Scalable High Performance Message Passing over Infiniband for Open MPI

Andrew Friedley, **Torsten Hoefler**
Matthew L. Leininger, Andrew Lumsdaine
December 12, 2007

Motivation

- MPI is the *de facto* standard for HPC
- InfiniBand growing in popularity
  - Particularly on large-scale clusters
  - June 2005 Top500: 3% of machines
  - November 2007 Top500: 24% of machines
- Clusters growing in size
  - Thunderbird, 4,500 node InfiniBand

InfiniBand (IB) Architecture

- Queue Pair concept (QP)
  - Send a message by posting work to a queue
  - Post receive buffers to a queue for use by hardware
- Completion Queue
  - Signals local send completion
  - Returns receive buffers filled with data
- Shared Receive Queue
  - Multiple QPs share a single receive queue
  - Reduces network resources

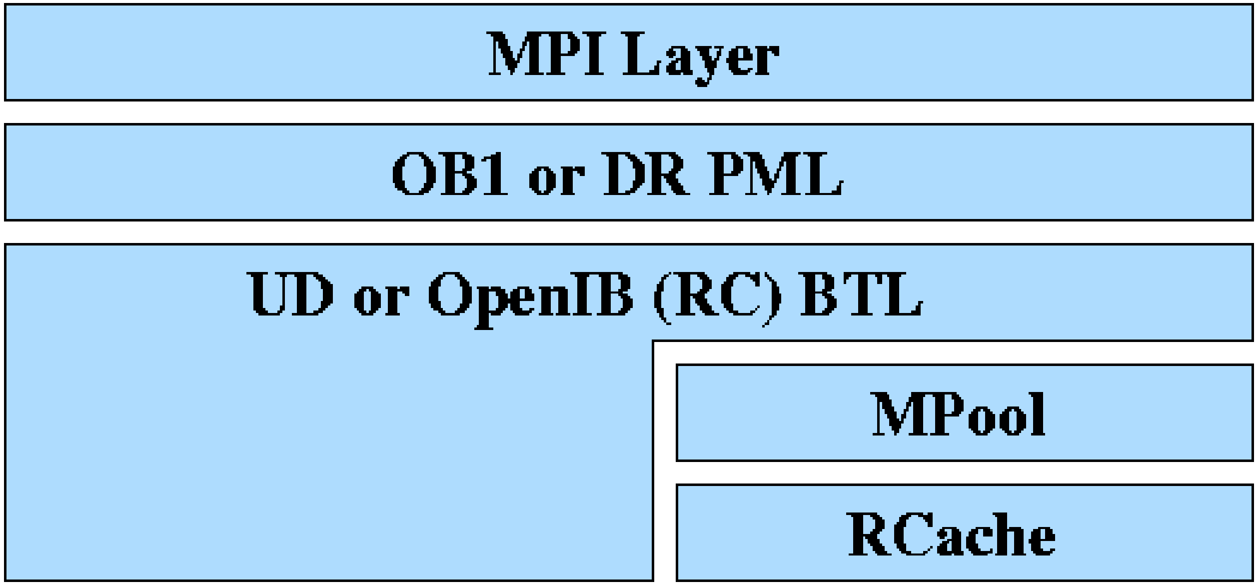
Reliable Connection (RC) Transport

- Traditional approach for MPI communication over InfiniBand
- Point-to-point connections
- Send/receive and RDMA semantics
- One queue pair per connection
  - Out-of-band handshake required to establish
- Memory requirements scale with number of connections
  - Memory buffer requirements reduced by using shared receive queue

Unreliable Datagram Transport

- Requires software (MPI) reliability protocol
  - Memory-to-Memory, not HCA-to-HCA
- Message size limited to network MTU
  - 2 kilobytes on current hardware
- Connectionless model
  - No setup overhead
  - One QP can communicate with any peer
  - Except for address information, memory requirement is constant

Open MPI Modular Component Architecture



- Framework consists of many components
- Component is instantiated into modules

PML Components

- OB1
  - Implements MPI point-to-point semantics
  - Fragmentation and scheduling of messages
  - Optimized for performance in common use
- Data Reliability (DR)
  - Extends OB1 with network fault tolerance
    - Message reliability protocol
    - Data checksumming

Byte Transport Layer (BTL)

- Components are interconnect specific
  - TCP, shmem, GM, OpenIB, uDAPL, et. al.
- Send/Receive Semantics
  - PML fragments, not MPI messages
- RDMA Put/Get Semantics
  - Optional – not always supported!

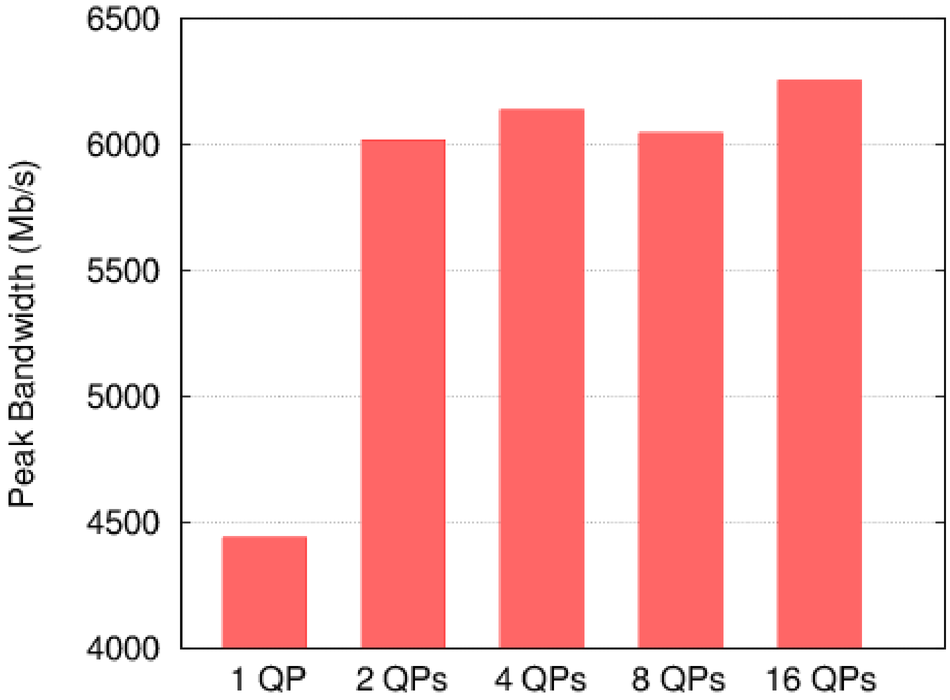
Byte Transport Layer (BTL)

- Entirely Asynchronous
  - Blocking is not allowed
  - Progress made via polling
- Lazy connection establishment
  - Point-to-point connections established as needed
- Option to multiplex physical interfaces in one module, or to provide many modules
- No MPI semantics
  - Simple, peer-to-peer data transfer operations

UD BTL Implementation

- RDMA not supported
- Use with DR PML
- Receiver buffer management
  - Messages dropped if no buffers available
  - Allocate a large, static pool
  - No flow control in current design

Queue Pair Striping

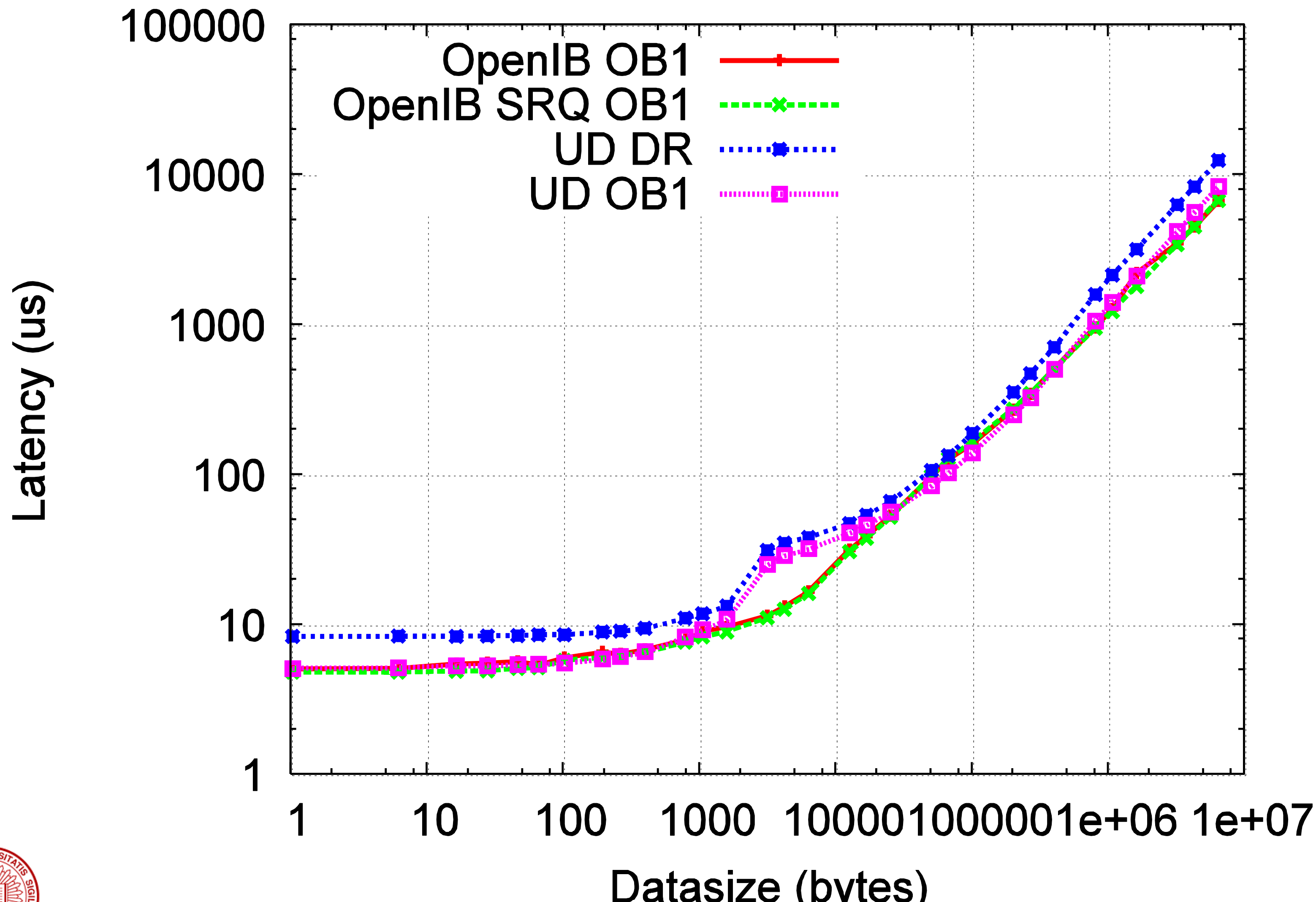


- Splitting sends across multiple queue pairs increases bandwidth
- Receive buffers still posted to one QP

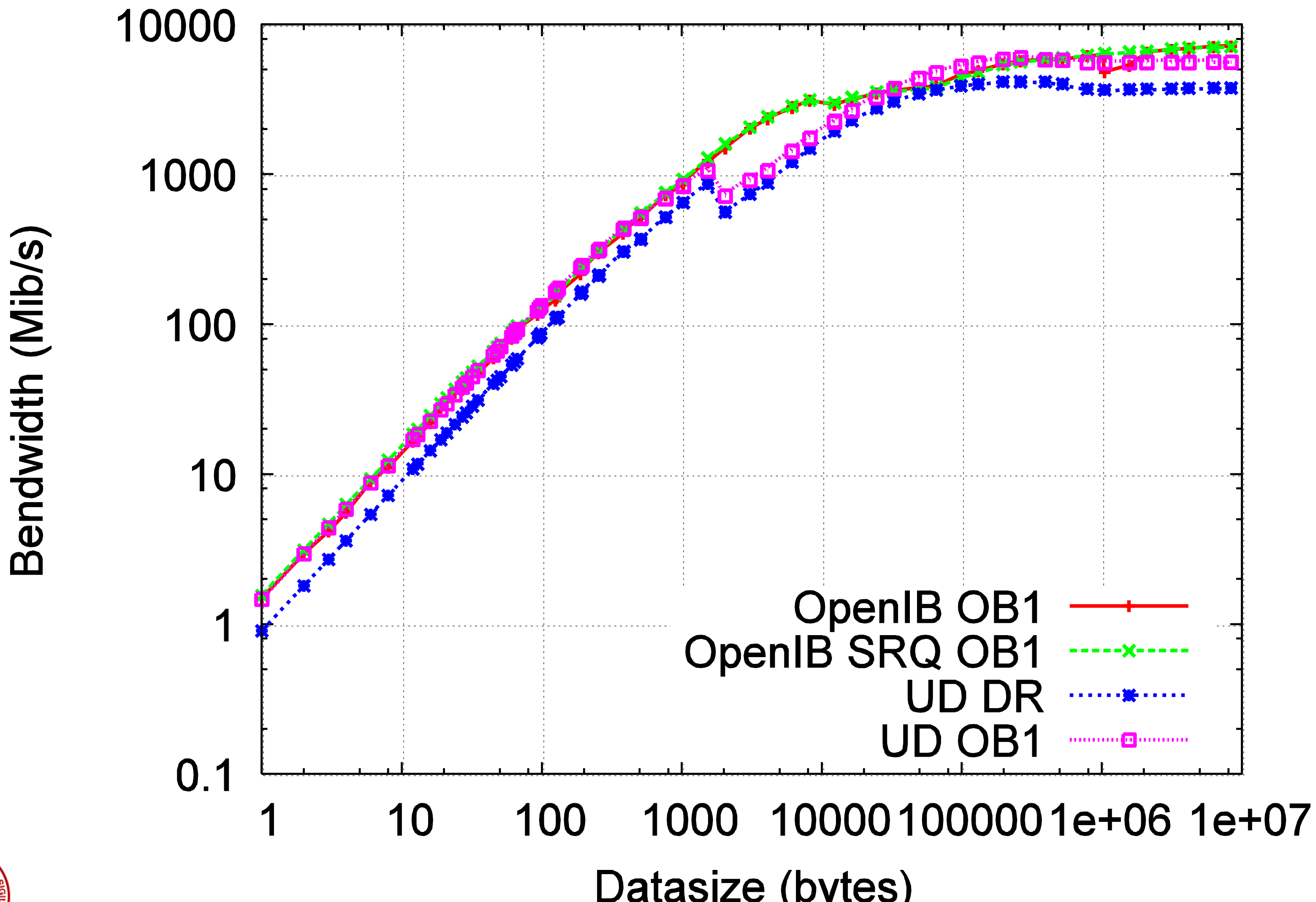
Results

- LLNL Atlas
  - 1,152 quad dual-core (8 core) nodes
  - InfiniBand DDR network
- Open MPI trunk r16080
  - Code publicly available since June 2007
- UD results with both DR and OB1
  - Compare DR reliability overhead
- RC with and without Shared Receive Queue

NetPIPE Latency



NetPIPE Bandwidth

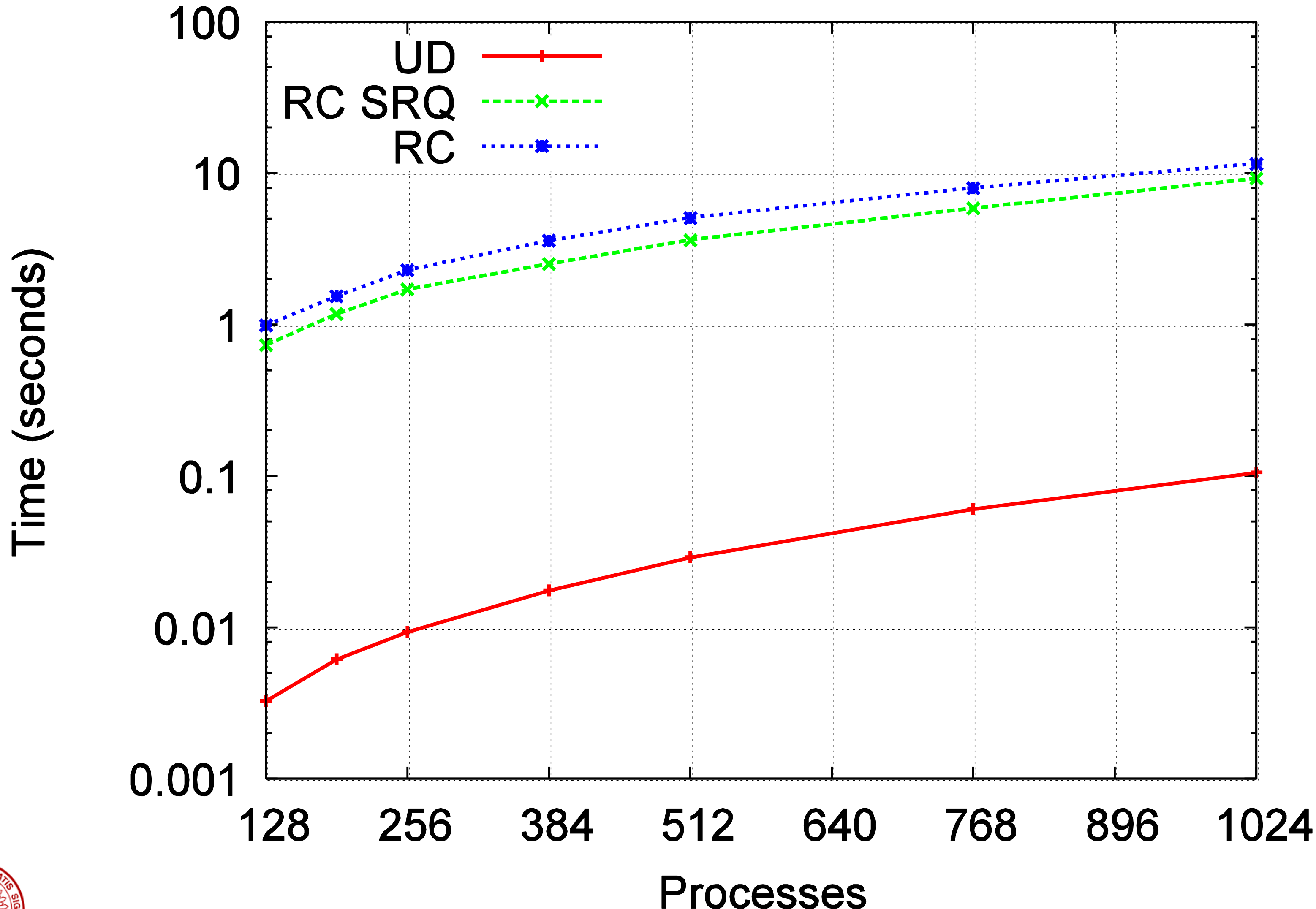


pervasivetechnologylabs
AT INDIANA UNIVERSITY

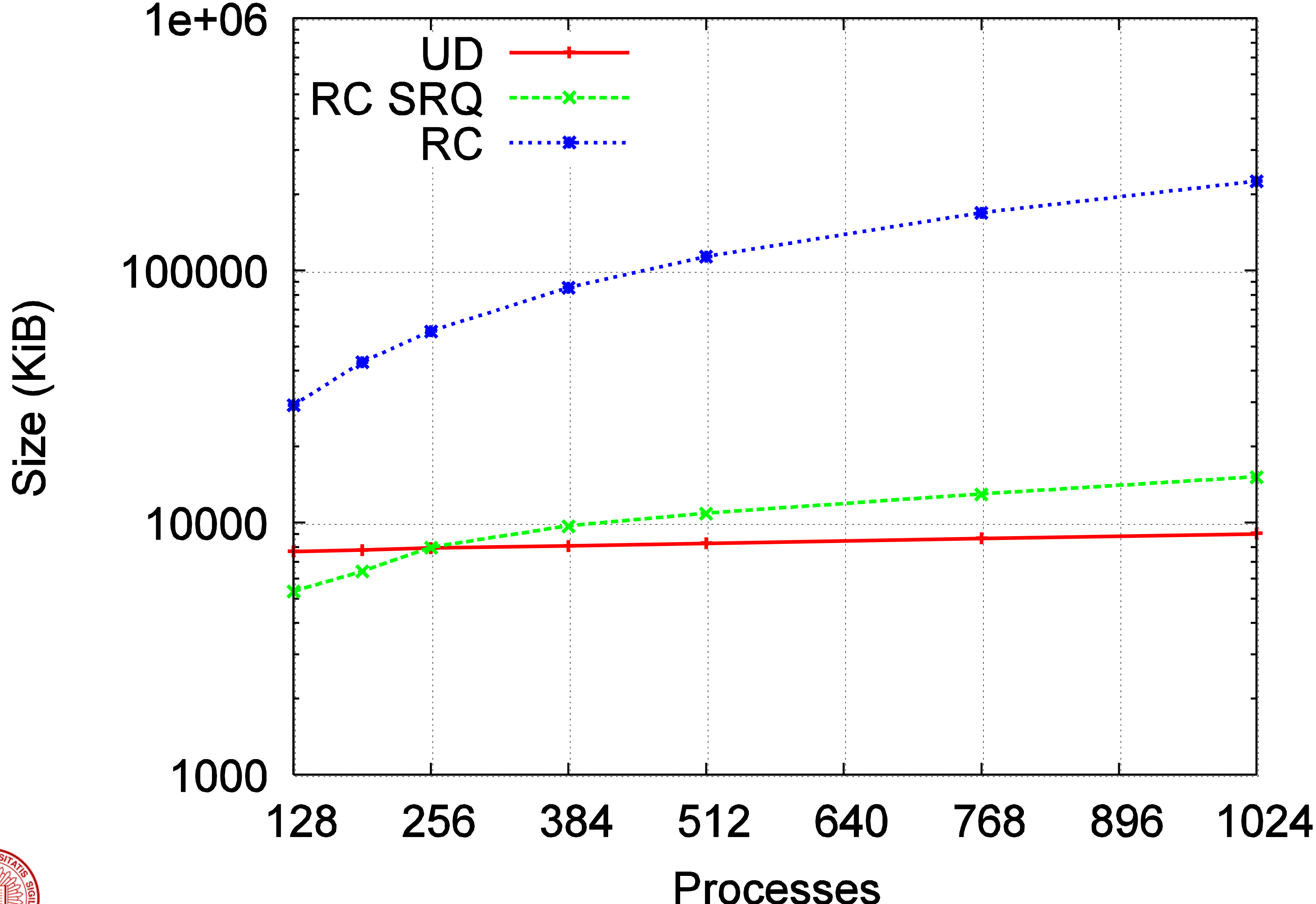
Allconn Benchmark

- Each MPI process sends a 0-byte message to every other process
  - Done in a ring-like fashion to balance load
- Measures time required to establish connections between all peers
  - For connection-oriented networks, at least
  - UD should only reflect time required to send messages – no establishment overhead

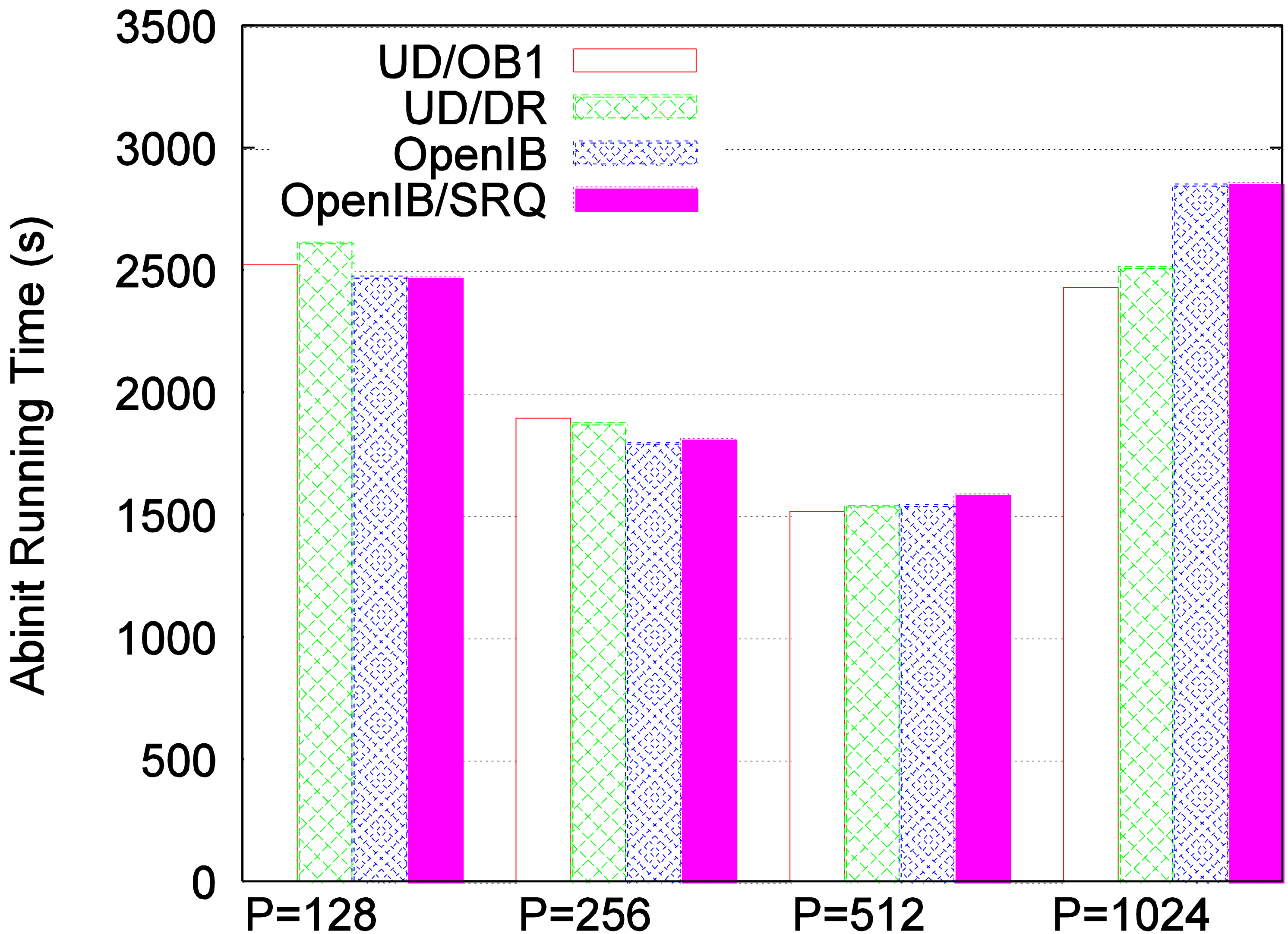
Allconn Startup Overhead



Allconn Memory Overhead

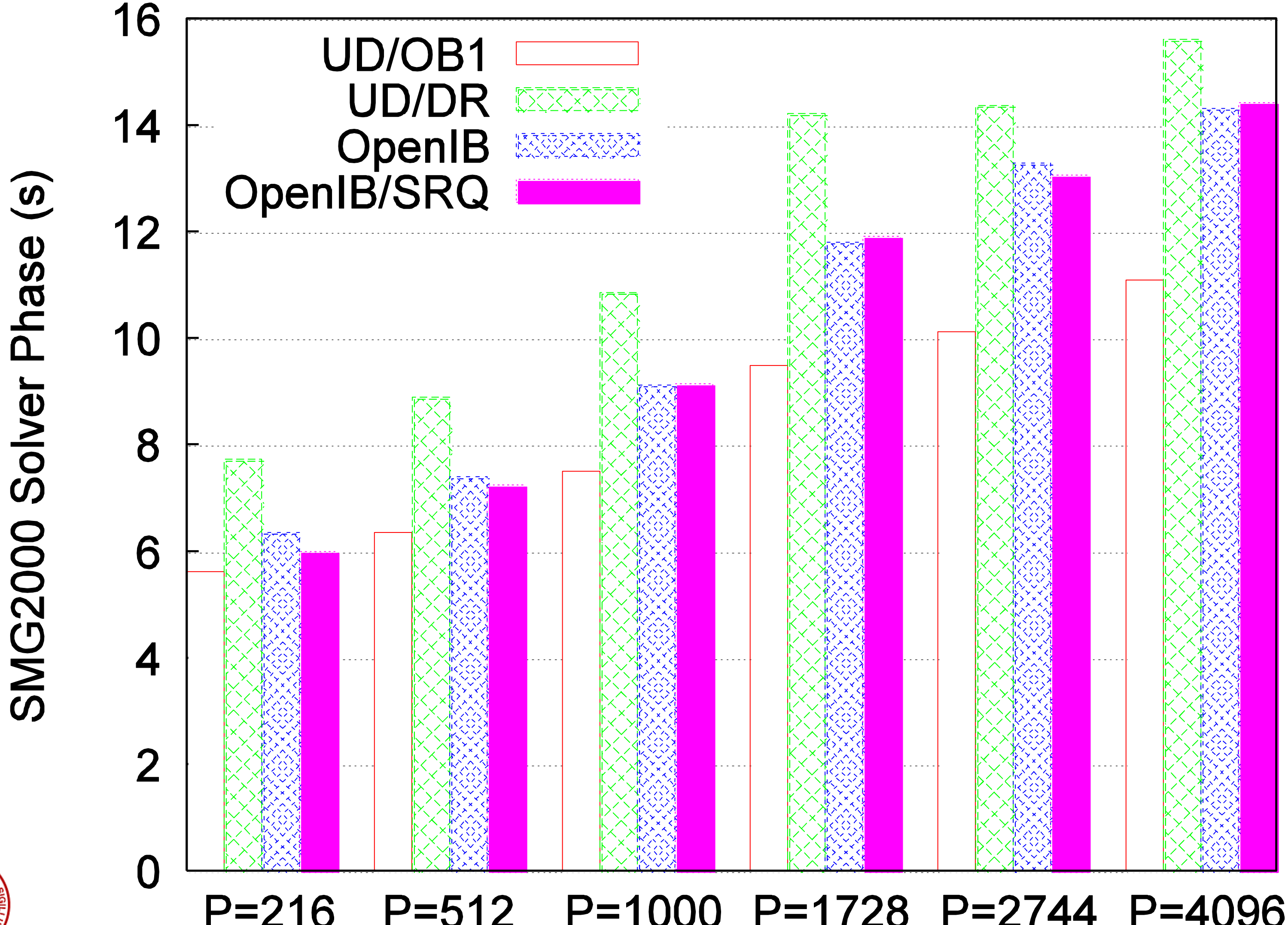


ABINIT



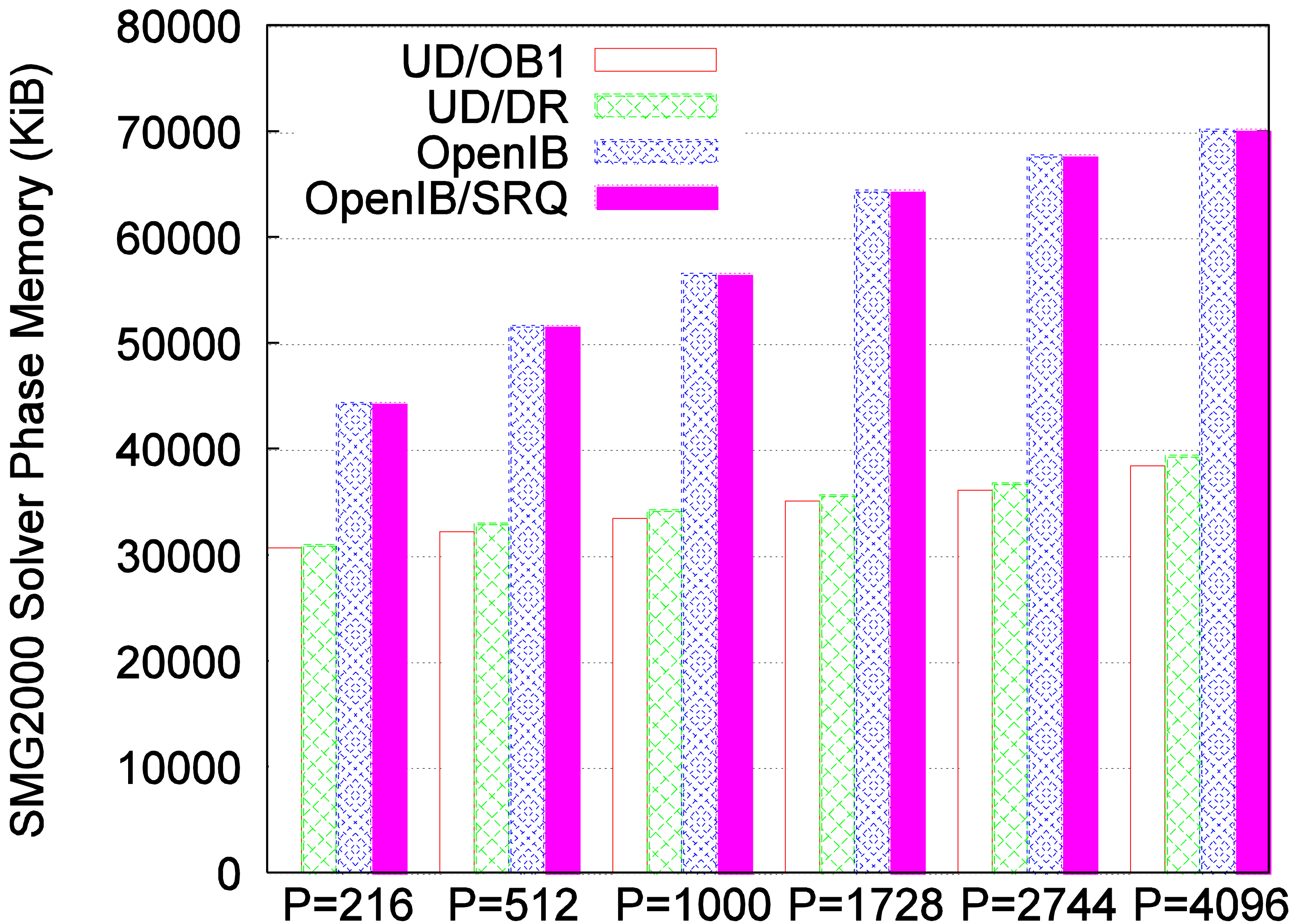
Abinit Running Time (s)
3500
3000
2500
2000
1500
1000
500
0
UD/OB1
UD/DR
OpenIB
OpenIB/SRQ
P=128
P=256
P=512
P=1024

SMG2000 Solver



pervasivetechnologylabs
AT INDIANA UNIVERSITY

SMG2000 Solver Memory



pervasivetechnologylabs
AT INDIANA UNIVERSITY

Conclusion

- UD is an excellent alternative to RC
  - Significantly reduced memory requirements
    - More memory for the application
  - Minimal startup/initialization overhead
    - Helps with job turnaround on large, busy systems
  - Advantage increases as scale increases
    - Clusters will continue to increase in size
- DR-based reliability incurs penalty
  - Minimal some some applications (ABINIT), significant for others (SMG2000)

Future Work

- Optimized reliability protocol in the BTL
  - Initial implementation working right now
  - Much lower latency impact
  - Bandwidth optimization in progress
- Improved flow control & buffer management
  - Hard problem

Flow Control Problems

- Lossy Network
  - No guarantee flow control signals are received
  - Probabilistic approaches are required
- Abstraction barrier
  - PML hides packet loss from BTL
  - Message storms are expected by PML, not BTL
- Throttling mechanisms
  - Limited ability to control message rate
- Who do we notify when congestion occurs?

Flow Control Solutions

- Use throttle signals instead of absolute credit counts
- Maintain a moving average of receive completion rate
- Enable/disable endpoint striping to throttle message rate
- Use multicast to send throttle signals
  - All peers receive information
  - Scalable?