

Design and Evaluation of Generalized Collective Communication Primitives with Overlap using ConnectX-2 Offload Engine

Hari Subramoni, Krishna Kandalla, **Sayantana Sur**
and Dhabaleswar. K. Panda

Department of Computer Science and Engineering
The Ohio State University

HotI '10

Outline

- Introduction
- Problem Statement
- Design of Communication Primitives
- Performance Evaluation
- Conclusions and Future Work

Introduction

- MPI has been the pre-dominant parallel programming model
- Scientific applications **heavily** use collective communication
- Collective communication performance is usually a scaling bottleneck
 - Non-blocking Collective Communication in upcoming MPI-3 may provide a solution for exascale level systems by overlapping time spent in collectives
- Mellanox's ConnectX-2 adapter features "task-list" offload interface
 - *Extension to existing InfiniBand APIs (patented by Mellanox)*
- Accordingly MPI software stacks need to be re-designed to leverage offload in a comprehensive manner

Quick Overview of InfiniBand

- An industry standard for low latency, high bandwidth, system area networks
- Many features
 - Two communication types: send/receive, RDMA
 - Multiple transport types: Reliable, Unreliable
 - Multiple connection types: connected, datagram
 - Non-contiguous data transfer
 - Switch-based Broadcast
 - Multiple virtual lanes, ...
- Multiple communication speeds – SDR / DDR / QDR
- Adoption in HPC is steadily increasing with 40% systems in Top500 (June 2010) using IB

Outline

- Introduction
- Problem Statement
- Design of Communication Primitives
- Performance Evaluation
- Conclusions and Future Work

Problem Statement

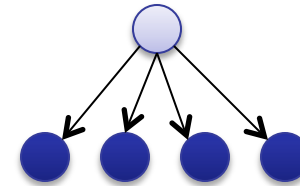
- Can we break down collective communication into smaller, *general*, communication primitives?
 - With the aim that using these general primitives most collectives can be formulated
- Can we implement them in both host-based and offloaded modes?
- How does the offload mode compare with host-based mode on a modern representative platform, such as Nehalem servers with PCIe-Gen2?

Outline

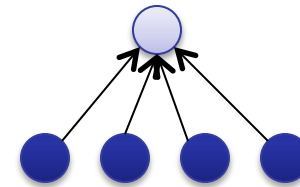
- Introduction
- Problem Statement
- Design of Communication Primitives
- Performance Evaluation
- Conclusions and Future Work

Communication Primitives

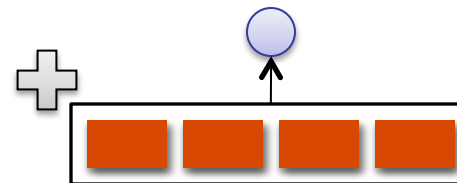
One-to-many Multi-send



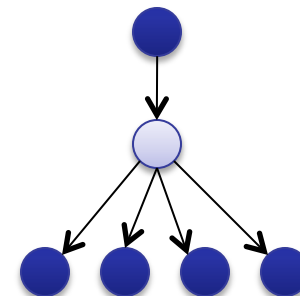
Multi-receive



Receive-reduce

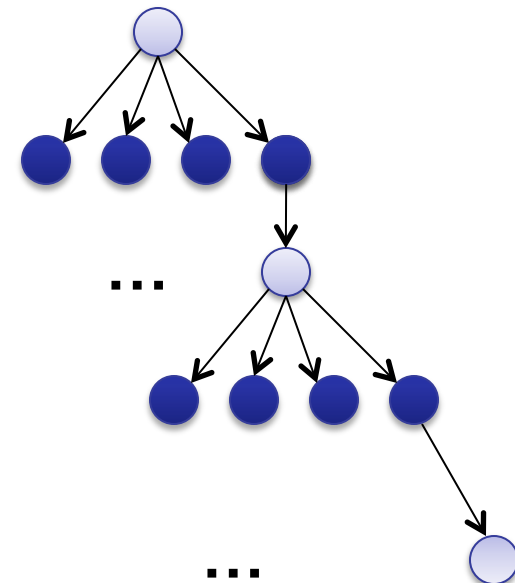


Receive-replicate

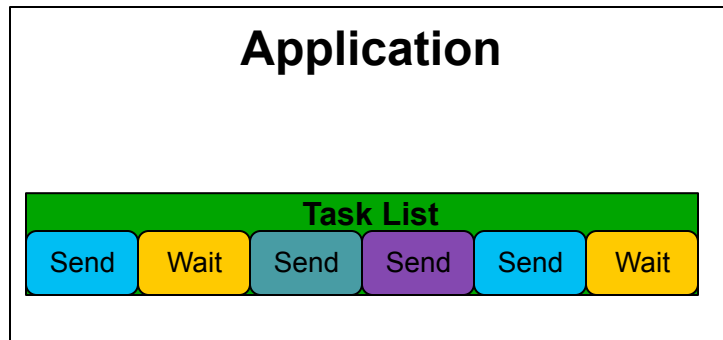


Composing Collectives from Primitives

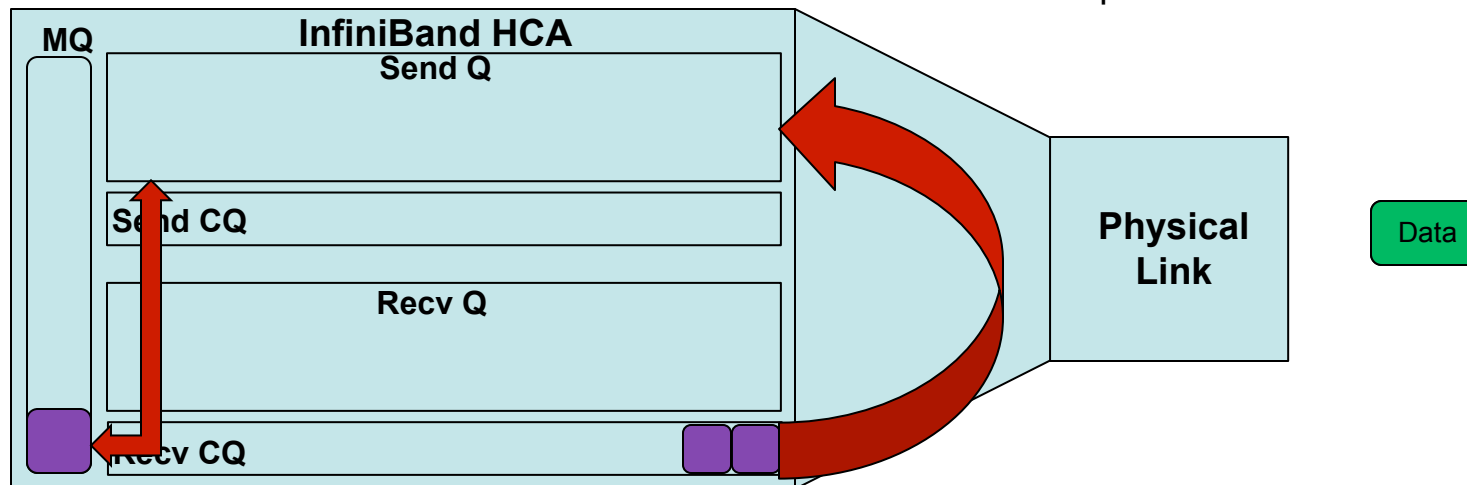
```
MPI_Bcast( ... ) // blocking one for now
{
  if(i_am_root) {
    task_list = one_to_many_multi_send(...);
  } else if (i_am_leaf) {
    task_list = multi_recv(...);
  } else {
    task_list = recv_replicate(...);
  }
  post_work_queue(task_list);
  wait();
}
```



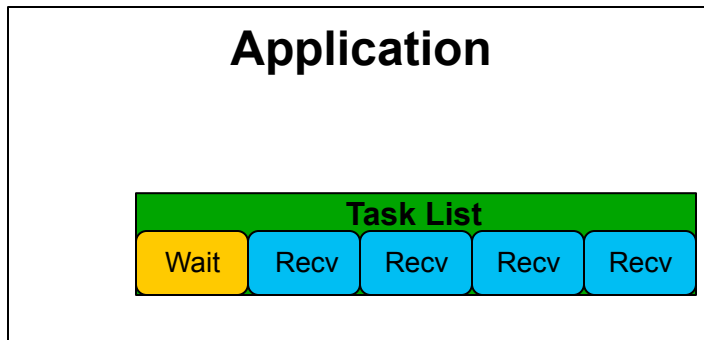
One-to-many Multi-Send



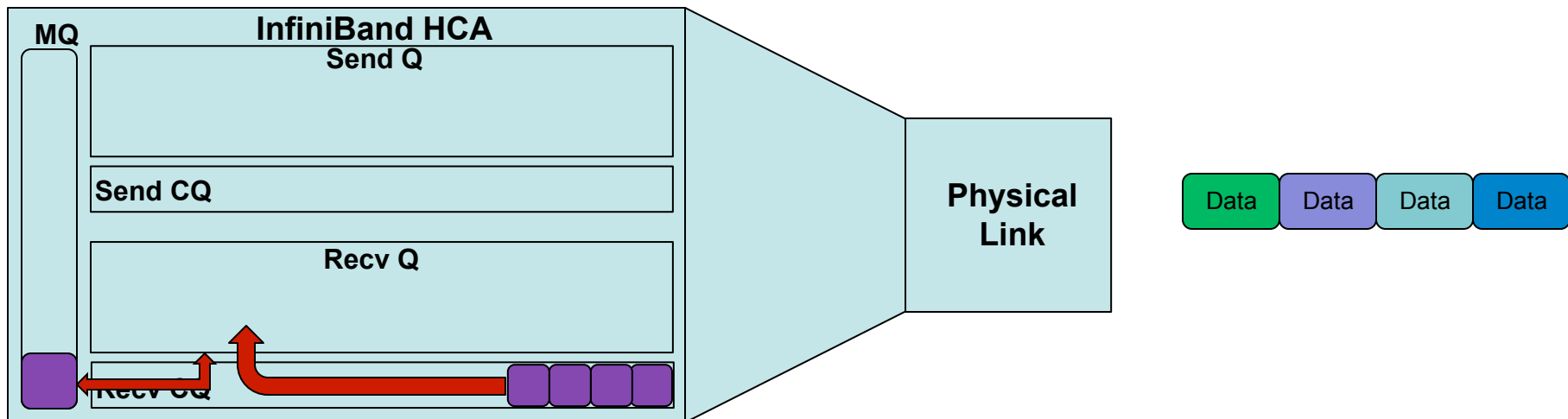
- Sender creates a task-list consisting of only send and wait WQEs
 - One send WQE is created for each registered receiver and is appended to the rear of a singly linked task-list
 - A wait WQE is added to make the ConnectX-2 HCA wait for acknowledgment packet from the receiver



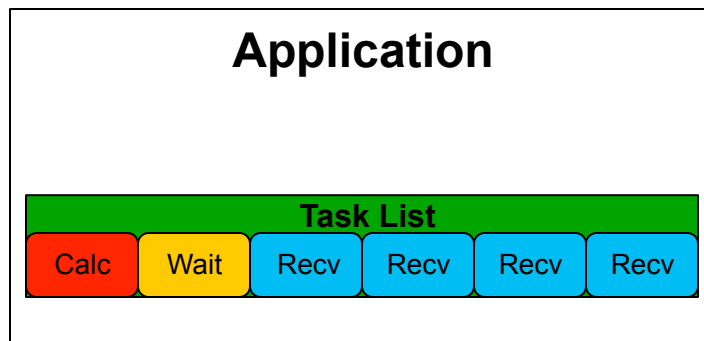
Multi-Receive



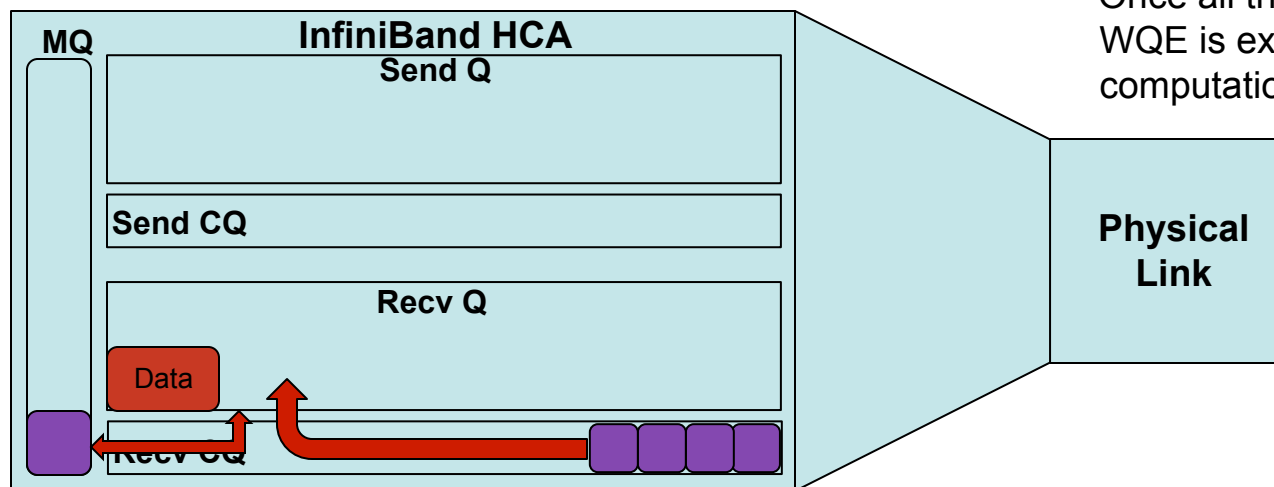
- Receiver creates a task-list consisting of receives and optionally wait WQEs
 - One recv WQE is created for each registered receiver and is appended to the rear of a singly linked task-list
 - Optionally a wait WQE can be added to make the HCA wait for some condition



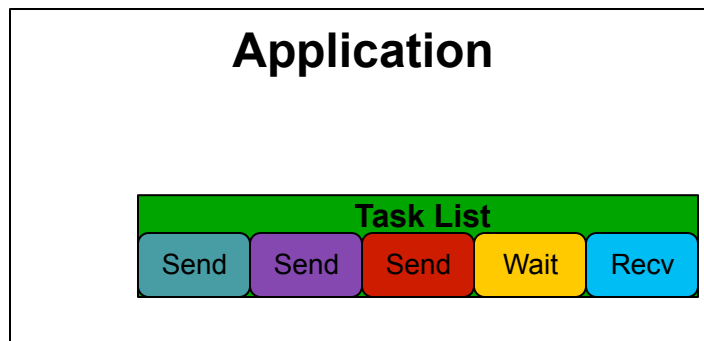
Receive-Reduce



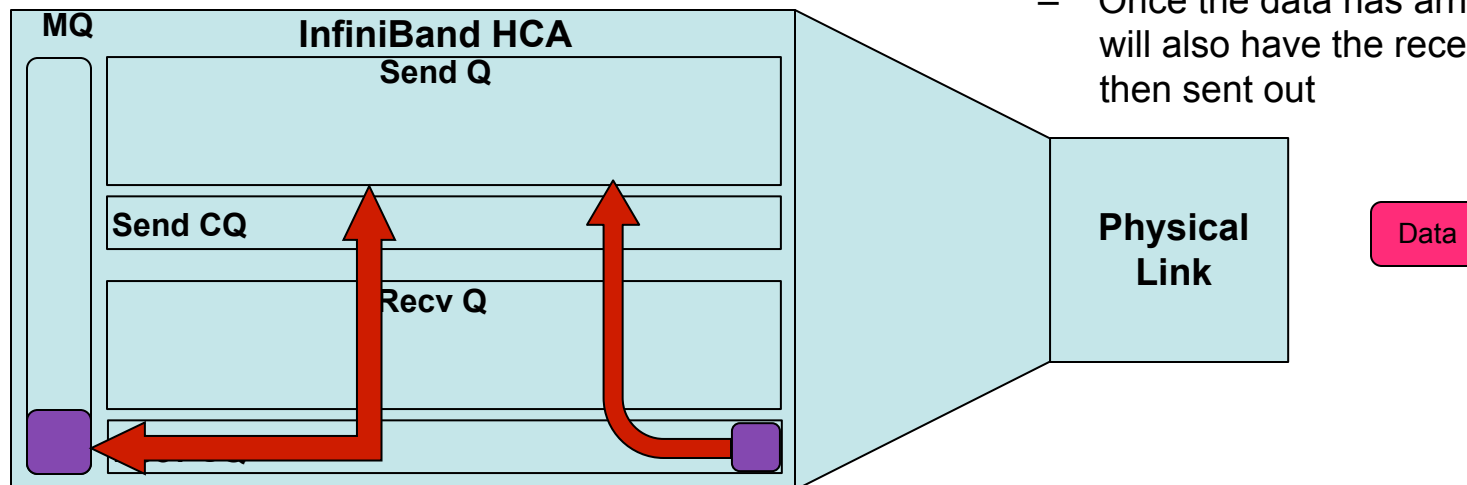
- Receiver creates a task-list consisting of receives, waits and calculate WQEs
 - One recv WQE is created for each registered receiver and is appended to the rear of a singly linked task-list
 - A wait WQE is to make the HCA wait for the arrival of all the data
 - Once all the data has arrived, the calculate WQE is executed to perform the required computation



Receive-Replicate



- Receiver creates a task-list consisting of receive, wait and send WQEs
 - One recv WQE is created for each registered receiver and is appended to the front of a singly linked task-list
 - A wait WQE is to make the HCA wait for the arrival of the data
 - The recv WQE is re-posted as a send WQE
 - Once the data has arrived, the send WQE will also have the received data which is then sent out



Outline

- Introduction
- Problem Statement
- Design of Communication Primitives
- Performance Evaluation
- Conclusions and Future Work

MVAPICH/MVAPICH2 Software

- High Performance MPI Library for IB and 10GE
 - MVAPICH (MPI-1) and MVAPICH2 (MPI-2)
 - Used by more than 1,185 organizations in 59 countries
 - More than 44,000 direct downloads from OSU site
 - Empowering many TOP500 clusters
 - 6th ranked 81,920-core cluster (Pleiades) at NASA
 - 7th ranked 71,680-core cluster (Tianhe-1) at NUDT, China
 - 11th ranked 62,976-core cluster (Ranger) at TACC
 - Available with software stacks of many IB, 10GE and server vendors including Open Fabrics Enterprise Distribution (OFED)
 - Also supports uDAPL device (for networks supporting uDAPL)
 - <http://mvapich.cse.ohio-state.edu/>

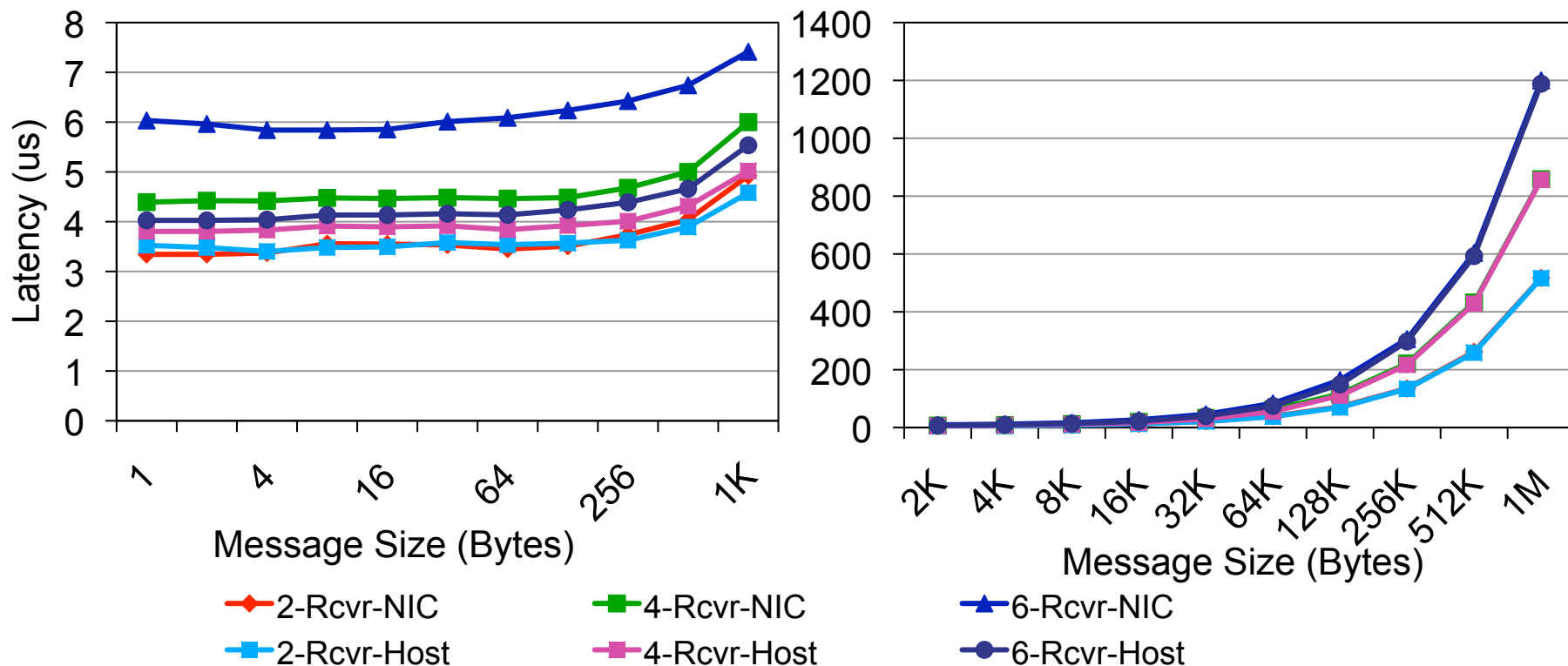
Experimental Testbed

- Compute platform
 - Intel Xeon E5530 (nehalme) dual quad-core processors @ 2.40 GHz
 - 12GB RAM, 8MB cache
 - PCIe 2.0 interface
- Network Equipment
 - MT26428 QDR ConnectX-2 HCAs
 - CORE-Direct enabled firmware from Mellanox
 - 36-port Mellanox QDR switch used to connect all the nodes
- Red Hat Enterprise Linux Server release 5.3 (Tikanga)
- Vendor modified version of OFED based on OFED-1.4.1
- Initial versions of firmware and software, improvements are expected in newer versions

Performance Results

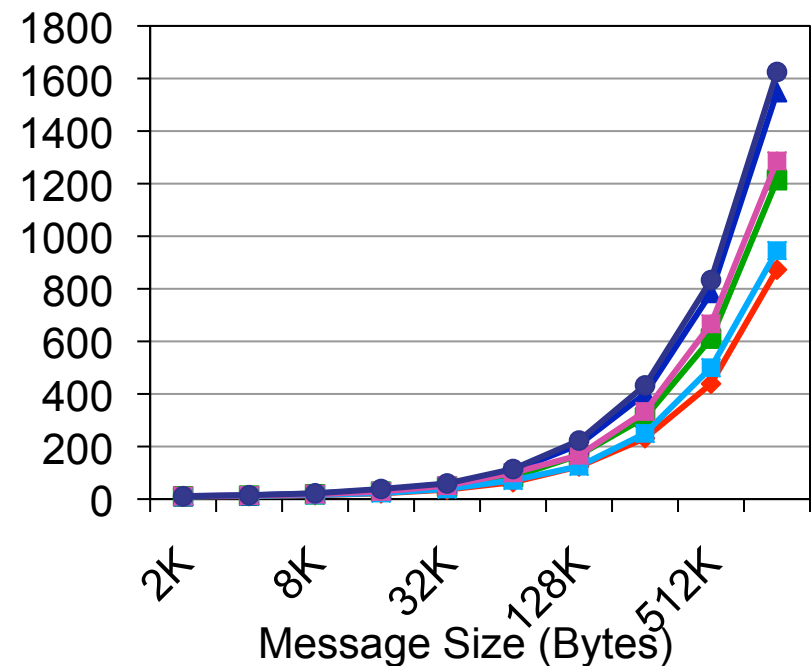
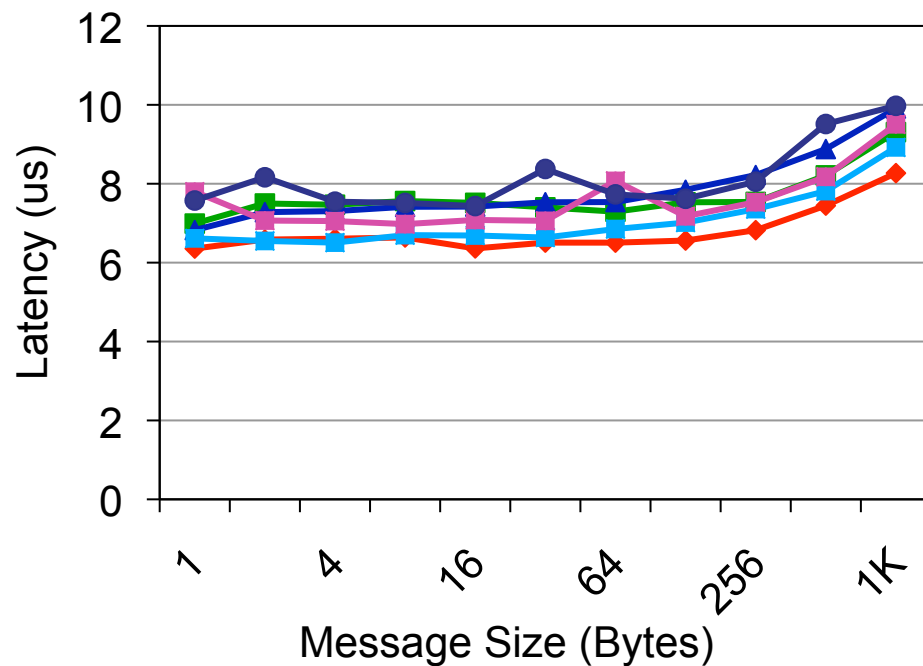
- Absolute Performance
 - Multi-send
 - Receive-Replicate
 - Receive-Reduce
- Overlap Percentage
 - Multi-send
 - Receive-Replicate
 - Receive-Reduce
- MPI level performance - MPI_Barrier

Multi-Send Performance



- Length of task list critical to performance of offloaded operations
- Long task lists are not ideal to get best latency performance for small messages

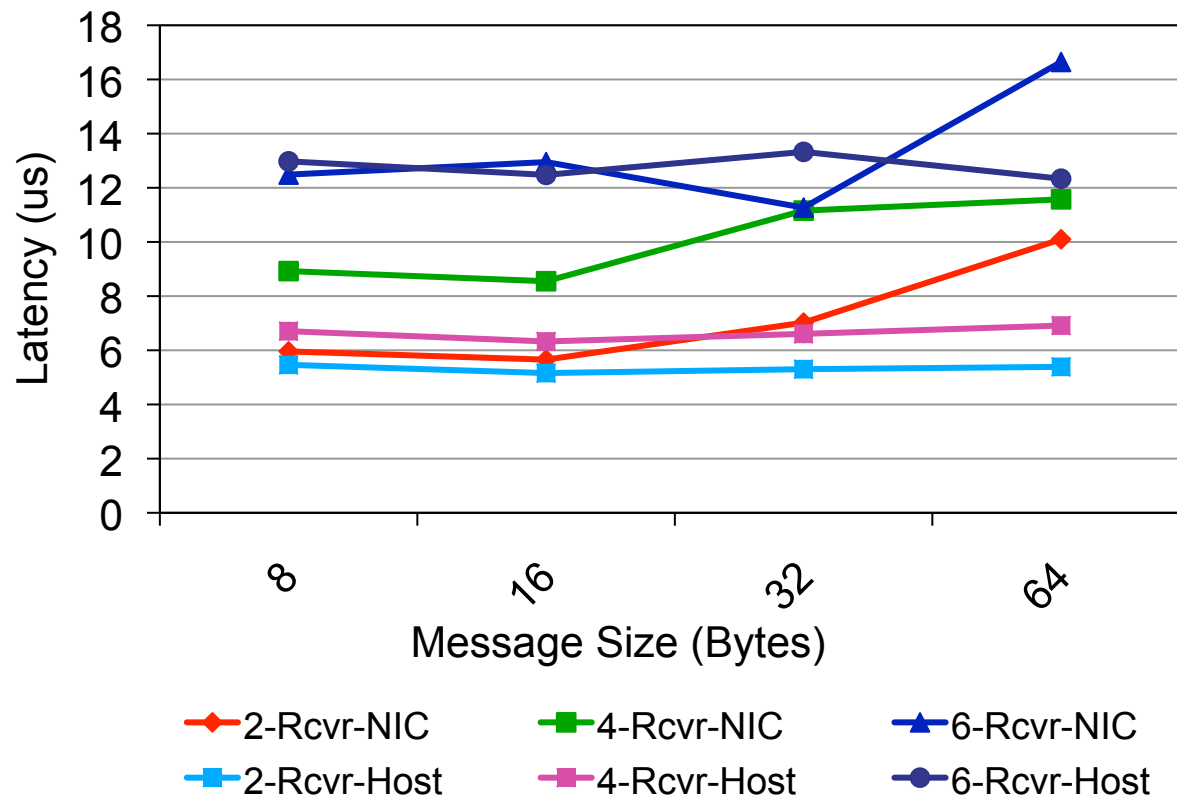
Receive-Replicate Performance



- ◆ 2-Rcvr-NIC
- 2-Rcvr-Host
- 4-Rcvr-NIC
- 4-Rcvr-Host
- ▲ 6-Rcvr-NIC
- 6-Rcvr-Host

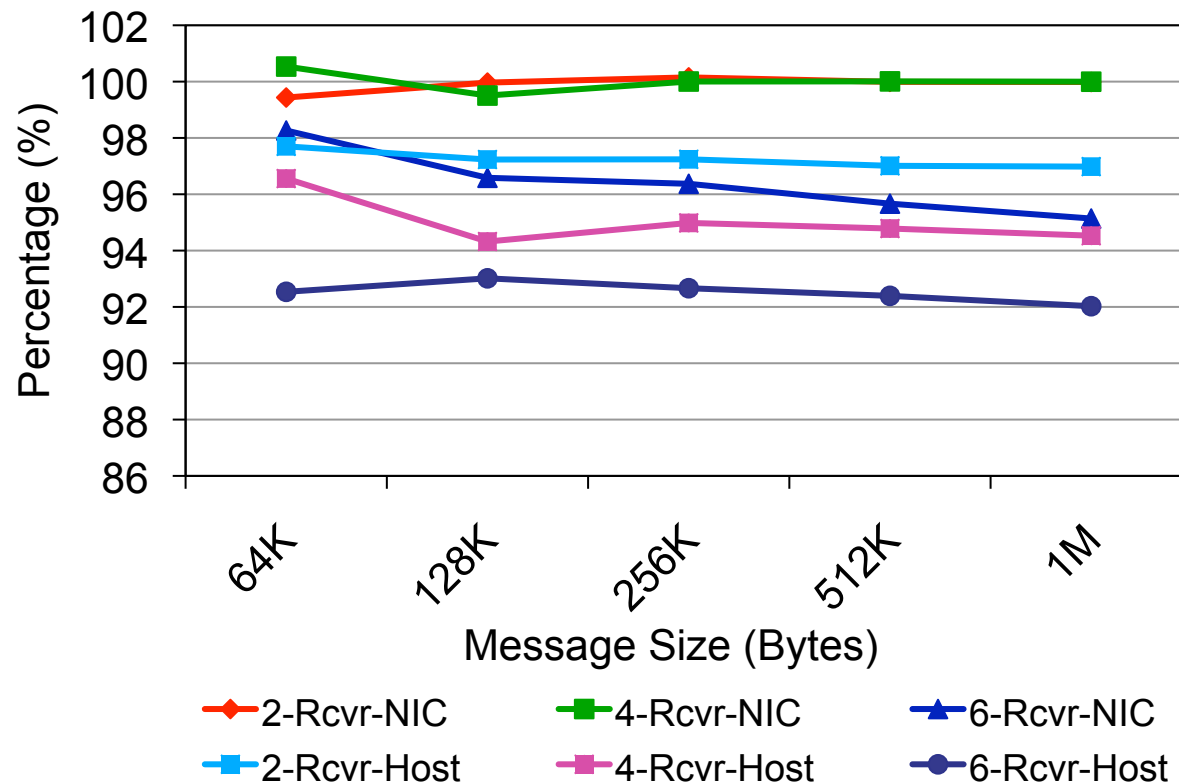
- Offloaded operations either out perform or match the performance of host based operations

Receive-Reduce Performance



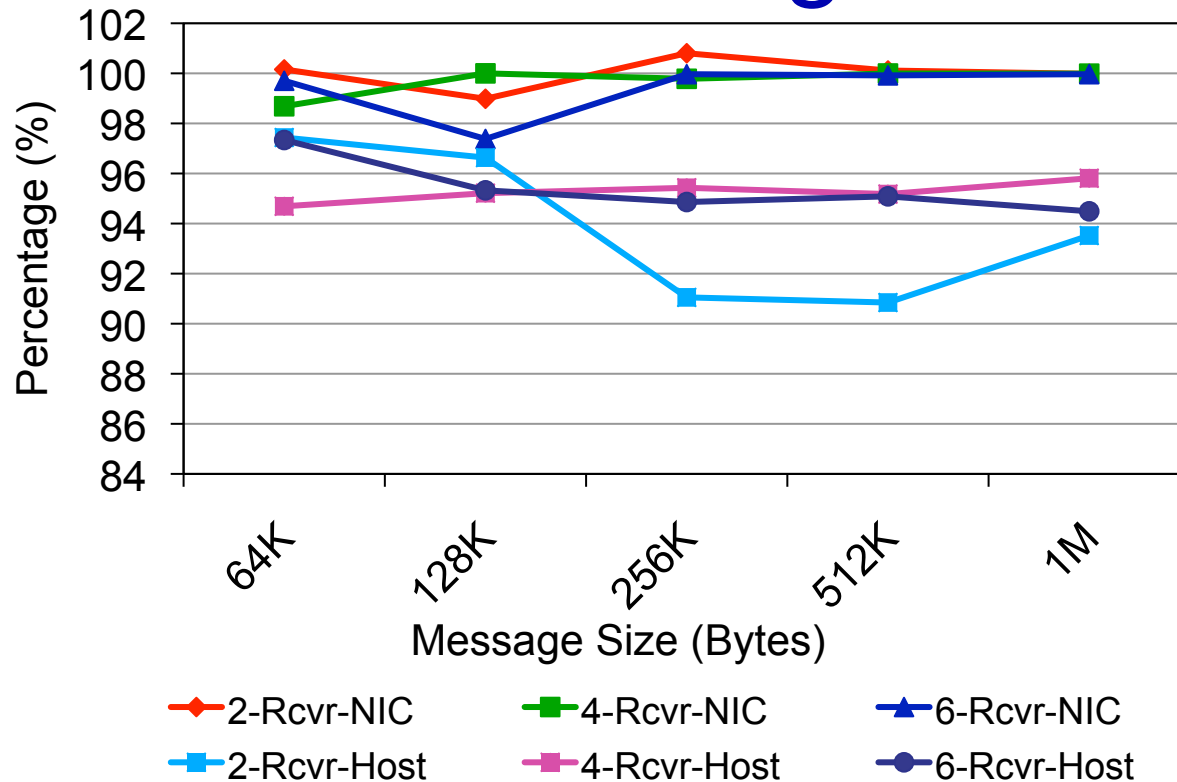
- Size of the reduction object critical factor in performance
- Offloaded performance matches host based performance as long as data fits in one WQE

Multi-Send Overlap Percentage



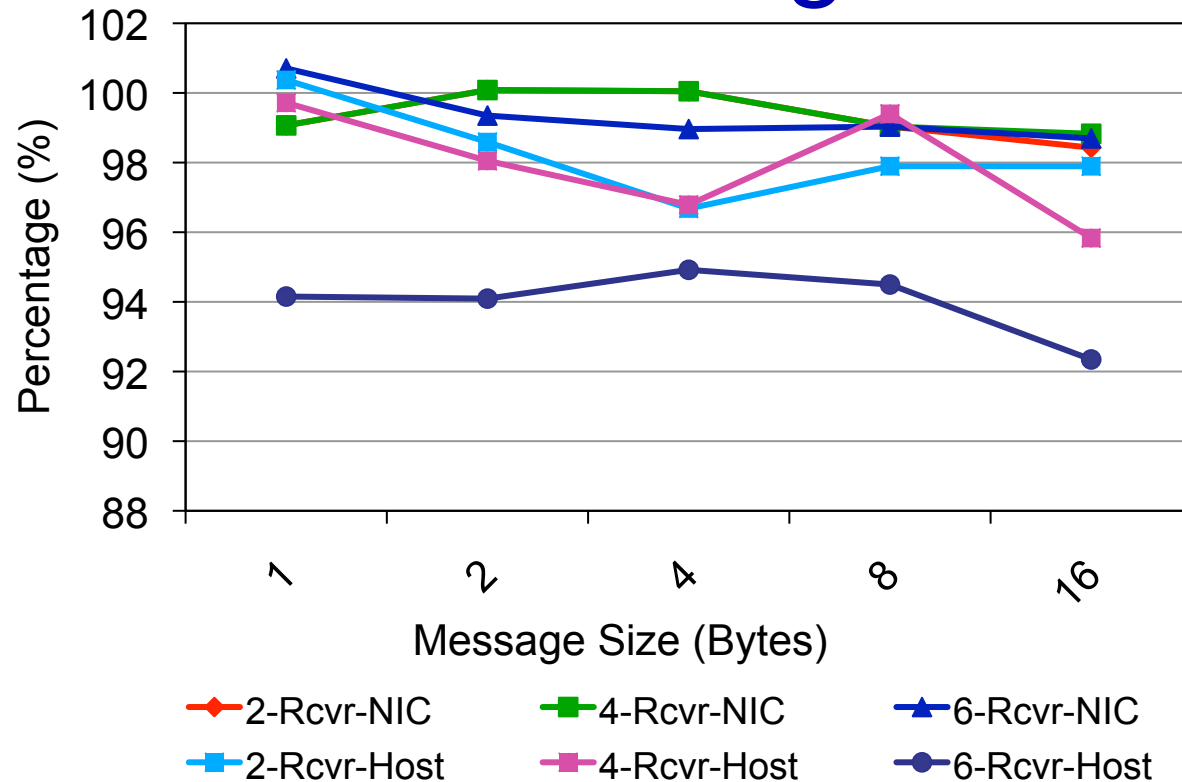
- Offloaded primitives can achieve very good overlap as compared to their host-based counterparts (95% or above)

Receive-Replicate Overlap Percentage



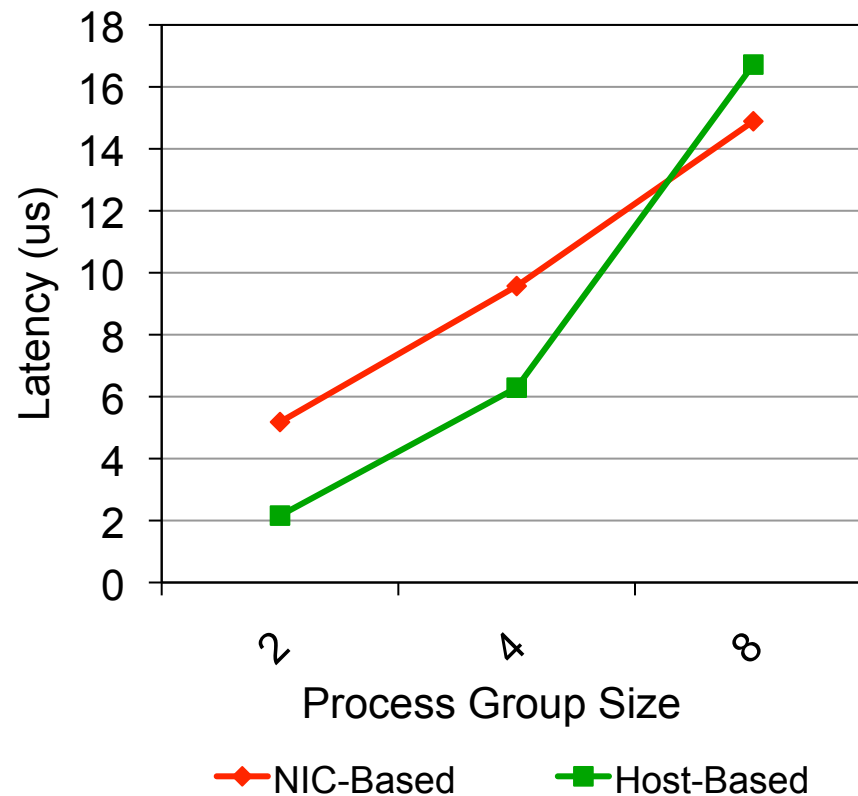
- Offloaded primitives achieve 97% or above overlap percentage

Receive-Reduce Overlap Percentage



- Offloaded collectives can achieve 98% or more overlap

Performance of MPI_Barrier



- Performance of offload based MPI_Barrier is better with increasing group size
- Tests are run on a 8x8=64 core machine with intra-node steps in shared memory

Outline

- Introduction
- Problem Statement
- Design of Communication Primitives
- Performance Evaluation
- Conclusions and Future Work

Conclusions and Future Work

- We describe our approaches towards designing collective communication primitives with good overlap characteristics
- Evaluation reveals that we are able to achieve near perfect (94% - 100%) overlap of computation and communication by using our primitives
- We also see an improvement in performance of up to 5% using the Recv-Replicate primitive for data transfer
- A 10% improvement in performance is seen with our MPI_Barrier implementation for 8 nodes
- The collective primitives along with MPI level collective operations will be available in MVAPICH2 public releases shortly

Thank you!

{subramon, kandalla, surs, panda}@cse.ohio-state.edu



Network-Based Computing Laboratory

<http://mvapich.cse.ohio-state.edu/>