# RDMA Read Based Rendezvous Protocol for MPI over InfiniBand: Design Alternatives and Benefits

**Sayantan Sur** Hyun-Wook Jin Lei Chai

D. K. Panda

**Network Based Computing Lab,**

**The Ohio State University**

# Presentation Outline

- Introduction and Motivation
- Problem Statement
- Detailed Design Description
- Design Evaluation Framework
- Micro-benchmark Level Evaluation
- Application Level Evaluation
- Conclusions and Future Work

# Introduction

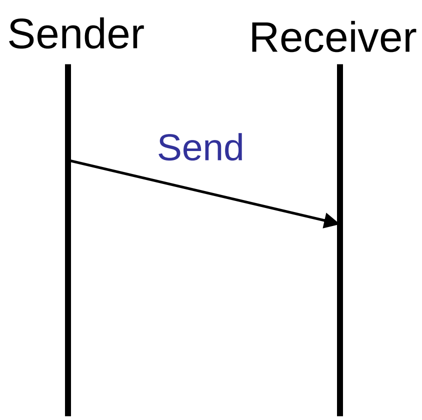
- MPI is a popular parallel programming model
- Offers several point-to-point communication semantics
  - Non-blocking (MPI_Isend, MPI_Irecv …)
  - Blocking (MPI_Send, MPI_Recv …)
  - Synchronous (MPI_Ssend, MPI_Issend …)
- Non-blocking point-to-point communication is hugely popular among application writers

# Why are Non-Blocking Semantics Popular?

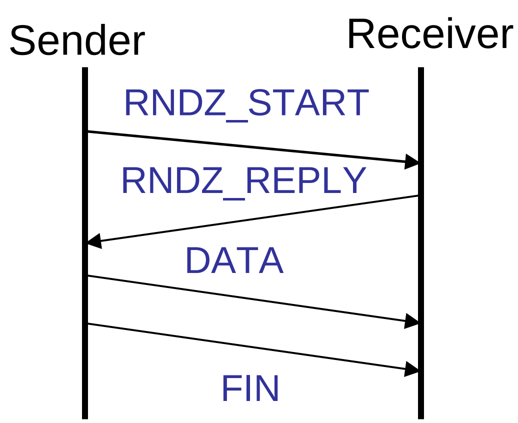
- Sending and receiving processes can progress independently without blocking
- Enables "Computation/Communication" overlap
- Several other parallel programming models feature non-blocking semantics
  - PGAS {UPC, HPF}
  - ARMCI

# Message Passing Protocols

- MPI utilizes two major types of protocols
  - Eager
    - Used for small messages (buffered)
  - Rendezvous
    - Used for large messages (un-buffered)
    - Reduces memory requirement by MPI library




*Eager Protocol*

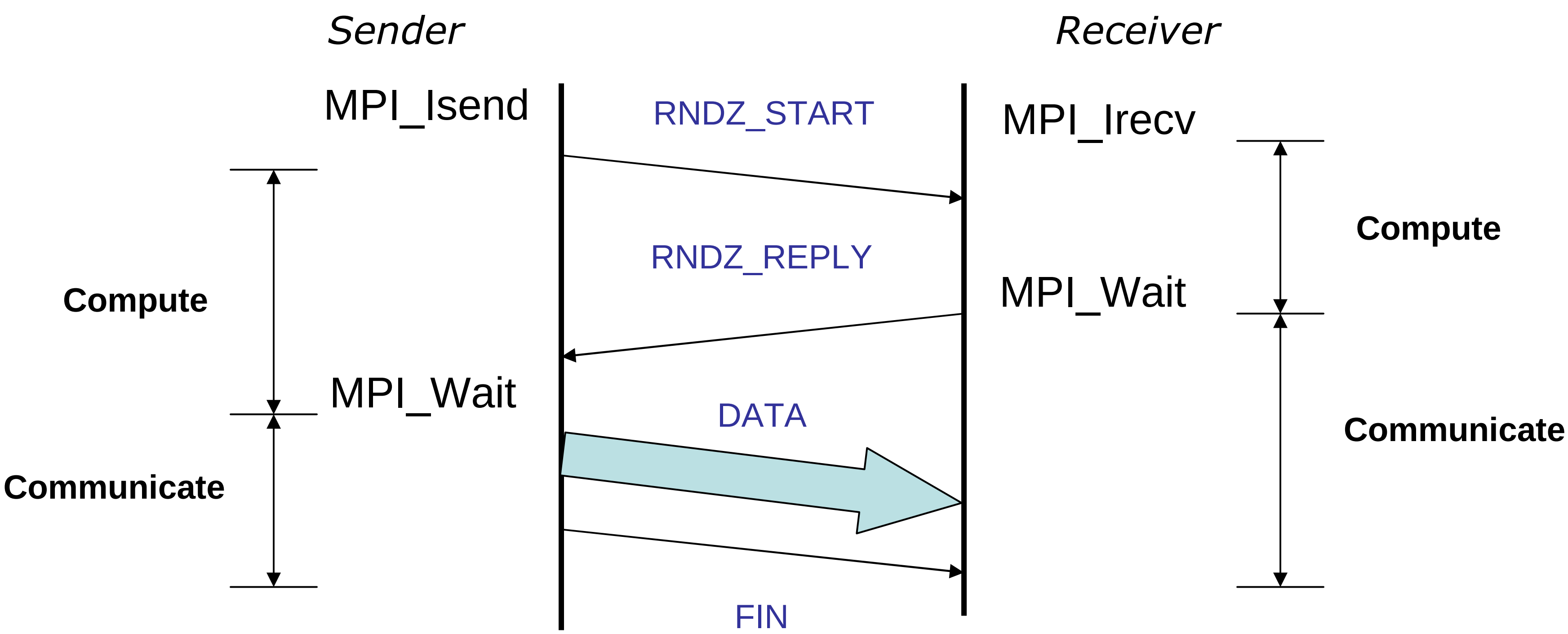



*Rendezvous Protocol*

# Is Overlap Always Possible?

```
/* Compute Large Array */
MPI_Isend(array);
long_compute();
MPI_Wait(send_req);
```

```
MPI_Irecv(array);
long_compute();
MPI_Wait(recv_req);
```



# How can InfiniBand help?

- InfiniBand is an industry standard HPC interconnect
- Very good performance with many features
  - Minimum Latency: ~2us, Peak Bandwidth: ~1500MB/s
  - One sided RDMA (Remote DMA), Atomic operations
  - Hardware multicast, Quality of Service …
- RDMA is a powerful mechanism
  - Zero copy (network can directly DMA from user buffers)
  - No remote side involvement
  - Both Write and Read semantics are supported
- Need to design Rendezvous Protocol which leverages all the novel features for InfiniBand in order to achieve Computation/Communication overlap

# Presentation Outline

- Introduction and Motivation
- Problem Statement
- Detailed Design Description
- Design Evaluation Framework
- Micro-benchmark Level Evaluation
- Application Level Evaluation
- Conclusions and Future Work

# Problem Statement

- Can we design a Rendezvous protocol which can achieve full overlap of computation and communication?
- Can this new protocol reduce the communication time experienced by end applications?

# Presentation Outline

- Introduction and Motivation
- Problem Statement
- Detailed Design Description
- Design Evaluation Framework
- Micro-benchmark Level Evaluation
- Application Level Evaluation
- Conclusions and Future Work

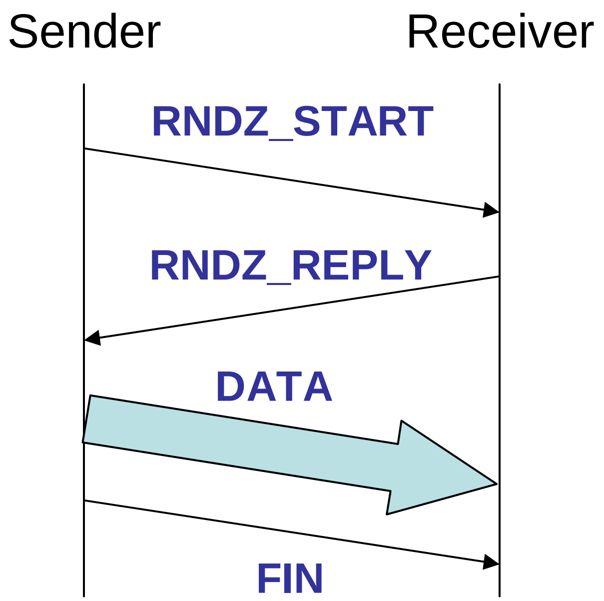
# Design Overview

- Design a new RDMA Read based Rendezvous protocol
  – Minimize control messages
- Trigger "automatic" progress with interrupts
  – Interrupts are costly (~2 times round-trip latency)
  – Reduce interrupts using
    - Selective Interrupts
    - Interrupt suppression
    - Dynamic Interrupt Requests
- Hybrid Communication Progress
  – Maintain polling nature (where possible) of MPI progress to allow low latency

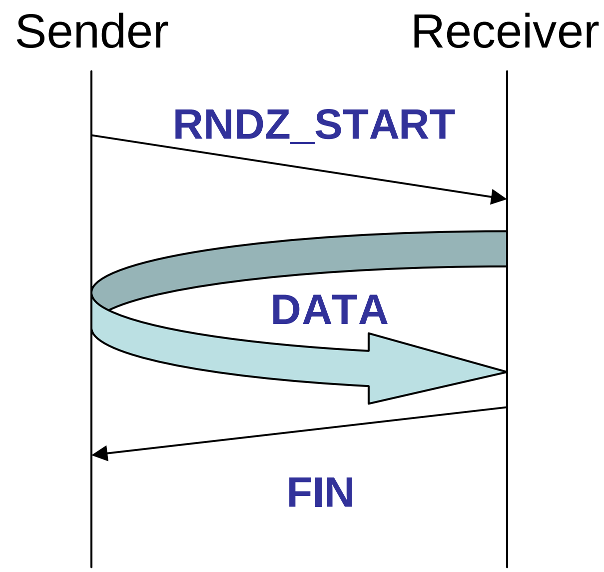
# Rendezvous Protocol: Design Alternatives

- MPI specification states that receiver may post a buffer larger than actual message
- Only sender knows the actual size of the message and can make the optimal decision on the protocol to be used:
  - Eager (buffered) if message is small
  - Rendezvous (un buffered) if message is large
- The Rendezvous protocol must be initiated by sender

# RDMA Write Vs. RDMA Read



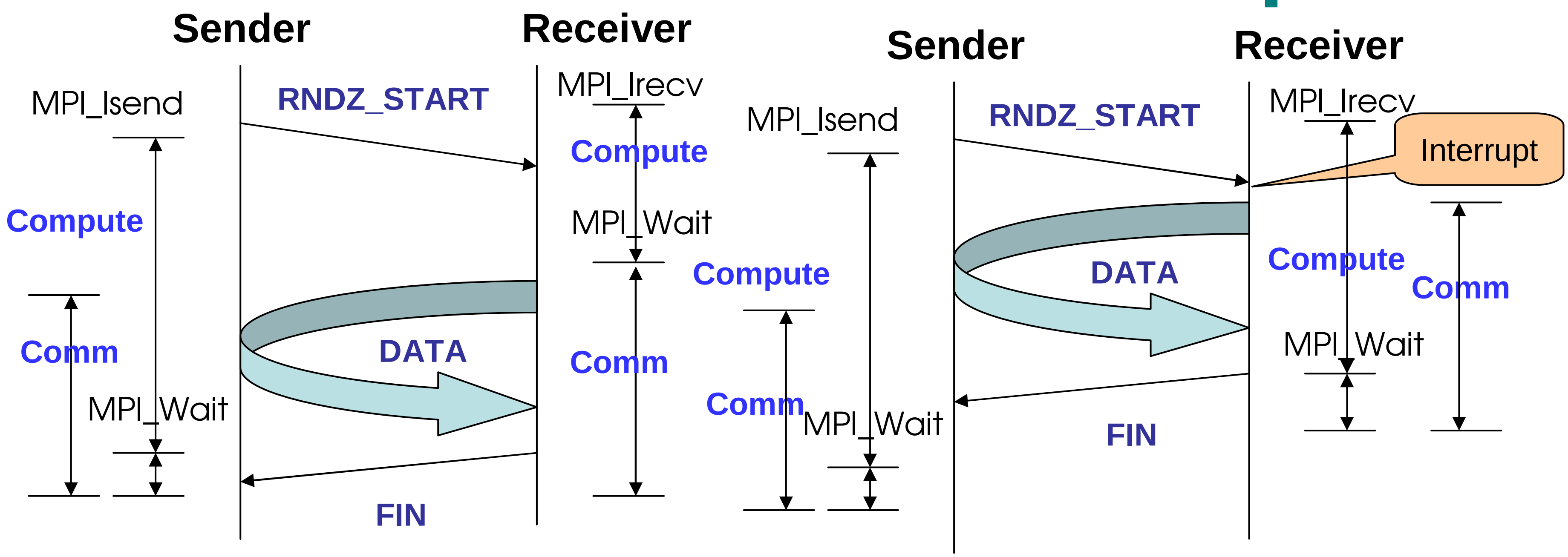

*RDMA Write Based Protocol*




*RDMA Read Based Protocol*

- RDMA Read based protocol need less control messages
- Sender can embed its buffer information with RNDZ_START message

# RDMA Read with Interrupt




*RDMA Read Based Protocol*

*RDMA Read with Interrupt Based Protocol*

- Interrupt triggers communication progress
- This enables overlap of computation and communication on receiver side
- Need to reduce overhead caused by Interrupts

# Interrupt Reduction Techniques

- Selective Interrupts
  - Only RNDZ_START messages cause interrupts
- Interrupt Suppression
  - Interrupt handler once awake, handles as many RNDZ_START messages it can find
  - Back-to-back messages don't cause interrupts
- Dynamic Interrupt Requests
  - Interrupts enabled only when large receives are posted
  - Unexpected RNDZ_START messages don't cause interrupts

# Hybrid Communication Progress

- Progress engine has an impact on MPI performance
- Hybrid progress engine allows two progress threads to simultaneously execute
- In event of no "progress critical" events, no extra interrupts are generated
- Progress engine was re-designed to be thread safe

| | Latency | Progress Rate |
|---|---|---|
| Interrupt Based | High | Good |
| Polling Based | Low | Bad |
| **Hybrid** | **Low** | **Good** |

# Presentation Outline

- Introduction and Motivation
- Problem Statement
- Detailed Design Description
- Design Evaluation Framework
- Micro-benchmark Level Evaluation
- Application Level Evaluation
- Conclusions and Future Work

# OSU MPI over InfiniBand

- High Performance Implementations
  - MPI-1 (MVAPICH)
  - MPI-2 (MVAPICH2)
- Open Source (BSD licensing)
- Has enabled a large number of production IB clusters all over the world to take advantage of IB
  - Largest being Sandia Thunderbird Cluster (4000 node with 8000 processors)
- Have been directly downloaded and used by more than 335 organizations worldwide (in 33 countries)
  - Time tested and stable code base with novel features
- Available in software stack distributions of many vendors
- Available in the OpenIB/gen2 stack
- More details at

  http://nowlab.cse.ohio-state.edu/projects/mpi-iba/

# Evaluation Framework

- Proposed designs were incorporated in MVAPICH 0.9.5
  - RDMA Write (RDMA-W)
  - RDMA Read (RDMA-R)
  - RDMA Read with Interrupt (RDMA-RI)
- RDMA-R protocol is available from version 0.9.6
- RDMA-RI protocol will be available from version 0.9.8

# Presentation Outline

- Introduction and Motivation
- Problem Statement
- Detailed Design Description
- Design Evaluation Framework
- Micro-benchmark Level Evaluation
- Application Level Evaluation
- Conclusions and Future Work

# Experimental Evaluation

- Micro-benchmark tests
  - Computation/Communication overlap performance
  - Communication progress performance
    - Measured with time stamps from overlap test
- Evaluation platforms
  - Cluster A: 8 Dual 3.0 GHz SMP; 2GB RAM; PCI-X
  - Cluster B: 32 Dual 2.6 GHz SMP; 2GB RAM; PCI-X
- Mellanox InfiniBand adapters (MT23108)
- Mellanox 144 port InfiniBand switch (MTS14400)

# Micro-benchmark Tests

- Sender Overlap:

T W

```
MPI_Isend(array);
compute();
MPI_Wait(send_req);
```

```
MPI_Recv(array);
```

- Receiver Overlap:

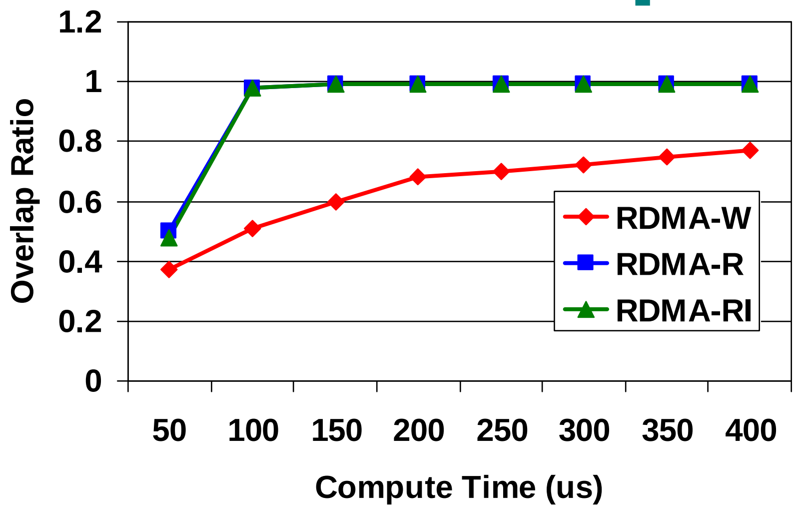
```
MPI_Send(array);
```

T W

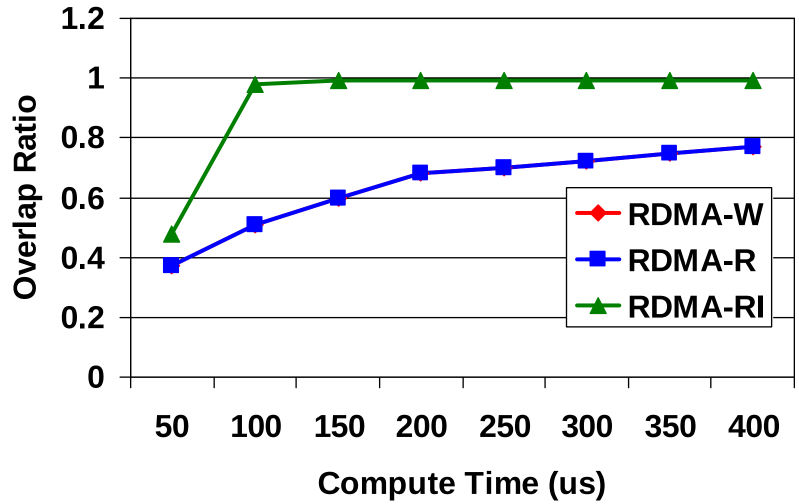
```
MPI_Irecv(array);
compute();
MPI_Wait(recv_req);
```

- Computation/Communication ratio is: **W/T**

# Computation/Communication Overlap Performance



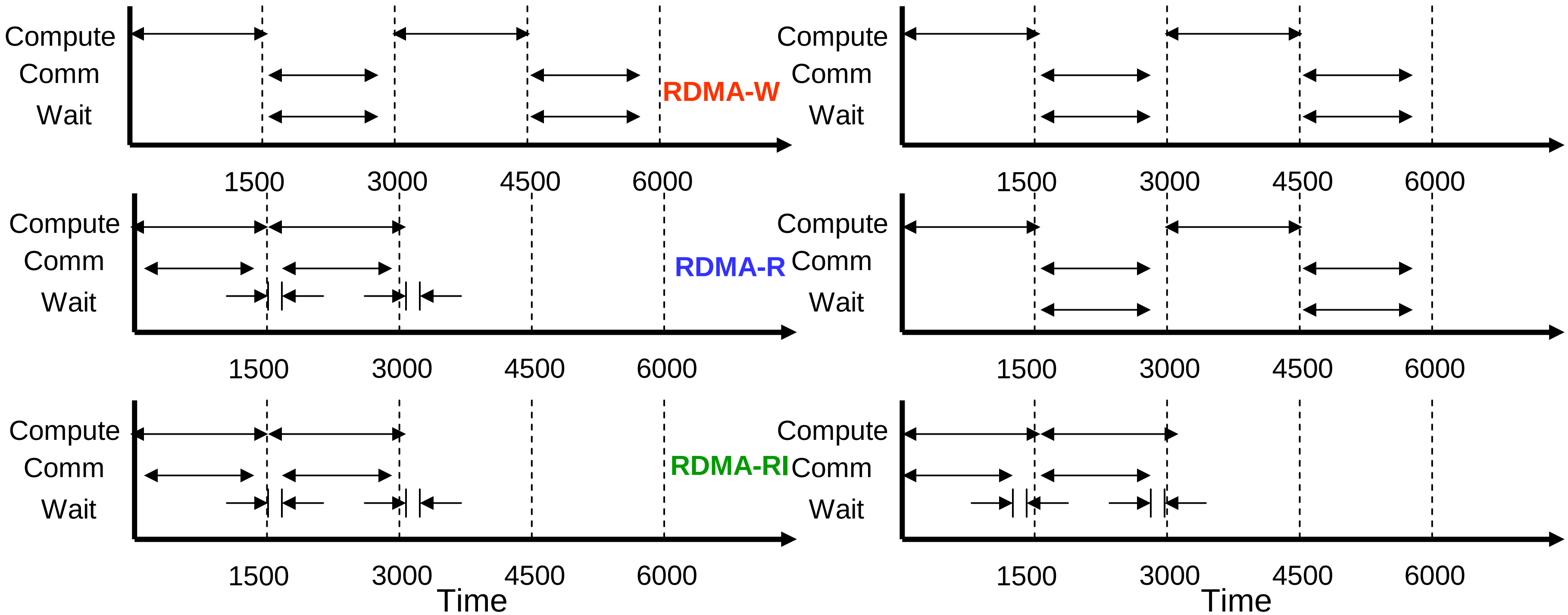

*Sender Overlap Performance (64KB)*




*Receiver Overlap Performance (64KB)*

- Sender Overlap:
  - RDMA-W has poor overlap due to inability to discover the RNDZ_REPLY message till computation is over
  - RDMA-R and RDMA-RI achieve nearly complete overlap
- Receiver Overlap:
  - RDMA-W and RDMA-R have poor overlap due to their inability to discover the rendezvous control (RNDZ_REPLY and RNDZ_START) messages respectively
  - RDMA-RI achieves nearly complete overlap

# Communication Progress Performance




*Sender Overlap*　　*Receiver Overlap*

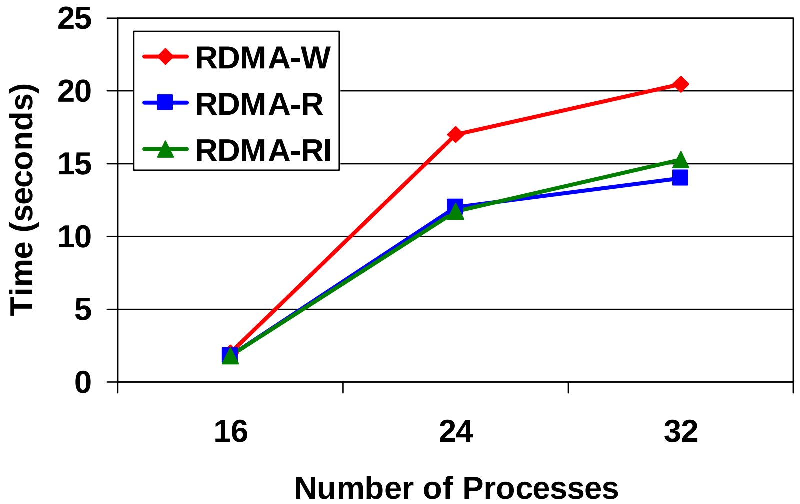
- Time stamps are taken during sender/receiver overlap tests when application enters compute/communication phase and from within MPI library when application enters MPI_Wait
- The RDMA-RI can achieve 50% faster communication in both sender and receiver overlap tests

# Presentation Outline

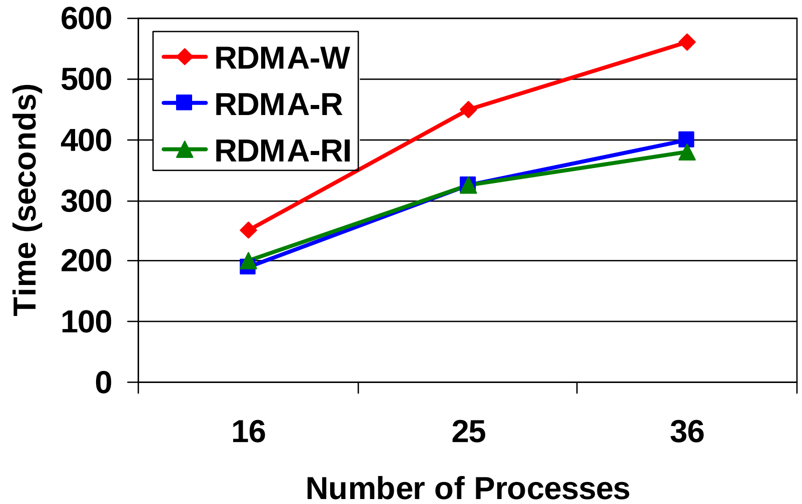
# Application level Evaluation

- Two well known applications
  - High Performance Linpack (HPL)
  - NAS Scalar Pentadiagonal (SP)
- Predominantly use MPI_Isend/Irecv
- Time spent in MPI library is profiled using mpiP (a lightweight MPI profiling tool)
- This wait time can be effectively utilized by application to compute rather than just waiting for network operations to complete

# Application Level Results




*MPI_Wait time for HPL*




*MPI_Wait time for NAS SP*

- Wait time for HPL
  - Reduced by ~30% for 32 processes by RDMA-R and RDMA-RI
- Wait time for NAS SP
  - Reduced by ~28% for 36 processes by RDMA-R and RDMA-RI

# Presentation Outline

- Introduction and Motivation
- Problem Statement
- Detailed Design Description
- Design Evaluation Framework
- Micro-benchmark Level Evaluation
- Application Level Evaluation
- Conclusions and Future Work

# Conclusions and Future Work

- New designs can achieve nearly complete overlap of computation/communication
- Communication progress can be sped up by 50%
- Application (HPL, NAS SP) wait times reduced by 30% and 28% respectively
- Unique study of Rendezvous Protocol and its effect on Computation/Communication overlap using RDMA
- Future Work
  - More exhaustive application oriented study on larger scale InfiniBand cluster

# Acknowledgements

Our research is supported by the following organizations

- Current Funding support by

Office of Science U.S. DEPARTMENT OF ENERGY
NSF
Mellanox Technologies
Cisco Systems Empowering the Internet Generation
intel
Linux Networx proven
NetApp
Sun microsystems

- Current Equipment support by

Mellanox Technologies
intel
AMD
Sun microsystems
APPRO HPC Cluster Solutions
IBM
Microway
PathScale
SilverStorm

# Web Pointers

Network Based Computing Laboratory

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

MVAPICH Web Page

http://nowlab.cse.ohio-state.edu/projects/mpi-iba/