Advanced Topics in InfiniBand and High-speed Ethernet for Designing HEC Systems

A Tutorial at ISC '13

by

Dhabaleswar K. (DK) Panda

The Ohio State University

E-mail: panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda

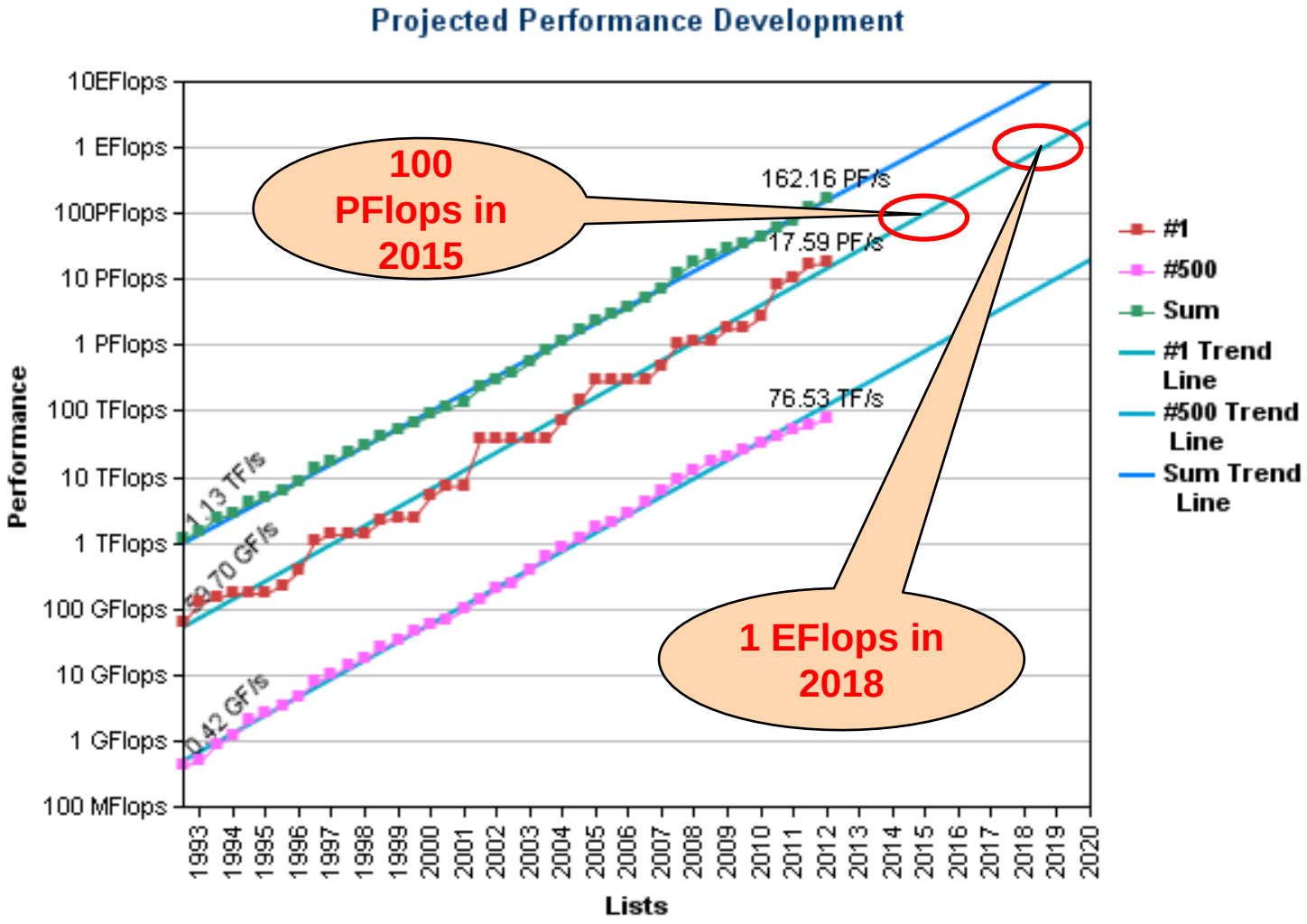
Hari Subramoni

The Ohio State University

E-mail: subramon@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~subramon

High-End Computing (HEC): PetaFlop to ExaFlop

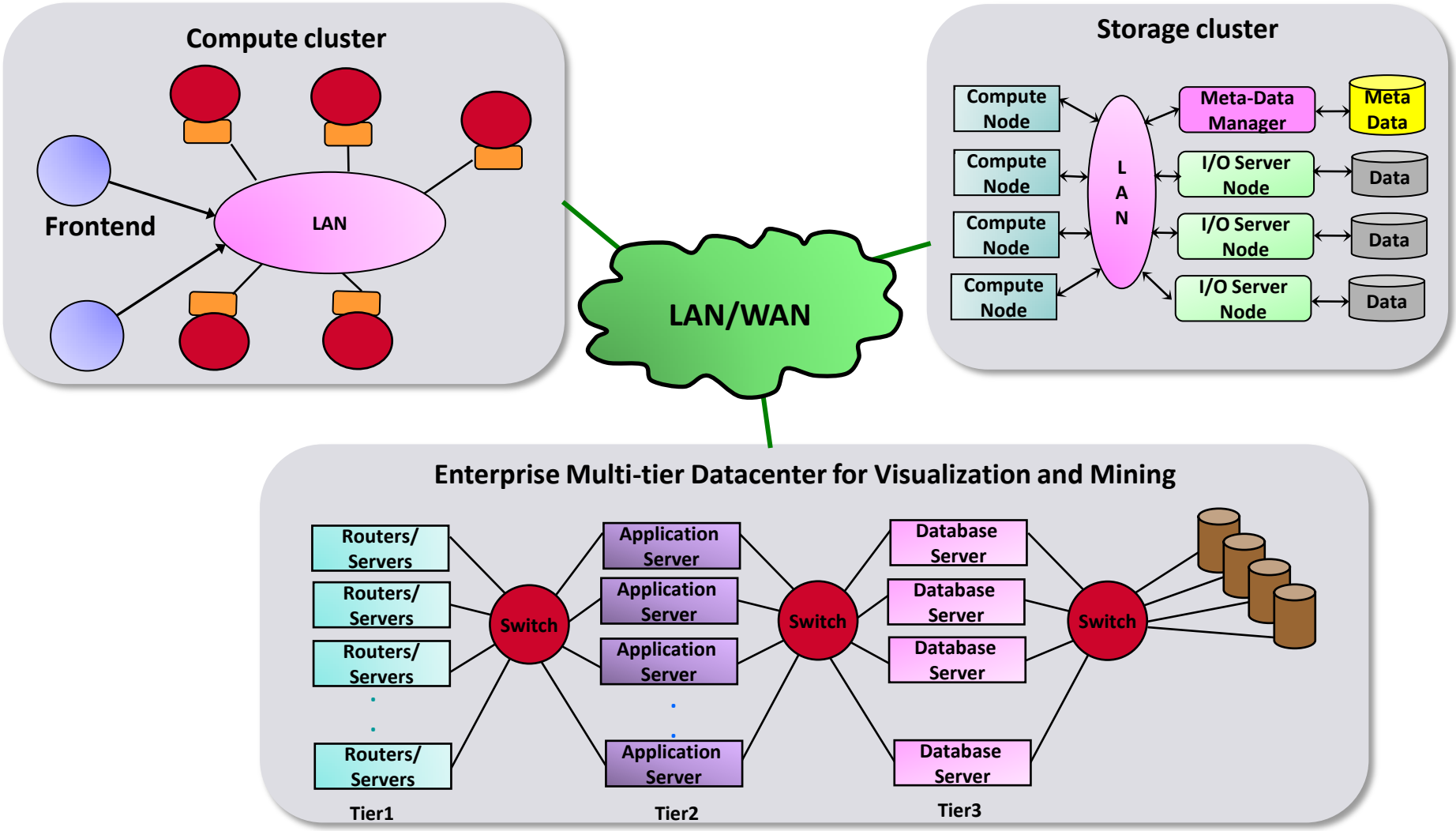


Expected to have an ExaFlop system in 2019 -2022!

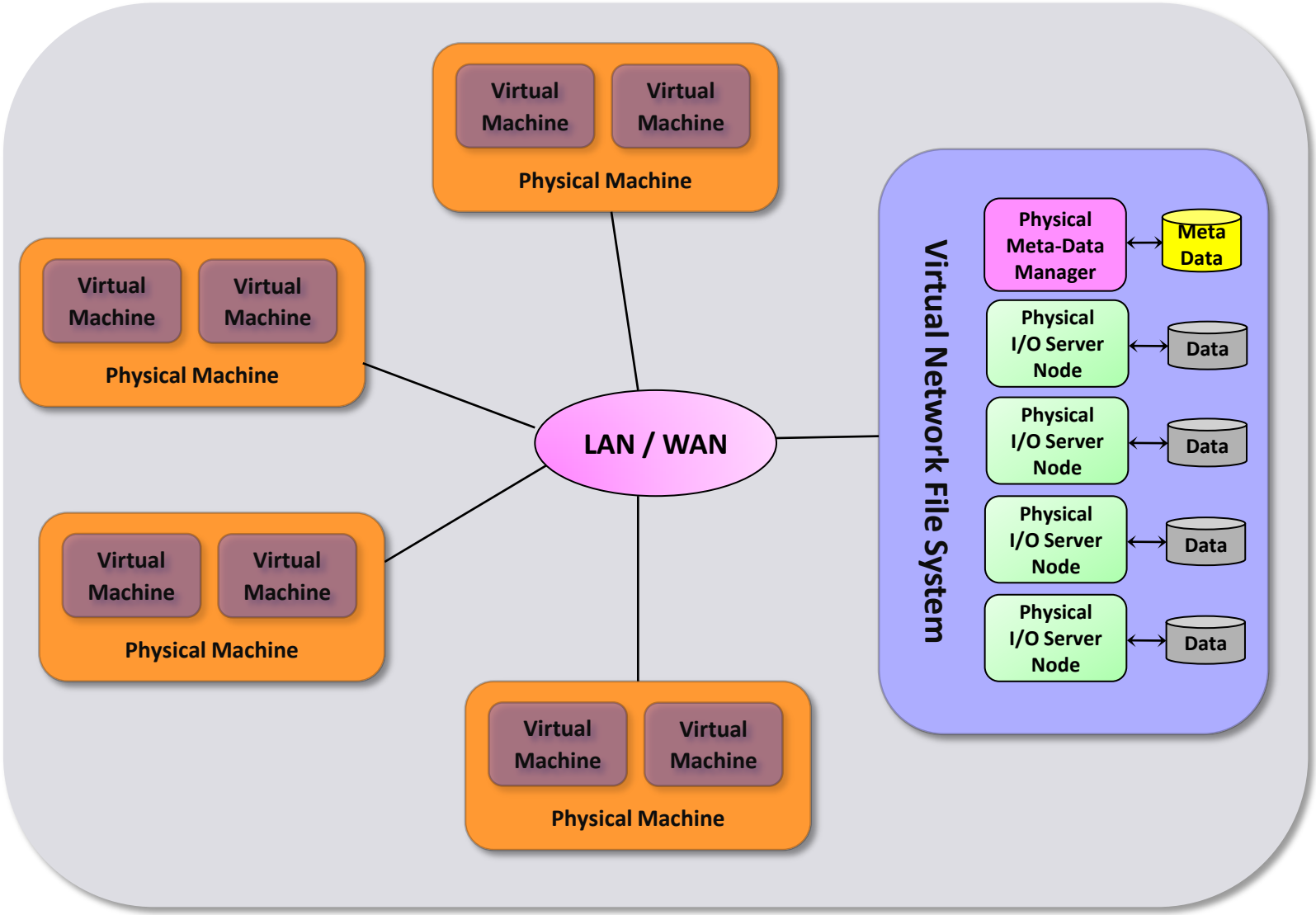
Various High-End Computing (HEC) Systems

- Compute Clusters
- Storage Clusters
- Multi-tier Data Centers
- Cloud Computing Environments
- Grid Computing Environments
- Big Data Processing (Hadoop with HDFS and HBase)
- Web 2.0 with Memcached

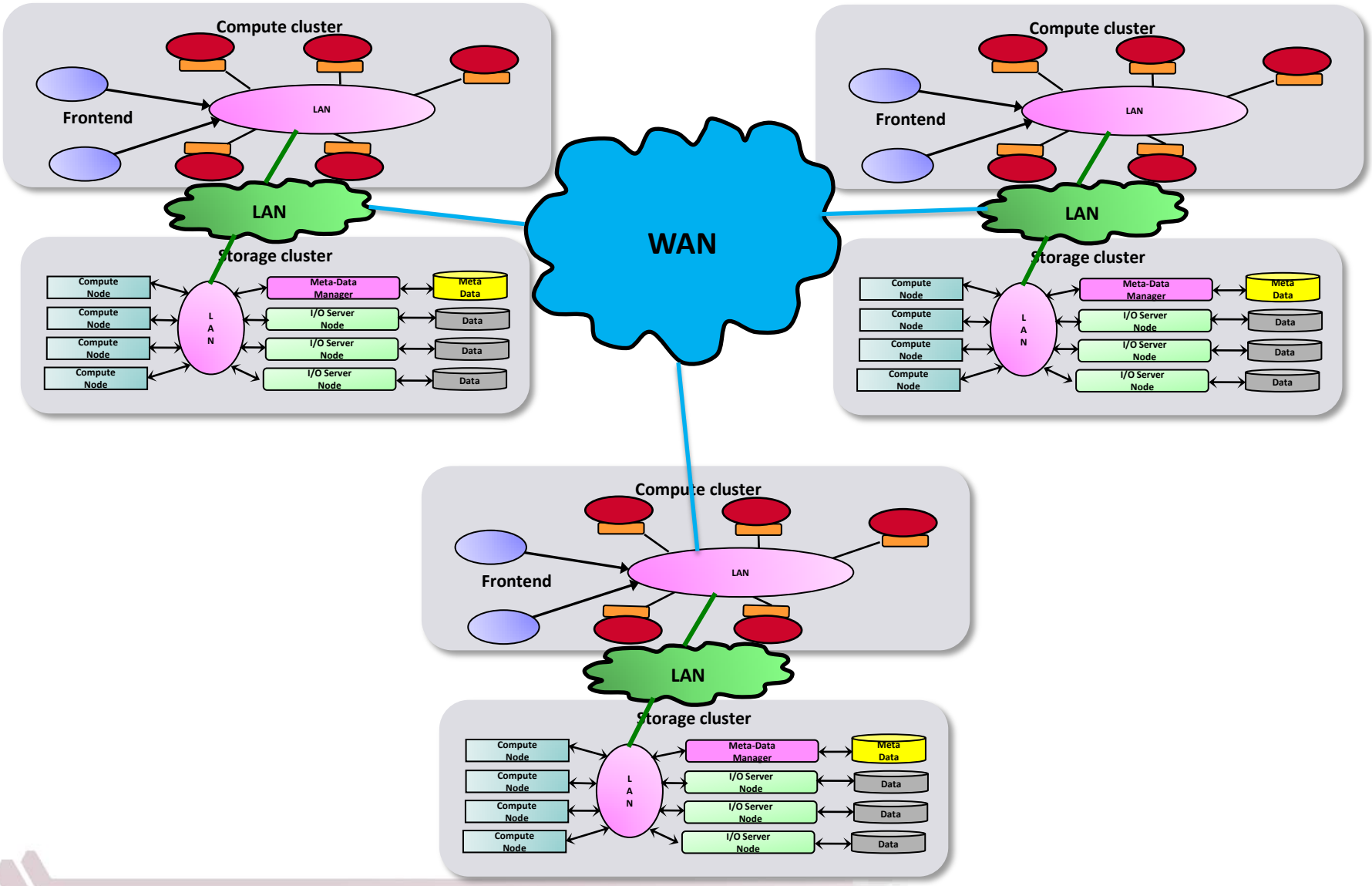
Various Clusters (Compute, Storage and Datacenters)



Cloud Computing Environments

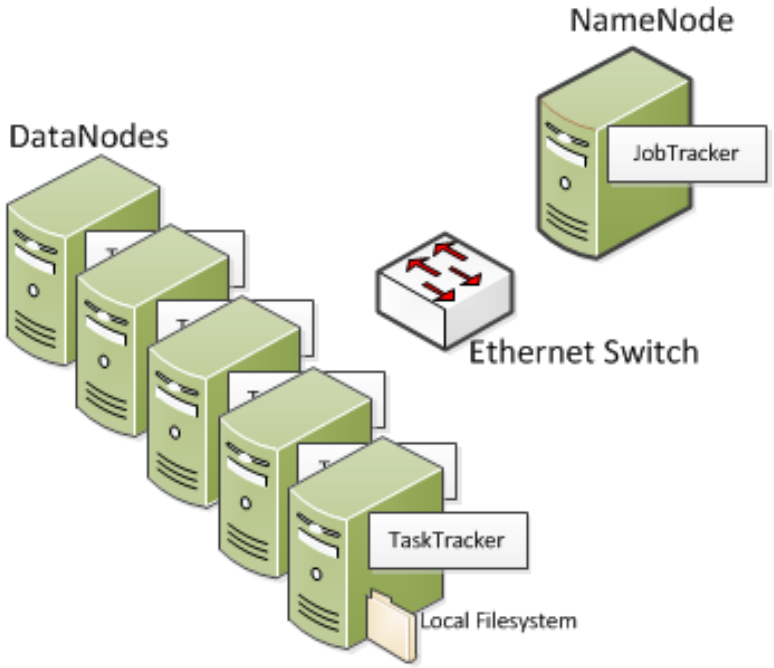


Grid Computing Environment



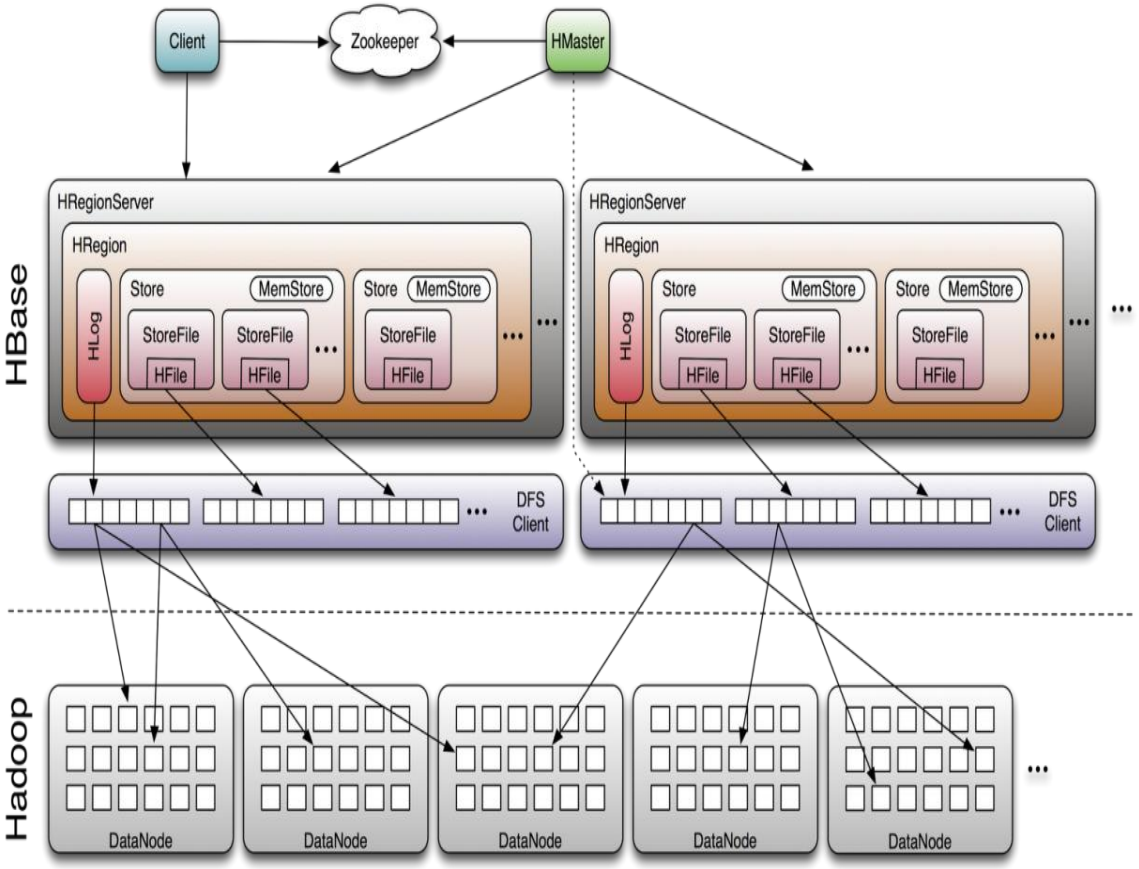
Big Data Processing with Hadoop: HDFS

- Underlying Hadoop Distributed File System (HDFS)
- Fault-tolerance by replicating data blocks
- NameNode: stores information on data blocks
- DataNodes: store blocks and host Map-reduce computation
- JobTracker: track jobs and detect failure
- Model scales but high amount of communication during intermediate phases

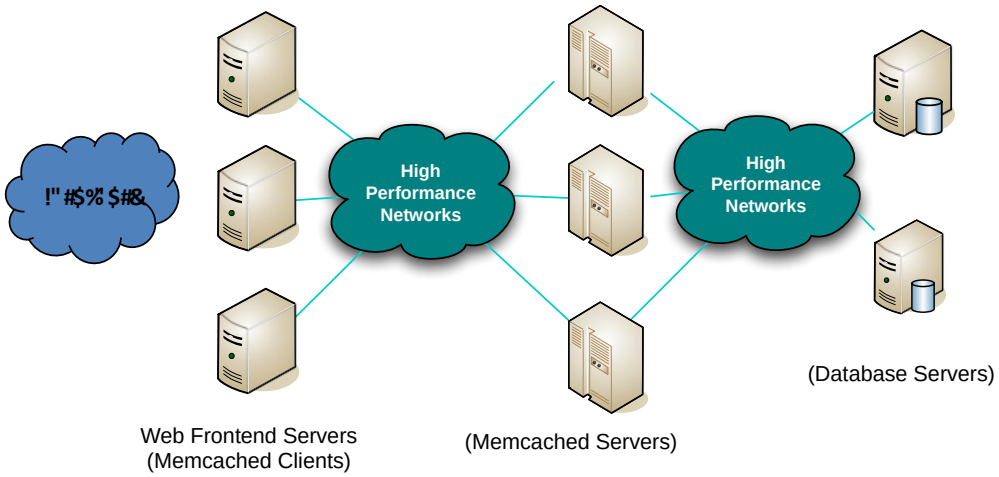


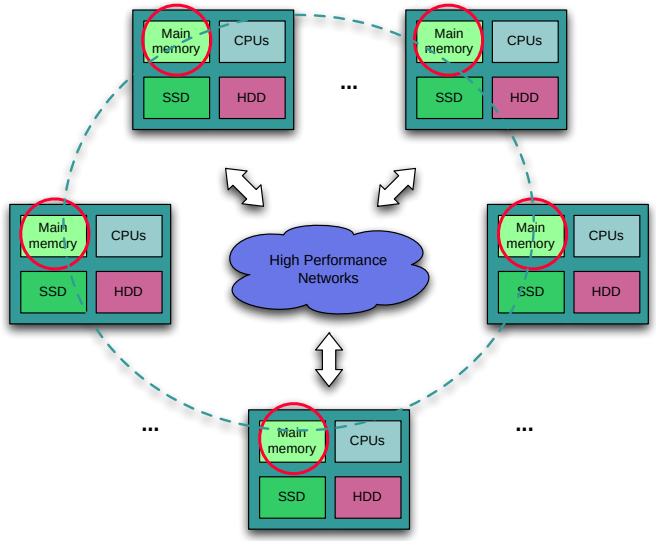
Big Data Processing with Hadoop: HBase

- An open source database project based on Hadoop framework for hosting very large tables
- Major components: HBaseMaster, HRegionServer and HBaseClient
- HBase and HDFS are deployed in the same cluster to get better data locality



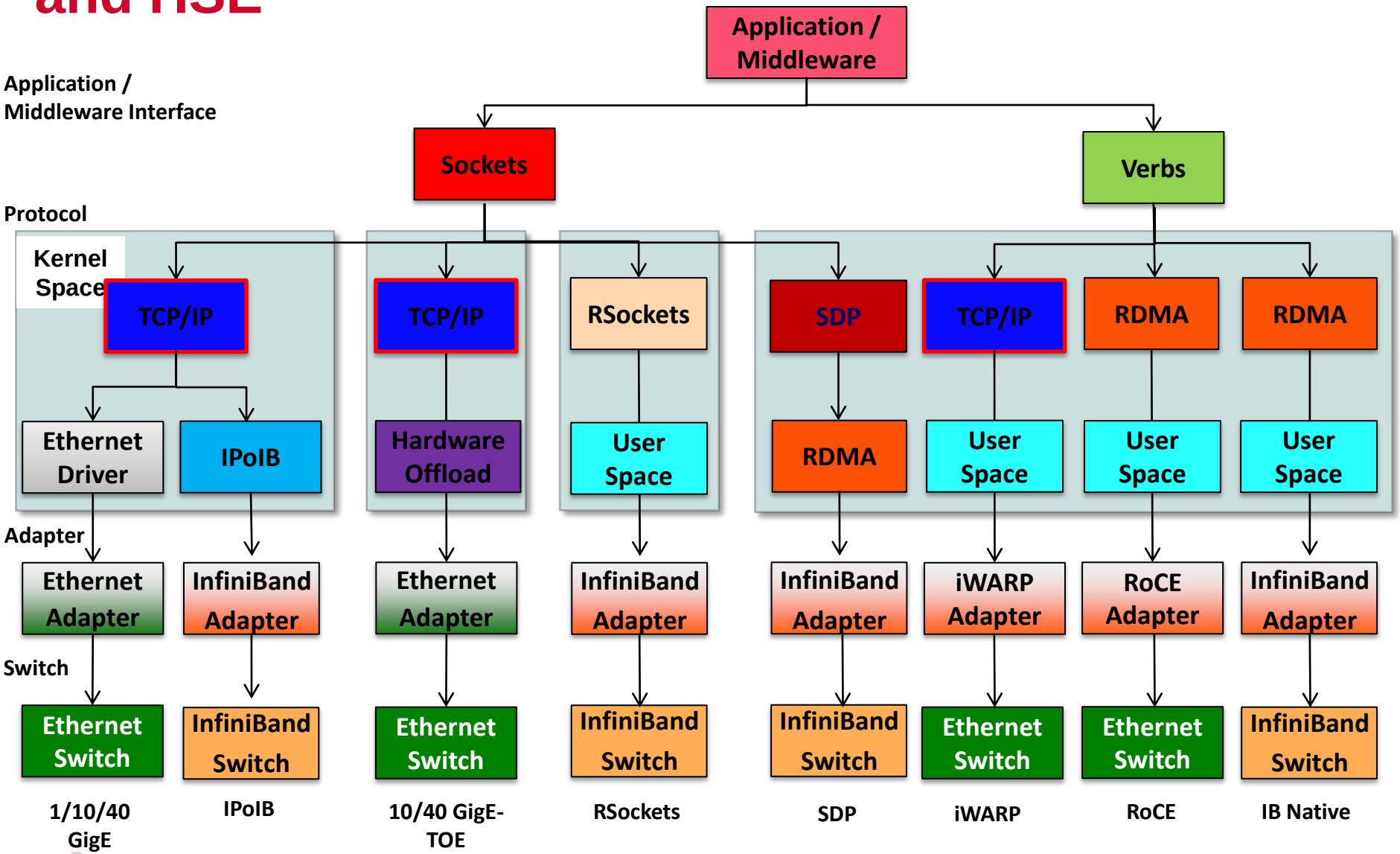
Web 2.0 with Memcached





- Distributed Caching Layer
  - Allows to aggregate spare memory from multiple nodes
  - General purpose
- Typically used to cache database queries, results of API calls
- Pillar of Web 2.0
- Scalable model, but typical usage very network intensive

Modern Interconnects and Protocols with IB and HSE



Large-scale InfiniBand Installations

- 224 IB Clusters (44.8%) in the November 2012 Top500 list (http://www.top500.org)
- Installations in the Top 40 (16 systems):

147, 456 cores (Super MUC) in Germany (6th)	122,400 cores (Roadrunner) at LANL (22nd)
204,900 cores (Stampede) at TACC (7th)	53,504 (PRIMERGY) at Australia/NCI (24th)
77,184 cores (Curie thin nodes) at France/CEA (11th)	78,660 cores (Lomonosov) in Russia (26th)
120, 640 cores (Nebulae) at China/NSCS (12th)	137,200 cores (Sunway Blue Light) in China (28th)
72,288 cores (Yellowstone) at NCAR (13th)	46,208 cores (Zin) at LLNL (29th)
125,980 cores (Pleiades) at NASA/Ames (14th)	33,664 (MareNostrum) at Spain/BSC (36th)
70,560 cores (Helios) at Japan/IFERC (15th)	32,256 (SGI Altix X) at Japan/CRIEPI (39th)
73,278 cores (Tsubame 2.0) at Japan/GSIC (17th)	***More are getting installed !***
138,368 cores (Tera-100) at France/CEA (20th)	

Presentation Overview

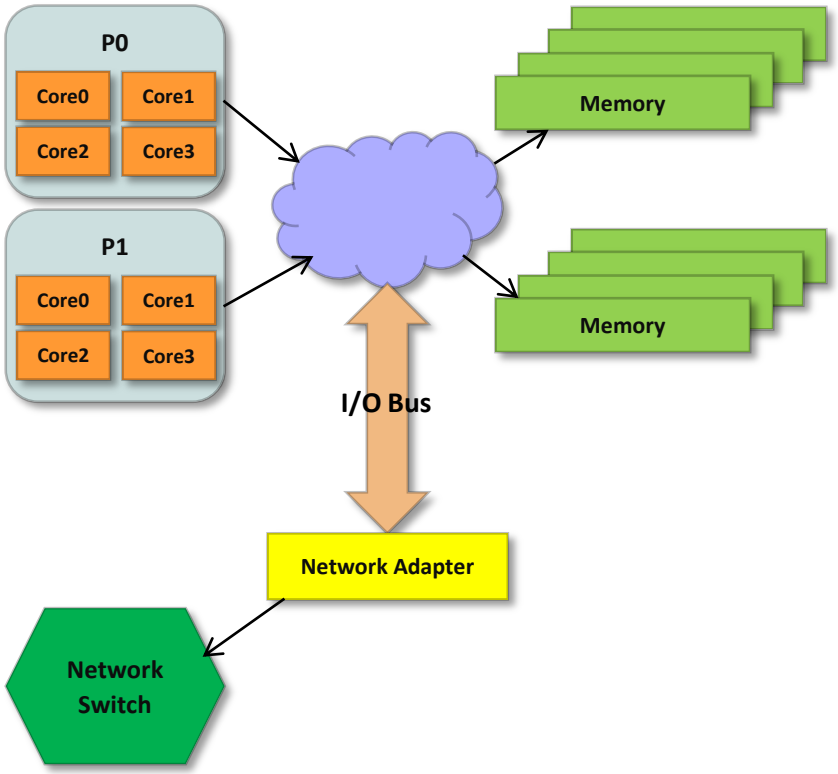
- **Common Challenges in Building HEC Systems with IB and HSE**
  - **Network Adapters and NUMA Interactions**
  - **Scalability and Memory Overheads**
  - **Network Switches, Topology and Routing**
  - **Network Bridges**
- System Specific Challenges and Case Studies
  - HPC (MPI, PGAS and GPU/MIC Computing)
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - Storage and File Systems
  - Grid Computing
- Open Fabrics Software Stack and RDMA Programming
- Network Management Infrastructure and Tools
- Conclusions and Final Q&A

Common Challenges for Large-Scale Installations

Common Challenges in Building HEC Systems with IB and HSE

- **Network adapters and interactions with other components**
  - **I/O bus interactions and limitations**
  - **Multi-port adapters and bottlenecks**
  - NUMA interactions
  - Memory overheads in large-scale systems
  - Collective offload support
- Network switches
- Network bridges

I/O bus limitations



- I/O link bandwidth:
  - Tricky because several aspects need to be considered
  - Connector capacity vs. link capacity
  - I/O link communication headers, etc.

- Data communication traverses three buses (or links) before it reaches the network switch
  - Memory bus (memory to IO hub)
  - I/O link (IO hub to the network adapter)
  - Network link (network adapter to switch)
- For optimal communication, all these need to be balanced
- Network bandwidth:
  - 4X SDR (8 Gbps), 4X DDR (16 Gbps), 4X QDR (32 Gbps), 4X FDR (56 Gbps)
  - 40 GigE (40 Gbps)
- Memory bandwidth:
  - Shared bandwidth (incoming and outgoing)
  - For IB FDR (56 Gbps), memory bandwidth greater than 112 Gbps is required to fully utilize the network

PCI Express

- Common I/O interconnect used on most current platforms
  - Can be configured as multiple lanes (1X, 4X, 8X, 16X, 32X)
    - Generation 1 provided 2 Gbps bandwidth per lane, Gen 2 provides 4 Gbps, and Gen 3 provides 8 Gbps per lane)
  - Compatible with adapters using lesser lanes

Beware

    - If a PCIe connector is 16X, it will still support an 8X adapter by using only 8 lanes

Beware

  - Provides multiplexing across a single lane
    - A 1X PCIe bus can be connected to an 8X PCIe connector (allowing an 8X adapter to be plugged in)
  - I/O interconnects are like networks with packetized communication
    - Communication headers for each packet
    - Reliability acknowledgments
    - Flow control acknowledgments

Use I/O bandwidth

Beware

- Typical efficiency is around 75-80% with 256 byte PCIe packets

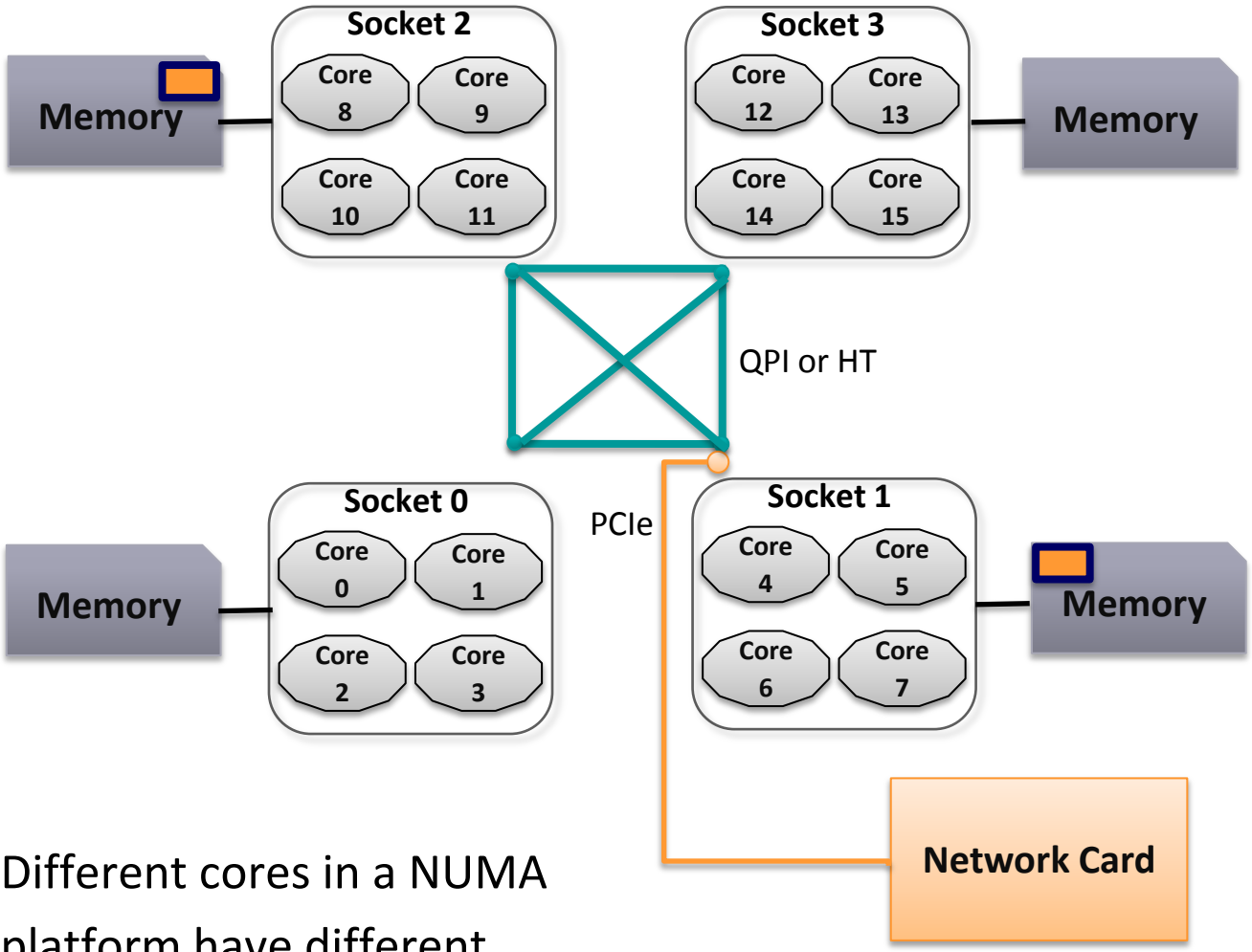
Multi-port adapters

- Several multi-port adapters available in the market
  - Single adapter can drive multiple network ports at full bandwidth
  - Important to measure other overheads (memory bandwidth and I/O link bandwidth) before assuming performance benefit
- Case Study: IB Dual-port 4x QDR adapter
  - Each network link is 32 Gbps (dual-port adapters can drive 64 Gbps)
  - PCIe Gen2 8X link can give 32 Gbps data rate → around 24 Gbps effective rate (20 % encoding overheads!!)
    - Dual-port IB QDR is not expected to give any benefit in this case
  - PCIe Gen3 8X link can give 64 Gbps data rate → 64 Gbps (minimal encoding overheads)
    - Delivers close to peak performance with Dual-port IB adapters

Common Challenges in Building HEC Systems with IB and HSE

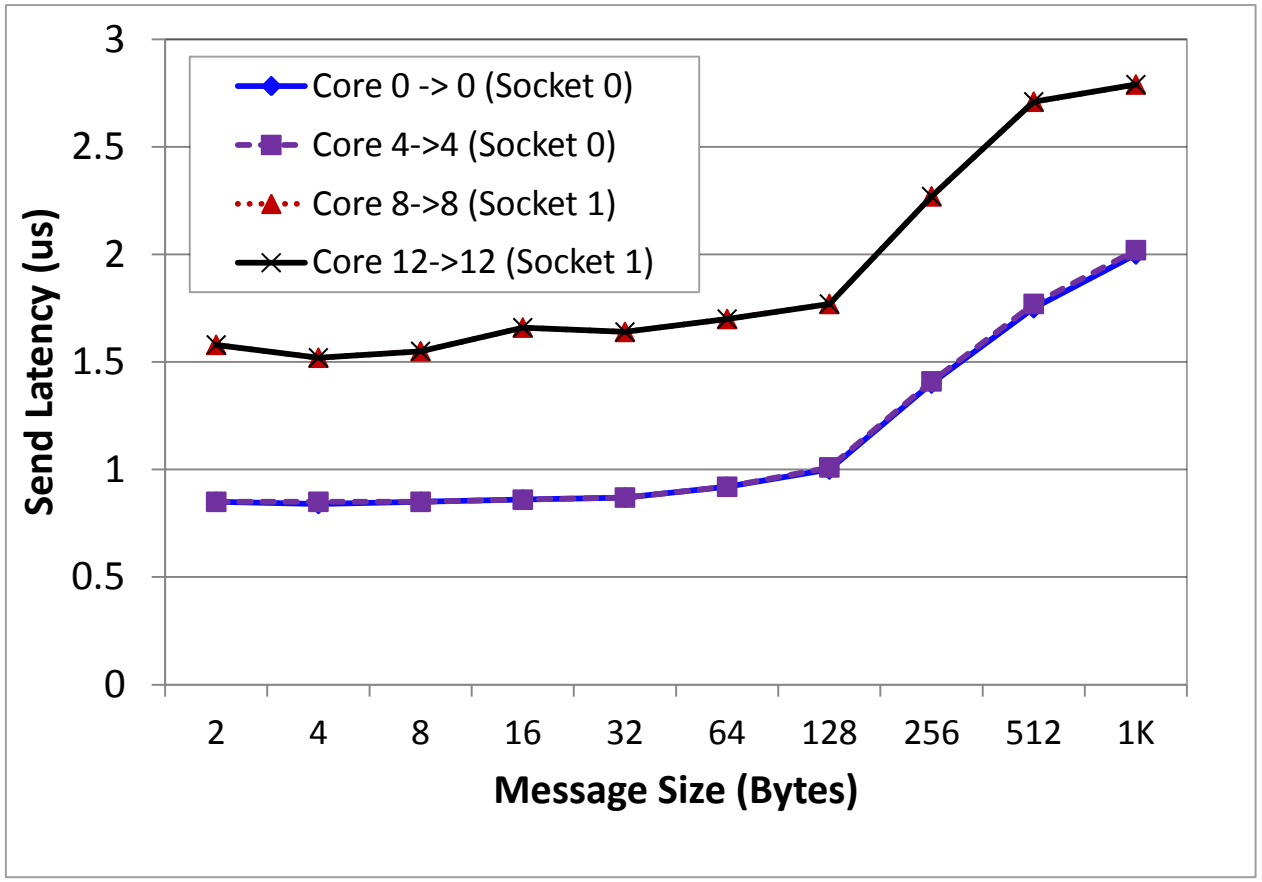
- **Network adapters and interactions with other components**
  - I/O bus interactions and limitations
  - Multi-port adapters and bottlenecks
  - **NUMA interactions**
  - Memory overheads in large-scale systems
  - Collective offload support
- Network switches
- Network bridges

NUMA Interactions



- Different cores in a NUMA platform have different communication costs

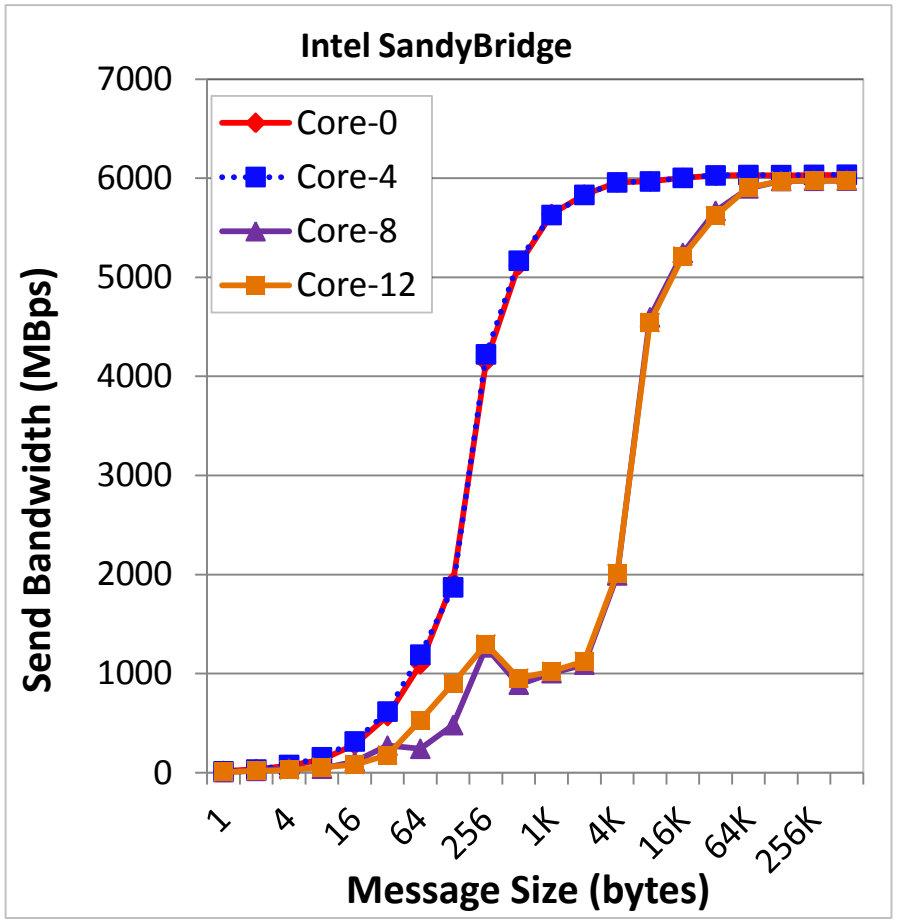
Impact of NUMA on IB Verbs Latency

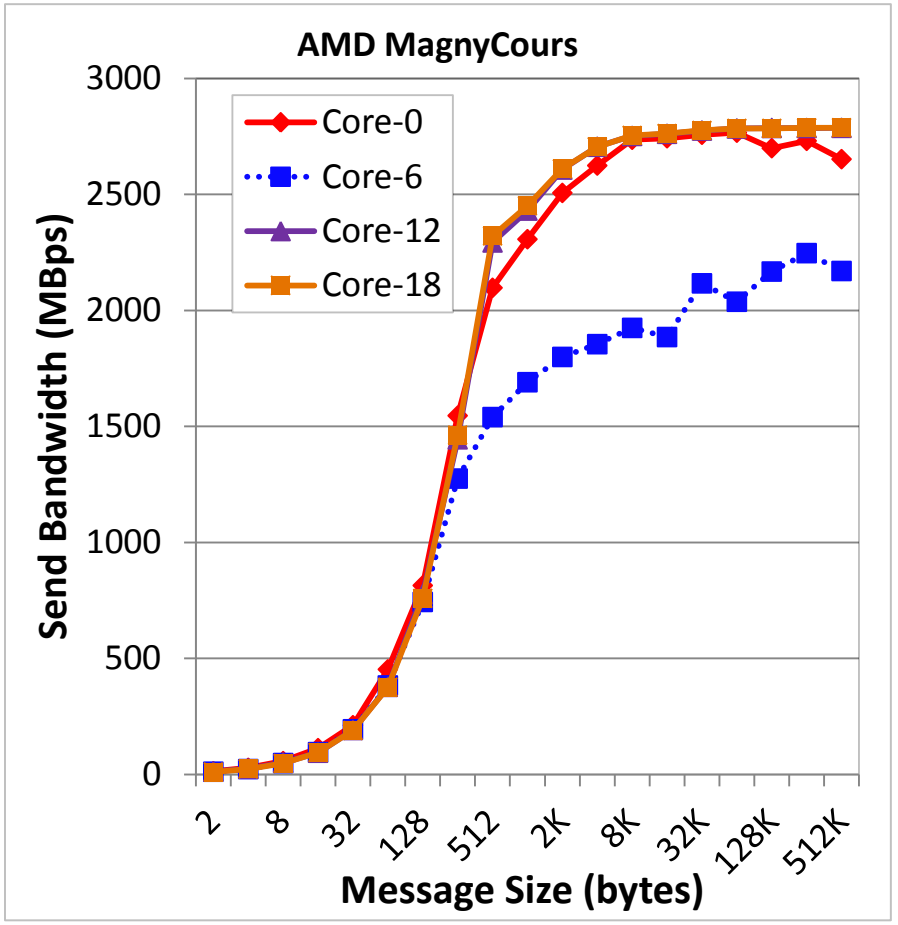


- Cores in Socket 0 (closest to network card) have lowest latency
- Cores in Socket 1 (one hop from network card) have highest latency

ConnectX-3 FDR (54 Gbps): 2.6 GHz Octa-core (SandyBridge) Intel with IB (FDR) switches

Impact of NUMA on IB Verbs Bandwidth





- NUMA interactions have significant impact on bandwidth

ConnectX-3 FDR (54 Gbps): 2.6 GHz Octa-core (SandyBridge) Intel with IB (FDR) switches
ConnectX-2-QDR (36 Gbps): 2.5 GHz Hex-core (MagnyCours) AMD with IB (QDR) switches

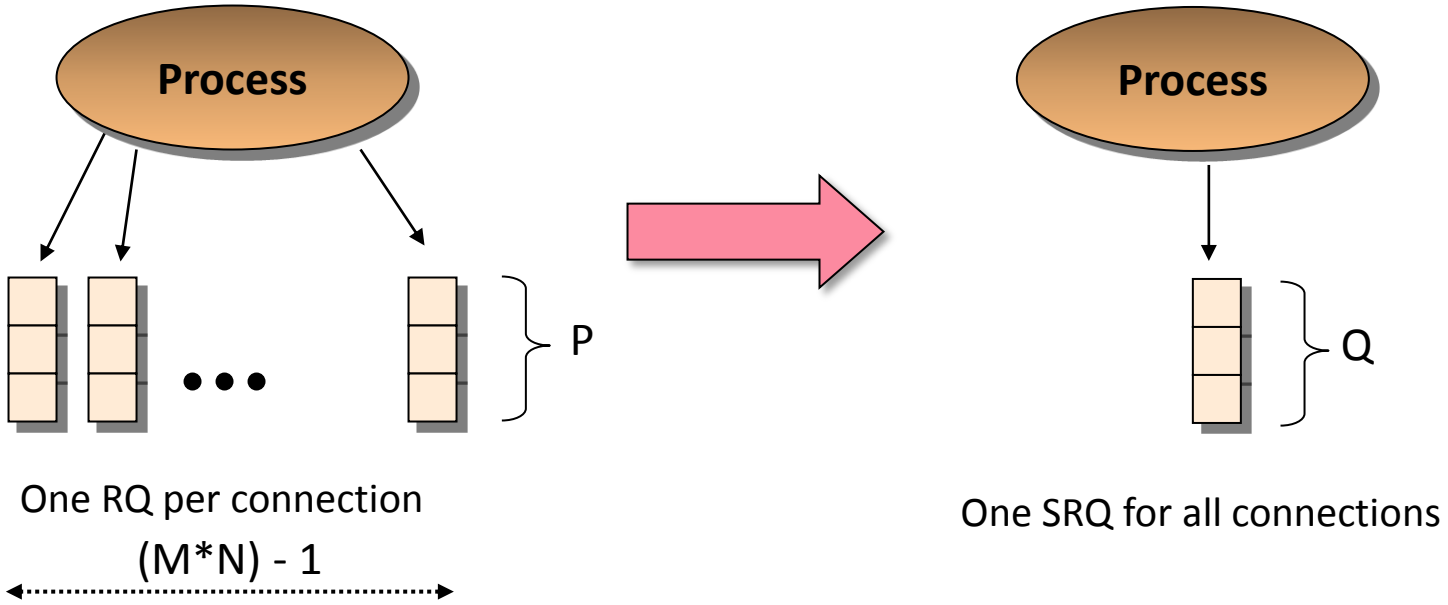
Common Challenges in Building HEC Systems with IB and HSE

- **Network adapters and interactions with other components**
  - I/O bus interactions and limitations
  - Multi-port adapters and bottlenecks
  - NUMA interactions
  - **Memory overheads in large-scale systems**
  - **Collective offload support**
- Network switches
- Network bridges

Memory overheads in large-scale systems

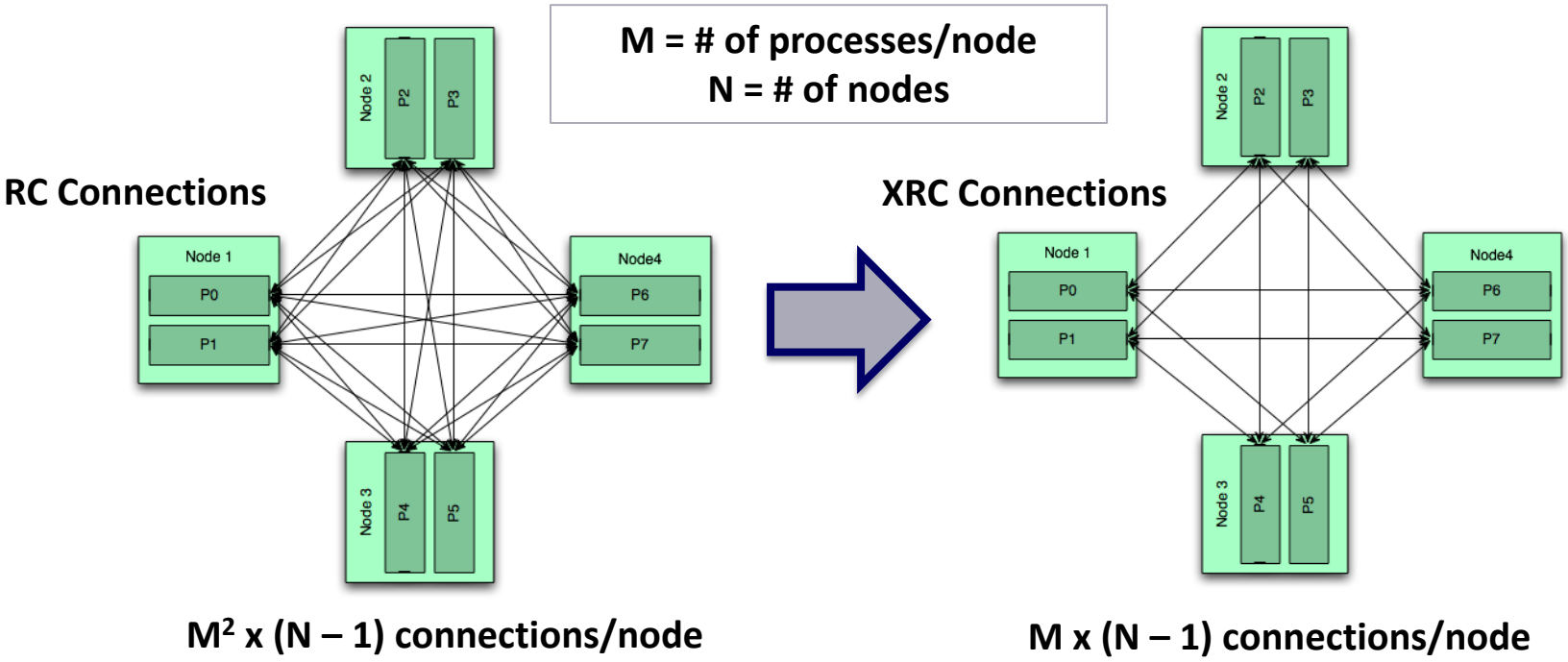
- Different transport protocols with IB
  - Reliable Connection (RC) is the most common
  - Unreliable Datagram (UD) is used in some cases
- Buffers need to be posted at each receiver to receive message from any sender
  - Buffer requirement can increase with system size
- Connections need to be established across processes under RC
  - Each connection requires certain amount of memory for handling related data structures
  - Memory required for all connections can increase with system size
- Both issues have become critical as large-scale IB deployments have taken place
  - Being addressed by both IB specification and upper-level middleware

Shared Receive Queue (SRQ)



- SRQ is a hardware mechanism for a process to share receive resources (memory) across multiple connections
  - Introduced in specification v1.2
- 0 < Q << P*((M*N)-1)

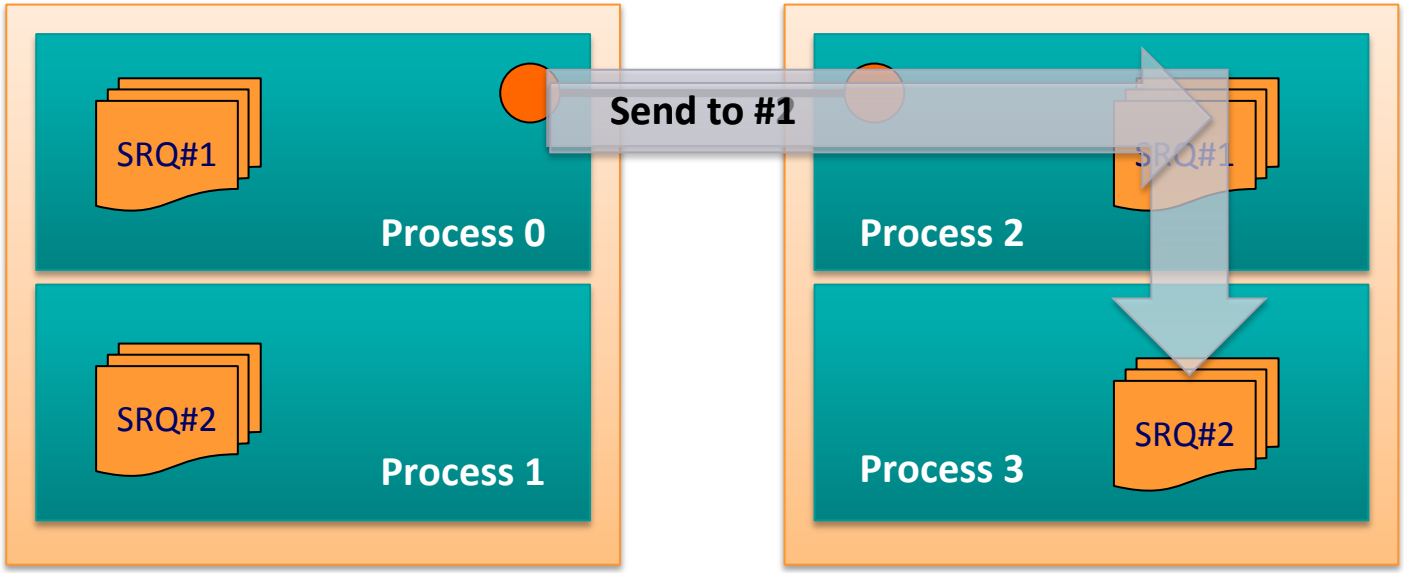
eXtended Reliable Connection (XRC)



- Each QP takes at least one page of memory
  - Connections between all processes is very costly for RC
- New IB Transport added: eXtended Reliable Connection
  - Allows connections between nodes instead of processes

XRC Addressing

- XRC uses SRQ Numbers (SRQN) to direct where a operation should complete



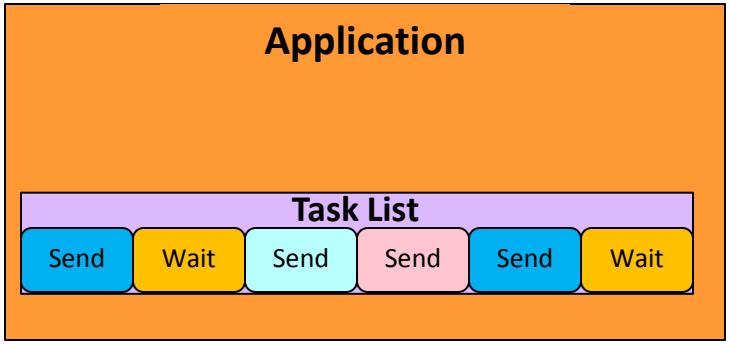
- Hardware does all routing of data, so p2 is not actually involved in the data transfer
- *Connections are not bi-directional*, so p3 cannot sent to p0

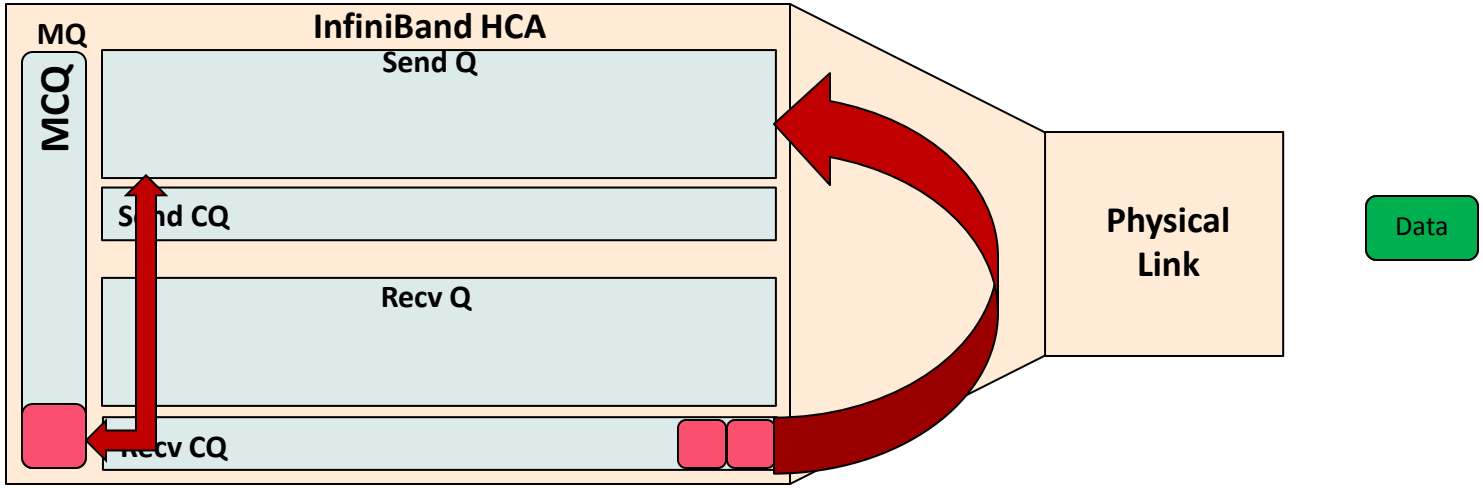
Collective Offload Support on the Adapters

- Performance of collective operations (broadcast, barrier, reduction, all-reduce, etc.) are very critical to the overall performance of MPI applications
- Currently being done with basic pt-to-pt operations (send/recv and RDMA) using host-based operations
- Mellanox ConnectX-2, ConnectX-3 and ConnectIB adapters support offloading some of these operations to the adapters (CORE-Direct)
  - Provides overlap of computation and collective communication
  - Reduces OS jitter (since everything is done in hardware)

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 HCA wait for ACK packet from the receiver





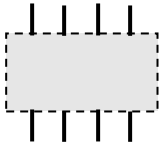
Common Challenges in Building HEC Systems with IB and HSE

- Network adapters and interactions with other components
- **Network switches**
  - **Switch topologies**
  - **Switching and Routing**
- Network bridges

Switch Topologies

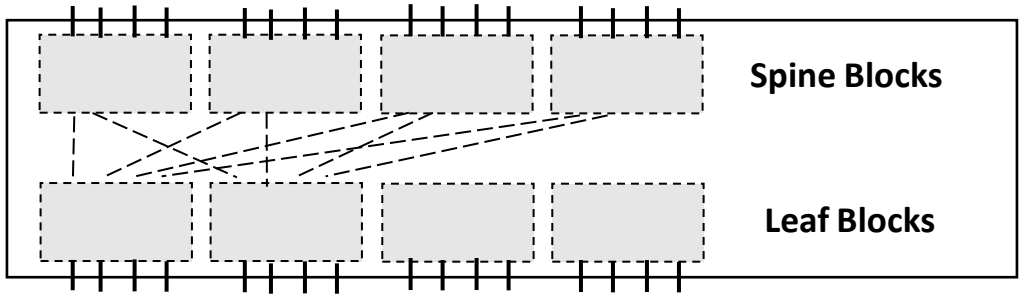
- InfiniBand installations come in multiple topologies
  - Single crossbar switches (up to 36-ports for QDR or FDR)
    - Applicable only to very small systems (hard to scale to large clusters)
  - Fat-tree topologies (medium scale topologies)
    - Provides full bisection bandwidth: Given independent communication between processes, you can find a switch configuration that provides fully non-blocking paths (though the same configuration might have contention if the communication pattern changes)
    - Issue: Number of switch components increases super-linearly with the number of nodes (Not scalable for large-scale systems)
- Large scale installations can use more conservative topologies
  - Partial fat-tree topologies (over-provisioning)
  - 3D Torus (Sandia Red Sky and SDSC Gordon), Hypercube (SGI Altix) topologies, and 10D Hypercube (NASA Pleiades)

Switch Topology: Absolute Performance vs. Scalability



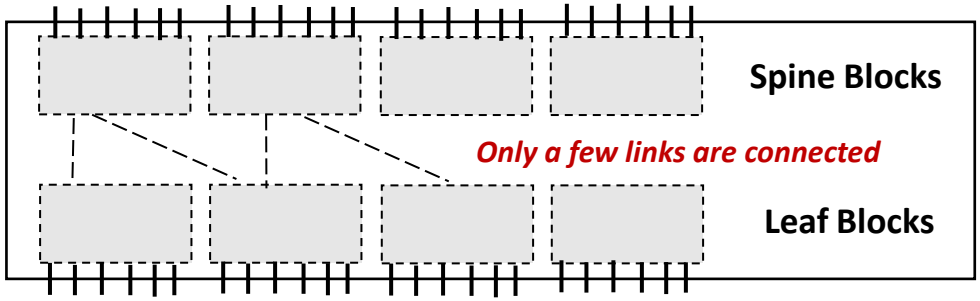
Crossbar ASIC

(all-to-all connectivity)



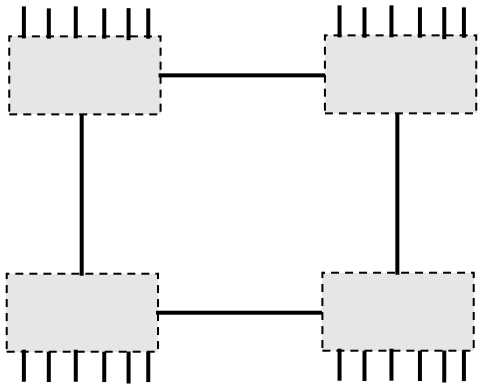
Full Fat-tree Topology

(full bisection bandwidth)



Partial Fat-tree Topology

(reduced inter-switch connectivity for more out-ports: super-linear scaling of switch components, but slower than a full fat-tree topology)



Torus/Hypercube Topology

(linear scaling of switch components)

Static Routing in IB + Adaptive Routing models from Qlogic and Mellanox

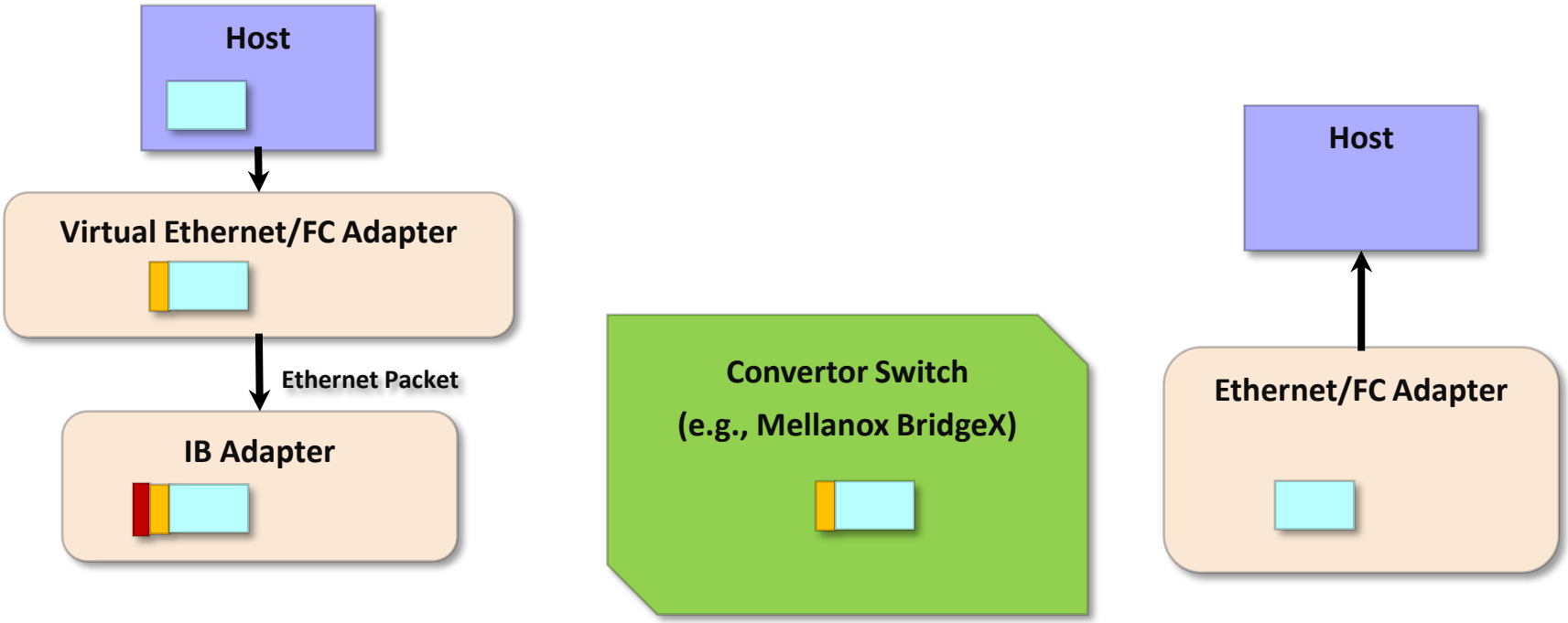
- IB standard only supports static routing
  - Not scalable for large systems where traffic might be non-deterministic causing hot-spots
- Next generation IB switches are supporting adaptive routing (in addition to static routing): Outside the IB standard
- Qlogic support for adaptive routing
  - Continually monitors application messaging patterns and selects the optimum path for each traffic flow, eliminating slowdowns caused by pathway bottlenecks
  - Dispersive routing load-balances traffic among multiple pathways
  - http://ir.qlogic.com/phoenix.zhtml?c=85695&p=irol-newsarticle&id=1428788
- Mellanox support for adaptive routing
  - Supports moving traffic via multiple parallel paths
  - Dynamically and automatically re-routes traffic to alleviate congested ports
  - http://www.mellanox.com/related-docs/prod_silicon/PB_InfiniScale_IV.pdf

Common Challenges in Building HEC Systems with IB and HSE

- Network adapters and interactions with other components
- Network switches
- **Network bridges**
  - **IB interoperability with Ethernet and FC**

IB-Ethernet and IB-FC Bridging Solutions

- Mainly developed for backward compatibility with existing infrastructure
  - Ethernet over IB (EoIB)
  - Fibre Channel over IB (FCoIB)



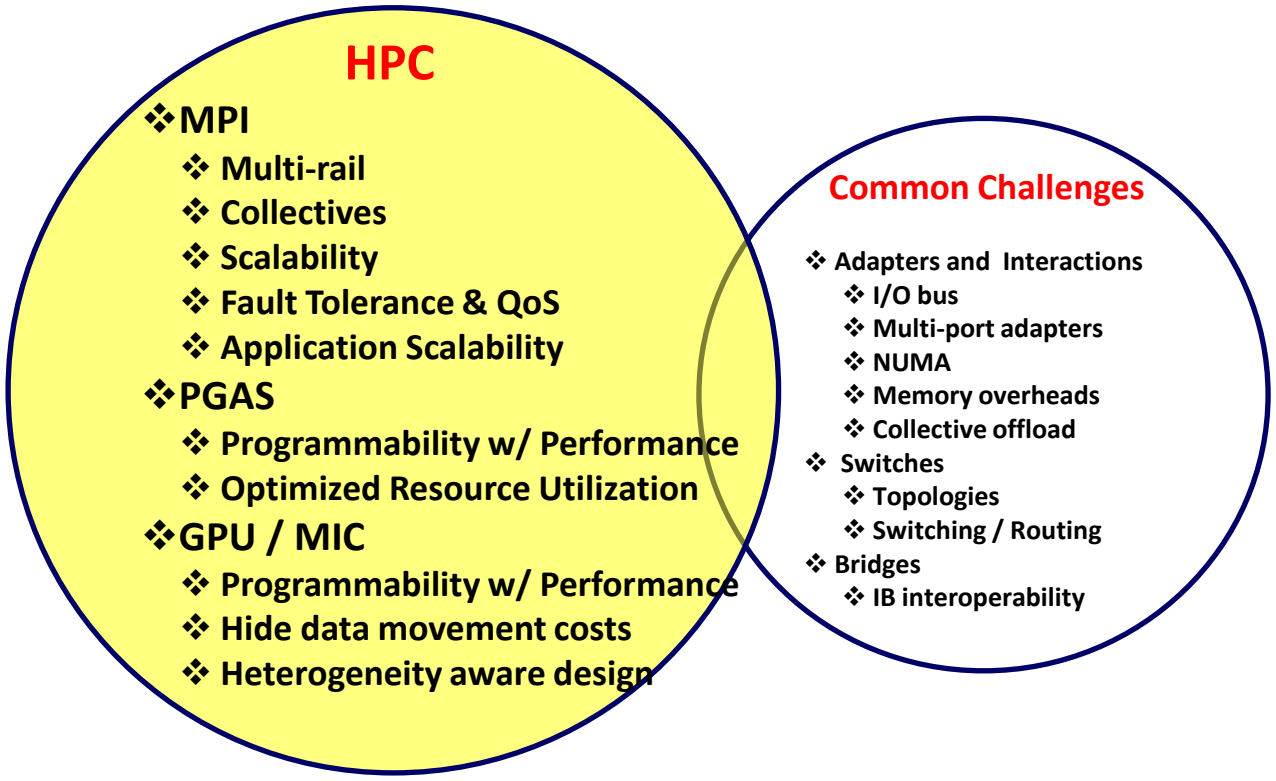
Ethernet/FC over IB

- Can be used in an infrastructure where a part of the nodes are connected over Ethernet or FC
  - All of the IB connected nodes can communicate over IB
  - The same nodes can communicate with nodes in the older infrastructure using Ethernet-over-IB or FC-over-IB
- Do not have the performance benefits of IB
  - Host thinks it is using an Ethernet or FC adapter
  - For example, with Ethernet, communication will be using TCP/IP
    - There is some hardware support for segmentation offload, but the rest of the IB features are unutilized
- Note that this is different from VPI, as there is only one network connectivity from the adapter

Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- **System Specific Challenges and Case Studies**
  - **HPC (MPI, PGAS and GPU/MIC Computing)**
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - Storage and File Systems
  - Grid Computing
- Open Fabrics Software Stack and RDMA Programming
- Network Management Infrastructure and Tools
- Conclusions and Final Q&A

System Specific Challenges for HPC Systems



HPC System Challenges and Case Studies

- **Message Passing Interface (MPI)**
- Partitioned Global Address Space (PGAS) models
- GPU Computing
- MIC Computing

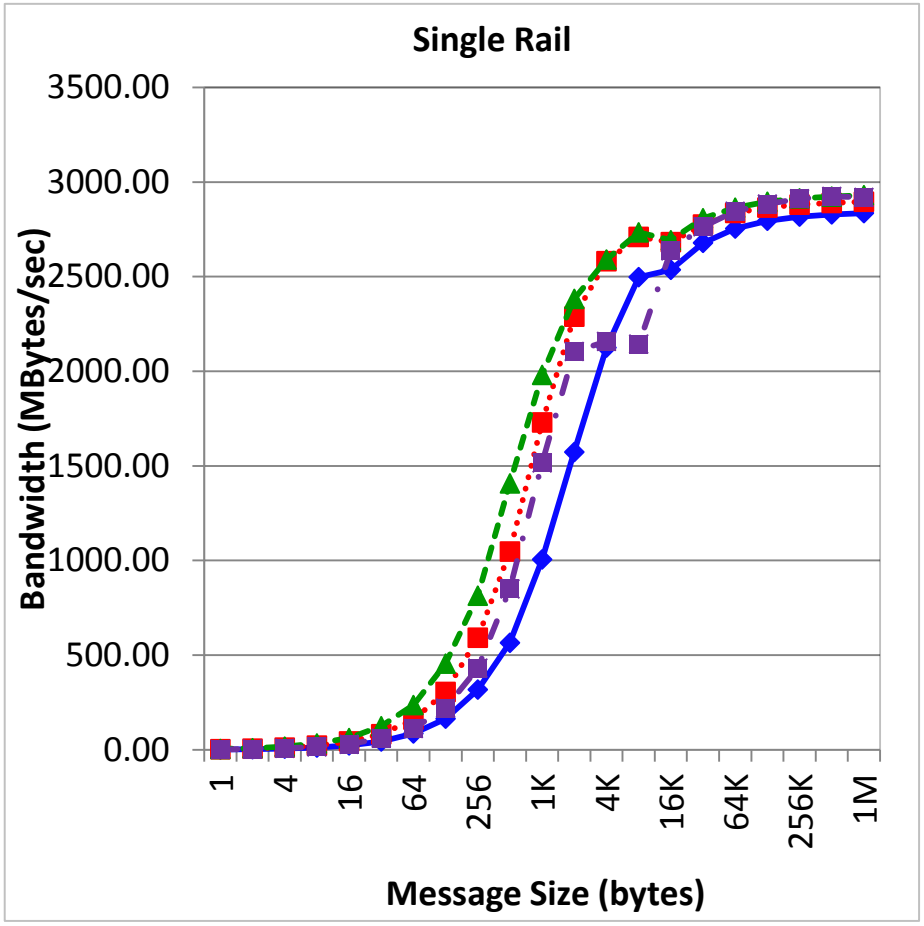
MVAPICH2/MVAPICH2-X Software

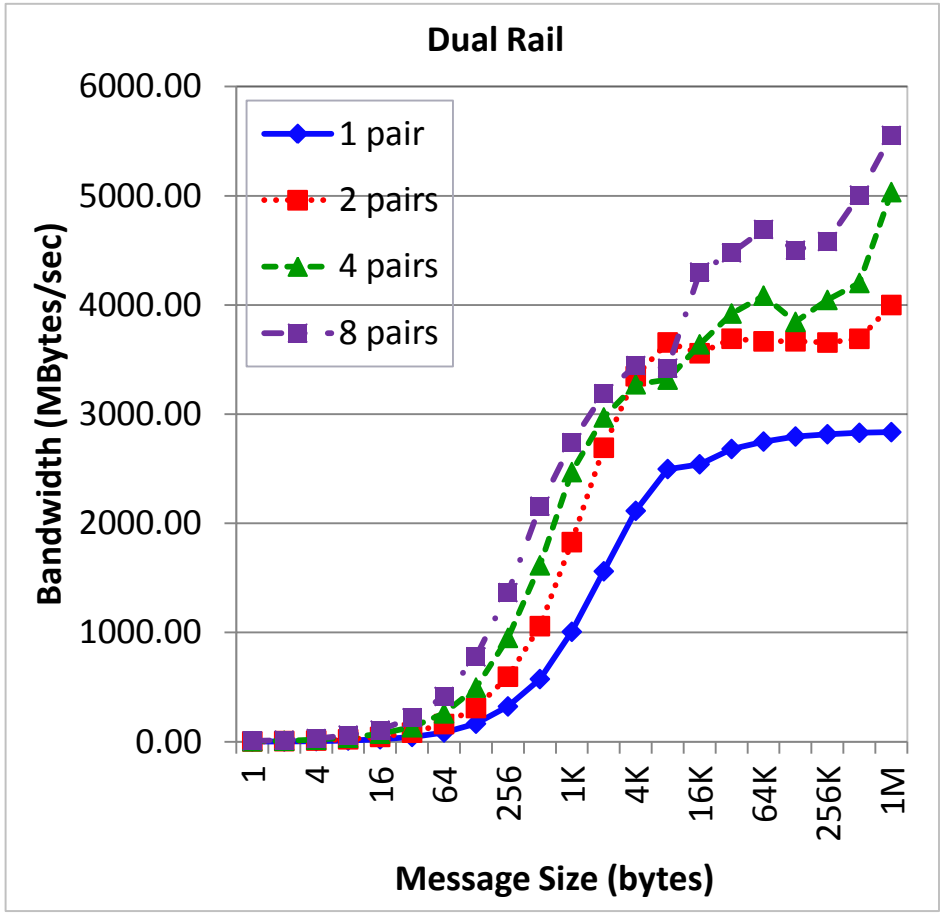
- High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1) ,MVAPICH2 (MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Used by more than 2,000 organizations (HPC Centers, Industry and Universities) in 70 countries
  - More than 173,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 7th ranked 204,900-core cluster (Stampede) at TACC
    - 14th ranked 125,980-core cluster (Pleiades) at NASA
    - 17th ranked 73,278-core cluster (Tsubame 2.0) at Tokyo Institute of Technology
    - and many others
  - Available with software stacks of many IB, HSE and server vendors including Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu
- Partner in the U.S. NSF-TACC Stampede System

Design Challenges and Sample Results

- **Interaction with Multi-Rail Environments**
- **Collective Communication**
- Scalability for Large-scale Systems
- Fault Tolerance
- Quality of Service
- Application Scalability

MPI Bandwidth on ConnectX with Multi-Rail

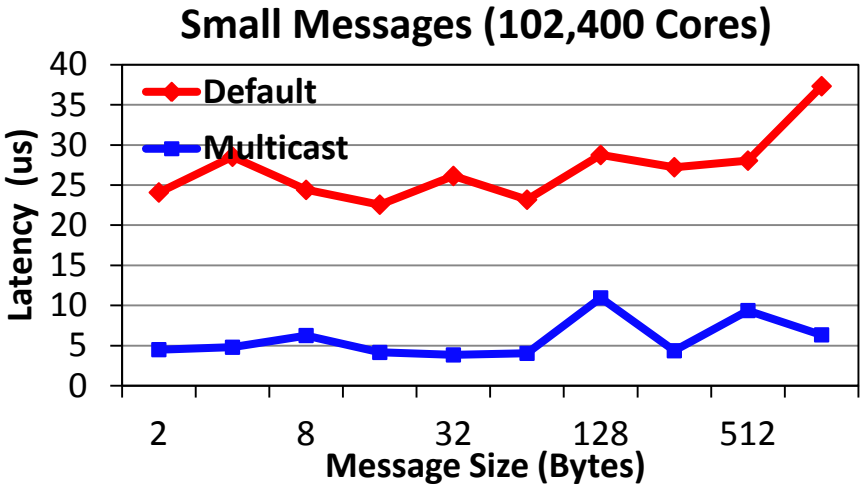


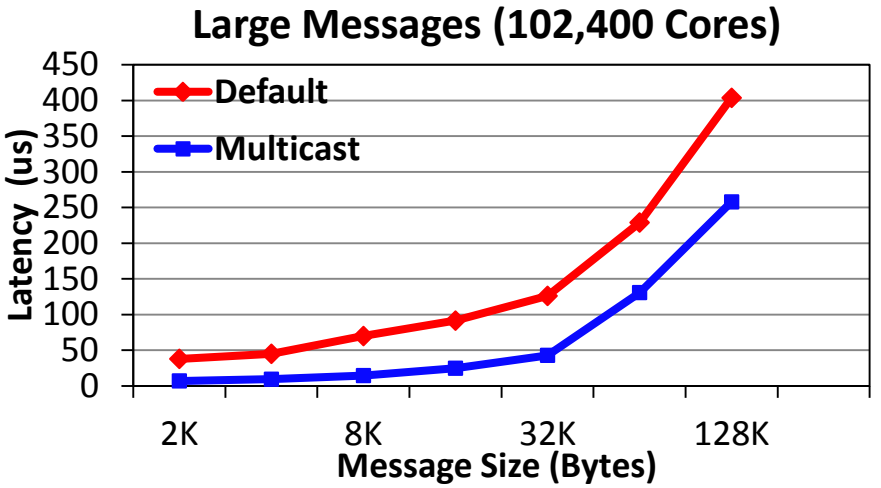


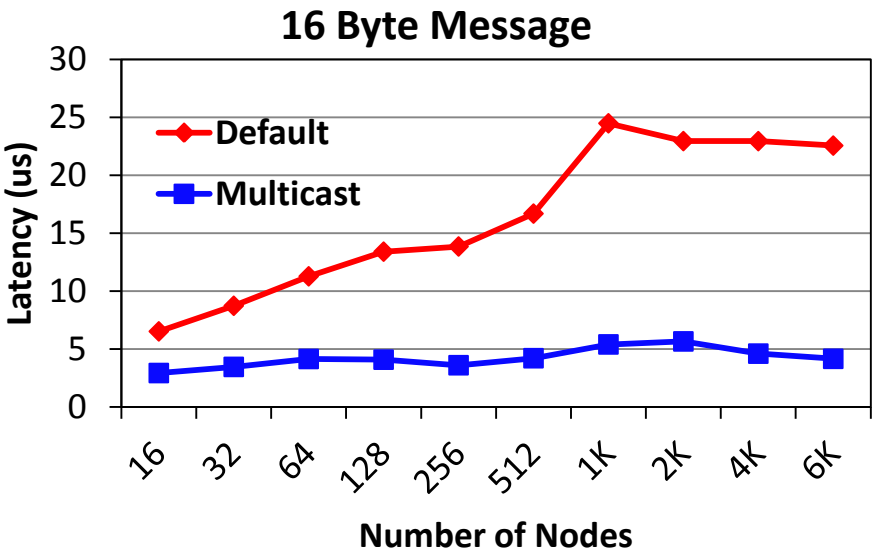
ConnectX-QDR: 2.2 GHz AMD Magny-Cours with IB switch

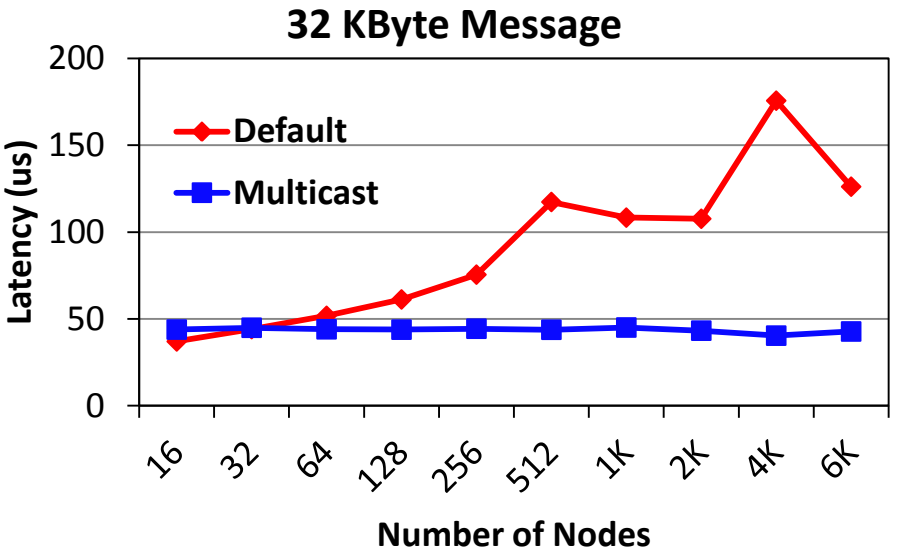
S. Sur, M. J. Koop, L. Chai and D. K. Panda, "Performance Analysis and Evaluation of Mellanox ConnectX InfiniBand Architecture with Multi-Core Platforms", IEEE Hot Interconnects, 2007

Hardware Multicast-aware MPI_Bcast on Stampede





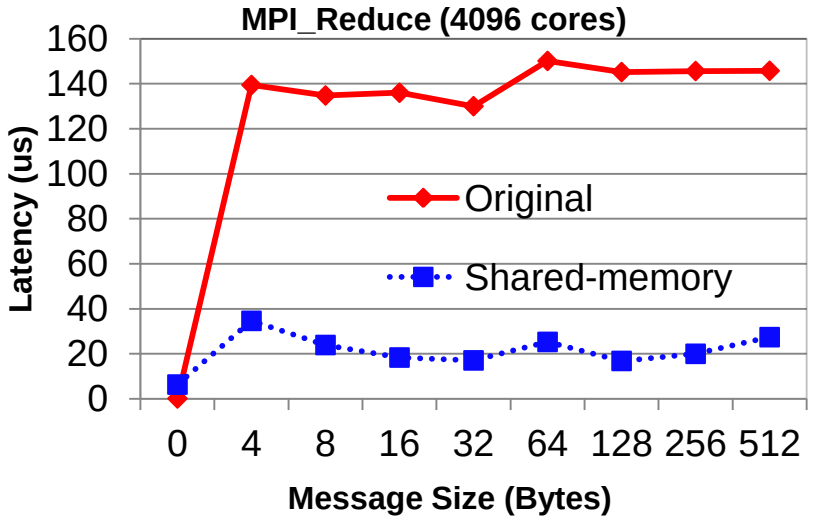




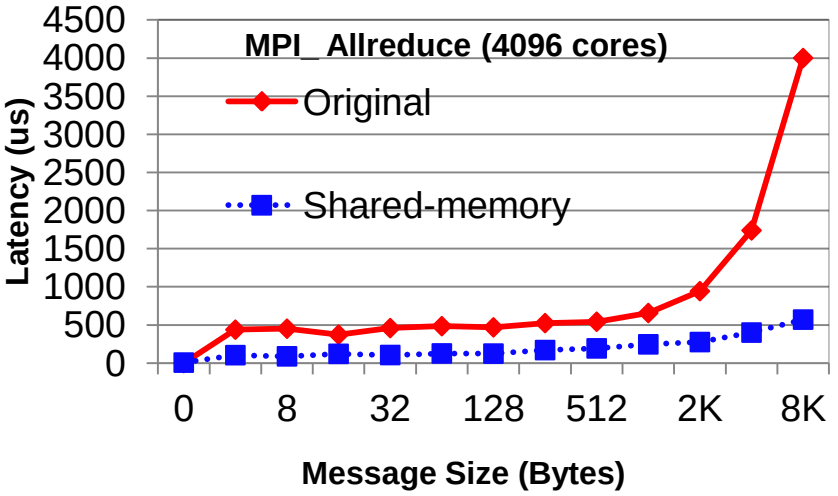
ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR switch

Shared-memory Aware Collectives

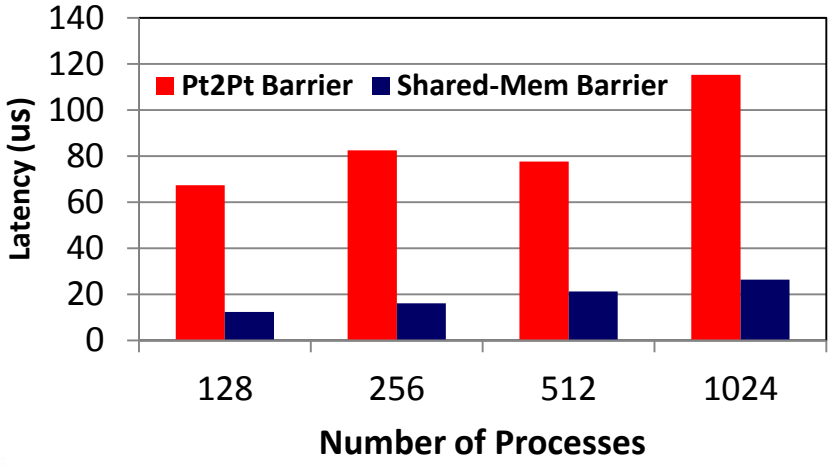
- MVAPICH2 Reduce/Allreduce with 4K cores on TACC Ranger (AMD Barcelona, SDR IB)



MV2_USE_SHMEM_REDUCE=0/1



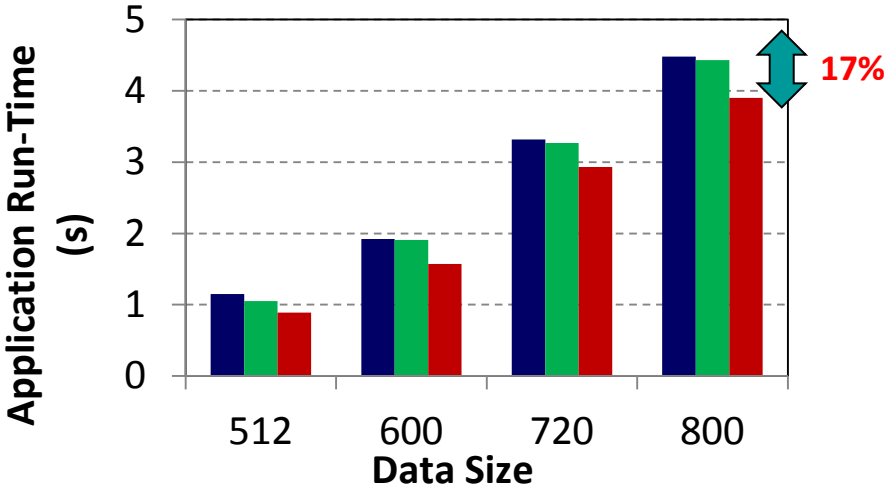
MV2_USE_SHMEM_ALLREDUCE=0/1



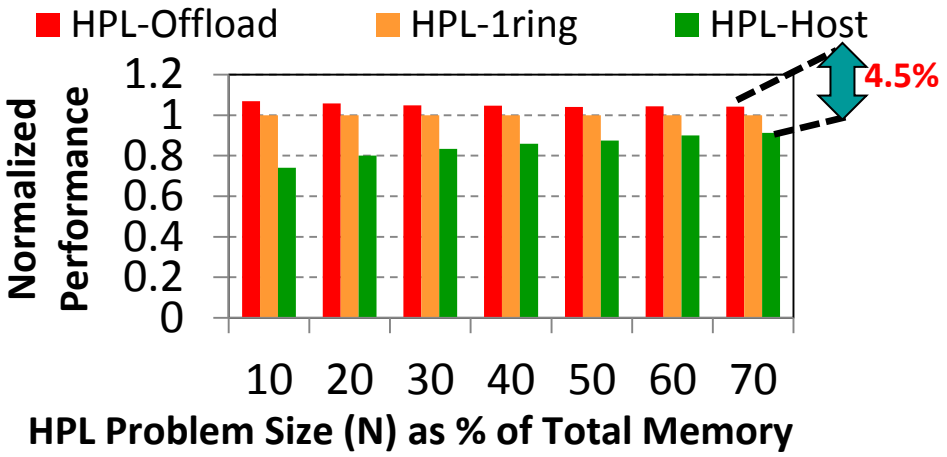
- MVAPICH2 Barrier with 1K Intel Westmere cores , QDR IB

MV2_USE_SHMEM_BARRIER=0/1

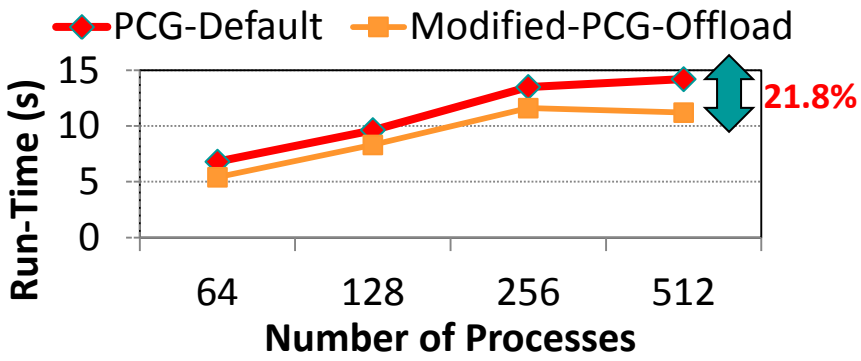
Application benefits with Non-Blocking Collectives based on CX-2 Collective Offload



Modified P3DFFT with Offload-Alltoall does up to 17% better than default version (128 Processes)



Modified HPL with Offload-Bcast does up to 4.5% better than default version (512 Processes)



Modified Pre-Conjugate Gradient Solver with Offload-Allreduce does up to 21.8% better than default version

K. Kandalla, et. al.. High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A Study with Parallel 3D FFT. ISC 2011

K. Kandalla, et. al, Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, HotI 2011

K. Kandalla, et. al., Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers, IPDPS '12

Can Network-Offload based Non-Blocking Neighborhood MPI Collectives Improve Communication Overheads of Irregular Graph Algorithms? K. Kandalla, A. Buluc, H. Subramoni, K. Tomko, J. Vienne, L. Oliker, and D. K. Panda, IWPAPS' 12

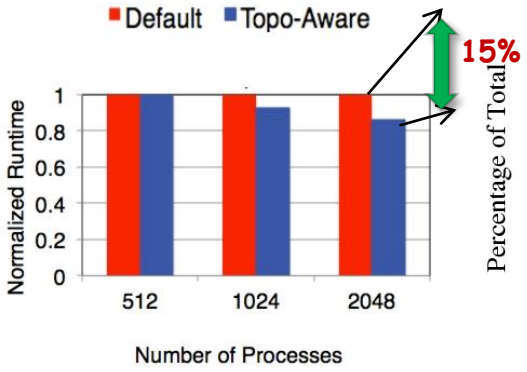
Network-Topology-Aware Placement of Processes

Can we design a highly scalable network topology detection service for IB?

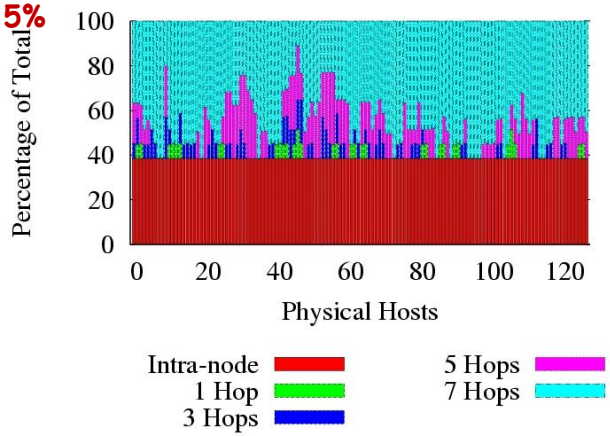
How do we design the MPI communication library in a network-topology-aware manner to efficiently leverage the topology information generated by our service?

What are the potential benefits of using a network-topology-aware MPI library on the performance of parallel scientific applications?

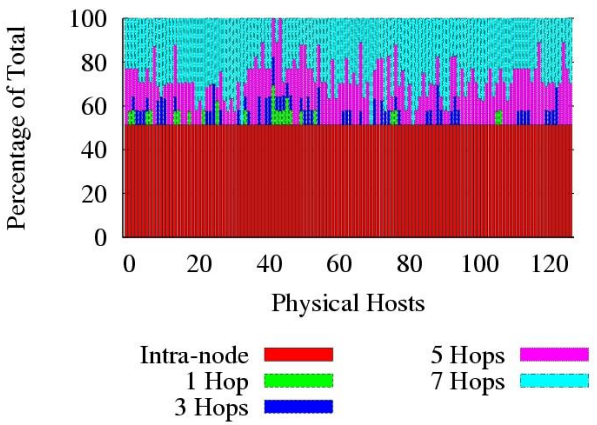
Overall performance and Split up of physical communication for MILC on Ranger



Performance for varying system sizes



Default for 2048 core run



Topo-Aware for 2048 core run

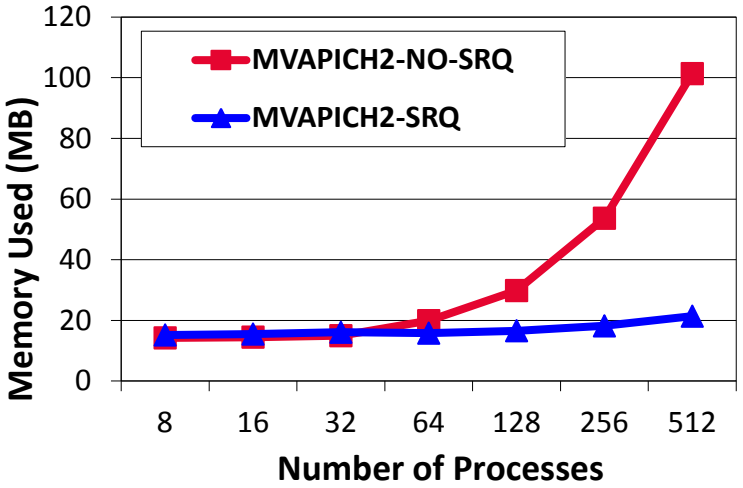
- Reduce network topology discovery time from $O(N^2_{hosts})$ to $O(N_{hosts})$
- 15% improvement in MILC execution time @ 2048 cores
- 15% improvement in Hypre execution time @ 1024 cores

H. Subramoni, S. Potluri, K. Kandalla, B. Barth, J. Vienne, J. Keasler, K. Tomko, K. Schulz, A. Moody, and D. K. Panda, Design of a Scalable InfiniBand Topology Service to Enable Network-Topology-Aware Placement of Processes, SC'12 . BEST Paper and BEST STUDENT Paper Finalist

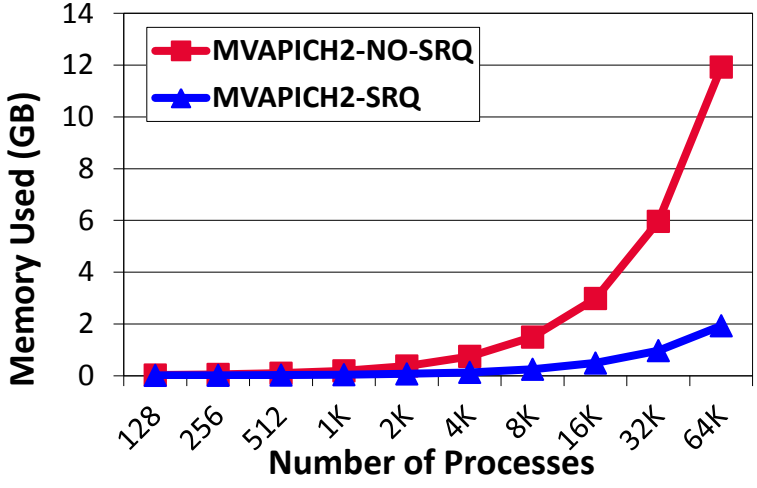
Design Challenges and Sample Results

- Interaction with Multi-core Environments
- Collective Communication
- **Scalability for Large-scale Systems**
  - **Memory Efficient Communication**
- Fault Tolerance
- Quality of Service
- Application Scalability

Memory Utilization using Shared Receive Queues, UD



MPI_Init memory utilization

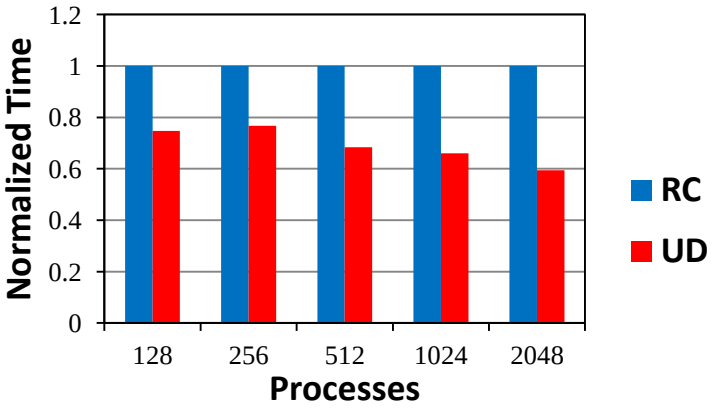


Analytical model

- **SRQ reduces the memory used by 1/6th at 64,000 processes**

S. Sur, L. Chai, H. –W. Jin and D. K. Panda, "Shared Receive Queue Based Scalable MPI Design for InfiniBand Clusters", IPDPS 2006

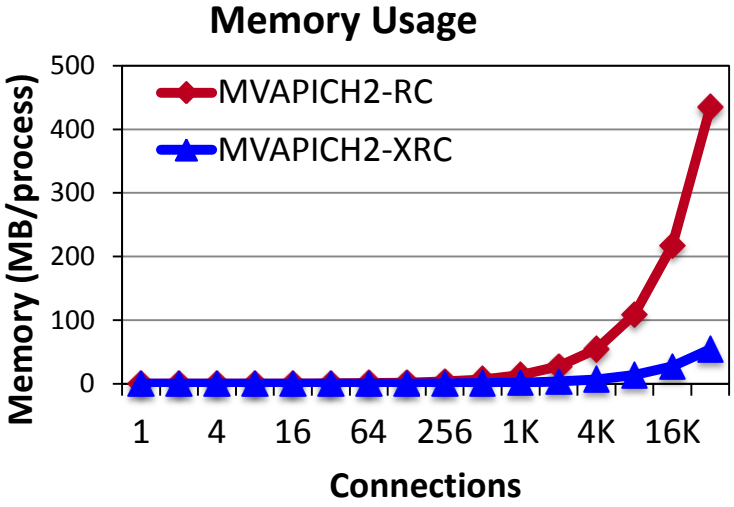
	RC (MVAPICH2 1.8)				UD (MVAPICH2 1.8)		
Number of Processes	**Conn.**	**Buffers**	**Struct**	**Total**	**Buffers**	**Struct**	**Total**
512	22.9	24	0.3	47.2	24	0.2	24.2
1024	29.5	24	0.6	54.1	24	0.4	24.4
2048	42.4	24	1.2	67.6	24	0.9	24.9

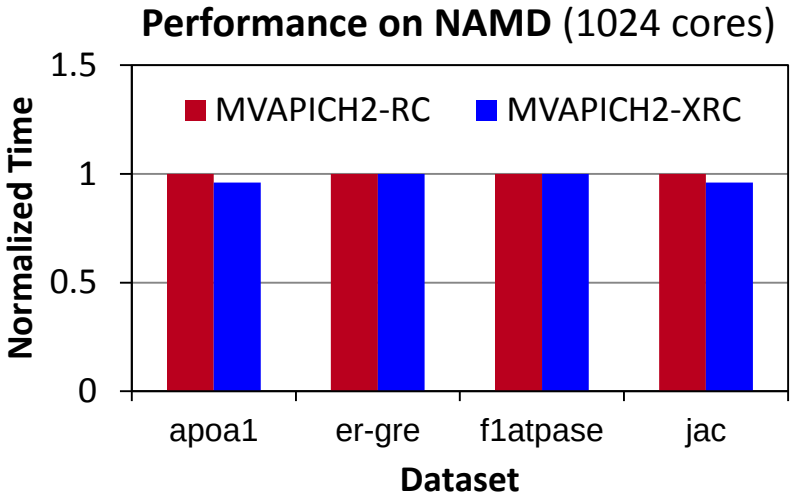


- **UD reduces HCA QP cache trashing**

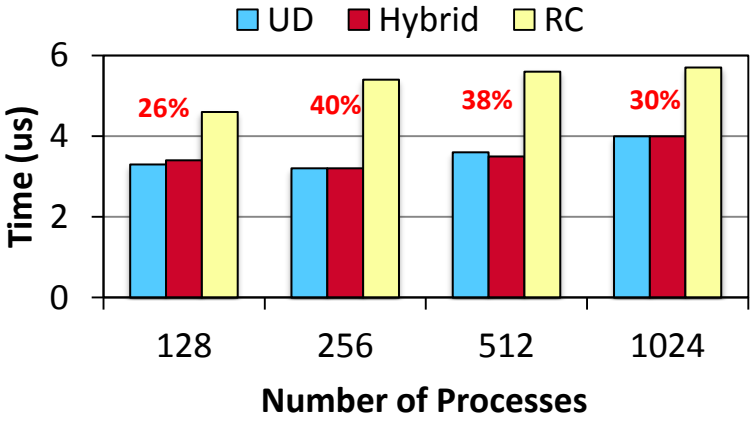
M. Koop, S. Sur, Q. Gao and D. K. Panda, "High Performance MPI Design using Unreliable Datagram for Ultra-Scale InfiniBand Clusters," ICS '07

eXtended Reliable Connection (XRC) and Hybrid Mode





- Memory usage for 32K processes with 8-cores per node can be 54 MB/process (for connections)
- NAMD performance improves when there is frequent communication to many peers



- Both UD and RC/XRC have benefits
  - Hybrid for the best of both
- Available since MVAPICH2 1.7 as integrated interface
- Runtime Parameters: RC - default;
  - UD - MV2_USE_ONLY_UD=1
  - Hybrid - MV2_HYBRID_ENABLE_THRESHOLD=1

M. Koop, J. Sridhar and D. K. Panda, "Scalable MPI Design over InfiniBand using eXtended Reliable Connection," Cluster '08

Design Challenges and Sample Results

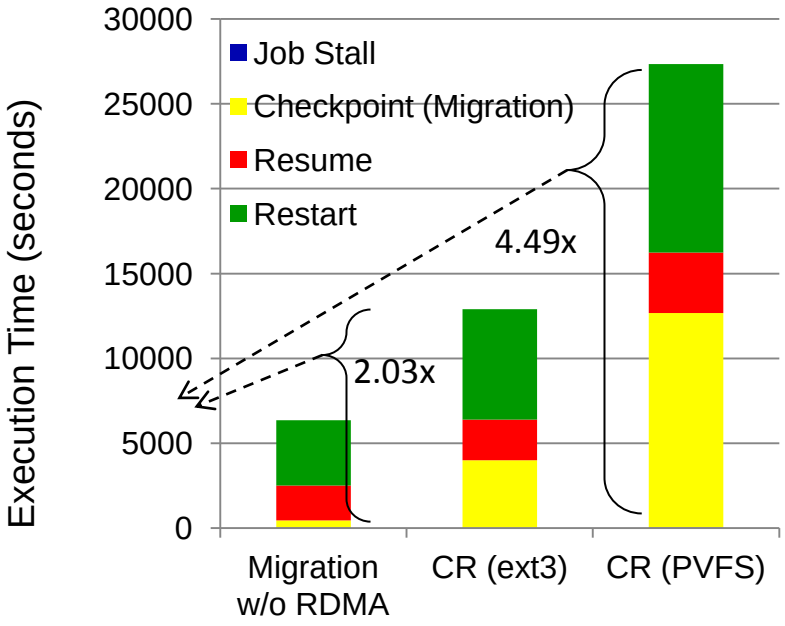
- Interaction with Multi-rail Environments
- Network Congestion and Hot-spots
- Collective Communication
- Scalability for Large-scale Systems
- **Fault Tolerance**
- **Quality of Service**
- **Application Scalability**

Fault Tolerance

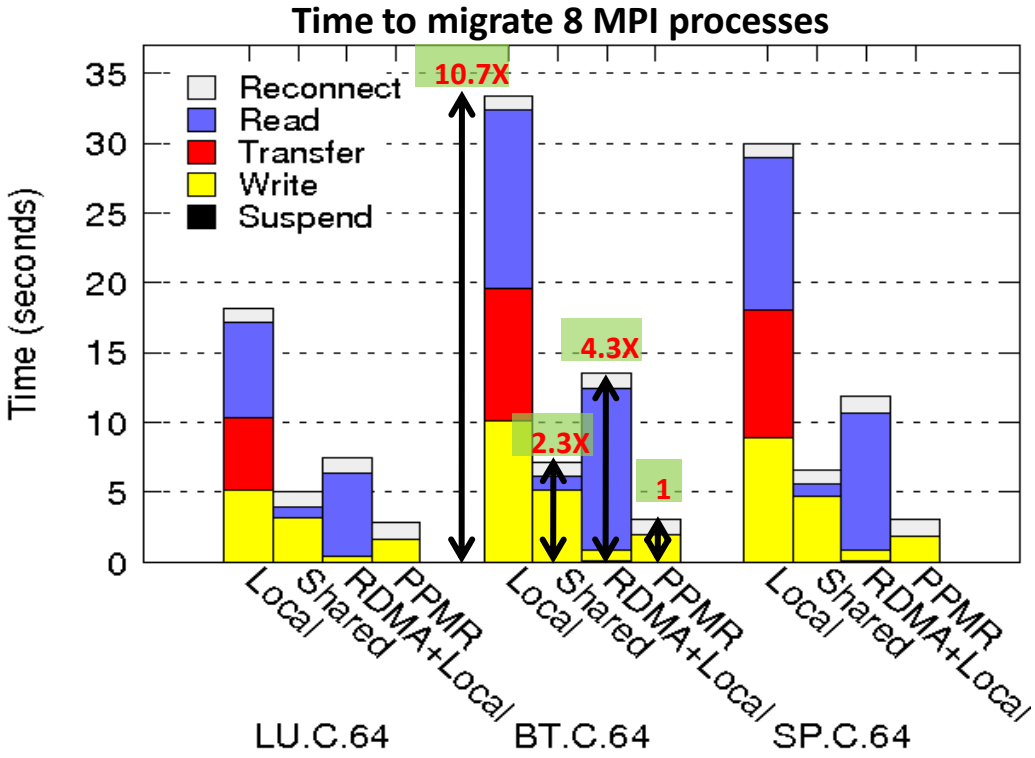
- Component failures are common in large-scale clusters
- Imposes need on reliability and fault tolerance
- Multiple challenges:
  - Checkpoint-Restart vs. Process Migration
  - Benefits of Scalable Checkpoint Restart (SCR) Support
    - Application guided
    - Application transparent

Checkpoint-Restart vs. Process Migration
Low Overhead Failure Prediction with IPMI

- Job-wide Checkpoint/Restart is not scalable
- Job-pause and Process Migration framework can deliver pro-active fault-tolerance
- Also allows for cluster-wide load-balancing by means of job compaction



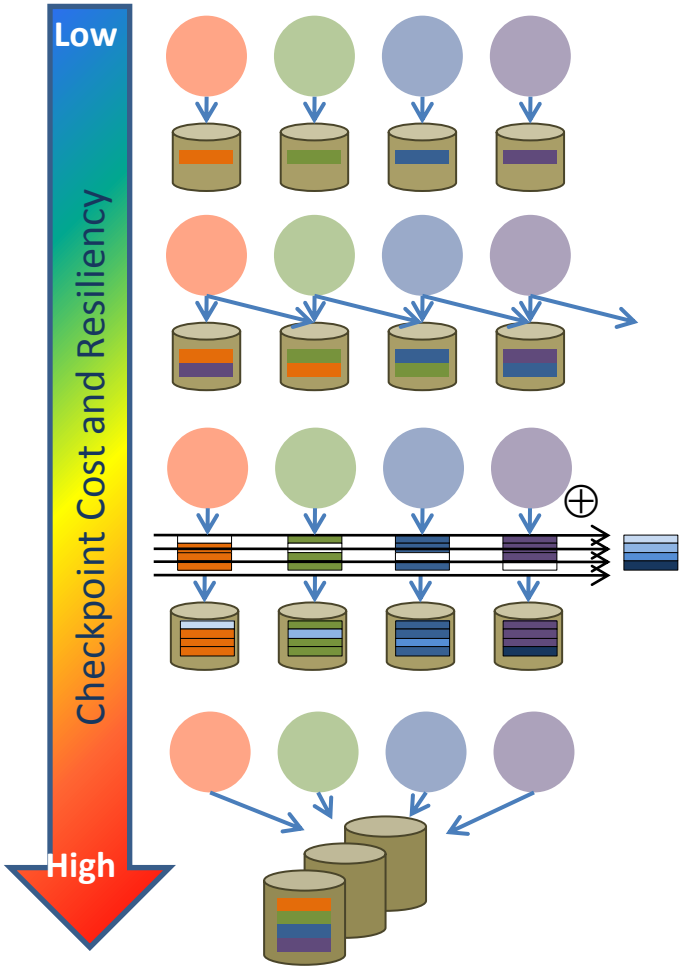
LU Class C Benchmark (64 Processes)



X. Ouyang, R. Rajachandrasekar, X. Besseron, D. K. Panda, High Performance Pipelined Process Migration with RDMA, CCGrid 2011

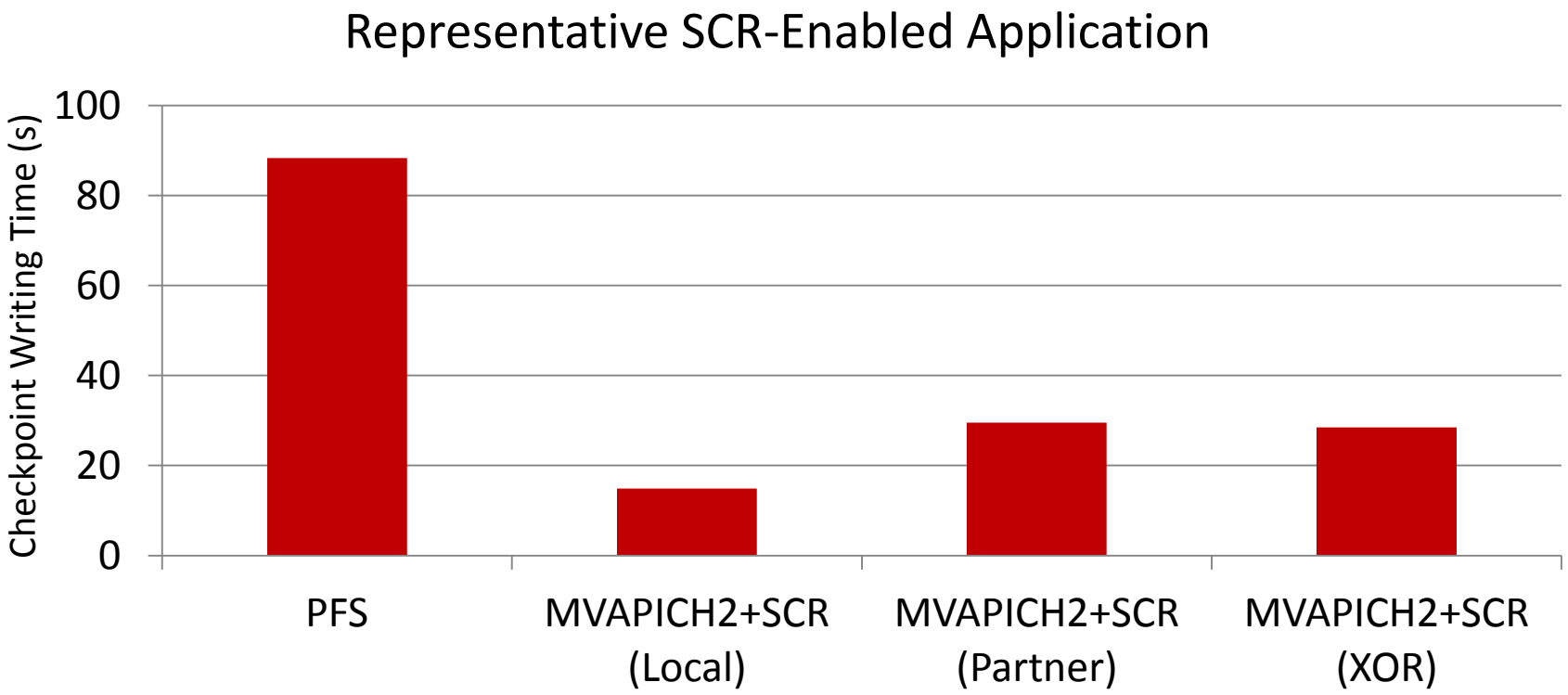
X. Ouyang, S. Marcarelli, R. Rajachandrasekar and D. K. Panda, RDMA-Based Job Migration Framework for MPI over InfiniBand, Cluster 2010

Multi-Level Checkpointing with ScalableCR (SCR)



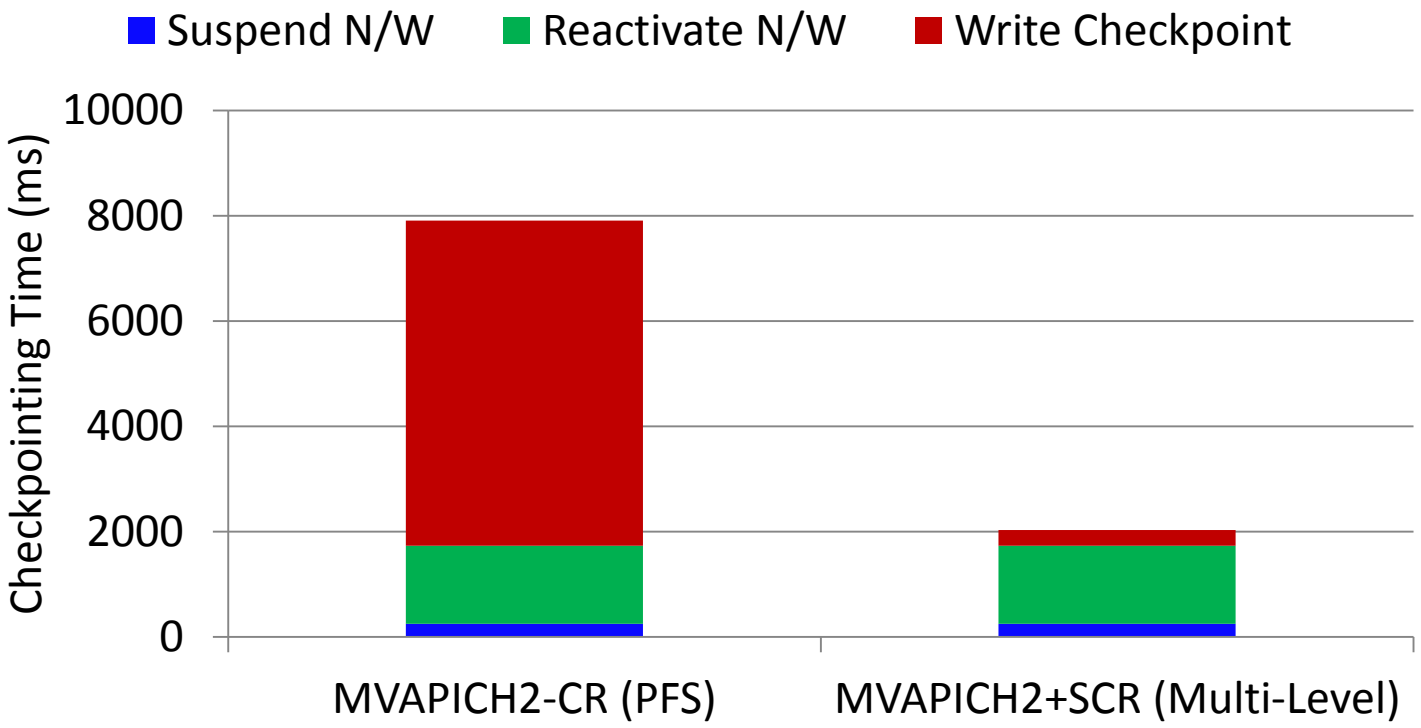
- LLNL's Scalable Checkpoint/Restart library
- Can be used for application guided and application transparent checkpointing
- Effective utilization of storage hierarchy
  - **Local:** Store checkpoint data on node's local storage, e.g. local disk, ramdisk
  - **Partner:** Write to local storage and on a partner node
  - **XOR:** Write file to local storage and small sets of nodes collectively compute and store parity redundancy data (RAID-5)
  - **Stable Storage:** Write to parallel file system

Application-guided Multi-Level Checkpointing



- Checkpoint writing phase times of representative SCR-enabled MPI application
- **512** MPI processes (8 procs/node)
- Approx. **51 GB** checkpoints

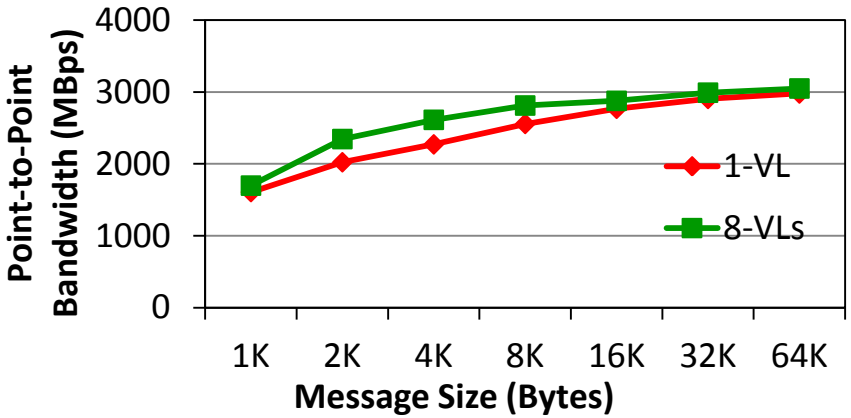
Transparent Multi-Level Checkpointing



- **ENZO Cosmology application** – Radiation Transport workload
- Using MVAPICH2's CR protocol instead of the application's in-built CR mechanism
- **512** MPI processes (8 procs/node)
- Approx. **12.8 GB** checkpoints

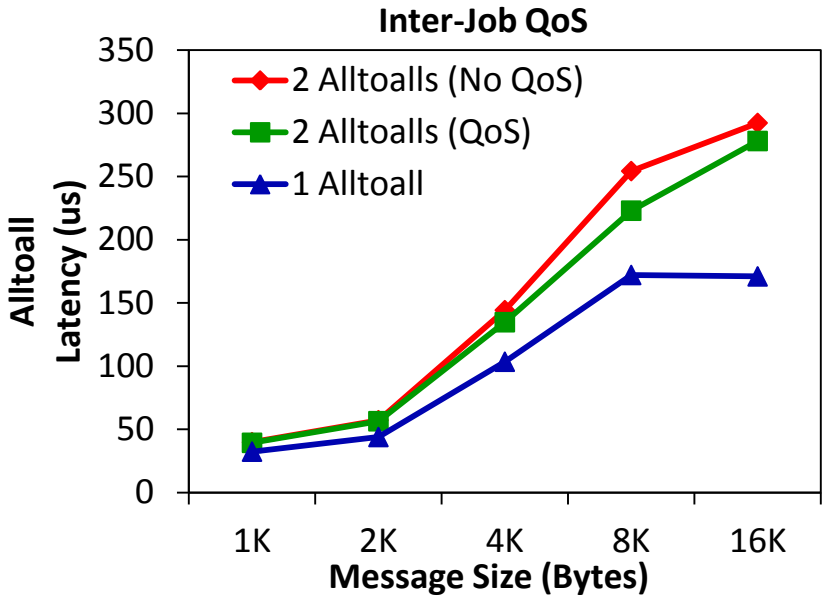
QoS in IB: MPI Performance with Multiple VLs & Inter-Job QoS

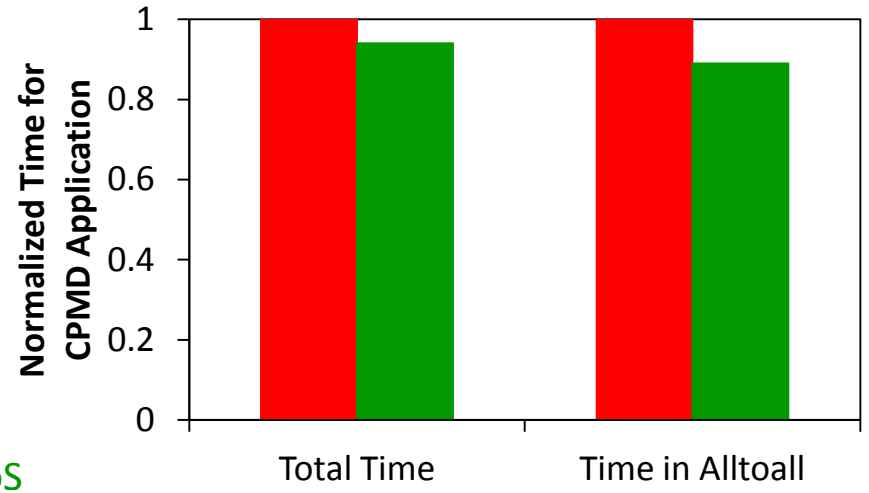
- IB is capable of providing network level differentiated service – QoS
- Uses Service Levels (SL) and Virtual Lanes (VL) to classify traffic



13% Performance improvement over One VL case

- Performance improvement over One VL case
  - Alltoall – 20 %
  - Application – 11%
- 12% performance improvement with Inter-Job QoS



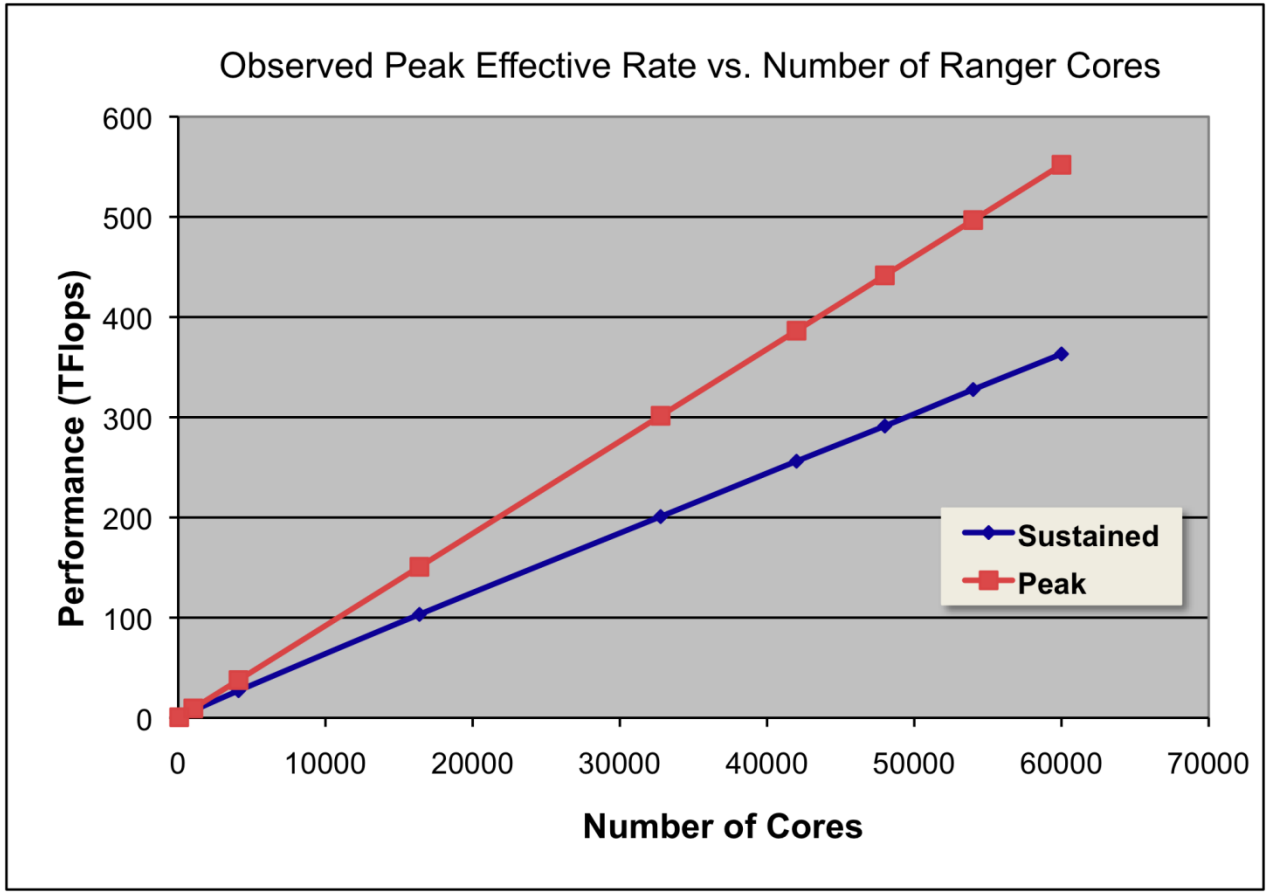


H. Subramoni, P. Lai, S. Sur and D. K. Panda, Improving Application Performance and Predictability using Multiple Virtual Lanes in Modern Multi-Core InfiniBand Clusters , Int'l Conference on Parallel Processing (ICPP '10), Sept. 2010.

Performance of HPC Applications on TACC Ranger using MVAPICH + IB

- Rob Farber's facial recognition application was run up to 60K cores using MVAPICH

- Ranges from 84% of peak at low end to 65% of peak at high end



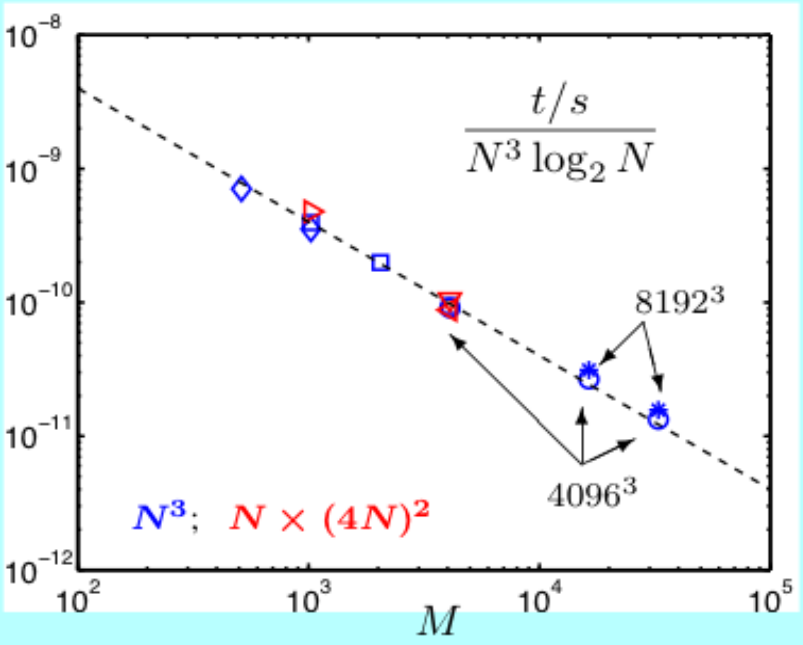
http://www.tacc.utexas.edu/research/users/features/index.php?m_b_c=farber

Performance of HPC Applications on TACC Ranger:
DNS/Turbulence

- 3D FFT flop count $\propto N^3 \log_2 N$
- Perfect scaling:

$$\frac{t/s}{N^3 \log_2 N} \propto M^{-1}$$

- **Strong scaling**: $> 98\%$ at both 4096^3 and 8192^3 from $M = 16K$ to $32K$
- **Weak scaling:** $\sim 80\%$ from $(N, M) = (2048, 2048)$ to $(8192, 32768)$

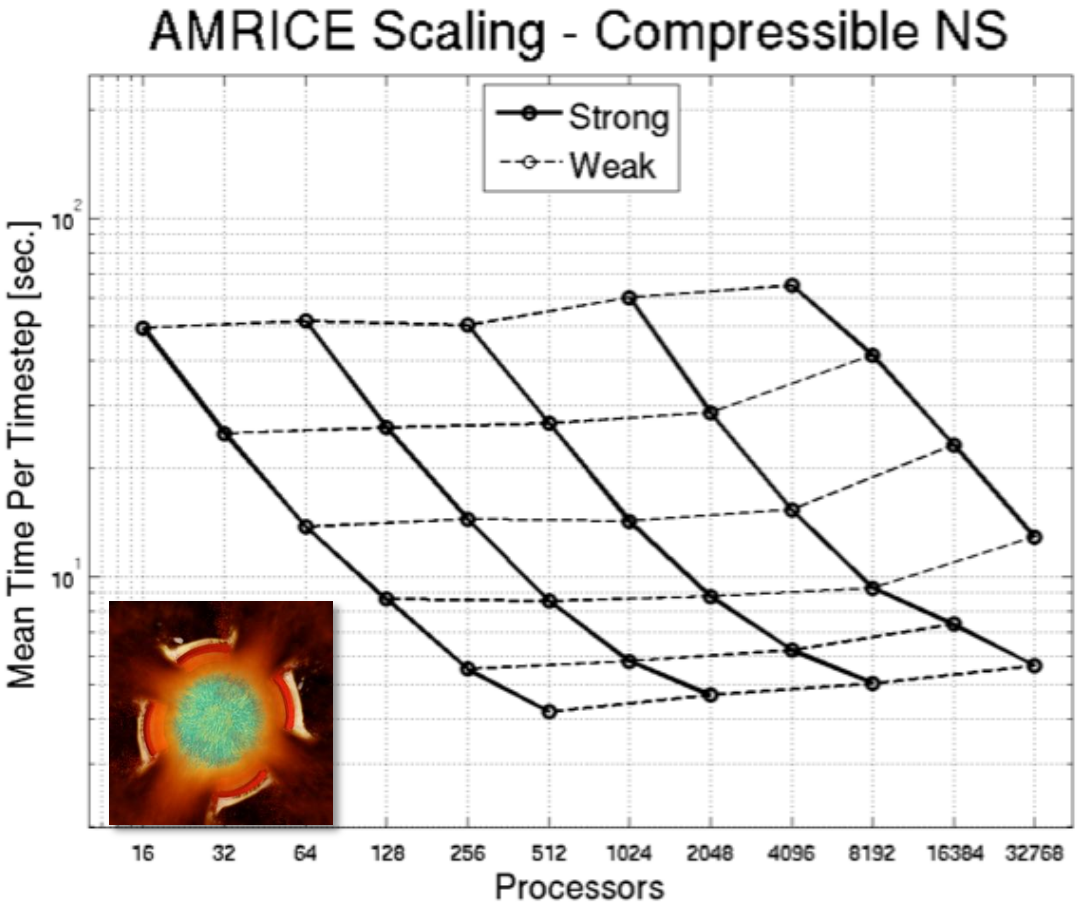


- Best timings for small M_1: row communicator within node (16 cores) or within socket (4 cores)

Courtesy: P.K. Yeung, Diego Donzis, TG 2008

Application Example: Blast Simulations

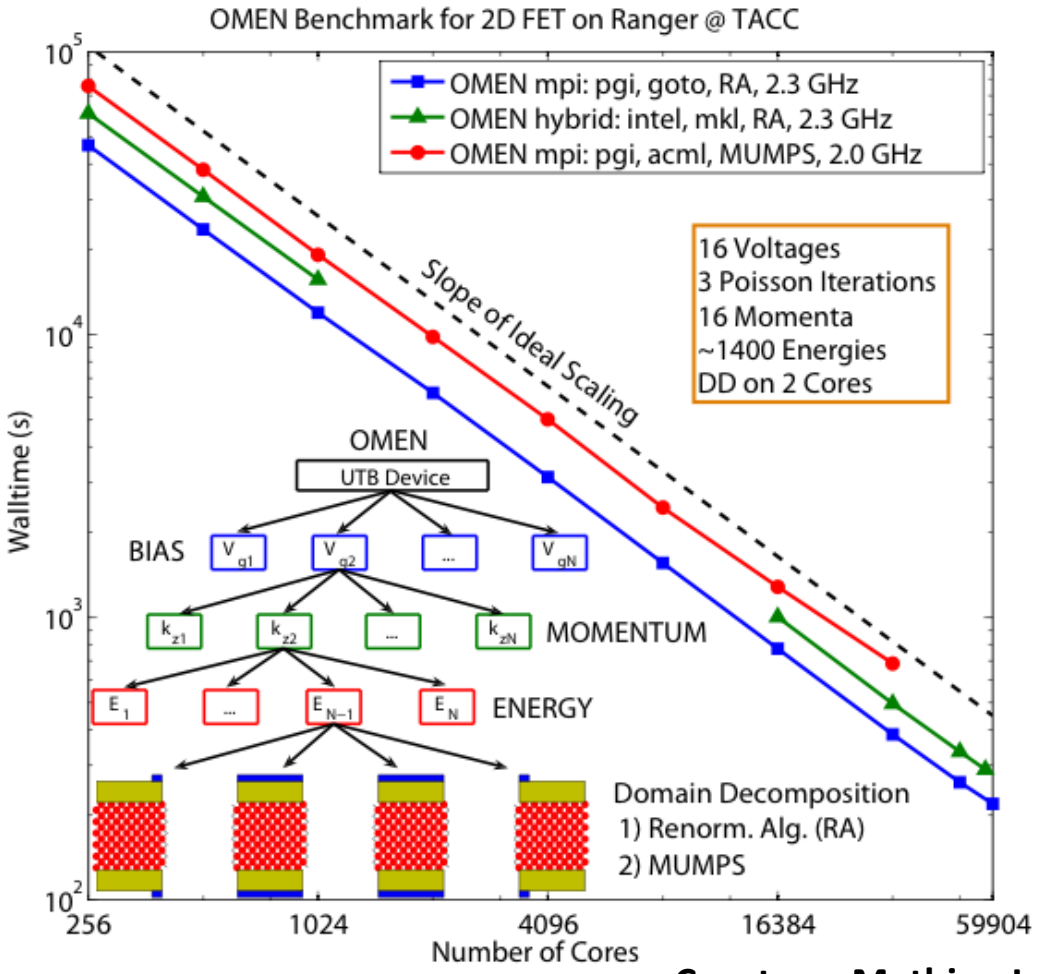
- Researchers from the University of Utah have developed a simulation framework, called Uintah
- Combines advanced mechanical, chemical and physical models into a novel computational framework
- Have run > 32K MPI tasks on Ranger
- Uses asynchronous communication



Courtesy: J. Luitjens, M. Bertzins, Univ of Utah

http://www.tacc.utexas.edu/news/feature-stories/2009/explosive-science/

Application Example: OMEN



- OMEN is a two- and three-dimensional Schrodinger-Poisson solver based
- Used in semi-conductor modeling
- Run to almost 60K tasks

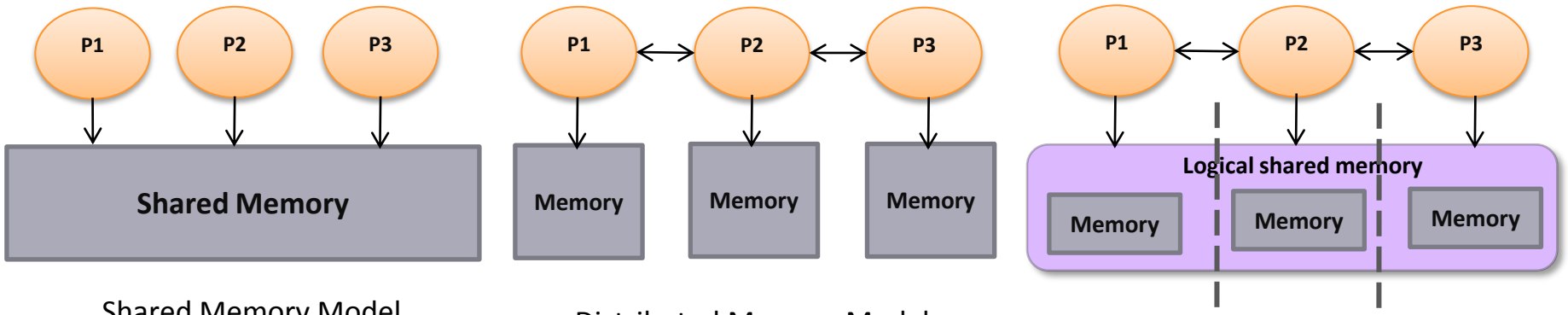
Courtesy: Mathieu Luisier, Gerhard Klimeck, Purde

http://www.tacc.utexas.edu/RangerImpact/pdf/Save_Our_Semiconductors.pdf

HPC System Challenges and Case Studies

- Message Passing Interface (MPI)
- **Partitioned Global Address Space (PGAS) models**
- GPU Computing
- MIC Computing

Partitioned Global Address Space (PGAS) Models



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

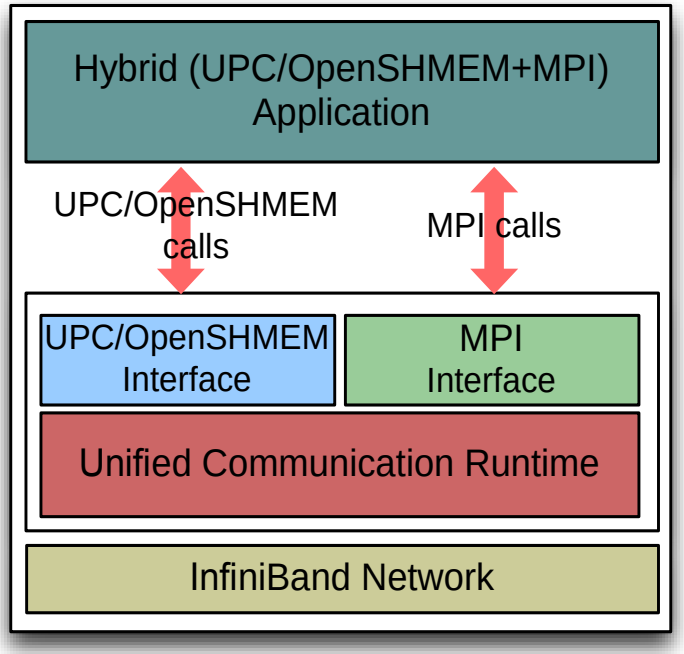
MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

- Global view improves programmer productivity
- Idea is to decouple data movement with process synchronization
- Processes should have asynchronous access to globally distributed data
- Well suited for irregular applications and kernels that require dynamic access to different data
- Different Approaches
  - Library-based (Global Arrays, OpenSHMEM)
  - Compiler-based (Unified Parallel C (UPC), Co-Array Fortran (CAF))
  - HPCS Language-based (X10, Chapel, Fortress)

Scalable OpenSHMEM/UPC and Hybrid (MPI, UPC and OpenSHMEM) designs

- Based on OpenSHMEM Reference Implementation (http://openshmem.org/) & UPC version 2.14.2 (http://upc.lbl.gov/)
  - Provides a design over GASNet
  - **Does not take advantage of all OFED features**
- Design Scalable and High-Performance OpenSHMEM & UPC over OFED
- Designing a Hybrid MPI + OpenSHMEM/UPC Model
  - Current Model – Separate Runtimes for OpenSHMEM/UPC and MPI
    - Possible deadlock if both runtimes are not progressed
    - Consumes more network resource
  - Our Approach – Single Unified Runtime for MPI and OpenSHMEM/UPC

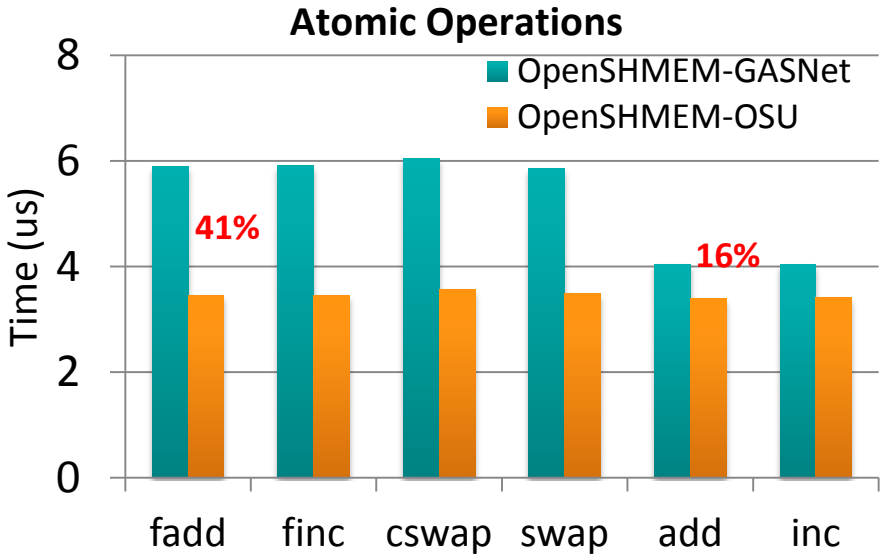


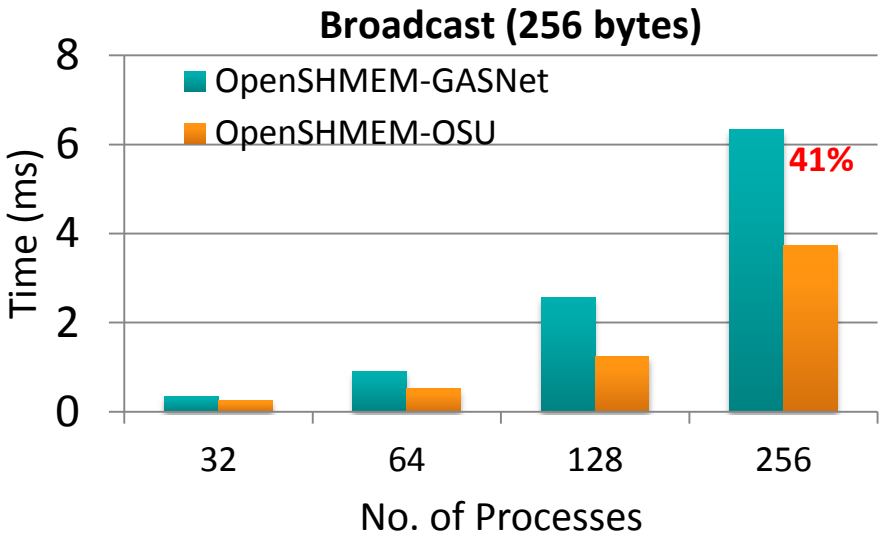
Hybrid MPI+OpenSHMEM/UPC

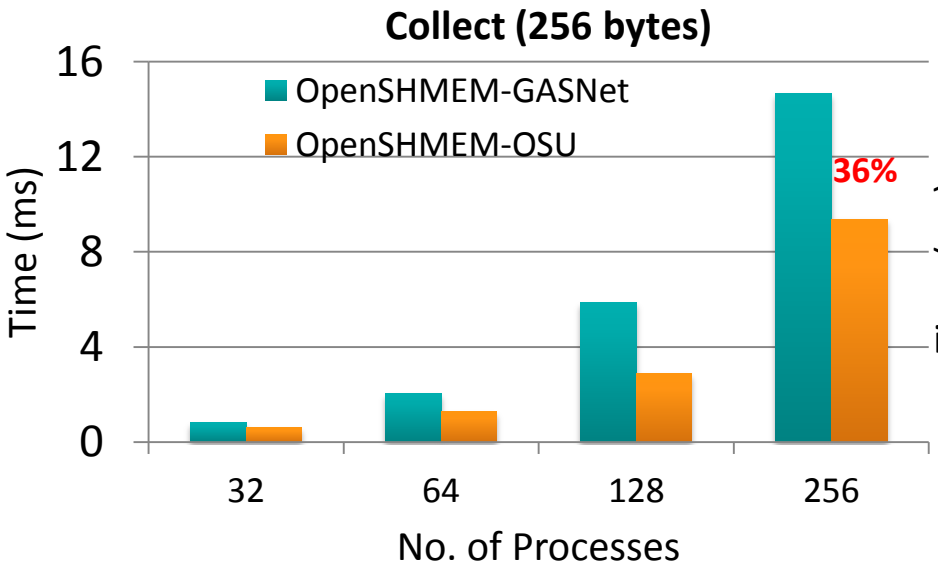
Available in

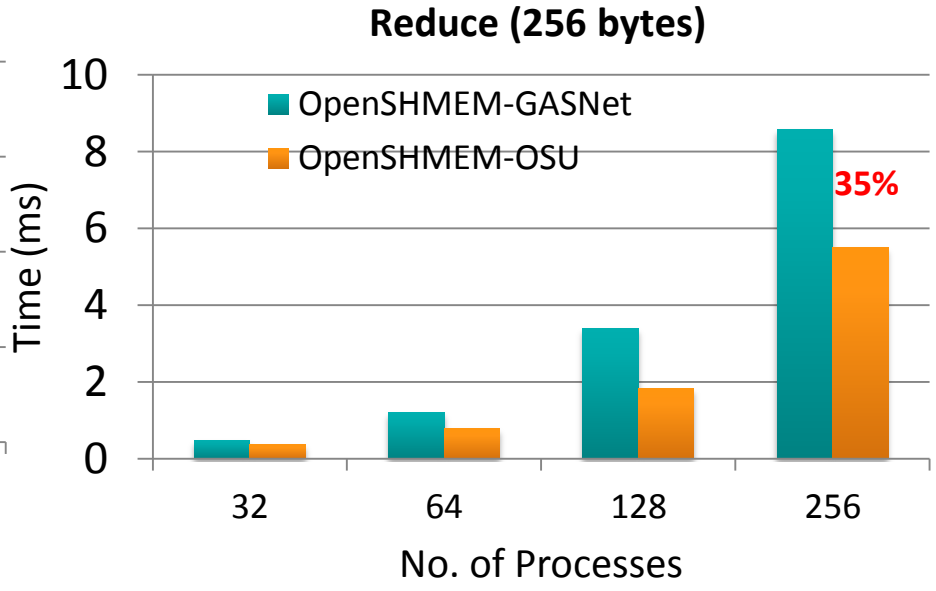
MVAPICH2-X 1.9

Micro-Benchmark Performance (OpenSHMEM)

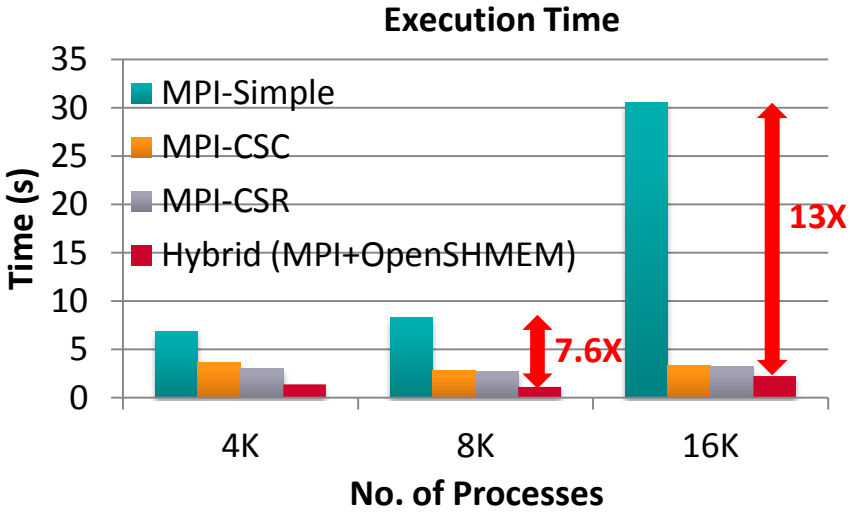


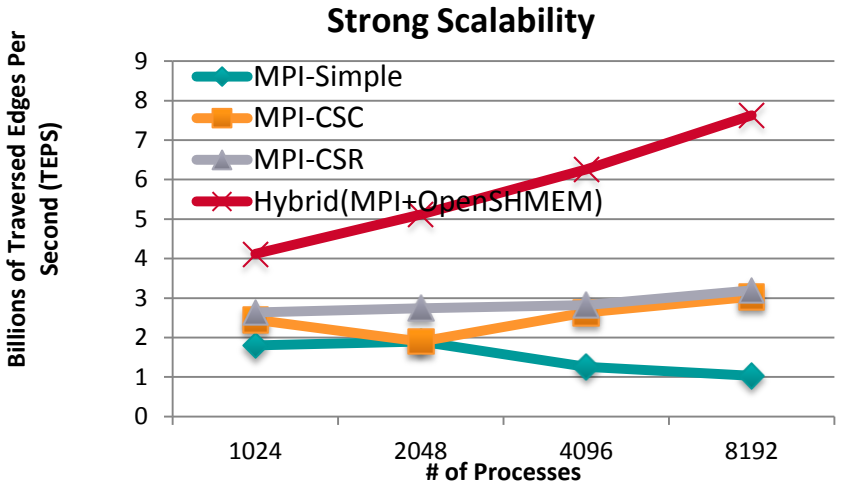


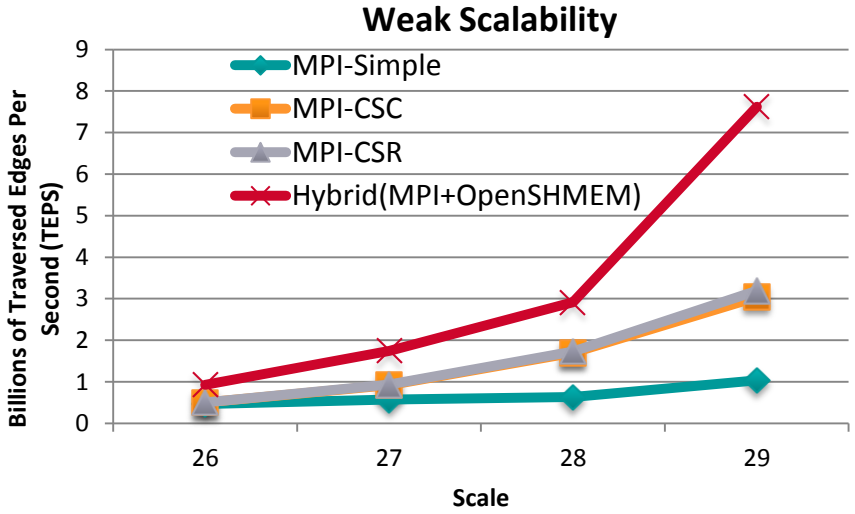




Hybrid MPI+OpenSHMEM Graph500 Design





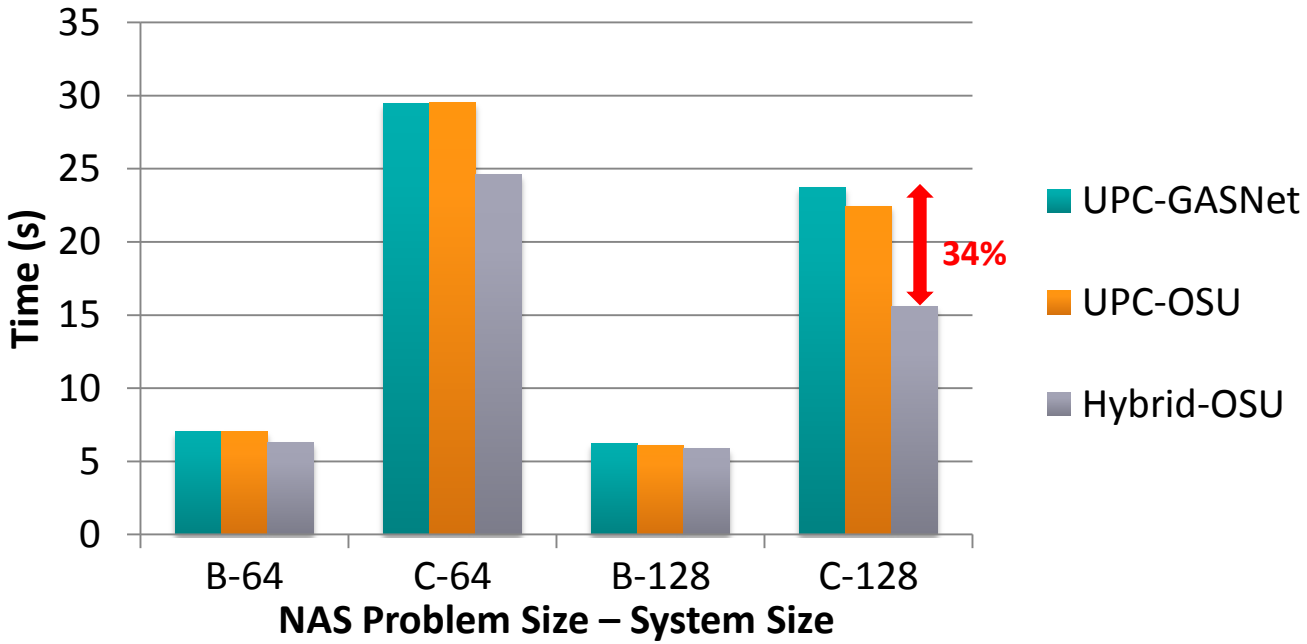


- Performance of Hybrid (MPI+OpenSHMEM) Graph500 Design
  - 8,192 processes
    - **2.4X** improvement over MPI-CSR
    - **7.6X** improvement over MPI-Simple
  - 16,384 processes
    - **1.5X** improvement over MPI-CSR
    - **13X** improvement over MPI-Simple

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC'13), June 2013 (Monday, Hall 5, 5:40 – 6:00 PM)

J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012

Hybrid MPI+UPC NAS-FT



- Modified NAS FT UPC all-to-all pattern using MPI_Alltoall
- Truly hybrid program
- For FT (Class C, 128 processes)
  - **34%** improvement over UPC-GASNet
  - **30%** improvement over UPC-OSU

Hybrid MPI + UPC Support Available in MVAPICH2-X 1.9

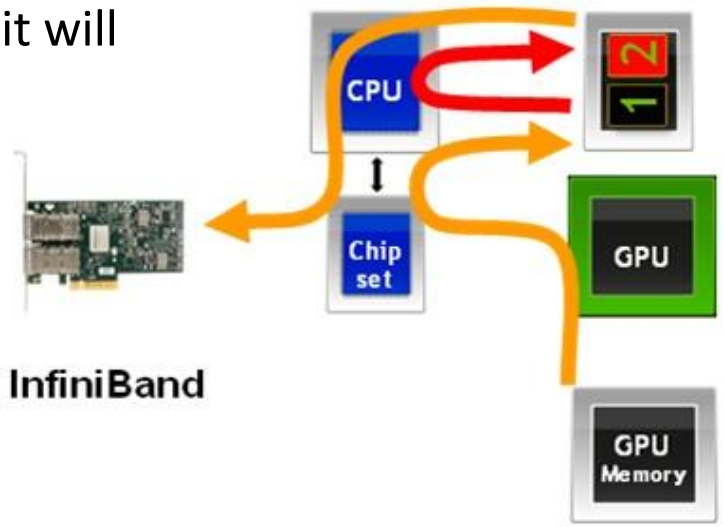
J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, Fourth Conference on Partitioned Global Address Space Programming Model (PGAS '10), October 2010

HPC System Challenges and Case Studies

- Message Passing Interface (MPI)
- Partitioned Global Address Space (PGAS) models
- **GPU Computing**
- MIC Computing

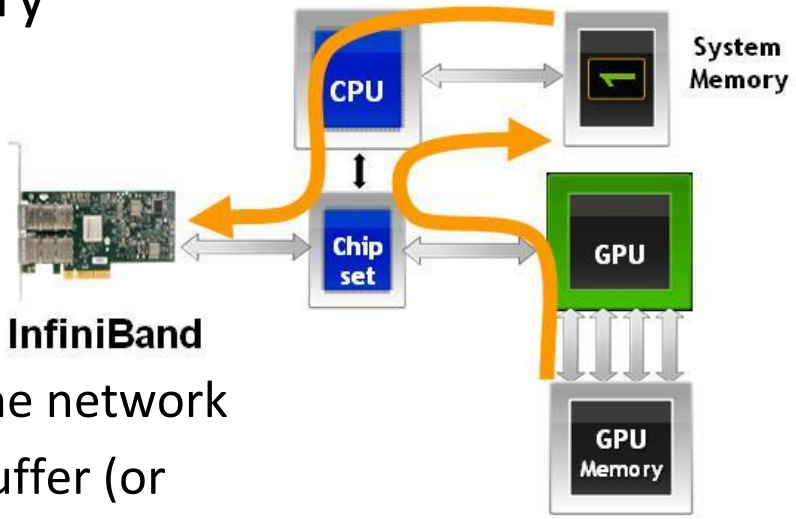
InfiniBand + GPU systems

- Many systems today want to use systems that have both GPUs and high-speed networks such as InfiniBand
- Problem: Lack of a common memory registration mechanism
  - Each device has to pin the host memory it will use
  - Many operating systems do not allow multiple devices to register the same memory pages
- Previous solution:
  - Use different buffer for each device and copy data



GPU-Direct

- Collaboration between Mellanox and NVIDIA to converge on one memory registration technique
- Both devices register a common host buffer
  - GPU copies data to this buffer, and the network adapter can directly read from this buffer (or vice-versa)
- *Note that GPU-Direct does not allow you to bypass host memory*



Sample Code - Without MPI integration

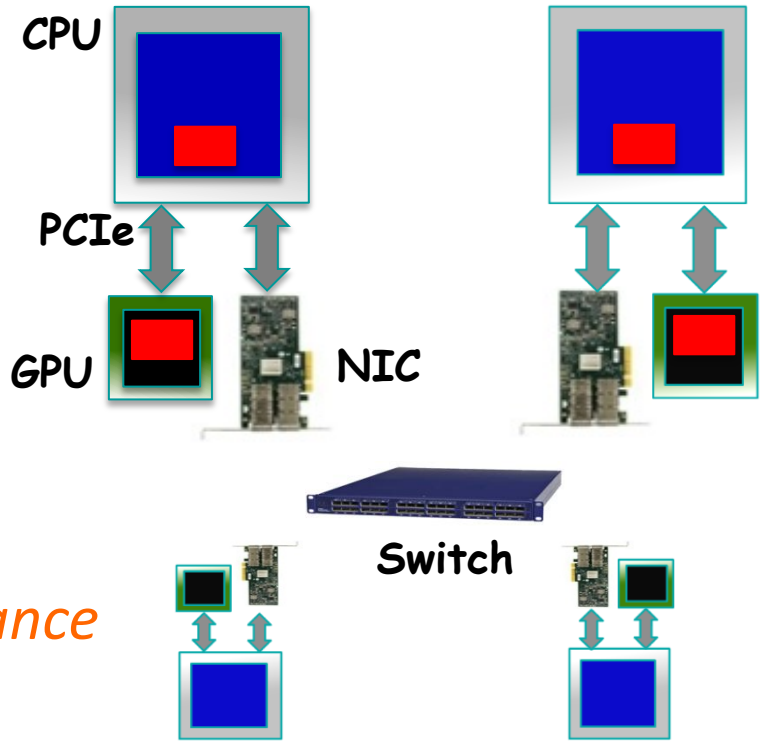
- Naïve implementation with standard MPI and CUDA

At Sender:

```
cudaMemcpy(sbuf, sdev, . . .);
MPI_Send(sbuf, size, . . .);
```

At Receiver:

```
MPI_Recv(rbuf, size, . . .);
cudaMemcpy(rdev, rbuf, . . .);
```



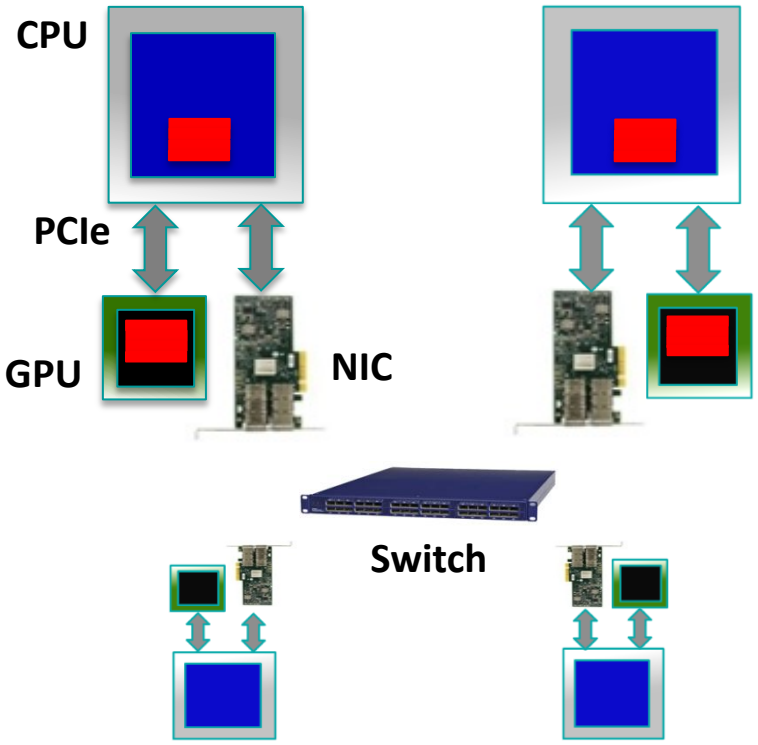
- *High Productivity and Poor Performance*

Sample Code – User Optimized Code

- Pipelining at user level with non-blocking MPI and CUDA interfaces
- Code at Sender side (and repeated at Receiver side)

At Sender:

```
for (j = 0; j < pipeline_len; j++)
    cudaMemcpyAsync(sbuf + j * blk, sdev + j *
      blksz,. . .);
for (j = 0; j < pipeline_len; j++) {
       while (result != cudaSucess) {
            result = cudaStreamQuery(…);
             if(j > 0) MPI_Test(…);
       }
       MPI_Isend(sbuf + j * block_sz, blksz . . .);
 }
MPI_Waitall();
```



- User-level copying may not match with internal MPI design
- *High Performance and Poor Productivity*

Can this be done within MPI Library?

- Support GPU to GPU communication through standard MPI interfaces
  - e.g. enable MPI_Send, MPI_Recv from/to GPU memory
- Provide high performance without exposing low level details to the programmer
  - Pipelined data transfer which *automatically* provides optimizations inside MPI library without user tuning
- **A new Design incorporated in MVAPICH2 to support this functionality**

Sample Code – MVAPICH2-GPU

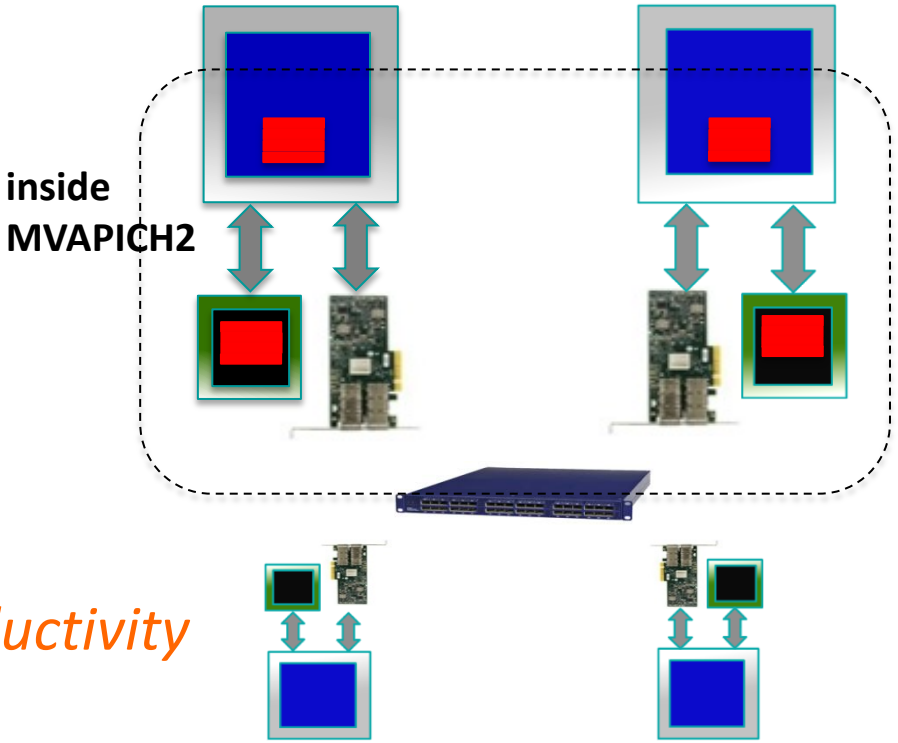
- MVAPICH2-GPU: standard MPI interfaces used
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_device, size, …);
```

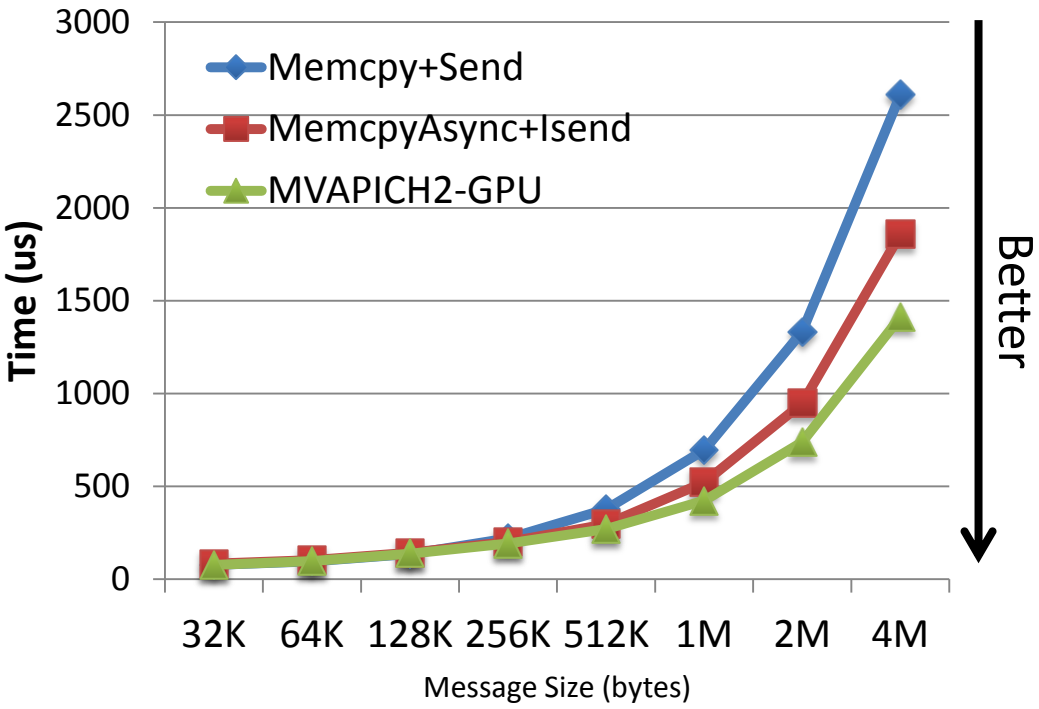
At Receiver:

```
MPI_Recv(r_device, size, …);
```



- *High Performance and High Productivity*

MPI-Level Two-sided Communication



- 45% improvement compared with a naïve user-level implementation (Memcpy+Send), for 4MB messages
- 24% improvement compared with an advanced user-level implementation (MemcpyAsync+Isend), for 4MB messages

H. Wang, S. Potluri, M. Luo, A. Singh, S. Sur and D. K. Panda, MVAPICH2-GPU: Optimized GPU to GPU Communication for InfiniBand Clusters, ISC '11

Other MPI Operations and Optimizations for GPU Buffers

- Overlap optimizations for
  - One-sided Communication
  - Collectives
  - Communication with Datatypes
- Optimized Designs for multi-GPUs per node
  - Use CUDA IPC (in CUDA 4.1), to avoid copy through host memory

• H. Wang, S. Potluri, M. Luo, A. Singh, X. Ouyang, S. Sur and D. K. Panda, Optimized Non-contiguous MPI Datatype Communication for GPU Clusters: Design, Implementation and Evaluation with MVAPICH2, IEEE Cluster '11, Sept. 2011

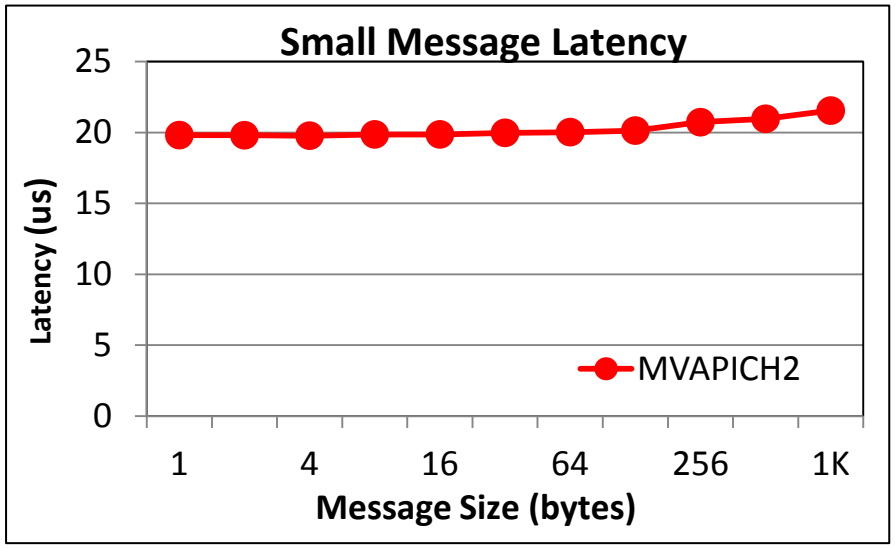
• A. Singh, S. Potluri, H. Wang, K. Kandalla, S. Sur and D. K. Panda, MPI Alltoall Personalized Exchange on GPGPU Clusters: Design Alternatives and Benefits, Workshop on Parallel Programming on Accelerator Clusters (PPAC '11), held in conjunction with Cluster '11, Sept. 2011

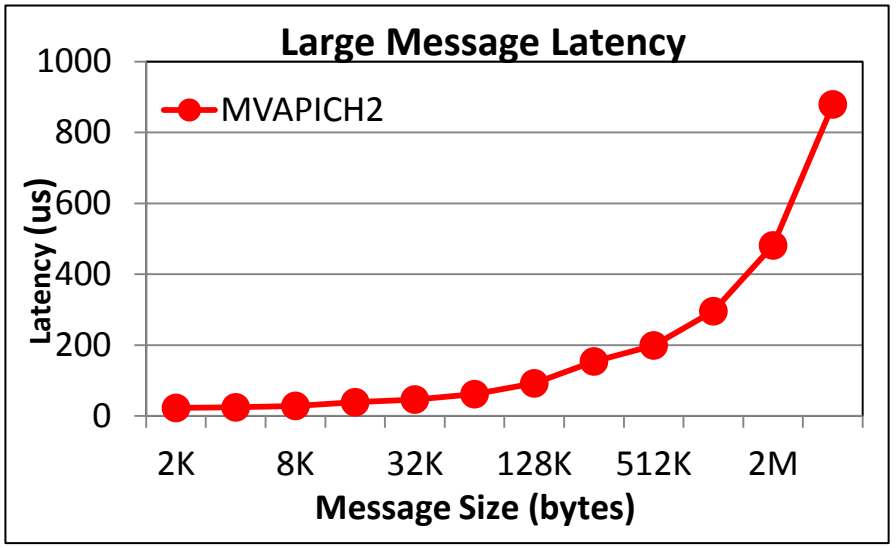
• S. Potluri et al. Optimizing MPI Communication on Multi-GPU Systems using CUDA Inter-Process Communication, Workshop on Accelerators and Hybrid Exascale Systems(ASHES), to be held in conjunction with IPDPS 2012, May 2012

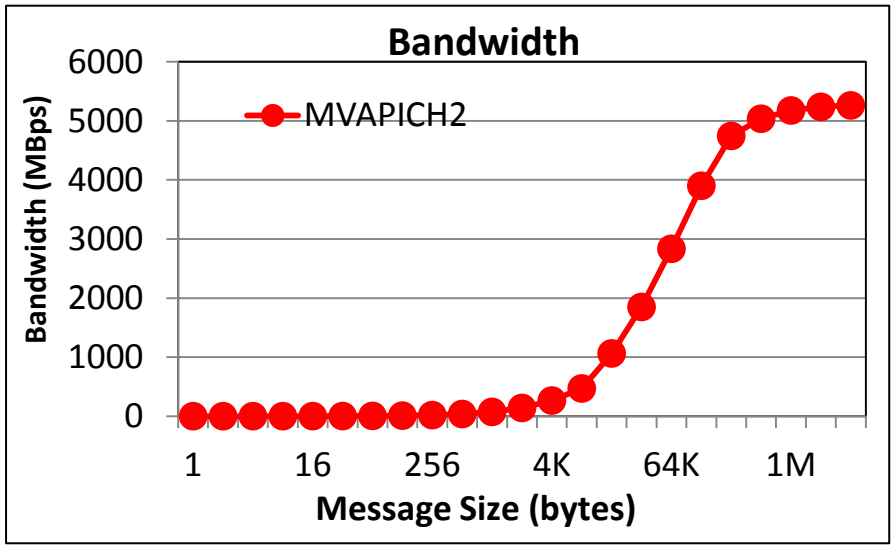
MVAPICH2 1.8 and 1.9 Release

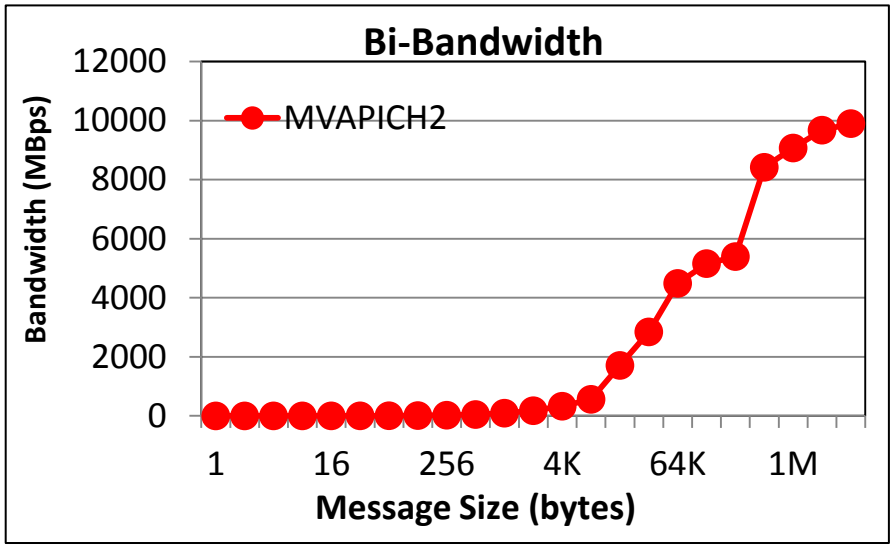
- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers

MVAPICH2 (Device-Device, Inter-node)



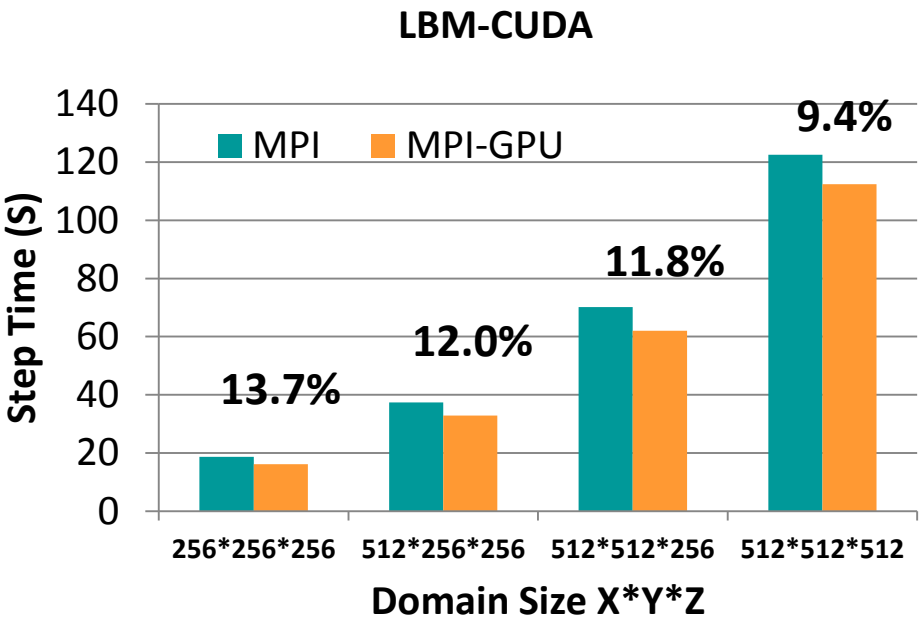


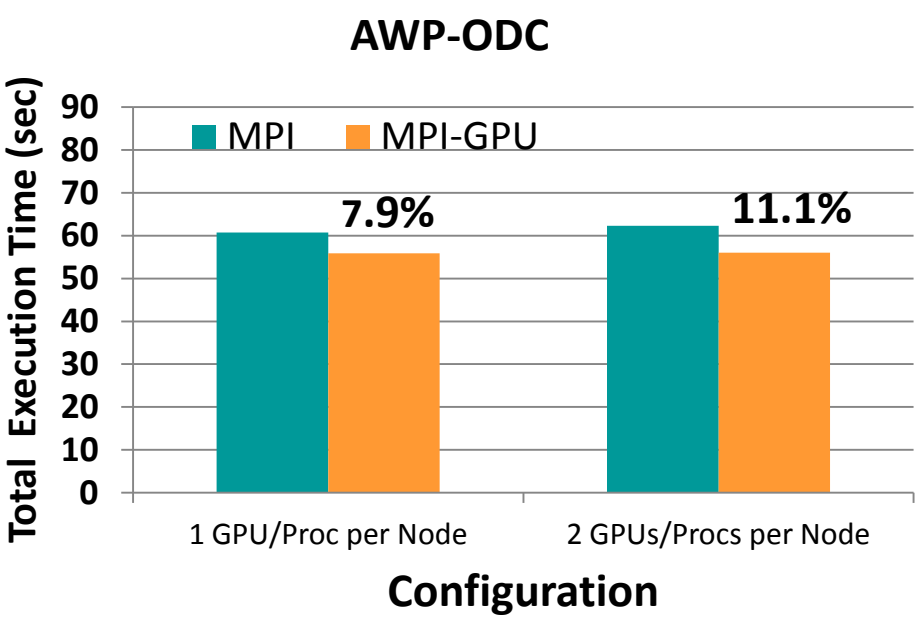




Intel Sandy Bridge (E5-2670) node with 16 cores; NVIDIA Telsa K20c GPU; Mellanox ConnectX-3 FDR HCA
MVAPICH2-1.9; CUDA 5.0, OFED 1.5.4.1

Applications-Level Evaluation

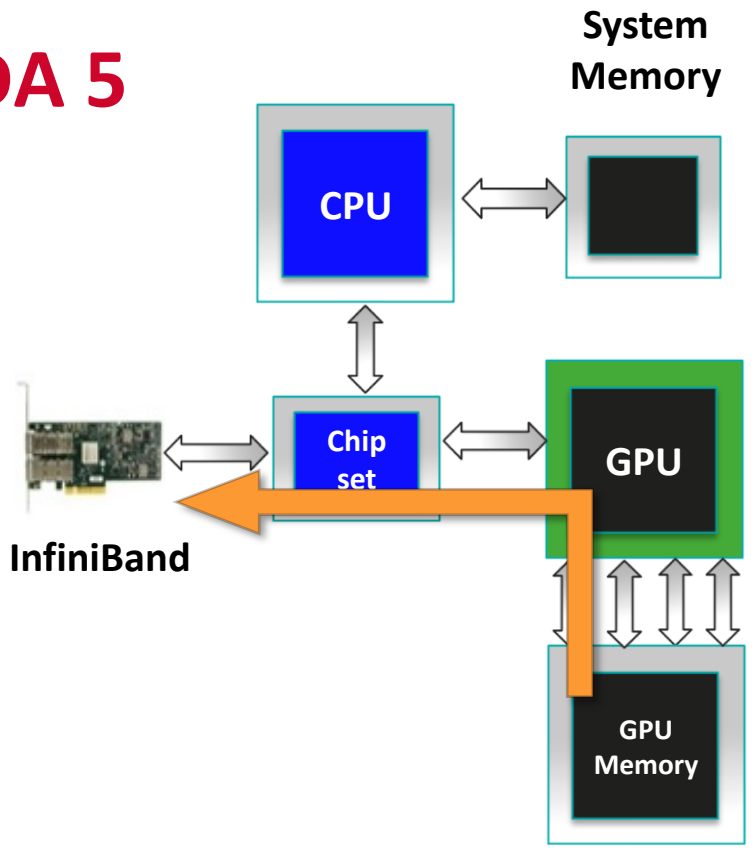




- LBM-CUDA (Courtesy: Carlos Rosale, TACC)
  - Lattice Boltzmann Method for multiphase flows with large density ratios
  - one process/GPU per node, 16 nodes
- AWP-ODC (Courtesy: Yifeng Cui, SDSC)
  - a seismic modeling code, Gordon Bell Prize finalist at SC 2010
  - 128x256x512 data grid per process, 8 nodes
- Oakley cluster at OSC: two hex-core Intel Westmere processors, two NVIDIA Tesla M2070, one Mellanox IB QDR MT26428 adapter and 48 GB of main memory

GPU-Direct RDMA with CUDA 5

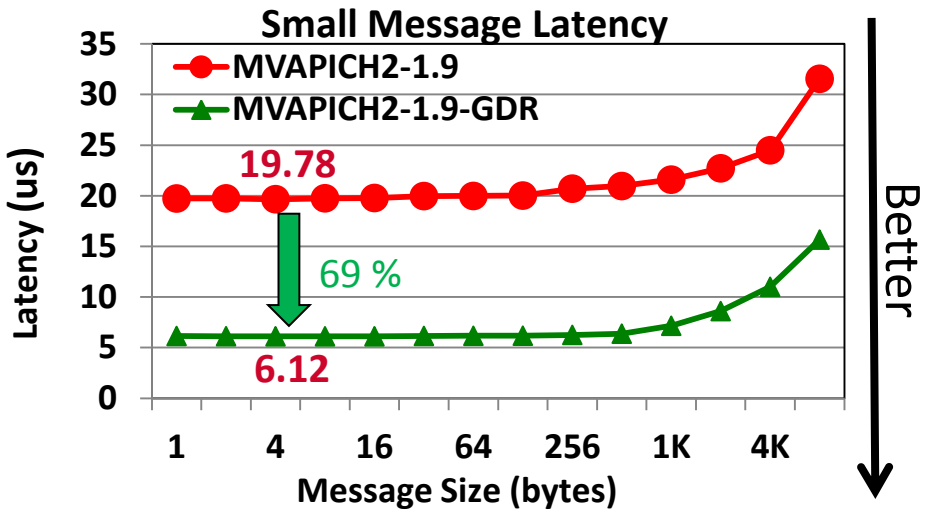
- Fastest possible communication between GPU and other PCI-E devices
- Network adapter can directly read data from GPU device memory
- Avoids copies through the host
- Allows for better asynchronous communication

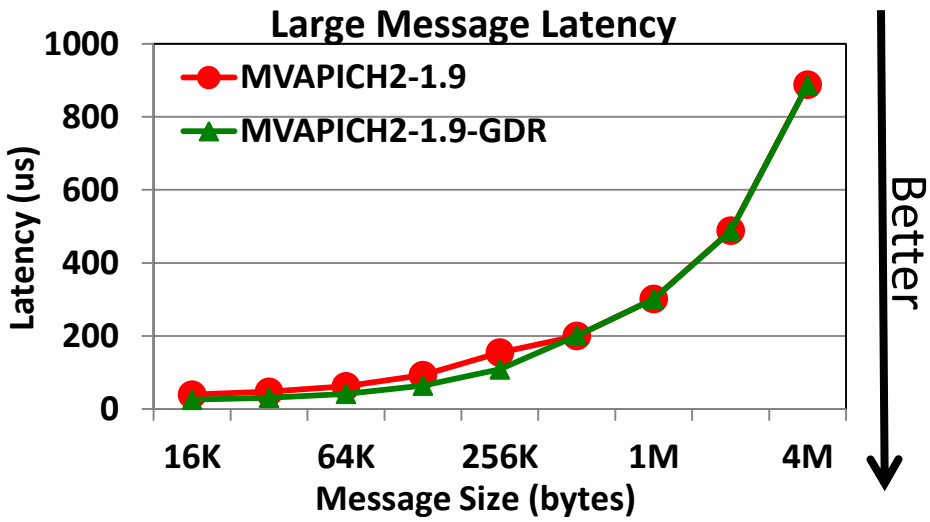


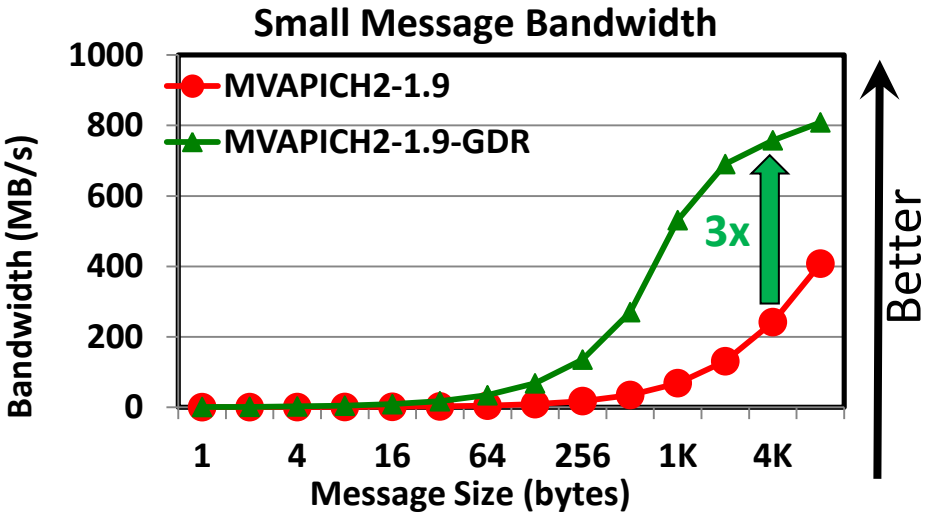
MVAPICH2 with GDR support under development

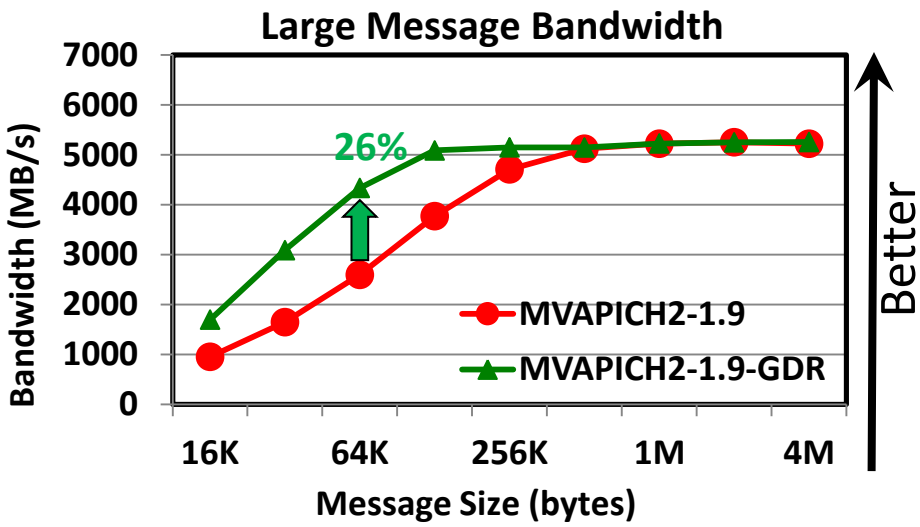
MVAPICH2-GDR Alpha will be demonstrated at ISC'13 Exhibition Floor (Mellanox Technologies, Booth #326)

Preliminary Performance of MVAPICH2 with GPU-Direct-RDMA





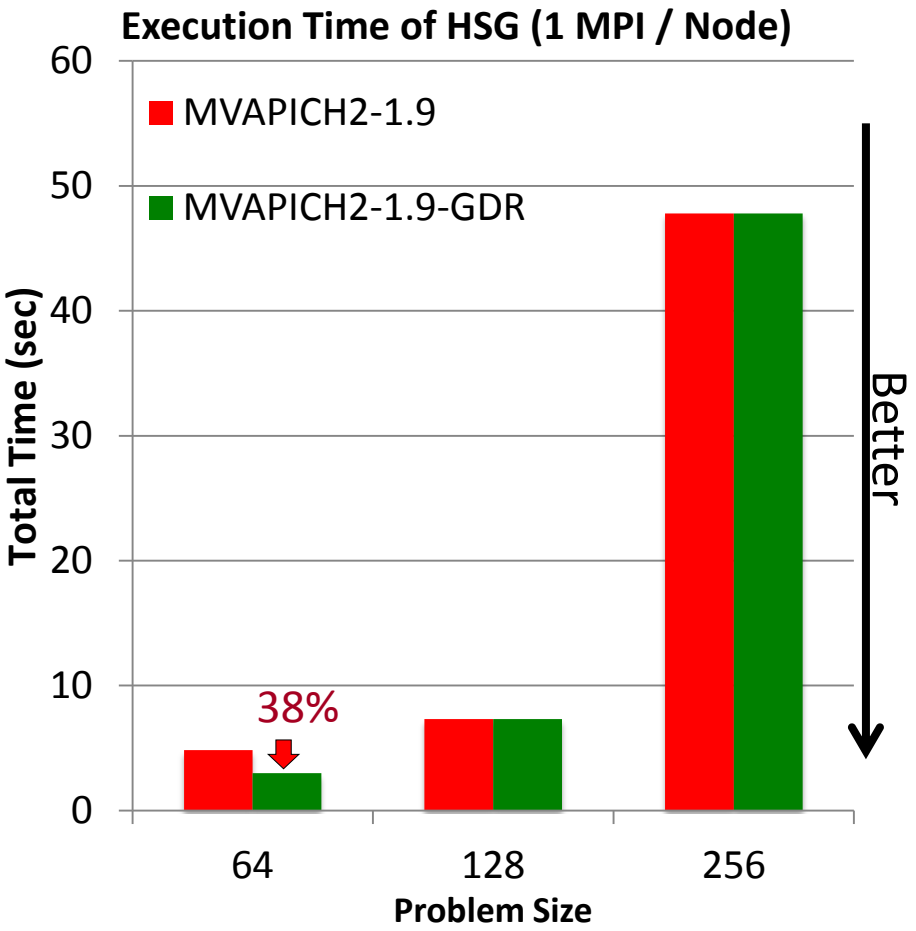


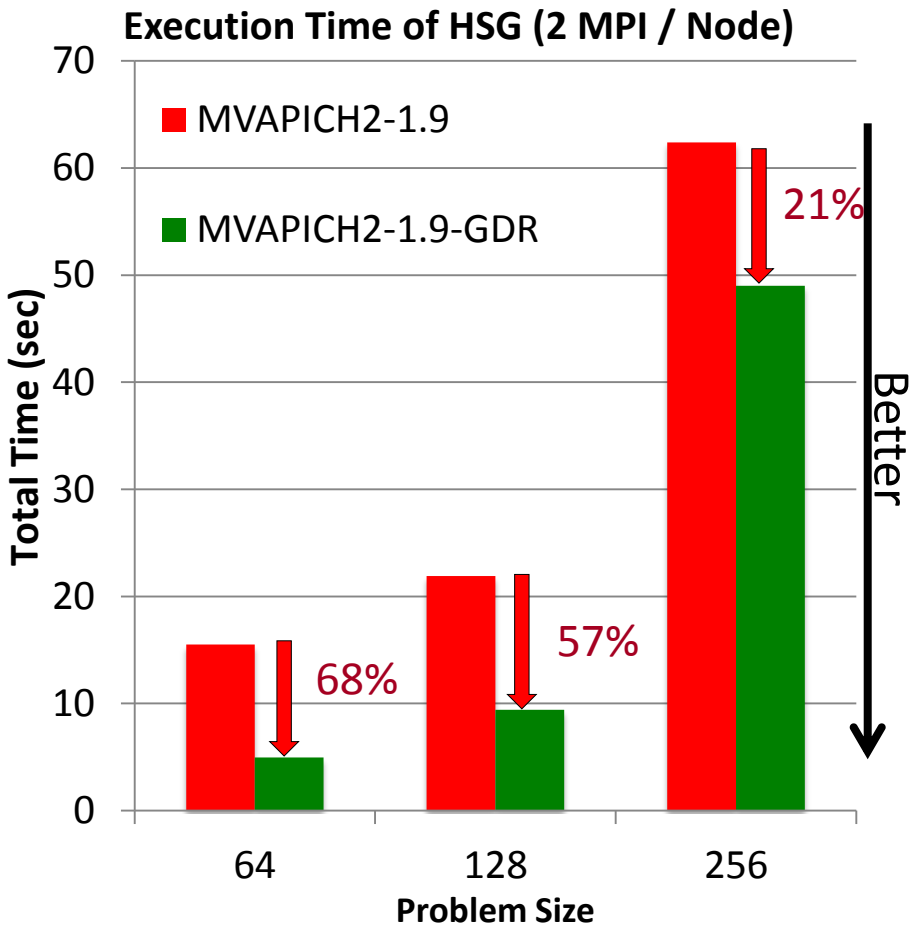


Intel Sandy Bridge (E5-2670) node with 16 cores; NVIDIA Telsa K20c GPU; Mellanox ConnectX-3 FDR HCA
MVAPICH2-1.9; CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

Execution Time of HSG Application

Application run with two GPU Nodes



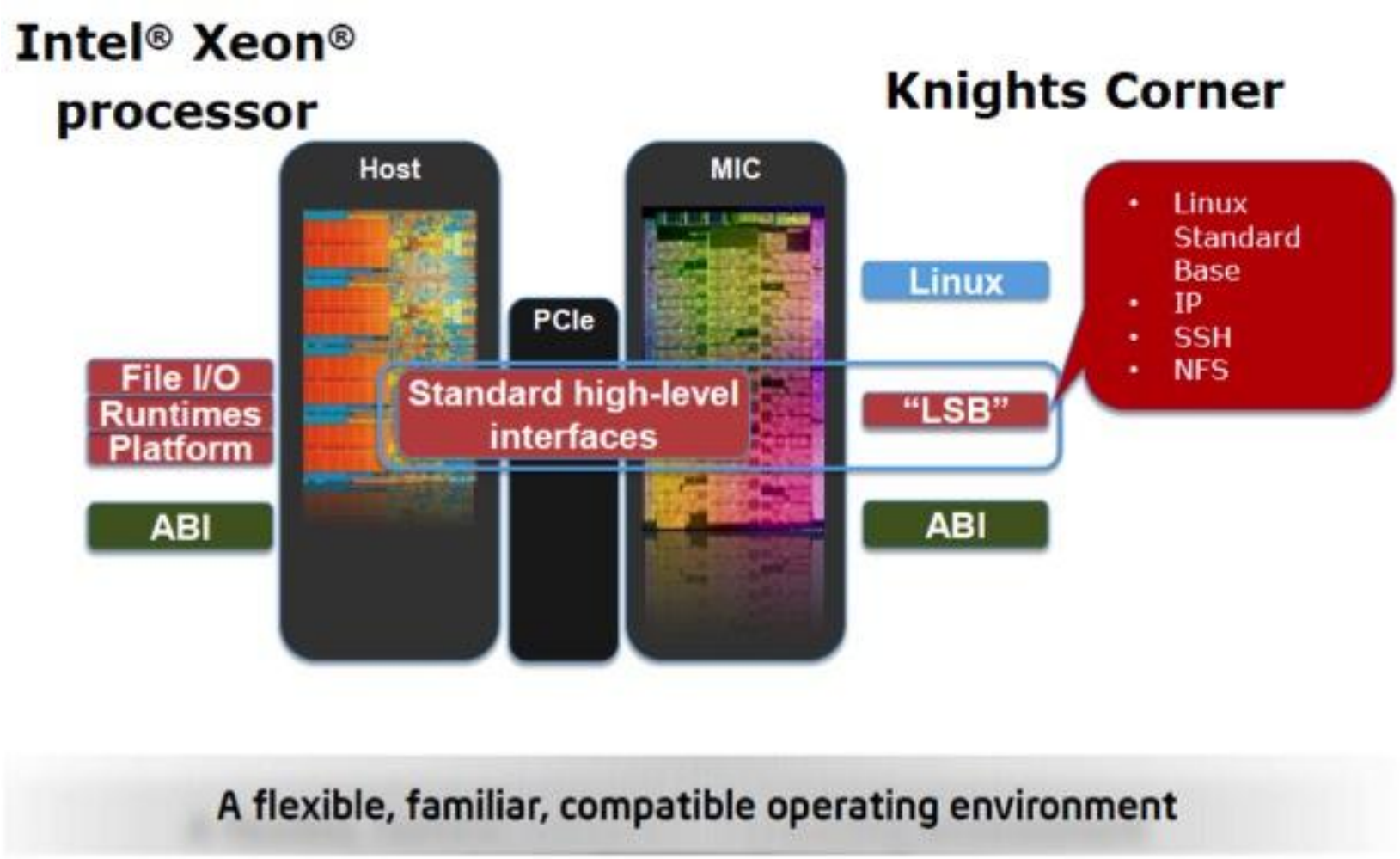


Based on MVAPICH2-1.9
Intel Sandy Bridge (E5-2670) node with 16 cores
NVIDIA Telsa K20c GPU, Mellanox ConnectX-3 FDR HCA
CUDA 5.0, OFED 1.5.4.1 with GPU-Direct-RDMA Patch

HPC System Challenges and Case Studies

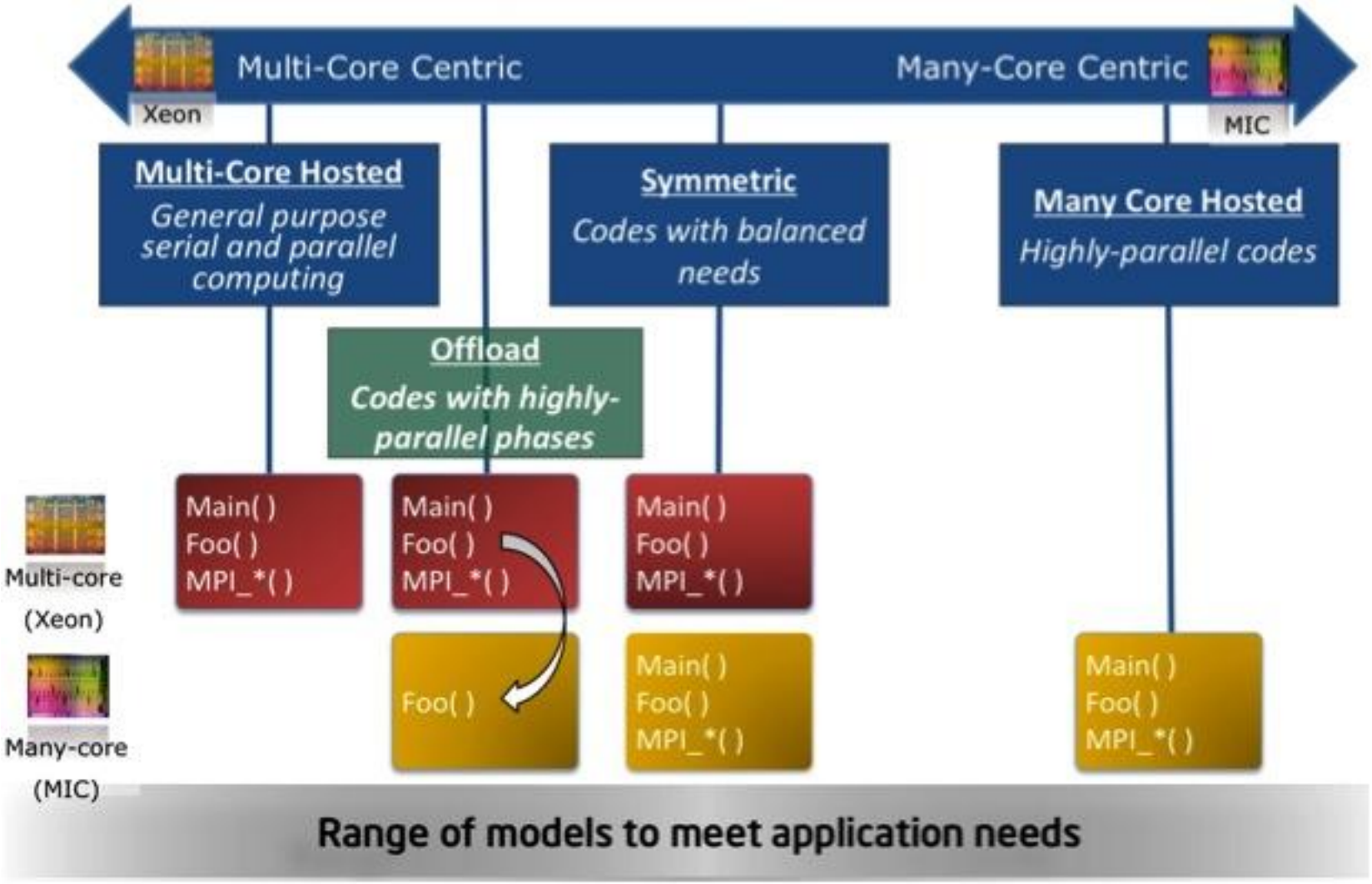
- Message Passing Interface (MPI)
- Partitioned Global Address Space (PGAS) models
- GPU Computing
- **MIC Computing**

InfiniBand + MIC systems



Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

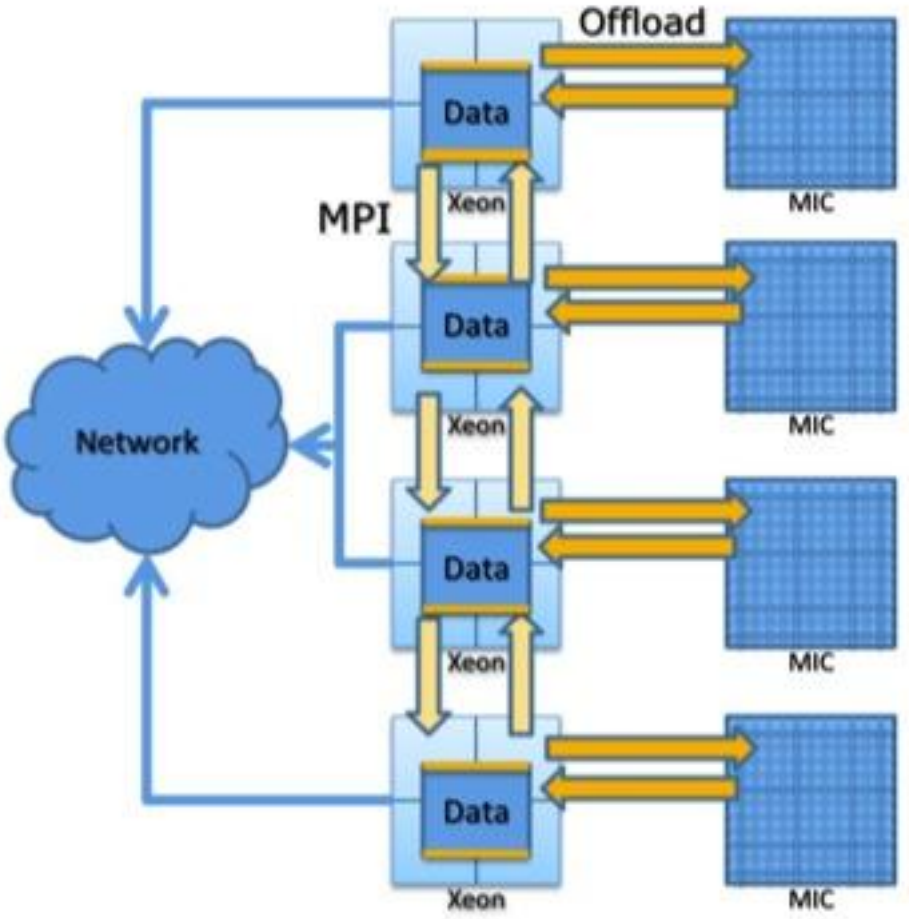
Programming Models for Intel MIC Architecture



Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Offload Model for Intel MIC-based Systems

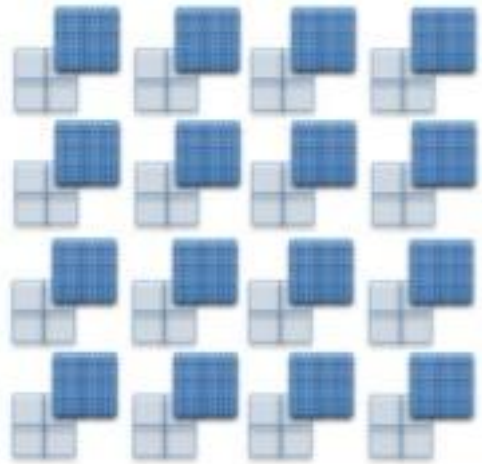
- MPI processes on the host
- Intel MIC used through offload – OpenMP, Intel Cilk Plus, Intel TBB and Pthreads
- MPI communication – Intranode and Internode

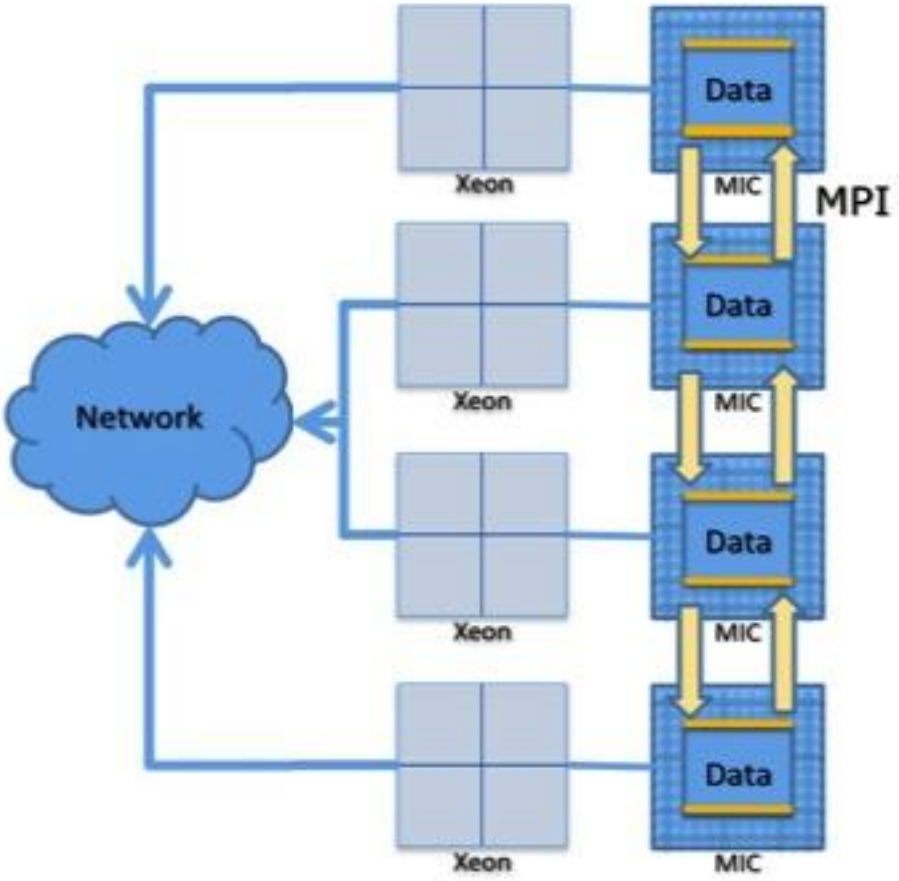


Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Many-core Hosted Model for Intel MIC-based Systems

- MPI processes on the Intel MIC
- MPI communication – IntraMIC, InterMIC (host-bypass/directly over the network)

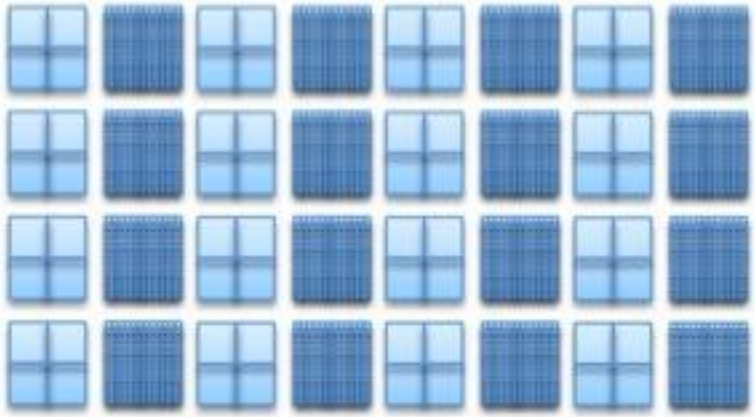


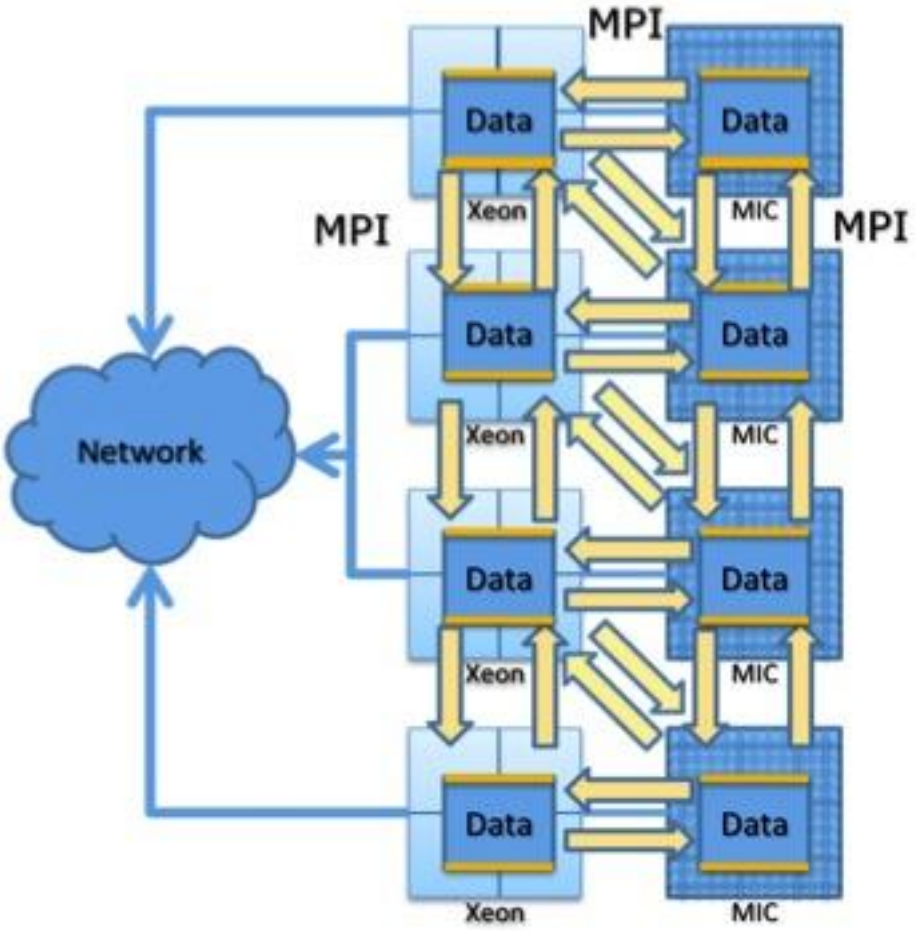


Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

Symmetric Model for Intel MIC-based Systems

- MPI processes on the host and the Intel MIC
- MPI communication – Intranode, Internode, IntraMIC, InterMIC, Host-MIC





Courtesy: Scott McMillan, Intel Corporation, Presentation at TACC-Intel Highly Parallel Computing Symposium` 12 (http://www.tacc.utexas.edu/documents/13601/d9d58515-5c0a-429d-8a3f-85014e9e4dab)

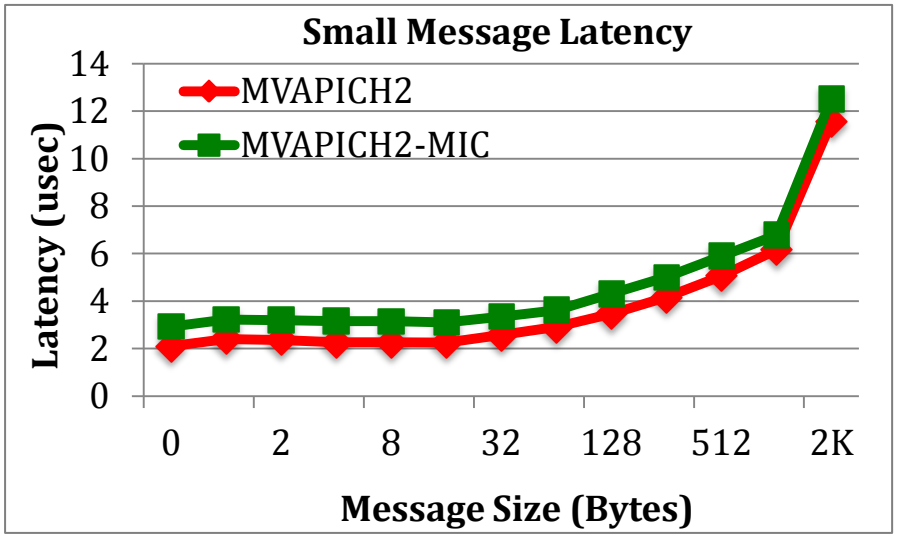
Enhancing MPI for MIC

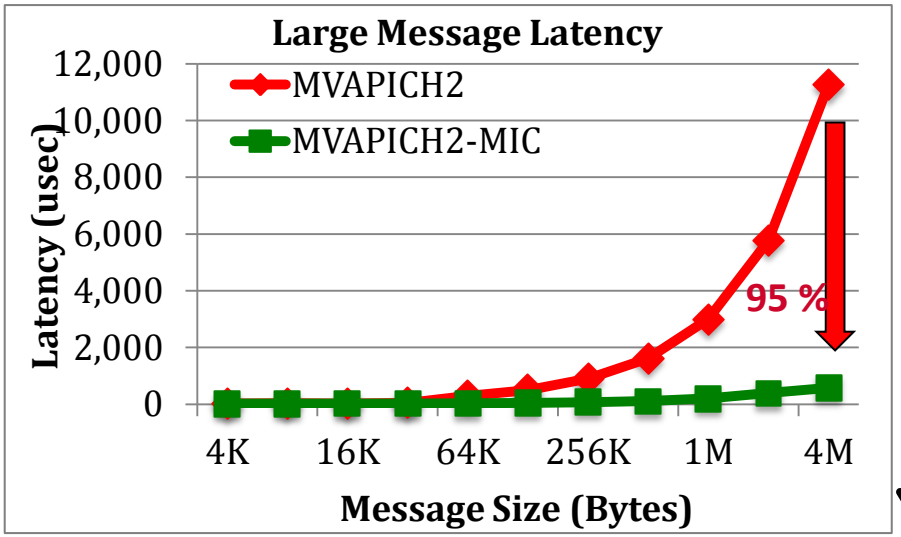
- Intel's Many Integrated Core (MIC) architecture geared for HPC
- Critical part of their solution for exascale computing
- Many low-power x86 processor cores with hardware threads and wider vector units
- Applications and libraries can run with minor or no modifications
- Using MVAPICH2 for Intra-MIC, Host-MIC communication
  - out-of-the box (**MVAPICH2**)
  - Enhanced using lower level API and Tuned (**MVAPICH2-MIC**)
- Partner in the NSF-TACC Stampede Project – first large-scale open deployment of Intel MIC

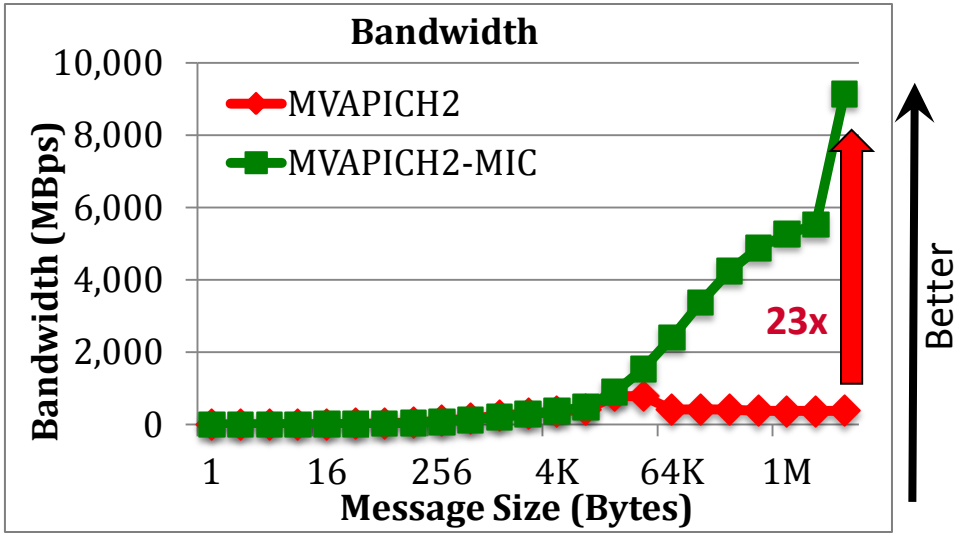
S. Potluri, K. Tomko, D. Bureddy and D. K. Panda – Intra-MIC MPI Communication using MVPAICH2: Early Experience, TACC-Intel Highly-Parallel Computing Symposium, April 2012 – Best Student Paper Award

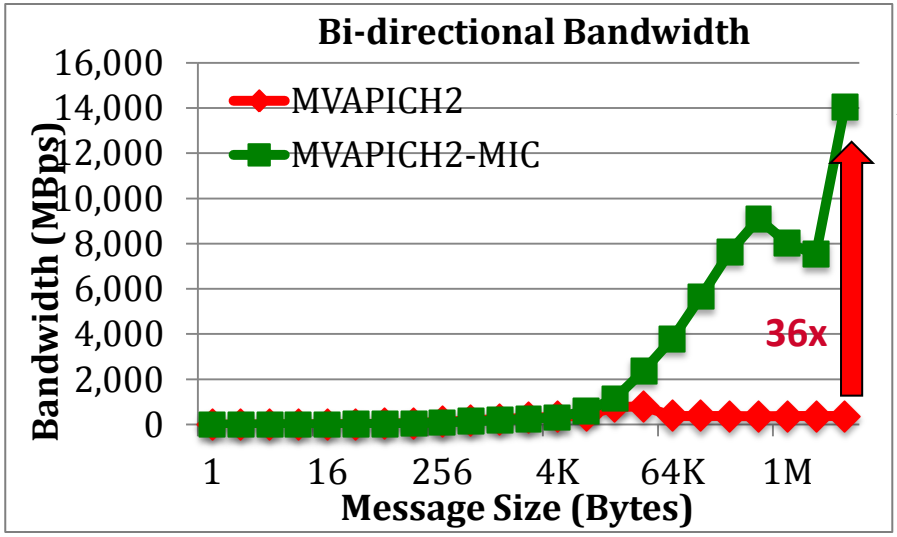
S. Potluri, A. Venkatesh, D. Bureddy, K. Kandalla and D. K. Panda, Efficient Intra-node Communication on Intel-MIC Clusters, Int'l Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2013), May 2013

Intra-MIC Communication





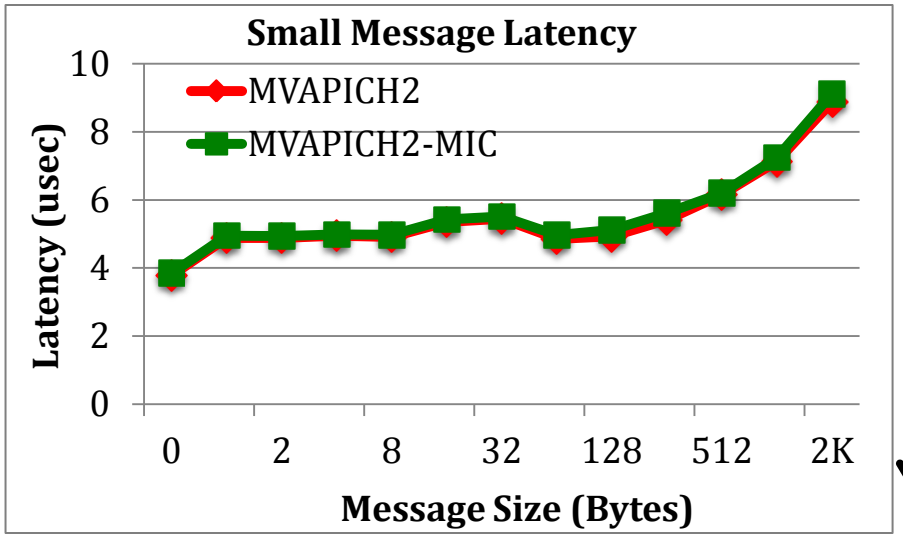


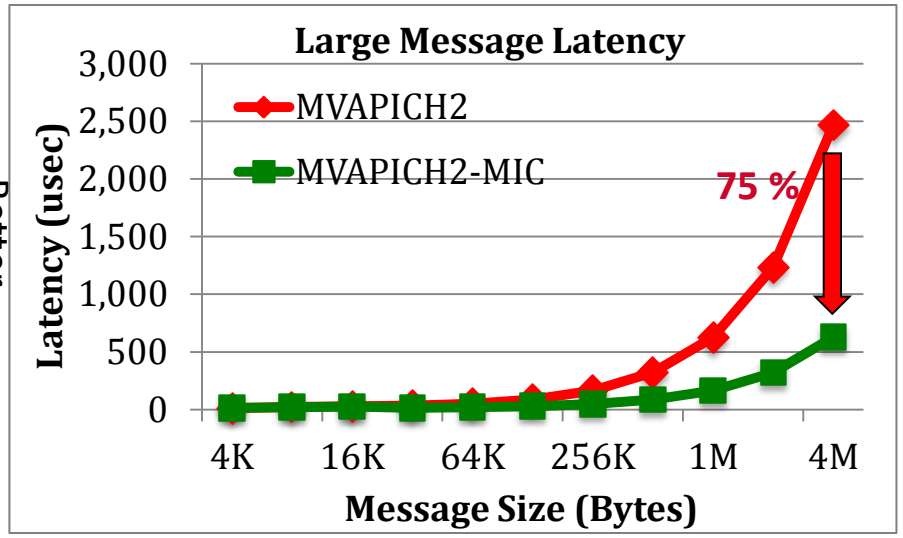


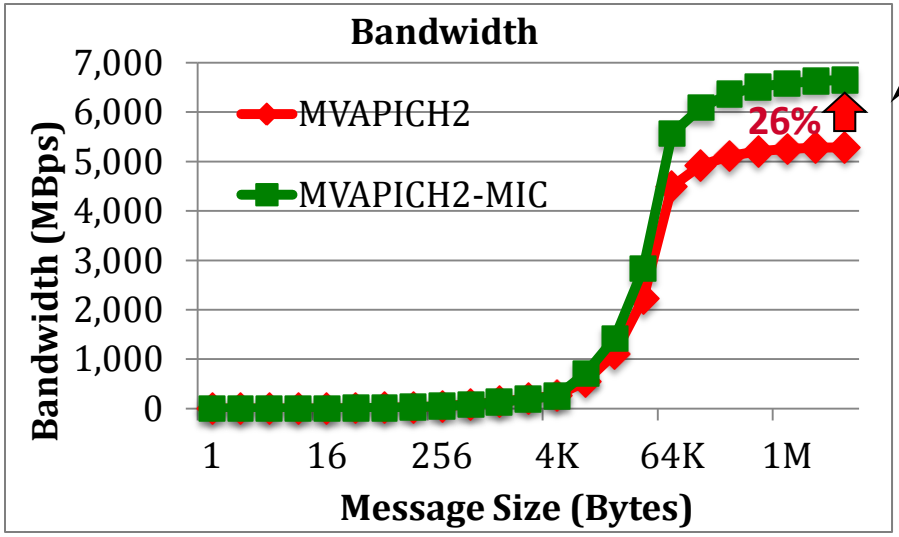
MVAPICH2-MIC (based on MVAPICH2 1.9)
Intel Sandy Bridge (E5-2670) node with 16 cores, SE10P (B0-KNC),
MPSS 2.1.6720-13, composer_xe_2013.4.183, and IB FDR MT4099 HCA

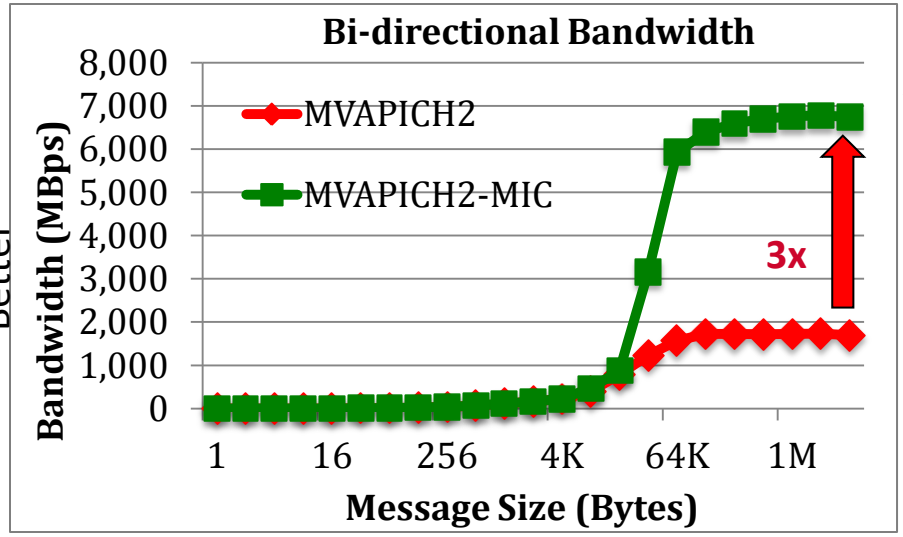
Host-to-MIC Communication

Similar improvements for MIC-to-Host







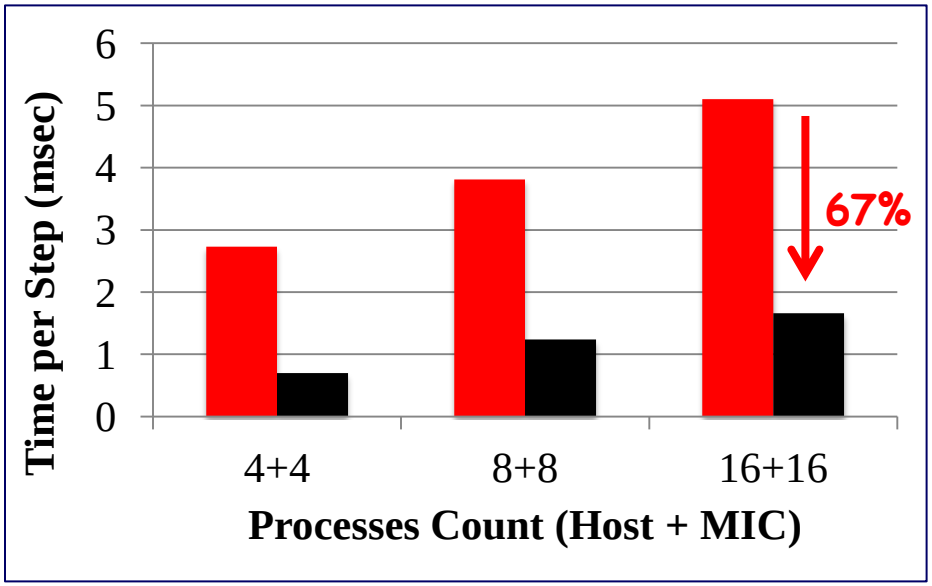


MVAPICH2-MIC (based on MVAPICH2 1.9)
Intel Sandy Bridge (E5-2670) node with 16 cores, SE10P (B0-KNC),
MPSS 2.1.6720-13, composer_xe_2013.4.183, and IB FDR MT4099 HCA

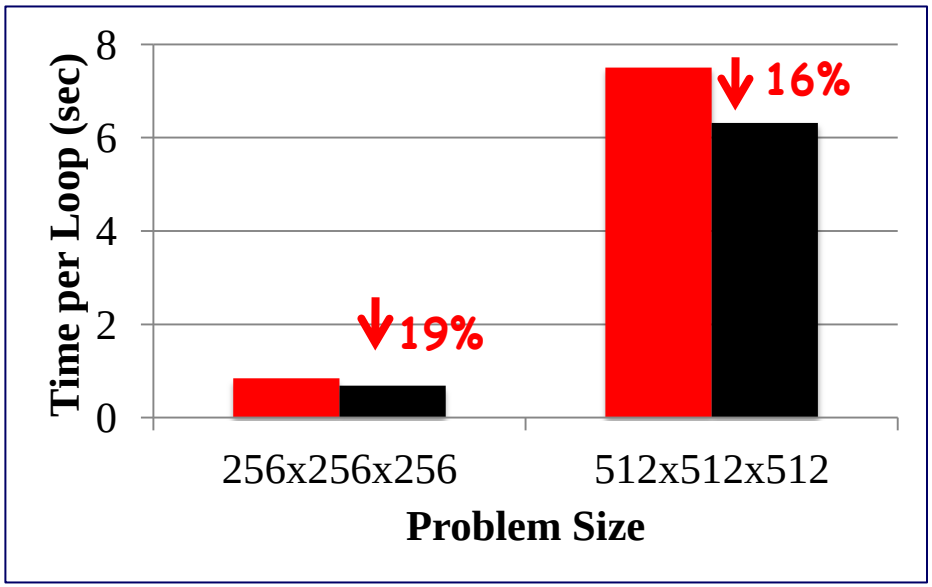
Communication Kernel and Application Level Evaluation

MVAPICH2 MVAPICH2-MIC

3D Stencil Communication Kernel



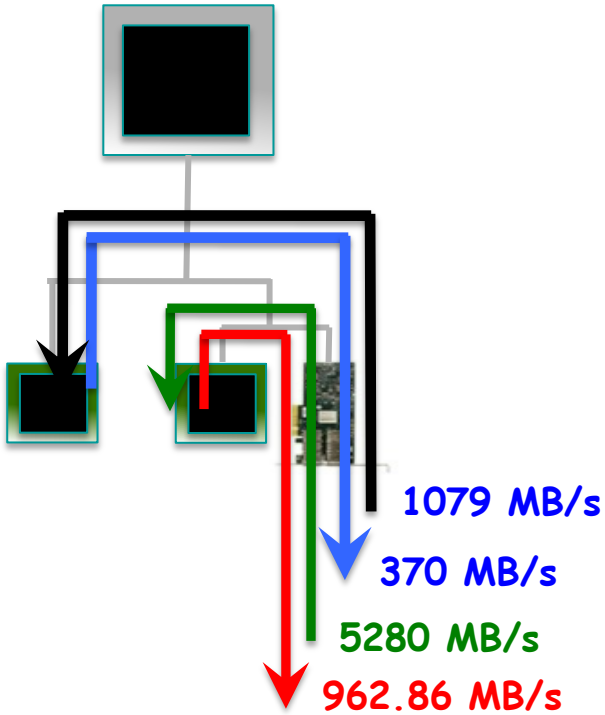
P3DFFT Library (MPI + OpenMP)

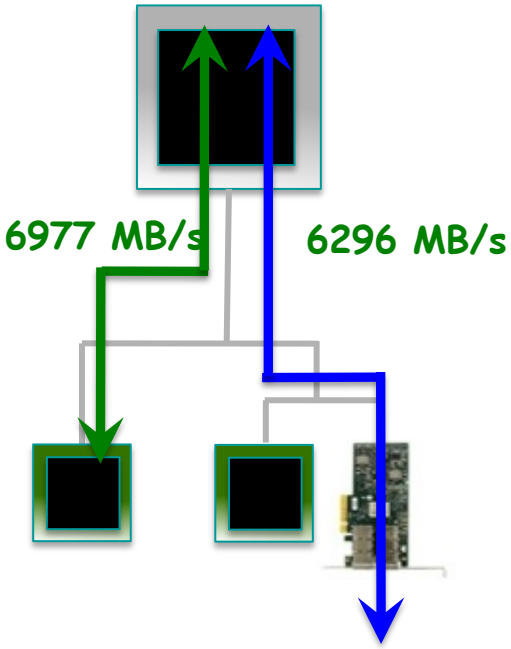


- Near-neighbor communication – upto 6 neighbors – 64KB message size - 67% improvement in time per step.
- P3DFFT, a popular library for 3D Fast Fourier Transforms – (MPI + OpenMP) – 2 procs on host, 8 procs on MIC – 8 threads/process – up to 19% improvement.

MVAPICH2-MIC (based on MVAPICH2 1.9)
Intel Sandy Bridge (E5-2680) node with 16 cores, SE10P (B0-KNC),
MPSS 4346-16 (Gold), composer_xe_2013.2.146, and IB FDR MT4099 HCA

Inter-node Communication: Performance Limitation with Sandybridge and Intel MIC

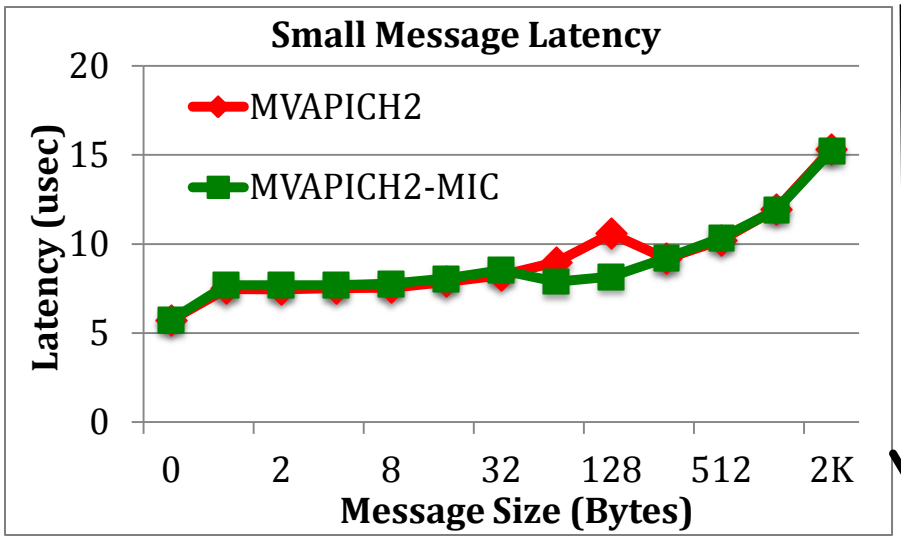


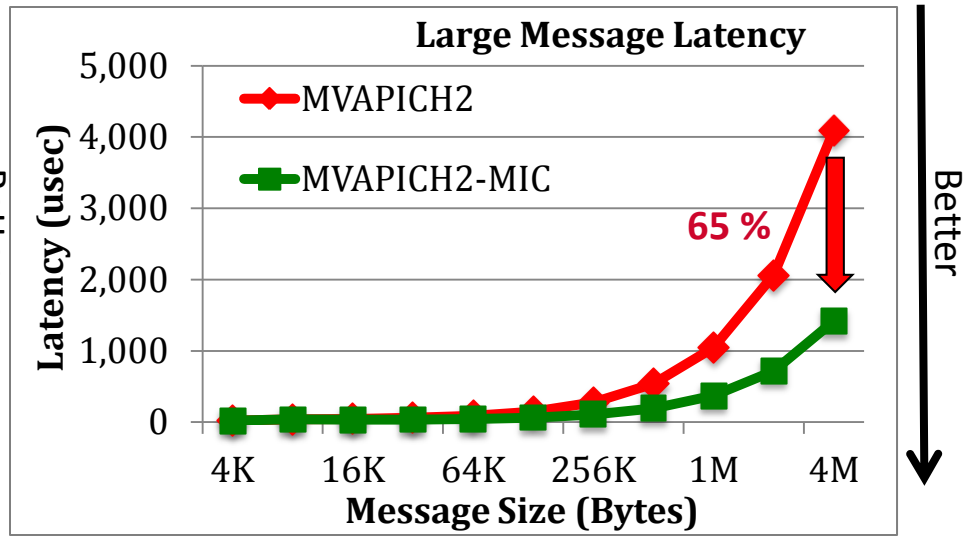


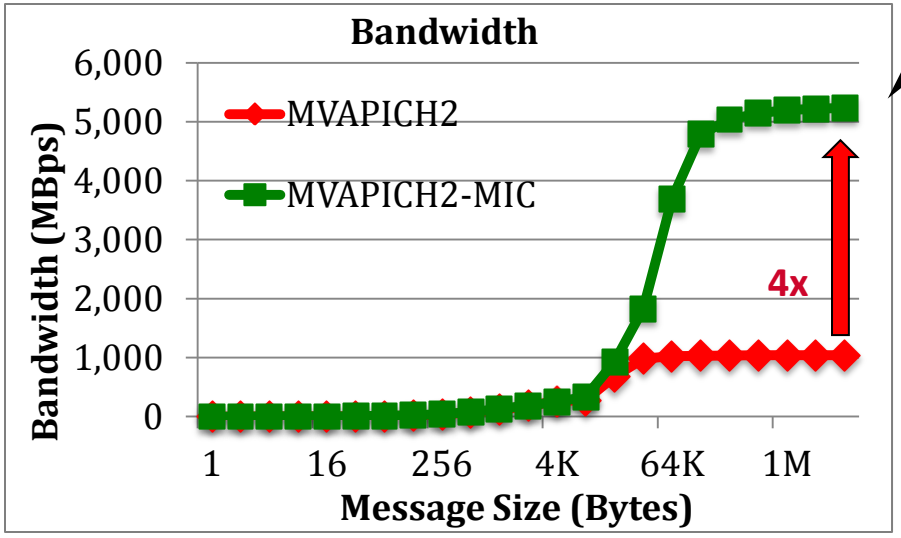
- MIC-to-Remote (Host/MIC) : Intra-IOH
- Remote (Host/MIC)-to-MIC : Intra-IOH
- MIC-to-Remote (Host/MIC) : Inter-IOH
- Remote (Host/MIC)-to-MIC : Inter-IOH
- MIC-to-Host/Host-to-MIC
- Host-to-Remote Host

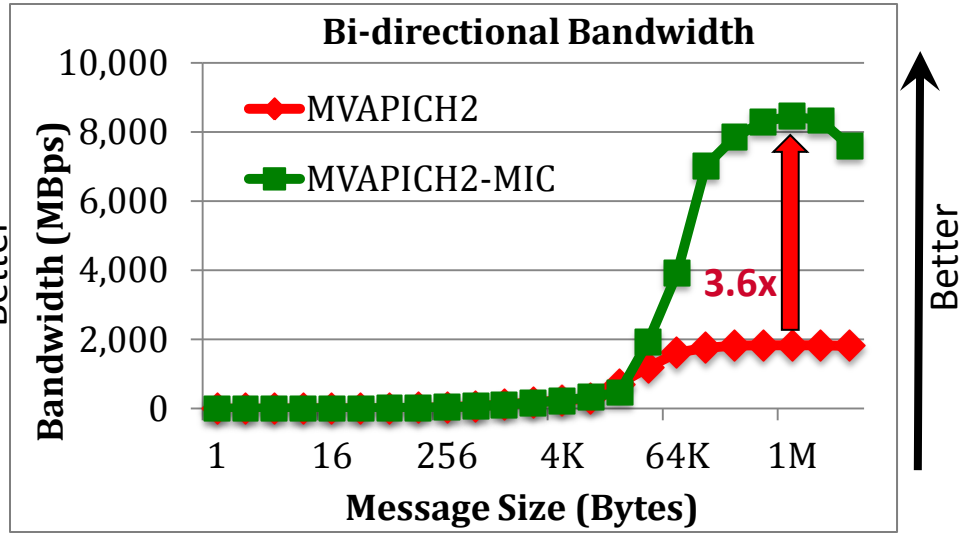
- Performance of IB reads from MIC is limited
- Proxy process on host to relay communication

Inter-node MIC-MIC Communication



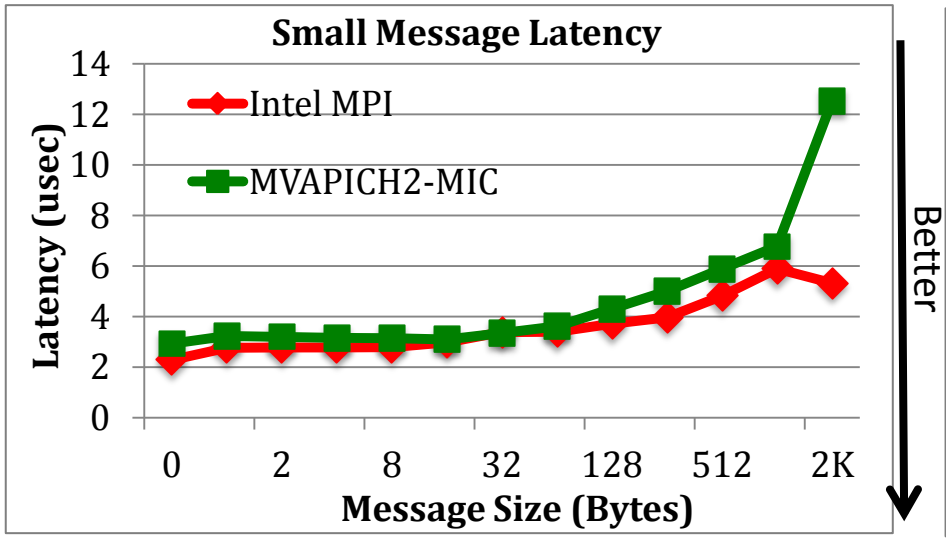


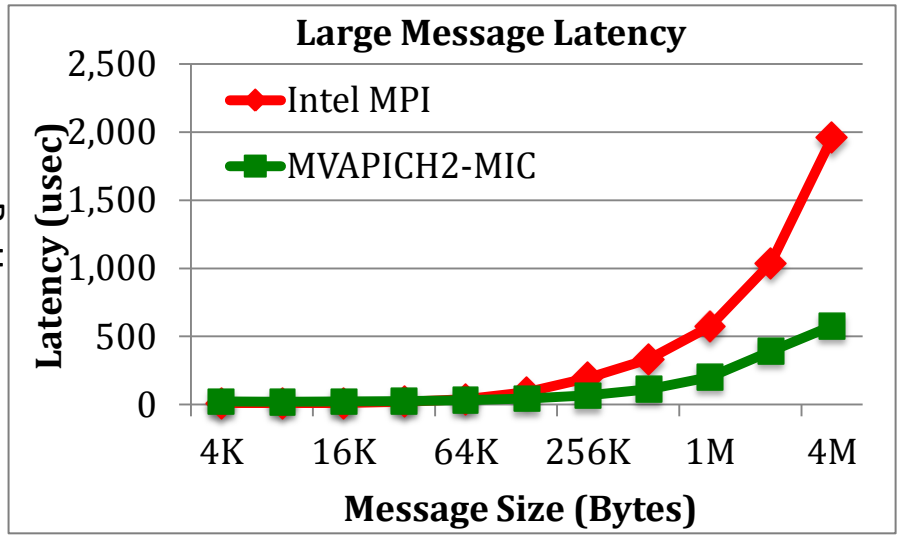


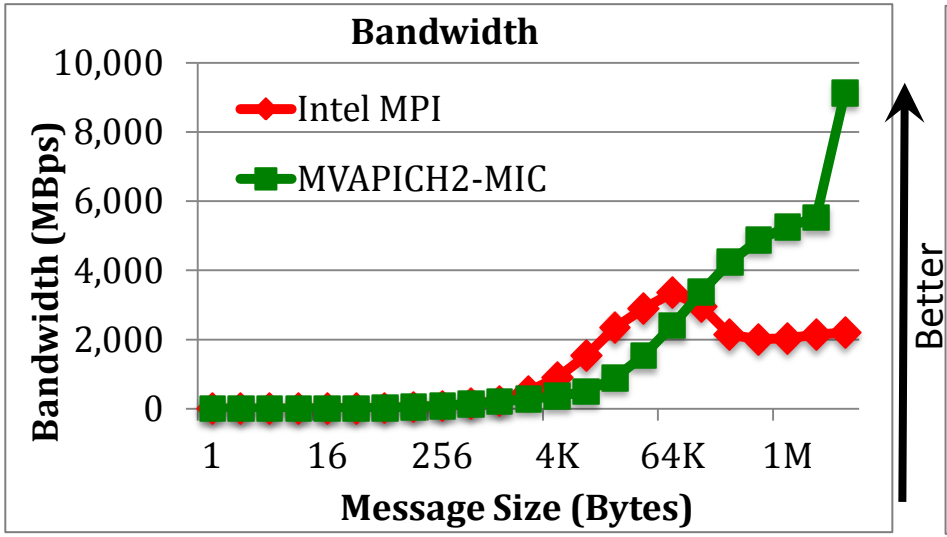


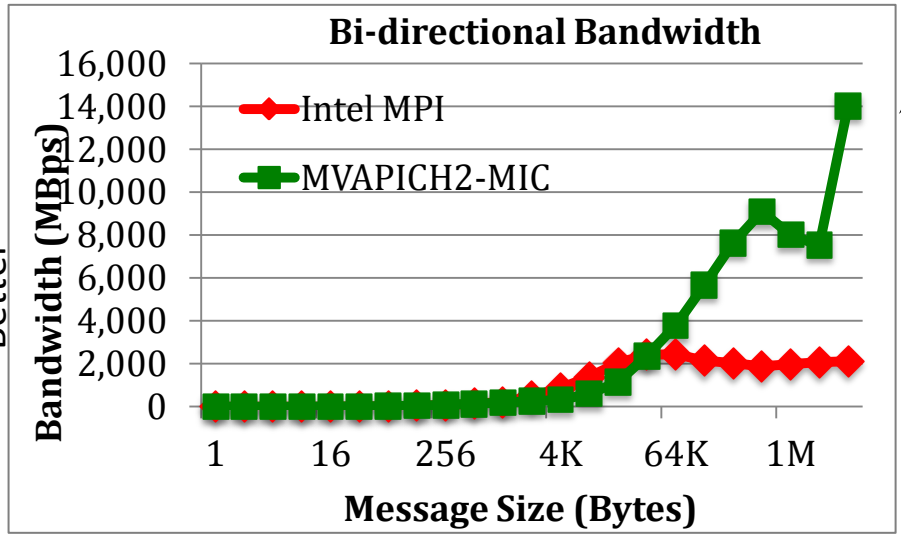
MVAPICH2-MIC (based on MVAPICH2 1.9)
Intel Sandy Bridge (E5-2670) node with 16 cores, SE10P (B0-KNC),
MPSS 2.1.6720-13, composer_xe_2013.4.183, and IB FDR MT4099 HCA

MVAPICH2 and Intel MPI – Intra-MIC



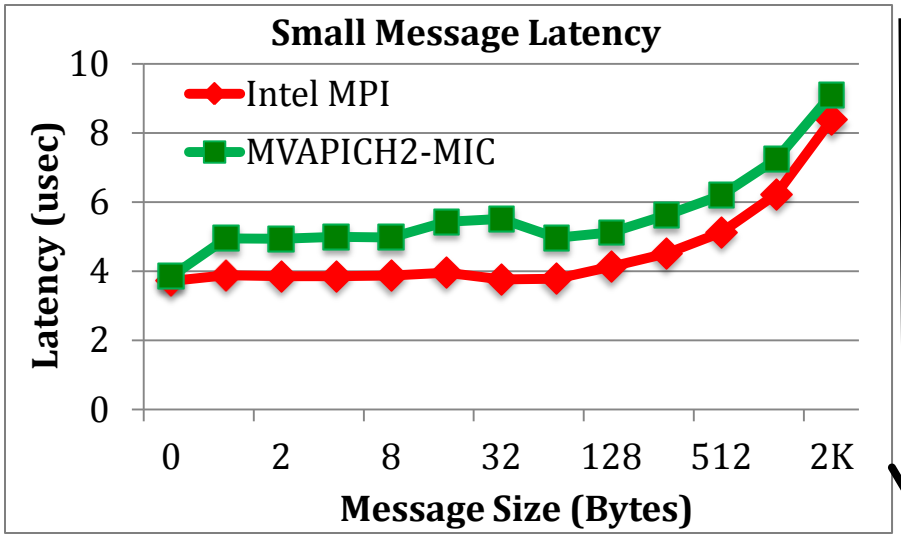


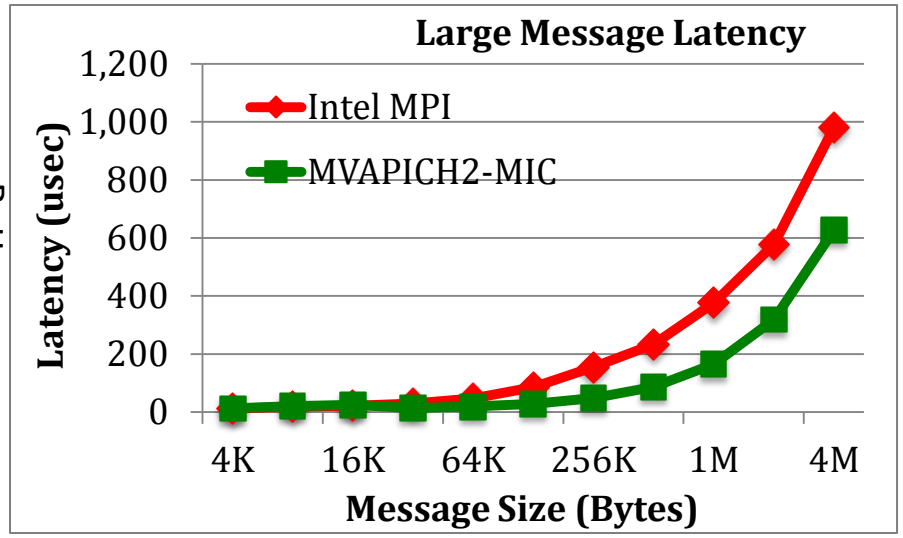


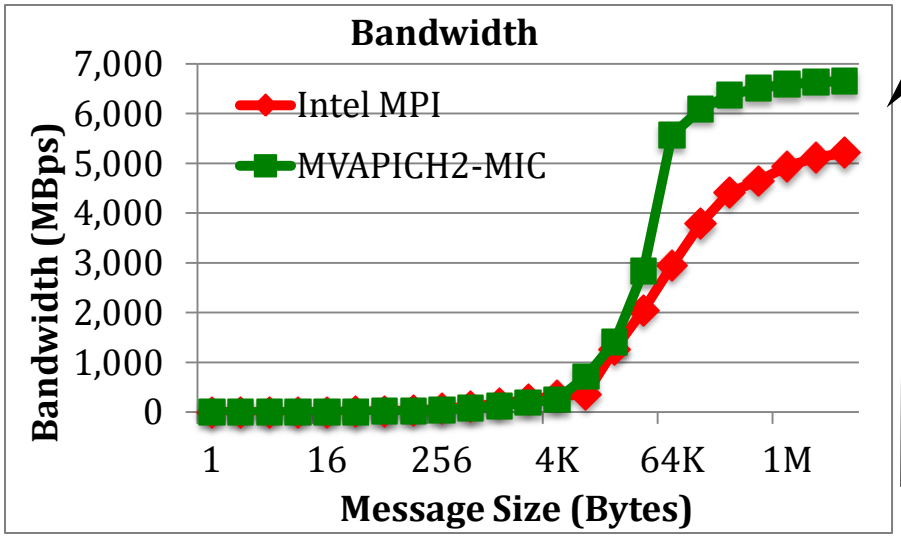


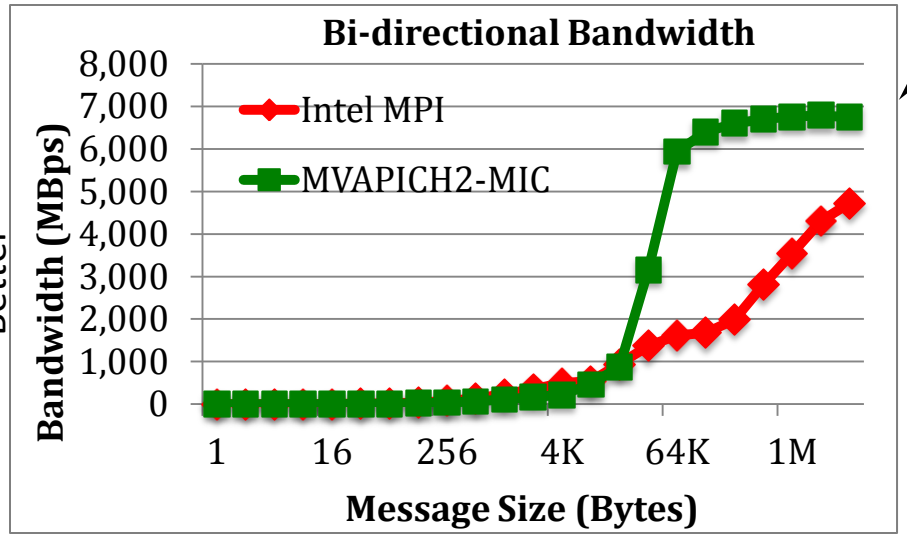
MVAPICH2-MIC (based on MVAPICH2 1.9), Intel MPI 4.1.1.036
Intel Sandy Bridge (E5-2670) node with 16 cores, SE10P (B0-KNC),
MPSS 2.1.6720-13, composer_xe_2013.4.183, and IB FDR MT4099 HCA

MVAPICH2 and Intel MPI – Intranode – Host-MIC



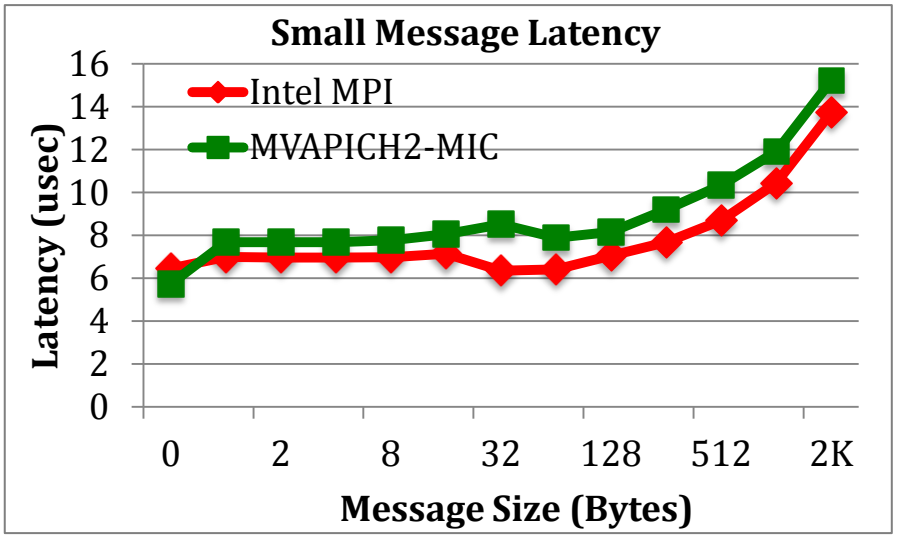


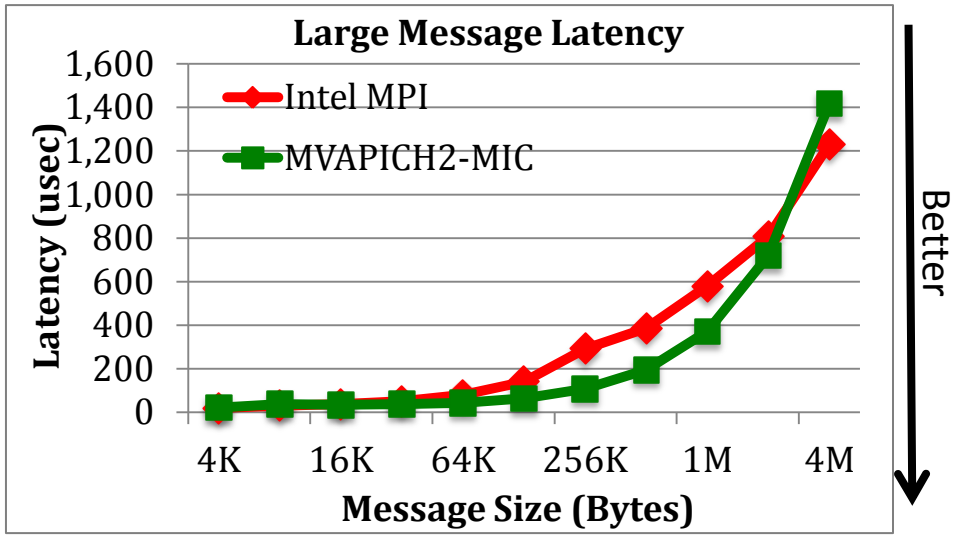


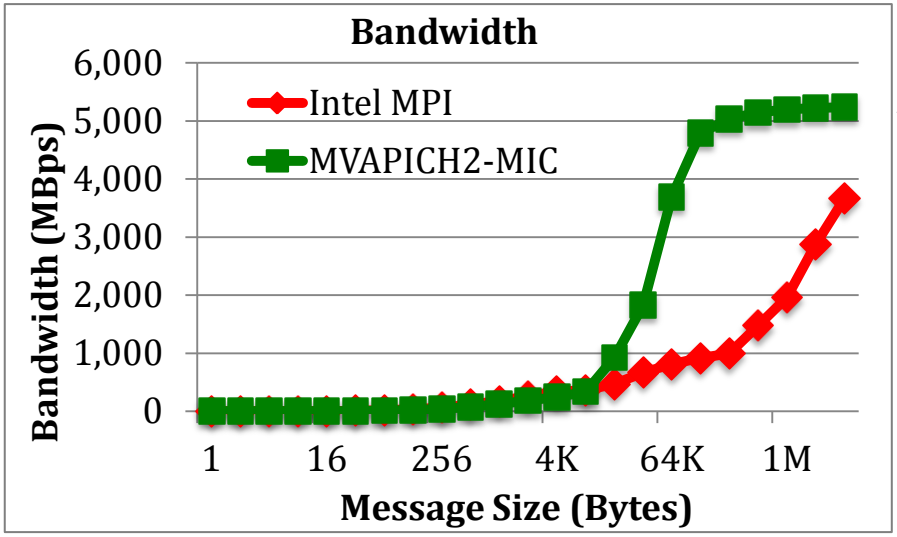


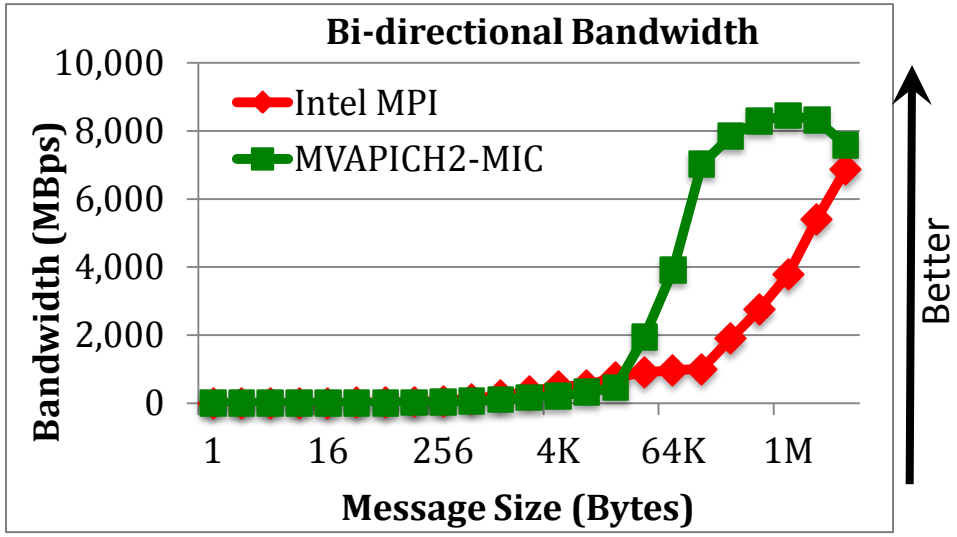
MVAPICH2-MIC (based on MVAPICH2 1.9), Intel MPI 4.1.1.036
Intel Sandy Bridge (E5-2670) node with 16 cores, SE10P (B0-KNC),
MPSS 2.1.6720-13, composer_xe_2013.4.183, and IB FDR MT4099 HCA

MVAPICH2 and Intel MPI – Internode MIC-MIC







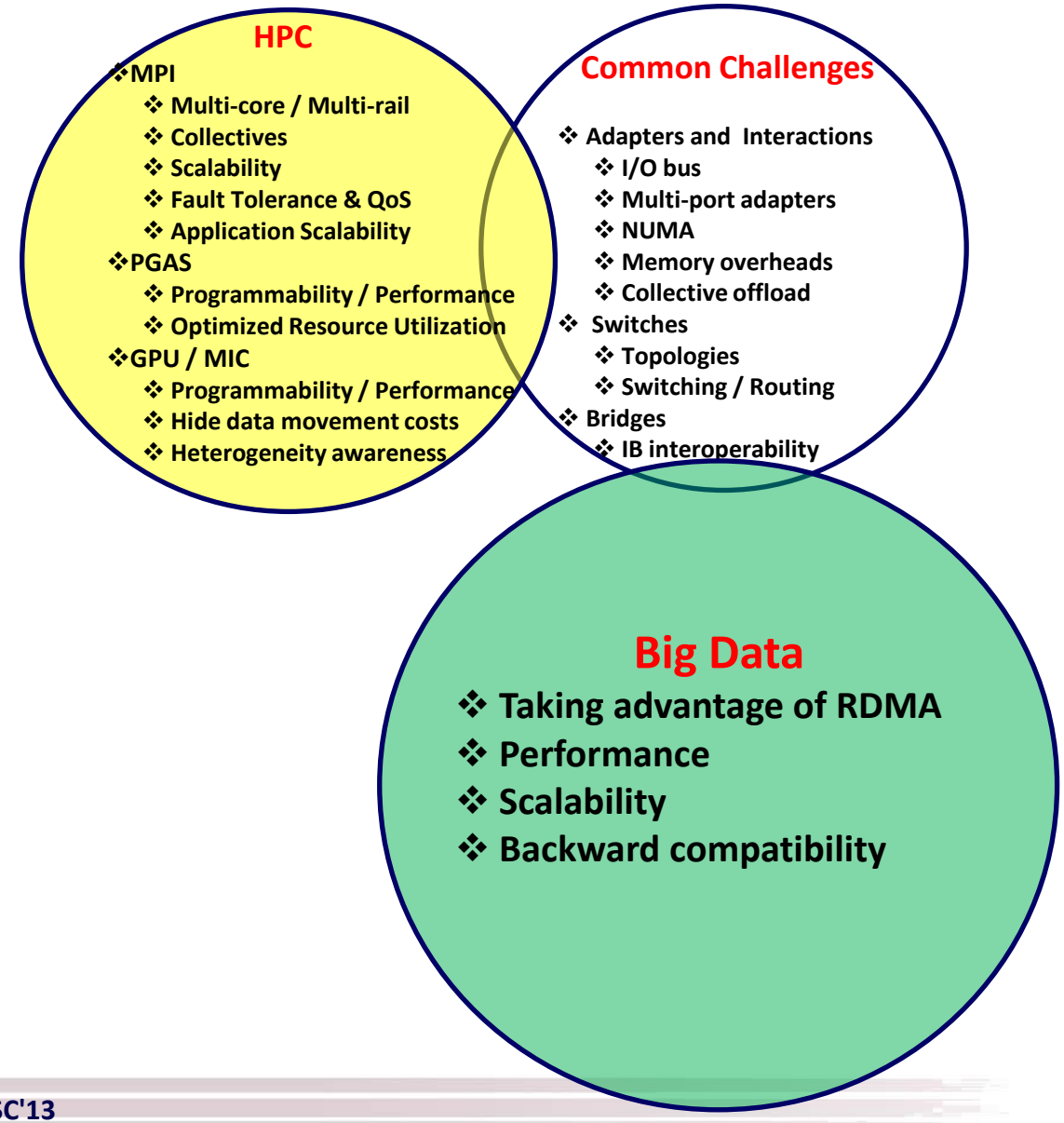


MVAPICH2-MIC (based on MVAPICH2 1.9), Intel MPI 4.1.1.036
Intel Sandy Bridge (E5-2670) node with 16 cores, SE10P (B0-KNC),
MPSS 2.1.6720-13, composer_xe_2013.4.183, and IB FDR MT4099 HCA

Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- **System Specific Challenges and Case Studies**
  - **HPC (MPI, PGAS and GPU/MIC Computing)**
  - **Big Data (Hadoop with HDFS and HBase; Memcached)**
  - Storage and File Systems
  - Grid Computing
- Open Fabrics Software Stack and RDMA Programming
- Network Management Infrastructure and Tools
- Conclusions and Final Q&A

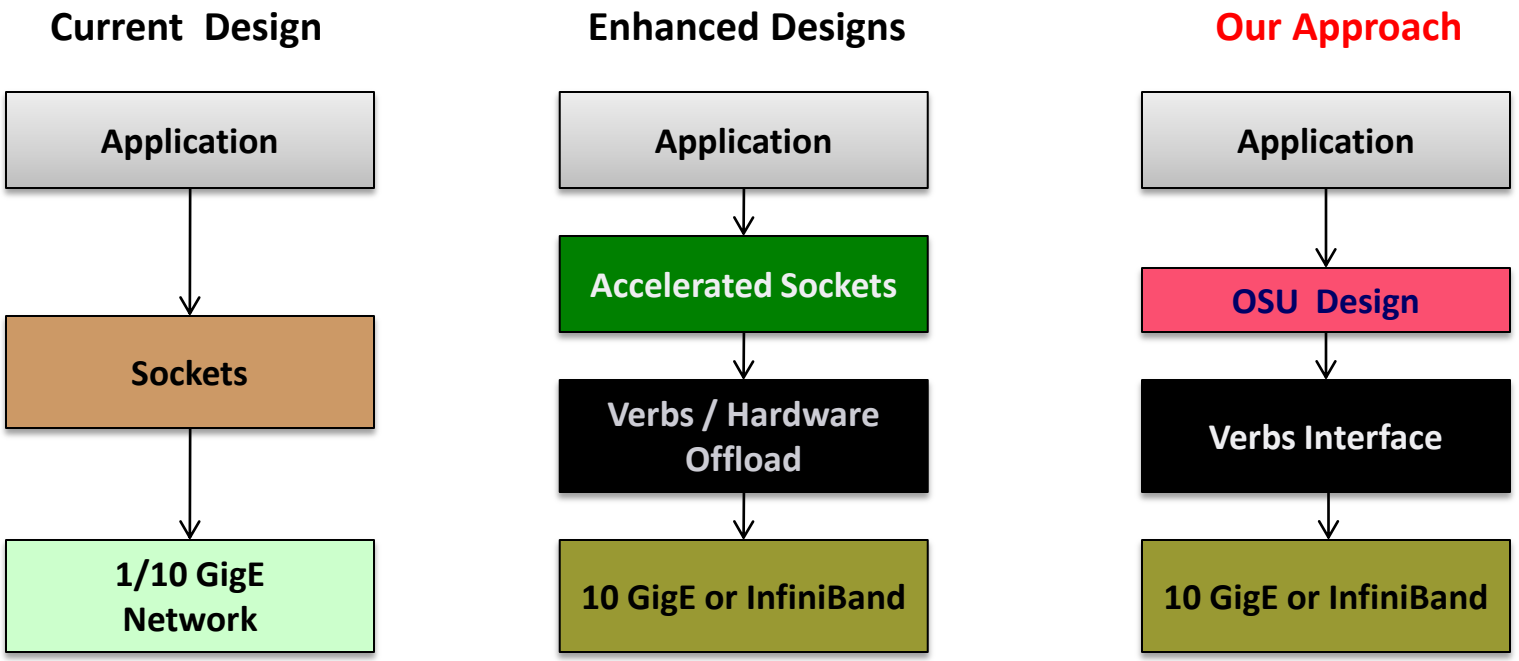
System Specific Challenges for Big Data Processing



Can High-Performance Interconnects Benefit Big Data Processing?

- Beginning to draw interest from the enterprise domain
  - Oracle, IBM, Google are working along these directions
- Performance in the enterprise domain remains a concern
- Where do the bottlenecks lie?
- Can these bottlenecks be alleviated with new designs?

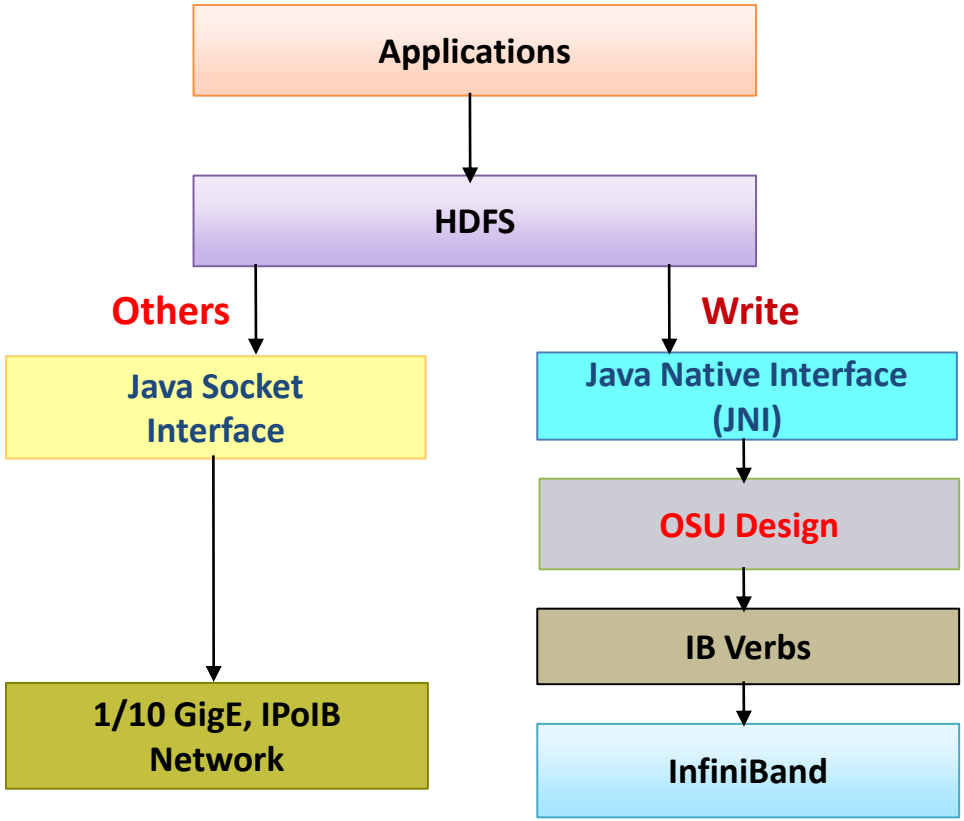
Can Big Data Processing Systems be Designed with High-Performance Networks and Protocols?



- Sockets not designed for high-performance
  - Stream semantics often mismatch for upper layers (Memcached, HBase, Hadoop)
  - Zero-copy not available for non-blocking sockets

HDFS-RDMA Design Overview

Enables high performance RDMA communication, while supporting traditional socket interface

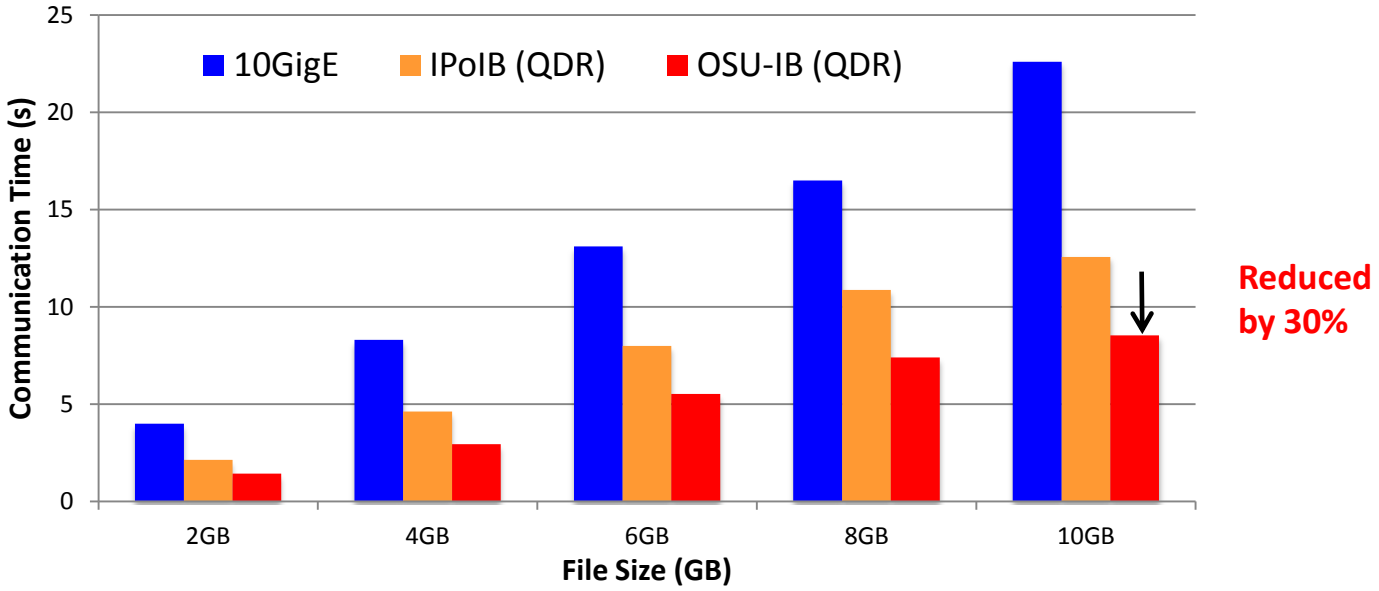


HDFS Write involves replication; more network intensive

HDFS Read is mostly node-local

- JNI Layer bridges Java based HDFS with communication library written in native code
- **Only the communication part of HDFS Write is modified; No change in HDFS architecture**

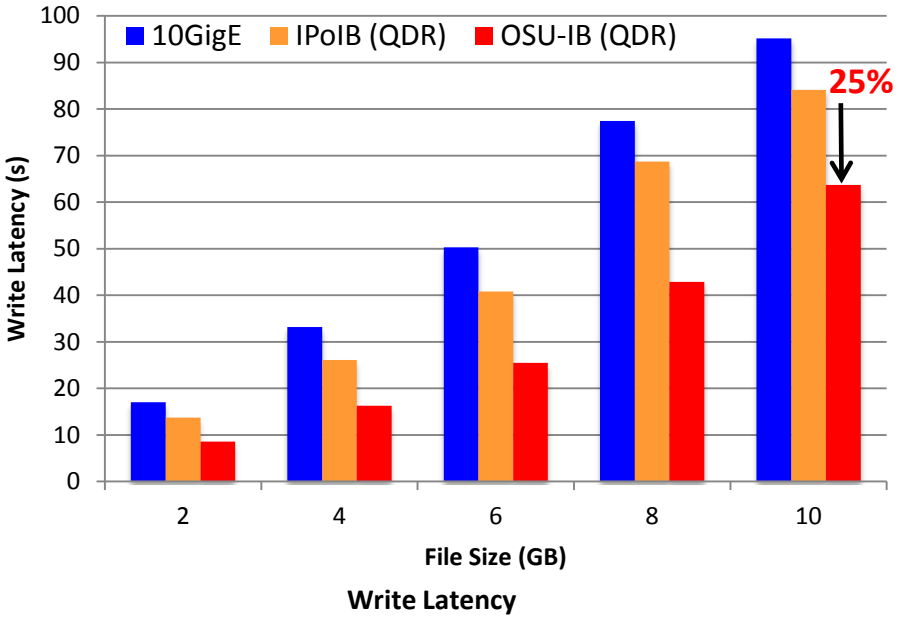
Communication Times in HDFS



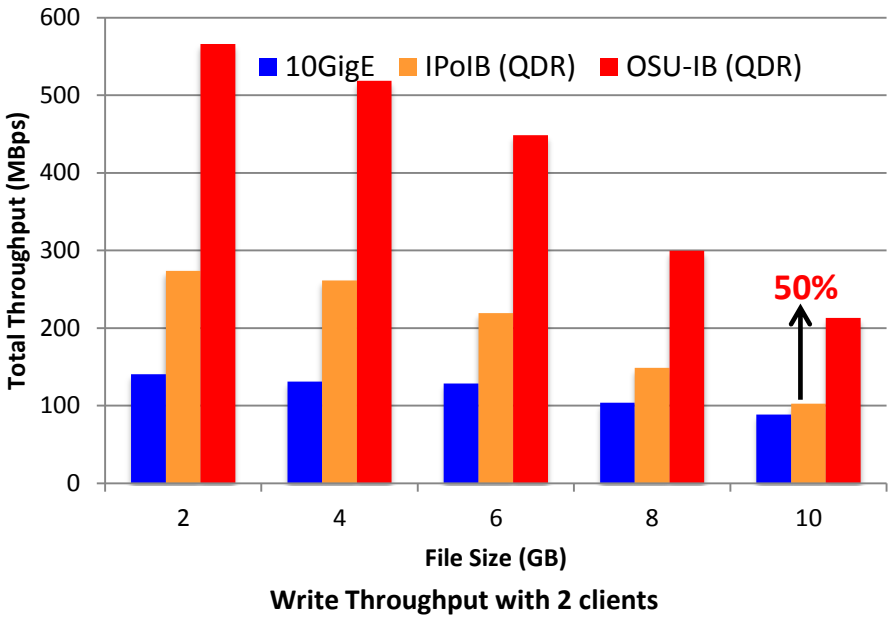
- Cluster with HDD DataNodes
  - **30%** improvement in communication time over IPoIB (QDR)
  - **56%** improvement in communication time over 10GigE
- Similar improvements are obtained for SSD DataNodes

N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy and D. K. Panda , High Performance RDMA-Based Design of HDFS over InfiniBand , Supercomputing (SC), Nov 2012

Evaluations using HDFS Micro-benchmark



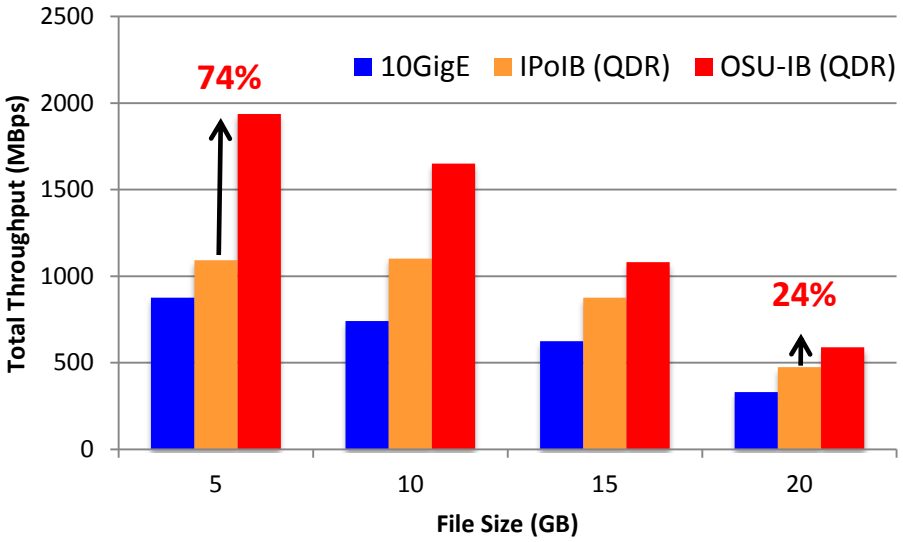
Write Latency



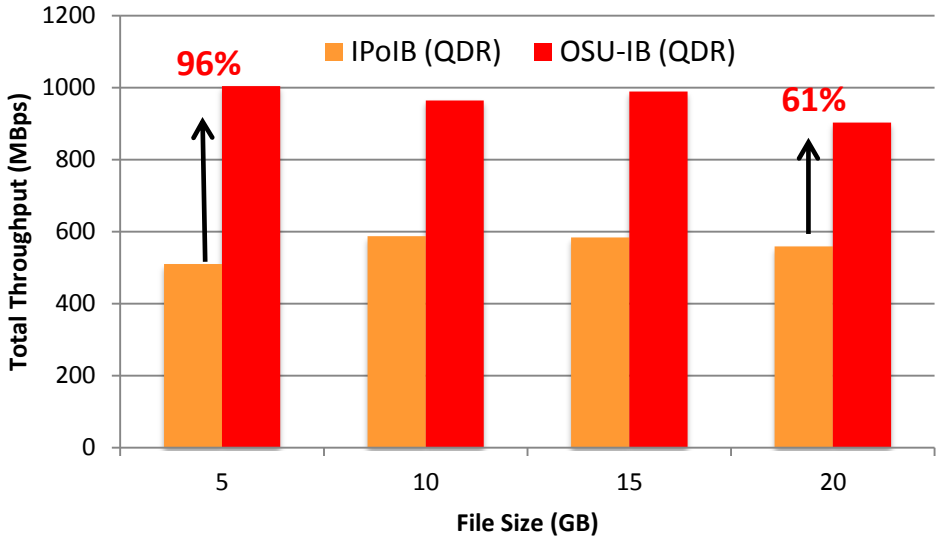
Write Throughput with 2 clients

- Cluster with 4 HDD DataNodes, single disk per node
  - **25%** improvement in latency over IPoIB (QDR) for 10GB file size
  - **50%** improvement in throughput over IPoIB (QDR) for 10GB file size

Evaluations using TestDFSIO



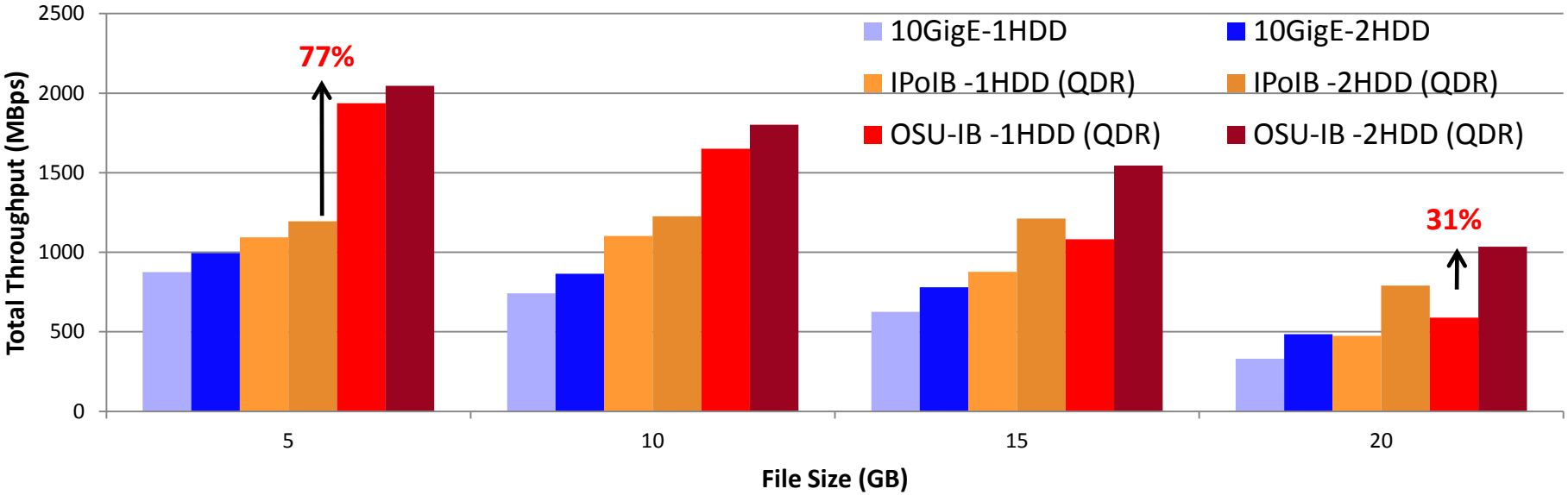
Cluster with 8 HDD Nodes, TestDFSIO with 8 maps



Cluster with 4 SSD Nodes, TestDFSIO with 4 maps

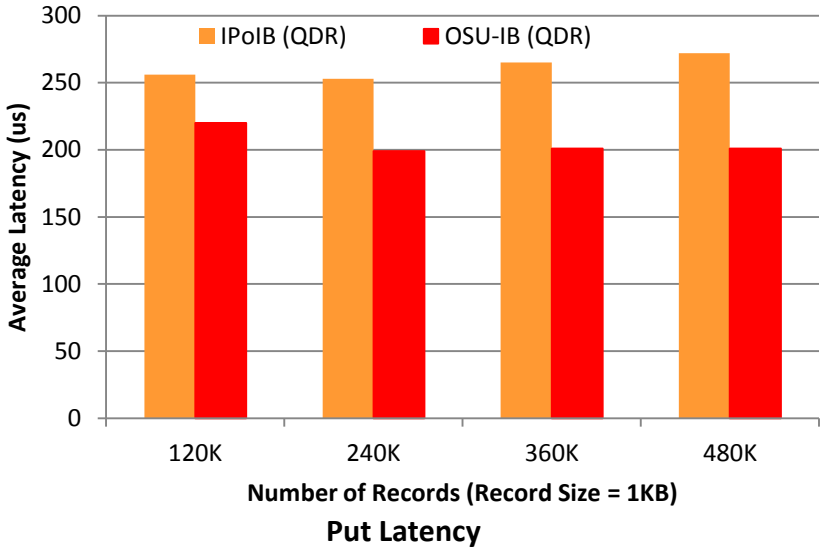
- Cluster with 8 HDD DataNodes, single disk per node
  - **24%** improvement over IPoIB (QDR) for 20GB file size
- Cluster with 4 SSD DataNodes, single SSD per node
  - **61%** improvement over IPoIB (QDR) for 20GB file size

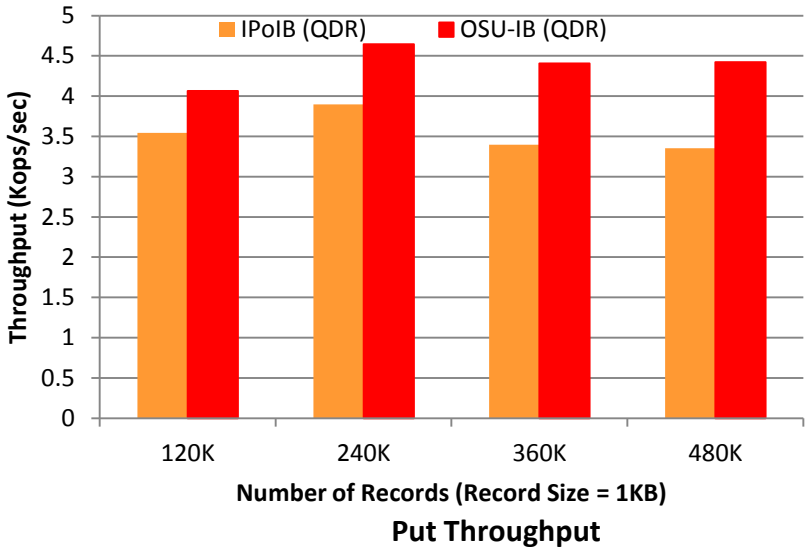
Evaluations using TestDFSIO



- Cluster with 8 DataNodes, **1 HDD** per node, TestDFSIO with 8 maps
  - **24%** improvement over IPoIB (QDR) for 20GB file size
- Cluster with 8 DataNodes, **2 HDD** per node, TestDFSIO with 8 maps
  - **31%** improvement over IPoIB (QDR) for 20GB file size
- 2 HDD vs 1 HDD
  - **76%** improvement for OSU-IB (QDR)
  - **66%** improvement for IPoIB (QDR)

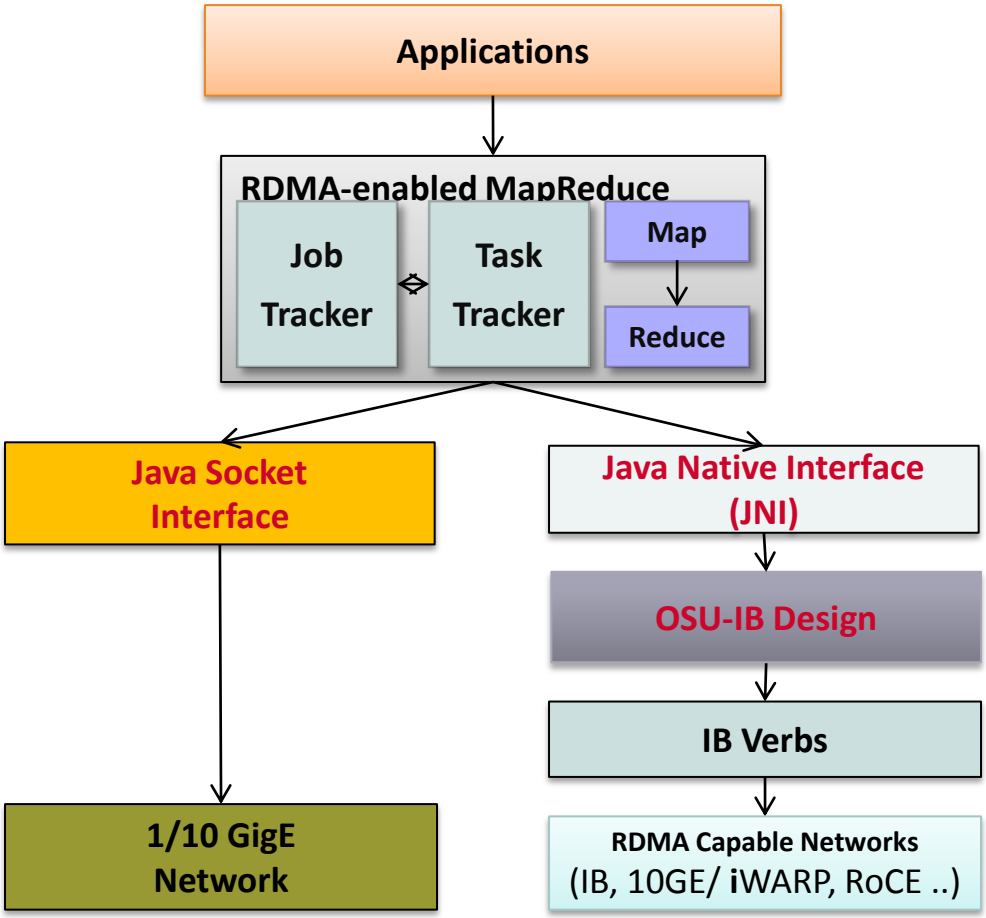
Evaluations using YCSB (32 Region Servers: 100% Update)





- HBase using TCP/IP, running over HDFS-IB
- HBase ***Put*** latency for 480K records
  - **201** us for OSU Design; **272** us for IPoIB (QDR)
- HBase ***Put*** throughput for 480K records
  - **4.42** Kops/sec for OSU Design; **3.63** Kops/sec for IPoIB (QDR)
- **26%** improvement in average latency; **24%** improvement in throughput

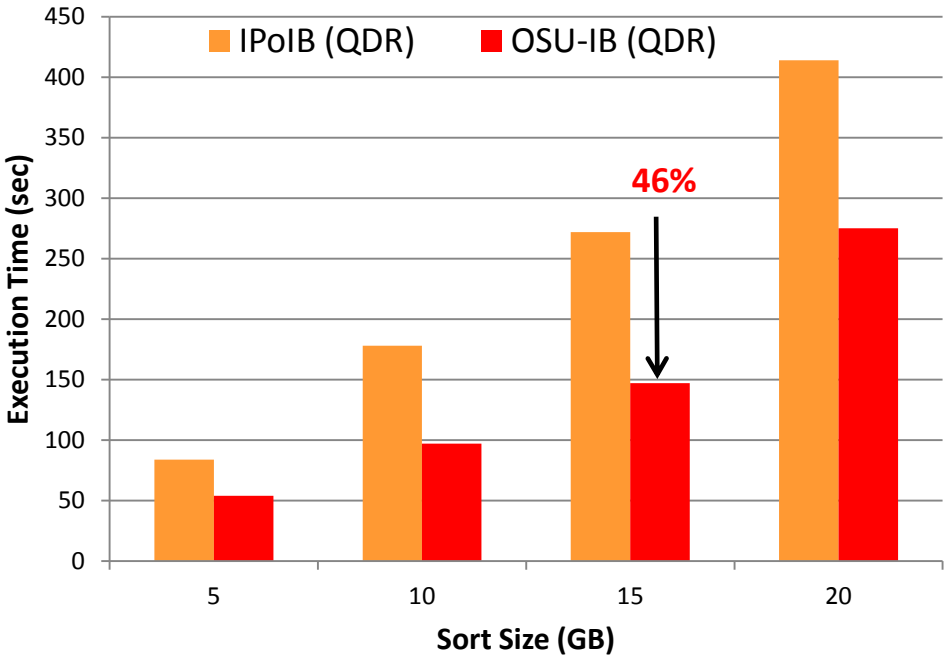
MapReduce-RDMA Design Overview

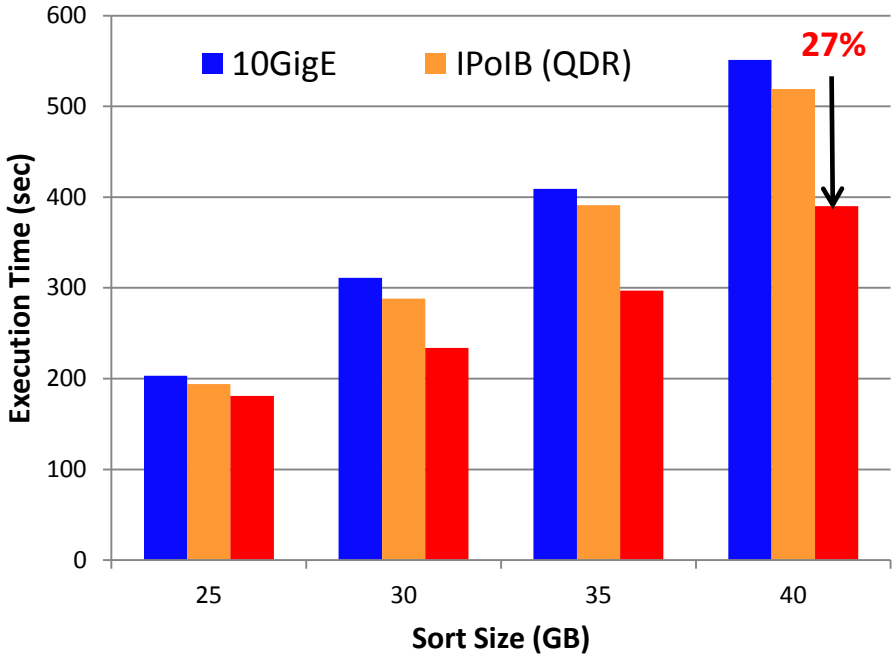


- Enables high performance RDMA communication, while supporting traditional socket interface

M. W. Rahman, N. S. Islam, X. Lu, J. Jose, H. Subramon, H. Wang and D. K. Panda, High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand, Int'l Workshop on High Performance Data Intensive Computing (HPDIC), held in conjunction with Int'l Parallel and Distributed Processing Symposium (IPDPS '13), May 2013.

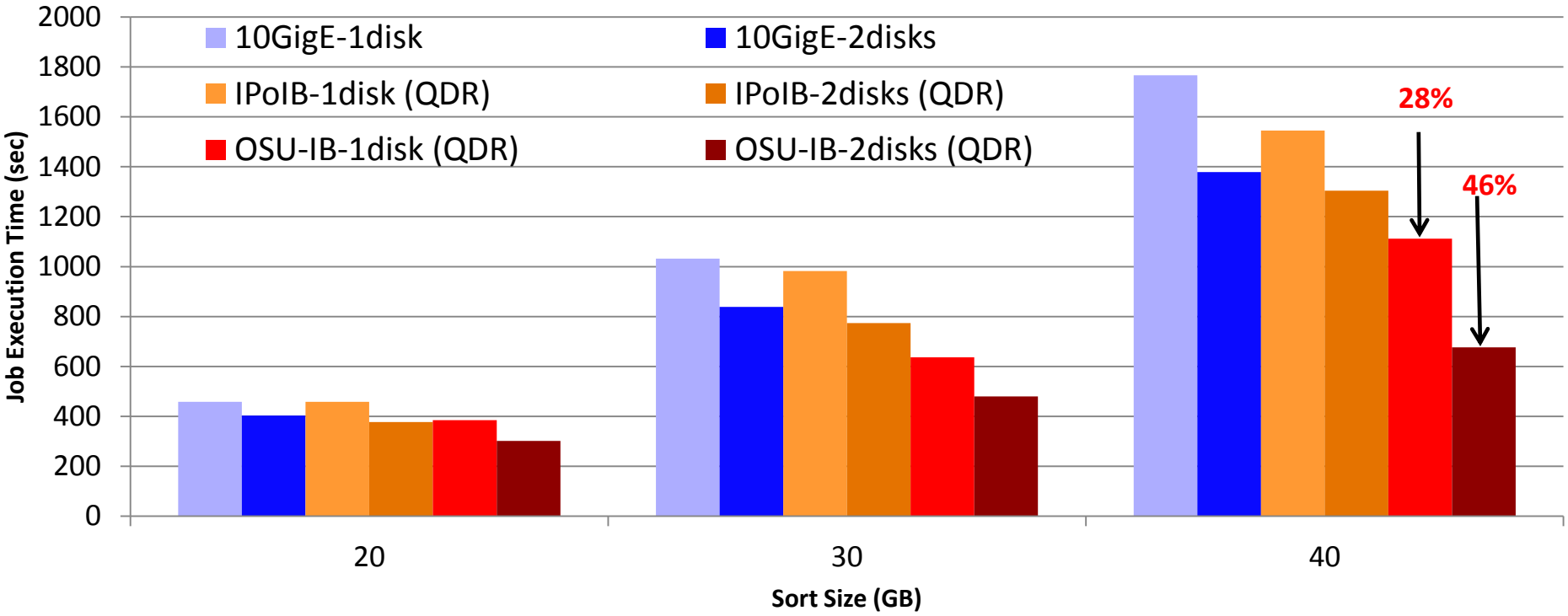
Evaluations using Sort





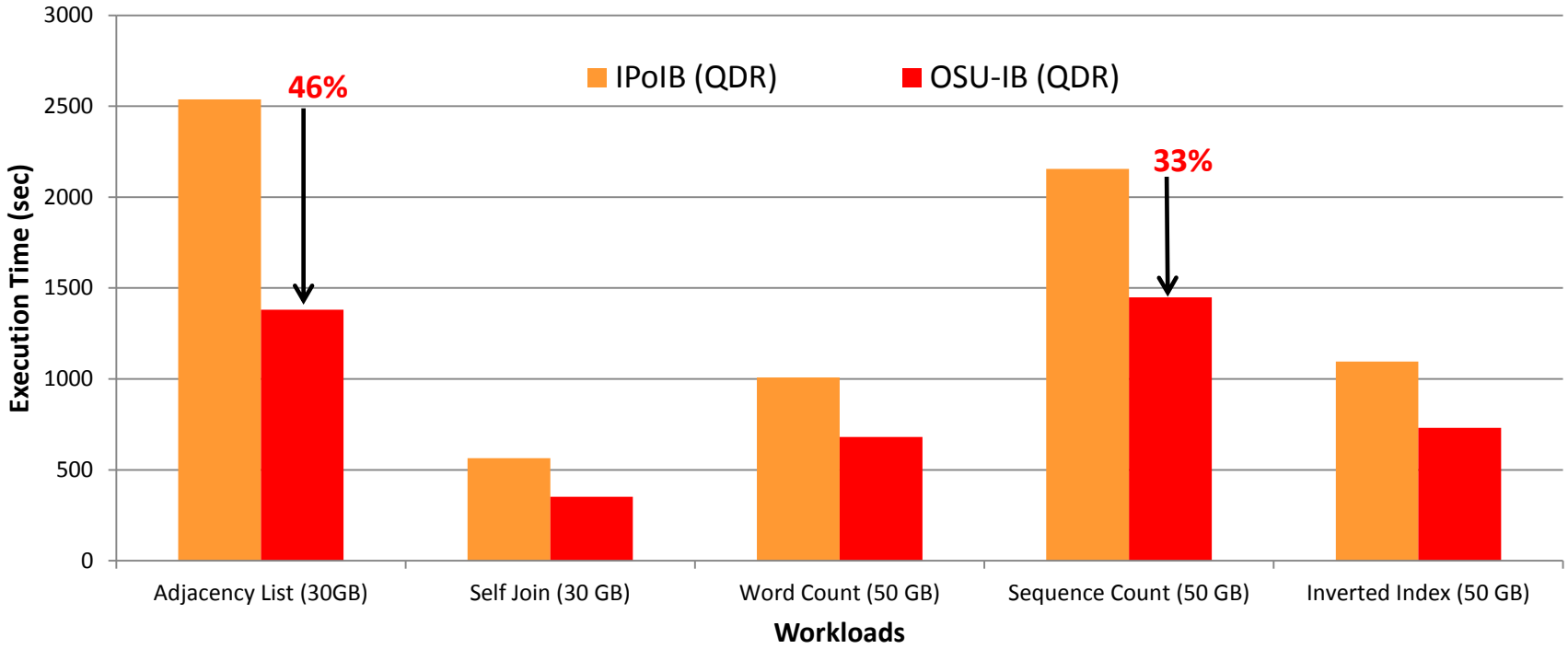
- For 5-20 GB Sort, 4-node cluster with a single SSD per node
- For 25-40 GB Sort, 8-node cluster with a single HDD per node
- 46% improvement over IPoIB (QDR) for 15 GB Sort
- 27% improvement over IPoIB (QDR) for 40 GB Sort

Evaluations using TeraSort



- 40 GB TeraSort with 4 DataNodes with single disk per node
  - **37%** improvement over 10GigE
  - **28%** improvement over IPoIB (QDR)
- 40 GB TeraSort with 4 DataNodes with 2 disks per node
  - **49%** improvement over 10GigE
  - **46%** improvement over IPoIB (QDR)

Evaluations using PUMA Workload

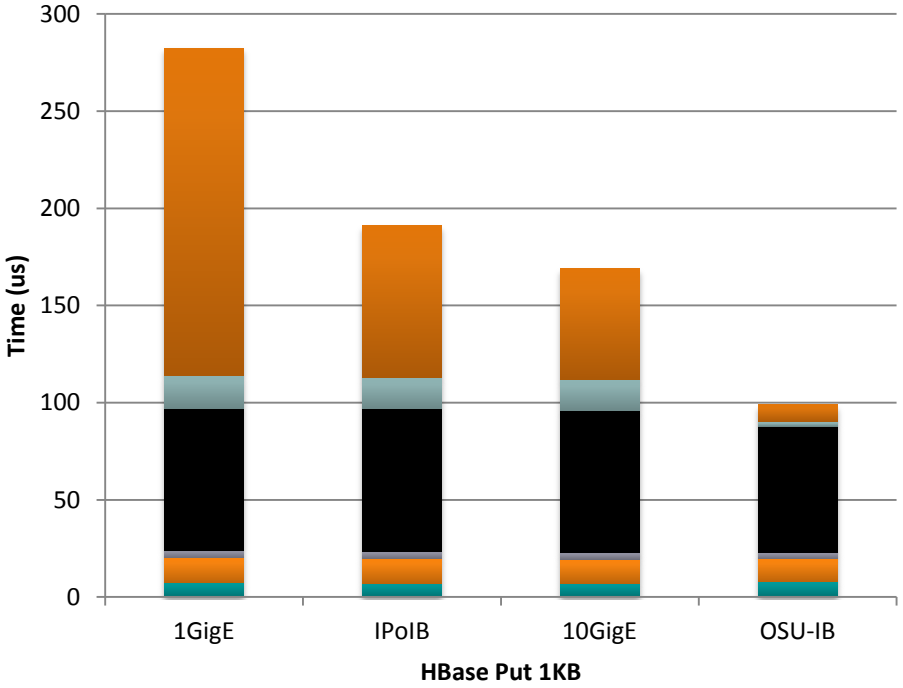


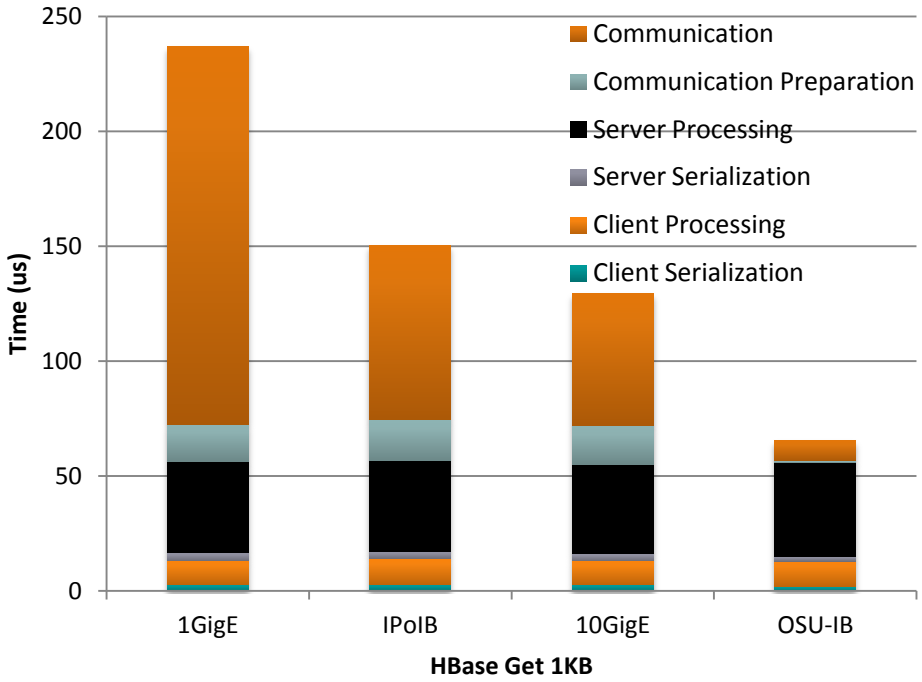
- The DataSet for these workloads are taken from PUMA (Purdue MapReduce Benchmark Suite)
- 46% improvement in Adjacency List over IPoIB (QDR) for 30 GB data size
- 33% improvement in Sequence Count over IPoIB (QDR) for 50 GB data size

Hadoop-RDMA Software

- High-Performance Design of Hadoop over RDMA-enabled Interconnects
  - High performance design with native InfiniBand support at the verbs-level for HDFS, MapReduce, and RPC components
  - Easily configurable for both native InfiniBand and the traditional sockets-based support (Ethernet and InfiniBand with IPoIB)
  - Current release: 0.9.0
    - Based on Apache Hadoop 0.20.2
    - Compliant with Apache Hadoop 0.20.2 APIs and applications
    - Tested with
      - Mellanox InfiniBand adapters (DDR, QDR and FDR)
      - Various multi-core platforms
      - Different file systems with disks and SSDs
  - Can be downloaded from http://hadoop-rdma.cse.ohio-state.edu

HBase Put/Get – Detailed Analysis

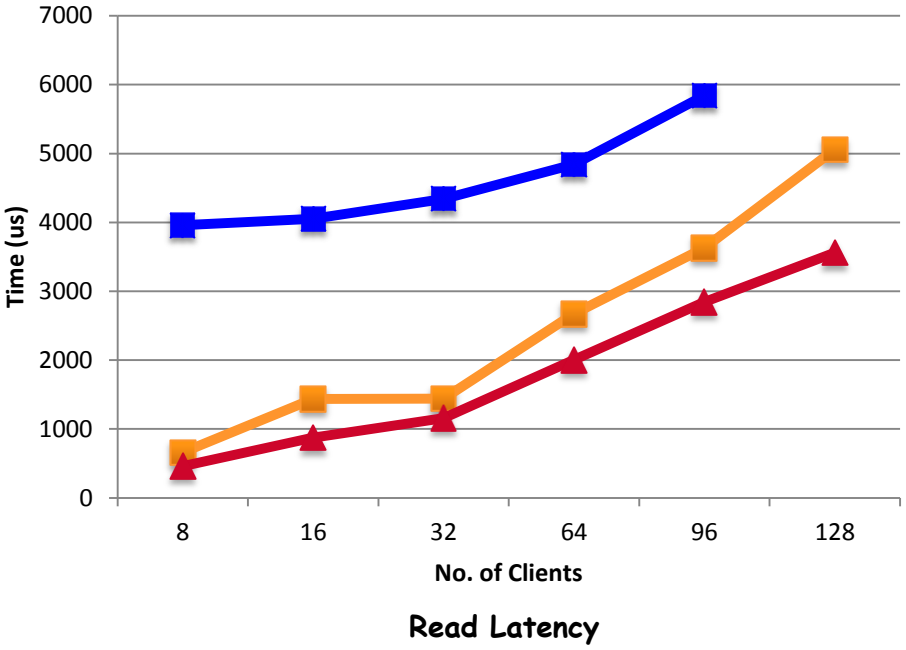




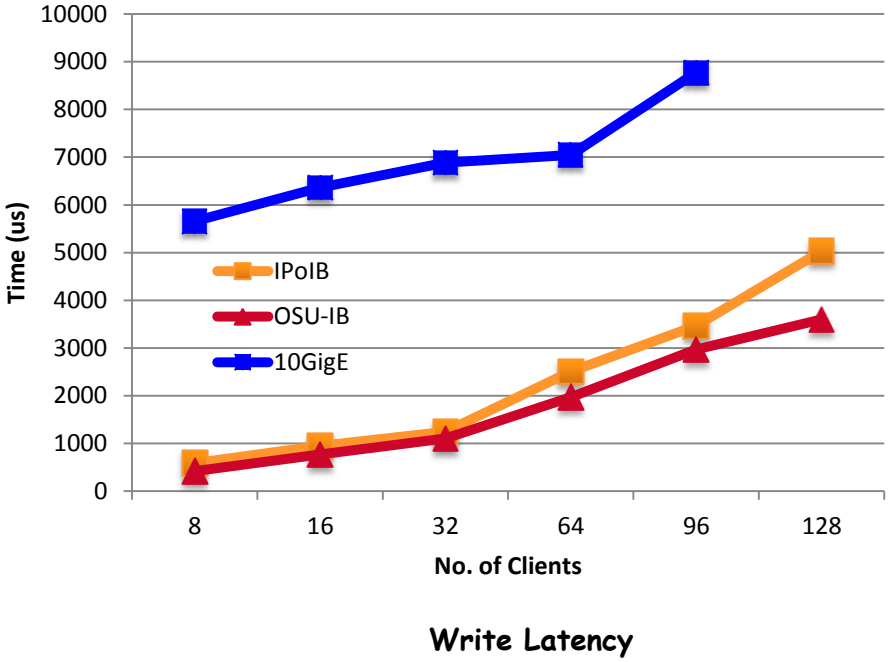
- HBase 1KB Put
  - Communication Time – 8.9 us
  - A factor of 6X improvement over 10GE for communication time
- HBase 1KB Get
  - Communication Time – 8.9 us
  - A factor of 6X improvement over 10GE for communication time

M. W. Rahman, J. Huang, J. Jose, X. Ouyang, H. Wang, N. Islam, H. Subramoni, Chet Murthy and D. K. Panda, Understanding the Communication Characteristics in HBase: What are the Fundamental Bottlenecks?, ISPASS'12

HBase – YCSB Read-Write Workload



Read Latency

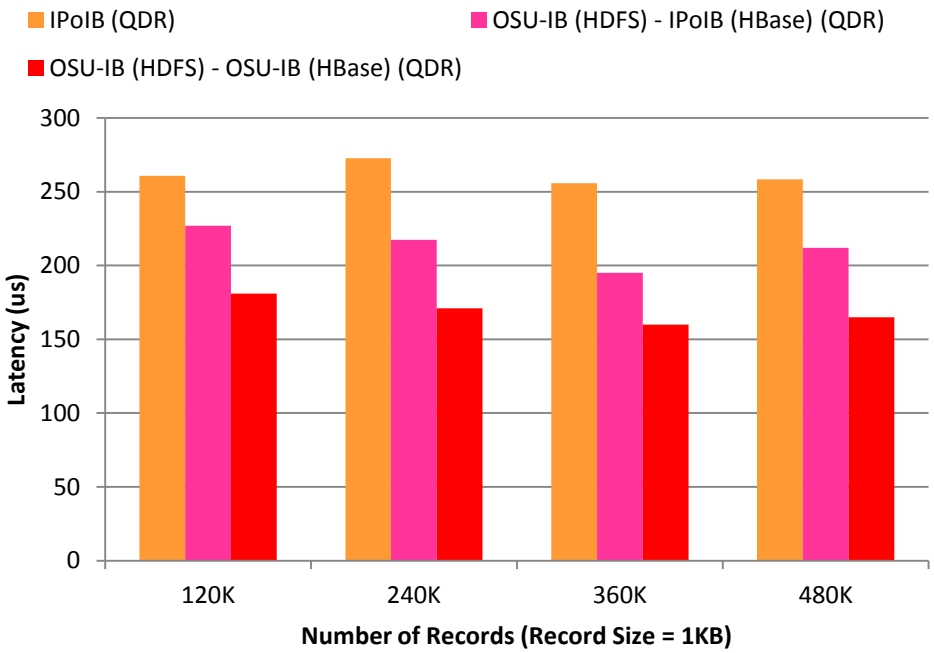


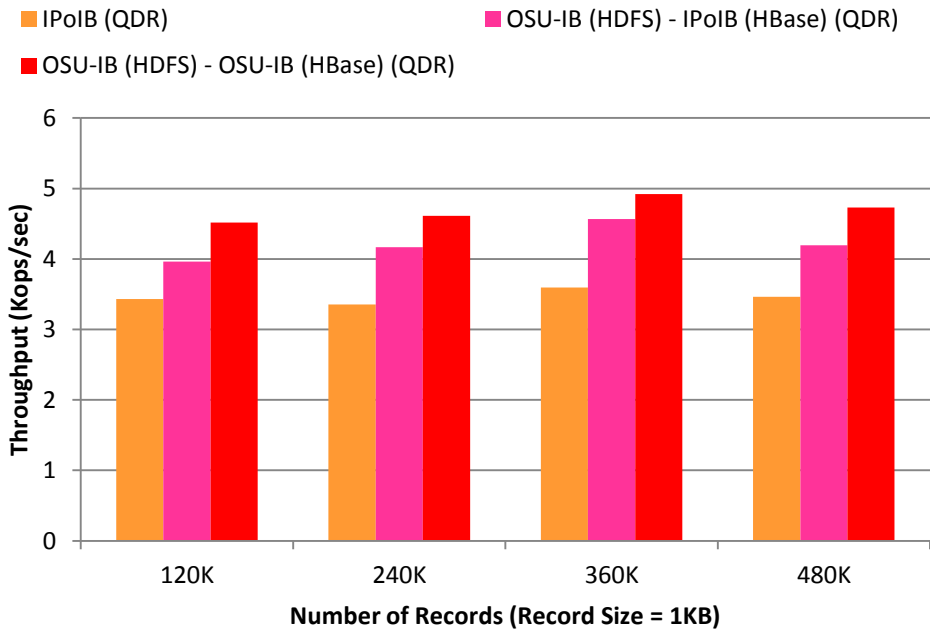
Write Latency

- HBase Get latency (Yahoo! Cloud Service Benchmark)
  - 64 clients: 2.0 ms; 128 Clients: 3.5 ms
  - 42% improvement over IPoIB for 128 clients
- HBase Get latency
  - 64 clients: 1.9 ms; 128 Clients: 3.5 ms
  - 40% improvement over IPoIB for 128 clients

J. Huang, X. Ouyang, J. Jose, M. W. Rahman, H. Wang, M. Luo, H. Subramoni, Chet Murthy and D. K. Panda, High-Performance Design of HBase with RDMA over InfiniBand, IPDPS'12

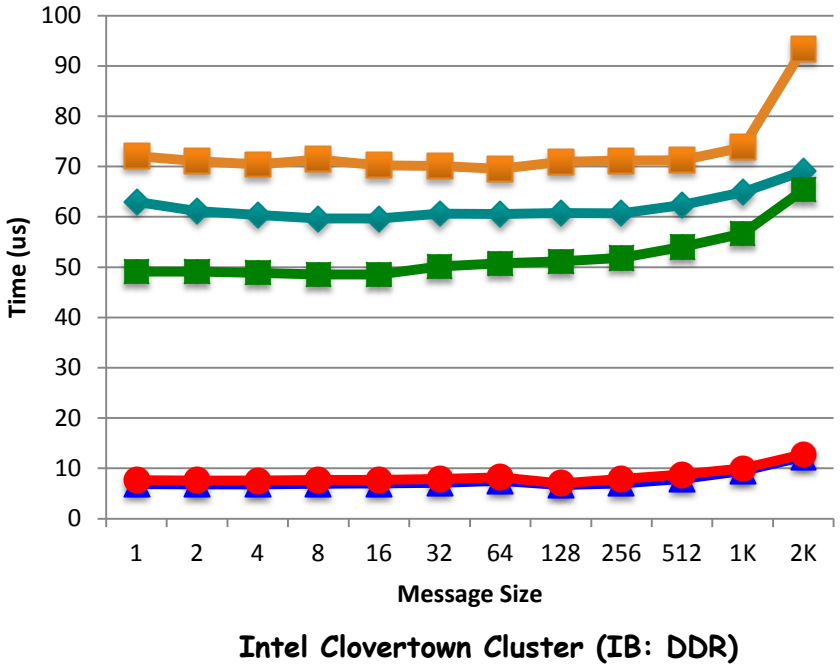
HDFS and HBase Integration over IB (OSU-IB)



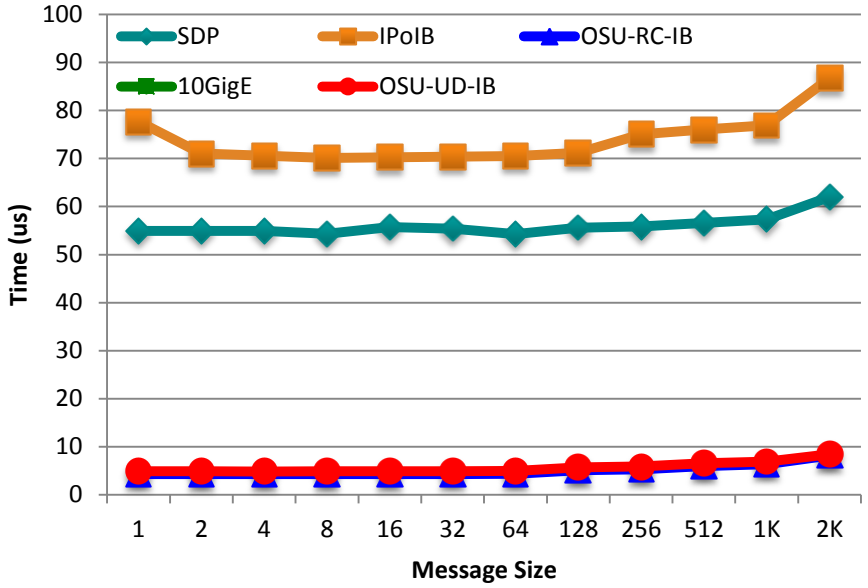


- YCSB Evaluation with 4 RegionServers (100% update)
- HBase Put Latency and Throughput for 360K Records
  - 37% improvement over IPoIB (QDR)
  - 18% improvement over OSU-IB HDFS only

Memcached Get Latency (Small Message)



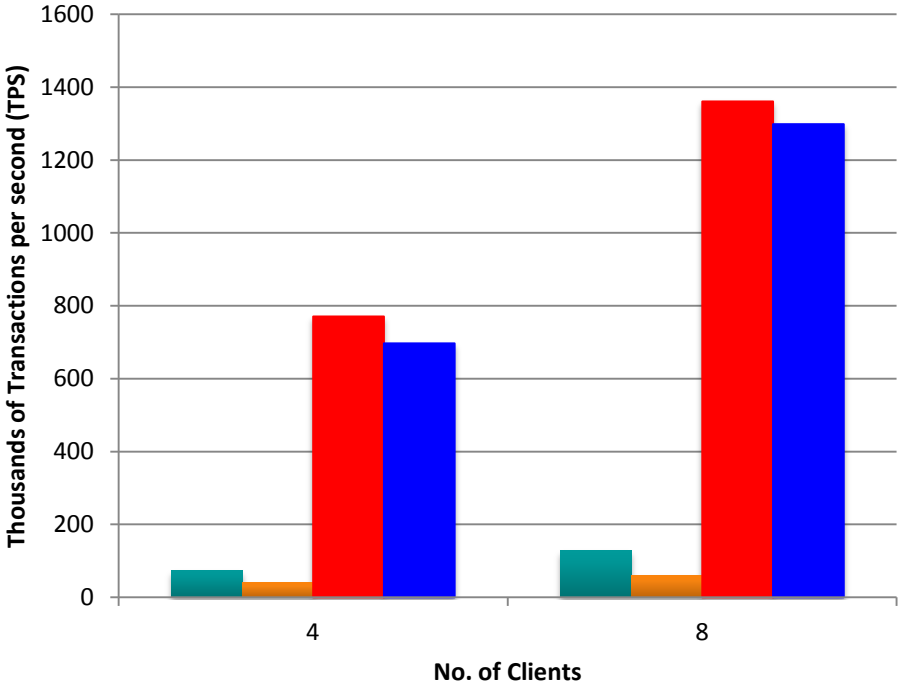
Intel Clovertown Cluster (IB: DDR)

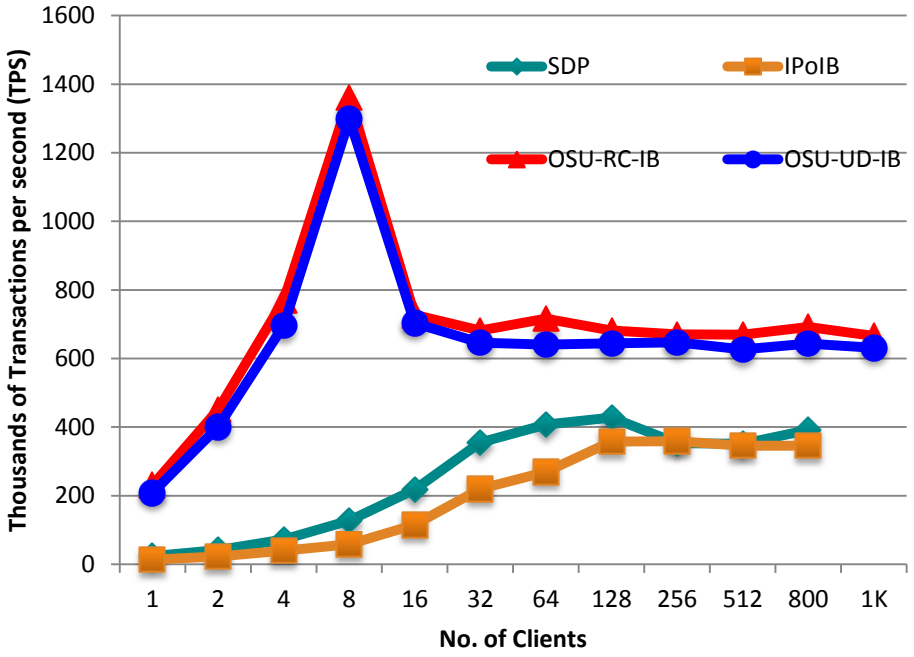


Intel Westmere Cluster (IB: QDR)

- Memcached Get latency
  - 4 bytes RC/UD – DDR: 6.82/7.55 us; QDR: 4.28/4.86 us
  - 2K bytes RC/UD – DDR: 12.31/12.78 us; QDR: 8.19/8.46 us
- Almost factor of *four* improvement over 10GE (TOE) for 2K bytes on the DDR cluster

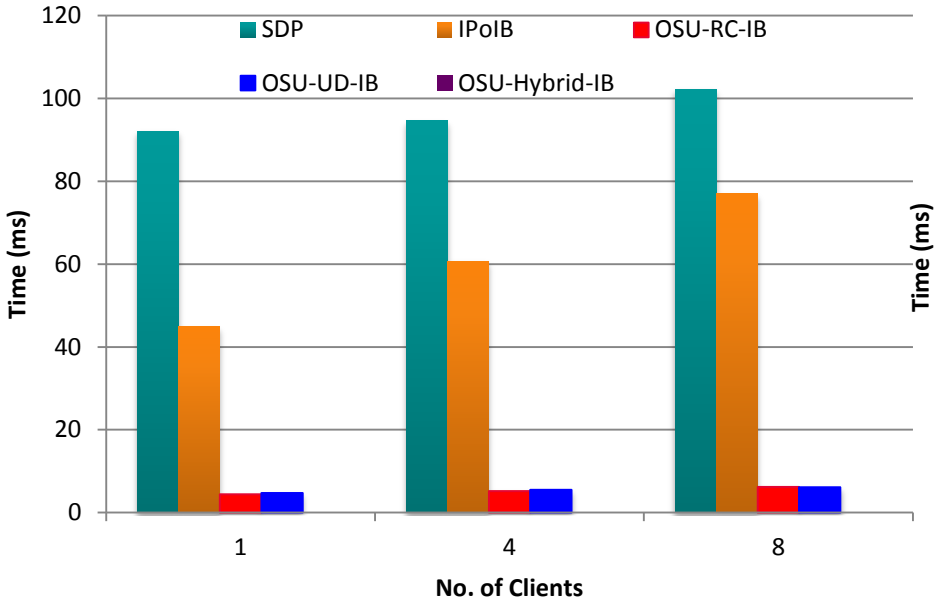
Memcached Get TPS (4byte)

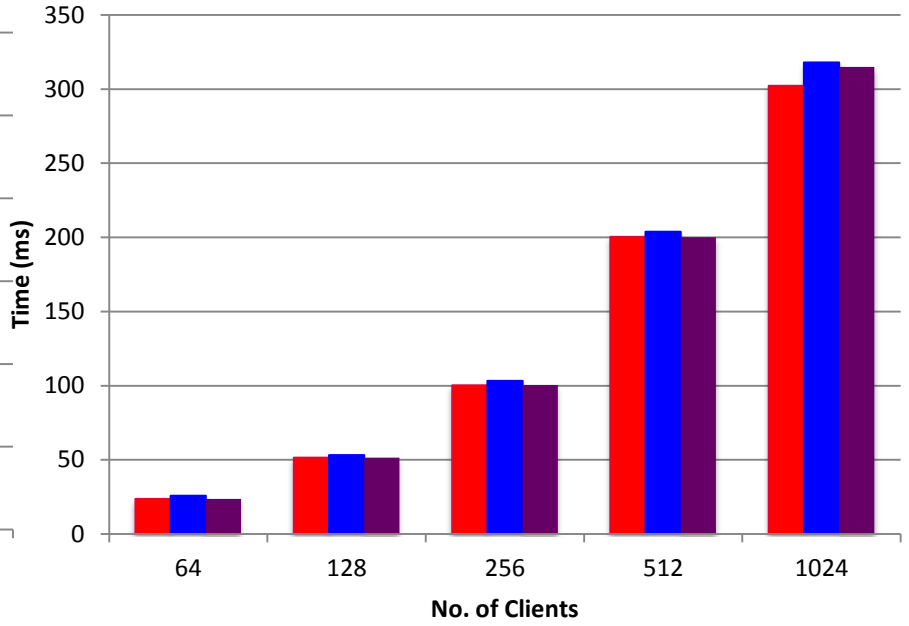




- Memcached Get transactions per second for 4 bytes
  - On IB QDR 1.4M/s (RC), 1.3 M/s (UD) for 8 clients
- Significant improvement with native IB QDR compared to SDP and IPoIB

Application Level Evaluation – Real Application Workloads





- Real Application Workload
  - RC – 302 ms, UD – 318 ms, Hybrid – 314 ms for 1024 clients
- 12X times better than IPoIB for 8 clients
- Hybrid design achieves comparable performance to that of pure RC design

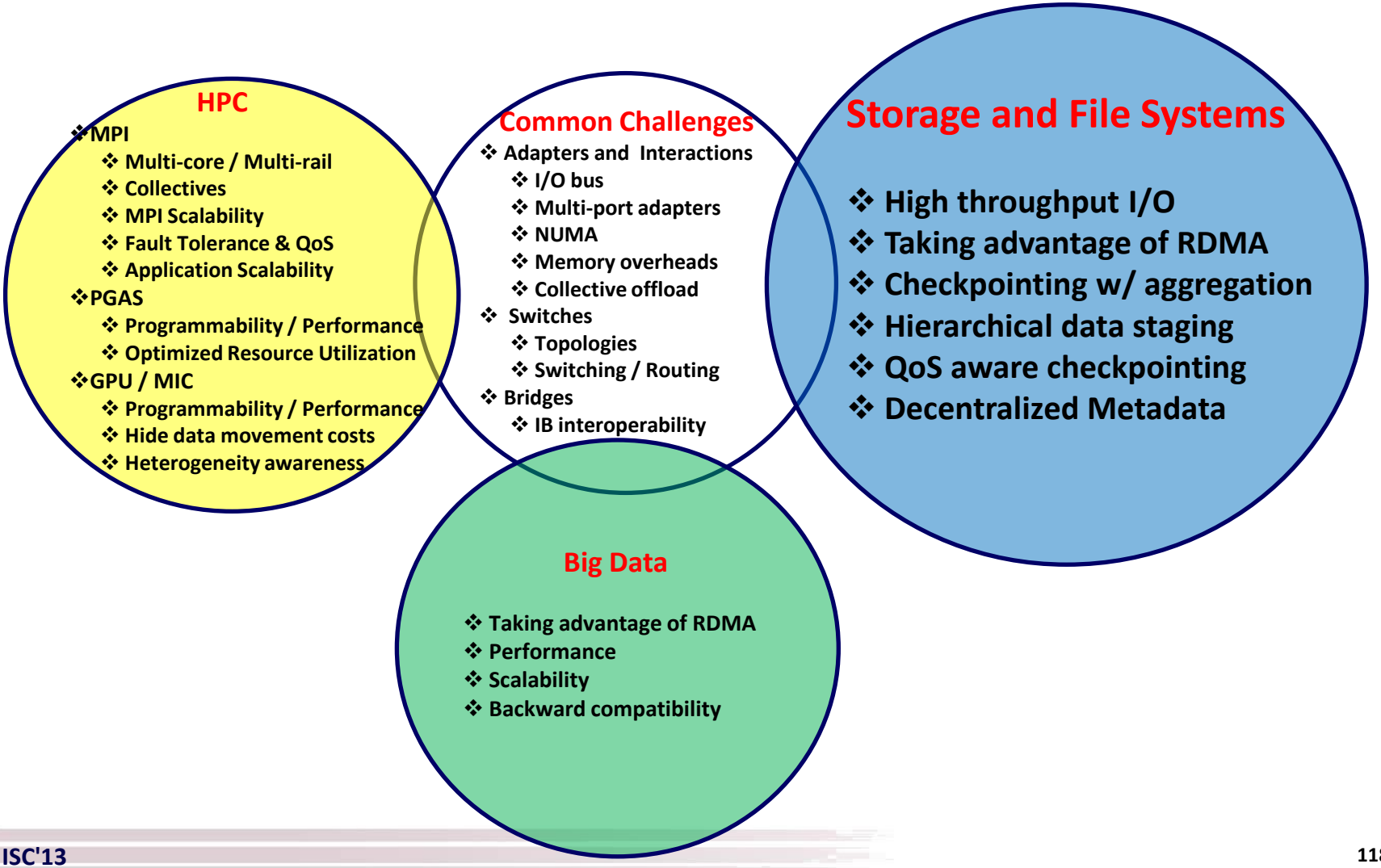
J. Jose, H. Subramoni, M. Luo, M. Zhang, J. Huang, W. Rahman, N. Islam, X. Ouyang, H. Wang, S. Sur and D. K. Panda, Memcached Design on High Performance RDMA Capable Interconnects, ICPP'11

J. Jose, H. Subramoni, K. Kandalla, W. Rahman, H. Wang, S. Narravula, and D. K. Panda, Scalable Memcached design for InfiniBand Clusters using Hybrid Transport, CCGrid'12

Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- **System Specific Challenges and Case Studies**
  - HPC (MPI, PGAS and GPU/MIC Computing)
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - **Storage and File Systems**
  - Grid Computing
- Open Fabrics Software Stack and RDMA Programming
- Network Management Infrastructure and Tools
- Conclusions and Final Q&A

System Specific Challenges for Storage and File Systems



Challenges in Designing HPC Storage Infrastructures

- Demand for networked-I/O driven by parallel programming models
- Constant need for higher throughput and lower latency
  - Graphics and visualization applications
  - Clustered DBs and Data-warehouses
  - HPC application file-I/O
  - Application checkpointing mechanisms for fault-tolerance
- Bottlenecks across the storage hierarchy: from local storage to parallel filesystem
- Contention in site-wide shared storage infrastructures
- Involvement of multiple network fabrics and protocols

State-of-Art in Storage Products

- Storage appliances
  - Fast and reliable interconnect as backplane (InfiniBand, Fibre Channel, SAS, etc.)
  - Common network connectivity interconnect as front-end (InfiniBand or Ethernet)

Company	Backplane Channel	Front-end Network
Oracle Exadata	InfiniBand	InfiniBand
EMC Symmetrix	Fibre Channel	Ethernet
Isilon	InfiniBand	Ethernet
Panasas	SAS	InfiniBand, Ethernet
Data Direct Networks	InfiniBand, SAS	InfiniBand

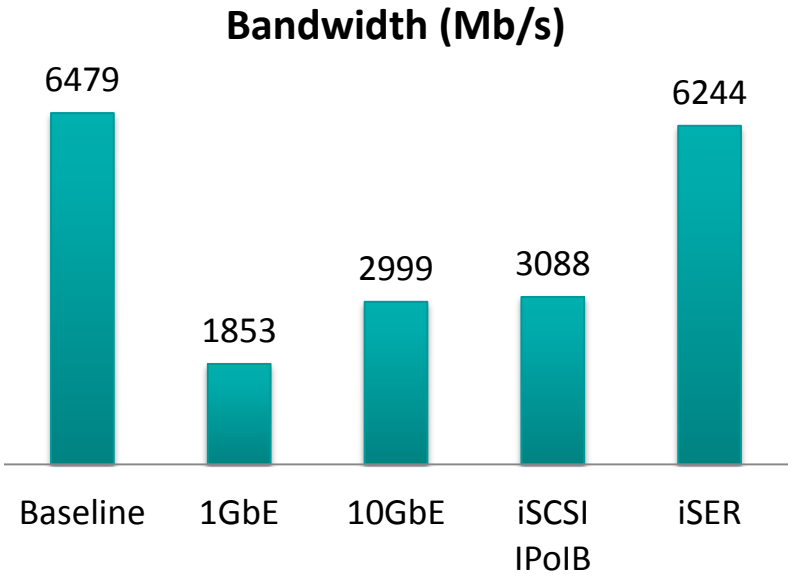
HSE in HPC Storage

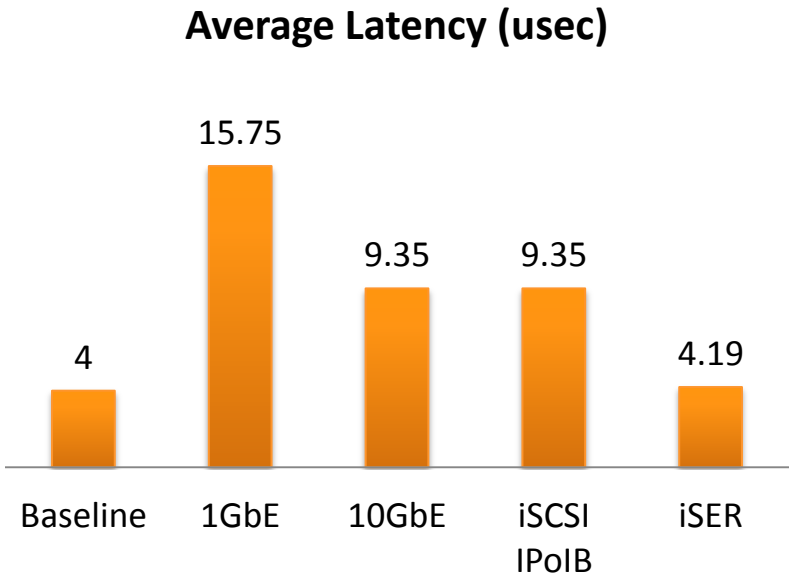
- Storage solutions that leverage already deployed and well-supported Ethernet fabrics
- HSE for NAS
  - IP-based storage consolidation and file-sharing
  - TCP/IP offload to Ethernet NICs
  - Virtual Interface technology achieved through Direct Access File System collaborative
- HSE for increasing fan-out to shared storage
  - Direct-attached storage proving to be expensive
  - Access to consolidated storage over IP network using iSCSI
  - Ethernet's distance and security features beneficial for storage appliances
- HSE for distributed data-center storage connectivity
  - Need for multi-location data storage for data centers fuels need for metro-storage
  - Ethernet technology reaches over 40Km distances without need for repeaters

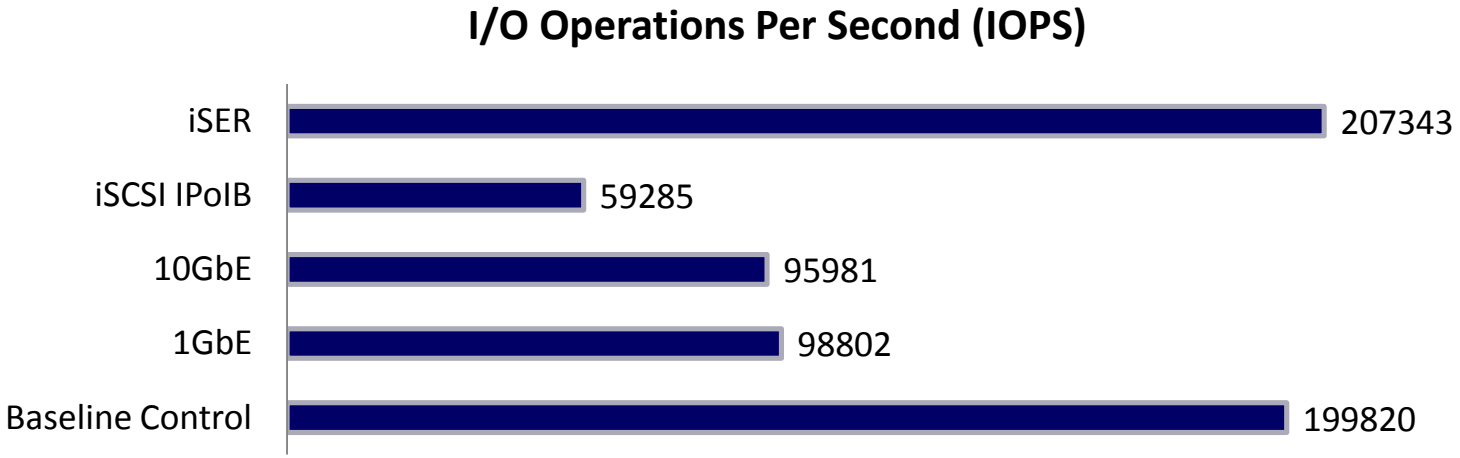
InfiniBand in HPC Storage

- Native IB storage efforts heavily driven by IB-based cluster deployments
- Supports block-level storage protocols
  - SCSI RDMA Protocol (SRP)
  - iSER (iSCSI Extension over RDMA)
  - Reliable Dataram Sockets (RDS)
- A vast amount of scalable distributed-filesystems that use IB transport
  - Lustre, PVFS, GPFS, Panasas, NFS… … …
- Follows the “One-Wire” model allowing the same interconnect to be used for both storage and application networking

State-of-Art in InfiniBand Storage

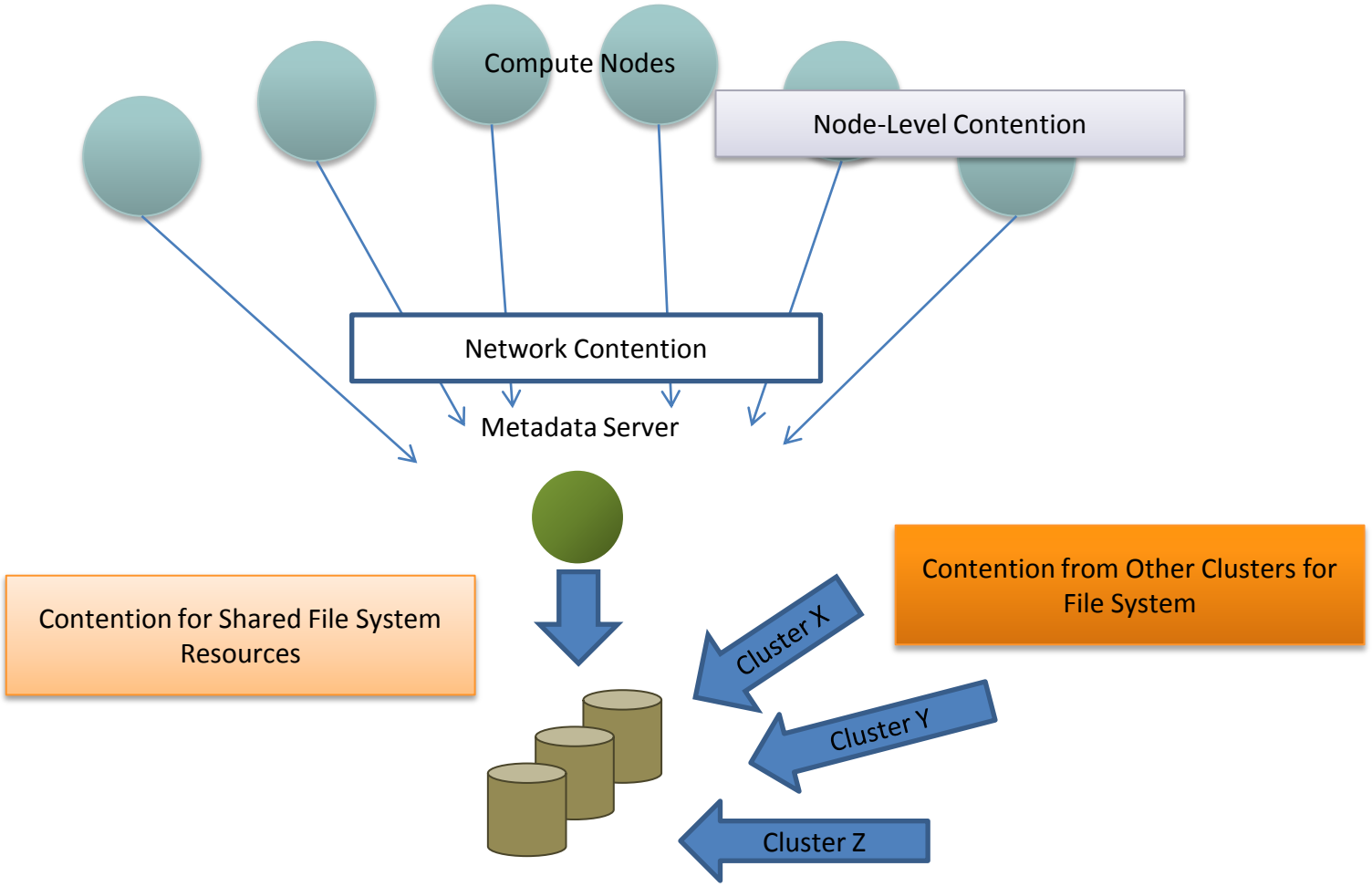






Courtesy: "Building a Scalable Storage with InfiniBand", Mellanox White Paper, 2012

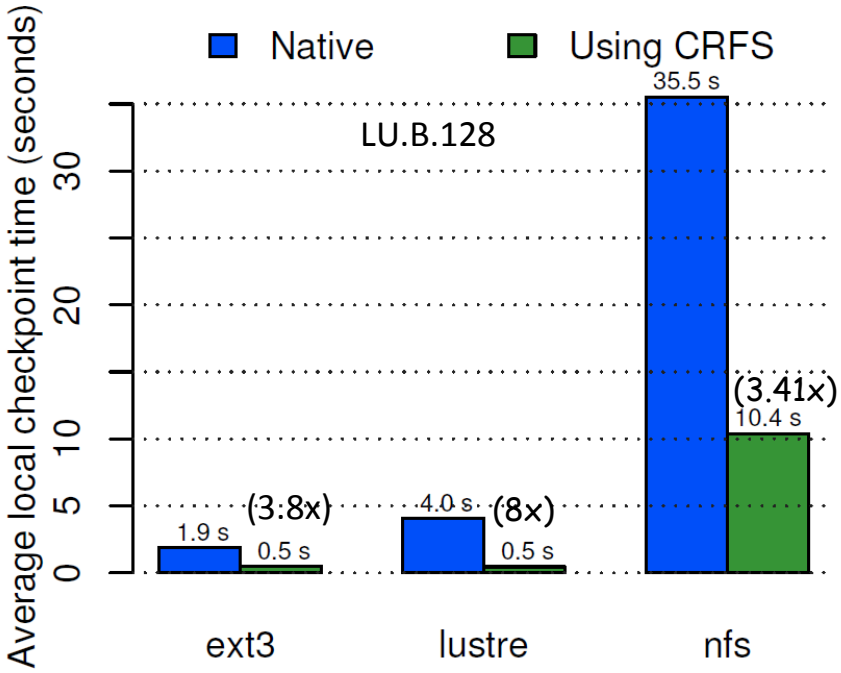
Issues with Traditional Parallel Filesystems



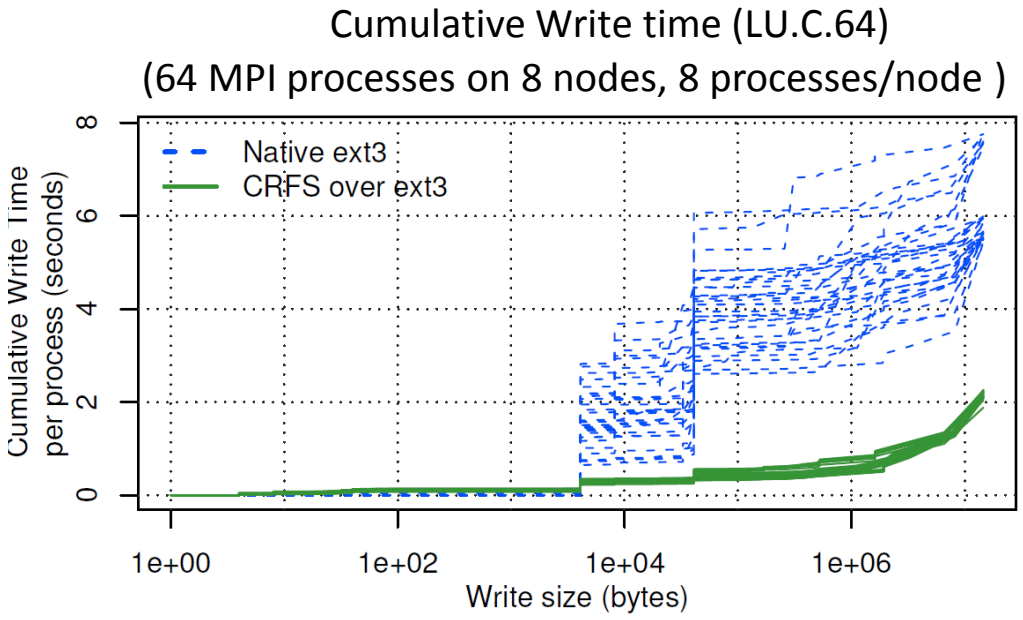
Checkpoint Writing Performance w/ Write-Aggregation

(CRFS: Write-Aggregation module within MVAPICH2)

- Simultaneous heavy-weight I/O from application processes cause intra-node contention
- Aggregating multiple write streams within a node reduces the contention at the kernel VFS
- Helps reduce application skew caused due to parallel I/O



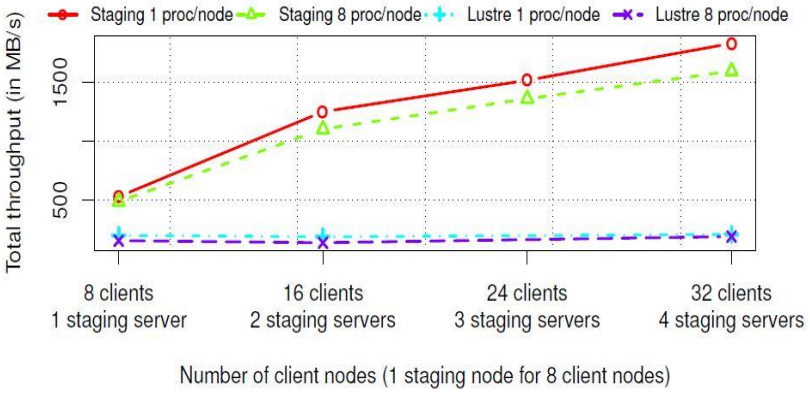
(128 MPI processes on 16 nodes, 8 processes/node)



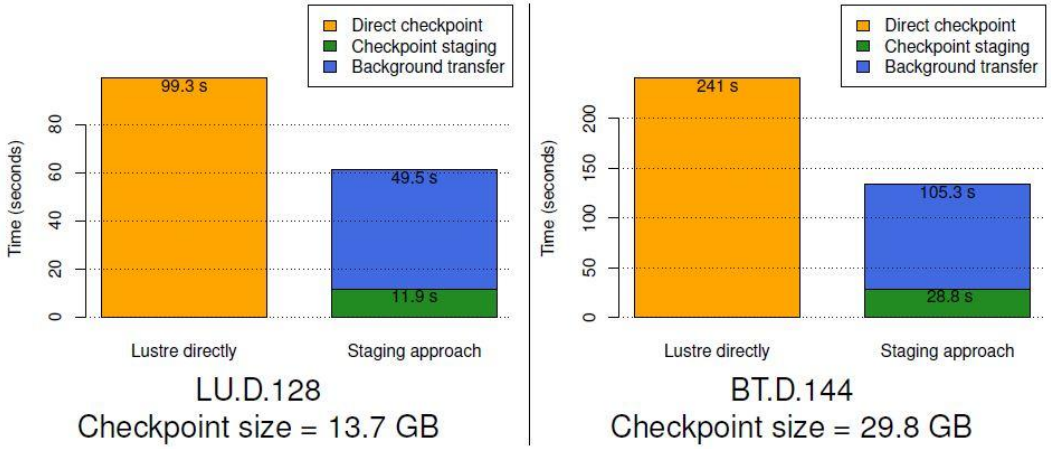
X. Ouyang, R. Rajachandrasekar, X. Besseron, H. Wang, J. Huang and D. K. Panda, CRFS: A Lightweight User-Level Filesystem for Generic Checkpoint/Restart, Int'l Conference on Parallel Processing (ICPP '11), Sept. 2011.

RDMA-Based Hierarchical Data-Staging Architecture

- Checkpoint/Restart systems cause heavy I/O contention at the Parallel Filesystem
- Hierarchical staging enables asynchronous I/O with the help of dedicated staging servers
- Can such staging architectures leverage high-speed networks like InfiniBand and fast-storage media like SSDs to reduce application-perturbation?



Throughput Scalability



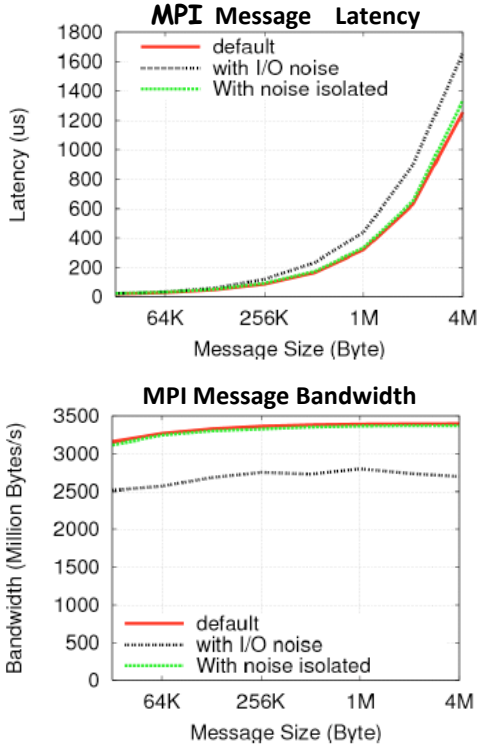
Application Checkpointing Time

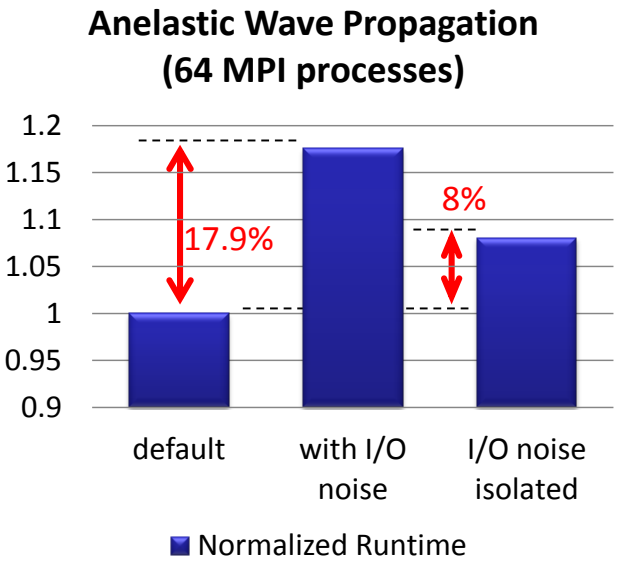
- **Architecture scales well with increasing number of 'Staging Groups'**
- **Total checkpointing protocol time reduced by a factor of 2**
- **Actual checkpoint-writing overhead reduced by a factor of 8**

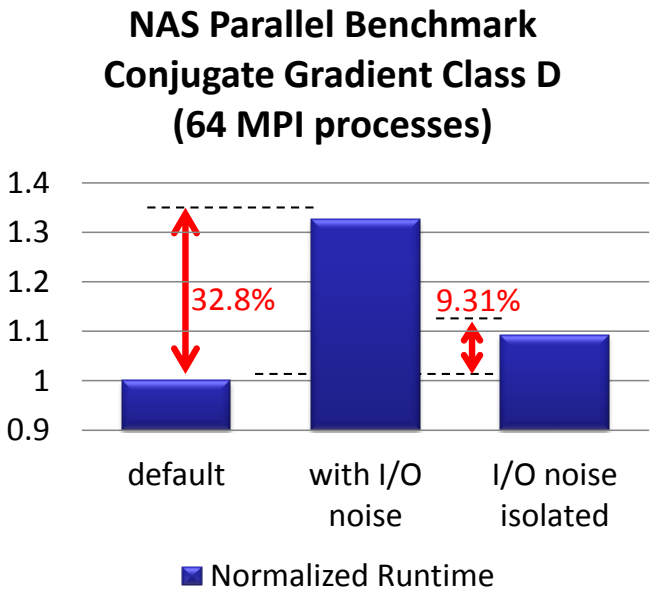
R. Rajachandrasekar, X. Ouyang, X. Besseron, V. Meshram and D. K. Panda, Can Checkpoint/Restart Mechanisms Benefit from Hierarchical Data Staging? Workshop on Resiliency in High Performance Computing in Clusters, Clouds, and Grids (Resilience '11), held in conjunction with EuroPar, Aug. 2011

Minimizing Network Contention w/ QoS-Aware Data-Staging

- Asynchronous I/O introduces contention for network-resources
- How should data be orchestrated in a data-staging architecture to eliminate such contention?
- Can the QoS capabilities provided by cutting-edge interconnect technologies be leveraged by parallel filesystems to minimize network contention?







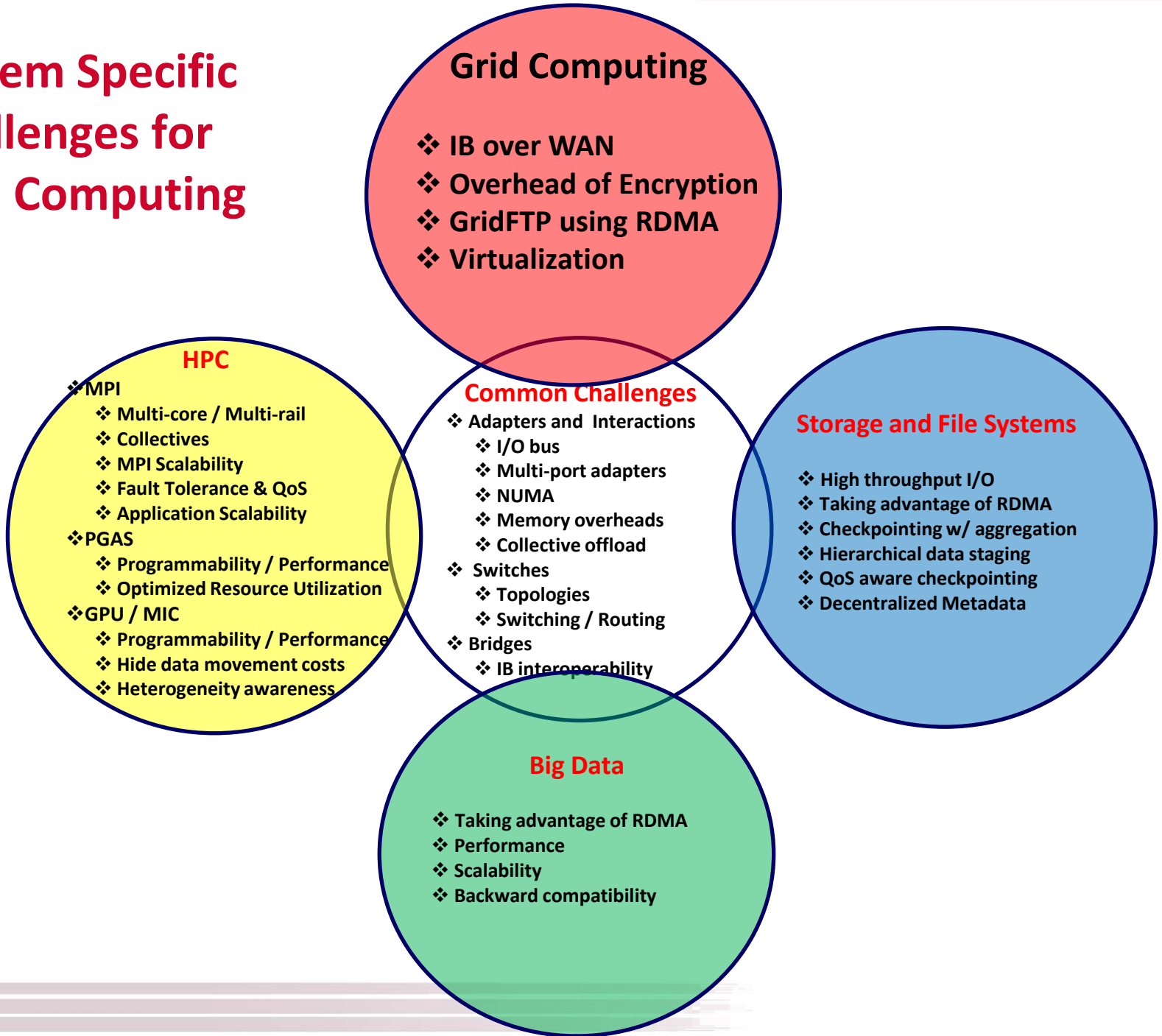
- **Reduces runtime overhead from 17.9% to 8% and from 32.8% to 9.31%, in case of AWP and NAS-CG applications respectively**

R. Rajachandrasekar, J. Jaswani, H. Subramoni and D. K. Panda, Minimizing Network Contention in InfiniBand Clusters with a QoS-Aware Data-Staging Framework, IEEE Cluster, Sept. 2012

Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- **System Specific Challenges and Case Studies**
  - HPC (MPI, PGAS and GPU/MIC Computing)
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - Storage and File Systems
  - **Grid Computing**
- Open Fabrics Software Stack and RDMA Programming
- Network Management Infrastructure and Tools
- Conclusions and Final Q&A

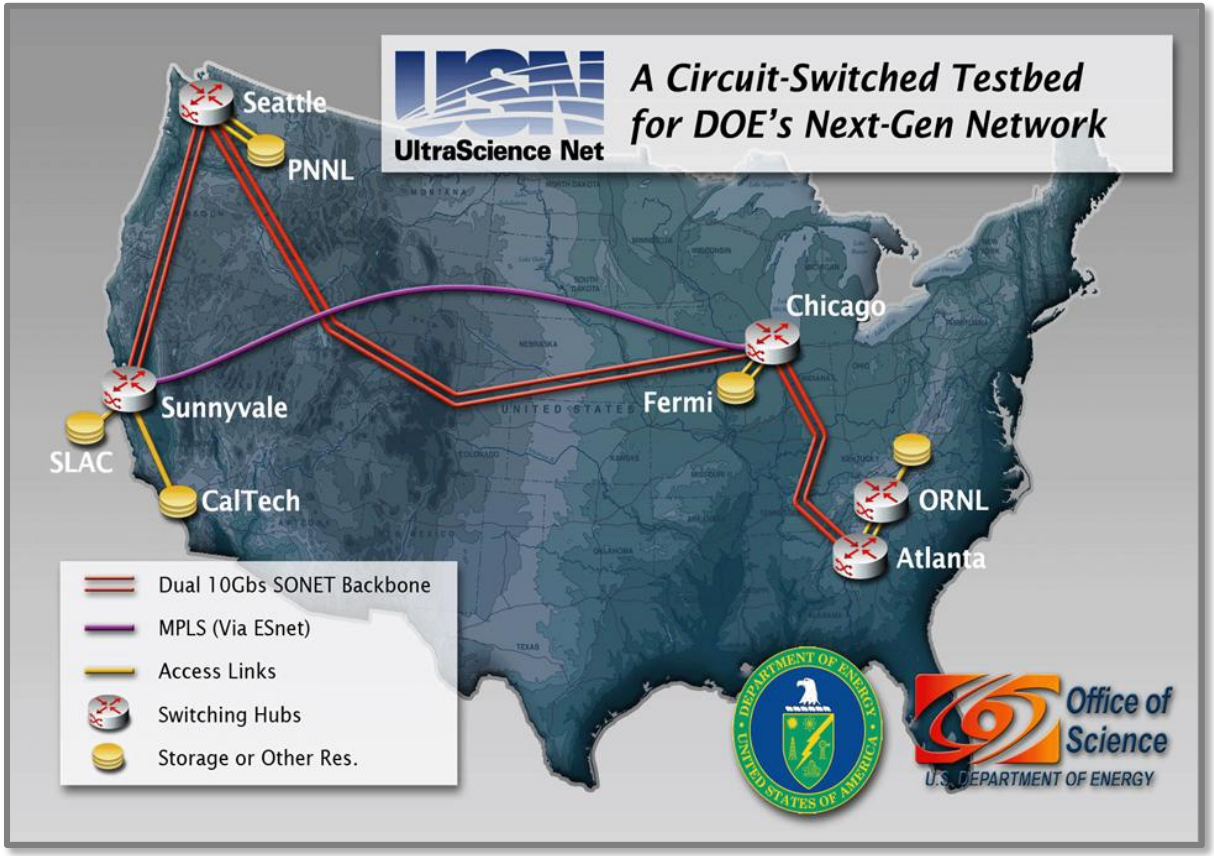
System Specific Challenges for Grid Computing



IB on the WAN

- Option 1: Layer-1 Optical networks
  - IB standard specifies link, network and transport layers
  - Can use any layer-1 (though the standard says copper and optical)
- Option 2: Link Layer Conversion Techniques
  - InfiniBand-to-Ethernet conversion at the link layer: switches available from multiple companies (e.g., Obsidian, Bay Microsystems, Mellanox)
    - Technically, it's not conversion; it's just tunneling (L2TP)
  - InfiniBand's network layer is IPv6 compliant

UltraScience Net: Experimental Research Network Testbed



Since 2005

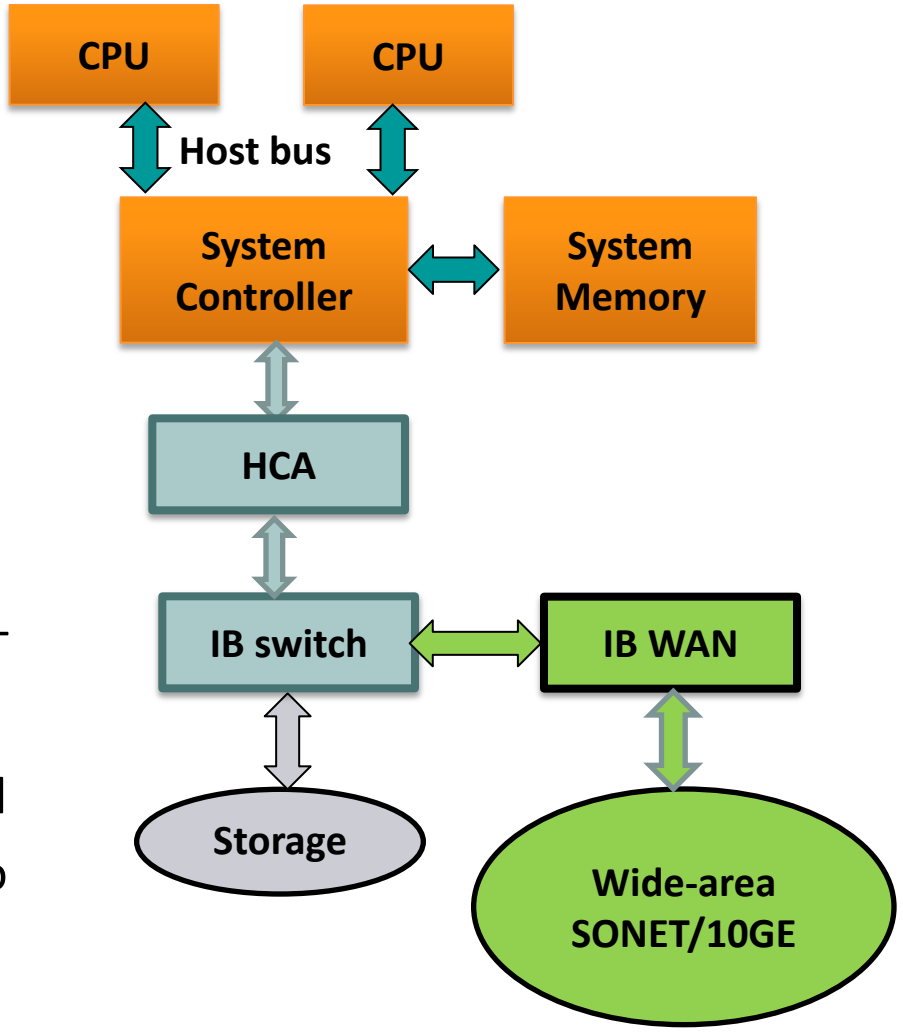
This and the following IB WAN slides are courtesy Dr. Nagi Rao (ORNL)

Features

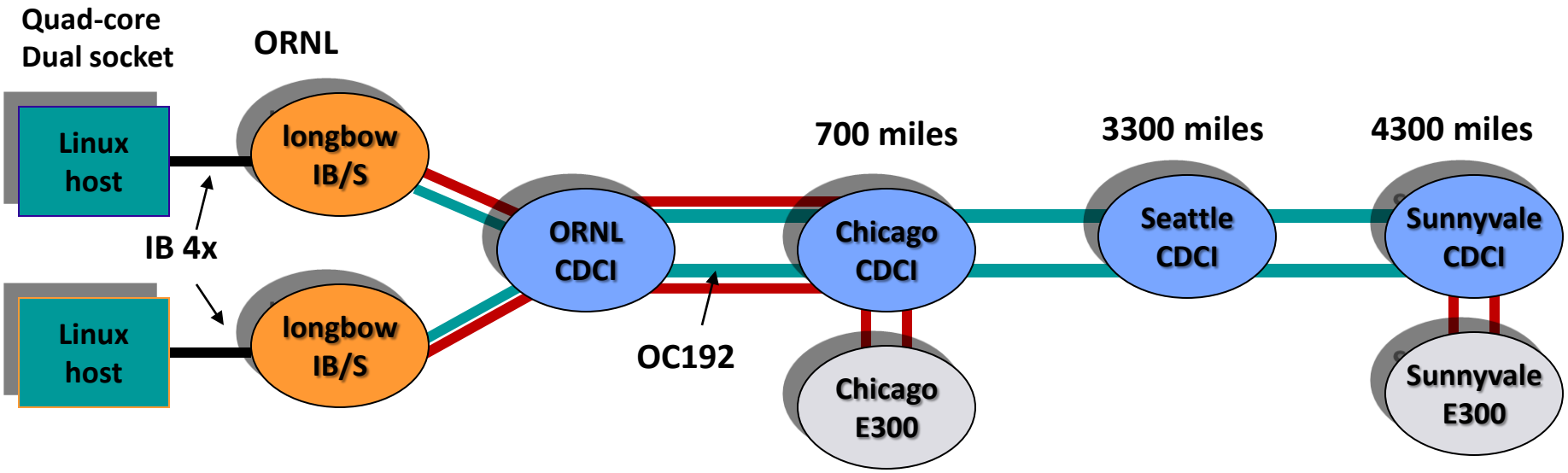
- End-to-end guaranteed bandwidth channels
- Dynamic, in-advance, reservation and provisioning of fractional/full lambdas
- Secure control-plane for signaling
- Peering with ESnet, National Science Foundation CHEETAH, and other networks

IB-WAN Connectivity with Obsidian Switches

- Supports SONET OC-192 or 10GE LAN-PHY/WAN-PHY
- Idea is to make remote storage "appear" local
- IB-WAN switch does frame conversion
  - IB standard allows per-hop credit-based flow control
  - IB-WAN switch uses large internal buffers to allow enough credits to fill the wire



IB over 10GE LAN-PHY and WAN-PHY



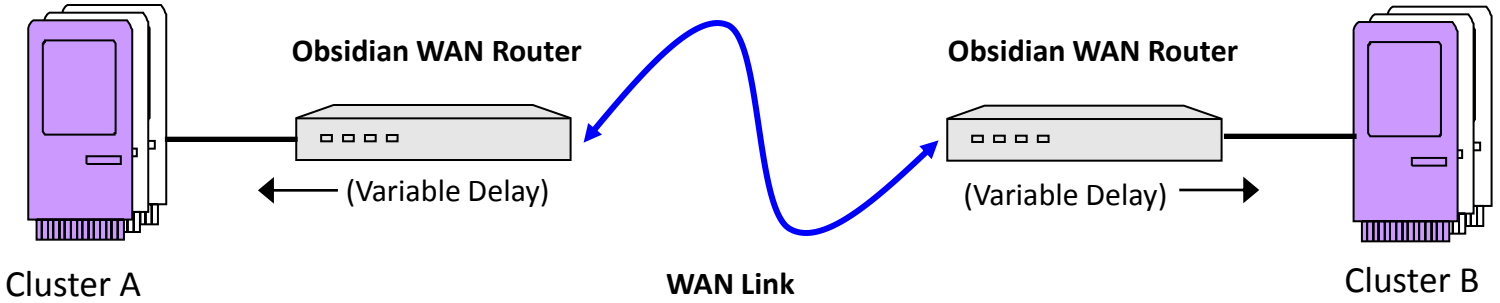
ORNL loop -0.2 mile: 7.5Gbps

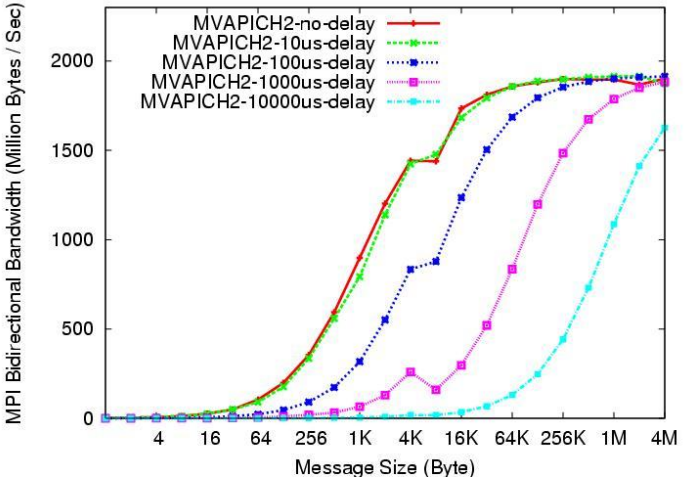
ORNL-Chicago loop – 1400 miles: 7.49Gbps

ORNL- Chicago - Seattle loop – 6600 miles: 7.39Gbps

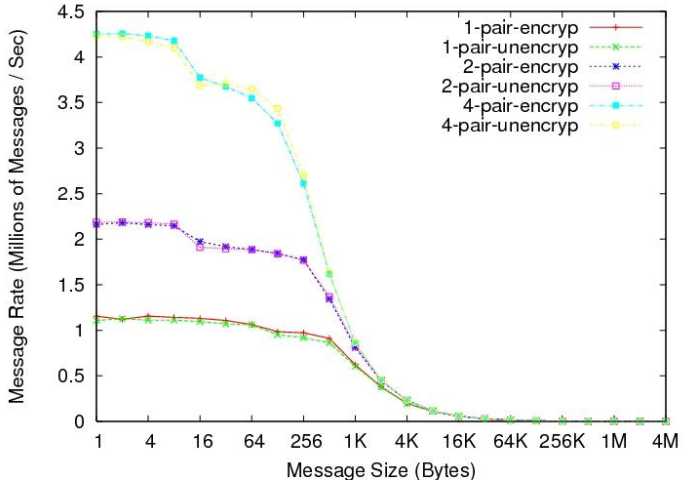
ORNL – Chicago – Seattle - Sunnyvale loop – 8600 miles: 7.36Gbps

MPI over IB-WAN and Encryption Overhead: Obsidian Routers





MPI Bidirectional Bandwidth



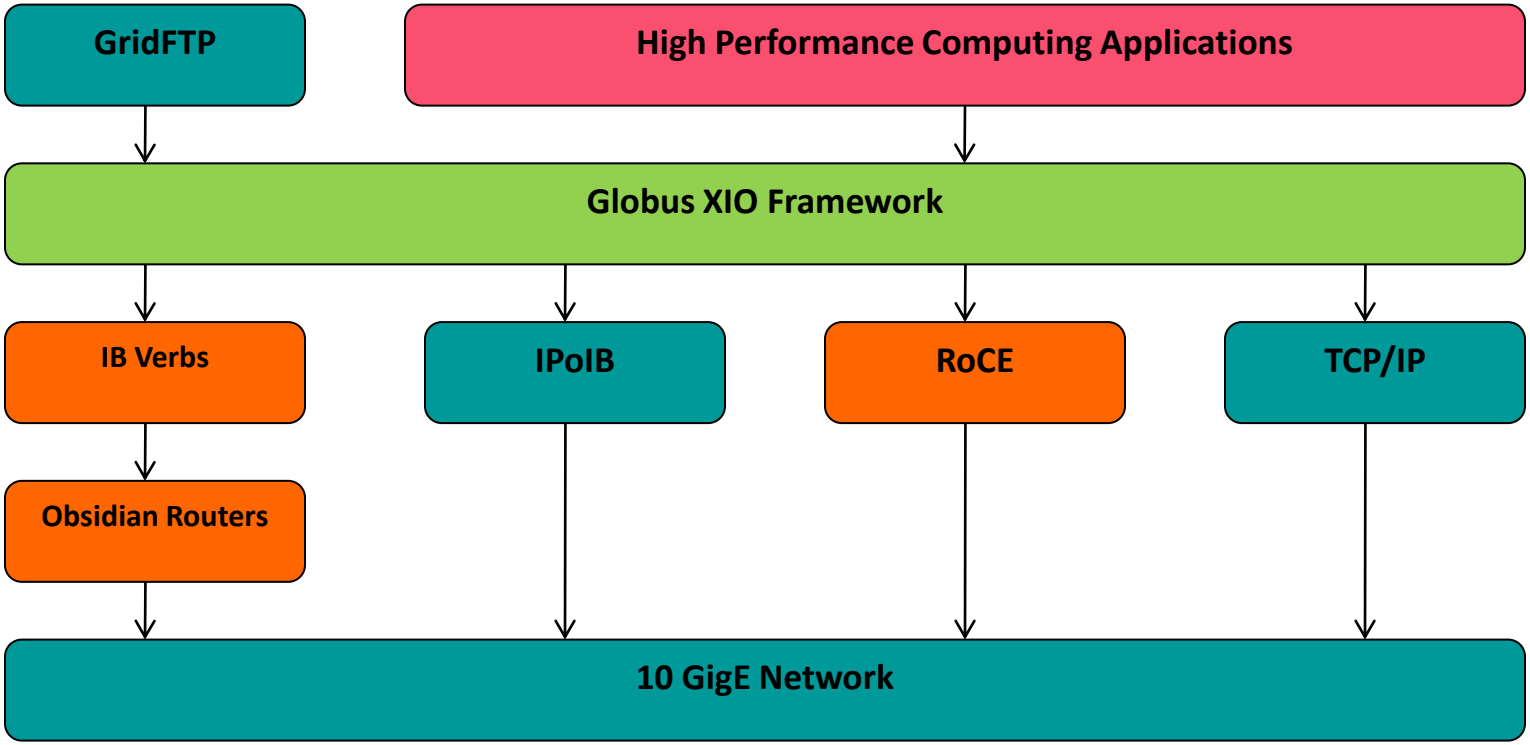
Impact of Encryption on Message Rate (Delay 0 ms)

Delay (us)	Distance (km)
10	2
100	20
1000	200
10000	2000

Hardware encryption has no impact on performance for less communicating streams

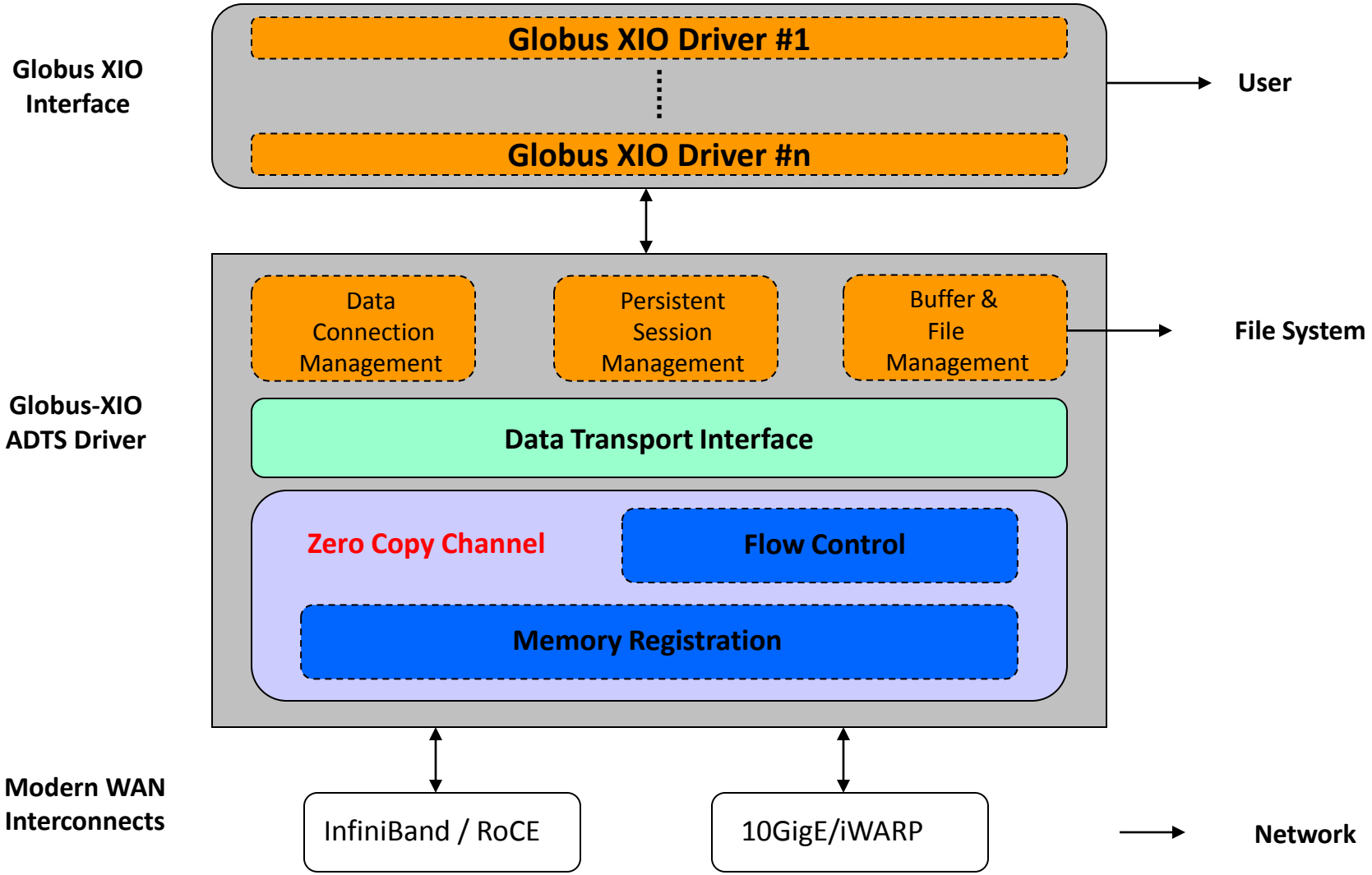
S. Narravula, H. Subramoni, P. Lai, R. Noronha and D. K. Panda, Performance of HPC Middleware over InfiniBand WAN, Int'l Conference on Parallel Processing (ICPP '08), September 2008.

Communication Options in Grid



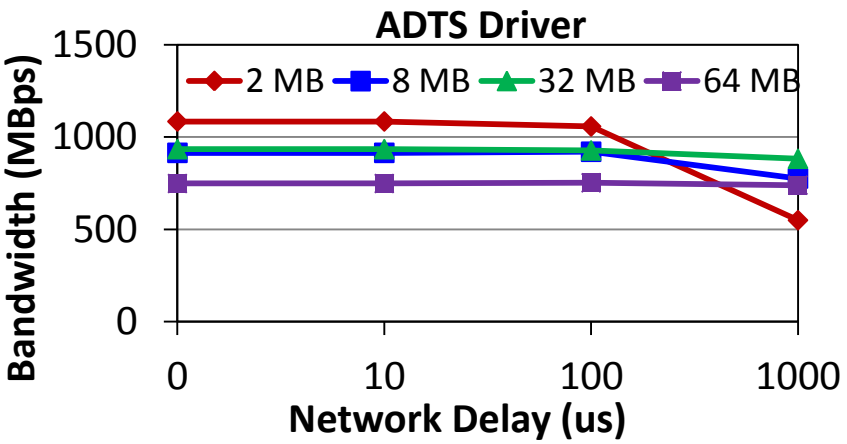
- Multiple options exist to perform data transfer on Grid
- Globus-XIO framework currently does not support IB natively
- We create the Globus-XIO ADTS driver and add native IB support to GridFTP

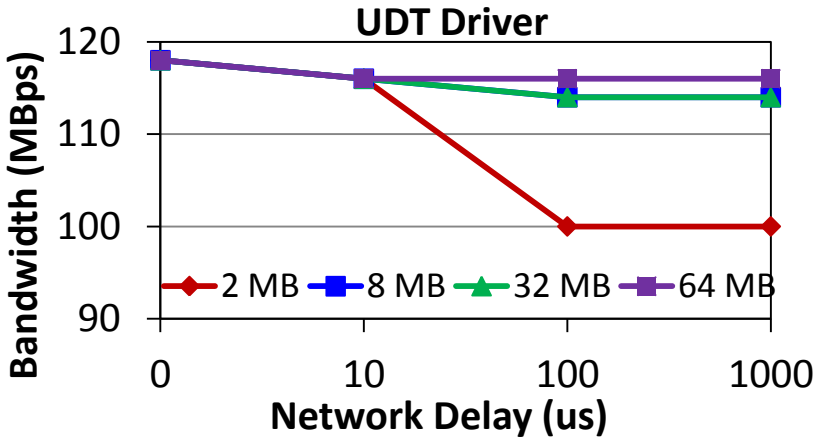
Globus-XIO Framework with ADTS Driver



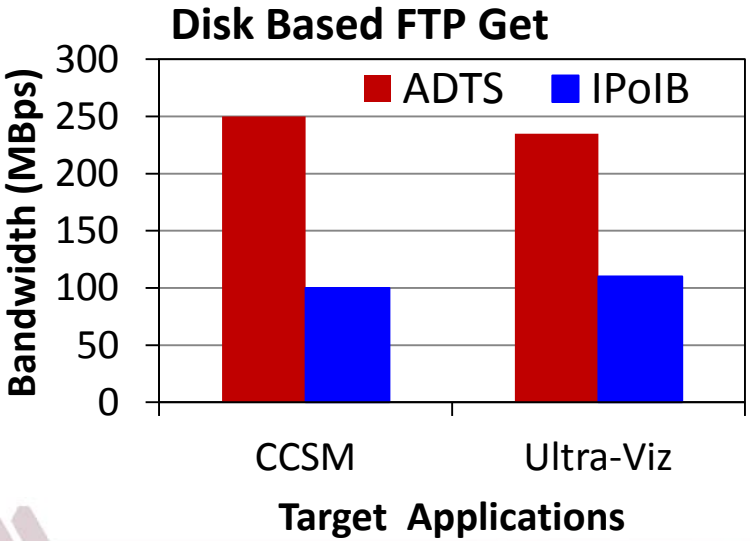
Performance Comparison of ADTS & UDT Drivers

In memory data transfer performance of ADTS & UDT drivers for different buffer sizes





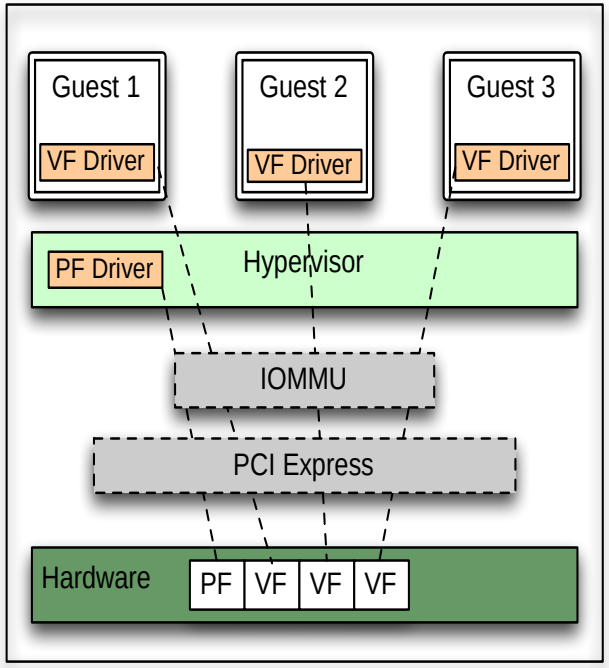
ADTS based implementation is able to saturate the link bandwidth



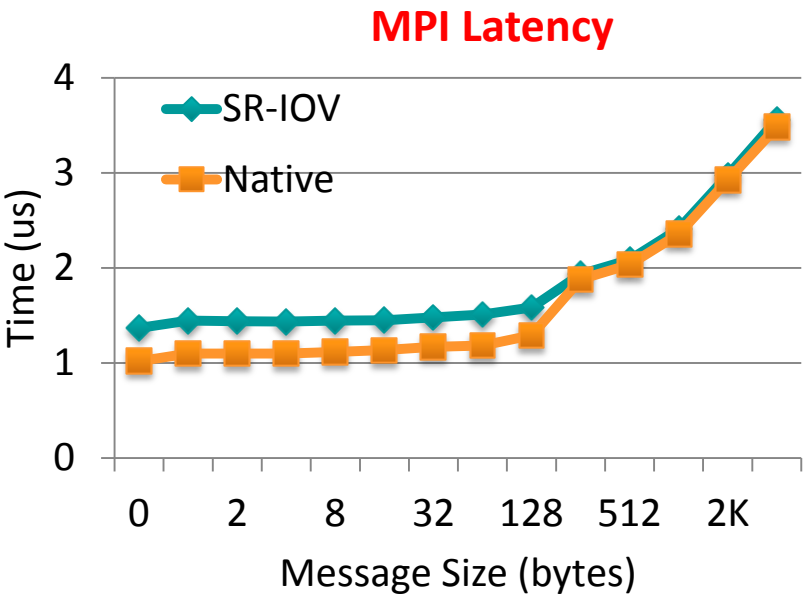
- Community Climate System Model (CCSM)
  - Part of Earth System Grid Project
  - Transfers 160 TB in chunks of 256 MB
  - Network latency - 30 ms
- Ultra-Scale Visualization (Ultra-Viz)
  - Transfers files of size 2.6 GB
  - Network latency - 80 ms
- **The ADTS driver out performs the UDT driver (IPoIB) by more than 100%**

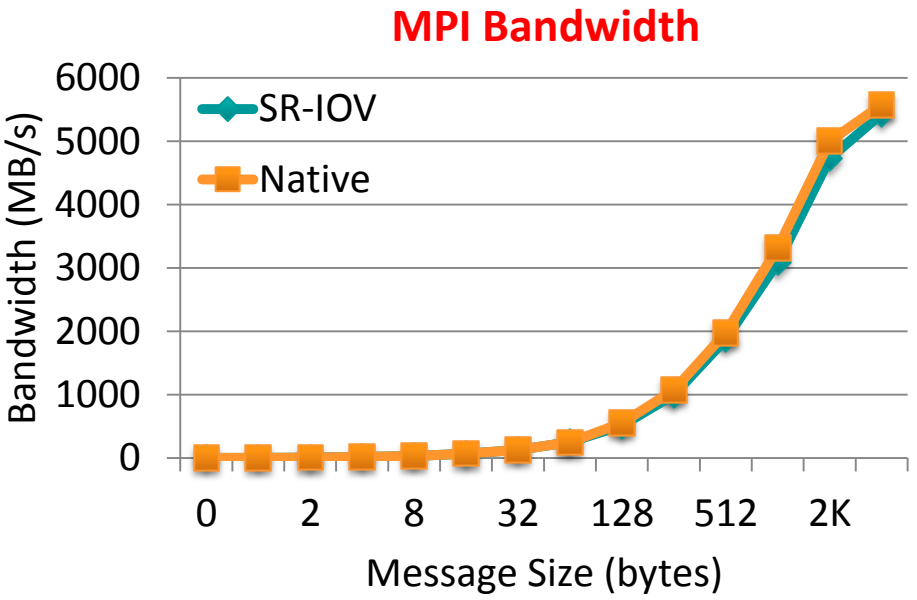
Single Root I/O Virtualization (SR-IOV)

- SR-IOV specifies native I/O Virtualization capabilities in the PCI Express (PCIe) adapters
- Physical Function (PF) presented as multiple Virtual Functions (VFs)
- Virtual device can be dedicated to a single VM through PCI pass-through
- VM can directly access the corresponding VF



MPI Level Performance

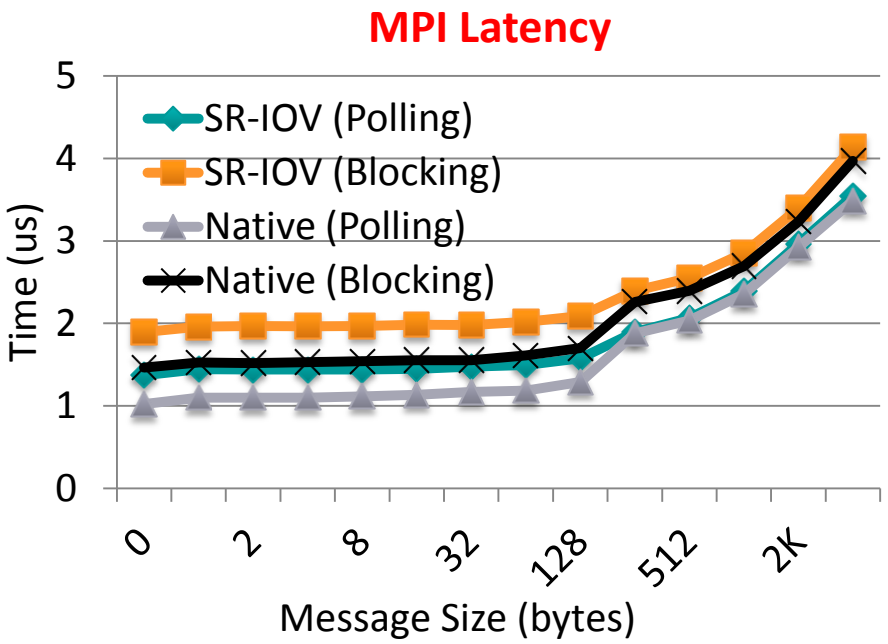


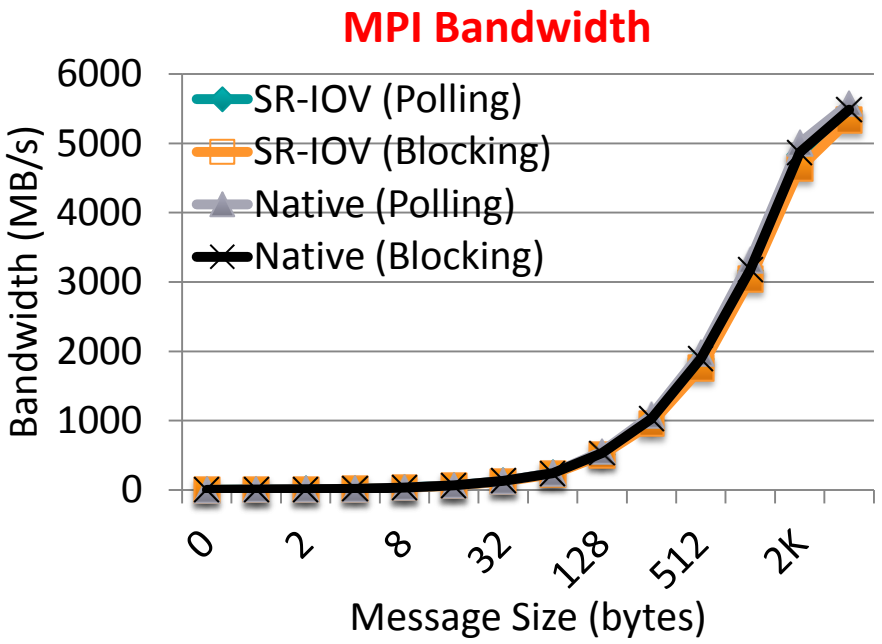


- Performance evaluations using OSU MPI benchmarks
- Used MVAPICH2-1.9a2 as the MPI Library
- Comparable performance for Native and SR-IOV
  - 1.02us (native) and 1.39us (SR-IOV) for one byte message size
- MVAPICH2 uses 'RDMA-FastPath' optimization for small messages
  - Similar characteristics as that of RDMA write

J. Jose, M. Li, X. Lu, K. Kandalla, M. Arnold and D. K. Panda, SR-IOV Support for Virtualization on InfiniBand Clusters: Early Experience, Int'l Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2013), May 2013.

Impact of Polling and Blocking Modes on Performance





- MVAPICH2 employs a hybrid scheme in blocking configuration
  - Polls for a specific number of times, then switches to blocking mode
- Polling Mode: 1.02us (native) and 1.39us (SR-IOV) for one byte message size
- Blocking Mode: 1.46us (native) and 1.89us (SR-IOV) for one byte message size
- Similar performance for MPI bandwidth

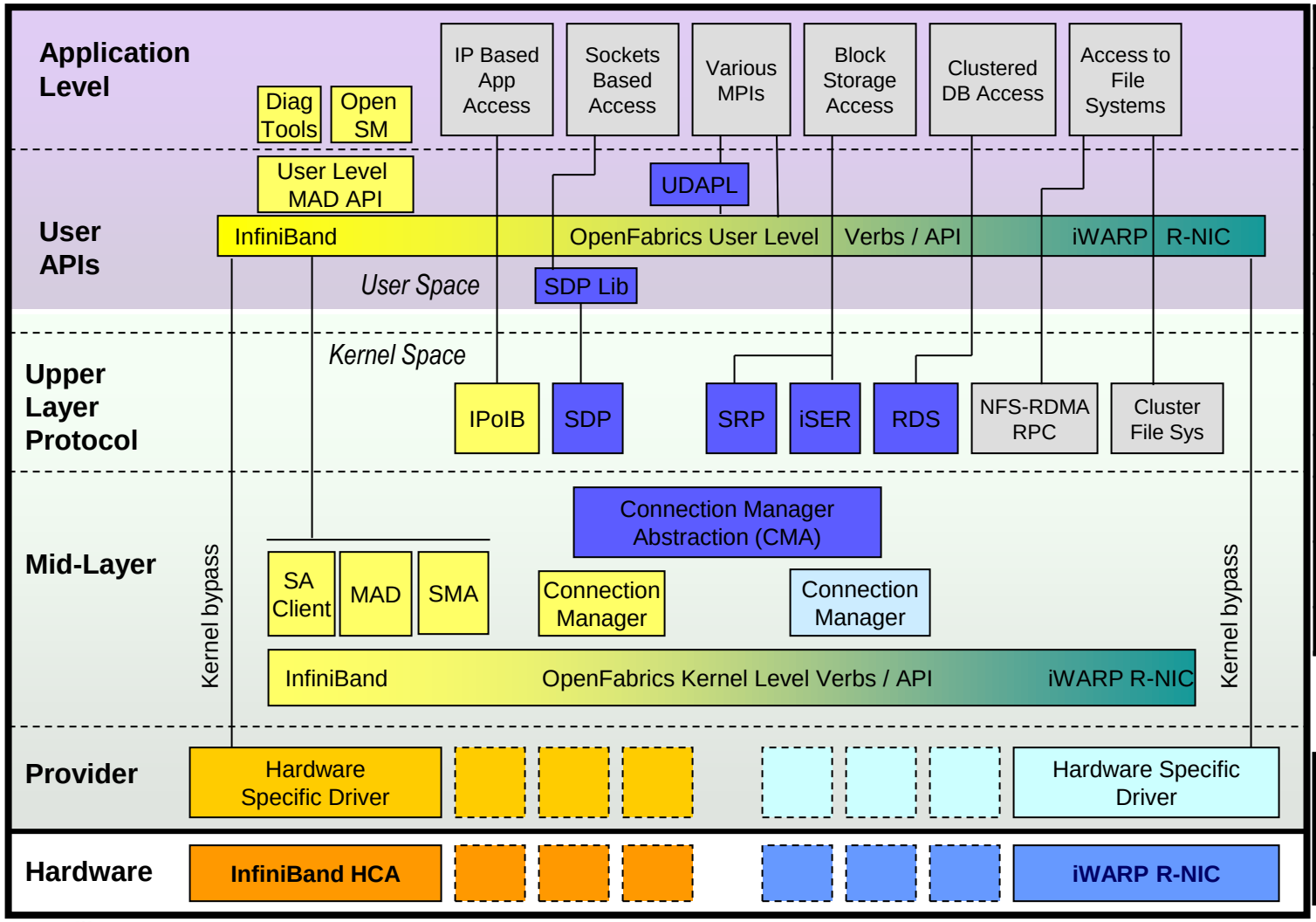
Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- System Specific Challenges and Case Studies
  - HPC (MPI, PGAS and GPU/MIC Computing)
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - Storage and File Systems
  - Grid Computing
- **Open Fabrics Software Stack and RDMA Programming**
- Network Management Infrastructure and Tools
- Conclusions and Final Q&A

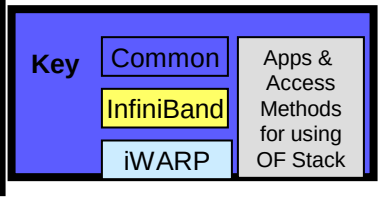
Software Convergence with OpenFabrics

- Open source organization (formerly OpenIB)
  - www.openfabrics.org
- Incorporates both IB and iWARP in a unified manner
  - Support for Linux and Windows
- Users can download the entire stack and run
  - Latest release is OFED 3.5
    - New naming convention to get aligned with Linux Kernel Development

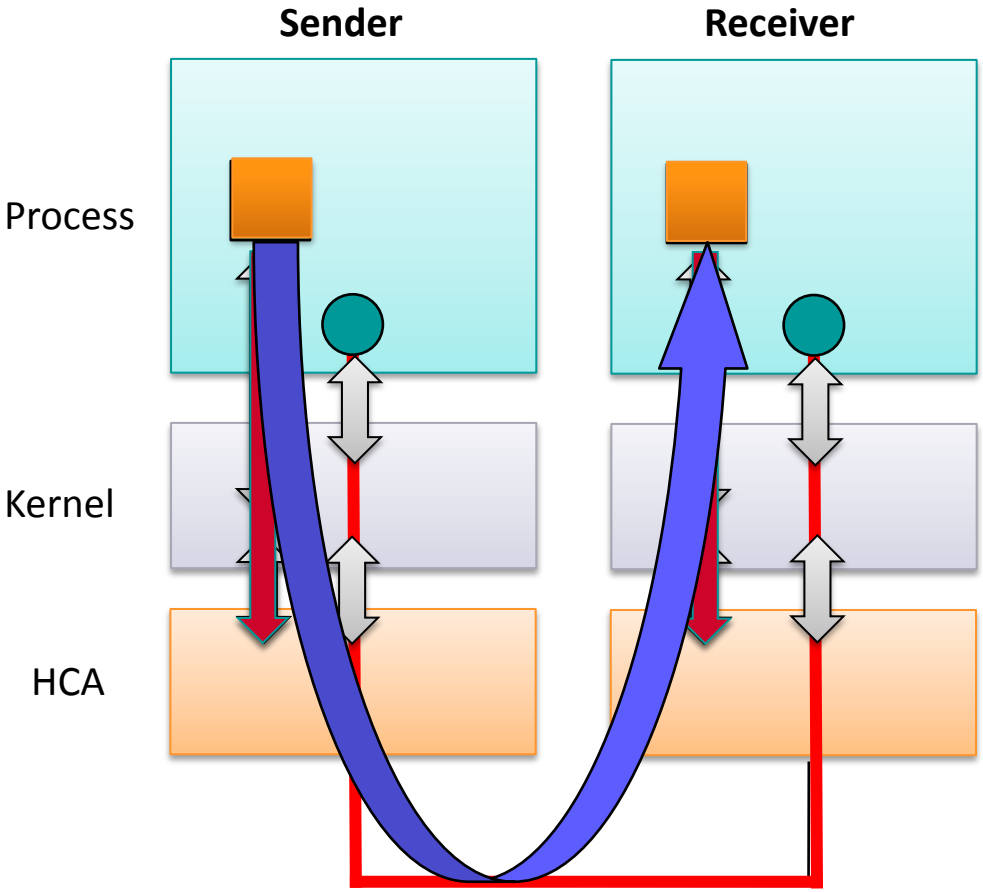
OpenFabrics Software Stack



SA	Subnet Administrator
MAD	Management Datagram
SMA	Subnet Manager Agent
PMA	Performance Manager Agent
IPoIB	IP over InfiniBand
SDP	Sockets Direct Protocol
SRP	SCSI RDMA Protocol (Initiator)
iSER	iSCSI RDMA Protocol (Initiator)
RDS	Reliable Datagram Service
UDAPL	User Direct Access Programming Lib
HCA	Host Channel Adapter
R-NIC	RDMA NIC



Programming with OpenFabrics



Sample Steps

1. Create QPs (endpoints)
2. Register memory for sending and receiving
3. Send
   - Channel semantics
     - Post receive
     - Post send
   - RDMA semantics

Verbs Steps

- Open HCA and create QPs to end nodes
  - Can be done with connection managers (rdma_cm or ibcm) or directly through verbs with out-of-band communication
- Register memory

```
ibv_mr * mrhandle = ibv_reg_mr(pd, *buffer, len,
   IBV_ACCESS_LOCAL_WRITE | IBV_ACCESS_REMOTE_WRITE |
   IBV_ACCESS_REMOTE_READ
```

Permissions can be set as needed

Verbs: Post Receive

- Prepare and post receive descriptor (channel semantics)

```
struct ibv_recv_wr *bad_wr;
struct ibv_recv_wr rr;
struct ibv_sge sg_entry;

rr.next = NULL;
rr.wr_id = 0;
rr.num_sge = 1;
rr.sg_list = &(sg_entry);
sg_entry.addr = (uintptr_t) buf;    /* local buffer address */
sg_entry.length = len;
sg_entry.lkey = mr_handle->lkey;    /* memory handle */

ret = ibv_post_recv(qp, &rr, &bad_wr);  /* post to QP  */
ret = ibv_post_srq_recv(srq, &rr, &bad_wr);  /* post to SRQ */
```

Verbs: Post Send

- Prepare and post send descriptor (channel semantics)

```
struct ibv_send_wr *bad_wr;
struct ibv_send_wr sr;
struct ibv_sge sg_entry;

sr.next = NULL;
sr.opcode = IBV_WR_SEND;
sr.wr_id = 0;
sr.num_sge = 1;
if (len < max_inline_size) {
   sr.send_flags = IBV_SEND_SIGNALED | IBV_SEND_INLINE;
} else {
   sr.send_flags = IBV_SEND_SIGNALED;
}
sr.sg_list = &(sg_entry);
sg_entry.addr = (uintptr_t) buf;
sg_entry.length = len;
sg_entry.lkey = mr_handle->lkey;

ret = ibv_post_send(qp, &sr, &bad_wr);
```

Verbs: Post RDMA Write

- Prepare and post RDMA write (memory semantics)

```
struct ibv_send_wr *bad_wr; struct ibv_send_wr sr;
struct ibv_sge sg_entry;

sr.next = NULL;
sr.opcode = IBV_WR_RDMA_WRITE;  /* set type to RDMA Write */
sr.wr_id = 0;
sr.num_sge = 1;
sr.send_flags = IBV_SEND_SIGNALED;
sr.wr.rdma.remote_addr = remote_addr; /* remote virtual addr. */
sr.wr.rdma.rkey = rkey;               /* from remote node */
sr.sg_list = &(sg_entry);
sg_entry.addr = buf;                  /* local buffer */
sg_entry.length = len;
sg_entry.lkey = mr_handle->lkey;

ret = ibv_post_send(qp, &sr, &bad_wr);
```

Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- System Specific Challenges and Case Studies
  - HPC (MPI, PGAS and GPU/MIC Computing)
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - Storage and File Systems
  - Grid Computing
- Open Fabrics Software Stack and RDMA Programming
- **Network Management Infrastructure and Tools**
- Conclusions and Final Q&A

Network Management Infrastructure and Tools

- **Management Infrastructure**
  - **Subnet Manager**
  - Diagnostic tools
    - System Discovery Tools
    - System Health Monitoring Tools
    - System Performance Monitoring Tools
  - Fabric management tools

Concepts in IB Management

- Agents
  - Processes or hardware units running on each adapter, switch, router (everything on the network)
  - Provide capability to query and set parameters
- Managers
  - Make high-level decisions and implement it on the network fabric using the agents
- Messaging schemes
  - Used for interactions between the manager and agents (or between agents)
- Messages

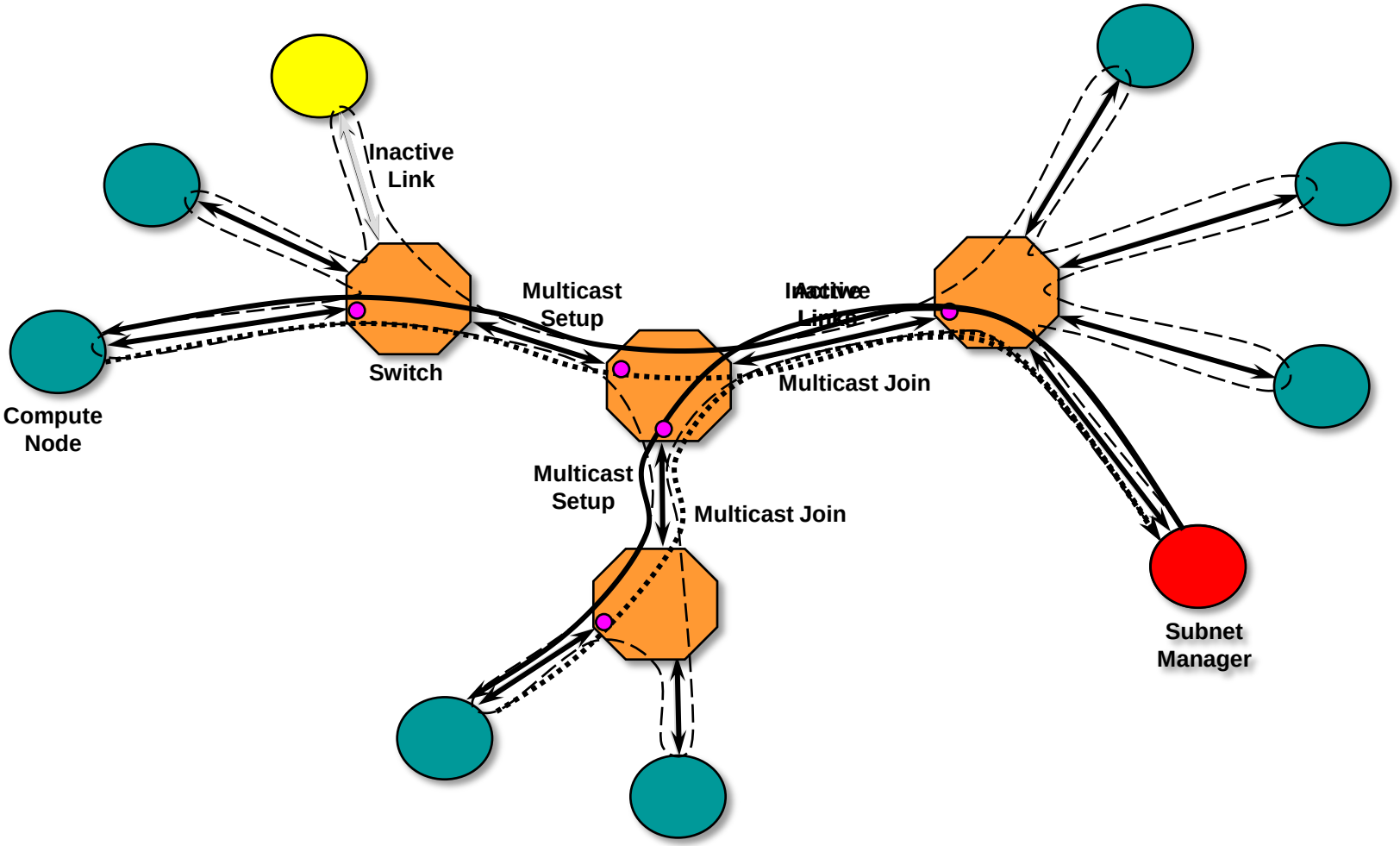
InfiniBand Management

- All IB management happens using packets called as Management Datagrams
  - Popularly referred to as “MAD packets”
- Four major classes of management mechanisms
  - Subnet Management
  - Subnet Administration
  - Communication Management
  - General Services

Subnet Management & Administration

- Consists of at least one subnet manager (SM) and several subnet management agents (SMAs)
  - Each adapter, switch, router has an agent running
  - Communication between the SM and agents or between agents happens using MAD packets called as Subnet Management Packets (SMPs)
- SM's responsibilities include:
  - Discovering the physical topology of the subnet
  - Assigning LIDs to the end nodes, switches and routers
  - Populating switches and routers with routing paths
  - Subnet sweeps to discover topology changes

Subnet Manager



Subnet Manager Sweep Behavior

- SM can be configured to sweep once or continuously
- On the first sweep:
  - All ports are assigned LIDs on the first sweep
  - All routes are setup on the switches
- On consequent sweeps:
  - If there has been any change to the topology, appropriate routes are updated
  - If DLID X is down, packet not sent all the way
    - First hop will not have a forwarding entry for LID X
- Sweep time configured by the system administrator
  - Cannot be too high or too low

Subnet Manager Scalability Issues

- Single subnet manager has issues on large systems
  - Performance and overhead of scanning
    - Hardware implementations on switches are faster, but will work only for small systems (memory usage)
    - Software implementations are more popular (OpenSM)
  - Multi-SM models
    - Two benefits: fault tolerance (if one SM dies) and scalability (different SMs can handle different portions of the network)
    - Current SMs only provide a fault-tolerance model
    - Network subsetting is still be investigated
- Asynchronous events specified to improve scalability
  - E.g., TRAPS are events sent by an agent to the SM when a link goes down

Multicast Group Management

- Creation, joining/leaving, deleting multicast groups occur as SA requests
  - The requesting node sends a request to a SA
  - The SA sends MAD packets to SMAs on the switches to setup routes for the multicast packets
    - Each switch contains information on which ports to forward the multicast packet to
- Multicast itself does not go through the subnet manager
  - Only the setup and teardown goes through the SM

General Services

- Several general service management features provided by the standard
  - Performance Management
    - Several required and optional performance counters
    - Flow control counters, RNR counters, Number of sent and received packets
  - Hardware Management
    - Baseboard Management
    - Device Management
    - SNMP Tunneling
    - Vendor Specific
    - Application Specific

Network Management Infrastructure and Tools

- **Management Infrastructure**
  - Subnet Manager
  - **Diagnostic tools**
    - **System Discovery Tools**
    - System Health Monitoring Tools
    - System Performance Monitoring Tools
  - Fabric management tools

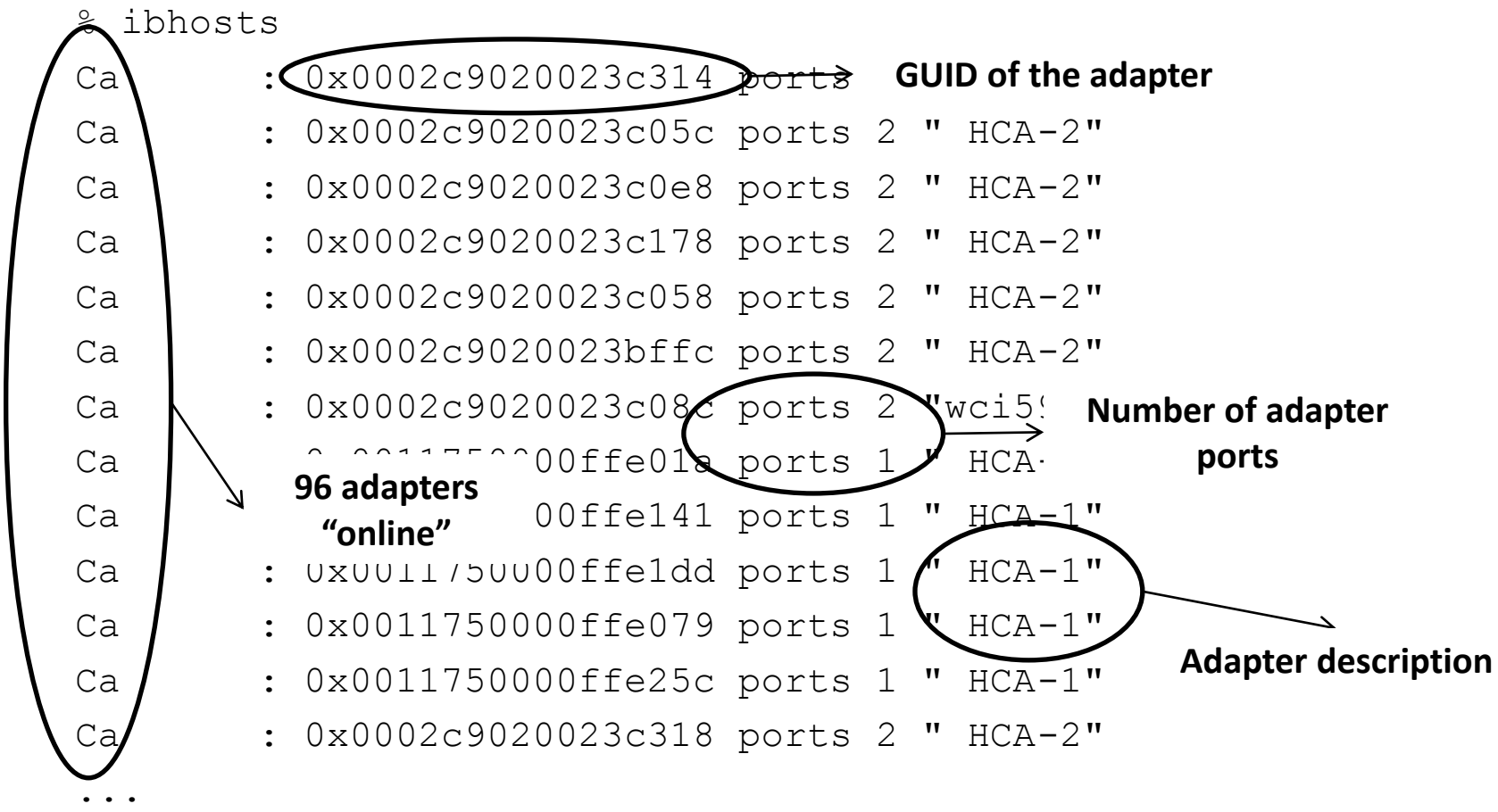
Tools to Analyze InfiniBand Networks

- Different types of tools exist:
  - High-level tools that internally talk to the subnet manager using management datagrams
  - Each hardware device exposes a few mandatory counters and a number of optional (sometimes vendor-specific) counters
- Possible to write your own tools based on the management datagram interface
  - Several vendors provide such IB management tools

Network Discovery Tools

- Starting with almost no knowledge about the system, we can identify several details of the network configuration
  - Example tools include:
    - ibstatus: shows adapter status
    - smpquery: SMP query tool
    - perfquery: reports performance/error counters of a port
    - ibportstate: shows status of IB port, enable/disable port
    - ibhosts: finds all the network adapters in the system
    - ibswitches: finds all the network switches in the system
    - ibnetdiscover: finds the connectivity between the ports
    - … and many others exist
  - Possible to write your own tools based on the management datagram interface
    - Several vendors provide such IB management tools

Discovering Network Adapters



Network Adapter Classification

```
% ibnetdiscover -H /* Some parts snipped out */
Ca      : ports 2 devid 0x6282 vendid 0x2c9 " HCA-2"
Ca      : ports 2 devid 0x6282 vendid 0x2c9 " HCA-2"
Ca      : ports 2 devid 0x6282 vendid 0x2c9 " HCA-2"
Ca      : ports 2 devid 0x6282 vendid 0x2c9 " HCA
Ca      : ports 2 devid 0x634a vendid 0x2c9 " HCA
Ca      : ports 2 devid 0x634a vendid 0x2c9 " HCA-1"
Ca      : ports 2 devid 0x634a vendid 0x2c9 " HCA-1"
Ca      : ports 2 devid 0x10 vendid 0x1fc1 " HCA-1"
Ca      : ports 1 devid 0x10 vendid 0x1fc1 " HCA
Ca      : ports 1 devid 0x10 vendid 0x1fc1 " HCA
Ca      : ports 1 devid 0x10 vendid 0x1fc1 " HCA-1"
...
```

Device ID

Vendor ID

InfiniHost III adapters

Mellanox adapters

ConnectX adapters

Qlogic adapters

Discovering Network Switches

```
% ibswitches
Switch  : 0x0002c90200424400 ports 36
"MF0;ibswitch:MTS3610/L11/U1" base port 0 lid 2 lmc 0
Switch  : 0x0002c90200424410 ports 36
"MF0;ibswitch:MTS3                    se port 0 lid 14 lmc 0
Switch  : 0x0002c90                   ts 36
"MF0;ibswitch:MTS3                    ase port 0 lid 8 lmc 0
Switch  : 0x0002c90200424418 ports 36
"MF0;ibswitch:MTS3610/L08/U1" base port 0 lid 22 lmc 0
Switch  : 0x0002c90200424428 ports 36
"MF0;ibswitch:MTS3610/L07/U1" base port 0 l                  0
Switch  : 0x0002c9020040d990 ports 36
"MF0;ibswitch:MTS3610/L06/U1" base port 0 lid 118 lmc 0
Switch  : 0x0002c9020040e518 ports 36
"MF0;ibswitch:MTS3610/L05/U1" base port 0 lid 23 lmc 0
Switch  : 0x0002c9020040e3d8 ports 36
"MF0;ibswitch:MTS3610/L03/U1" base port 0 lid 168 lmc 0
Switch  : 0x0002c9020040d970 ports 36
"MF0;ibswitch:MTS3610/L02/U1" base port 0 lid 89 lmc 0
. . .
```

Switch vendor information

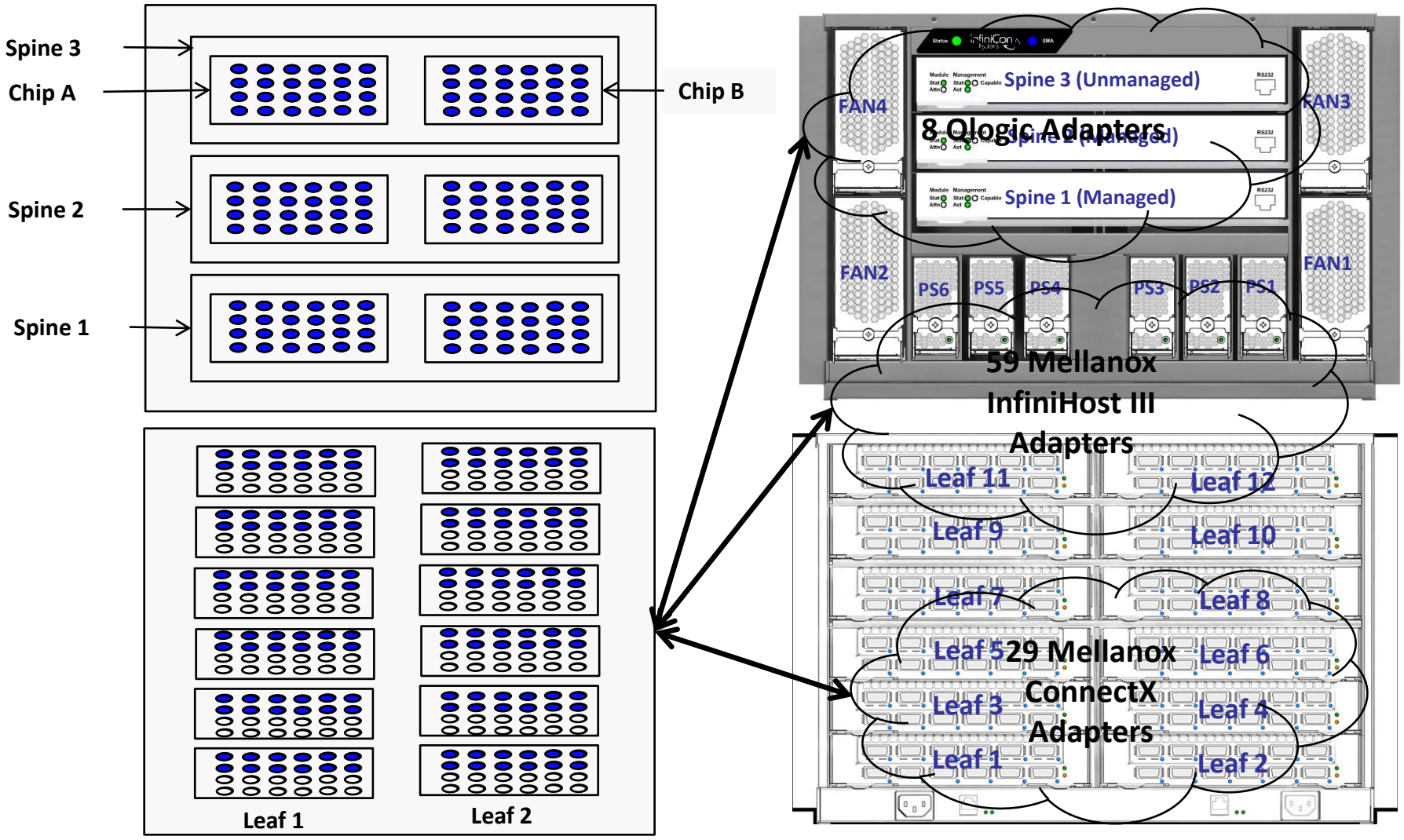
Ports per switch

Discovering Network Connectivity

```
% ibnetdiscover
Switch  36 "S-0002c90200424400"          #
"MF0;ibswitch:MTS3610/L11/U1" base port 0 lid 2 lmc 0
[1]     "H-0002c903000a83cc"[1](2c903000a83cd)          #
"node133 HCA-1" lid 157 4xQDR
[2]     "H-0002c903000a8a6c"[1](2c903000a8a6d)          #
"node134 HCA-1" lid 84 4xQDR
[3]     "H-0002c903000a9258"[1](2c903000a9259)          #
"node135 HCA-1" lid 188 4xQDR
[4]     "H-0002c903000a9388"[1](2c903000a9389)          #
"node136 HCA-1" lid
[5]     "H-0002c9030                3000a8a71)          #
"node137 HCA-1" lid 182 4xQDR
[6]     "H-0002c903000a8df0"[1](2c903000a8df1)          #
"node138 HCA-1" lid 193 4xQDR
[7]     "H-0002c903000a8dc0"[1](2c903000a8dc1)          #
"node139 HCA-1" lid 146 4xQDR
[8]     "H-0002c903000a89b4"[1](2c903000a89b5)          #
"node140 HCA-1" lid 137 4xQDR
[9]     "H-0002c903000a8b6c"[1](2c903000a8b6d)          #
"node141 HCA-1" lid 85 4xQDR
```

Connectivity of each switch port

Roughly Constructed Network Fabric



Network Management Infrastructure and Tools

- **Management Infrastructure**
  - Subnet Manager
  - **Diagnostic tools**
    - System Discovery Tools
    - **System Health Monitoring Tools**
    - **System Performance Monitoring Tools**
  - Fabric management tools

Overall Diagnostics

- Tools to query overall fabric health
  - Ibdiagnet: scans the fabric using directed route packets and extracts information regarding its connectivity and devices

```
[ib1 ]# ibdiagnet -r
…
STAGE                                    Errors Warnings
    Bad GUIDs/LIDs Check                    0        0
    Link State Active Check                 0        0
    Performance Counters Report             0        0
    Partitions Check                        0        0
    IPoIB Subnets Check                     0        0
    Subnet Manager Check                    0        0
    Fabric Qualities Report                 0        0
    Credit Loops Check                      0        0
    Multicast Groups Report                 0        0
```

End-node Adapter State

– ibportstate: handle port (physical) state and link speed of an InfiniBand port

```
[ib1 ]# ibportstate 8 1
PortInfo:
# Port info: Lid 8 port 1
LinkState:.....................Active
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps
```

End-node Adapter Counters

– Ibdatacounts: get InfiniBand port data counters

```
[ib1 ]# ibdatacounts 119 1
# Port counters: Lid 119 port 1
XmtData:.........................2102127705
RcvData:.........................2101904109
XmtPkts:.........................9069780
RcvPkts:.........................9068305
```

```
[ib1 ]# ibdatacounts 119 1
# Port counters: Lid 119 port 1
XmtData:.........................432
RcvData:.........................432
XmtPkts:.........................6
RcvPkts:.........................6
```

```
[ib1 ]# ibcheckerrs -v 20 1
Error check on lid 20 (ib12 HCA-2) port 1:  OK
```

Verbs Level Performance

ibv_send_lat

```
Send Latency Test
Number of qps   : 1
Connection type : RC
RX depth        : 600
CQ Moderation   : 50
Mtu             : 2048B
Link type       : IB
Max inline data : 400B
rdma_cm QPs     : OFF
Data ex. method : Ethernet
..
#bytes #iterations    t_min[usec]    t_max[usec]  t_typical[usec]
2       1000          1.44           6.73         1.46
```

ibv_write_lat

```
RDMA_Write Latency Test
Number of qps   : 1
Connection type : RC
Mtu             : 2048B
Link type       : IB
Max inline data : 400B
rdma_cm QPs     : OFF
Data ex. method : Ethernet
..
#bytes #iterations    t_min[usec]    t_max[usec]  t_typical[usec]
2       1000          1.37           37.99        1.38
```

Perfquery

Before ibv_send_lat

```
# Port counters: Lid 149 port 1
PortSelect:.....................1
CounterSelect:..................0x1400
SymbolErrorCounter:.............0
LinkErrorRecoveryCounter:.......0
LinkDownedCounter:..............0
PortRcvErrors:..................0
PortRcvRemotePhysicalErrors:....0
PortRcvSwitchRelayErrors:.......0
PortXmitDiscards:...............0
PortXmitConstraintErrors:.......0
PortRcvConstraintErrors:........0
CounterSelect2:.................0x00
LocalLinkIntegrityErrors:.......0
ExcessiveBufferOverrunErrors:....0
VL15Dropped:....................0
PortXmitData:...................0
PortRcvData:....................0
PortXmitPkts:...................0
PortRcvPkts:....................0
PortXmitWait:...................0
```

After ibv_send_lat

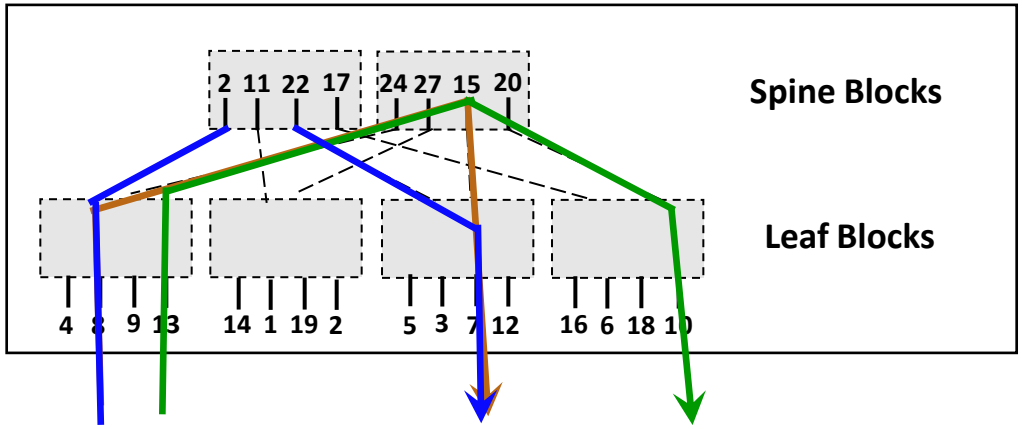
```
# Port counters: Lid 149 port 1
PortSelect:.....................1
CounterSelect:..................0x1400
SymbolErrorCounter:.............0
LinkErrorRecoveryCounter:.......0
LinkDownedCounter:..............0
PortRcvErrors:..................0
PortRcvRemotePhysicalErrors:....0
PortRcvSwitchRelayErrors:.......0
PortXmitDiscards:...............0
PortXmitConstraintErrors:.......0
PortRcvConstraintErrors:........0
CounterSelect2:.................0x00
LocalLinkIntegrityErrors:.......0
ExcessiveBufferOverrunErrors:....0
VL15Dropped:....................0
PortXmitData:...................14000
PortRcvData:....................14000
PortXmitPkts:...................2000
PortRcvPkts:....................2000
PortXmitWait:...................0
```

Network Switching and Routing

```
% ibroute -G 0x66a000700067c
Lid  Out   Destination
        Port     Info
0x0001 001  : (Ch                                uid 0x0002c9030001e3f3: ' HCA-1')
0x0002 013  : (Ch                                uid 0x0002c9020023c301: ' HCA-1')
0x0003 014  : (Ch                                uid 0x0002c9030001e603: ' HCA-1')
0x0004 015  : (Channel Adapter portguid 0x0002c9020023c305: ' HCA-2')
0x0005 016  : (Channel Adapter portguid 0x0011750000ffe005: ' HCA-1')
0x0014 017  : (Switch portguid 0x00066a0007000728: 'SilverStorm 9120
GUID=0x00066a00020001aa Leaf 8, Chip A')
0x0015 020  : (Channel Adapter portguid 0x0002c9020023c131: ' HCA-2')
0x0016 019  : (Switch portguid 0x00066a0007000732: 'SilverStorm 9120
GUID=0x00066a00020001aa Leaf 10, Chip A')
0x0017 019  : (Channel Adapter portguid 0x0002c9030001c937: ' HCA-1')
0x0018 019  : (Channel Adapter portguid 0x0002c9020023c039: ' HCA-2')
...
```

Packets to LID 0x0001 will be sent out on Port 001

Static Analysis of Network Contention



- Based on destination LIDs and switching/routing information, the exact path of the packets can be identified
  - If application communication pattern is known, we can statically identify possible network contention

Dynamic Analysis of Network Contention

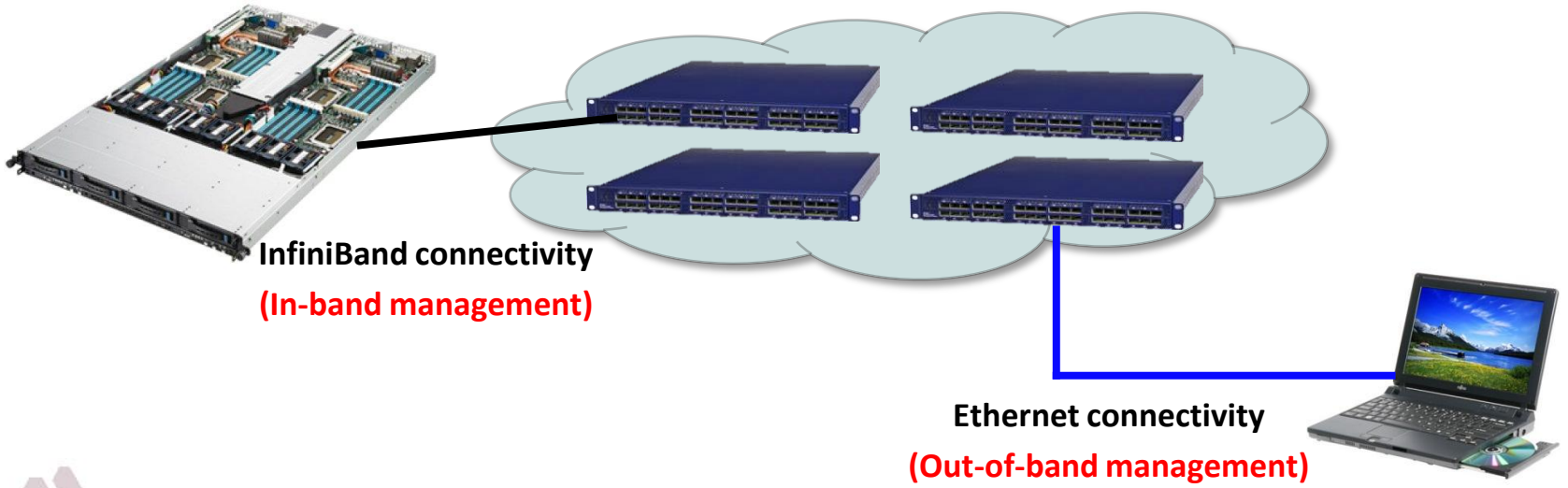
- IB provides many optional counters to query performance counters
  - PortXmitWait: Number of ticks in which there was data to send, but no flow-control credits
  - RNR NAKs: Number of times a message was sent, but the receiver has not yet posted a receive buffer
    - This can timeout, so it can be an error in some cases
  - PortXmitFlowPkts: Number of (link-level) flow-control packets transmitted on the port
  - SWPortVLCongestion: Number of packets dropped due to congestion

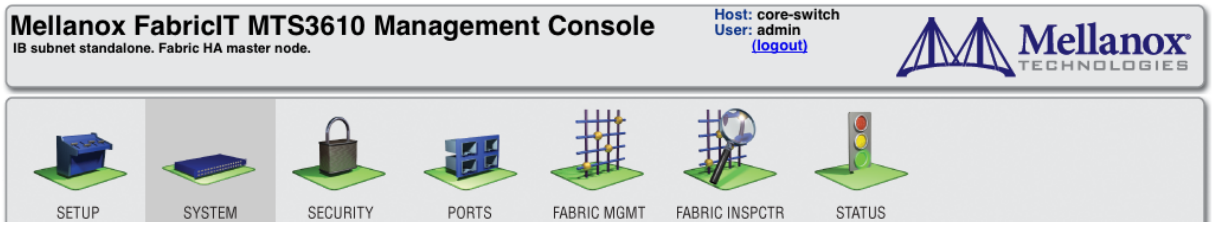
Network Management Infrastructure and Tools

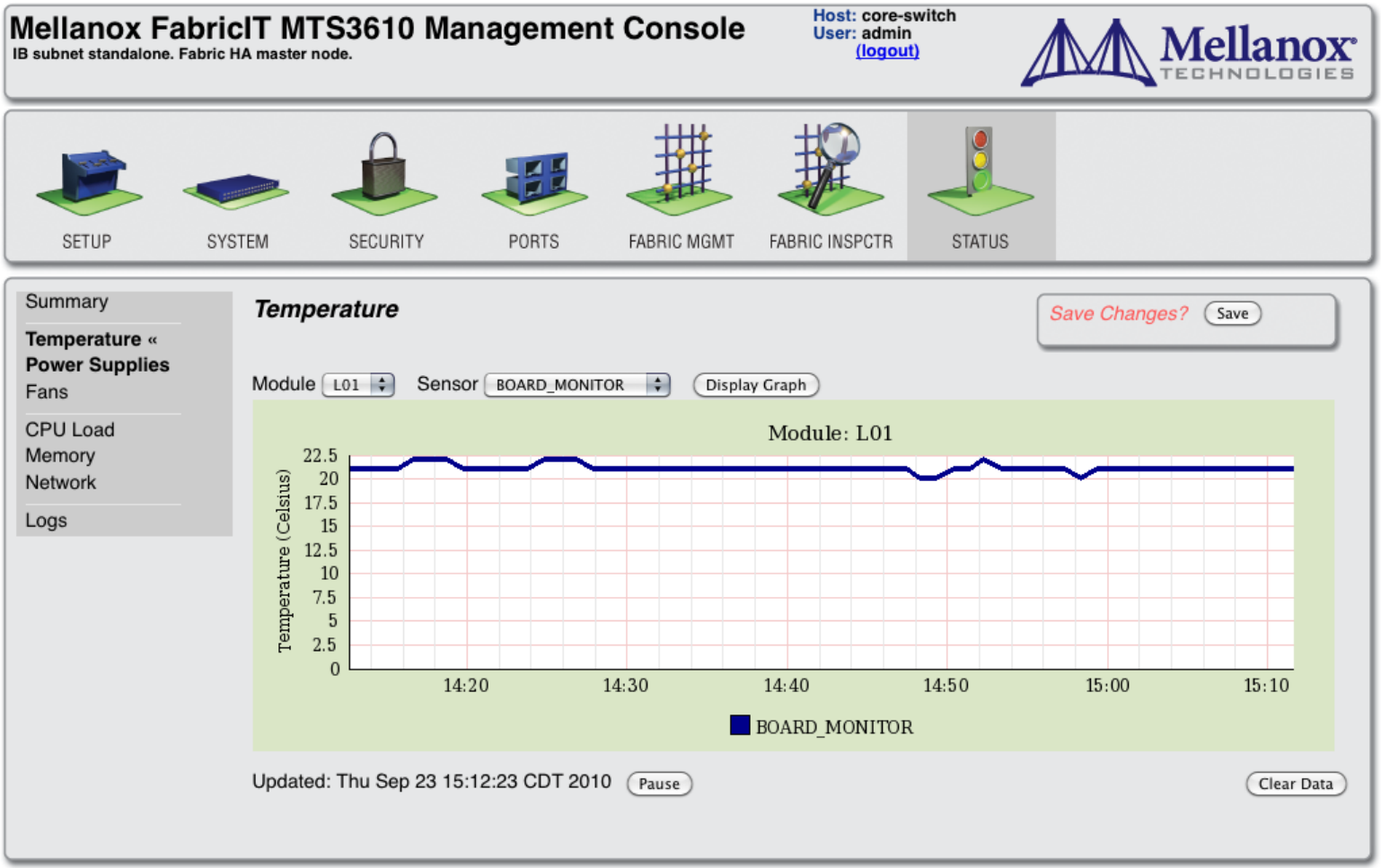
- **Management Infrastructure**
  - Subnet Manager
  - Diagnostic tools
    - System Discovery Tools
    - System Health Monitoring Tools
    - System Performance Monitoring Tools
  - **Fabric management tools**

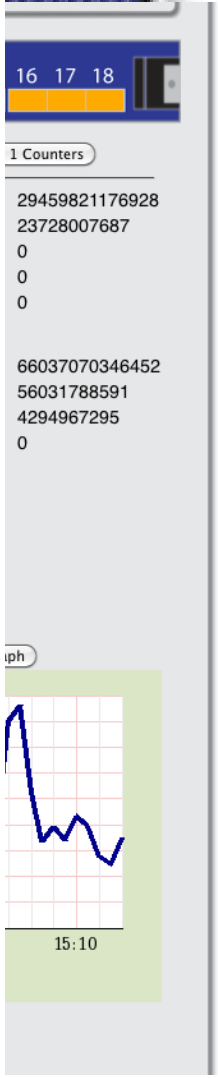
In-band management vs. Out-of-band management

- InfiniBand provides two forms of management
  - Out-of-band management (similar to other networks)
  - In-band management (used by the subnet manager)
- Out-of-band management requires a separate Ethernet port on the switch, where an administrator can plug in his/her laptop
- In-band management allows the switch to receive management commands directly over the regular communication network




Mellanox FabricIT MTS3610 Management Console
IB subnet standalone. Fabric HA master node.
Host: core-switch
User: admin
(logout)
Mellanox TECHNOLOGIES
SETUP
SYSTEM
SECURITY
PORTS
FABRIC MGMT
FABRIC INSPCTR
STATUS


Mellanox FabricIT MTS3610 Management Console
IB subnet standalone. Fabric HA master node.
Host: core-switch
User: admin
(logout)
Mellanox TECHNOLOGIES
SETUP
SYSTEM
SECURITY
PORTS
FABRIC MGMT
FABRIC INSPCTR
STATUS
Summary
Temperature «
Power Supplies
Fans
CPU Load
Memory
Network
Logs
Temperature
Save Changes?
Save
Module L01
Sensor BOARD_MONITOR
Display Graph
Module: L01
Temperature (Celsius)
22.5
20
17.5
15
12.5
10
7.5
5
2.5
0
14:20
14:30
14:40
14:50
15:00
15:10
BOARD_MONITOR
Updated: Thu Sep 23 15:12:23 CDT 2010
Pause
Clear Data
© 2009-2010 Mellanox Technologies, Inc.


16 17 18
1 Counters
29459821176928
23728007687
0
0
0
66037070346452
56031788591
4294967295
0
15:10

INAM – InfiniBand Network Analysis and Monitoring using OpenSM

- Monitors the following in real time
  - Performance Counters
  - Subnet Management Attributes
  - Subnet Manager information in real time

Configure

Switch InfinIO3008 #3 ○	Switch InfinIO3008 #4 ○	Switch InfinIO3008 #6 ○	Switch InfinIO3008 #8 ○
ws26 HCA1 ○	ws25 HCA1 ○	ws5 HCA1 ○	ws3 HCA1 ○

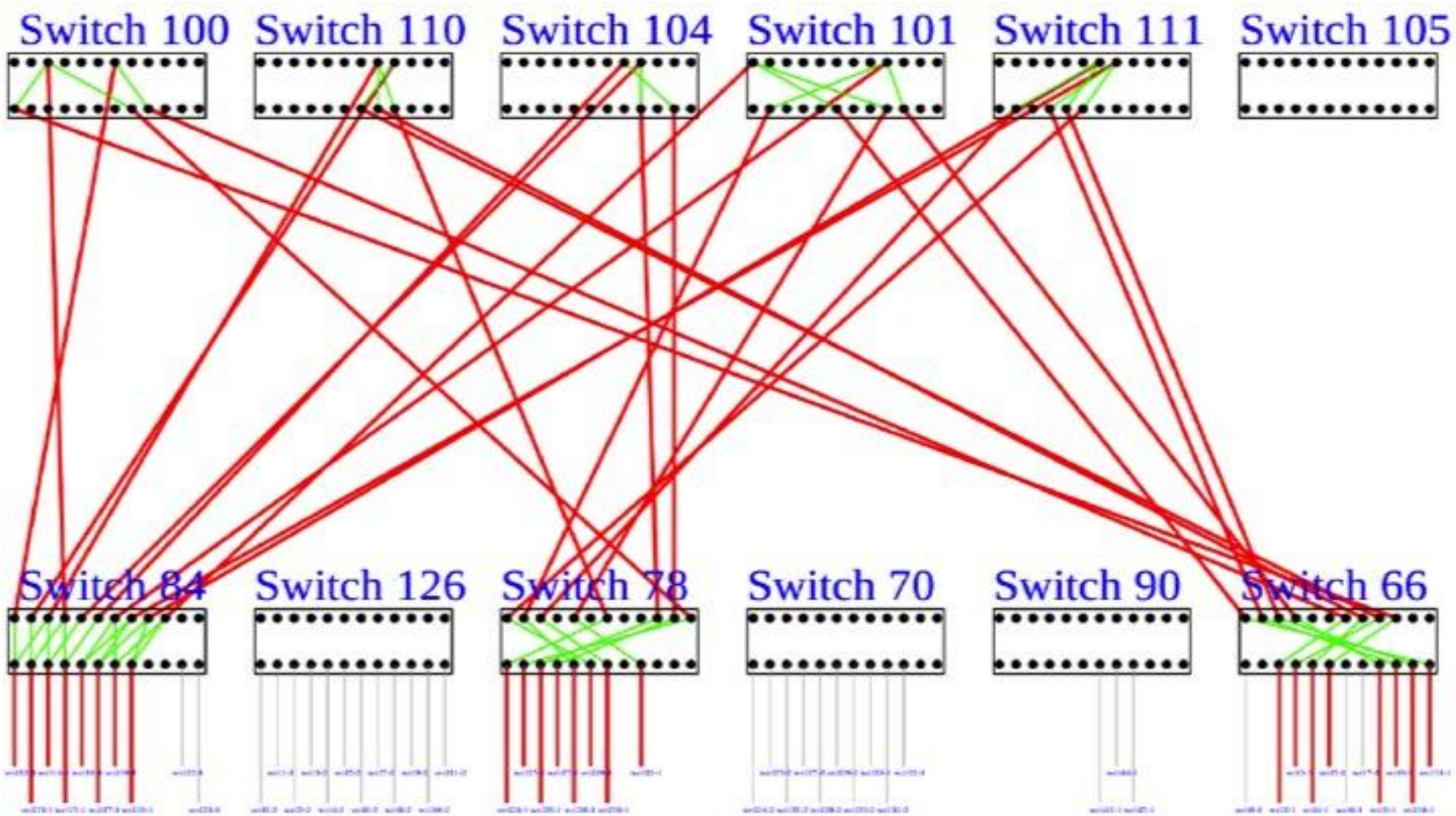
☐ SymbolErrors	☐ LinkRecovers	☐ LinkDowned	☐ RcvErrors
☐ RcvRemotePhysErrors	☐ RcvSwRelayErrors	☐ XmtDiscards	☐ XmtConstraintErrors
☐ RcvConstraintErrors	☐ LinkIntegrityErrors	☐ ExcBufOverrunErrors	☐ VL15Dropped
☐ XmtData	☐ RcvData	☐ XmtPkts	☐ RcvPkts

Submit

Flexibility for Selecting Performance Counters to monitor

N. Dandapanthula, H. Subramoni, J. Vienne, K. Kandalla, S. Sur, D. K. Panda, and R. Brightwell, INAM - A Scalable InfiniBand Network Analysis and Monitoring Tool, 4th Int'l Workshop on Productivity and Performance (PROPER 2011), in conjunction with EuroPar, Aug. 2011.

Visualizing Network Traffic Pattern and Contention for A Given Program Execution



P2P communication with 32 processes on switch 84 and 16 processes each on switch 78 and 66

Presentation Overview

- Common Challenges in Building HEC Systems with IB and HSE
  - Network Adapters and NUMA Interactions
  - Scalability and Memory Overheads
  - Network Switches, Topology and Routing
  - Network Bridges
- System Specific Challenges and Case Studies
  - HPC (MPI, PGAS and GPU/MIC Computing)
  - Big Data (Hadoop with HDFS and HBase; Memcached)
  - Storage and File Systems
  - Grid Computing
- Open Fabrics Software Stack and RDMA Programming
- Network Management Infrastructure and Tools
- **Conclusions and Final Q&A**

Concluding Remarks

- Presented networking requirements for High-End Computing Systems with IB and HSE
- Discussed common set of challenges in designing HEC Systems
- Presented Challenges and Solutions in designing various High-End Computing systems with IB and HSE
- IB and HSE are emerging as new architectures leading to a new generation of networked computing systems, opening many research issues needing novel solutions

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- N. Islam (Ph.D.)
- J. Jose (Ph.D.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- M. Luo (Ph.D.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekhar (Ph.D.)
- M. Rahman (Ph.D.)
- R. Shir (Ph.D.)
- H. Subramoni (Ph.D.)
- A. Venkatesh (Ph.D.)

Current Post-Docs

- X. Lu
- K. Hamidouche

Current Programmers

- M. Arnold
- D. Bureddy
- J. Perkins

Past Students

- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

Past Post-Docs

- H. Wang
- X. Besseron
- H.-W. Jin
- E. Mancini
- S. Marcarelli
- J. Vienne

Past Research Scientist

- S. Sur

Web Pointers

http://www.cse.ohio-state.edu/~panda

http://www.cse.ohio-state.edu/~subramon

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

MVAPICH Web Page

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

panda@cse.ohio-state.edu

subramon@cse.ohio-state.edu

MVAPICH User Group (MUG) Meeting
August 26-27, 2013, Columbus, Ohio, U.S.A

- The MUG meeting will provide an open forum for all attendees (users, researchers, system administrators, engineers, and students) to share their knowledge about MVAPICH2/MVAPICH2-X on large-scale systems and diverse applications.
- The event includes:
  - Talks from experts in the field
  - Presentations from the MVAPICH team on tuning and optimization strategies
  - Troubleshooting guidelines
  - Contributed presentations
  - Open mic session
  - Interactive one-on-one session with the MVAPICH developers

Call for Presentation

- The MVAPICH team is requesting the submission of presentations from MVAPICH2 and MVAPICH2-X users to be included in the event.

Presentation Submission Deadline: July 1, 2013

Notification of Acceptance: July 8, 2013

Advanced Registration Deadline: July 15, 2013

The preliminary program has been posted at mug.mvapich.cse.ohio-state.edu/program/