Network Offloaded Hierarchical Collectives Using ConnectX-2's CORE-Direct

Noam Bloch

Mellanox Technologies, LTD

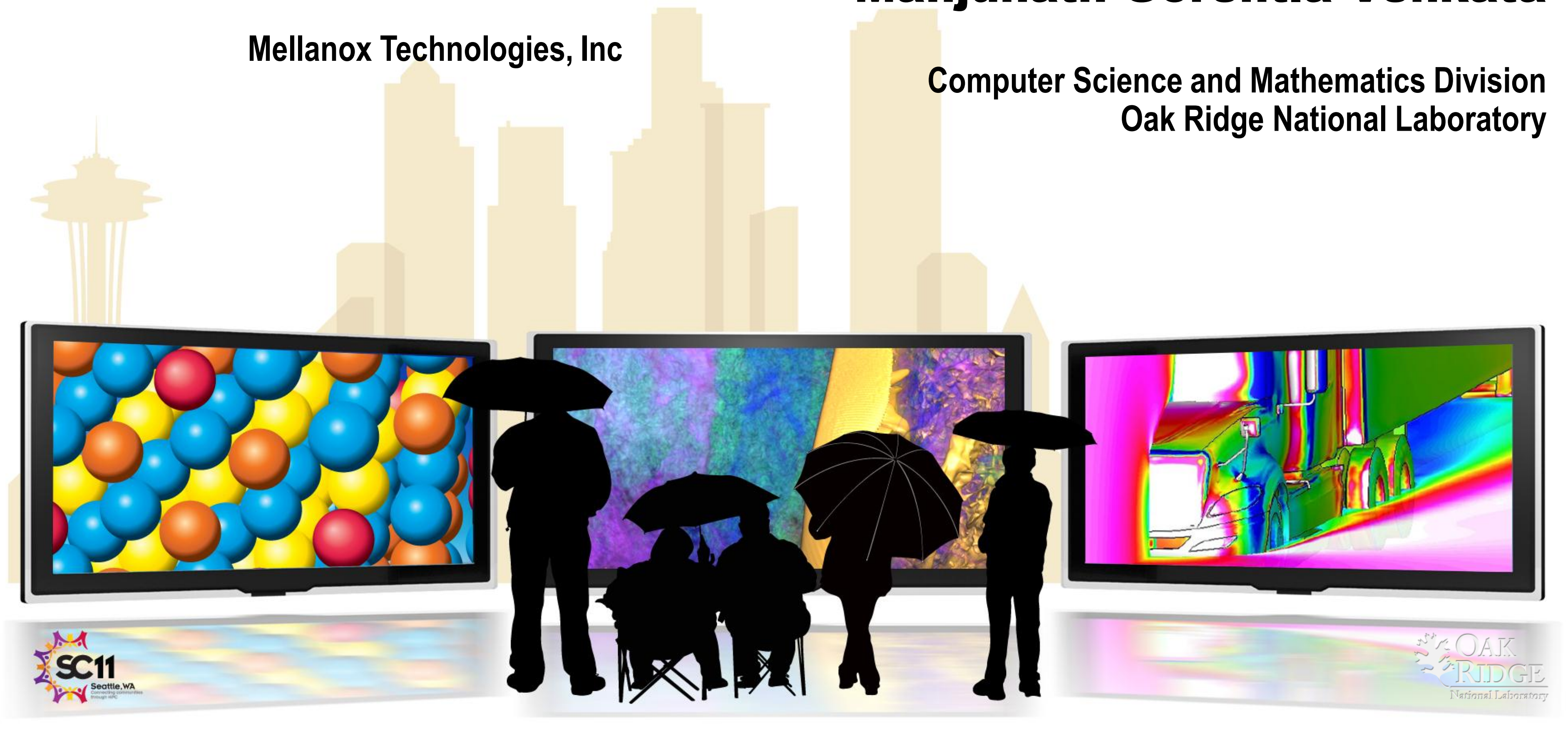
Gilad Shainer

Mellanox Technologies, Inc

Project members

Richard L. Graham
Pavel Shamis
Joshua S. Ladd
Manjunath Gorentla Venkata

Computer Science and Mathematics Division
Oak Ridge National Laboratory



Acknowledgments

- U.S. Department of Energy ASCR FASTOS program
- HPC Advisory Council (computer resources)
  - www.hpcadvisorycouncil.com

Outline

- Problems being addressed
- InfiniBand overview
- New InfiniBand capabilities
- Software design for collective operations
- Results

Problems being addressed – collective operations

- Communication characteristics at scale
- Overlapping computation with communication—true asynchronous communications
  - **Goal:** Avoid using the CPU for communication processing
- System noise
- Application skew

➔ Scalability

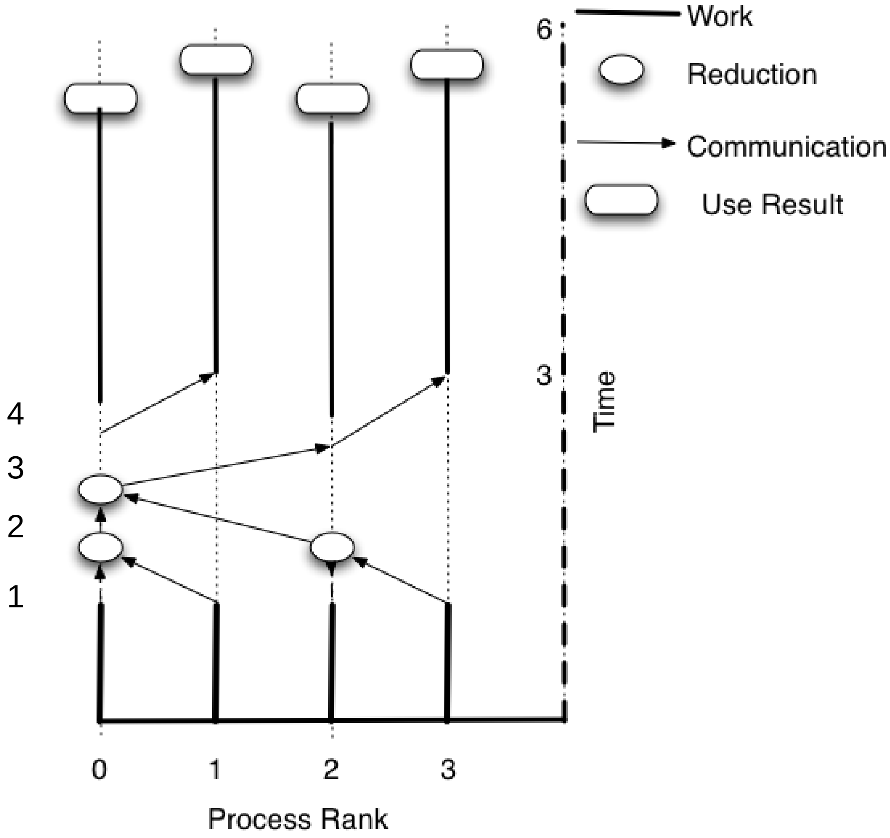
- Collective communication performance

Collective communications

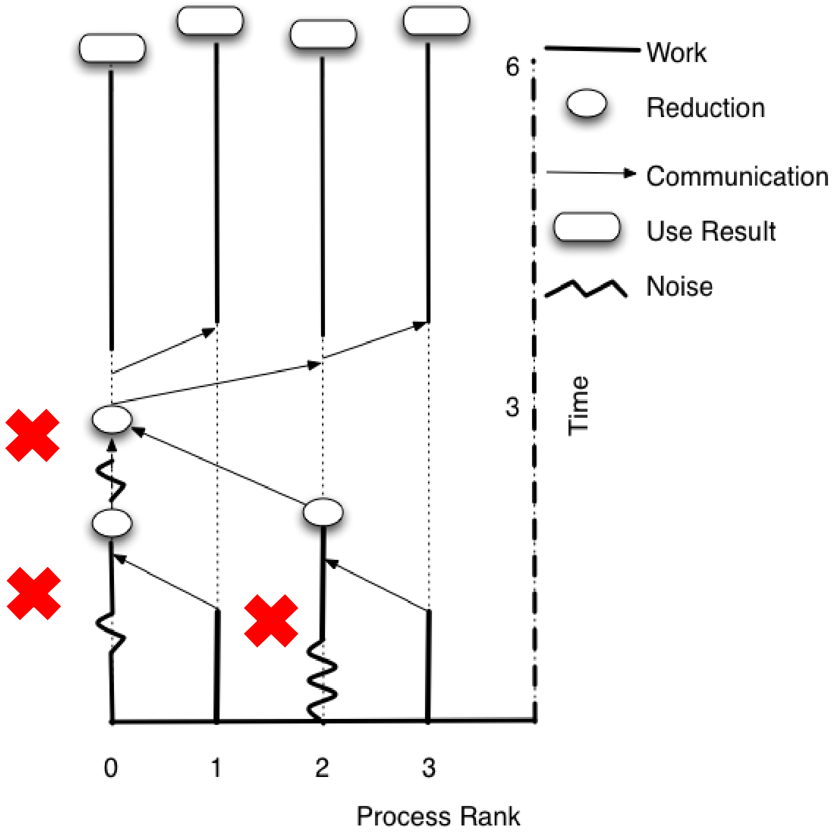
- Communication pattern involving multiple processes (in MPI, all ranks in the communicator are involved)
- Optimized collectives involve a communicator-wide data-dependent communication pattern
- Data needs to be manipulated at intermediate stages of a collective operation
- Collective operations limit application scalability
- Collective operations magnify the effects of system noise

Scalability of collective operations

Ideal Algorithm

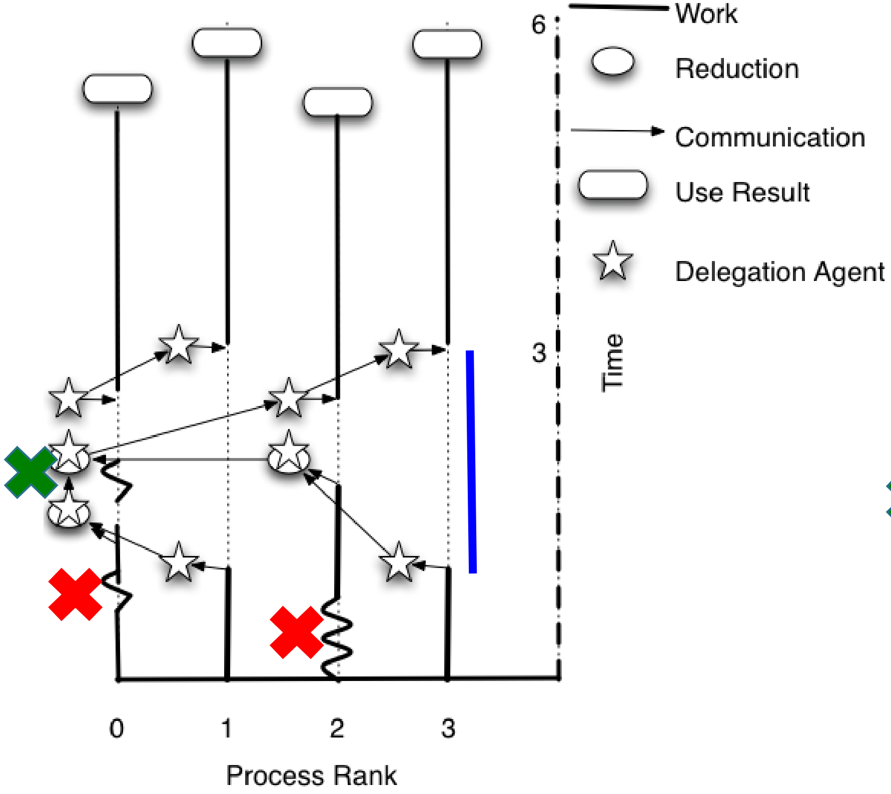


Impact of System Noise

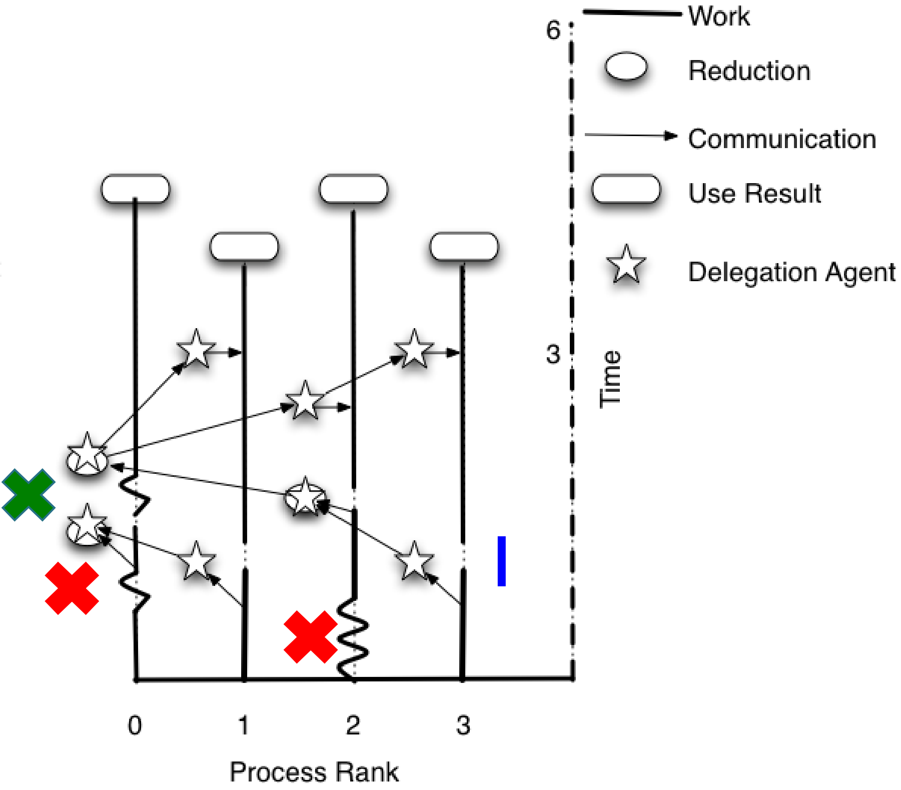


Scalability of collective operations II

Offloaded Algorithm



Nonblocking Algorithm



▎- Communication processing

InfiniBand collective offload – key idea

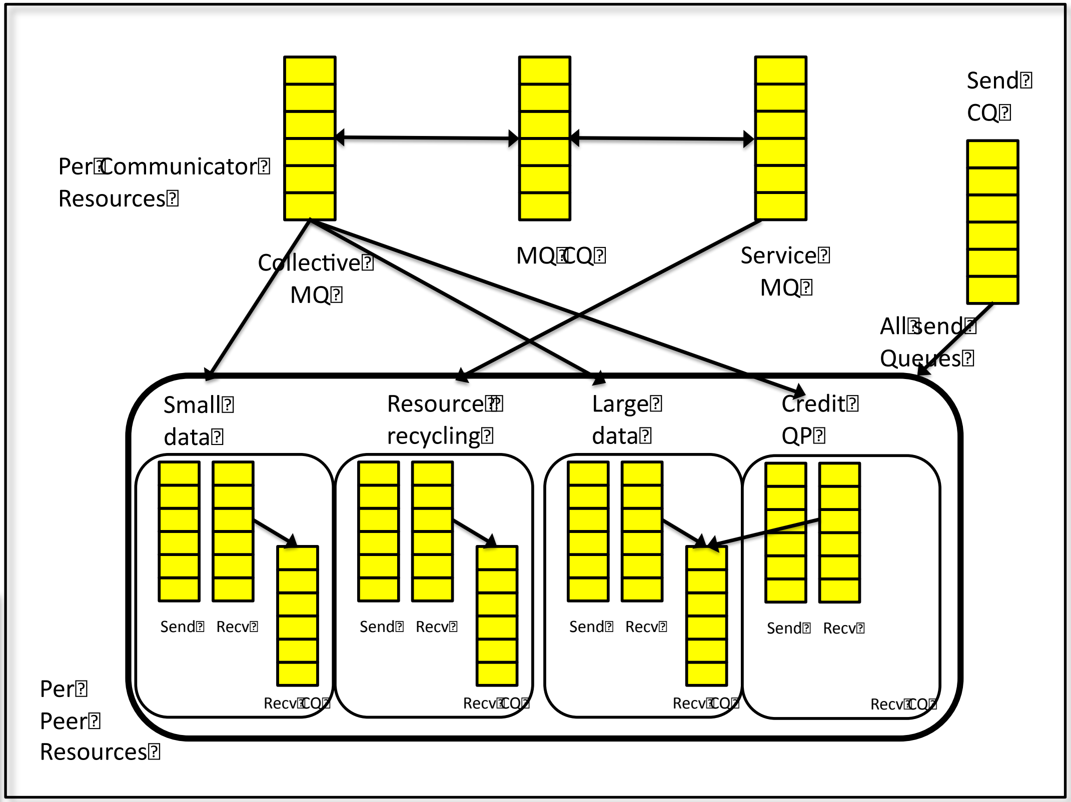
- Create local description of the communication patterns
- Hand the description to the HCA
- Manage collective communications at the network level
- Poll for collective completion
- Add new support for
  - Synchronization primitives (hardware)
    - Send Enable task
    - Receive Enable task
    - Wait task
  - Multiple Work Request
    - A sequence of network tasks
  - Management Queue

InfiniBand hardware changes

- Tasks defined in the current standard
  - Send
  - Receive
  - Read
  - Write
  - Atomic

- New support
  - Synchronization primitives (hardware)
    - Send Enable task
    - Receive Enable task
  - Calc Operations
- Wait task
  - Multiple Work Request
    - A sequence of network tasks
- Management Queue

MPI queue design



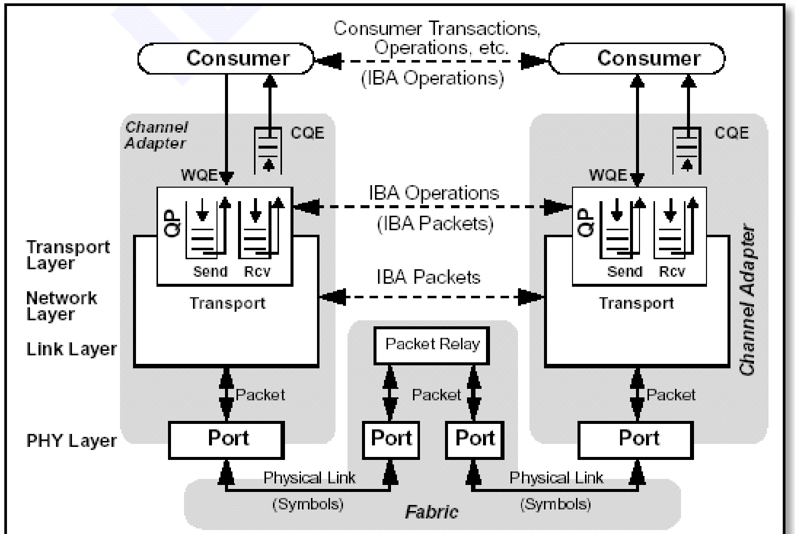
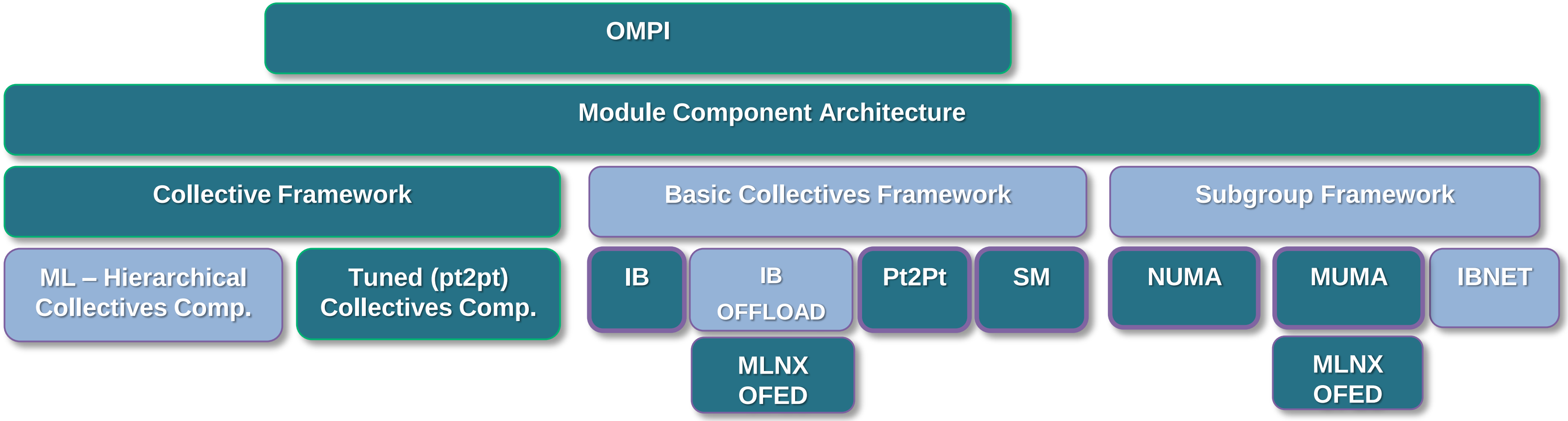
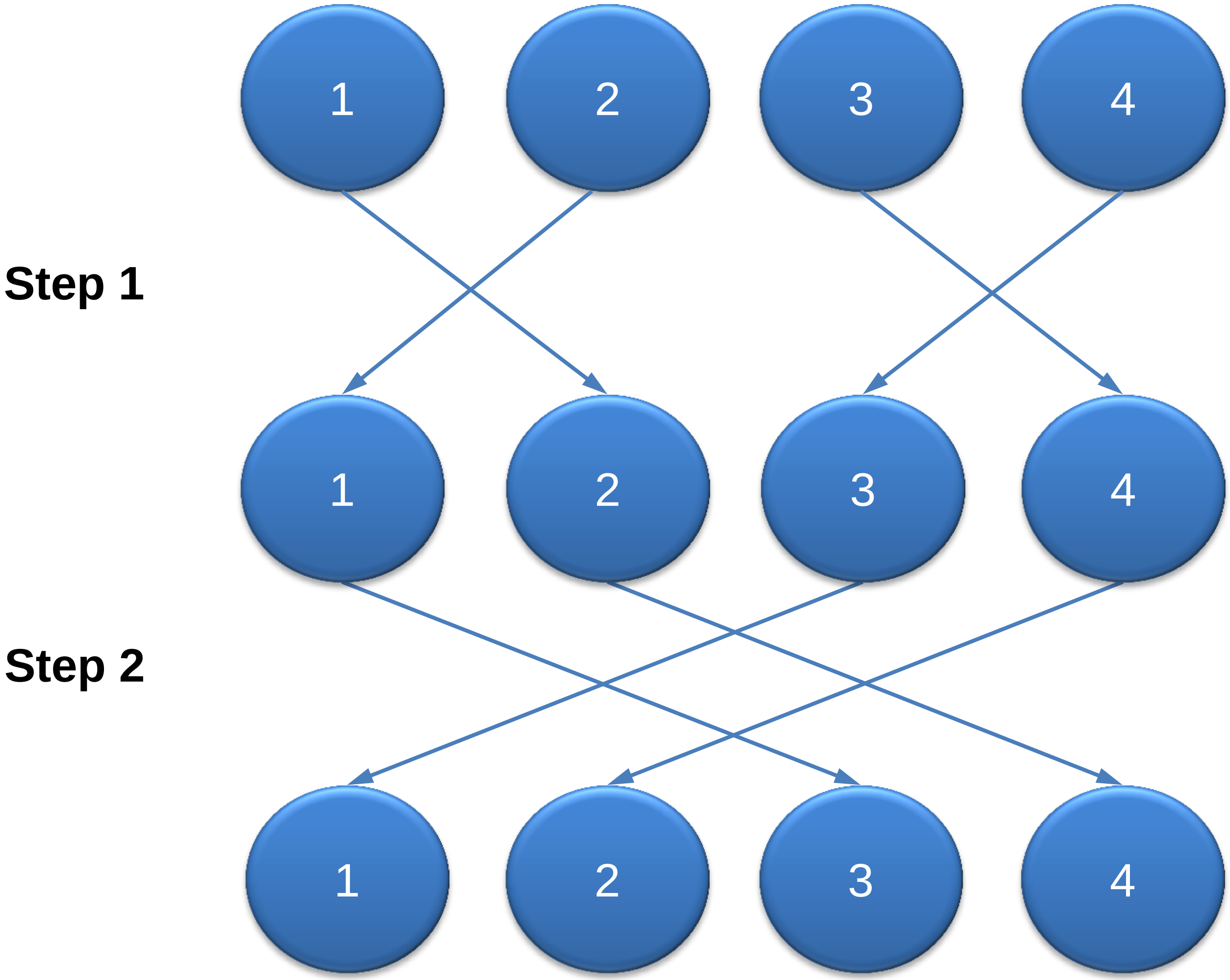


Figure 13 IBA Communication Stack

Collectives – software layers


OMPI
Module Component Architecture
Collective Framework
Basic Collectives Framework
Subgroup Framework
ML – Hierarchical Collectives Comp.
Tuned (pt2pt) Collectives Comp.
IB
IB OFFLOAD
Pt2Pt
SM
NUMA
MUMA
IBNET
MLNX OFED
MLNX OFED

Example – 4 process recursive doubling



4 Process barrier example

Algorithm

Proc 0	Proc 1	Proc 2	Proc 3
Exchange with proc 1	Exchange with proc 0	Exchange with proc 3	Exchange with proc 2
Exchange with proc 2	Exchange with proc 3	Exchange with proc 0	Exchange with proc 1

MWR

Proc 0	Proc 1	Proc 2	Proc 3
Send to proc 1	Send to proc 0	Send to proc 3	Send to proc 2
Wait on recv from 1	Wait on recv from 0	Wait on recv from 3	Wait on recv from 2
Send to proc 2	Send to proc 3	Send to proc 0	Send to proc 1
Wait on recv from 2	Wait on recv from 3	Wait on recv from 0	Wait on recv from 1

4 Process barrier example – queue view

Send QP

Proc 0	Proc 1	Proc 2	Proc 3
Send to proc 1 – enabled	Send to proc 0 – enabled	Send to proc 3 – enabled	Send to proc 2 – enabled
Send to 2 – not enabled	Send to 3 – not enabled	Send to 0 – not enabled	Send to 1 – not enabled

MQ

Proc 0	Proc 1	Proc 2	Proc 3
Recv wait from 1	Recv wait from 0	Recv wait from 3	Recv wait from 2
Send enable 1	Send enable 0	Send enable 3	Send enable 2
Recv wait from 2	Recv wait from 3	Recv wait from 0	Recv wait from 1

8 Process barrier example – queue view – no MQ, view at rank 0

QP 1	QP 2	QP 4
Send QP 1	Wait QP 1	Wait QP 1
	Send QP 2	Wait QP 2
		Send QP 4
		Wait QP 4

Cheetah Core-DIRECT component status

- Supported Collectives (blocking and nonblocking)
  - Barrier
  - Bcast
  - AlltoAll
  - Allgather
  - Fan-in/out
- Offloaded protocols
  - Small messages protocol with support for heterogeneous communication layers
  - Zero-Copy offload for large messages

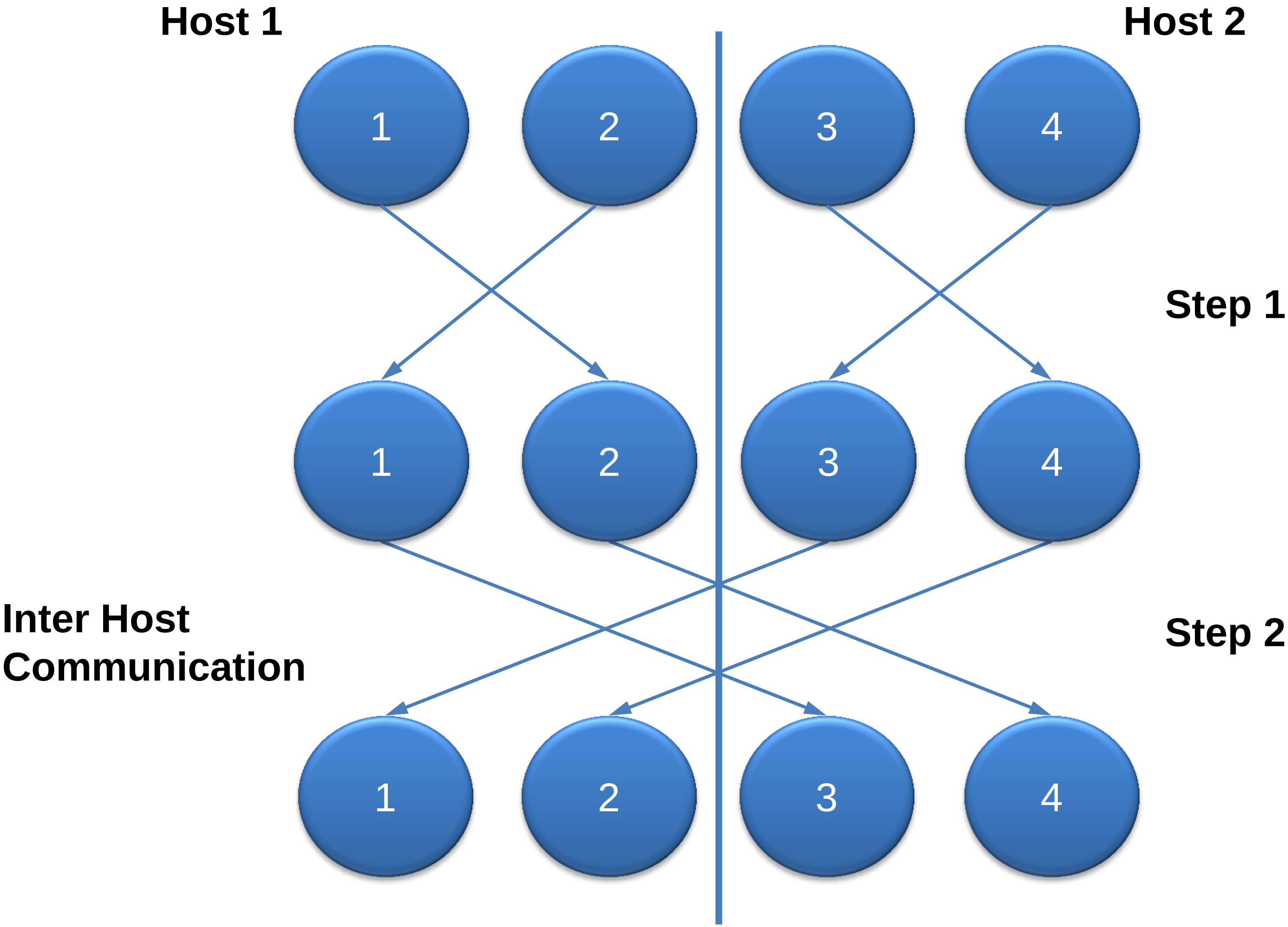
Benchmarks

System setup

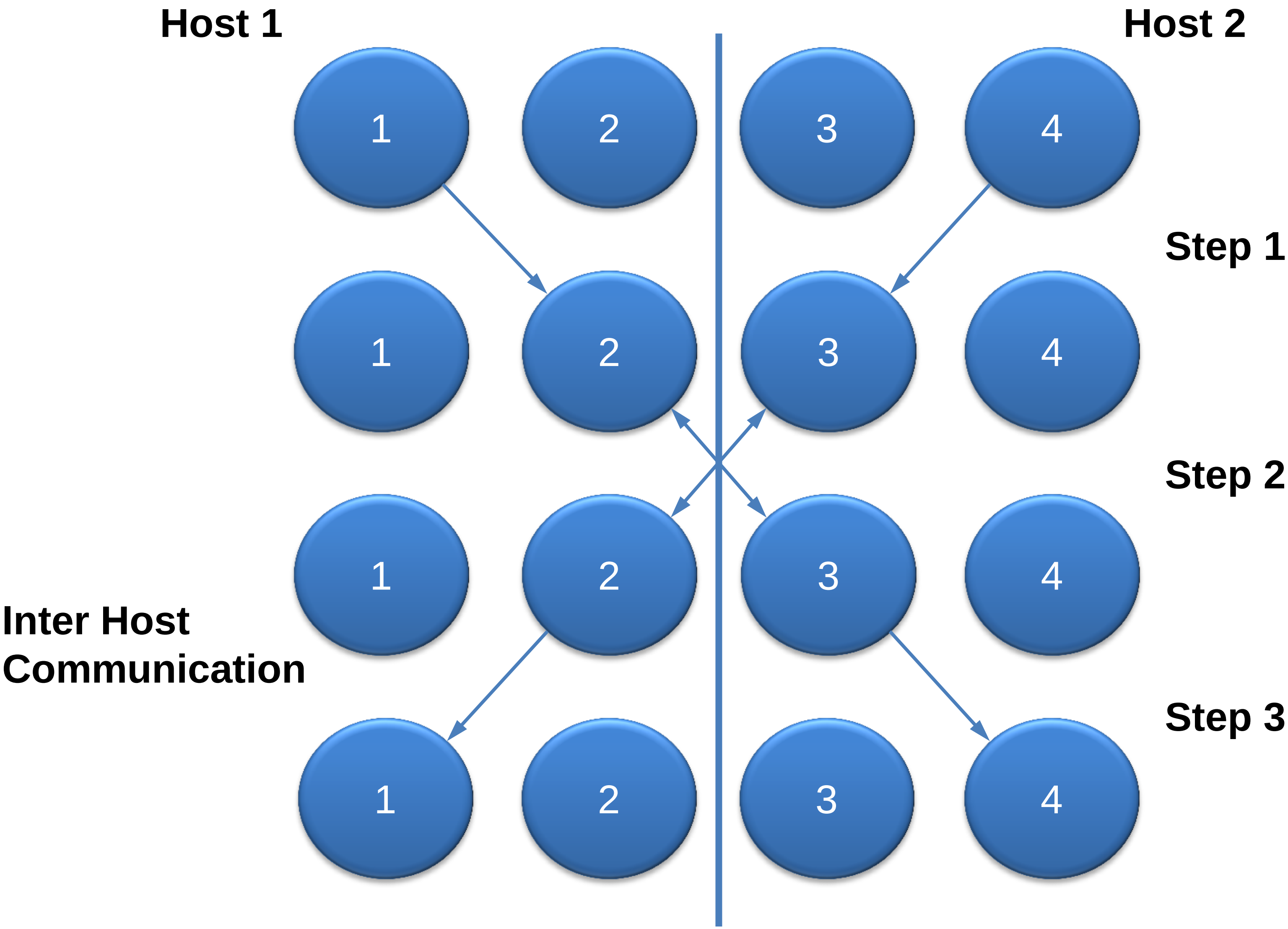
- 8 node cluster
- Node architecture
  - 3 GHz Intel Xeon
  - Dual socket
  - Quad core
- Network
  - ConnextX-2 HCA
  - 36 port QDR switch running prerelease firmware

Barrier Data

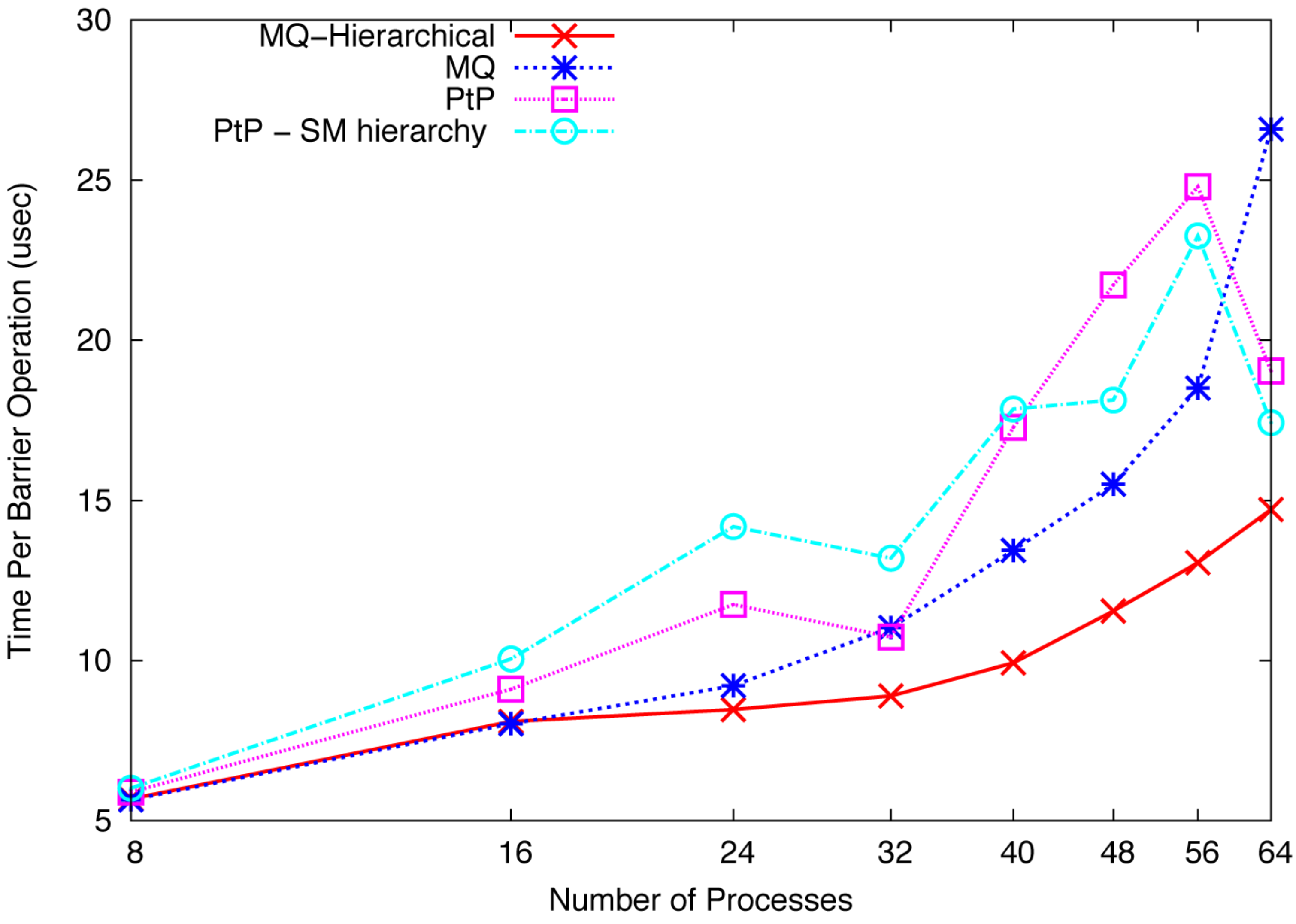
Flat barrier algorithm



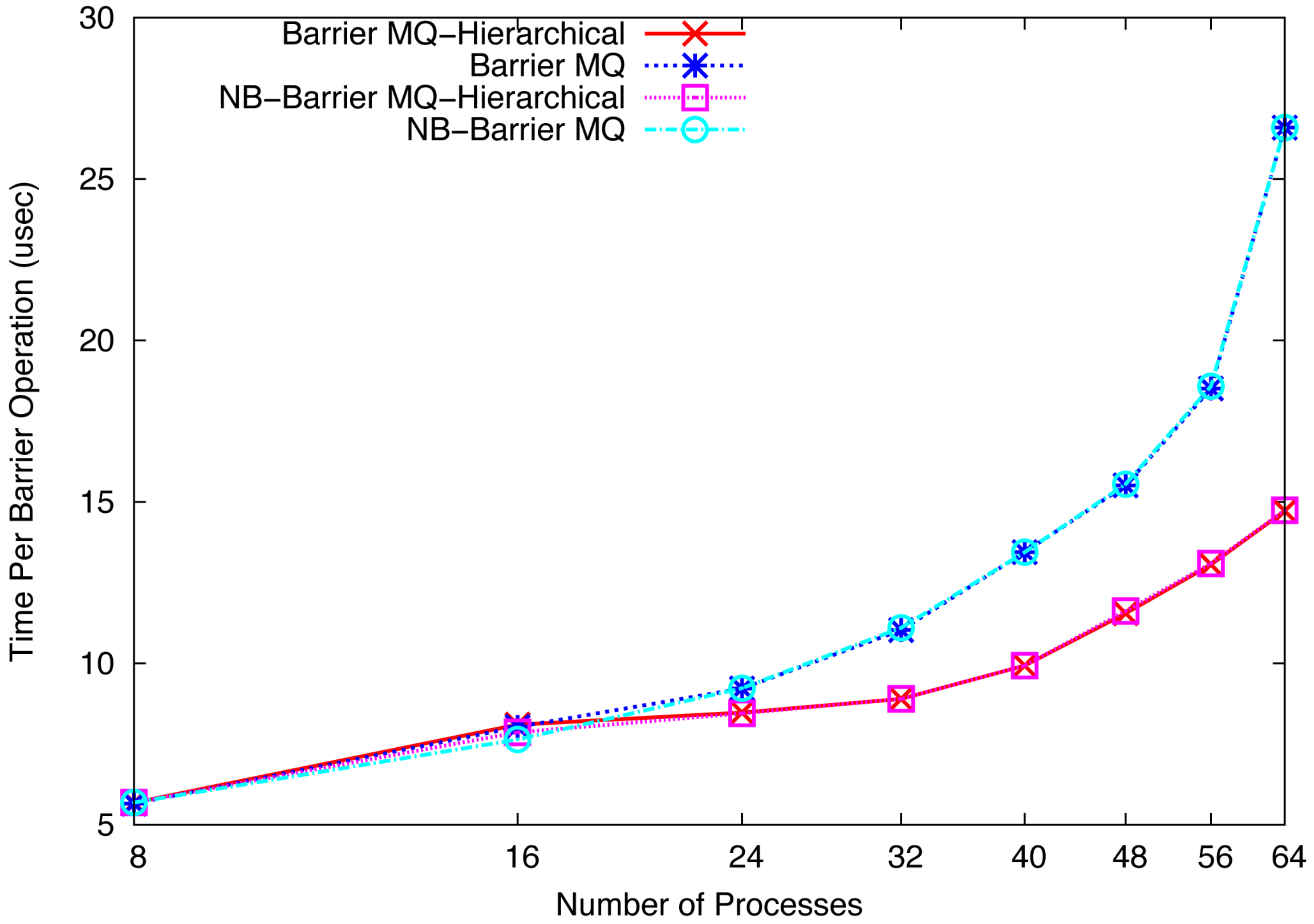
Hierarchical barrier algorithm



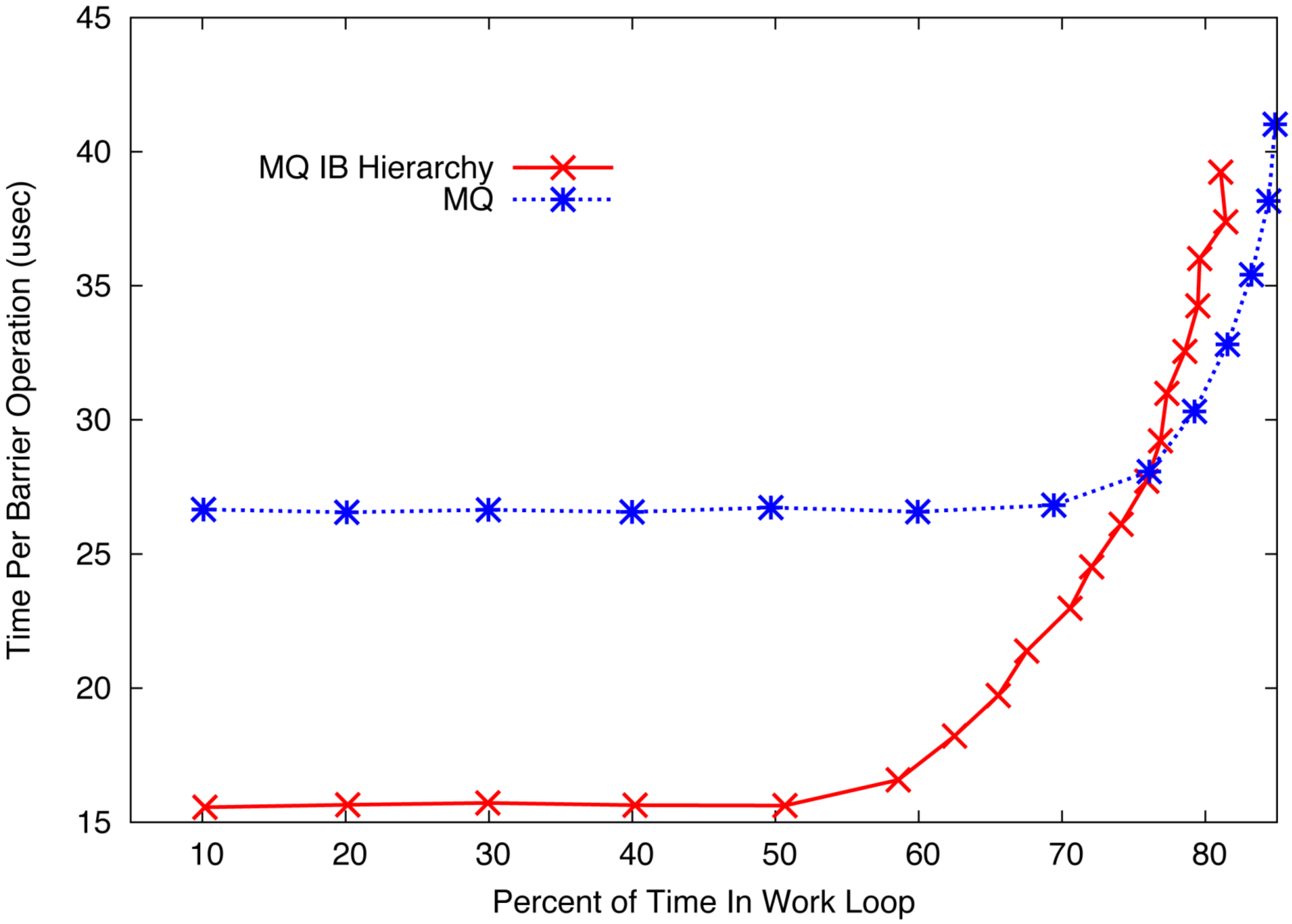
MPI barrier timings



Barrier timings – blocking vs. nonblocking



Nonblocking barrier overlap

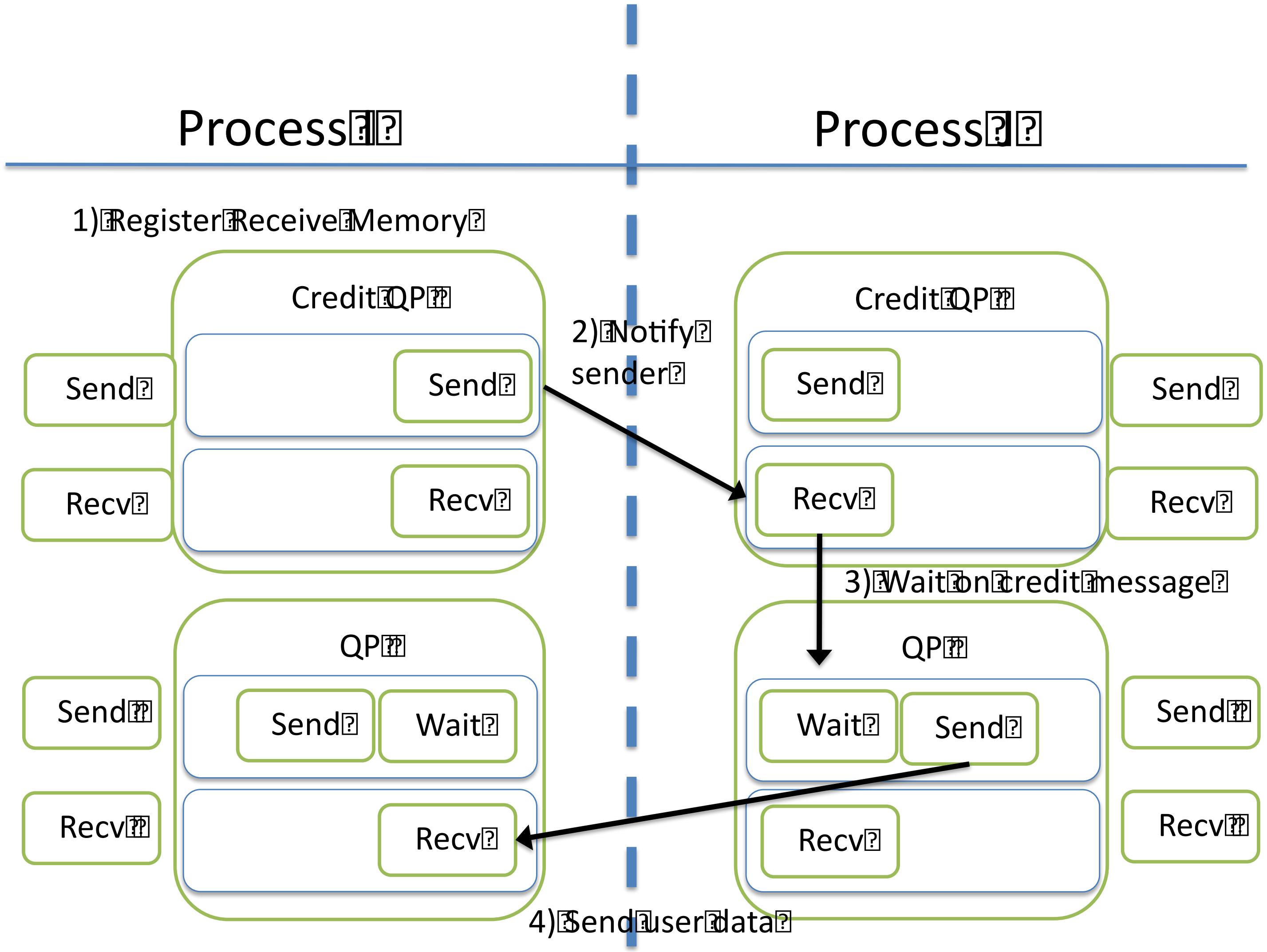


Broadcast Data

Broadcast algorithm features

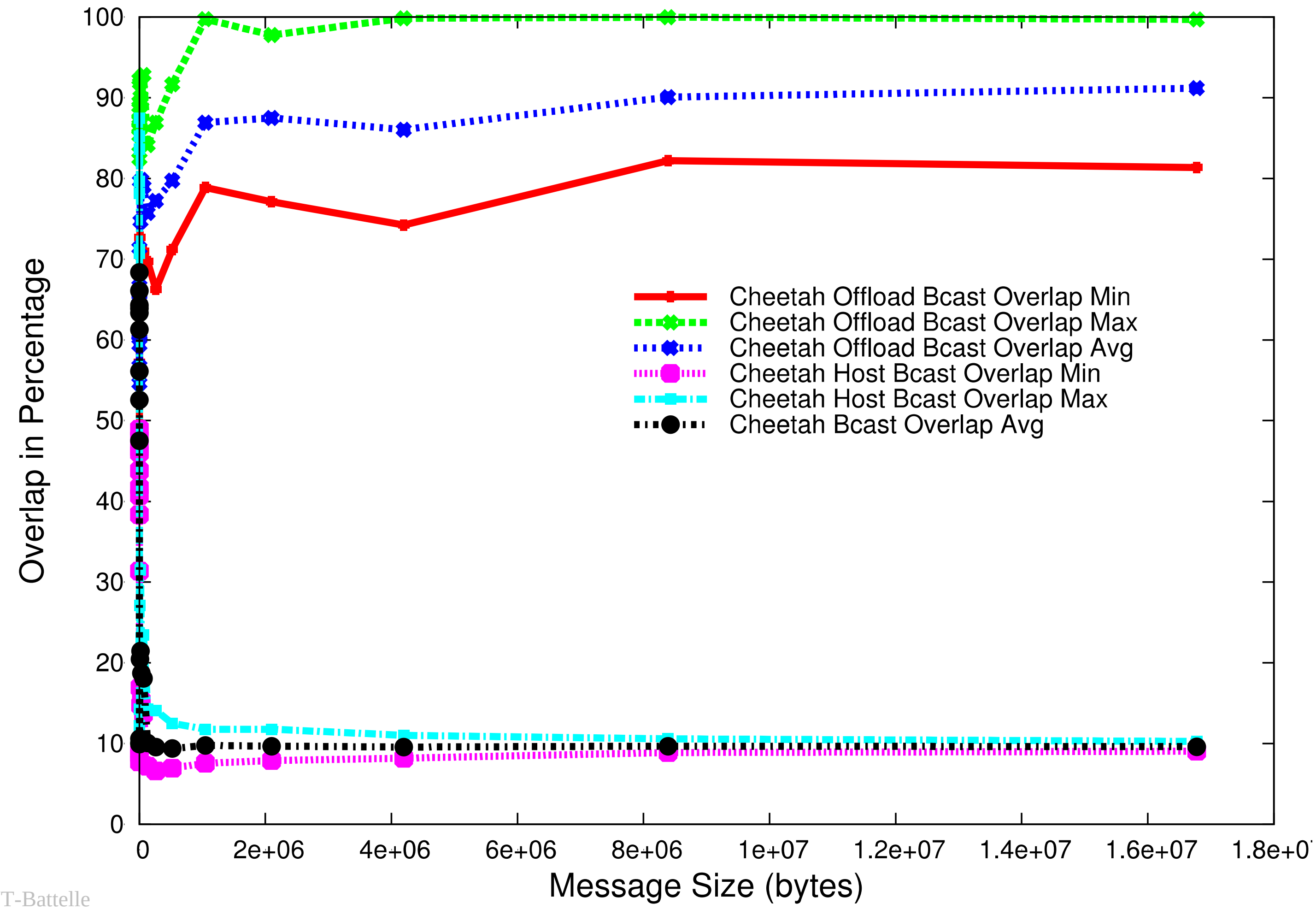
- Reduced memory footprint
  - K-nomial tree: (K-1)$\mathrm{Log}_k(N)$ connections
- Reduced memory overhead
  - Memory blocks are shared between multiple communication layers
  - Novel Zero-Copy offload for large messages
- Parallel execution on multiple communication layers
- Support for Blocking and Non-Blocking Broadcast

Zero-Copy Offload Algorithm for large message broadcast



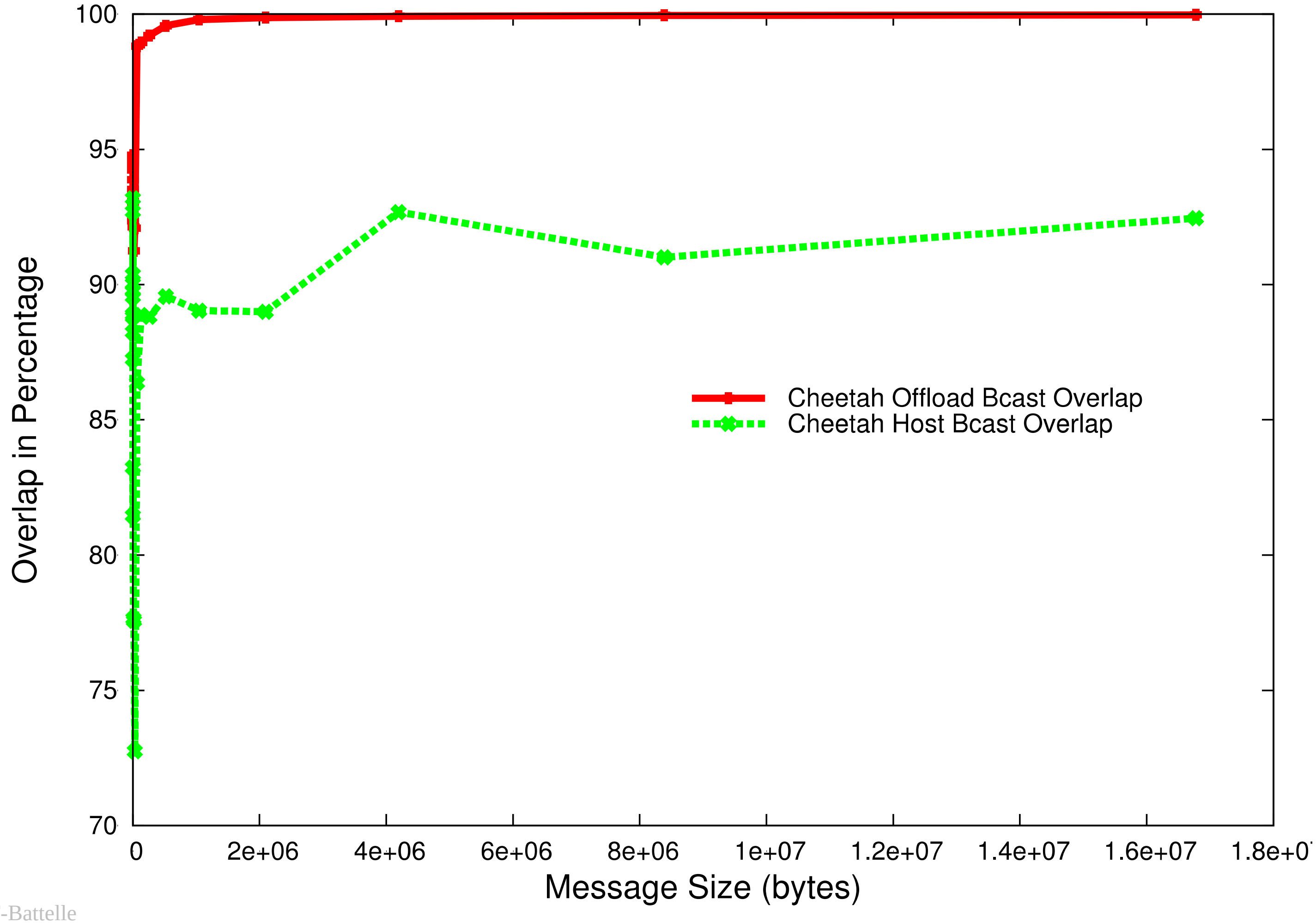
Broadcast Overlap – Wait Based

- **Percentage of the nonblocking broadcast available for work as measured with the wait-based test**



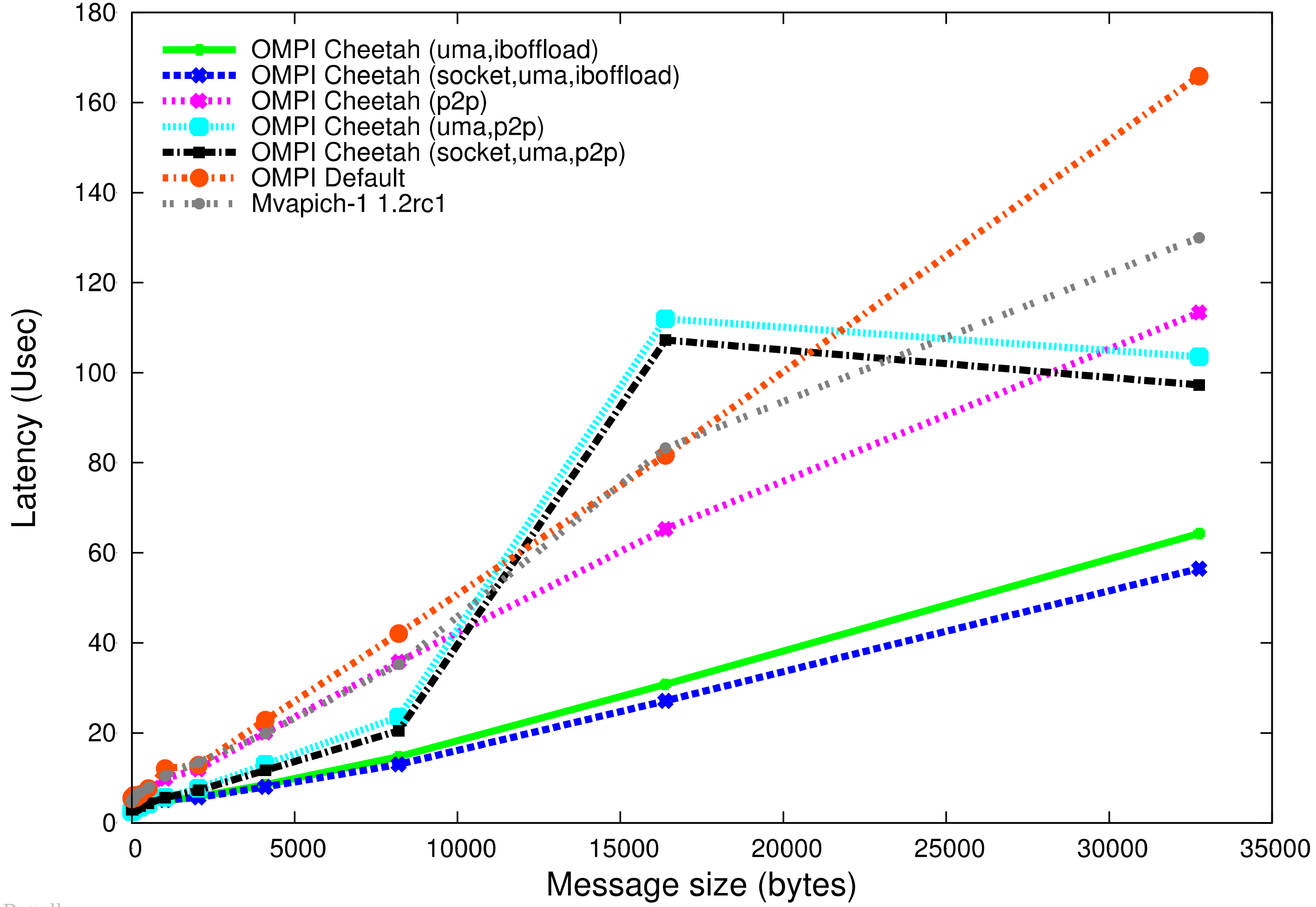
Broadcast Overlap – Polling Based

- Percentage of the nonblocking broadcast available for work as measured with the polling-based test



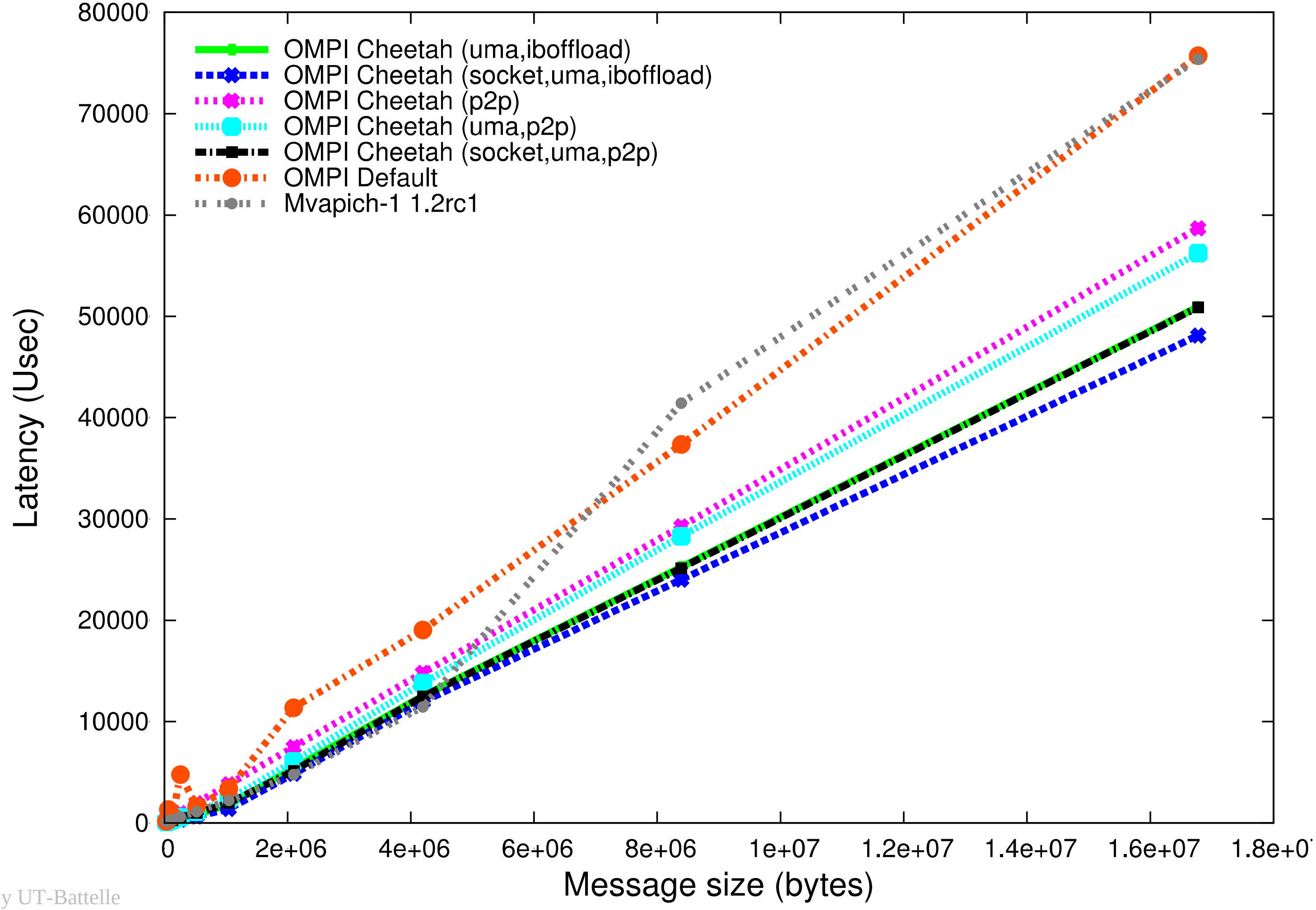
Broadcast latency (small messages)

- Small data algorithm broadcast latency as a function of message size and implementation, and 64 ranks. Message sizes very from one byte to 32 KB.



Broadcast latency (large messages)

- Large-data algorithm broadcast latency as a function of message size and implementation, and 64 ranks. Message sizes very from 32 KB to 16MB



Summary

- Added hardware support for offloading collective operations
- Developed MPI-level support for asynchronous collectives
- Good barrier and broadcast performance
- Good overlap capabilities

Publications

- Pavel Shamis, Richard L. Graham, Manjunath Gorentla Venkata, Joshua S. Ladd. "Design and Implementation of Broadcast Algorithms for Extreme-Scale Systems," IEEE Cluster 2011, accepted for publication.

- Joshua Ladd, Manjunath Gorentla Venkata, Richard Graham, Pavel Shamis. "Analyzing the Effects of Multicore Architectures and On-host Communication Characteristics on Collective Communications," The Seventh International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS 2011), accepted for publication.

- Manjunath Gorentla Venkata, Richard Graham, Joshua Ladd, Pavel Shamis, Ishai Rabinovitz, Vasily Filipov and Gilad Shainer. "ConnectX-2 CORE-Direct Enabled Asynchronous Broadcast Collective Communications," The 1st Workshop on Communication Architecture for Scalable Systems (CASS2011), May 2011.

- Richard Graham, Manjunath Gorentla Venkata, Joshua Ladd, Pavel Shamis, Ishai Rabinovitz, Vasily Filipov and Gilad Shainer. "Cheetah: A Framework for Scalable Hierarchical Collective Operations," The 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2011), May 2011.

- Ishai Rabinovitz, Pavel Shamis, Richard L. Graham, Noam Bloch, Gilad Shainer. "Network Offloaded Hierarchical Collectives Using ConnectX-2's CORE-Direct capabilities," EuroMPI 2010 - Stuttgart, Germany, September 2010

- Richard L. Graham, Steve Poole, Pavel Shamis, Gil Bloch, Noam Bloch, Hillel Chap- man, Michael Kagan, Ariel Shahar, Ishai Rabinovitz, Gilad Shainer. "ConnectX-2 InfiniBand Management Queues: First investigation of the new support for network offloaded collective operations," The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2010), May 2010

- Richard L. Graham, Steve Poole, Pavel Shamis, Gil Bloch, Noam Bloch, Hillel Chap- man, Michael Kagan, Ariel Shahar, Ishai Rabinovitz, Gilad Shainer. "Overlapping computation and communication: Barrier algorithms and ConnectX-2 CORE-Direct capabilities," The 10th Workshop on Communication Architecture for Clusters (CAC 2010), April 2010

Contact

Richard L. Graham

Application Performance Tools
Computer Science and Mathematics Division
(865) 356-3469
rlgraham@ornl.gov