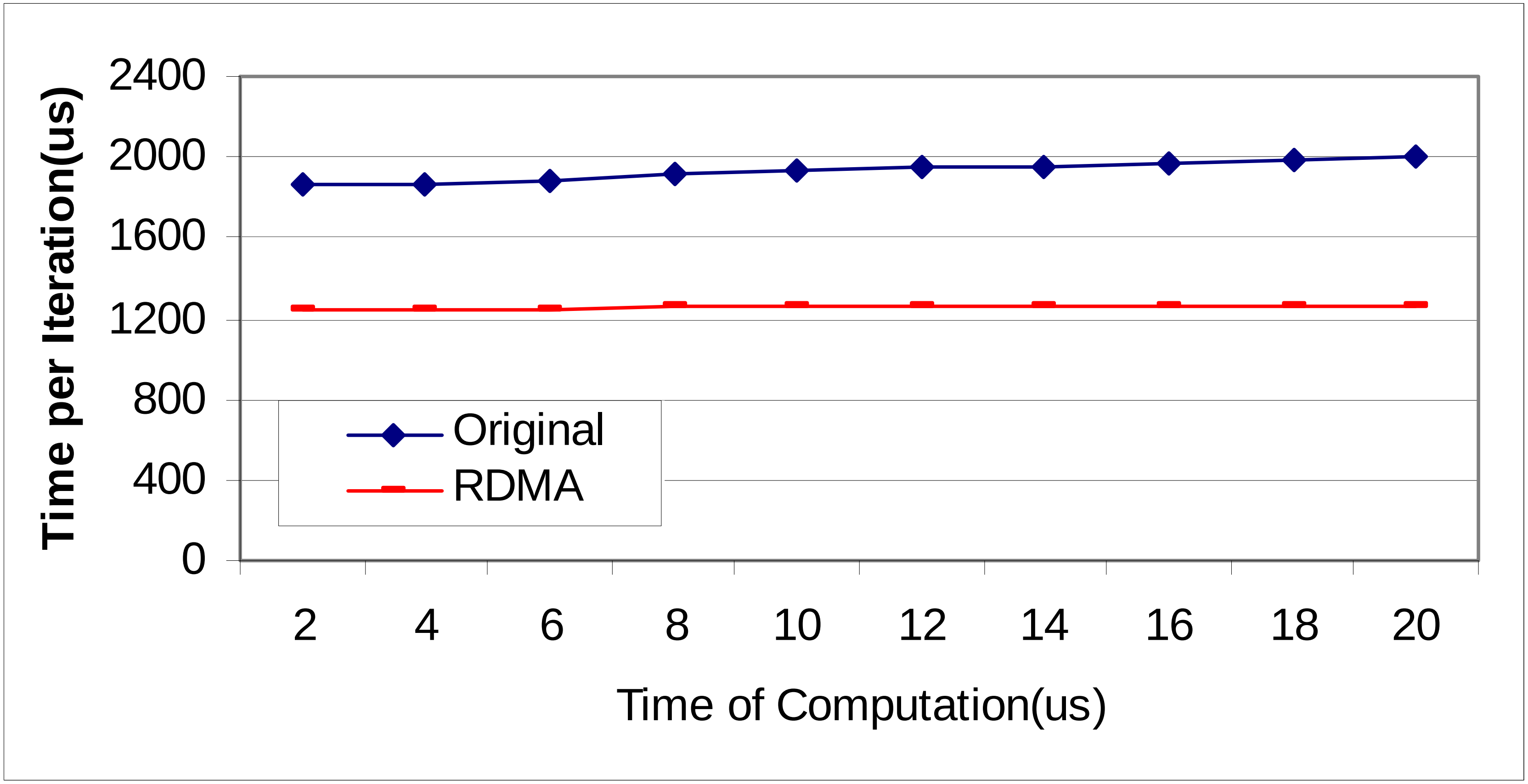


RDMA-Based Implementation: 865MillionB/s
Send/Receive-Based Implementation: 748MillionB/s
For certain message size improvement can be up to 77%

Communication/Computation Overlap Test

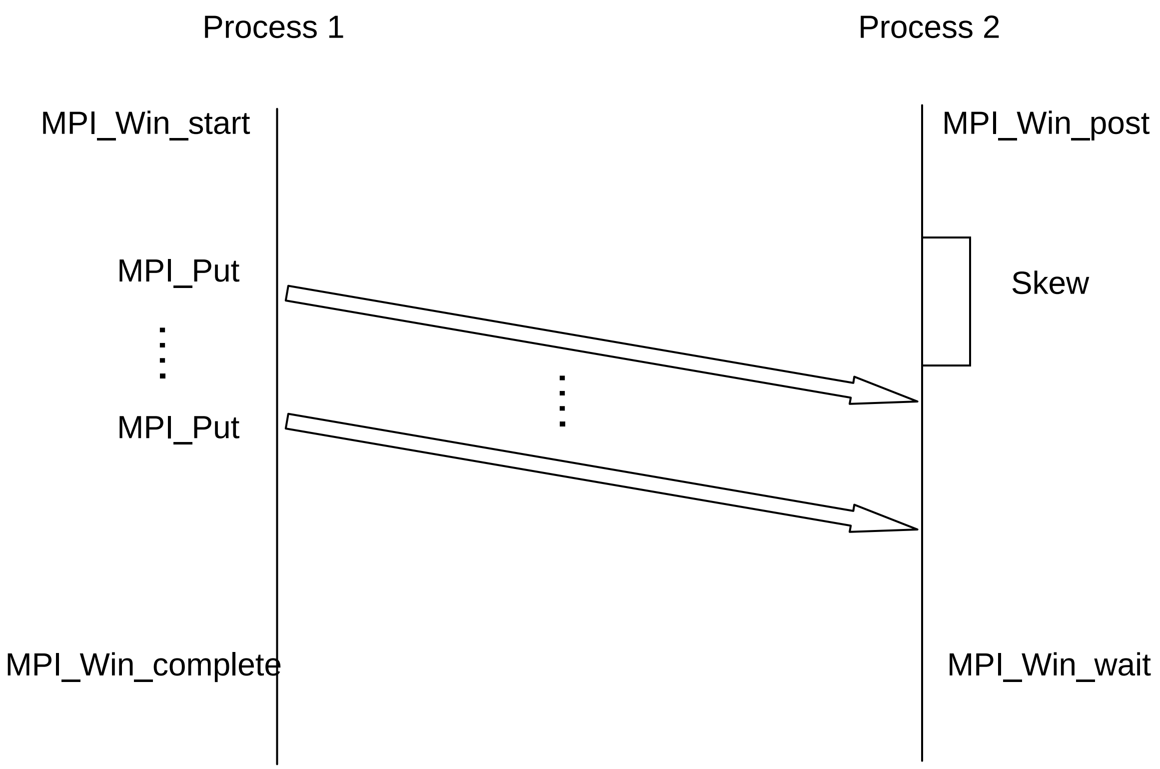


Communication/Computation Overlap

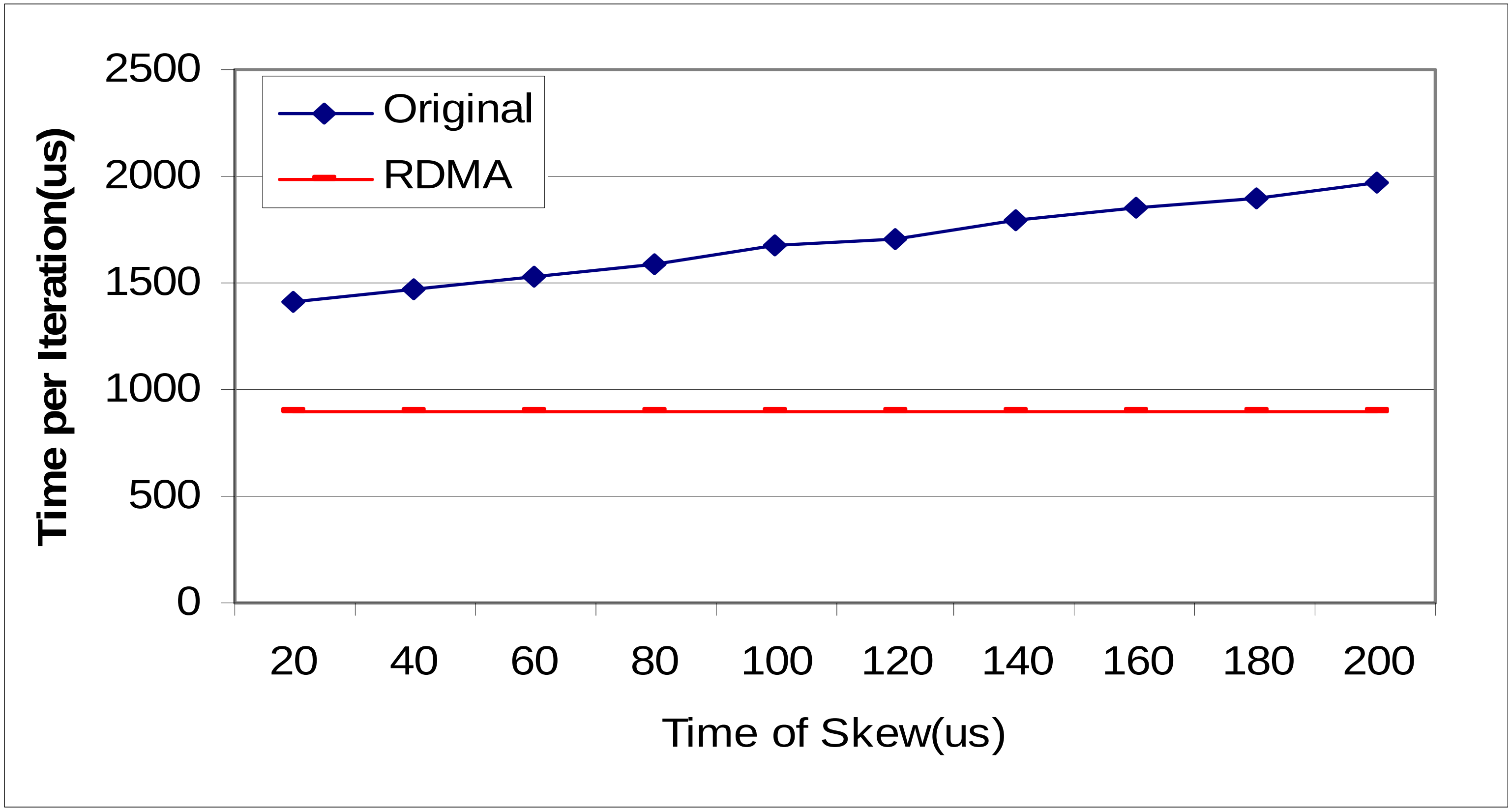


RDMA-Based Implementation: overlaps communication and computation well.
Send/Receive-Based Implementation: shows lower performance when the amount of computation increases.

Process Skew Test

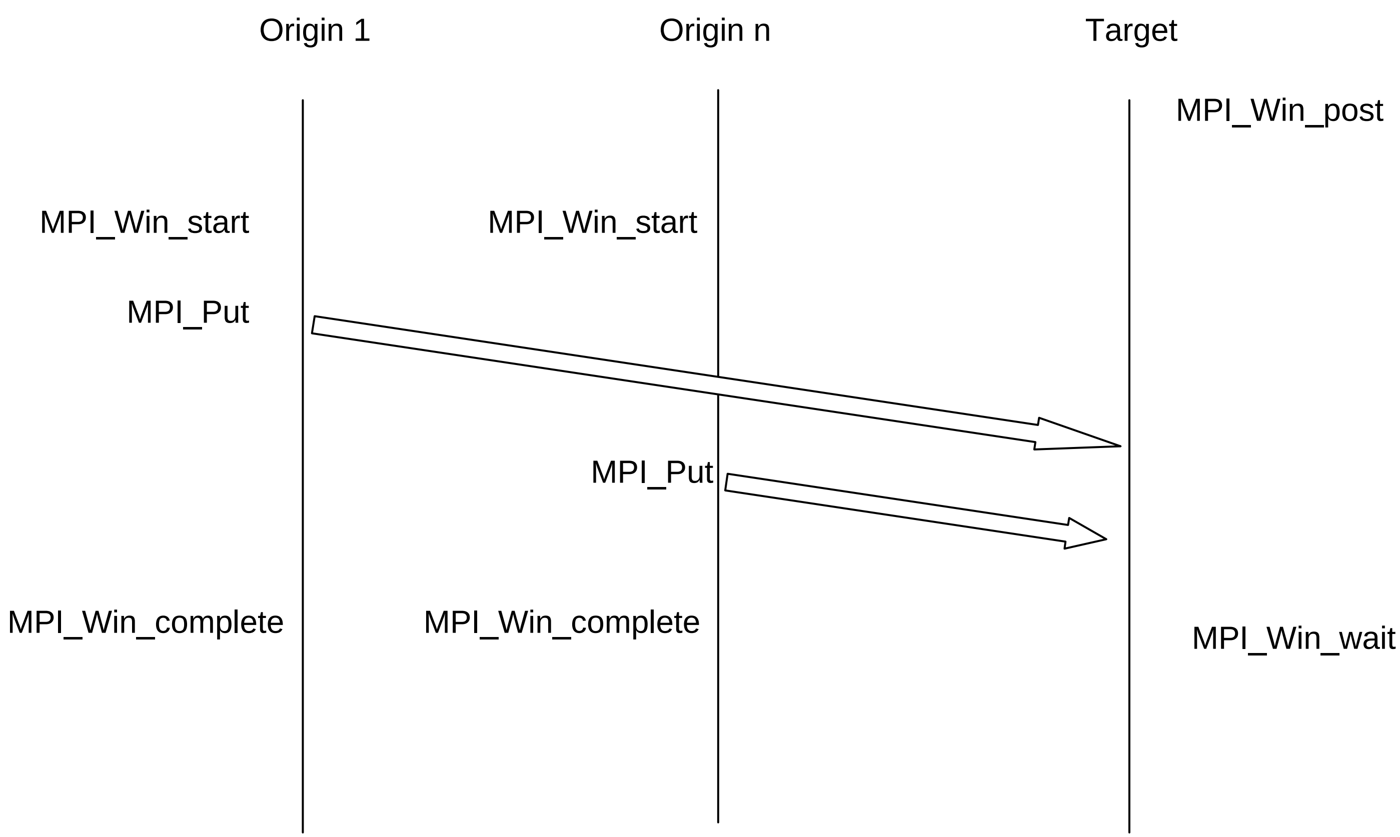


Process Skew

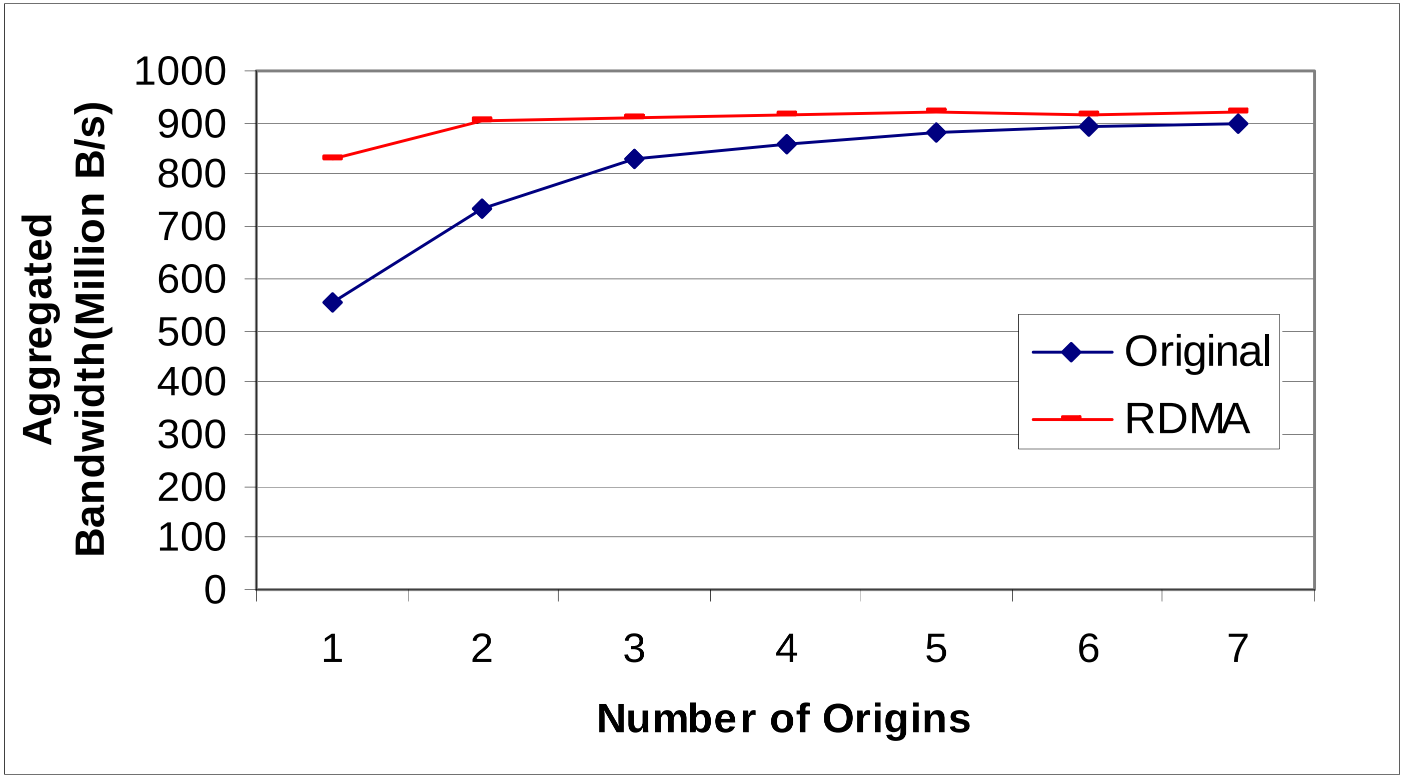


RDMA-Based Implementation: not affected by process skew.
Send/Receive-Based Implementation : shows slower performance with the increase of process skew.

Scalability Test



Performance with Multiple Origin Processes



RDMA-Based Implementation: reaches a peak bandwidth of 920Miliion B/s.
Send/Received-Based Implementation: can only deliver a maximum bandwidth of 895Million B/s.

Conclusions

- RDMA-Based implementation can achieve:
  - Lower overhead and higher communication performance
    - Reduce latency up to 19%
    - Reduce synchronization overhead up to 13%
    - Increase throughput up to 77%
  - Better overlapping between computation and communication
  - Suffer less from process skew
  - Better scalability with multiple origin processes

Future Work

- Passive target one-sided communication
- Non-contiguous data type in one-sided communication

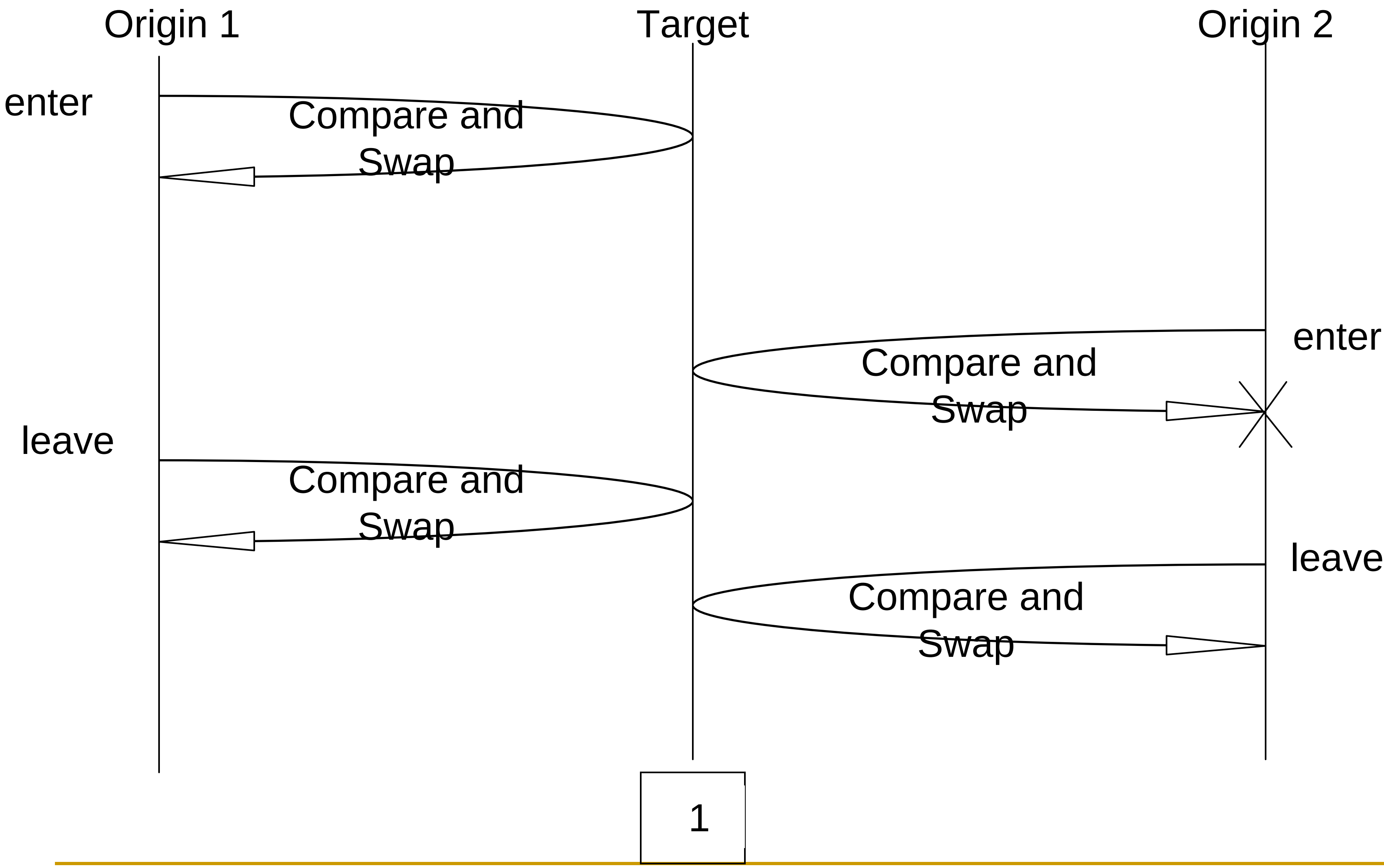
Thank You

NBC home page

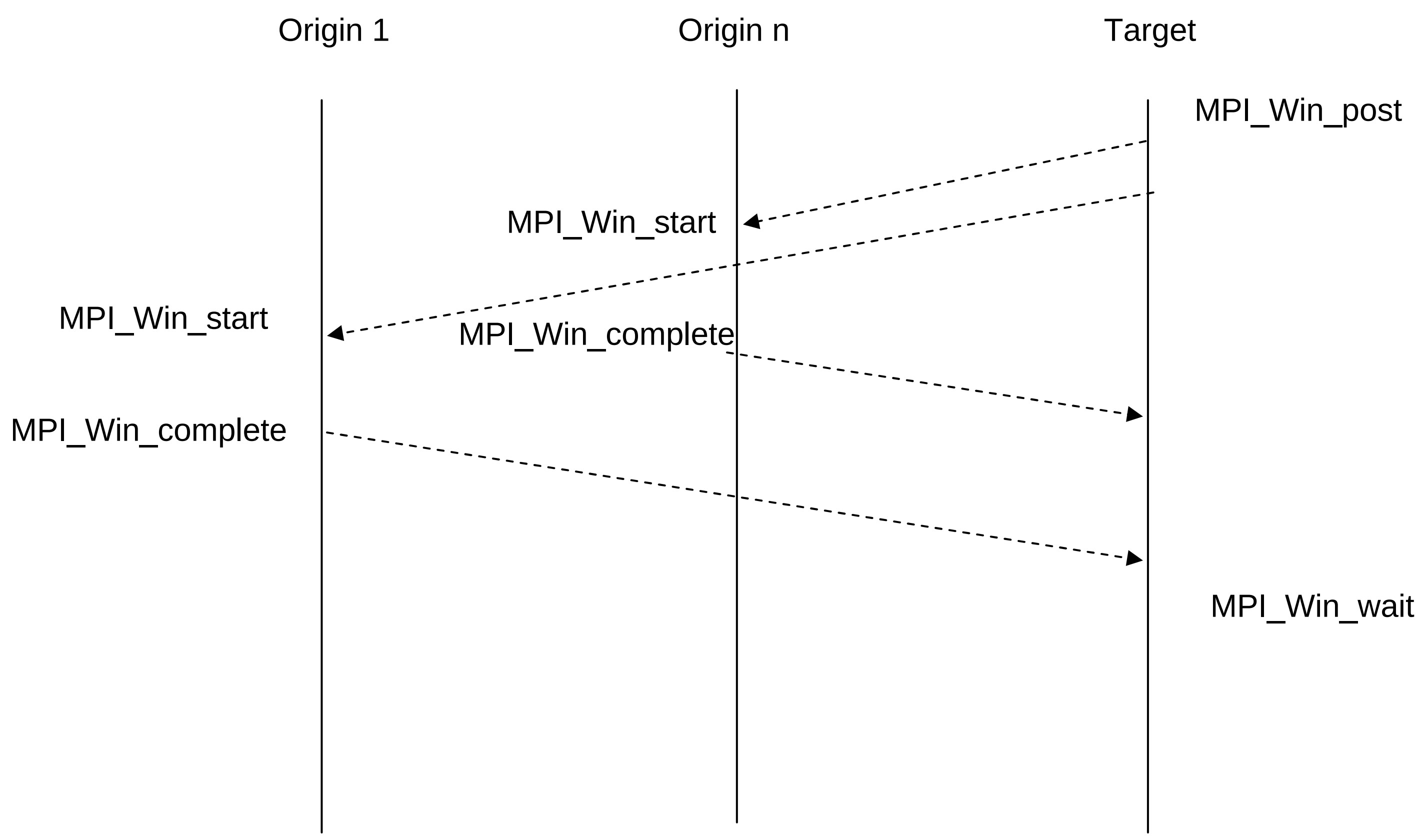
http://nowlab.cis.ohio-state.edu/

E-mail: {jiangw, liuj, jinhy,panda}
@cis.ohio-state.edu

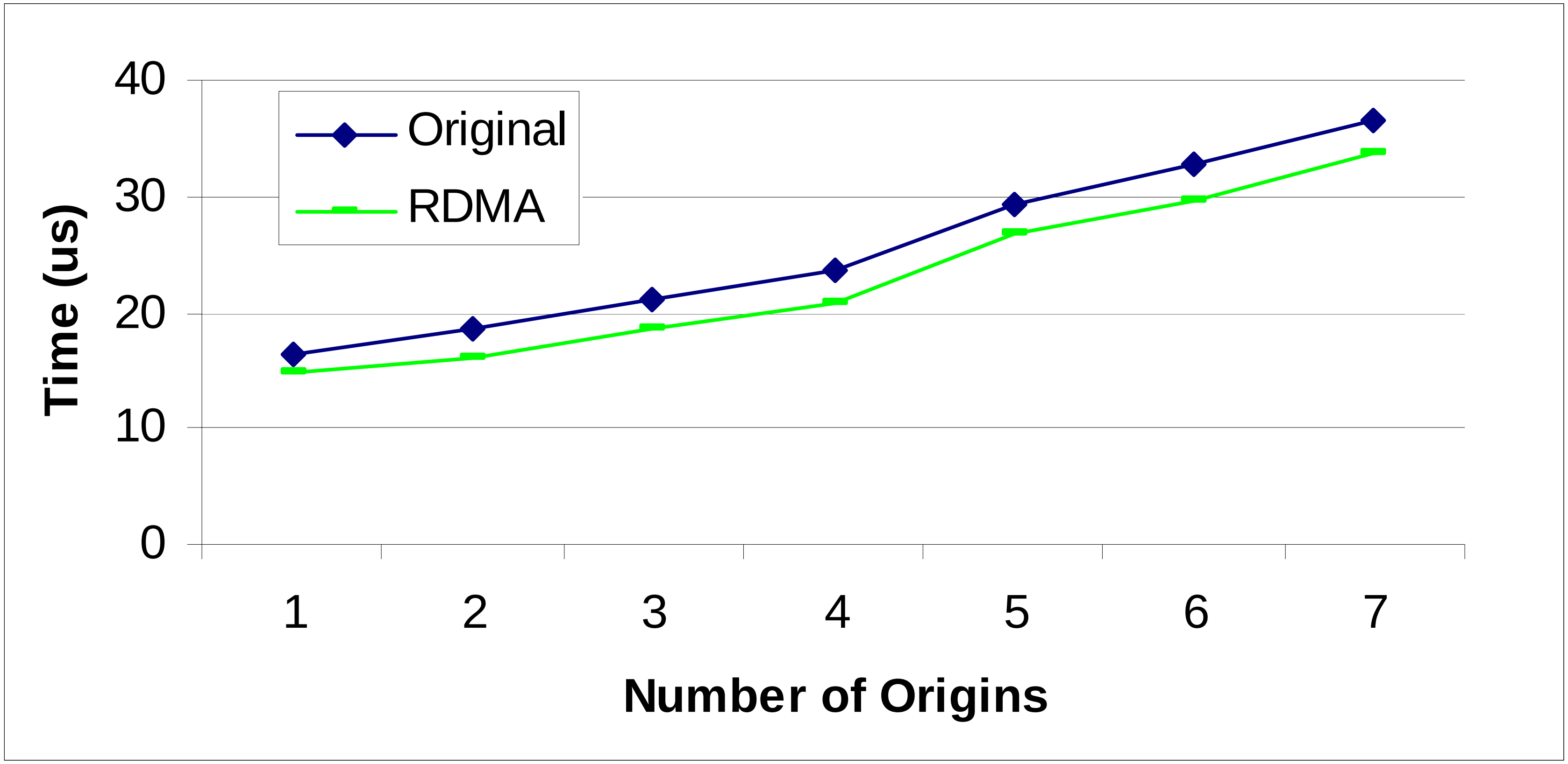
Mutual Exclusion



Synchronization overhead Test

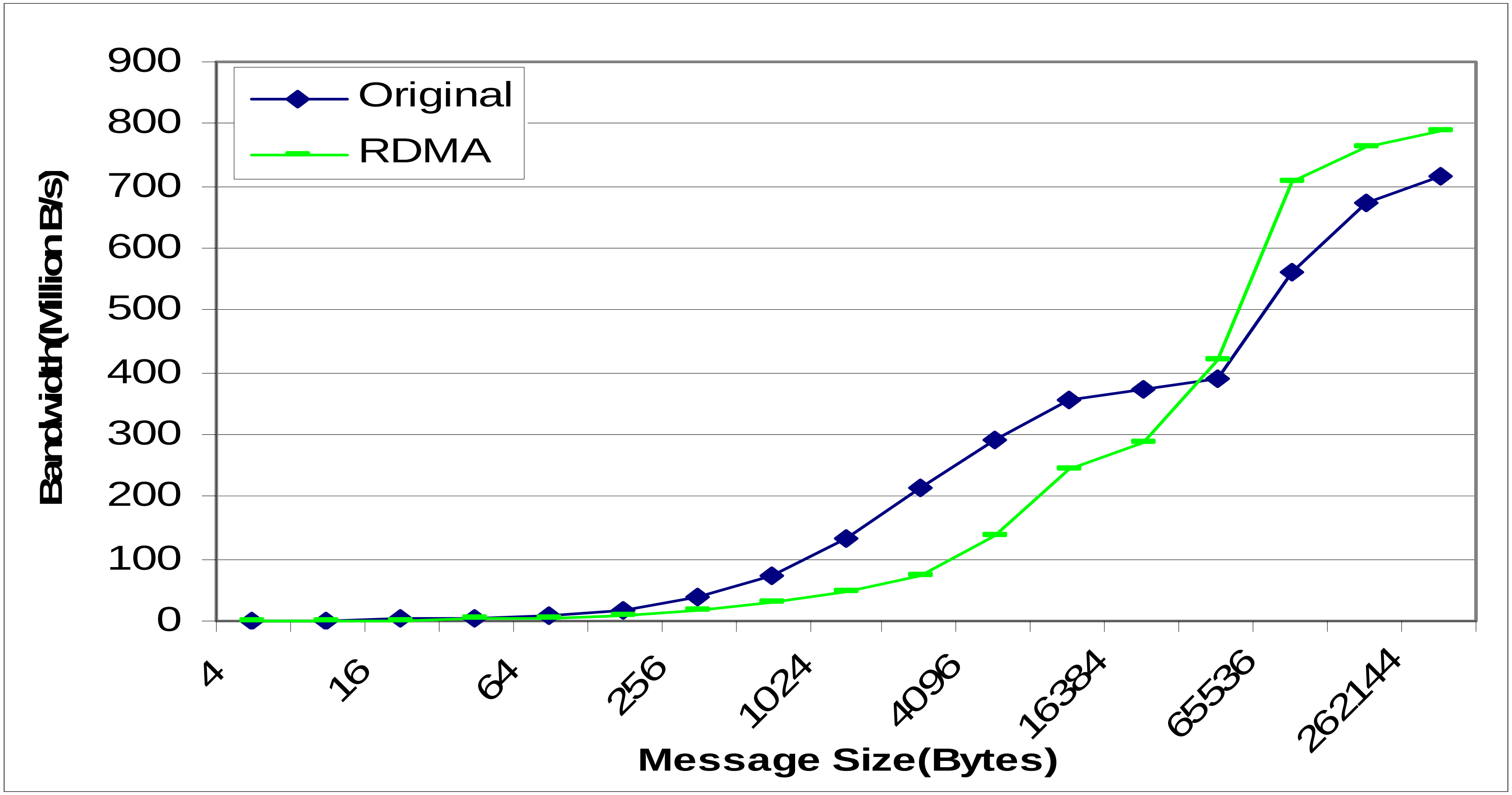


Synchronization overhead

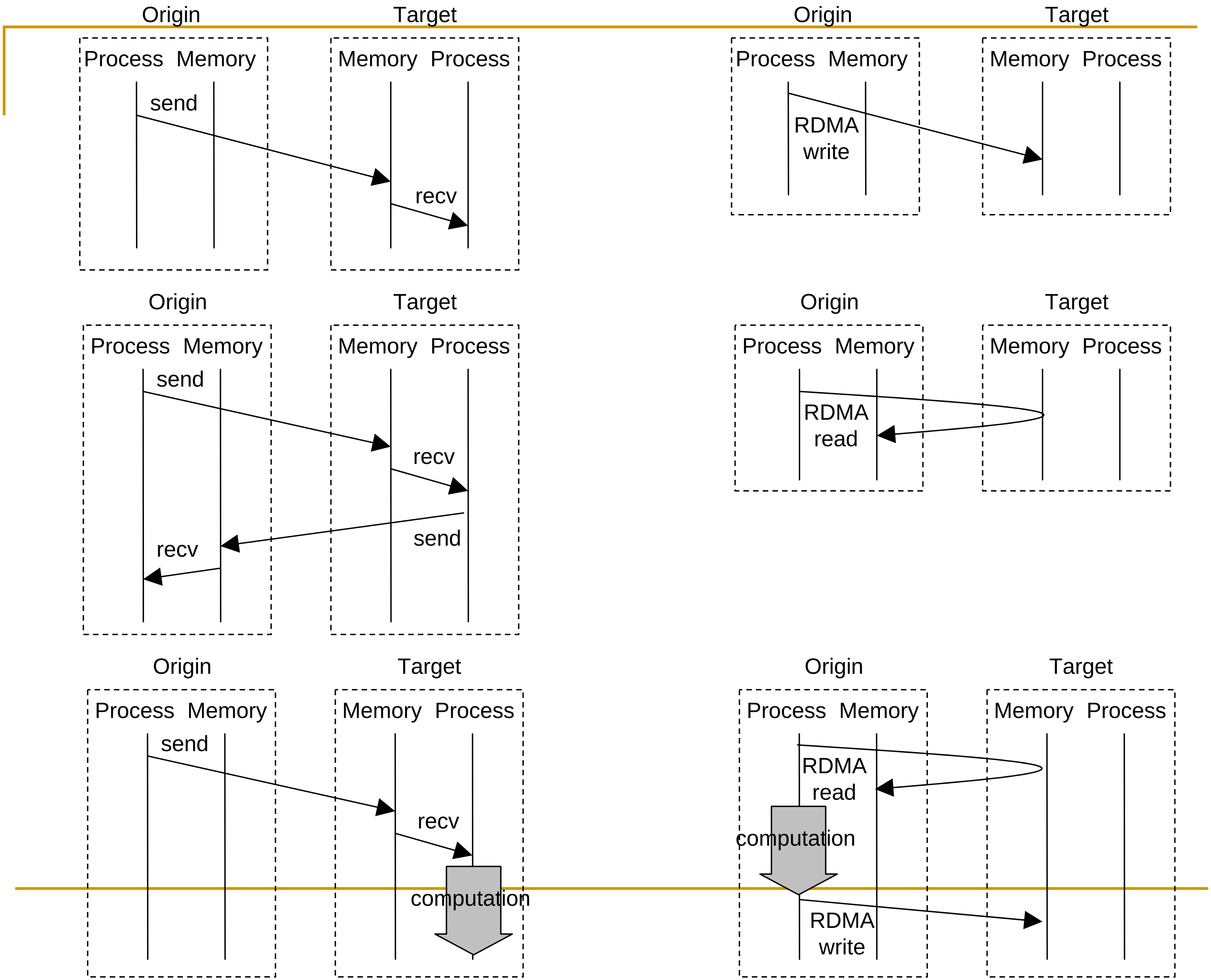


RDMA-Based Implementation: 14.78 microseconds (13% improvement)
Send/Receive-Based Implementation: 16.52 microseconds.

Bandwidth (Get)



The Bandwidth drop is due to the performance difference between InfiniBand RDMA read and RDMA write.


Origin
Target
Process Memory
Memory Process
send
recv
Origin
Target
Process Memory
Memory Process
RDMA
write
Origin
Target
Process Memory
Memory Process
send
recv
send
recv
Origin
Target
Process Memory
Memory Process
RDMA
read
Origin
Target
Process Memory
Memory Process
send
recv
computation
Origin
Target
Process Memory
Memory Process
RDMA
read
computation
RDMA
write

Synchronization

- Origin Side
  - Maintain a bit vector (Origin), each bit represents the status of a target.
  - Start : Check Origin vector, if one bit is changed, starts communication to that target
  - Complete: use RDMA write to change the corresponding bit at target side (Target vector).
- Target side.
  - Maintain a bit vector (Target), each bit represents the status of a origin.
  - Post: use RDMA write to change the corresponding bit at origin side (Origin vector).
  - Wait: wait until all the bits in the Target vector have been changed

Synchronization

- MPICH2-0.96p1 only supports active synchronization, this work focused on active synchronization.