Scheduling of MPI-2 One Sided Operations over InfiniBand

Wei Huang
Gopalakrishnan Santhanaraman
Hyun-Wook Jin
Dhabaleswar K. Panda

Network Based Computing Laboratory
The Ohio State University

Outline

- Background
  - InfiniBand, MPI2 and MVAPICH2
- Motivation
- Design and Implementation
- Performance Evaluation
- Conclusion and Future Work

InfiniBand

- The InfiniBand Architecture (IBA): new industry standard for high speed interconnect
- IBA supports channel semantics (send/recv) and RDMA semantics.
- RDMA (Remote Direct Memory Access) has better performance than Send/Recv
- VAPI: Mellanox implementation of InfiniBand Verbs interface

MPI-2

- MPI-2 Standard: An extension to MPI-1
- Among the new features of MPI-2:
  - One Sided Communication (Or Remote Memory Access, *RMA*)
  - Dynamic Process Management
  - Parallel I/O
- MPICH-2: MPI-2 implementation from ANL

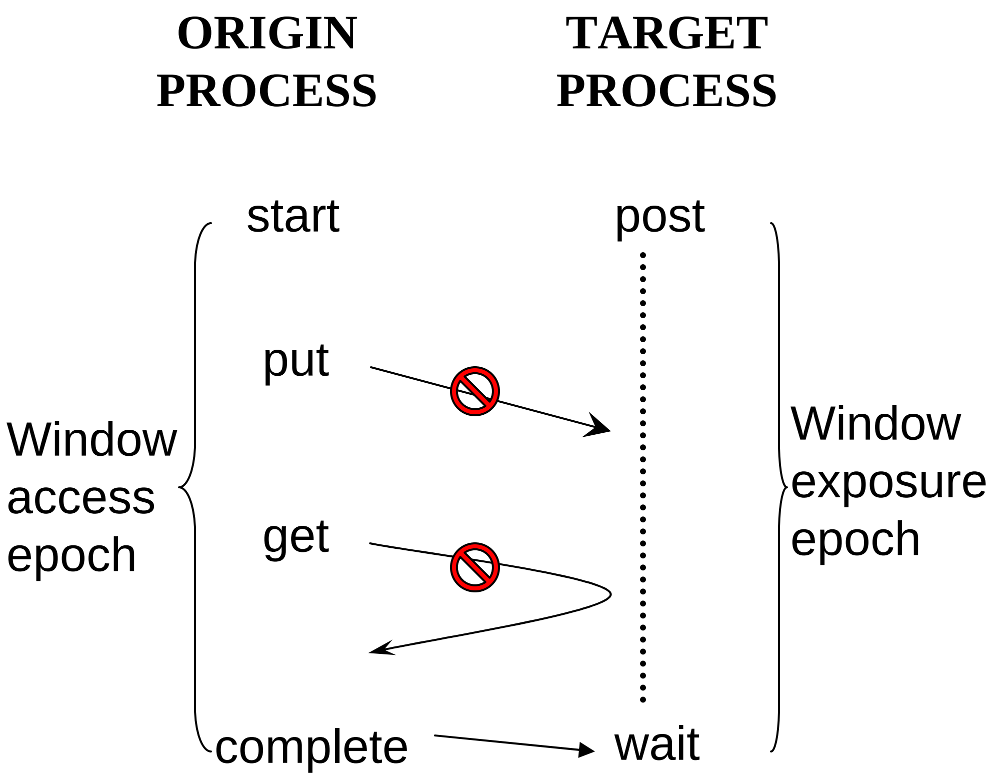
MVAPICH2

- MPICH-2 is highly modulated and is layered structure
- MVAPICH2 is an open-source MPI-2 implementation over InfiniBand at RDMA channel level
  - http://nowlab.cis.ohio-state.edu/projects/mpi-iba/index.html
  - Latest release is MVAPICH2-0.6.0
  - Incorporates Most of the concepts presented in this paper
- MVAPICH2-0.6.0 and MVAPICH-0.9.4 (MPI-1 version) are currently being used by more than 190 organizations (in 26 countries)

One Sided Operations

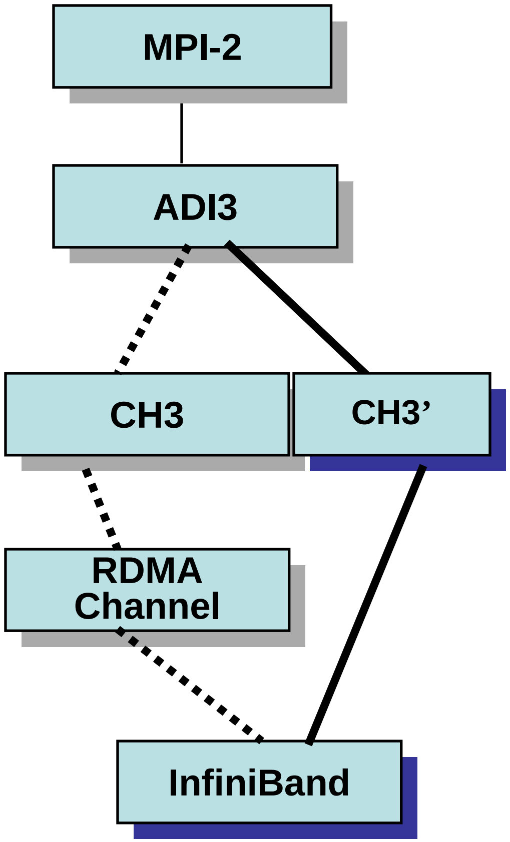
- Window: Must be created before RMA happen:
  - Definition: Process memory made accessible by remote processes.
  - All RMA operations access memory in window
- Supported Communication (RMA) Calls:
  - MPI_Put, MPI_Get, MPI_Accumulate
- Synchronization Schemes:
  - Active Synchronization
  - Passive Synchronization: lock and unlock
  - Win fence

Active One Sided Synchronization



- Win_post and Win_wait start and end a window exposure epoch
- Win_start and win_complete start and end a window access epoch
- RMAs can only issue during access epoch
- RMAs can only take effect at remote side during exposure epoch

One Sided in MVAPICH-2



- Point-to-Point based:
  - Implement RMA operations based on p2p operations
- Direct One Sided:
  - Extend CH3 layer
  - Map One sided operations directly to VAPI calls:
    - MPI_Put → RDMA write
    - MPI_Get → RDMA read
  - Generally achieve better performance

Outline

- Background
  - InfiniBand, MPI2 and MVAPICH2
- Motivation
- Design and Implementation
- Performance Evaluation
- Conclusion and Future Work

From MPI-2 Standard

- *The implementation can delay communication operations until the synchronization calls occur, for efficiency*
- *It is erroneous to have concurrent conflicting accesses to the same memory location in a window; if a location is updated by a put or accumulate operation, then this location cannot be accessed by a load or another RMA operation until the updating operation has completed at the target.*

Motivation

- We interpret last slide as: we have the freedom to schedule RMA operations
- Current design does not make full use of this
- Can such scheduling be performed?
- Does such scheduling benefit?
  - Better overlap: utilize bidirectional bandwidth
  - Higher network bandwidth utilization

Outline

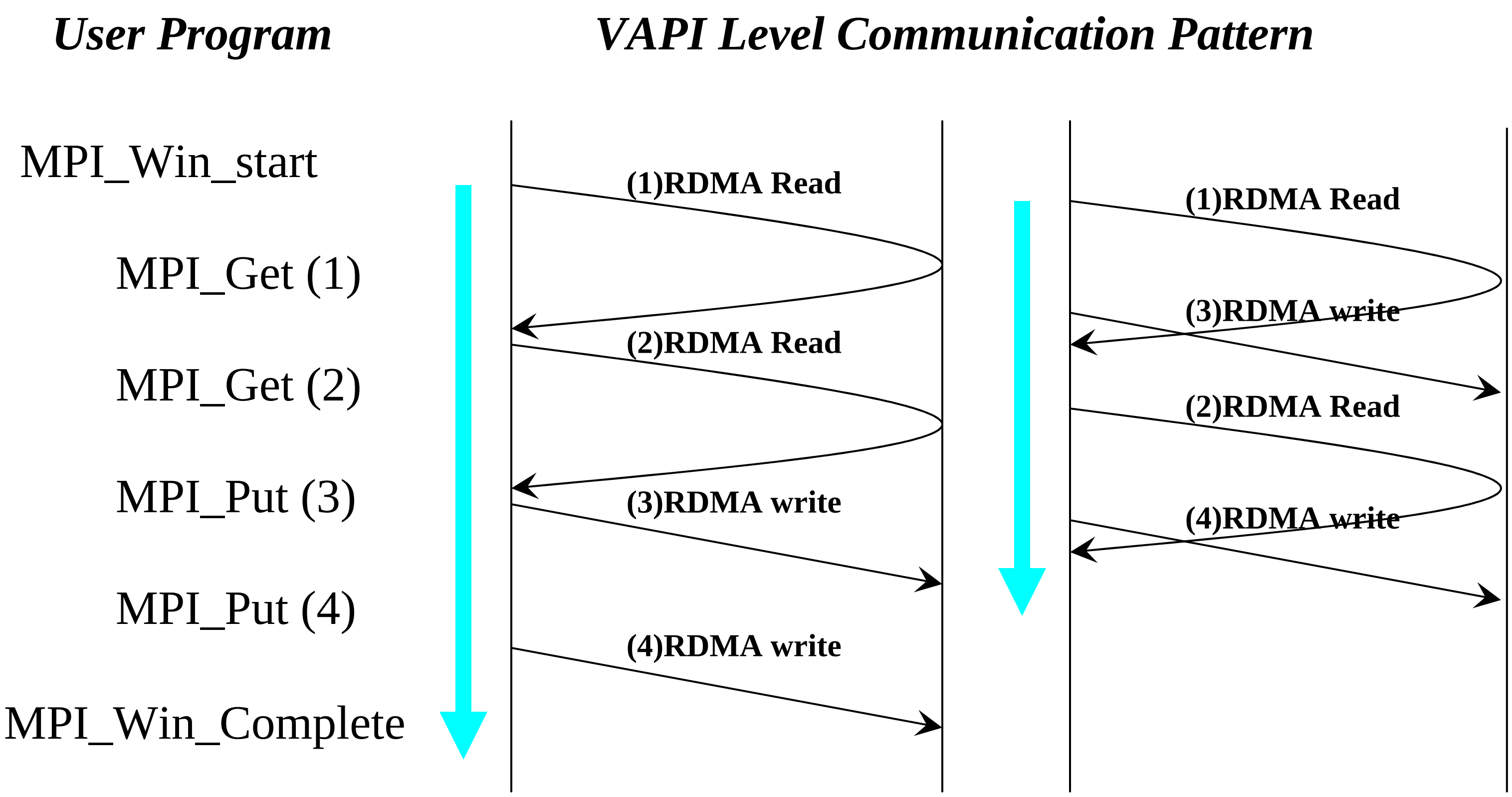
- Background
- Motivation
- Design and Implementation
  - Re-ordering Schemes
  - Aggregation Schemes
- Performance Evaluation
- Conclusion and Future Work

Re-ordering Approaches

Re-order the actual issuing of RMA operations to utilize network more efficiently:

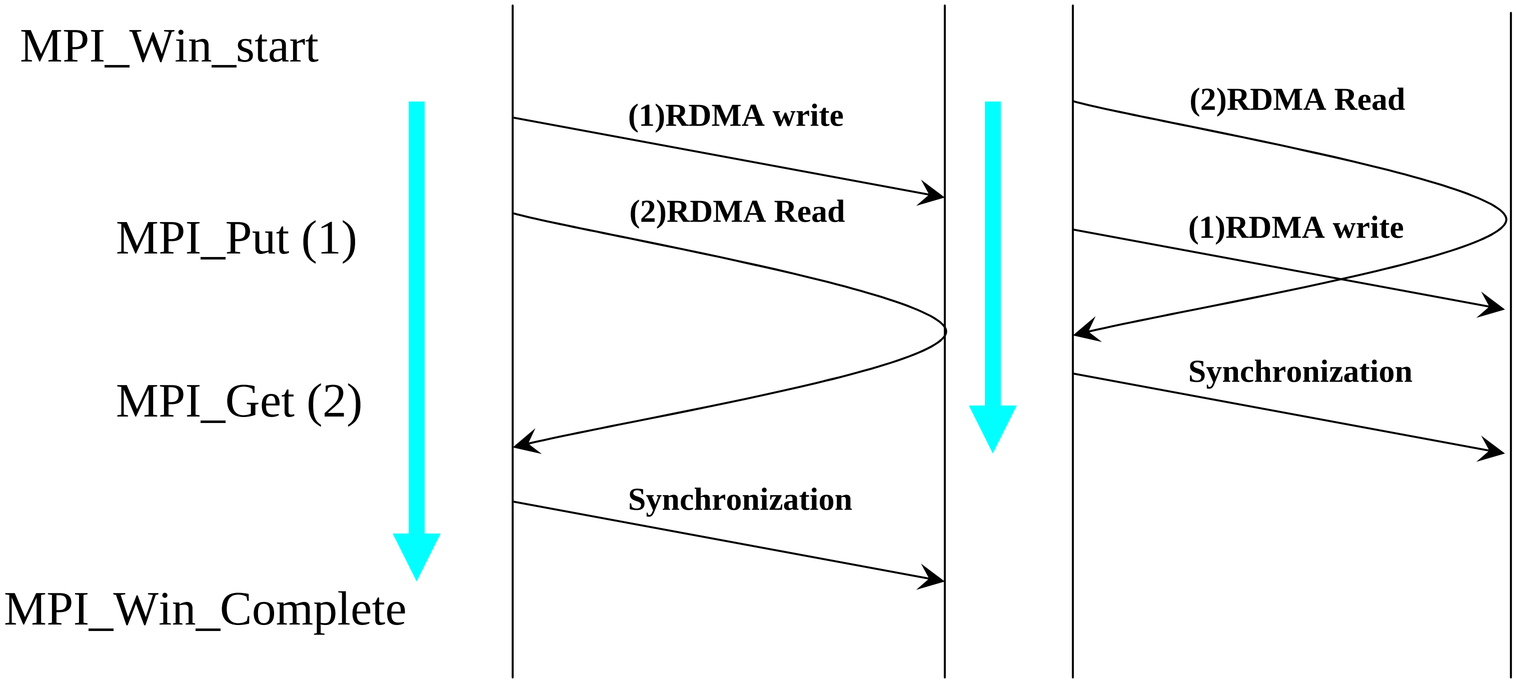
- Interleaving
  – Interleave put and get operations
- Prioritizing
  – Give priority to get operations

Interleaving



Prioritizing

User Program | *VAPI Level Communication Pattern*



Aggregation

- Combine several RMA operations to amortize the overhead of initializing and completing the operation:
  - RMA Operation + Synchronization
  - RMA Operation + RMA Operation
- Origin side aggregates the operations (data and operations) and target side does scatter
- Only point-to-point scheme can utilize
- Aggregating small size operations is more beneficial

Aggregation of RMA Operation & Synchronization

User Program

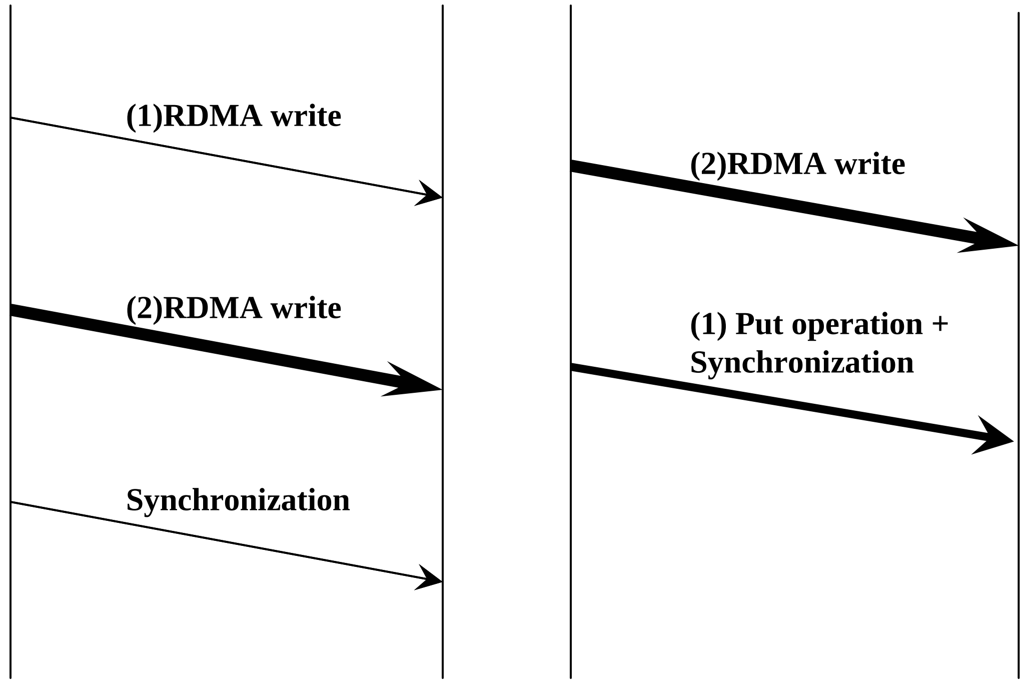
MPI_Win_start

MPI_Put (1)

MPI_Put (2)

MPI_Win_Complete

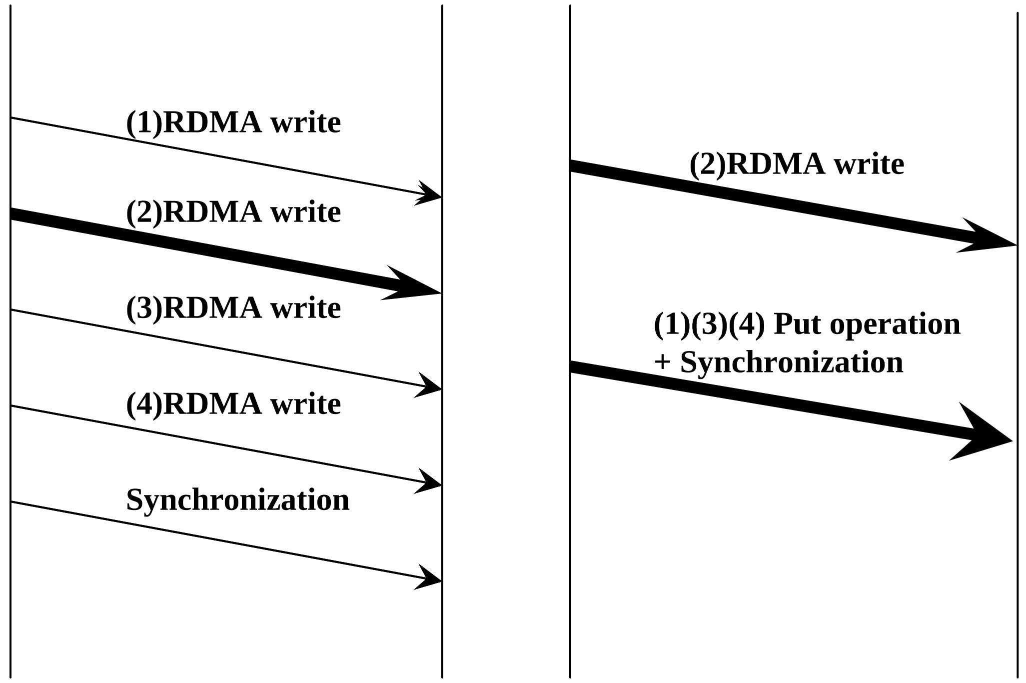
VAPI Level Communication Pattern



Aggregation of RMA Operations

User Program
MPI_Win_start
MPI_Put (1)
MPI_Put (2)
MPI_Put (3)
MPI_Put (4)
MPI_Win_Complete

VAPI Level Communication Pattern



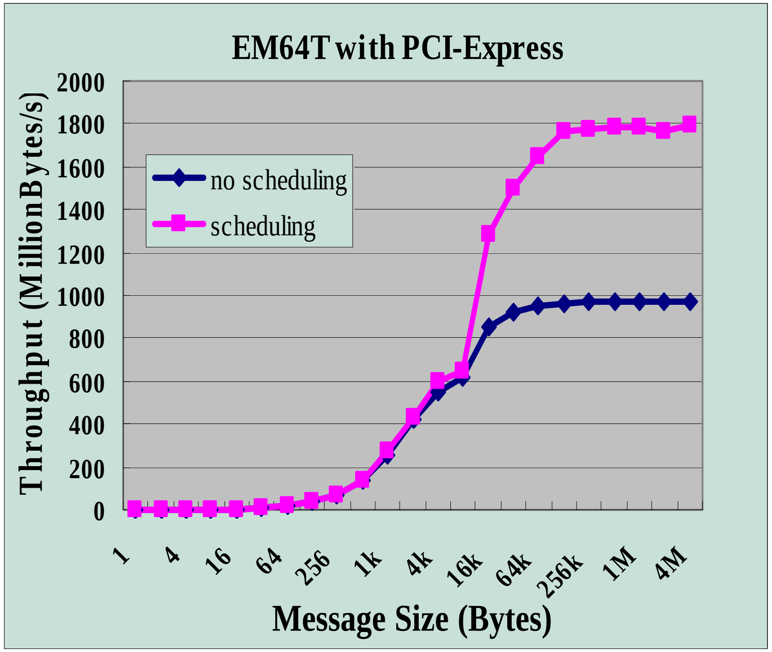
Outline

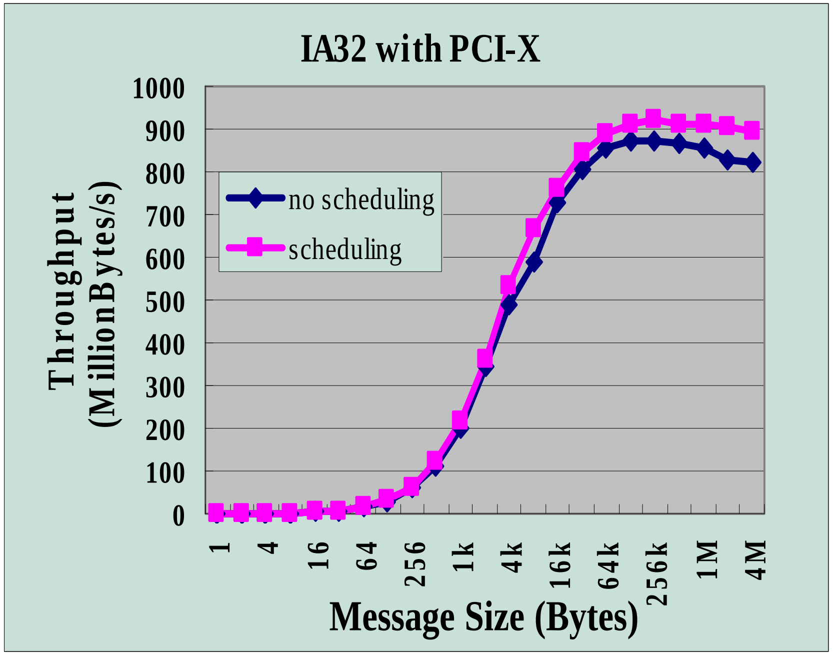
- Background
- Motivation
- Design and Implementation
  - Re-ordering Schemes
  - Aggregation Schemes
- Performance Evaluation
- Conclusion and Future Work

Performance Evaluation

- Test bed:
  - EM64T Cluster with PCI-Express HCAs
    - Dual 3.4 GHz Xeon Processors
    - 512 MB memory
  - IA32 Cluster with PCI-X HCAs
    - Dual 3.0 GHz Xeon Processors
    - 2GB memory
  - Both clusters connected by InfiniScale MTS2400 switch

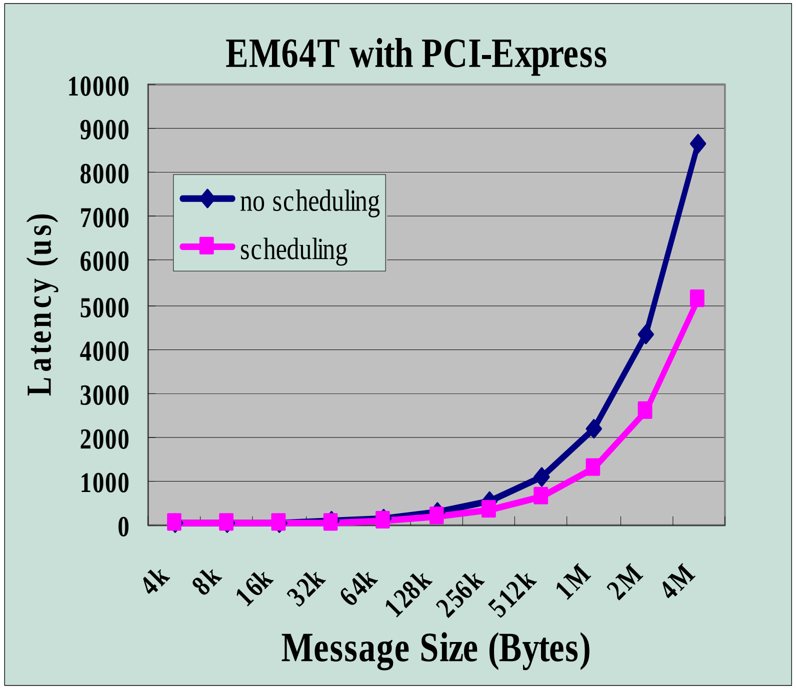
Impact of Interleaving on Throughput

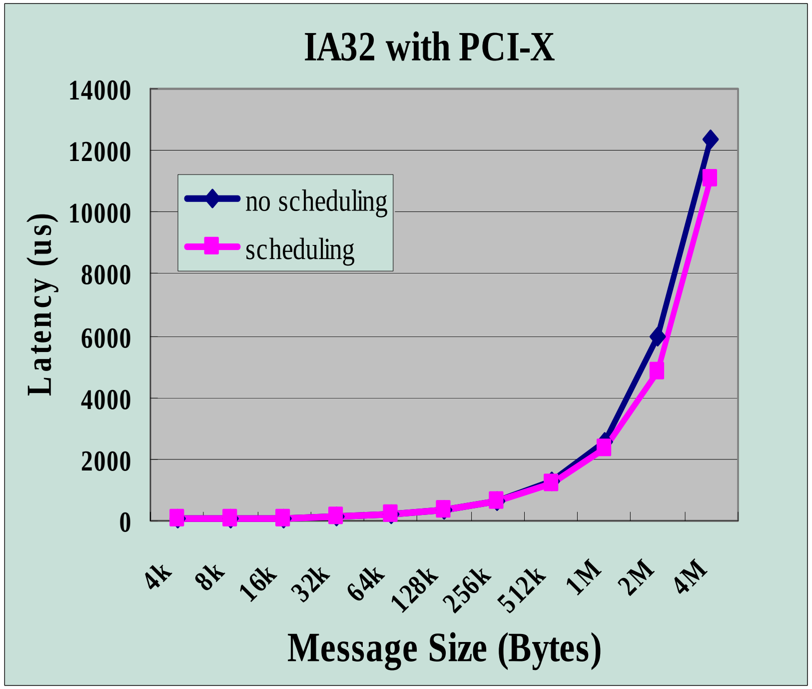




- Test: 16 MPI_Put followed by 16 MPI_Get during one access epoch
- Throughput test shows up to 76% (980 MB/s → 1788 MB/s) improvement by scheduling on PCI-Express System and 8% on PCI-X system.

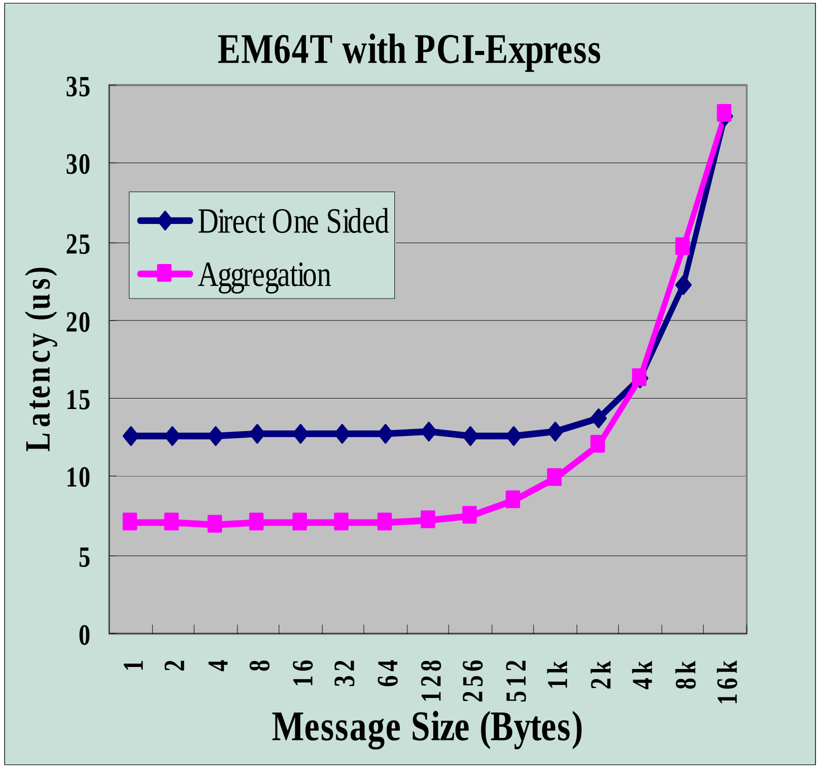
Impact of Prioritizing on Latency

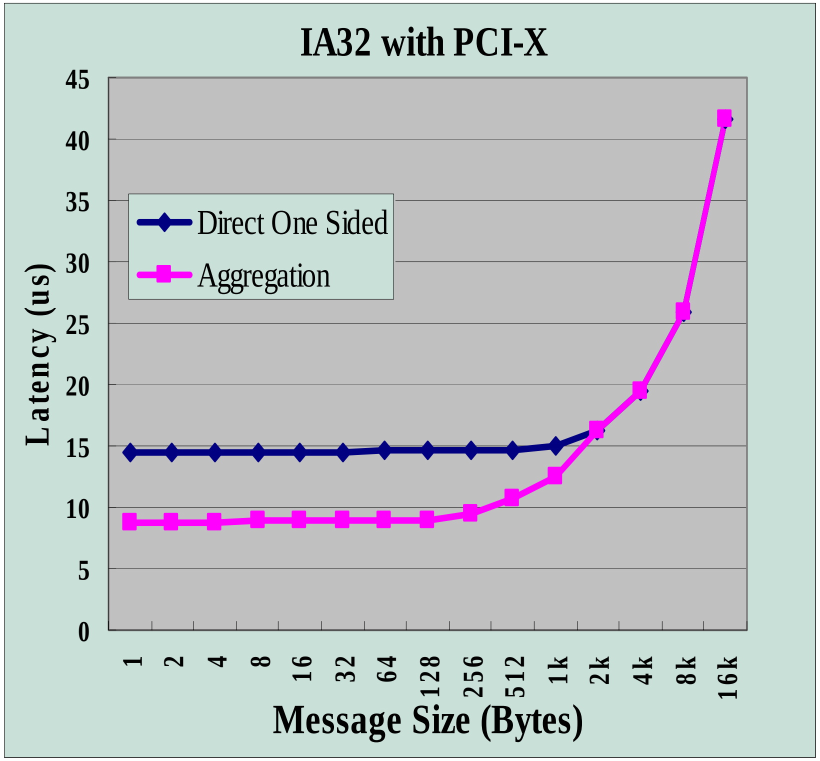




- One MPI_Put followed by one MPI_Get during one access epoch
- Improvement up to 40% (8661 ms → 5.144 ms) on PCI-Express system and 20% on PCI-X system

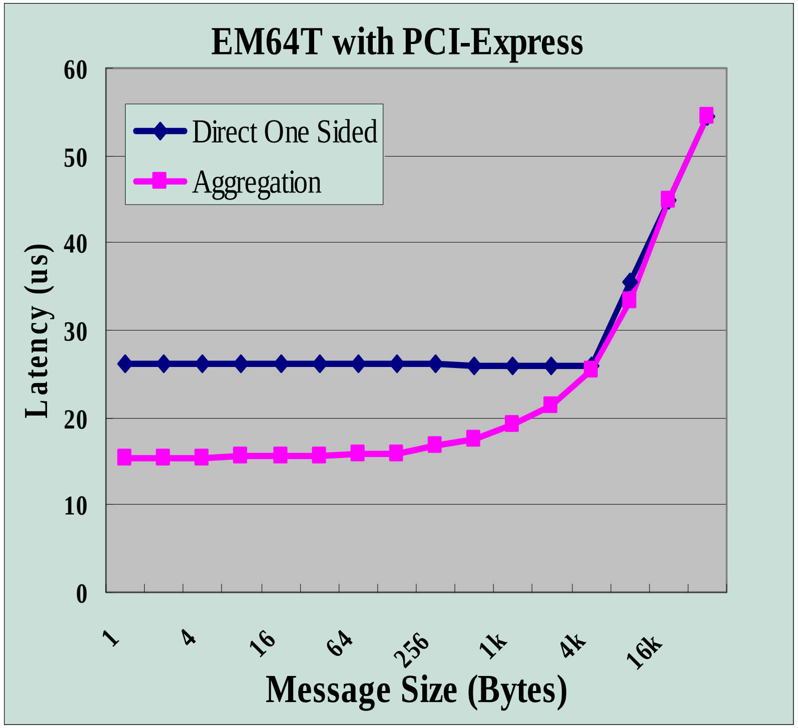
Impact of Aggregation on MPI_Put Latency

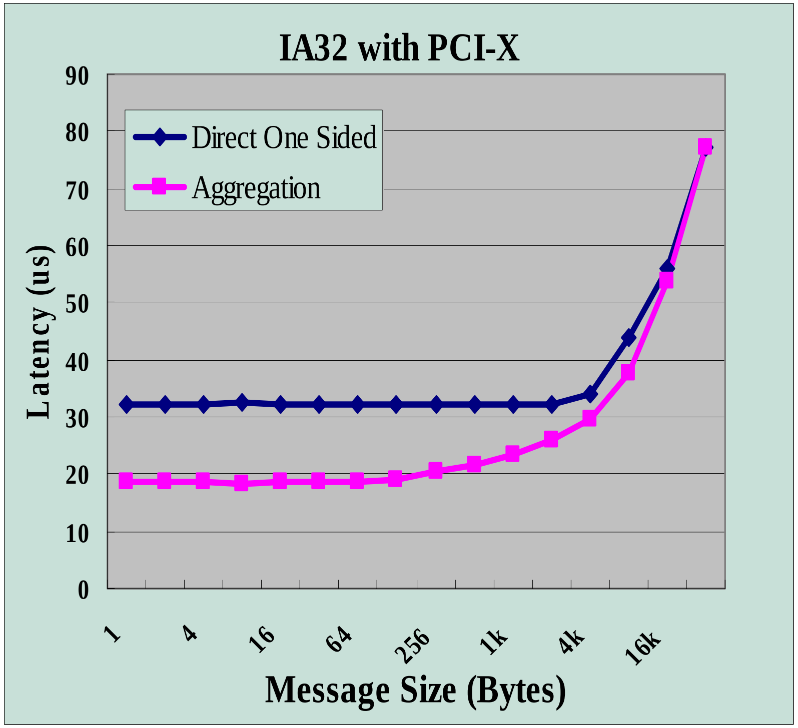




- One MPI_Put + synchronization during one access epoch
- For small message, MPI_Put latency significantly reduces: 44% (12.6us→ 7.0us) for PCI-Express system and 38% (14.4us→8.8us) for PCI-X system

Impact of Aggregation on MPI_Get Latency





- One MPI_Get + synchronization during one access epoch
- Same trends for MPI_Get latency. 42% (26.1us→15.4us) for PCI-Express system and 42% (32.2us→18.6us) for PCI-X system

Conclusions

- Different scheduling and aggregation schemes can improve the performance of MPI-2 one sided communication.
- With re-ordering scheme, we observe an improvement in the throughput up to 76%, latency up to 40% for certain scenarios.
- With aggregation scheme, we observe an improvement of 44% and 42% for MPI_Put and MPI_Get latency on PCI-Express platform.
- Similar trends were observed for PCI-X platform.

Future Work

- Extend our implementation of our aggregation scheme for combining multiple RMA operations.
- Explore more optimized scheduling schemes.
- Merge the different schemes into one framework which can adaptively choose based on the communication pattern.
- Separate thread implementation
- Application level study

Web Pointers

NBC-LAB

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

Emails: {huanwei, santhana, jinhy, panda}@cse.ohio-state.edu