MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

How to Boost the Performance of your MPI and PGAS Applications with MVAPICH2 Libraries?

A Tutorial at

MVAPICH User Group Meeting 2016

by

The MVAPICH Group

The Ohio State University

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

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

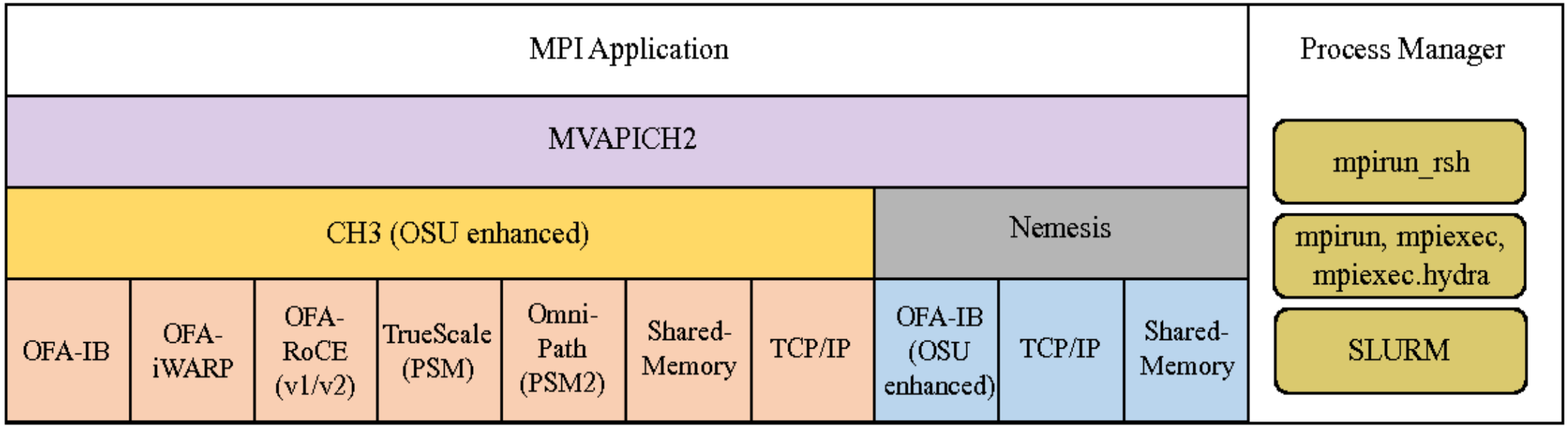
MVAPICH2 Software Family

Requirements	Library
MPI with IB, iWARP and RoCE	MVAPICH2
Advanced MPI, OSU INAM, PGAS and MPI+PGAS with IB and RoCE	MVAPICH2-X
MPI with IB & GPU	MVAPICH2-GDR
MPI with IB & MIC	MVAPICH2-MIC
HPC Cloud with MPI & IB	MVAPICH2-Virt
Energy-aware MPI with IB, iWARP and RoCE	MVAPICH2-EA
MPI Energy Monitoring Tool	OEMT
InfiniBand Network Analysis and Monitoring	OSU INAM
Microbenchmarks for Measuring MPI and PGAS Performance	OMB

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Process Mapping and Point-to-point Intra-node Protocols
    - MPI-3 RMA
    - Collectives
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

MVAPICH2 Interfaces (Latest Release 2.2rc2)



Support for Different PCI/PCIe Interfaces

Major Computing Platforms: IA-32, EM64T, OpenPower, Nehalem, Westmere, Sandybridge, Ivybridge, Haswell, Opteron, Magny, ..

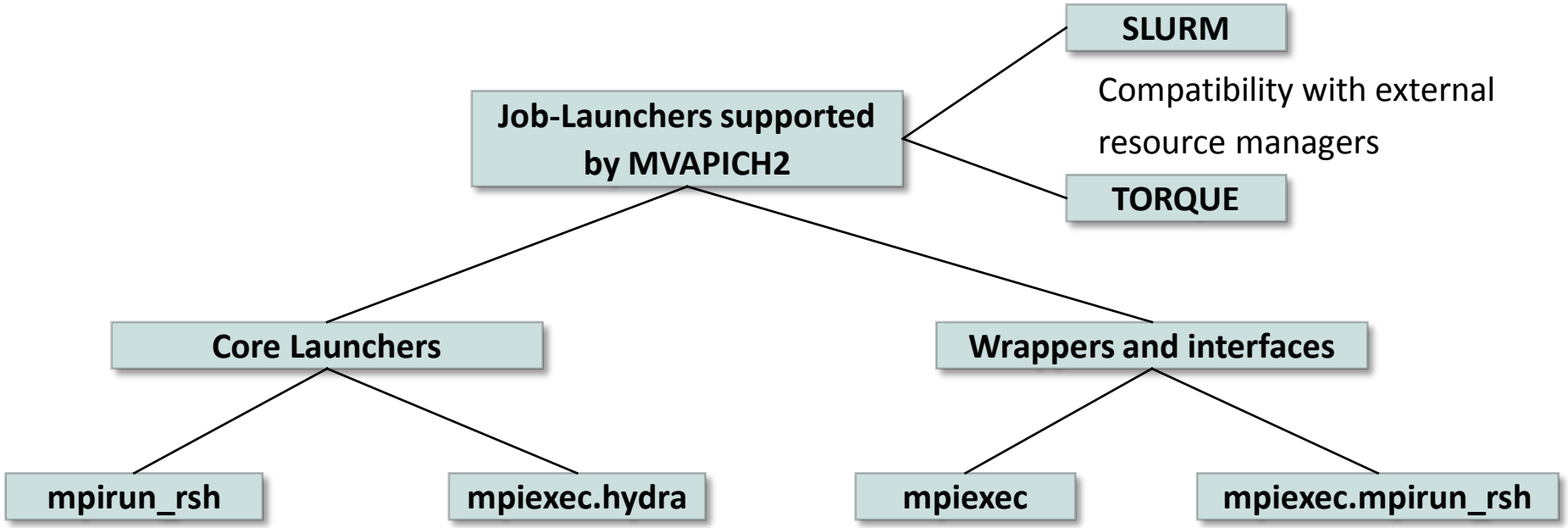
MVAPICH2 2.2rc2

- Released on 08/08/2016
- Major Features and Enhancements
  - Based on MPICH-3.1.4
  - Enhanced performance for MPI_Comm_split through new bitonic algorithm
  - Enable graceful fallback to Shared Memory if LiMIC2 or CMA transfer fails
  - Enable support for multiple MPI initializations
  - Remove verbs dependency when building the PSM and PSM2 channels
  - Allow processes to request MPI_THREAD_MULTIPLE when socket or NUMA node level affinity is specified
  - Point-to-point and collective performance optimization for Intel Knights Landing
  - Automatic detection and tuning for InfiniBand EDR HCAs
  - Collective tuning for Opal@LLNL, Bridges@PSC, and Stampede-1.5@TACC
  - Tuning and architecture detection for Intel Broadwell processors
  - Warn user to reconfigure library if rank type is not large enough to represent all ranks in job
  - Unify process affinity support in Gen2, PSM and PSM2 channels

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - **Job start-up**
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Process Mapping and Point-to-point Intra-node Protocols
    - MPI-3 RMA
    - Collectives
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

Job-Launchers supported by MVAPICH2



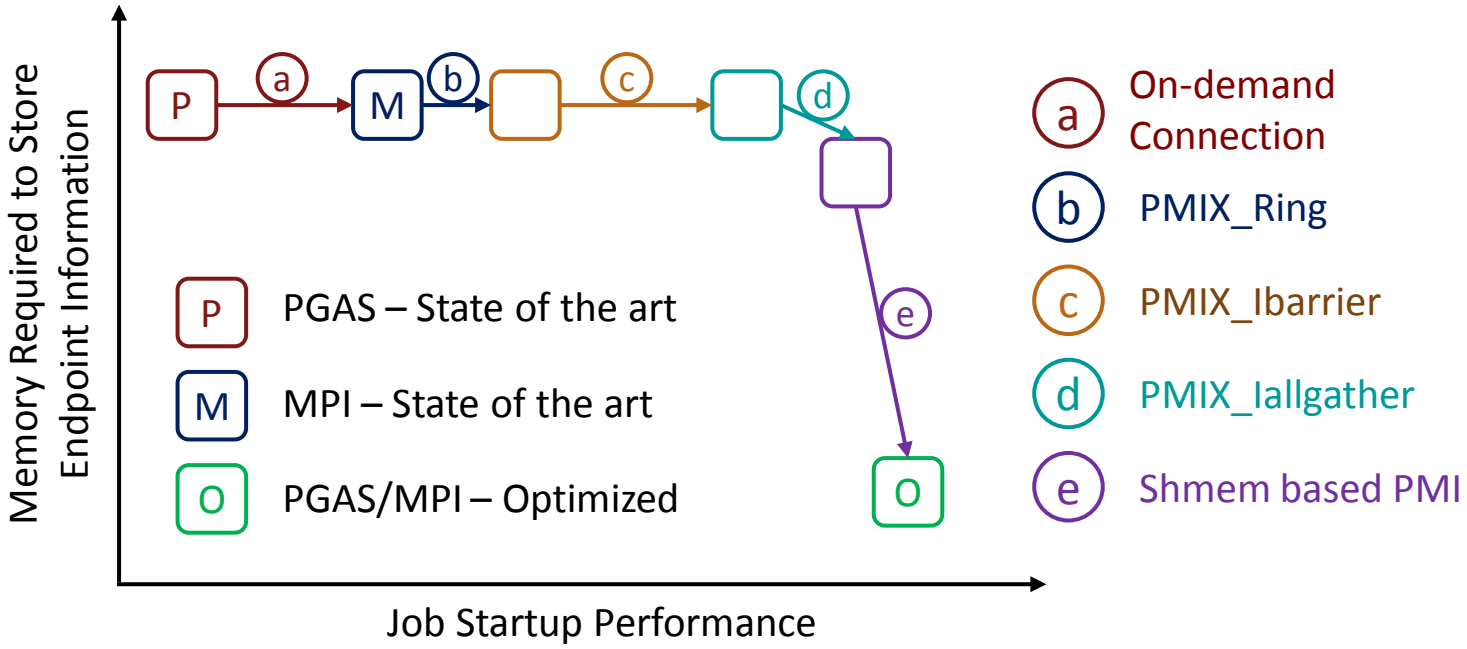
- Default and preferred launcher for all interfaces
- Supports scalable hierarchical job-startup mechanism

Developed at ANL by the MPICH team

To support interfaces prescribed by the MPI standard

Hydra-like interface to mpirun_rsh

Towards High Performance and Scalable Startup at Exascale



- Near-constant MPI and OpenSHMEM initialization time at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM respectively at 16,384 processes
- Memory consumption reduced for remote endpoint information by O(processes per node)
- 1GB Memory saved per node with 1M processes and 16 processes per node

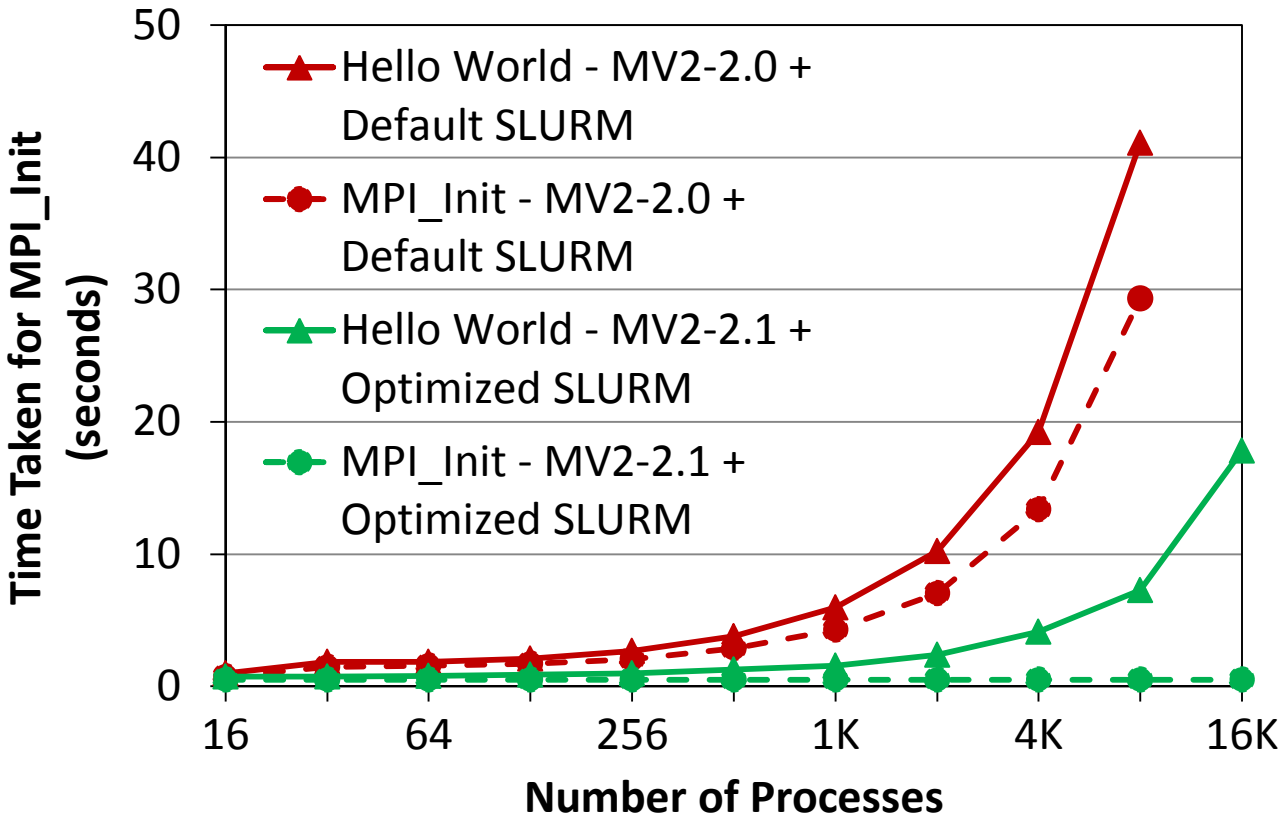
(a) **On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI.** S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS '15)

(b) **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia '14)

(c) (d) **Non-blocking PMI Extensions for Fast MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '15)

(e) **SHMEMPMI – Shared Memory based PMI for Improved Performance and Scalability.** S. Chakraborty, H. Subramoni, J. Perkins, and D K Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '16) , *Accepted for Publication*

Non-blocking Process Management Interface (PMI) Primitives for Scalable MPI Startup



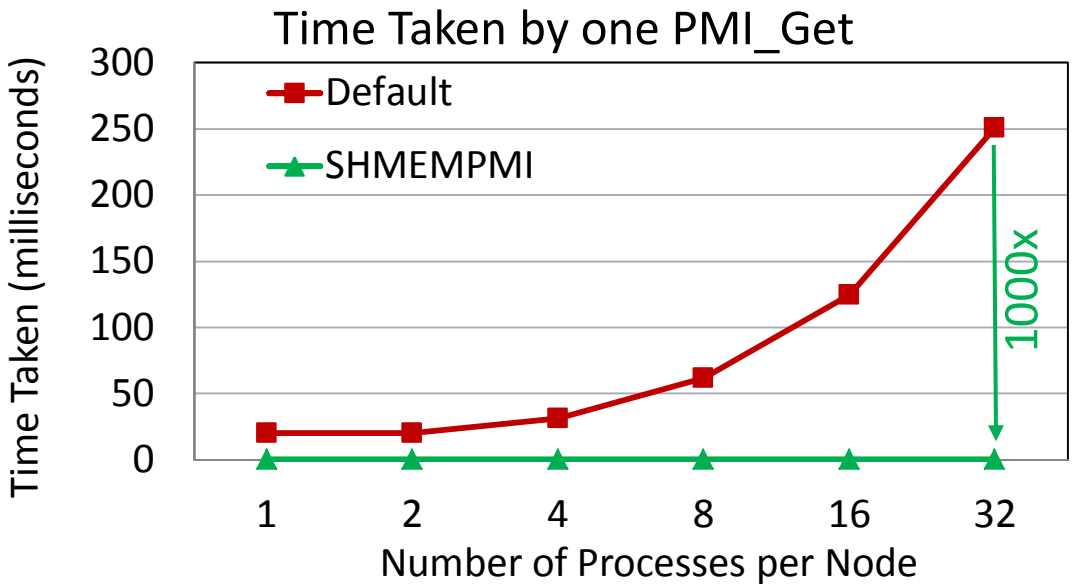
- Address exchange over PMI is the major bottleneck in job startup
- Non-blocking PMI exchange hides this cost by overlapping it with application initialization and computation
- New PMI operation PMIX_Allgather for improved symmetric data transfer

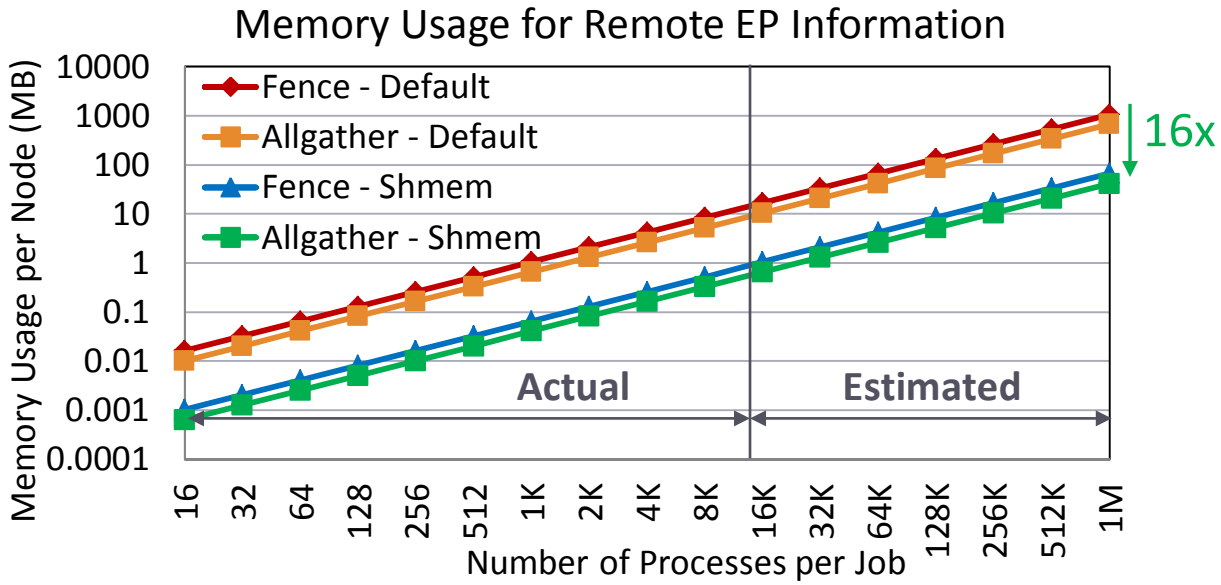
- Near-constant MPI_Init at any scale
- MPI_Init is 59 times faster at 8,192 processes (512 nodes)
- Hello World (MPI_Init + MPI_Finalize) takes 5.7 times less time at 8,192 processes

Available since MVAPICH2-2.1 and as patch for SLURM-15.08.8 and SLURM-16.05.1

Process Management Interface (PMI) over Shared Memory (SHMEMPMI)

- SHMEMPMI allows MPI processes to directly read remote endpoint (EP) information from the process manager through shared memory segments
- Only a single copy per node - O(processes per node) reduction in memory usage
- Estimated savings of 1GB per node with 1 million processes and 16 processes per node
- Up to 1,000 times faster PMI Gets compared to default design
- **Available for MVAPICH2 2.2rc1 and SLURM-15.08.8 and SLURM-16.05.1**





How to Get the Best Startup Performance with MVAPICH2?

- **MV2_HOMOGENEOUS_CLUSTER=1** //Set for homogenous clusters
- **MV2_ON_DEMAND_UD_INFO_EXCHANGE=1** //Enable UD based address exchange

Using SLURM as launcher

- **Use PMI2**
  - ./configure --with-pm=slurm --with-pmi=pmi2
  - srun --mpi=pmi2 ./a.out
- **Use PMI Extensions**
  - Patch for SLURM 15 and 16 available at http://mvapich.cse.ohio-state.edu/download/mvapich
  - PMI Extensions are automatically detected by MVAPICH2

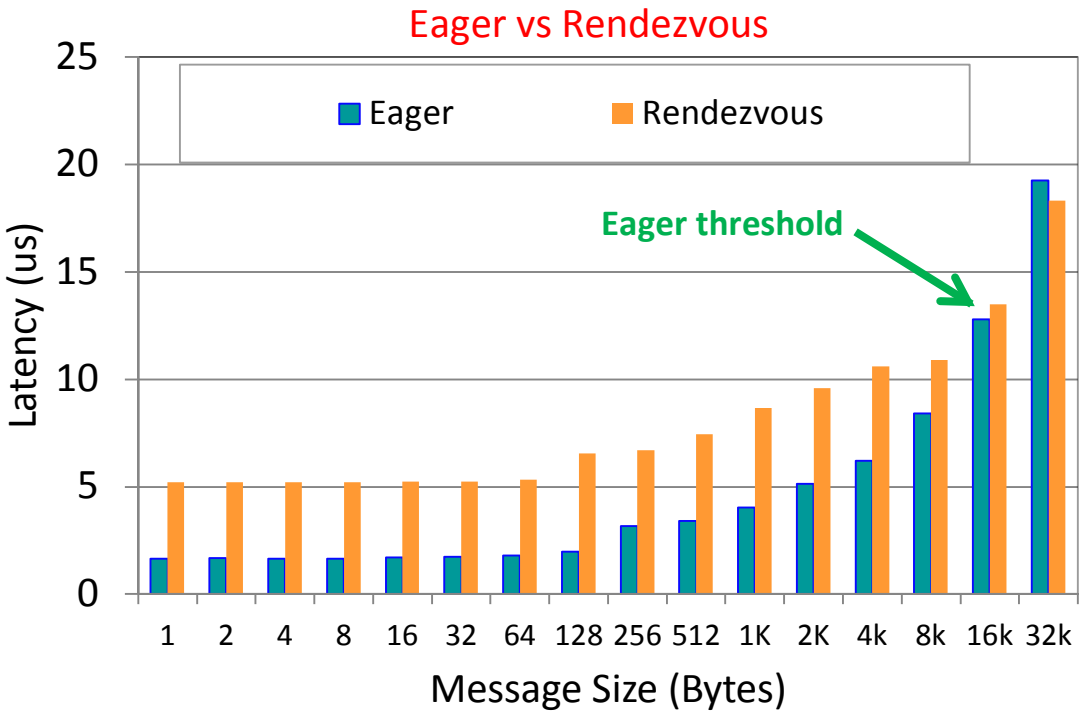
Using mpirun_rsh as launcher

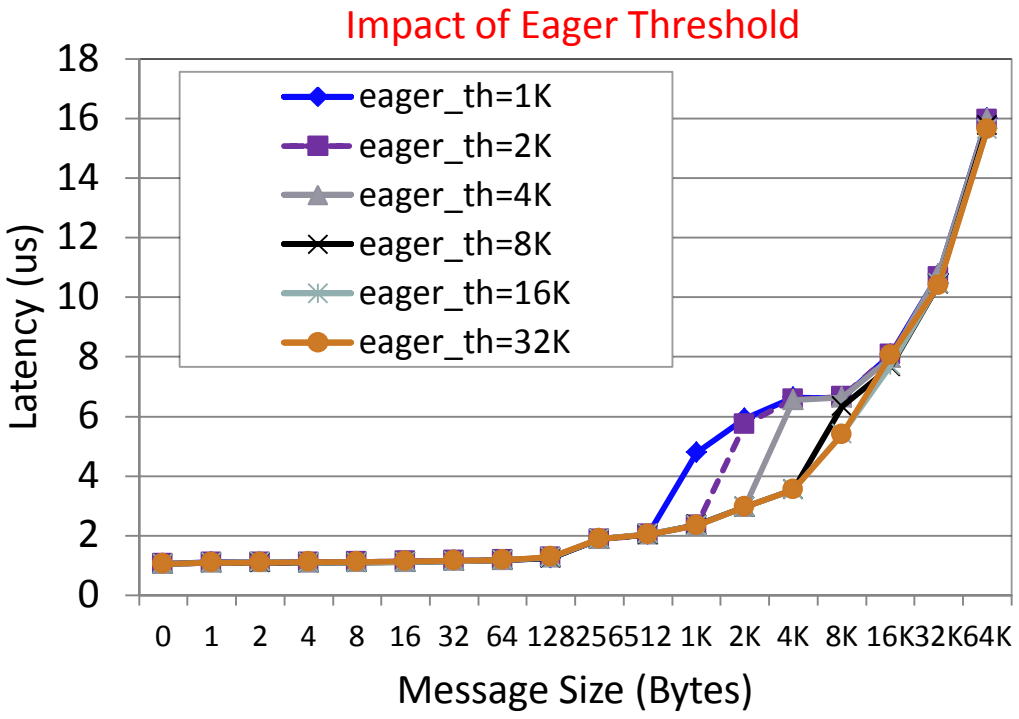
- **MV2_MT_DEGREE**
  - degree of the hierarchical tree used by mpirun_rsh
- **MV2_FASTSSH_THRESHOLD**
  - #nodes beyond which hierarchical-ssh scheme is used
- **MV2_NPROCS_THRESHOLD**
  - #nodes beyond which file-based communication is used for hierarchical-ssh during start up

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - **Point-to-point Inter-node Protocol**
      - **Eager and Rendezvous Protocols**
      - **RDMA Fast Path**
    - Transport Type Selection
    - Process Mapping and Point-to-point Intra-node Protocols
    - MPI-3 RMA
    - Collectives
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

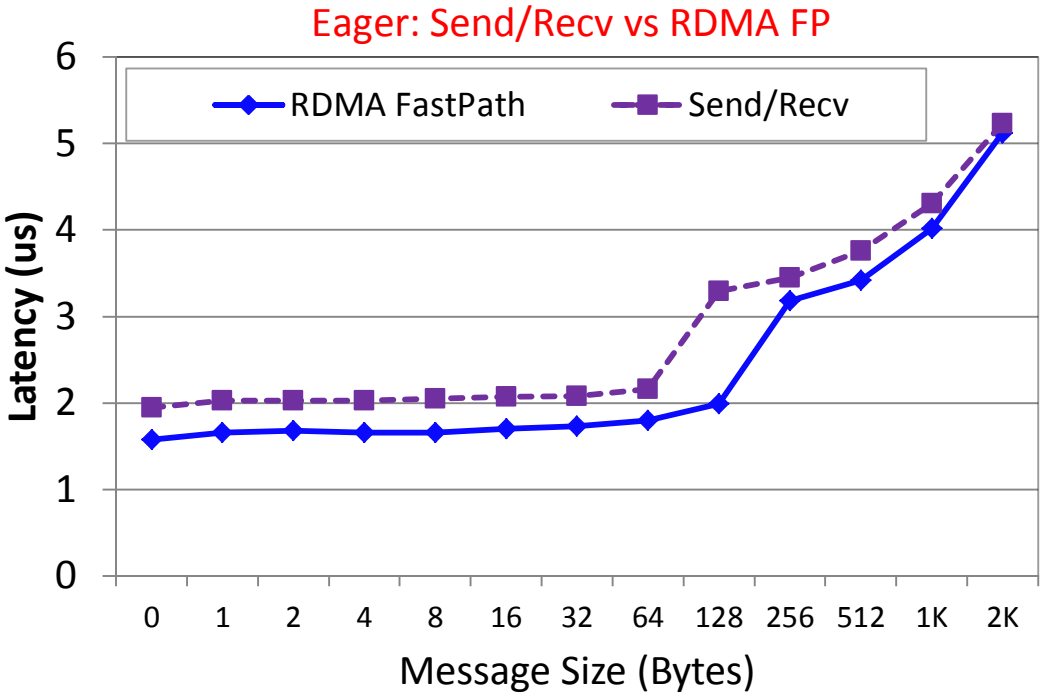
Inter-node Point-to-Point Tuning: Eager Thresholds

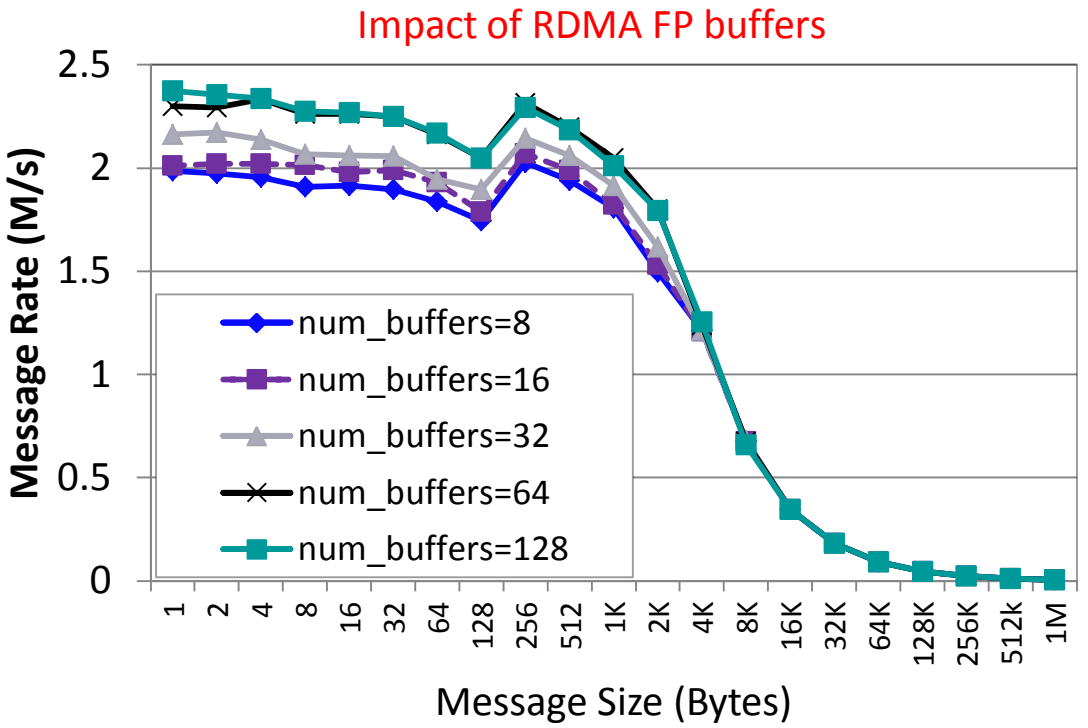




- Switching Eager to Rendezvous transfer
  - Default: Architecture dependent on common platforms, in order to achieve both best performance and memory footprint
- Threshold can be modified by users to get smooth performance across message sizes
  - mpirun_rsh –np 2 –hostfile hostfile MV2_IBA_EAGER_THRESHOLD=32K a.out
  - Memory footprint can increase along with eager threshold

Inter-node Point-to-Point Tuning: Number of Buffers and RNDV Protocols



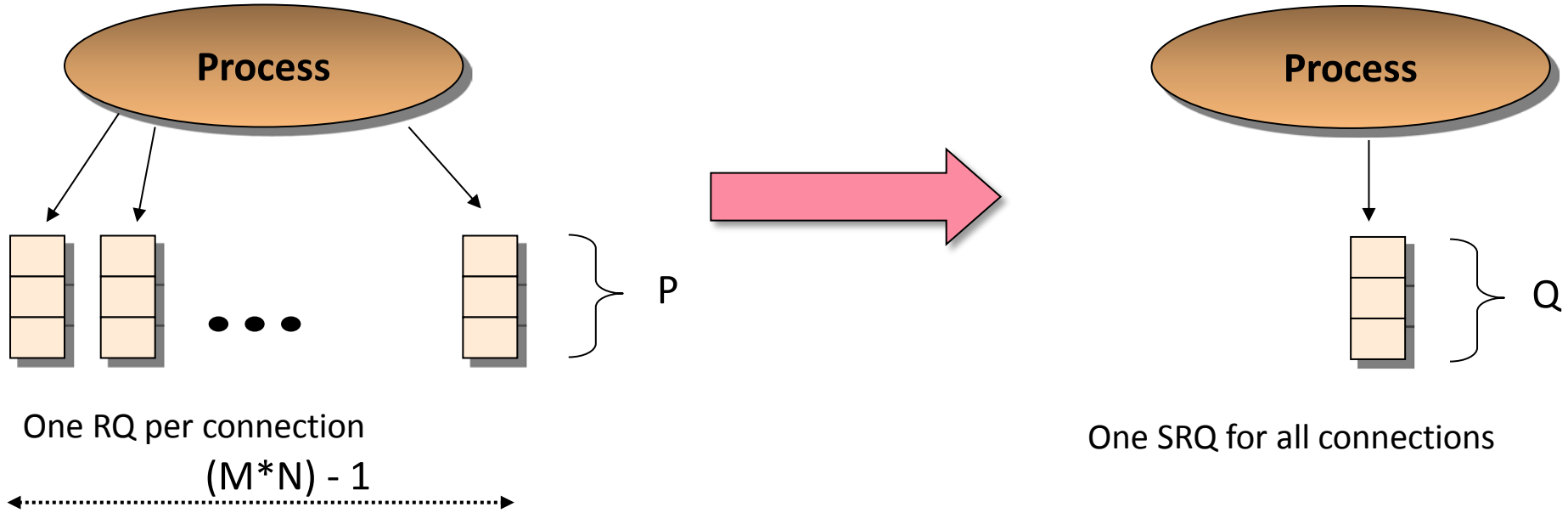


- RDMA Fast Path has advantages for smaller message range (default is on)
  - Disable: mpirun_rsh –np 2 –hostfile hostfile MV2_USE_RDMA_FASTPATH=0 a.out
- Adjust the number of RDMA Fast Path buffers (benchmark window size = 64):
  - mpirun_rsh –np 2 –hostfile hostfile MV2_NUM_RDMA_BUFFER=64 a.out
- Switch between Rendezvous protocols depending on applications:
  - mpirun_rsh –np 2 –hostfile hostfile MV2_RNDV_PROTOCOL=RGET a.out (Default: RPUT)

Presentation Overview

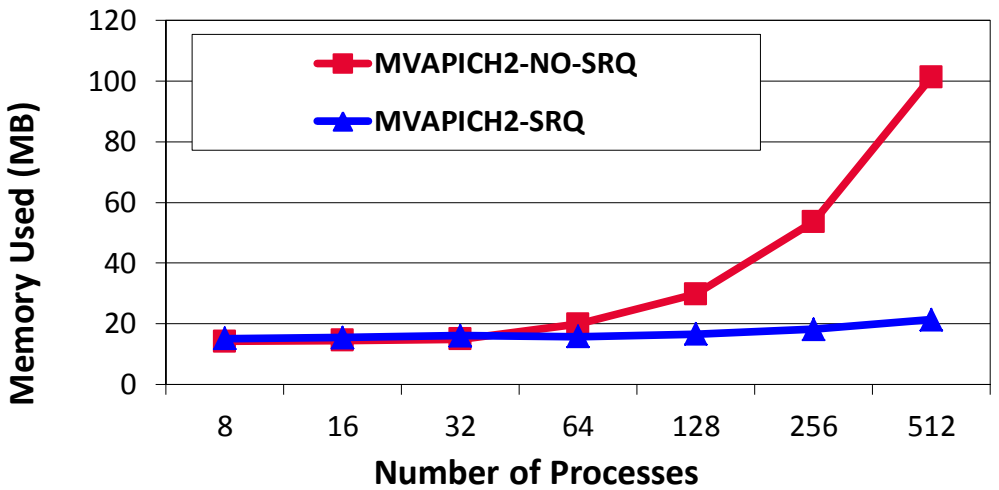
- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - Point-to-point Inter-node Protocol
    - **Transport Type Selection**
      - **Shared Receive Queue**
      - **eXtended Reliable Connect transport protocol**
      - **UD transport protocol and Hybrid**
      - **Direct Connected, User Mode Memory Registration and On Demand Paging**
    - Process Mapping and Point-to-point Intra-node Protocols
    - MPI-3 RMA
    - Collectives
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

Shared Receive Queue (SRQ)

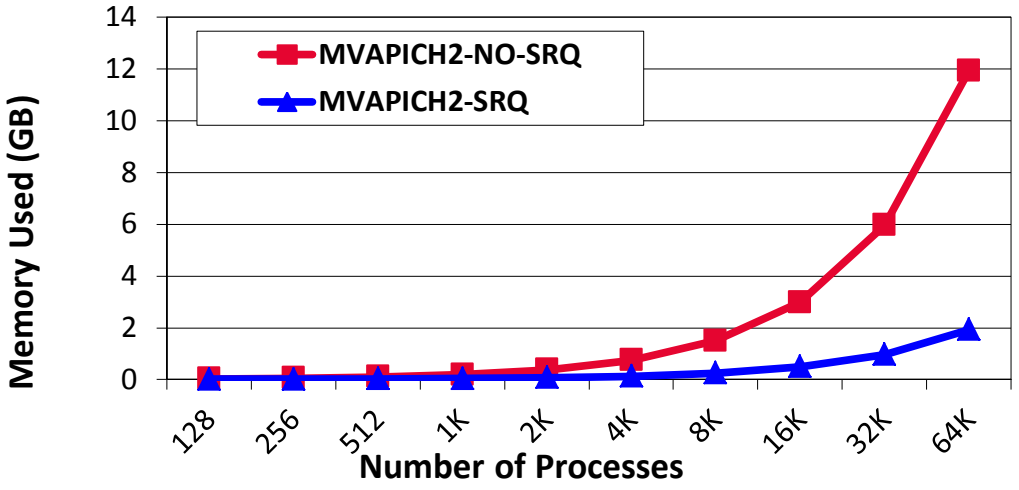


- SRQ is a hardware mechanism for a process to share receive resources (memory) across multiple connections
  - Introduced in InfiniBand specification v1.2
- 0 < Q << P*((M*N)-1)

Using Shared Receive Queues with MVAPICH2



MPI_Init memory utilization



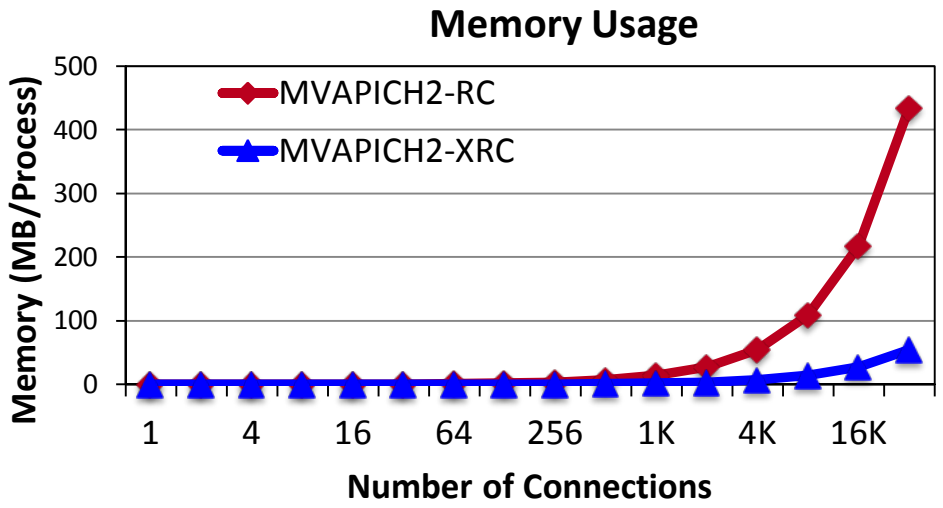
Analytical model

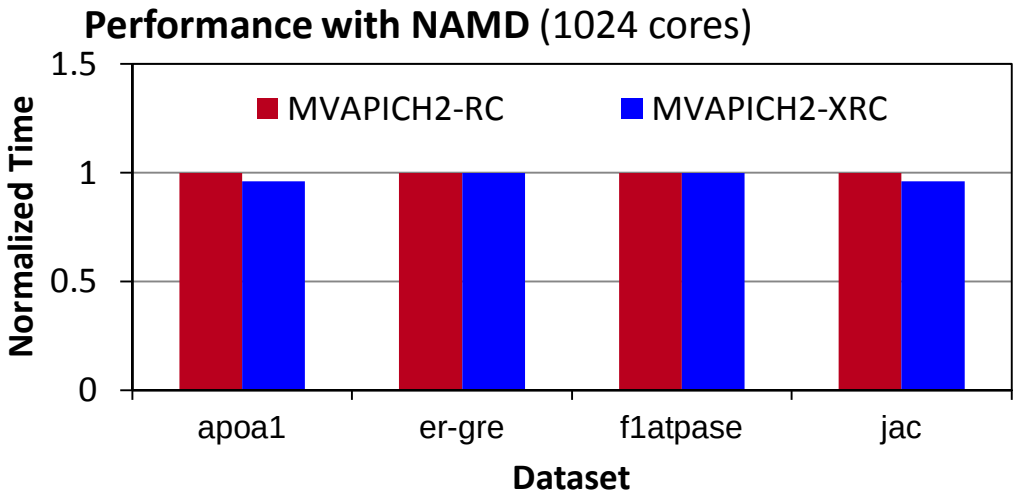
- **SRQ reduces the memory used by 1/6th at 64,000 processes**

Parameter	Significance	Default	Notes
MV2_USE_SRQ	• Enable / Disable use of SRQ in MVAPICH2	Enabled	• Always Enable
MV2_SRQ_MAX_SIZE	• Limits the maximum size of the SRQ • Places upper bound on amount of memory used for SRQ	4096	• Increase to 8192 for large scale runs
MV2_SRQ_SIZE	• Number of buffers posted to the SRQ • Automatically doubled by MVAPICH2 on receiving SRQ LIMIT EVENT from IB HCA	256	• Upper Bound: MV2_SRQ_MAX_SIZE

- **Refer to Shared Receive Queue (SRQ) Tuning section of MVAPICH2 user guide for more information**
- **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-1020008.5**

Using eXtended Reliable Connection (XRC) in MVAPICH2





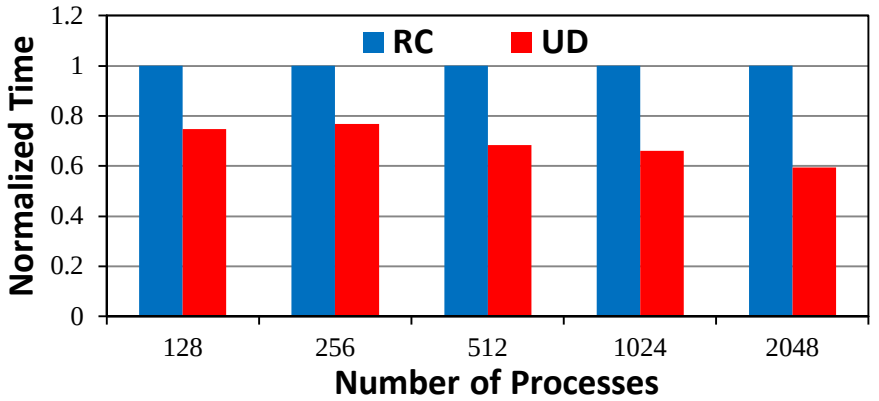
- Memory usage for 32K processes with 8-cores per node can be 54 MB/process (for connections)
- NAMD performance improves when there is frequent communication to many peers
- Enabled by setting **MV2_USE_XRC** to 1 (Default: Disabled)
- Requires OFED version > 1.3
  - Unsupported in earlier versions (< 1.3), OFED-3.x and MLNX_OFED-2.0
  - MVAPICH2 build process will automatically disable XRC if unsupported by OFED
- Automatically enables SRQ and ON-DEMAND connection establishment
  - **Refer to eXtended Reliable Connection (XRC) section of MVAPICH2 user guide for more information**
  - **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-1030008.6**

Using UD Transport with MVAPICH2

Memory Footprint of MVAPICH2

	RC (MVAPICH2 2.0b)				UD (MVAPICH2 2.0b)		
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

Performance with SMG2000

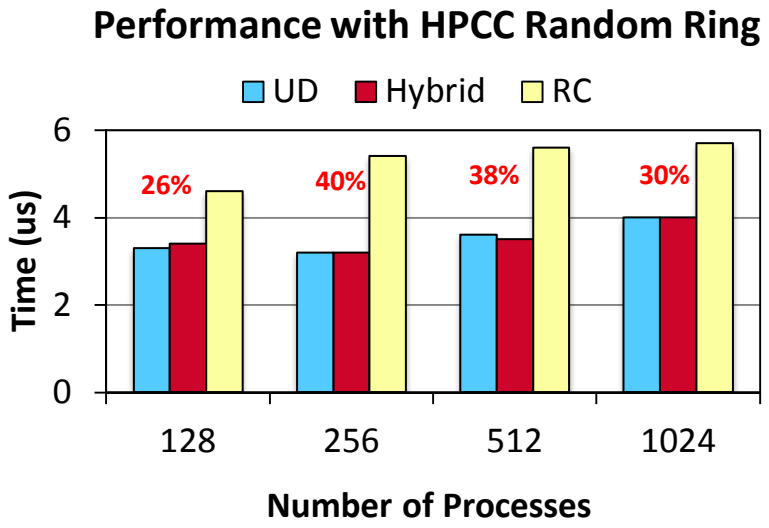


- Can use UD transport by configuring MVAPICH2 with the **–enable-hybrid**
  - Reduces QP cache trashing and memory footprint at large scale

Parameter	Significance	Default	Notes
MV2_USE_ONLY_UD	• Enable only UD transport in hybrid configuration mode	Disabled	• RC/XRC not used
MV2_USE_UD_ZCOPY	• Enables zero-copy transfers for large messages on UD	Enabled	• Always Enable when UD enabled
MV2_UD_RETRY_TIMEOUT	• Time (in usec) after which an unacknowledged message will be retried	500000	• Increase appropriately on large / congested systems
MV2_UD_RETRY_COUNT	• Number of retries before job is aborted	1000	• Increase appropriately on large / congested systems

- **Refer to Running with scalable UD transport section of MVAPICH2 user guide for more information**
- **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-640006.10**

Hybrid (UD/RC/XRC) Mode in MVAPICH2

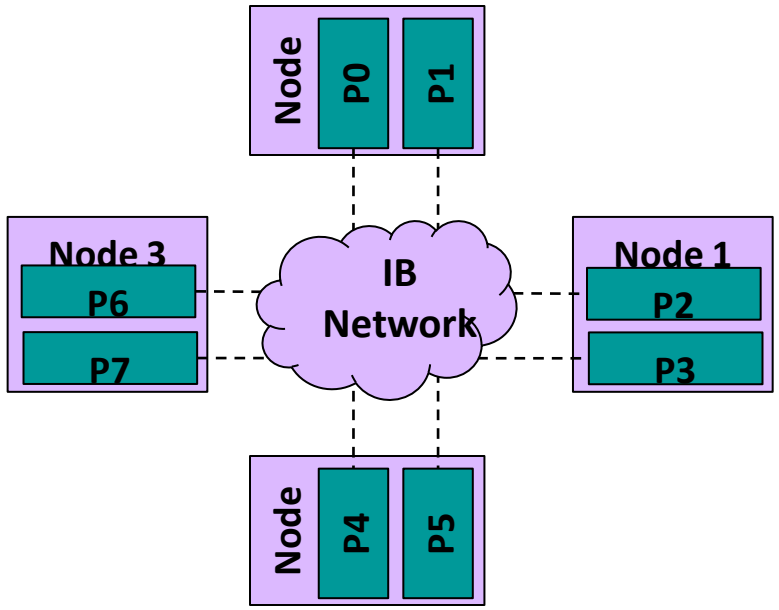


- Both UD and RC/XRC have benefits
  - Hybrid for the best of both
- Enabled by configuring MVAPICH2 with the –enable-hybrid
- Available since MVAPICH2 1.7 as integrated interface

Parameter	Significance	Default	Notes
MV2_USE_UD_HYBRID	• Enable / Disable use of UD transport in Hybrid mode	Enabled	• Always Enable
MV2_HYBRID_ENABLE_THRESHOLD_SIZE	• Job size in number of processes beyond which hybrid mode will be enabled	1024	• Uses RC/XRC connection until job size < threshold
MV2_HYBRID_MAX_RC_CONN	• Maximum number of RC or XRC connections created per process • Limits the amount of connection memory	64	• Prevents HCA QP cache thrashing

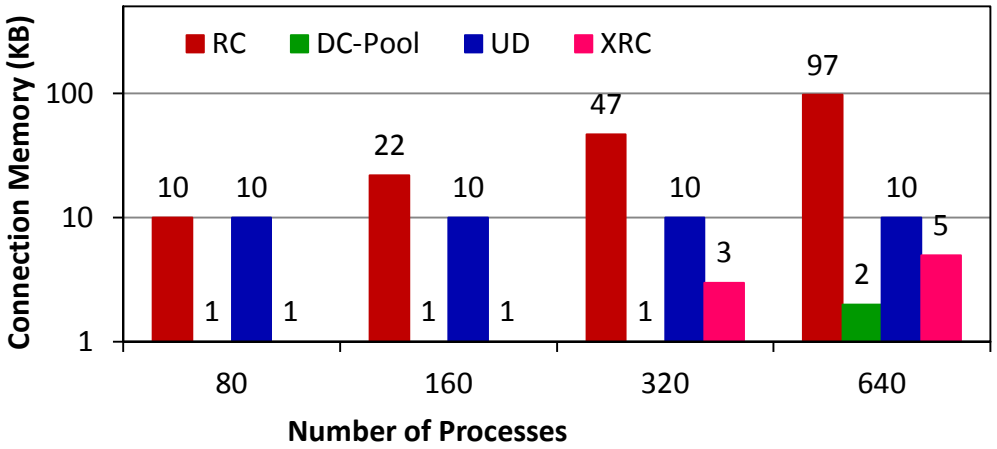
- **Refer to Running with Hybrid UD-RC/XRC section of MVAPICH2 user guide for more information**
- **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-650006.11**

Minimizing Memory Footprint by Direct Connect (DC) Transport

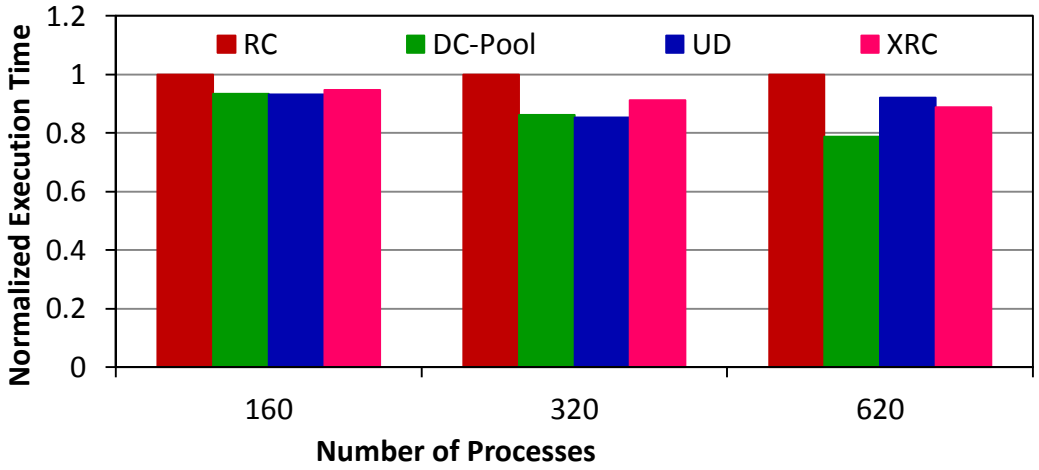


- Constant connection cost (*One QP for any peer*)
- Full Feature Set (RDMA, Atomics etc)
- Separate objects for send (DC Initiator) and receive (DC Target)
  - DC Target identified by "DCT Number"
  - Messages routed with (DCT Number, LID)
  - Requires same "DC Key" to enable communication
- Available since MVAPICH2-X 2.2a

Memory Footprint for Alltoall



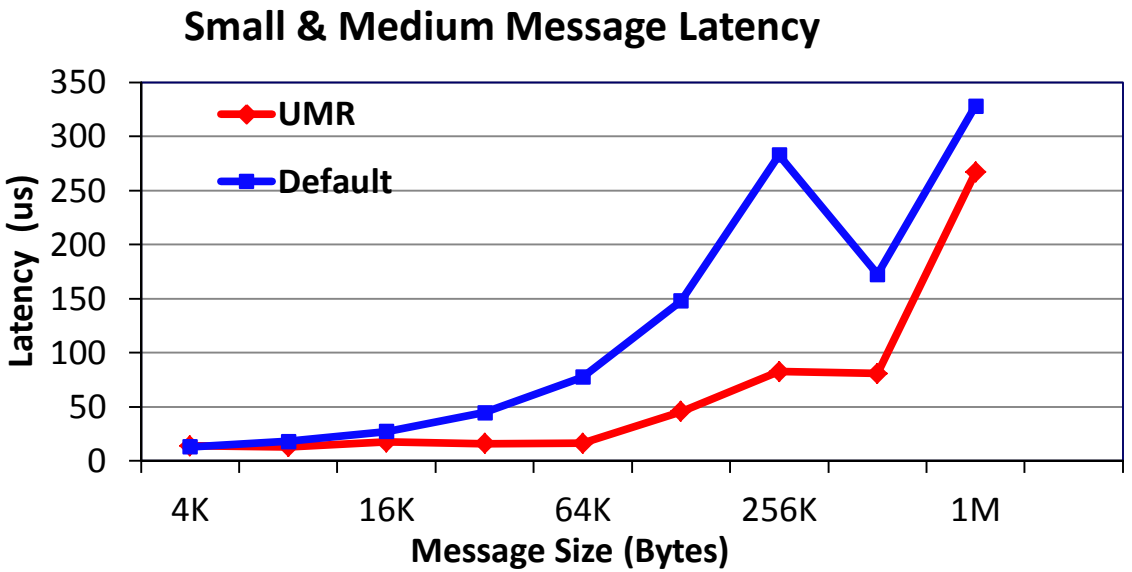
NAMD - Apoa1: Large data set

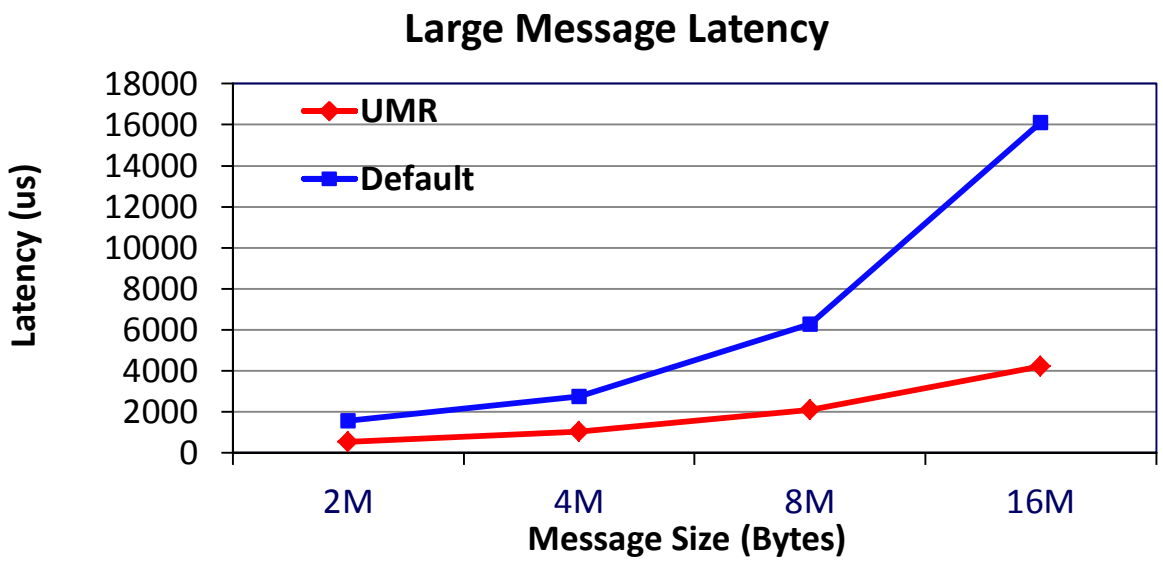


H. Subramoni, K. Hamidouche, A. Venkatesh, S. Chakraborty and D. K. Panda, Designing MPI Library with Dynamic Connected Transport (DCT) of InfiniBand : Early Experiences. IEEE International Supercomputing Conference (ISC '14)

User-mode Memory Registration (UMR)

- Introduced by Mellanox to support direct local and remote noncontiguous memory access
  - Avoid packing at sender and unpacking at receiver
- Available since MVAPICH2-X 2.2b



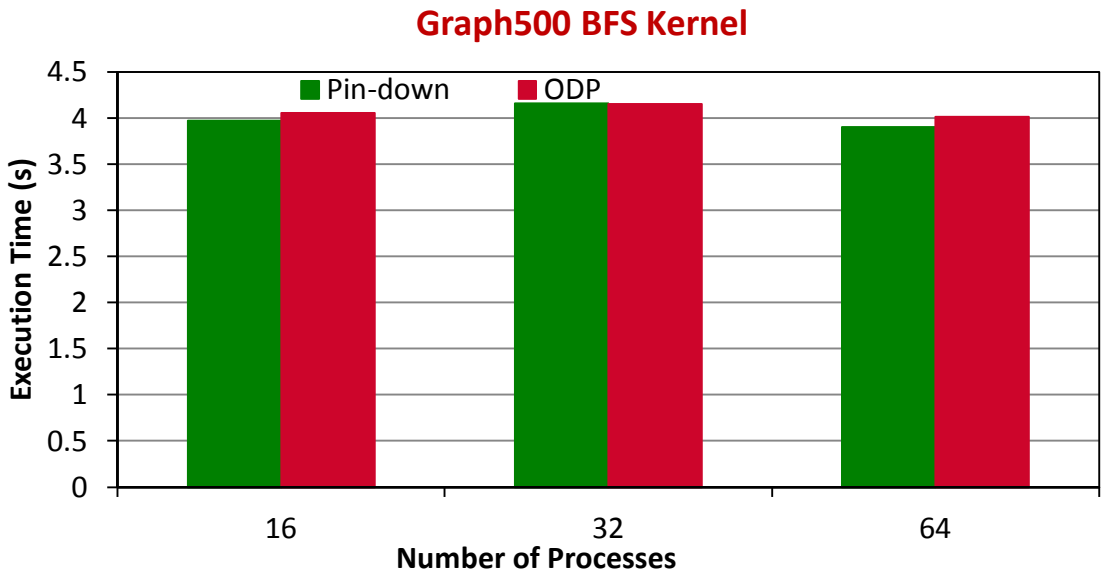


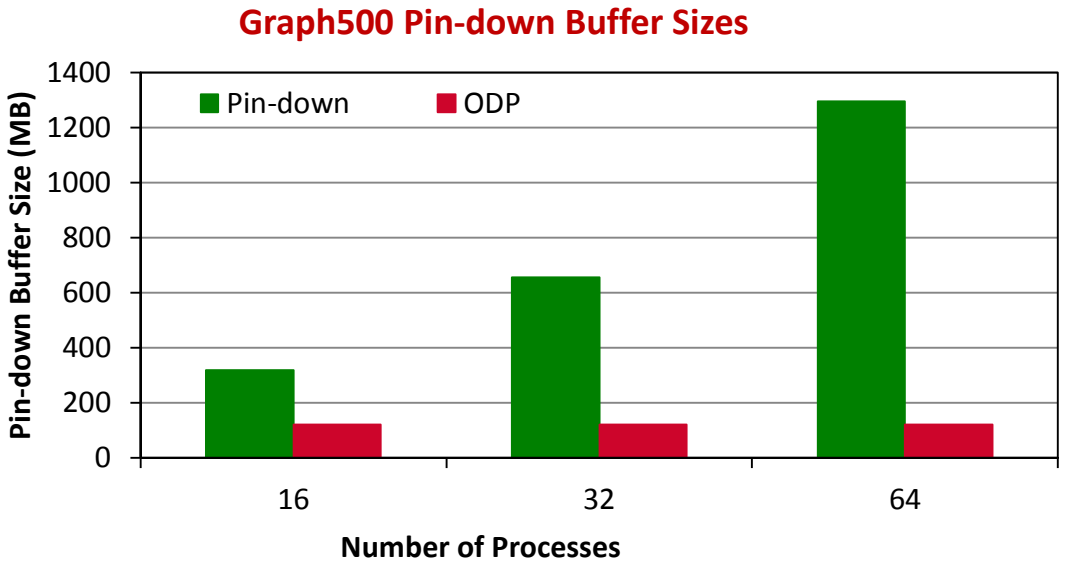
Connect-IB (54 Gbps): 2.8 GHz Dual Ten-core (IvyBridge) Intel PCI Gen3 with Mellanox IB FDR switch

M. Li, H. Subramoni, K. Hamidouche, X. Lu and D. K. Panda, High Performance MPI Datatype Support with User-mode Memory Registration: Challenges, Designs and Benefits, CLUSTER, 2015

On-Demand Paging (ODP)

- Introduced by Mellanox to support direct remote memory access without pinning
- Memory regions paged in/out dynamically by the HCA/OS
- Size of registered buffers can be larger than physical memory
- Will be available in future MVAPICH2 releases



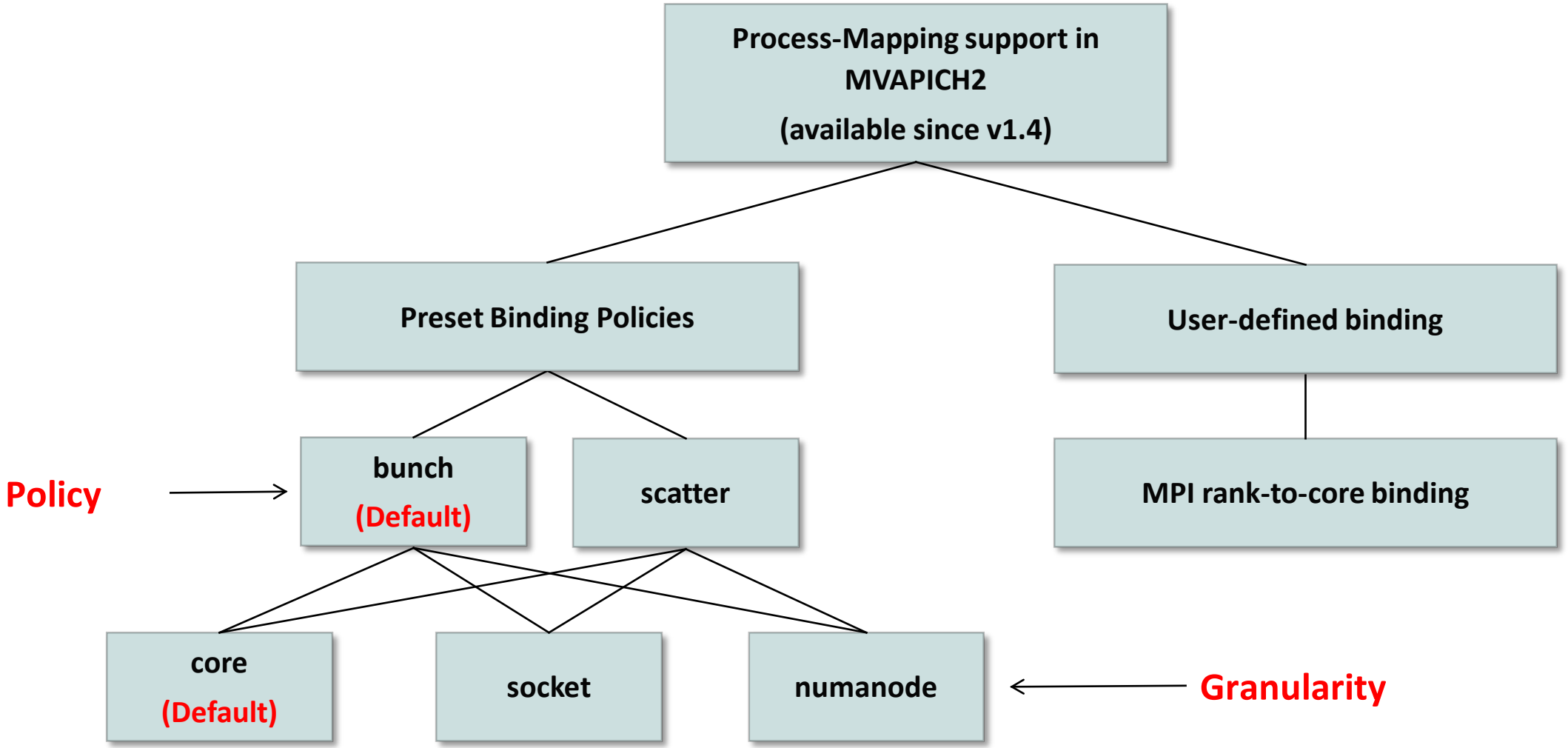


Connect-IB (54 Gbps): 2.6 GHz Dual Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR switch

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - **Process Mapping and Point-to-point Intra-node Protocols**
      - **Process to core mapping**
      - **Shared-memory and LiMIC2/CMA based Communication**
      - **Architecture-based Tuning**
    - MPI-3 RMA
    - Collectives
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

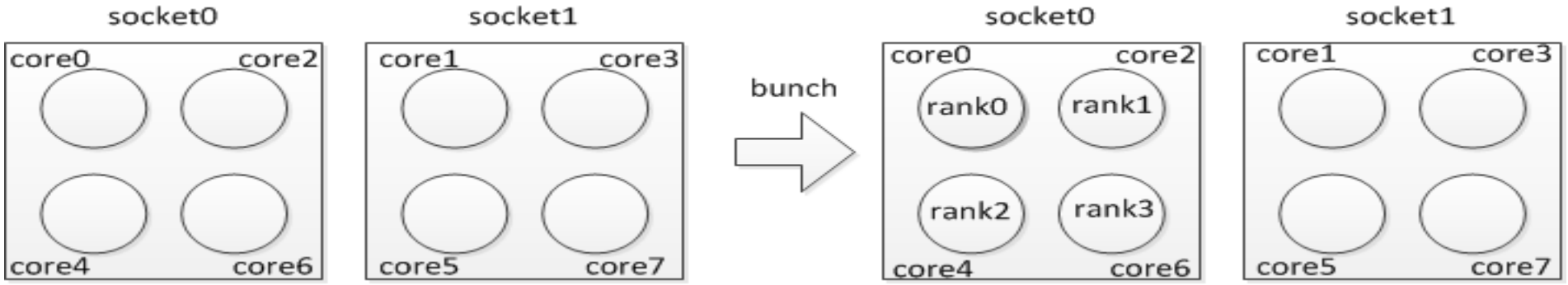
Process Mapping support in MVAPICH2



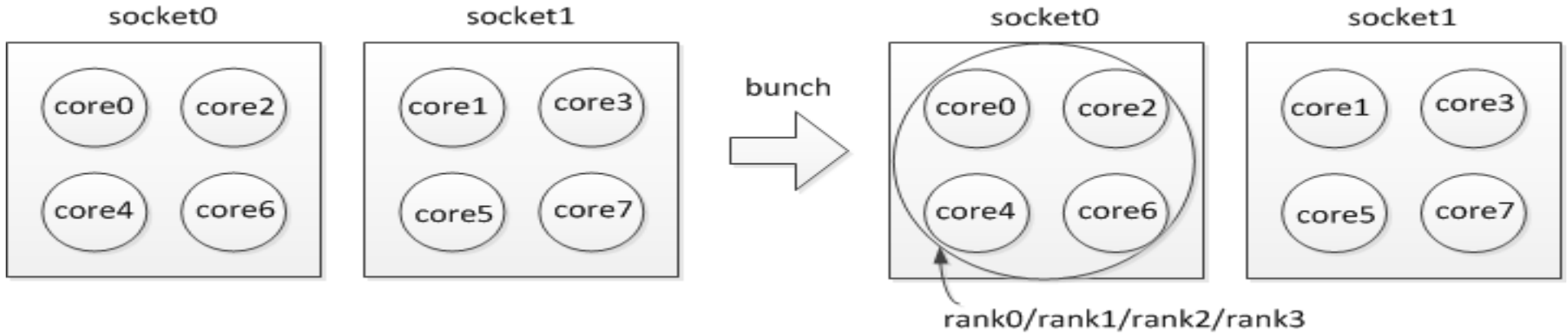
- MVAPICH2 detects processor architecture at job-launch

Preset Process-binding Policies – Bunch

- "Core" level "Bunch" mapping (Default)
  - MV2_CPU_BINDING_POLICY=bunch

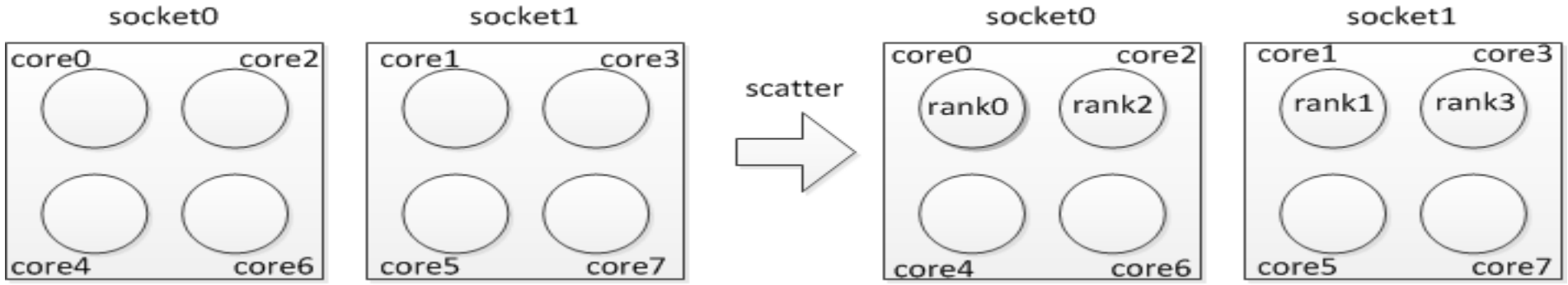


- "Socket/Numanode" level "Bunch" mapping
  - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=bunch

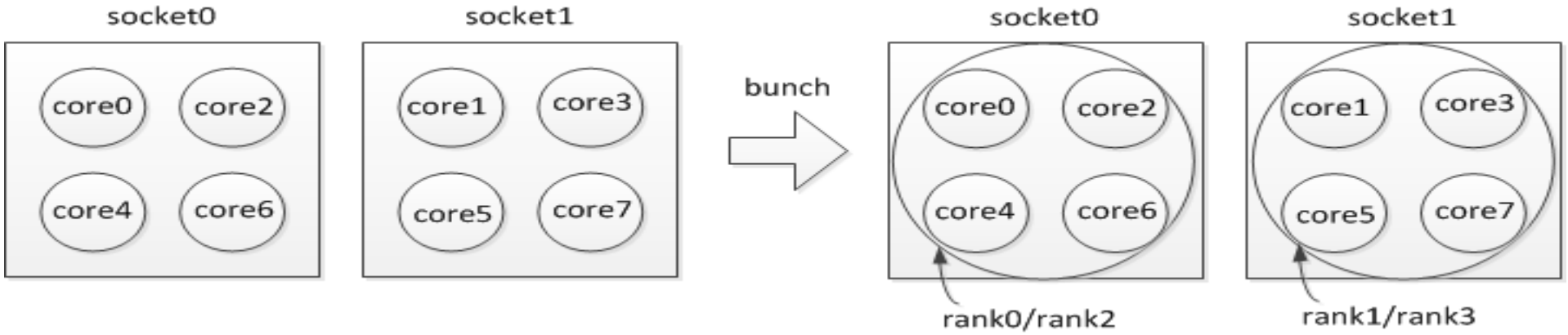


Preset Process-binding Policies – Scatter

- "Core" level "Scatter" mapping
  - MV2_CPU_BINDING_POLICY=scatter



- "Socket/Numanode" level "Scatter" mapping
  - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=scatter



User-Defined Process Mapping

- User has complete-control over process-mapping
- To run 4 processes on cores 0, 1, 4, 5:
  - $ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_MAPPING=0:1:4:5 ./a.out
- Use ',' or '-' to bind to a set of cores:
  - $mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0,2-4:1:5:6 ./a.out
- Is process binding working as expected?
  - **MV2_SHOW_CPU_BINDING=1**
    - Display CPU binding information
    - Launcher independent
    - Example
      - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=scatter

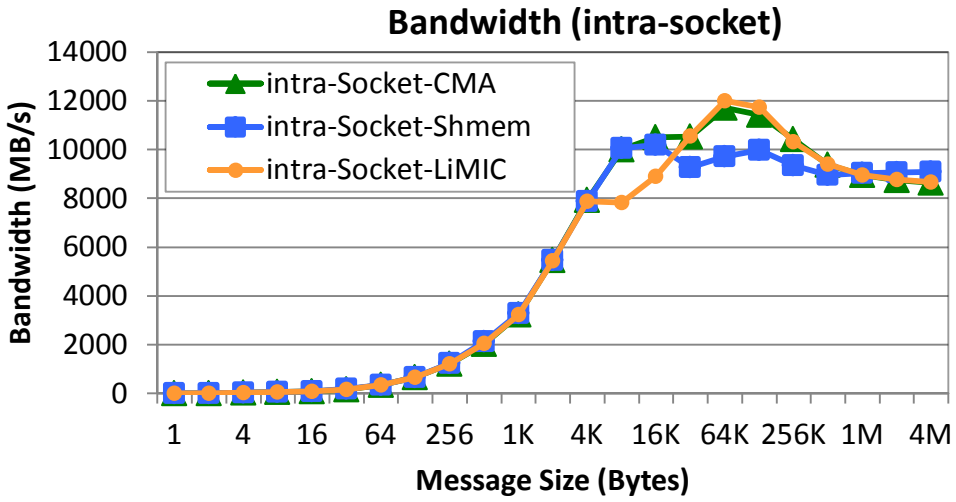
```
-------------CPU AFFINITY-------------
        RANK:0  CPU_SET:   0
        RANK:1  CPU_SET:   8
```

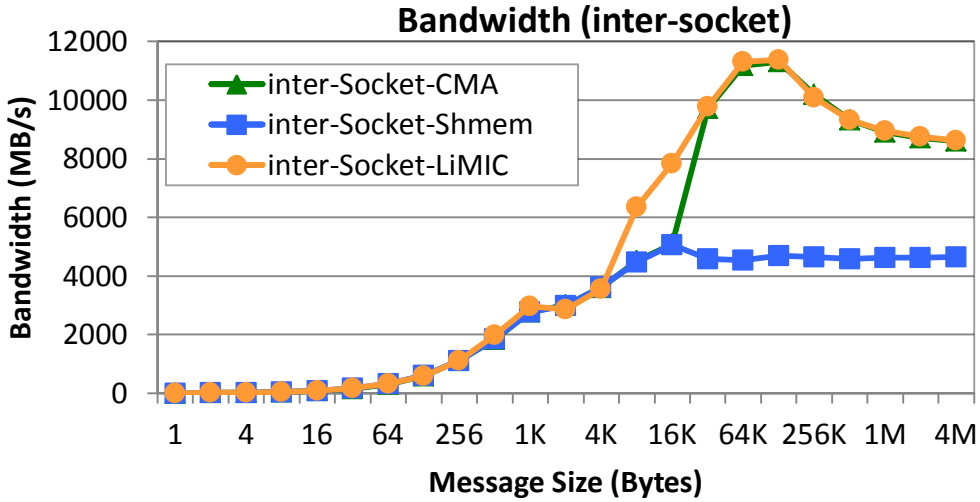
- **Refer to Running with Efficient CPU (Core) Mapping section of MVAPICH2 user guide for more information**
- **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-560006.5**

Intra-node Communication Support in MVAPICH2

- Shared-Memory based two-copy intra-node communication
  - Copy from the sender's user buffer to the shared buffer
  - Copy from the shared buffer to the receiver's user buffer
- LiMIC2 on modern multi-core platforms
  - Kernel-level module for achieving single copy intra-node communication
  - LiMIC2 is used for rendezvous protocol message size
  - LiMIC2 module is required
- CMA (Cross Memory Attach) support
  - Single copy intra-node communication through Linux syscalls
  - Available from Linux kernel 3.2

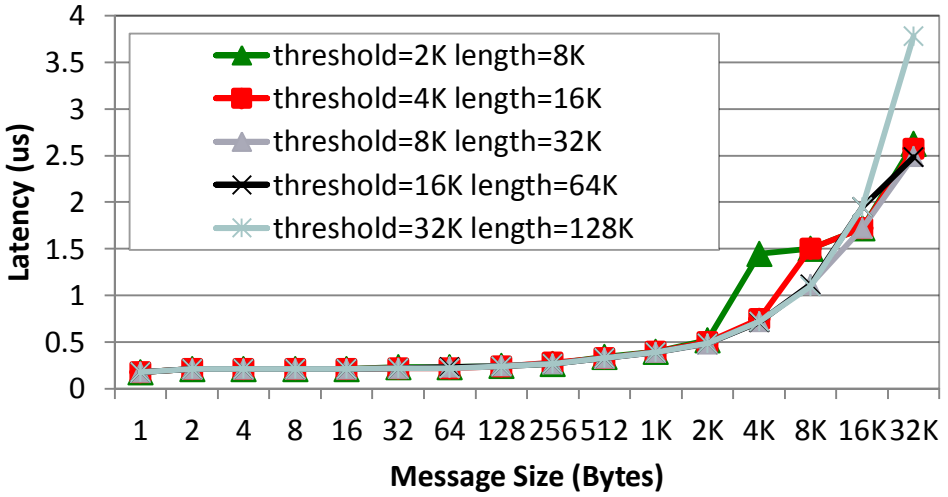
MVAPICH2 Two-Sided Intra-Node Tuning:
Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA)

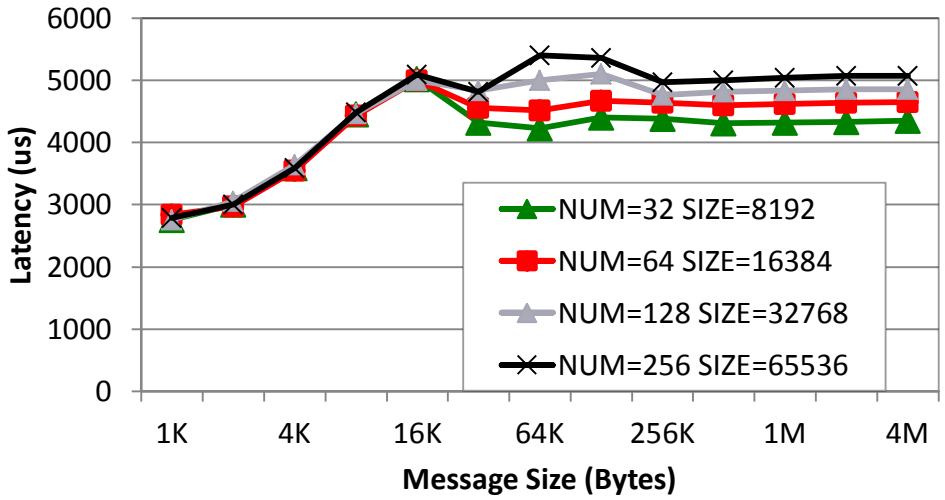




- LiMIC2:
  - configure the library with '--with-limic2'
  - mpirun_rsh –np 2 –hostfile hostfile a.out (To disable: MV2_SMP_USE_LIMIC2=0)
- CMA:
  - configure the library with '--with-cma'
  - mpirun_rsh –np 2 –hostfile hostfile a.out (To disable: MV2_SMP_USE_CMA=0)
- When both '--with-limic2' and '--with-cma' are included at the same time, LiMIC2 is chosen by default
- When neither '--with-limic2' or '--with-cma' is used during in configuration, shared-memory based design is chosen

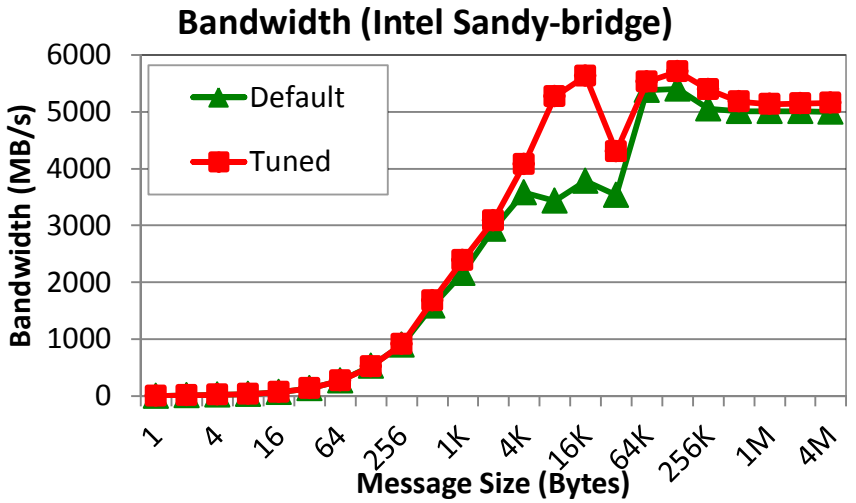
MVAPICH2 Two-Sided Intra-Node Tuning:
Shared-Memory based Runtime Parameters

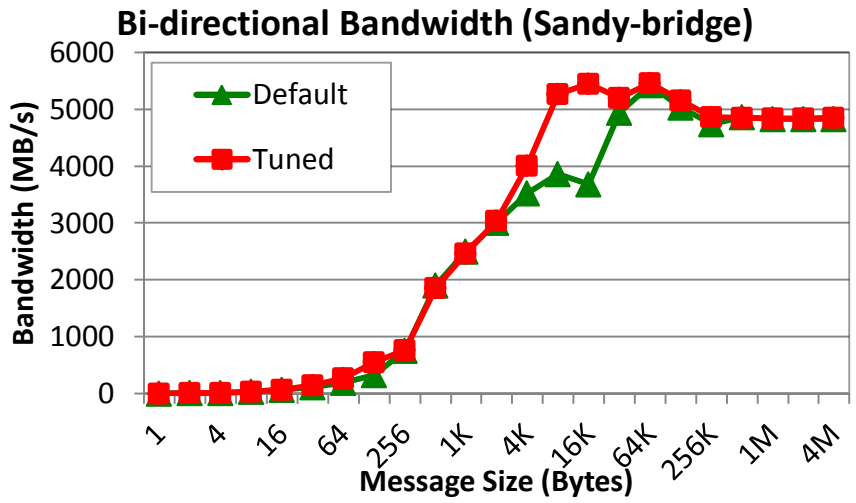


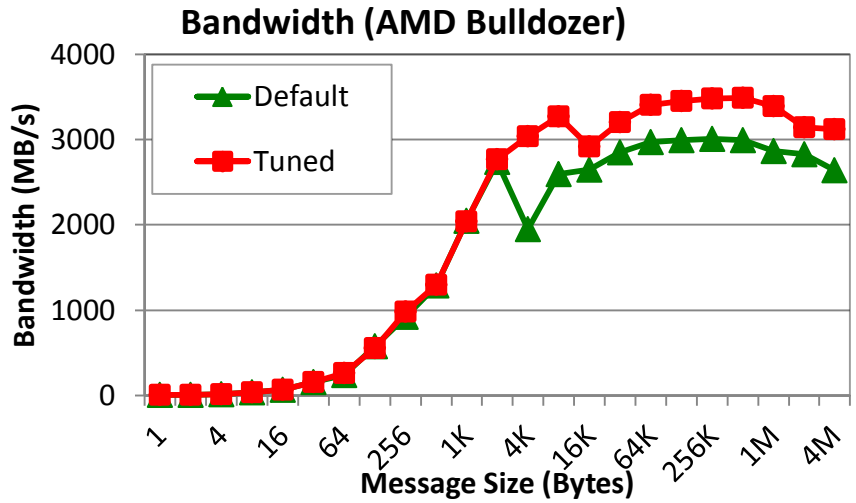


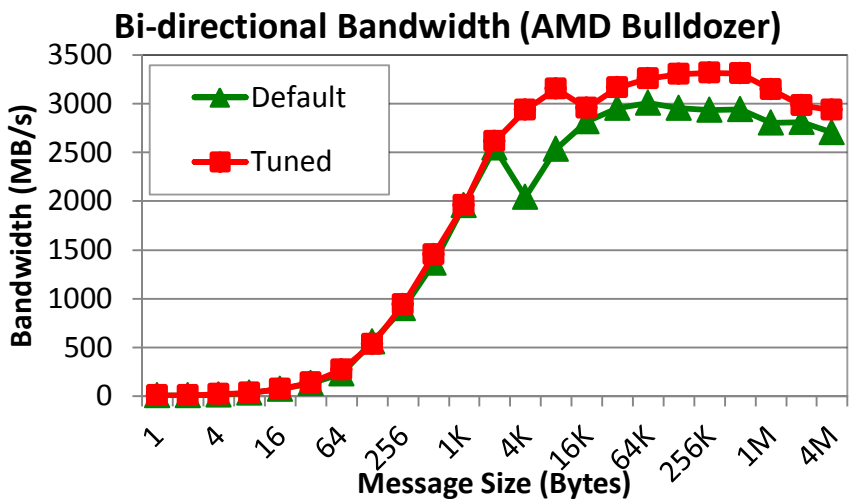
- Adjust eager threshold and eager buffer size:
  - mpirun_rsh –np 2 –hostfile hostfile MV2_SMP_EAGERSIZE=16K MV2_SMPI_LENGTH_QUEUE=64 a.out
  - Will affect the performance of small messages and memory footprint
- Adjust number of buffers and buffer size for shared-memory based Rendezvous protocol:
  - mpirun_rsh –np 2 –hostfile hostfile MV2_SMP_SEND_BUFFER=32 MV2_SMP_SEND_BUFF_SIZE=8192 a.out
  - Will affect the performance of large messages and memory footprint

Impact of Architecture-Specific Tuning







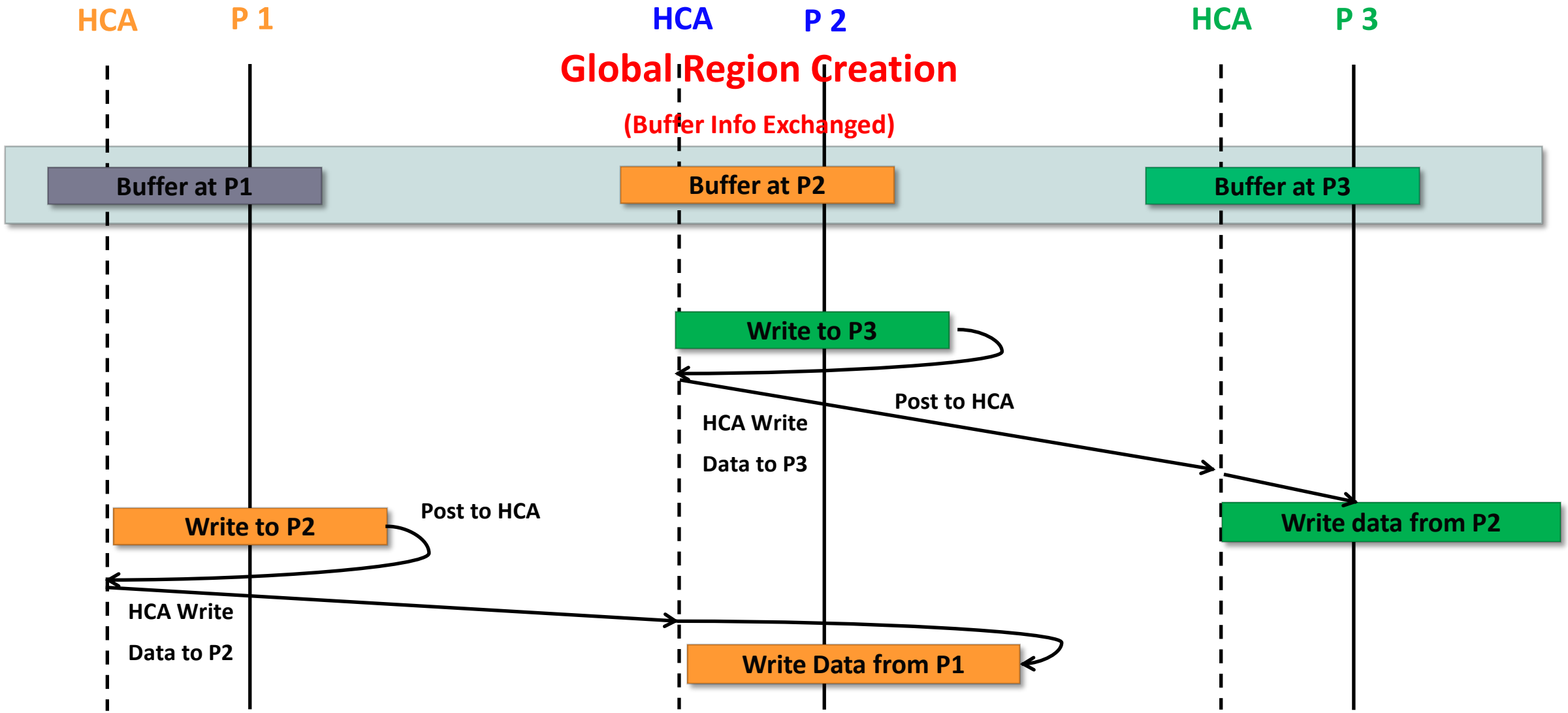


- Architecture-specific tuning is executed for new architectures and new designs introduced into MV2
- MV2_SMP_EAGERSIZE and MV2_SMP_SEND_BUFF_SIZE are updated from Default (1.8) to Tuned (1.9)

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Process Mapping and Point-to-point Intra-node Protocols
    - **MPI-3 RMA**
      - **InterNode Communication**
      - **IntraNode Communication**
      - **MPI-3 RMA Model**
    - Collectives
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

One-sided Communication Model



Internode One-sided Communication: Direct RDMA-based Designs

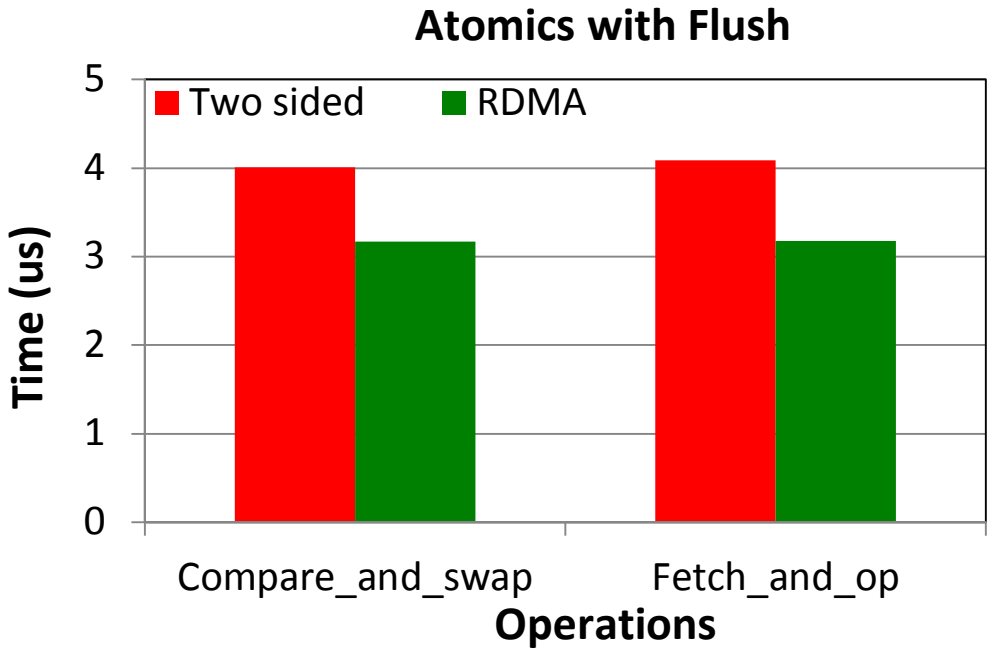
- MPI RMA offers one-sided communication
  - Separates communication and synchronization
  - Reduces synchronization overheads
  - Better computation and communication overlap
- Most MPI libraries implement RMA over send/recv calls
- MVAPICH2 offers direct RDMA-based implementation
  - Put/Get implemented as RDMA Write/Read
  - Better performance
  - Better computation-communication overlap

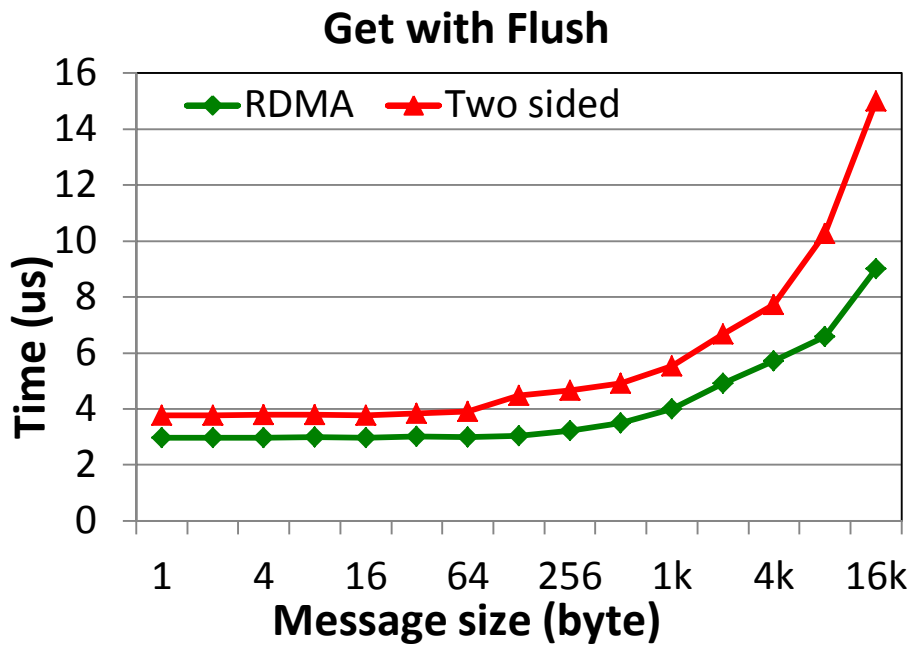
Parameter	Significance	Default	Notes
MV2_USE_RDMA_ONE_SIDED	• Enable / Disable RDMA-based designs	1 (Enabled)	• Disable only for debugging purposes

- **Refer to MV2_USE_RDMA_ONE_SIDED section of MVAPICH2 user guide for more information**
- **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-24700011.87**

MPI-3 RMA Model: Performance

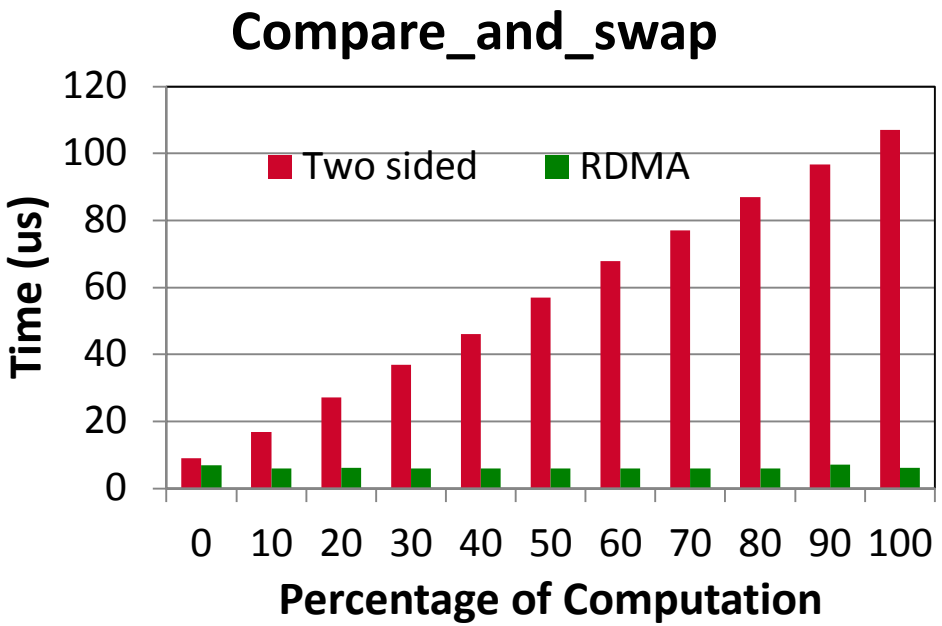
- RDMA-based and truly one-sided implementation of MPI-3 RMA in progress

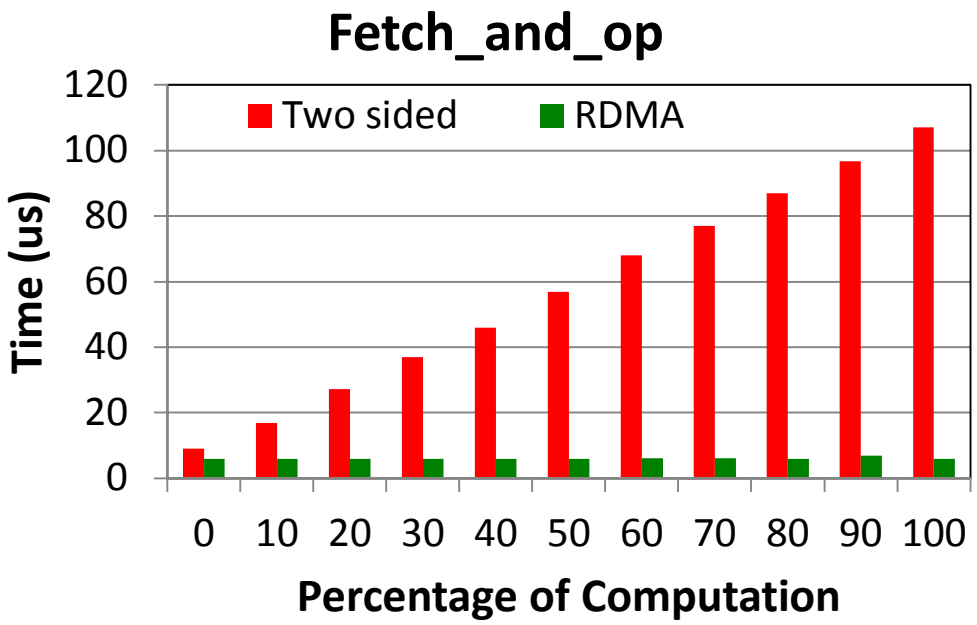




- MVAPICH2-2.1 and OSU micro-benchmarks (OMB v4.1)
- Better performance for MPI_Compare_and_swap and MPI_Fetch_and_op and MPI_Get performance with RDMA-based design

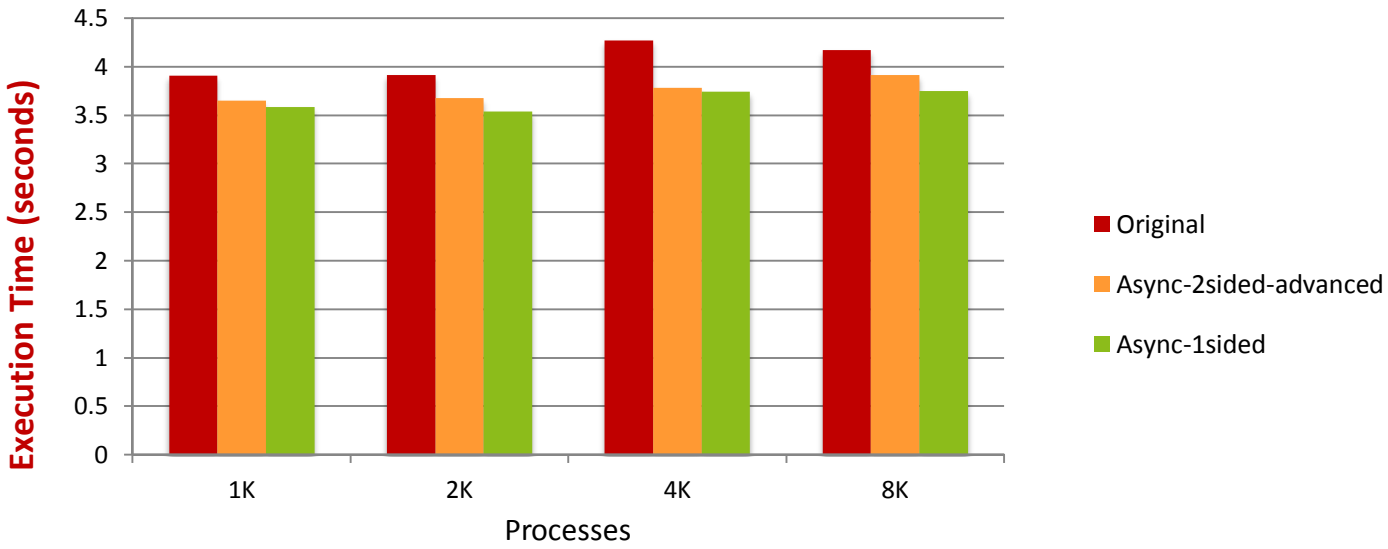
MPI-3 RMA Model: Overlap

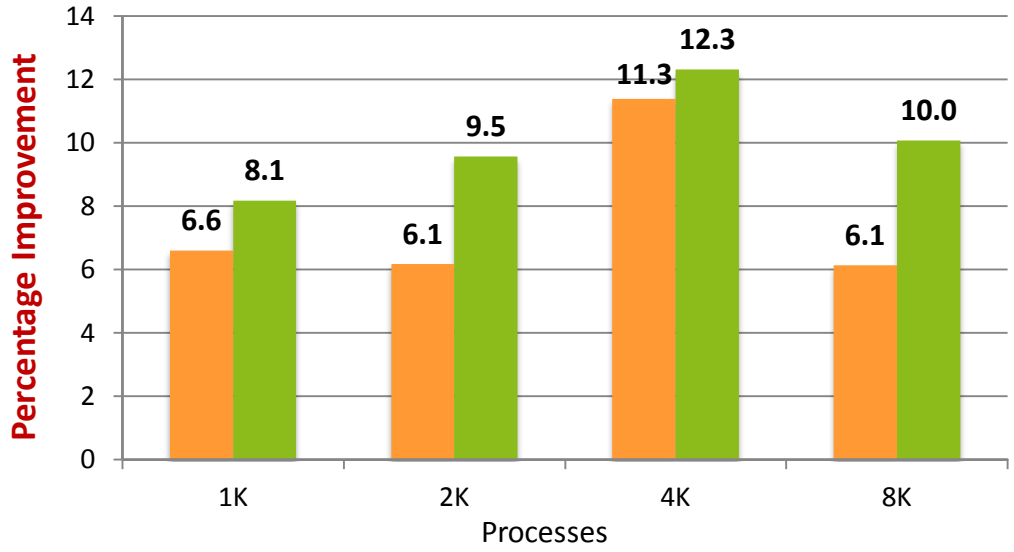




- Process 0 is busy in computation, Process 1 performance atomic operations at P0
- These benchmarks show the latency of atomic operations. For RDMA based design, the atomic latency at P1 remains consistent even as the busy time at P0 increases

Performance of AWP-ODC using MPI-3-RMA





- Experiments on TACC Ranger cluster 64x64x64 data grid per process – 25 iterations – 32KB messages
- On 4K processes
  - 8% with 2sided basic, 11% with 2sided advanced, 12% with RMA
- On 8K processes
  - 2% with 2sided basic, 6% with 2sided advanced, 10% with RMA

S. Potluri, P. Lai, K. Tomko, S. Sur, Y. Cui, M. Tatineni, K. Schulz, W. Barth, A. Majumdar and D. K. Panda, Quantifying Performance Benefits of Overlap using MPI-2 in a Seismic Modeling Application, ICS '10.

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Process Mapping and Point-to-point Intra-node Protocols
    - MPI-3 RMA
    - **Collectives**
      - **Overview of Collective Communication Operations**
      - **Improved Hardware Based Collectives in MVAPICH2**
      - **Tuning Collective Communication Operations in MVAPICH2**
    - OMB
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

Collective Communication Operations

- Involves all processes in the communicator
  - Unexpected behavior if some processes do not participate
- Different types
  - Synchronization
    - Barrier
  - Data movement
    - Broadcast, Scatter, Gather, Alltoall
  - Collective computation
    - Reduction
- Data movement collectives can use pre-defined (int, float, char…) or user-defined datatypes (struct, union etc)

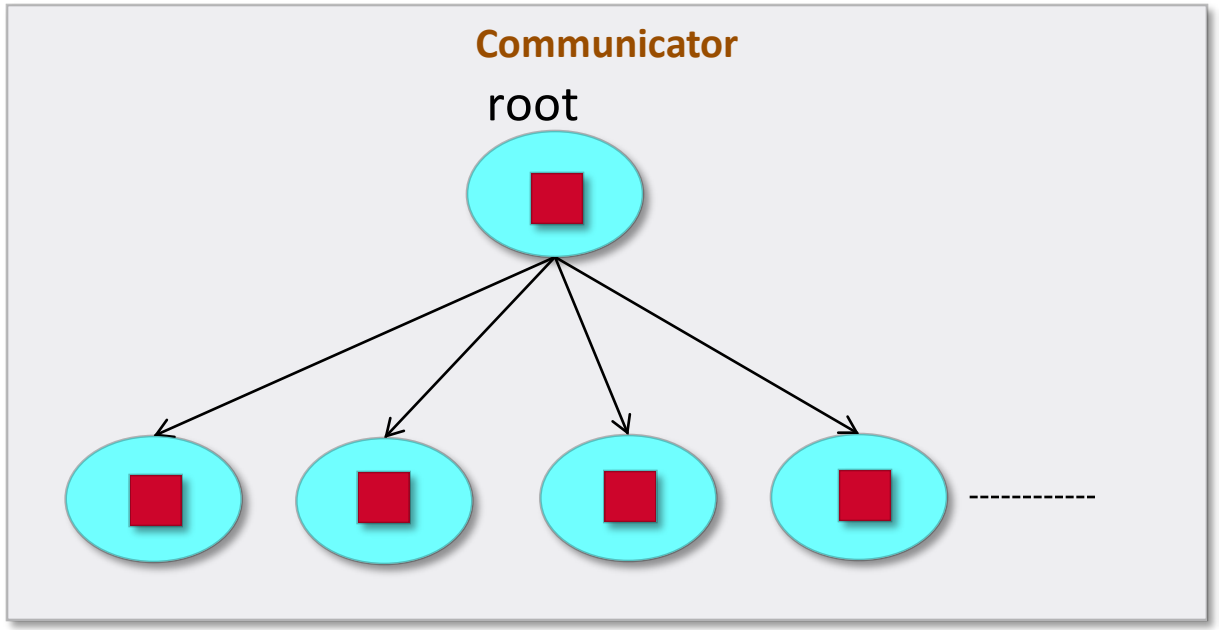
Sample Collective Communication Routines

- Broadcast a message from process with rank of "root" to all other processes in the communicator

```
int MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root,  MPI_Comm comm )
```

Input-only Parameters	
Parameter	**Description**
count	Number of entries in buffer
datatype	Data type of buffer
root	Rank of broadcast root
comm	Communicator handle

Input/Output Parameters	
Parameter	**Description**
buffer	Starting address of buffer



Sample Collective Communication Routines (Cont'd)

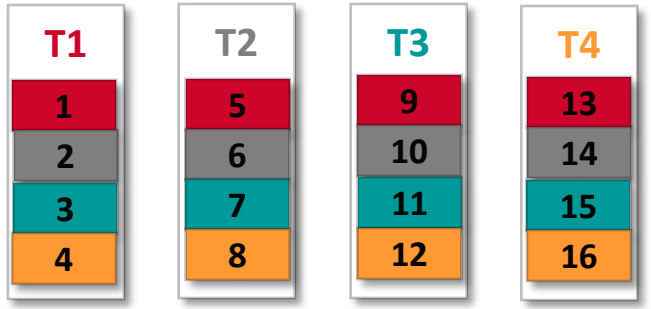
- Sends data from all processes to all processes

int MPI_Alltoall (const void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)

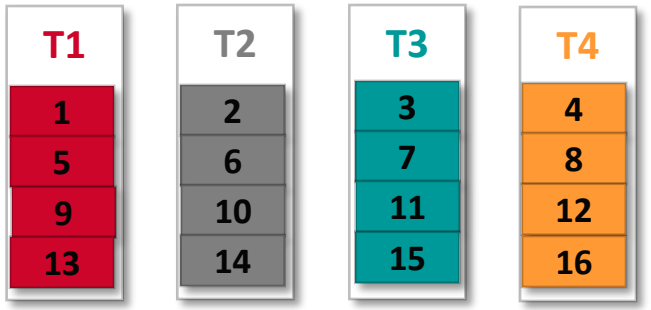
Input-only Parameters	
Parameter	**Description**
sendbuf	Starting address of send buffer
sendcount	Number of elements to send to each process
sendtype	Data type of send buffer elements
recvcount	Number of elements received from any process
recvtype	Data type of receive buffer elements
comm	Communicator handle

Input/Output Parameters	
Parameter	**Description**
recvbuf	Starting address of receive buffer

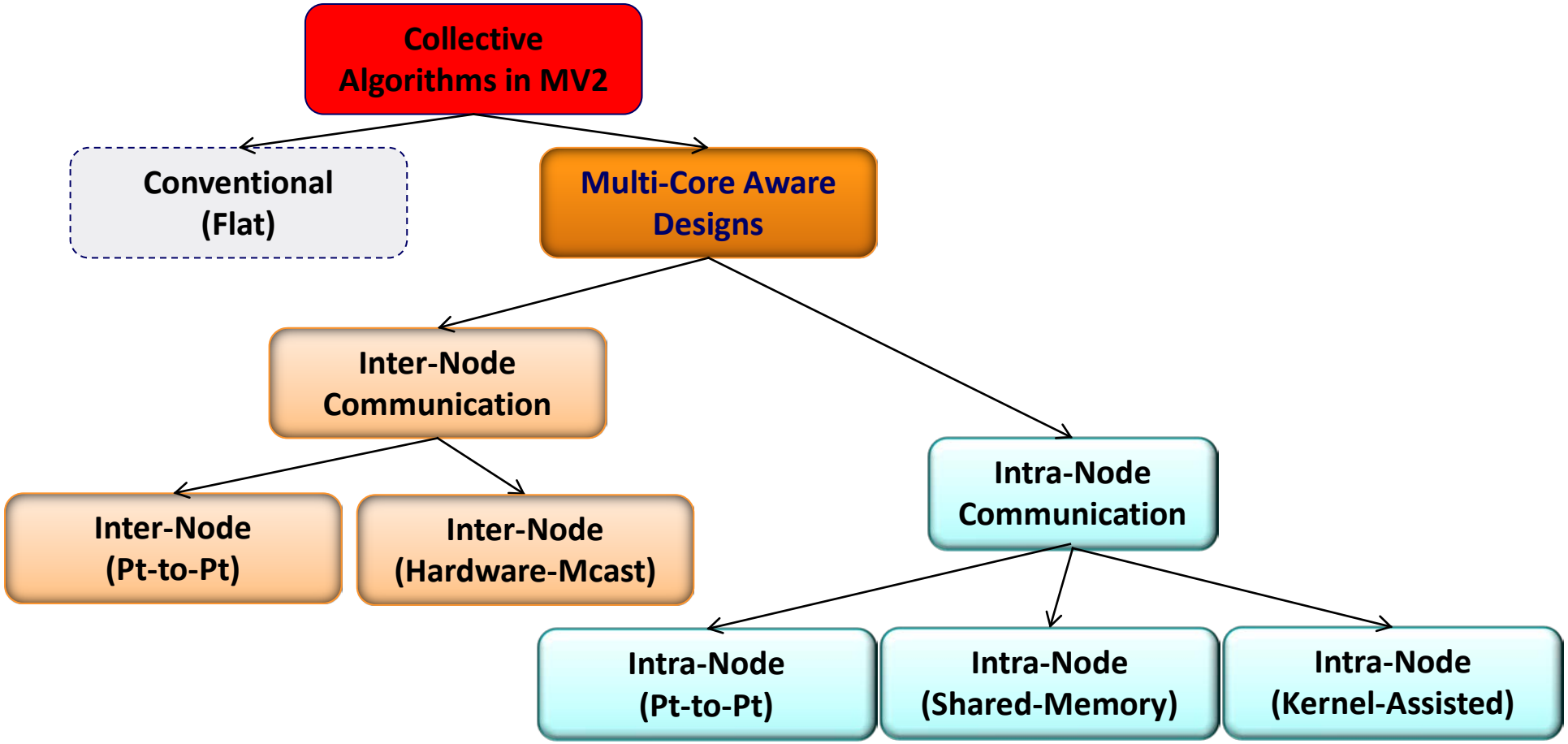
Sendbuf (Before)



Recvbuf (After)



Collective Communication in MVAPICH2

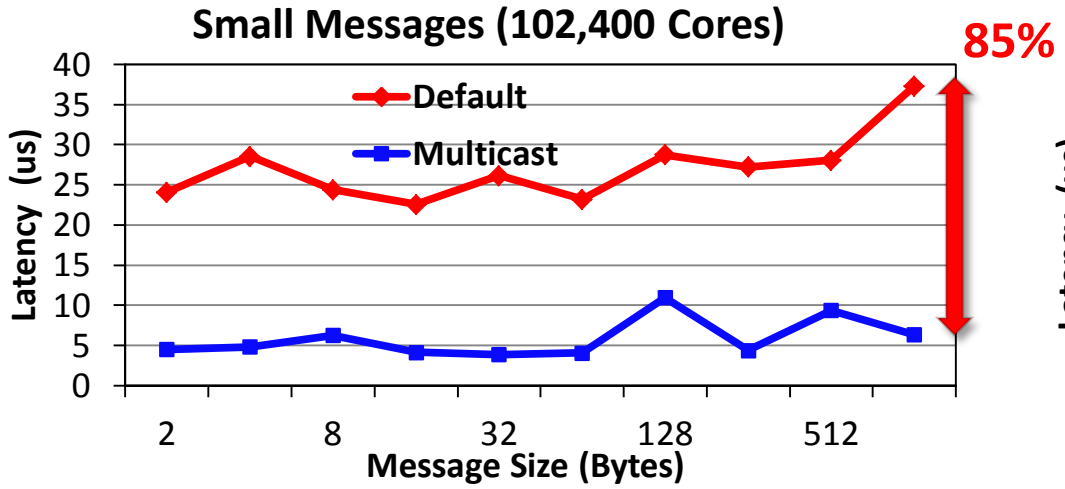


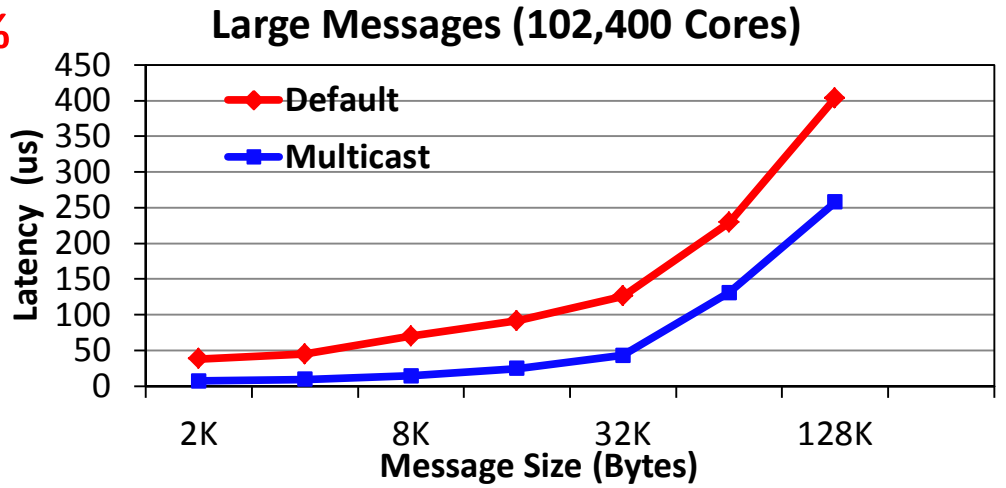
Run-time flags:

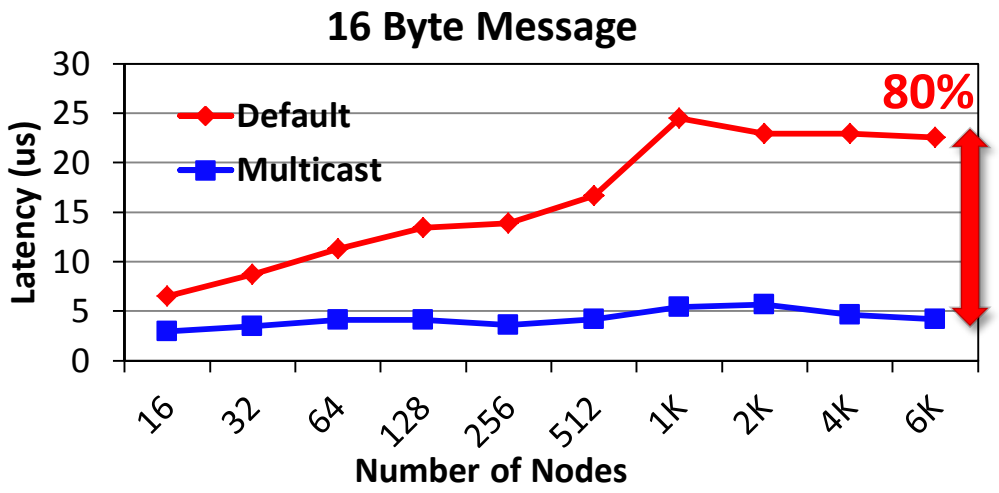
All shared-memory based collectives : MV2_USE_SHMEM_COLL (Default: ON)

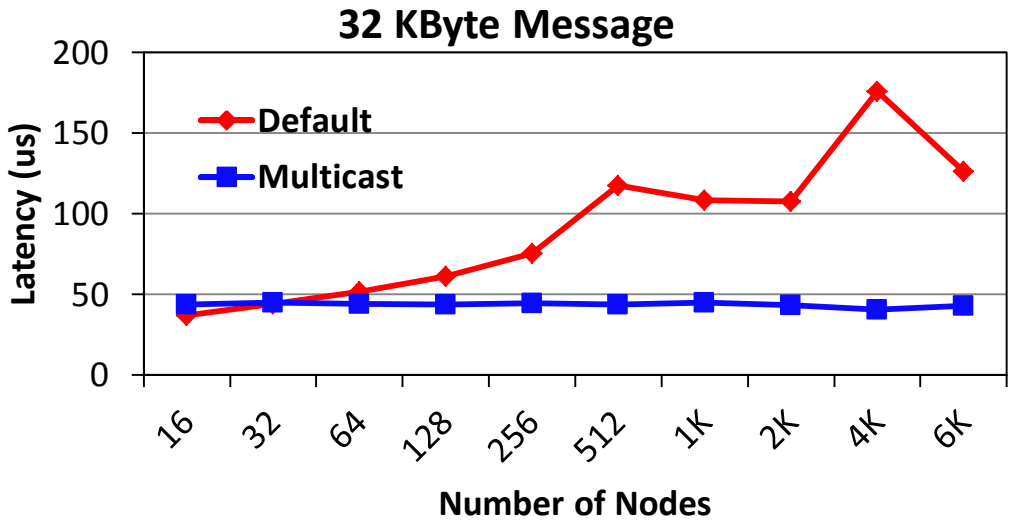
Hardware Mcast-based collectives : MV2_USE_MCAST (Default : OFF)

Hardware Multicast-aware MPI_Bcast on TACC Stampede



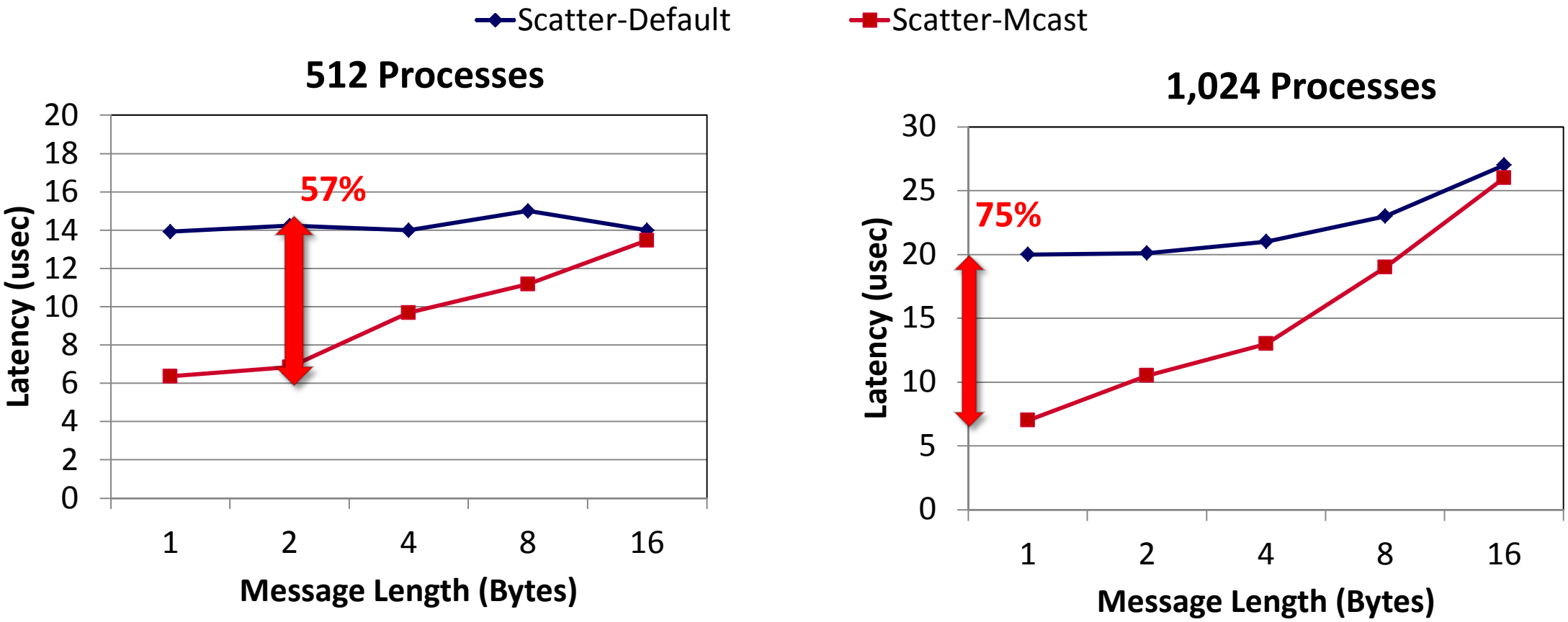






- MCAST-based designs improve latency of MPI_Bcast by up to **85%**
- Use MV2_USE_MCAST=1 to enable MCAST-based designs

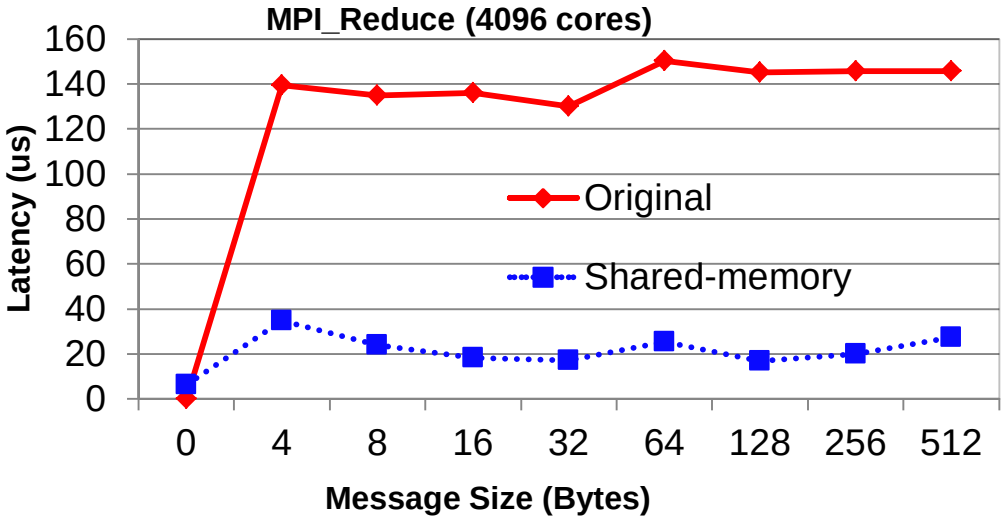
MPI_Scatter - Benefits of using Hardware-Mcast



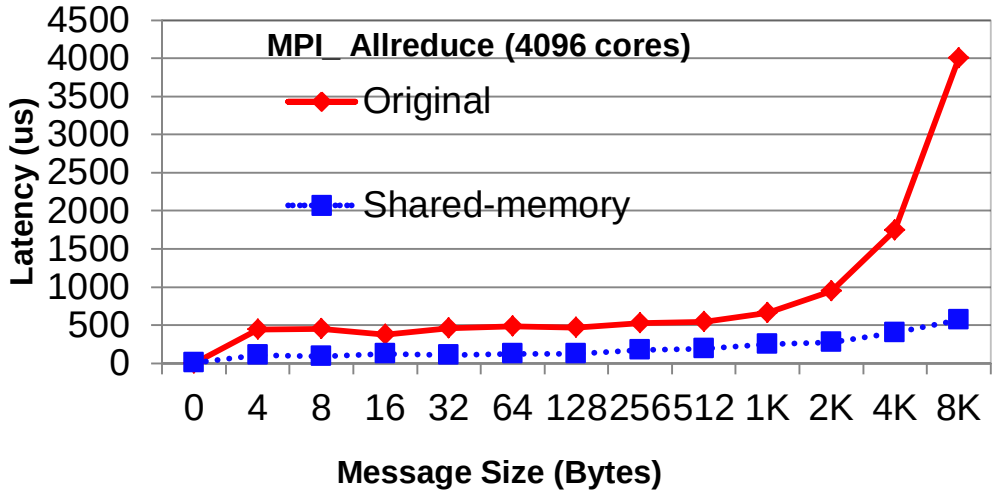
- Enabling MCAST-based designs for MPI_Scatter improves small message up to **75%**
- Use MV2_USE_MCAST=1 to enable MCAST-based designs

Shared-memory Aware Collectives

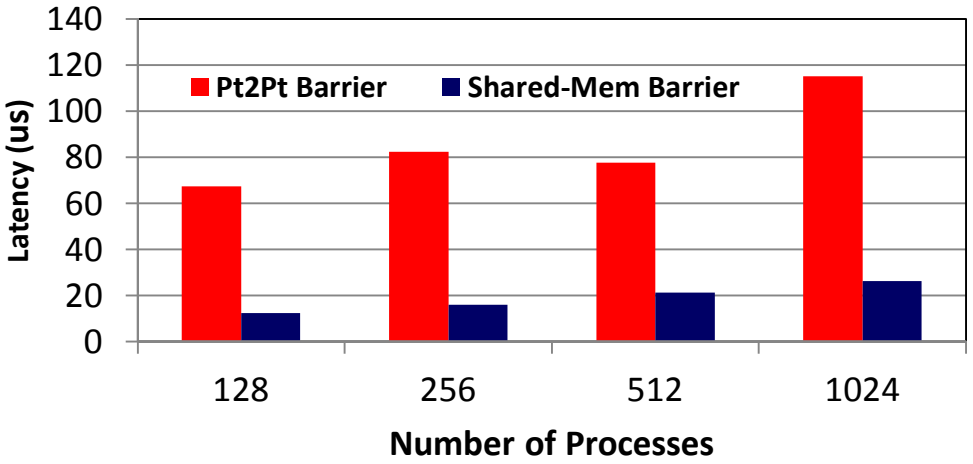
- MVAPICH2 Reduce/Allreduce with 4K cores on TACC Ranger (AMD Barcelona, SDR IB)



MV2_USE_SHMEM_REDUCE=0/1



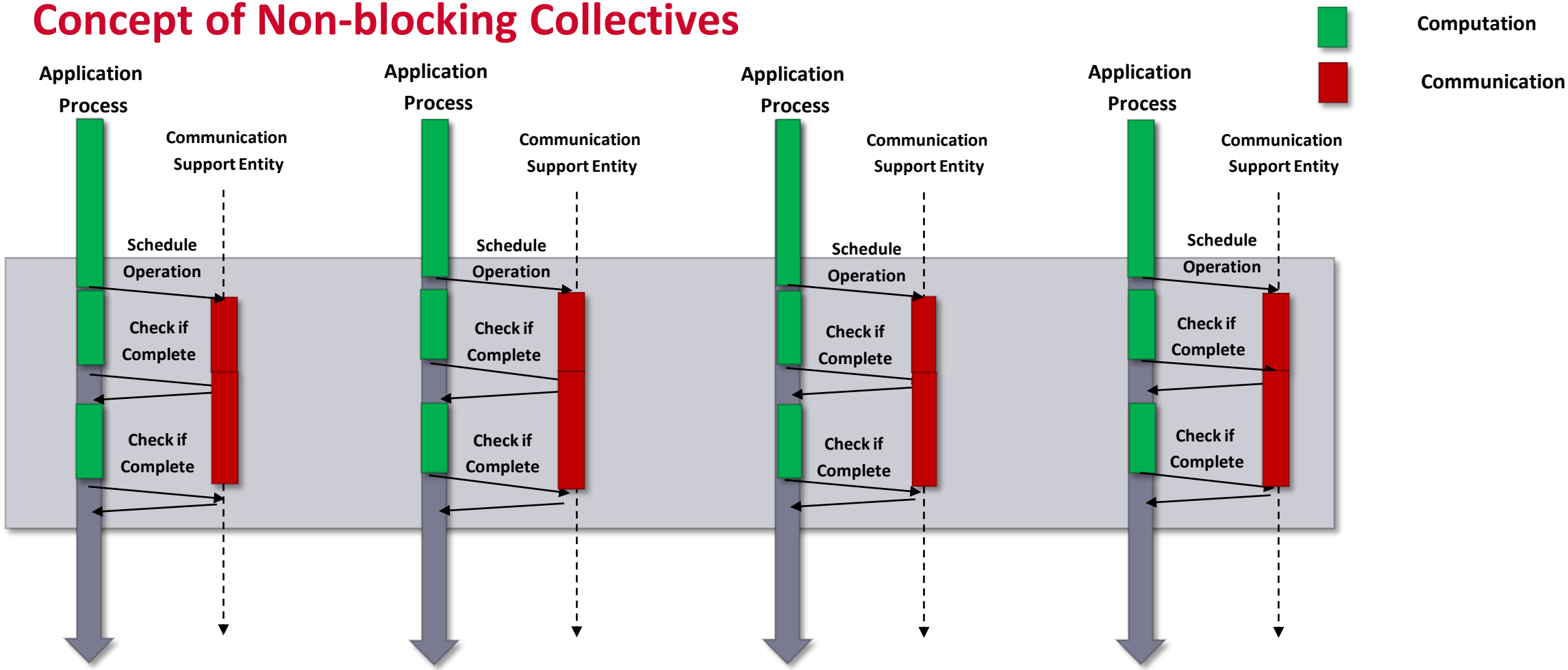
MV2_USE_SHMEM_ALLREDUCE=0/1



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

MV2_USE_SHMEM_BARRIER=0/1

Concept of Non-blocking Collectives



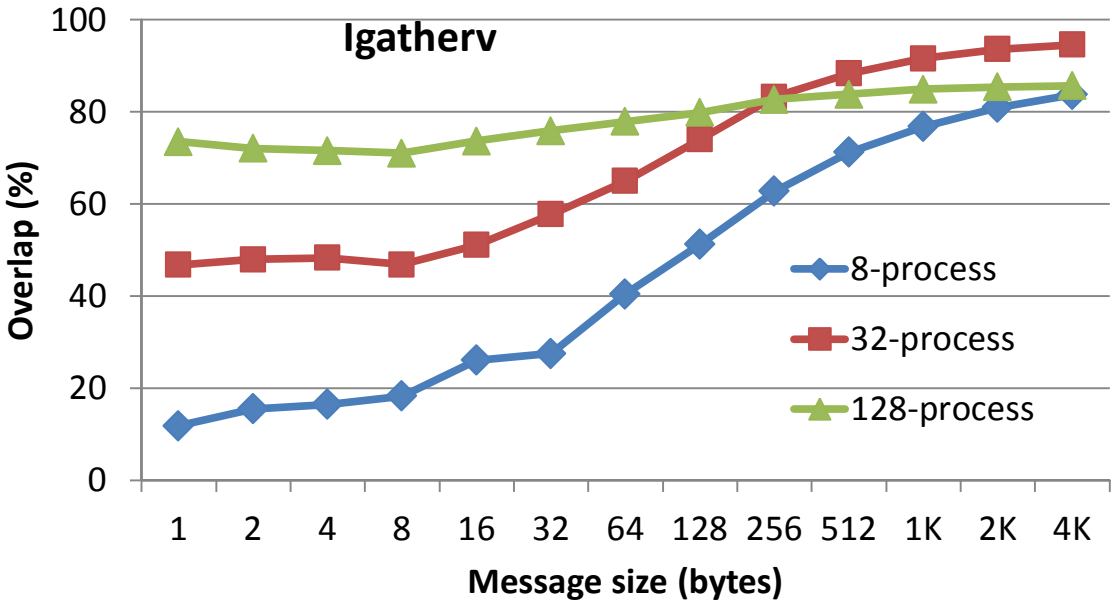
- Application processes schedule collective operation
- Check periodically if operation is complete
- **Overlap of computation and communication => Better Performance**
- *Catch: Who will progress communication*

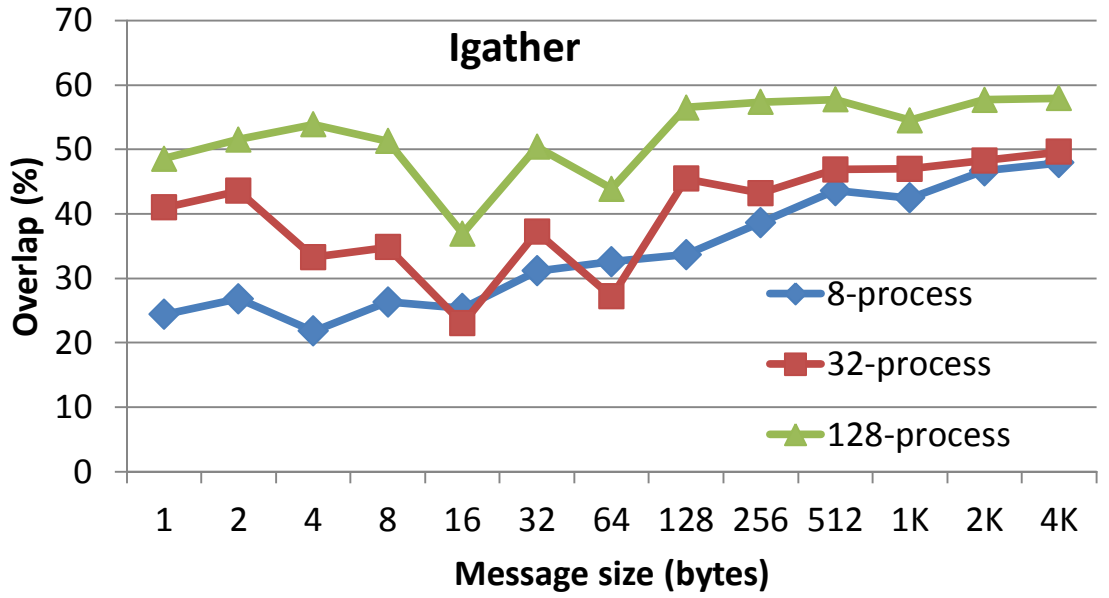
Non-blocking Collective (NBC) Operations

- Enables overlap of computation with communication
- Non-blocking calls do not match blocking collective calls
  - MPI may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: Optimized for latency
  - Non-blocking collectives: Optimized for overlap
- A process calling a NBC operation
  - Schedules collective operation and immediately returns
  - Executes application computation code
  - Waits for the end of the collective
- The communication progress by
  - Application code through MPI_Test
  - Network adapter (HCA) with hardware support
  - Dedicated processes / thread in MPI library
- There is a non-blocking equivalent for each blocking operation
  - Has an "I" in the name
    - MPI_Bcast -> MPI_Ibcast; MPI_Reduce -> MPI_Ireduce

Strong Overlap Example Benchmarks

- Pure case = Igatherv+Wait
- Overlapped case = Igatherv+compute+Wait
- Igather and Igatherv show good overlap due to combination of use of eager protocol and one-sided designs through InfiniBand

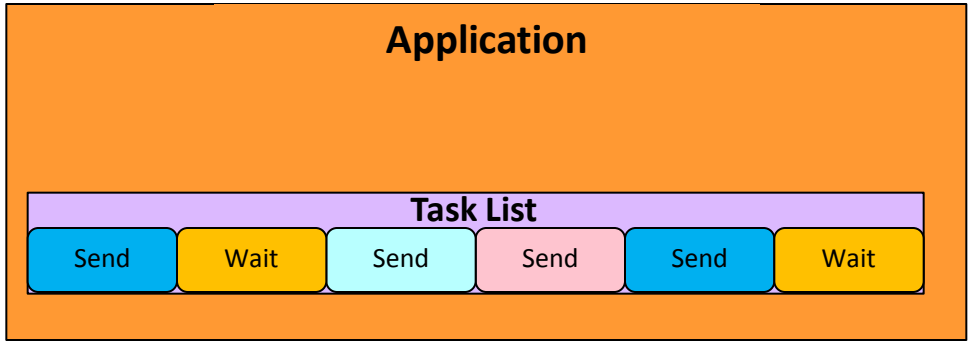




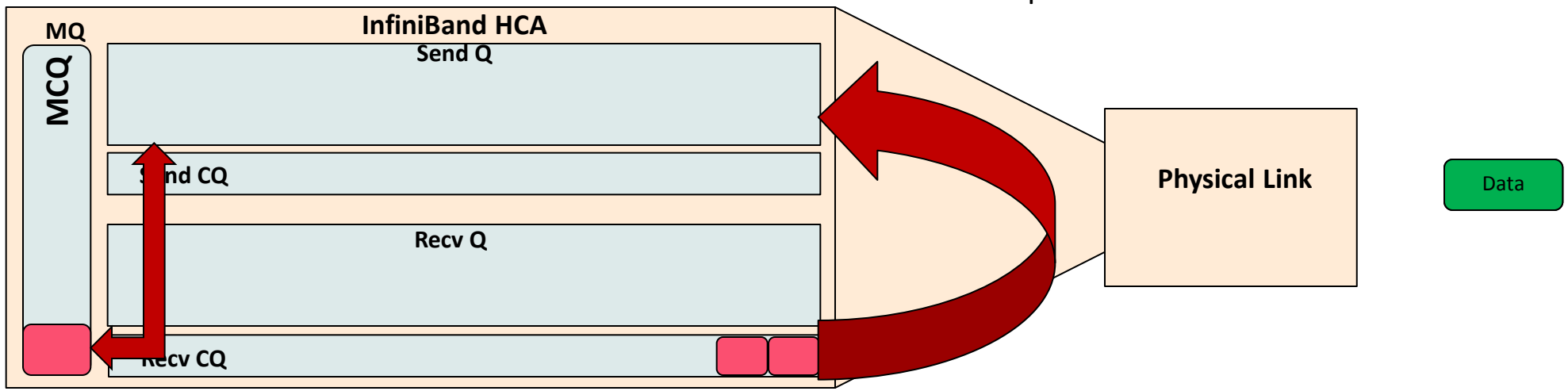
How do I write applications with NBC?

```
void main()
{
    MPI_Init()
    …..
    MPI_Ialltoall(…)
    Computation that does not depend on result of Alltoall
    MPI_Test(for Ialltoall) /* Check if complete (non-blocking) */
    Computation that does not depend on result of Alltoall
    MPI_Wait(for Ialltoall) /* Wait till complete (Blocking) */
    …
    MPI_Finalize()
}
```

Collective Offload Support in ConnectX InfiniBand Adapter (Recv followed by Multi-Send)



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



- **Available in MVAPICH2-X 2.2rc1**

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - **MVAPICH2**
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Process Mapping and Point-to-point Intra-node Protocols
    - MPI-3 RMA
    - Collectives
    - MPI_T Support
    - **OMB**
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

OSU Microbenchmarks

- Available since 2004
- Suite of microbenchmarks to study communication performance of various programming models
- Benchmarks available for the following programming models
  - Message Passing Interface (MPI)
  - Partitioned Global Address Space (PGAS)
    - Unified Parallel C (UPC)
    - Unified Parallel C++ (UPC++)
    - OpenSHMEM
- Benchmarks available for multiple accelerator based architectures
  - Compute Unified Device Architecture (CUDA)
  - OpenACC Application Program Interface
- Part of various national resource procurement suites like NERSC-8 / Trinity Benchmarks
- Please visit the following link for more information
  - http://mvapich.cse.ohio-state.edu/benchmarks/

OSU Microbenchmarks v5.3.1

- Released on 08/08/16
- Introduce new UPC++ Benchmarks
  - osu_upcxx_allgather, osu_upcxx_alltoall, osu_upcxx_async_copy_get, osu_upcxx_async_copy_put, osu_upcxx_bcast, osu_upcxx_gather, osu_upcxx_reduce, osu_upcxx_scatter
- Benchmarks for MPI
  - Point-to-Point Benchmarks (blocking and non-blocking)
  - Collective Benchmarks (blocking and non-blocking)
  - One-sided Benchmarks
  - Startup Benchmarks
- Benchmarks for UPC
  - Point-to-Point Benchmarks
  - Collective Benchmarks
- Benchmarks for UPC++
  - Point-to-Point Benchmarks
  - Collective Benchmarks
- Benchmarks for OpenSHMEM
  - Point-to-Point Benchmarks
  - Collective Benchmarks
- MPI Benchmarks for CUDA and OpenACC
  - Point-to-Point Benchmarks (blocking and non-blocking)
  - Collective Benchmarks (blocking and non-blocking)
  - One-sided Benchmarks

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - Basic CUDA-Aware Support
    - Support for Efficient Small Message Communication with GPUDirect RDMA
    - Multi-rail Support
    - Support for Efficient Intra-node Communication using CUDA IPC
    - MPI Datatype Support
    - Support for OpenACC Constructs
    - CUDA Support in OMB
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

GPU-Aware MPI Library: MVAPICH2-GPU

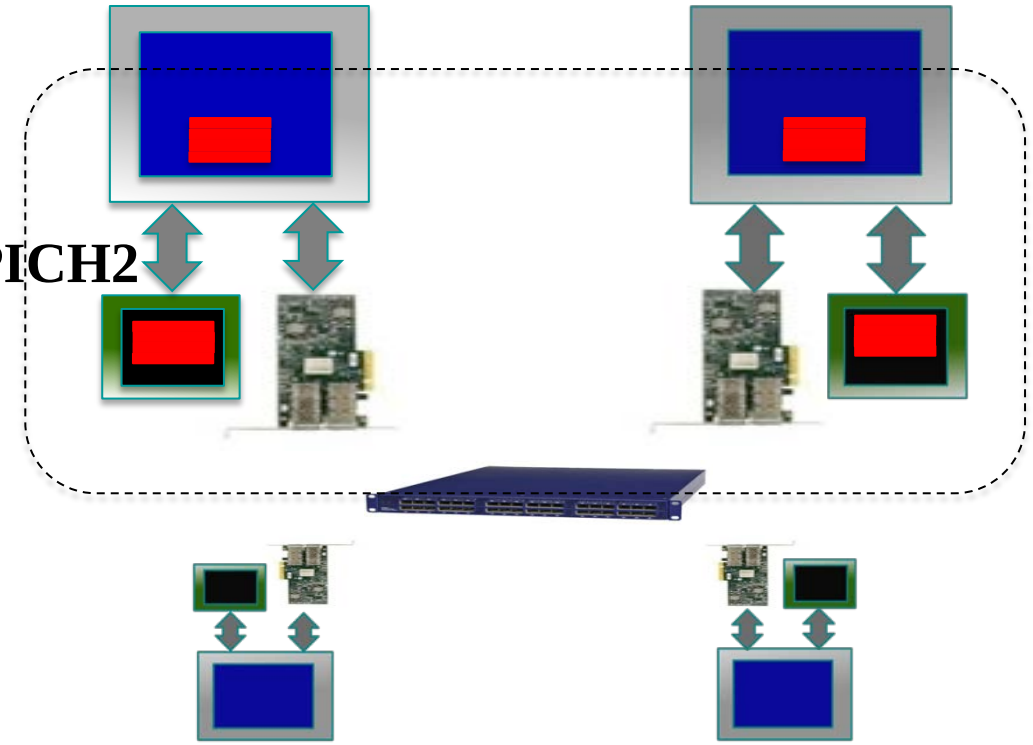
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

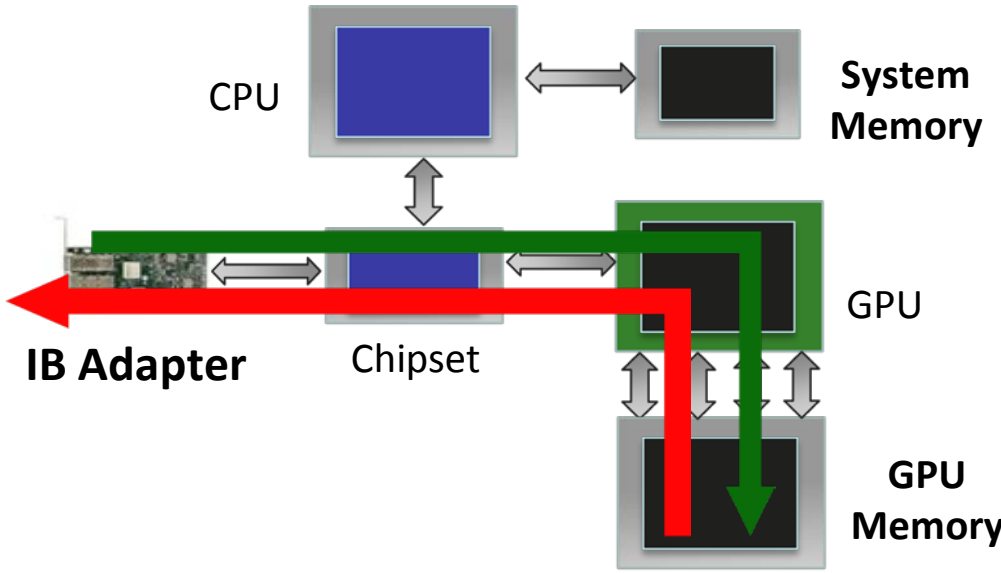
```
MPI_Recv(r_devbuf, size, …);
```



High Performance and High Productivity

GPU-Direct RDMA (GDR) with CUDA

- OFED with support for GPUDirect RDMA is developed by NVIDIA and Mellanox
- OSU has a design of MVAPICH2 using GPUDirect RDMA
  - Hybrid design using GPU-Direct RDMA
    - GPUDirect RDMA and Host-based pipelining
    - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters



SNB E5-2670

P2P write: 5.2 GB/s

P2P read: < 1.0 GB/s

IVB E5-2680V2

P2P write: 6.4 GB/s

P2P read: 3.5 GB/s

CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.2 Releases

- 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

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - **Basic CUDA-Aware Support**
    - Support for Efficient Small Message Communication with GPUDirect RDMA
    - Multi-rail Support
    - Support for Efficient Intra-node Communication using CUDA IPC
    - MPI Datatype Support
    - Support for OpenACC Constructs
    - CUDA Support in OMB
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

Pipelined Data Movement in MVAPICH2-GDR: Tuning

Parameter	Significance	Default	Notes
MV2_USE_CUDA	• Enable / Disable GPU designs	0 (Disabled)	• Disabled to avoid pointer checking overheads for host communication • Always enable to support MPI communication from GPU Memory
MV2_CUDA_BLOCK_SIZE	• Controls the pipeline blocksize	256 KByte	• Tune for your system and application • Varies based on - CPU Platform, IB HCA and GPU - CUDA driver version - Communication pattern (latency/bandwidth)

- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

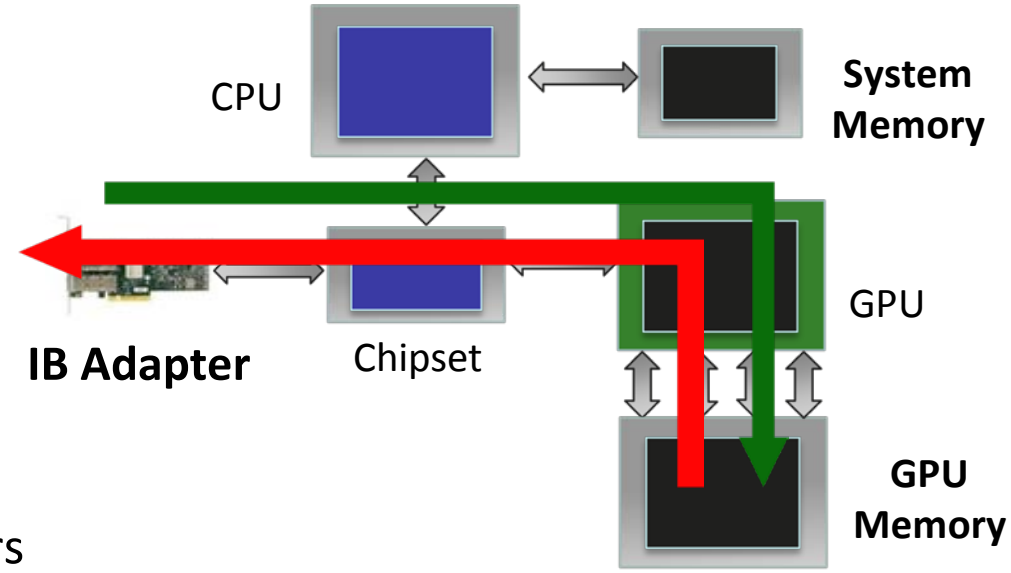
Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - Basic CUDA-Aware Support
    - **Advanced Support for GPUDirect RDMA**
      - **Designs for Enhanced Small Message Performance**
      - **Support for MPI-3 RMA**
      - **Multi-rail Support**
    - Support for Efficient Intra-node Communication using CUDA IPC
    - MPI Datatype Support
    - Support for OpenACC Constructs
    - CUDA Support in OMB
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

GPU-Direct RDMA (GDR) with CUDA

- OFED with support for GPUDirect RDMA is developed by NVIDIA and Mellanox
- OSU has a design of MVAPICH2 using GPUDirect RDMA
  - Hybrid design using GPU-Direct RDMA
    - GPUDirect RDMA and Host-based pipelining
    - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
    - Similar bottlenecks on Haswell
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters

SNB E5-2670 / IVB E5-2680V2



	SNB E5-2670		IVB E5-2680V2	
	Intra-socket	**Inter-sockets**	**Intra-socket**	**Inter-sockets**
P2P read	<1.0 GBs	<300 MBs	3.5 GBs	<300 MBs
P2P write	5.2 GBs	<300 MBs	6.4 GBs	<300 MBs

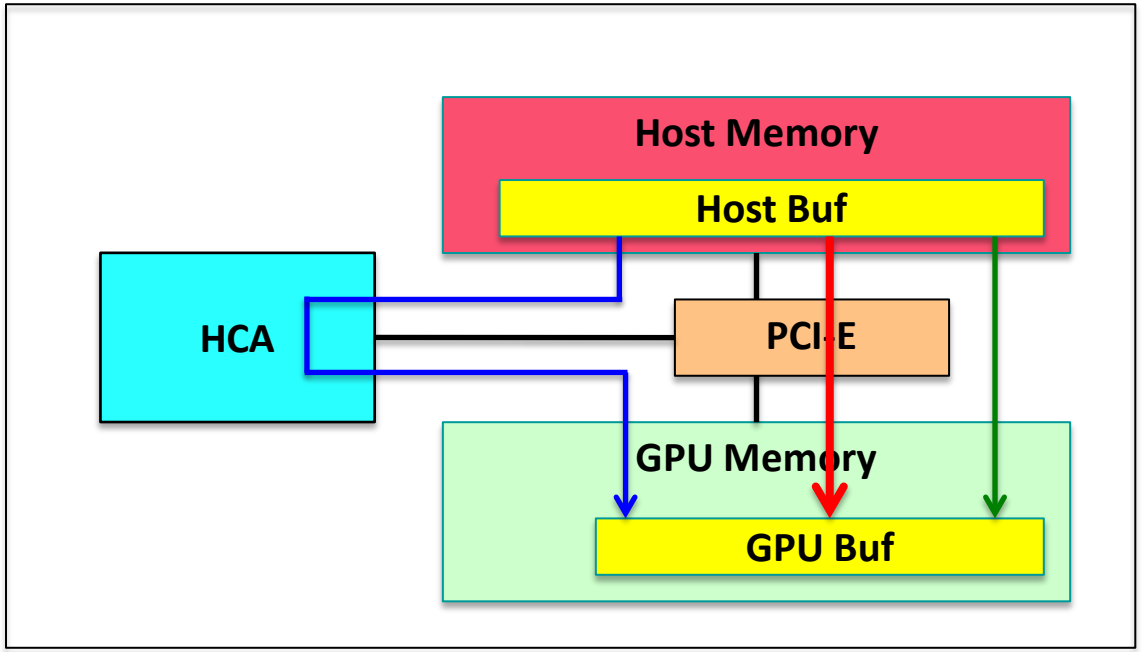
Tuning GPUDirect RDMA (GDR) Designs in MVAPICH2-GDR

Parameter	Significance	Default	Notes
MV2_USE_GPUDIRECT	• Enable / Disable GDR-based designs	1 (Enabled)	• Always enable
MV2_GPUDIRECT_LIMIT	• Controls messages size until which GPUDirect RDMA is used	8 KByte	• Tune for your system • GPU type, host architecture and CUDA version: impact pipelining overheads and P2P bandwidth bottlenecks
MV2_USE_GPUDIRECT_RECEIVE_LIMIT	• Controls messages size until which 1 hop design is used (GDR Write at the receiver)	256KBytes	• Tune for your system • GPU type, HCA type and configuration

- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

Enhanced MPI Design with GPUDirect RDMA

- Can eager protocol be supported to improve performance for small messages?
  - CudaMemcpy: Default Scheme
    - Big overhead for small message
  - Loopback-based design: Uses GDR feature
    - Process establishes self-connection
    - Copy H-D ⇒ RDMA write (H, D)
    - Copy D-H ⇒ RDMA write (D, H)
    - P2P bottleneck ⇒ good for small and medium sizes
  - GDRCOPY-based design: New module for fast copies
    - Involves GPU PCIe BAR1 mapping
    - CPU performing the copy ⇒ block until completion
    - Very good performance for H-D for small and medium sizes
    - Very good performance for D-H only for very small sizes



Tuning GDRCOPY Designs in MVAPICH2-GDR

Parameter	Significance	Default	Notes
MV2_USE_GPUDIRECT_GDRCOPY	• Enable / Disable GDRCOPY-based designs	1 (Enabled)	• Always enable
MV2_GPUDIRECT_GDRCOPY_LIMIT	• Controls messages size until which GDRCOPY is used	8 KByte	• Tune for your system • GPU type, host architecture. Impacts the eager performance
MV2_GPUDIRECT_GDRCOPY_LIB	• Path to the GDRCOPY library	Unset	• Always set
MV2_USE_GPUDIRECT_D2H_GDRCOPY_LIMIT	• Controls messages size until which GDRCOPY is used at sender	16Bytes	• Tune for your systems • CPU and GPU type

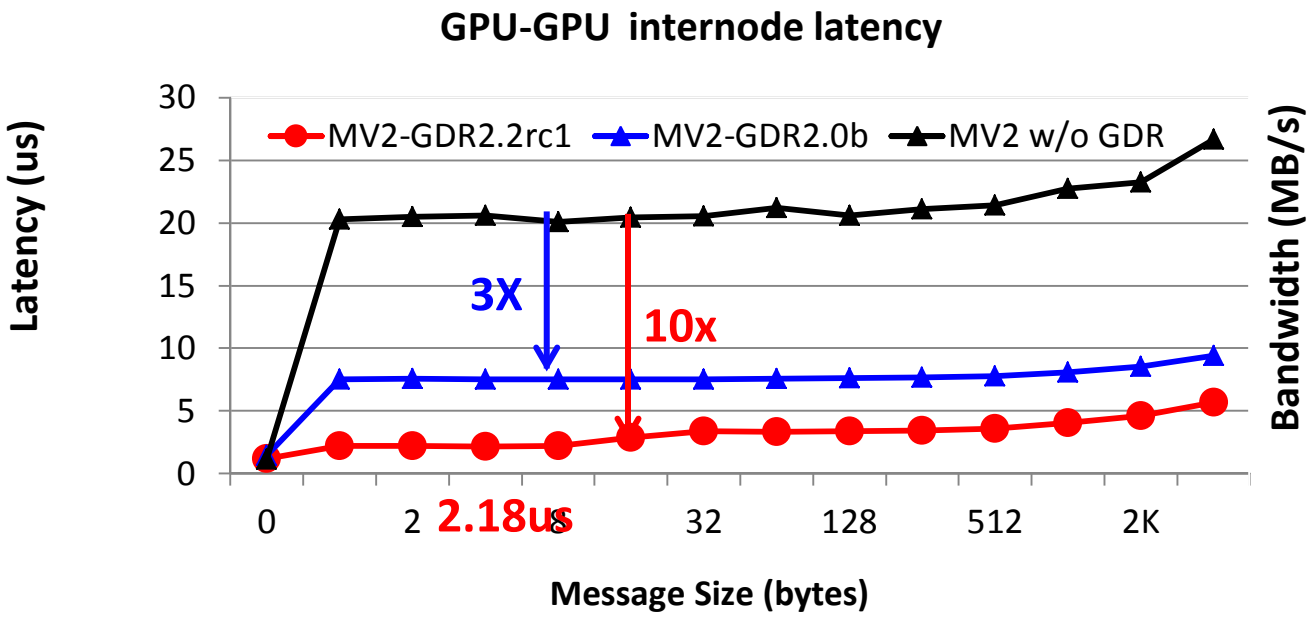
- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

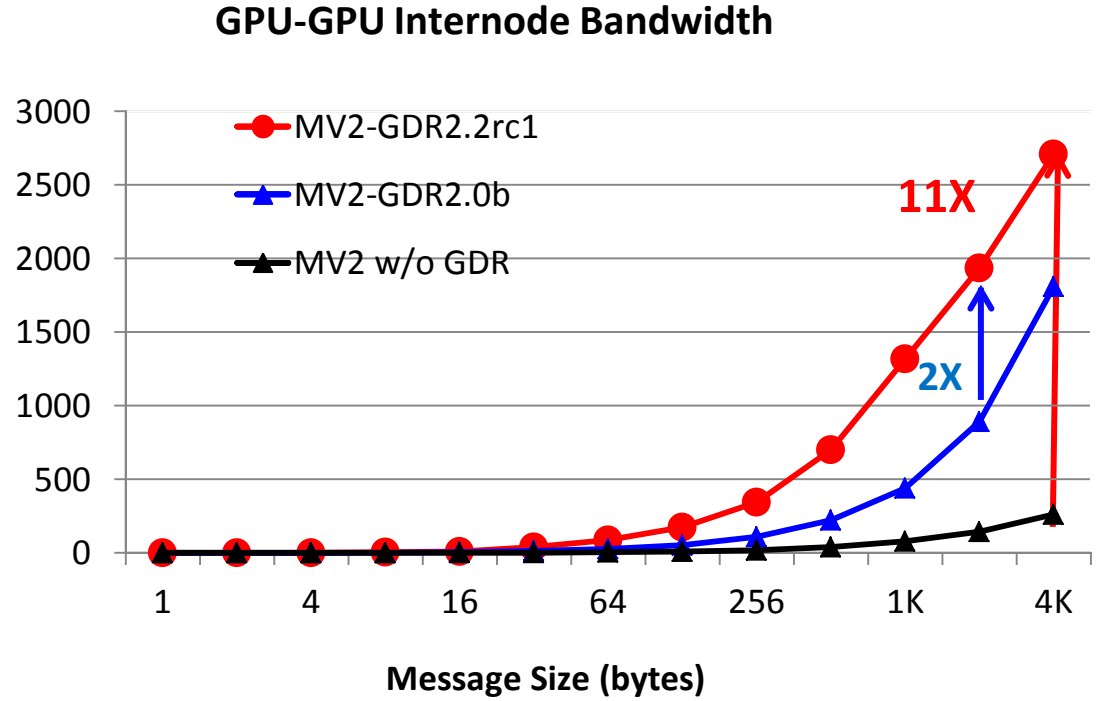
Tuning Loopback Designs in MVAPICH2-GDR

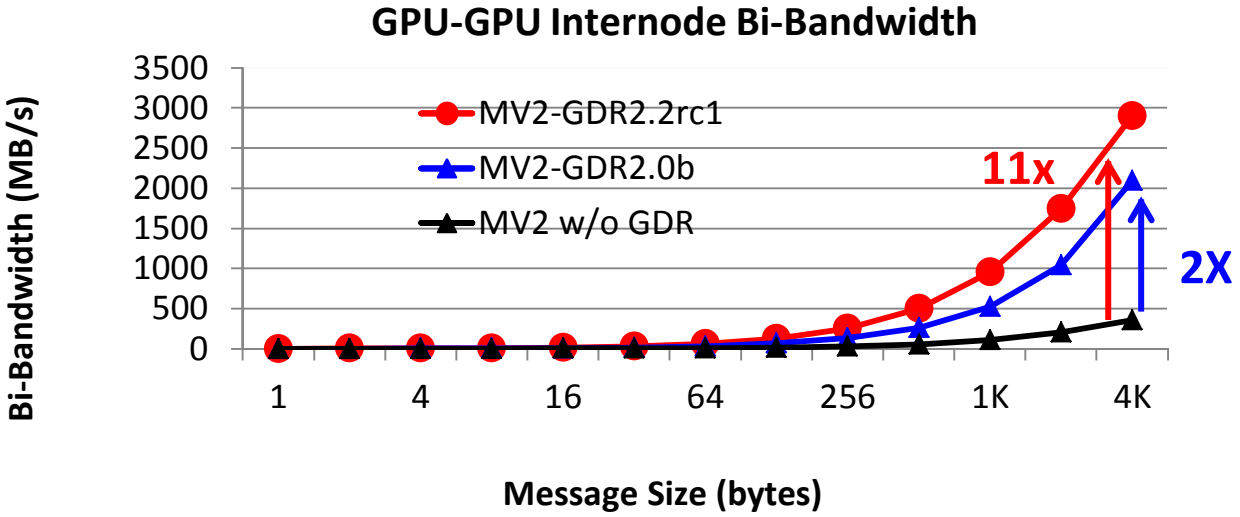
Parameter	Significance	Default	Notes
MV2_USE_GPUDIRECT_LOOPBACK	• Enable / Disable LOOPBACK-based designs	1 (Enabled)	• Always enable
MV2_GPUDIRECT_LOOPBACK_LIMIT	• Controls messages size until which LOOPBACK is used	8 KByte	• Tune for your system • GPU type, host architecture and HCA. Impacts the eager performance •Sensitive to the P2P issue

- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

Performance of MVAPICH2-GPU with GPU-Direct RDMA (GDR)



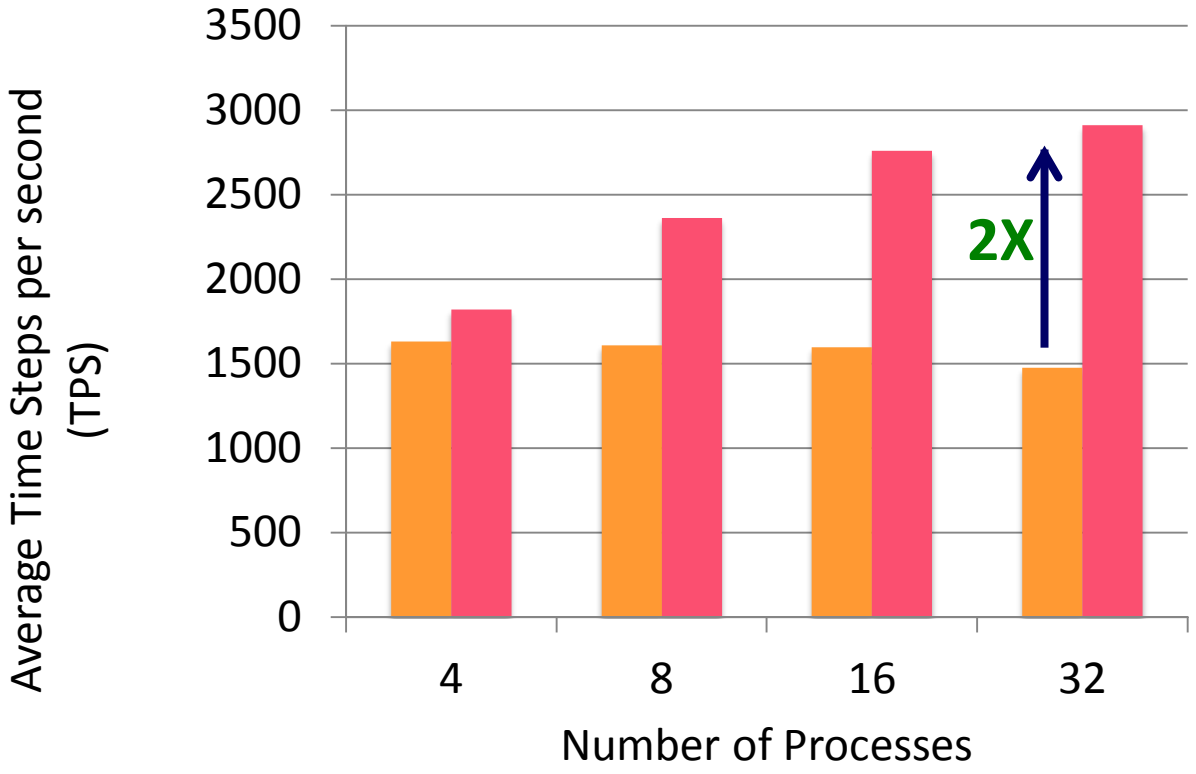




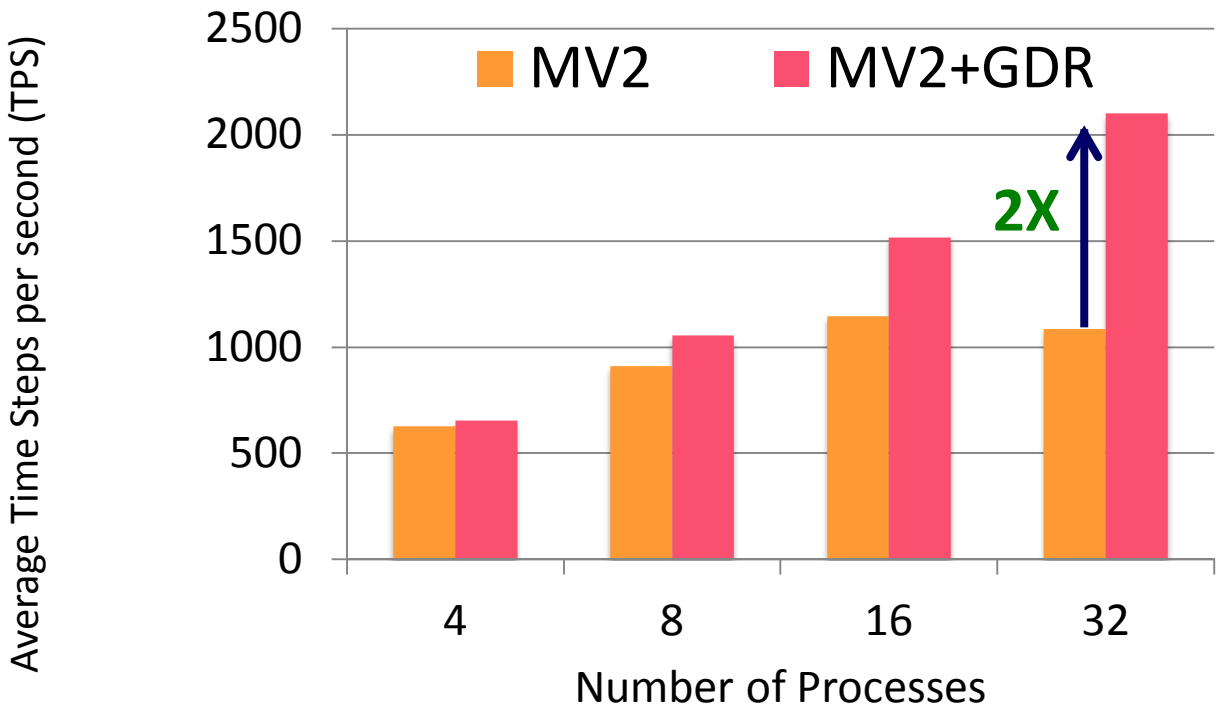
MVAPICH2-GDR-2.2rc1
Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 7.5
Mellanox OFED 3.0 with GPU-Direct-RDMA

Application-Level Evaluation (HOOMD-blue)

64K Particles



256K Particles

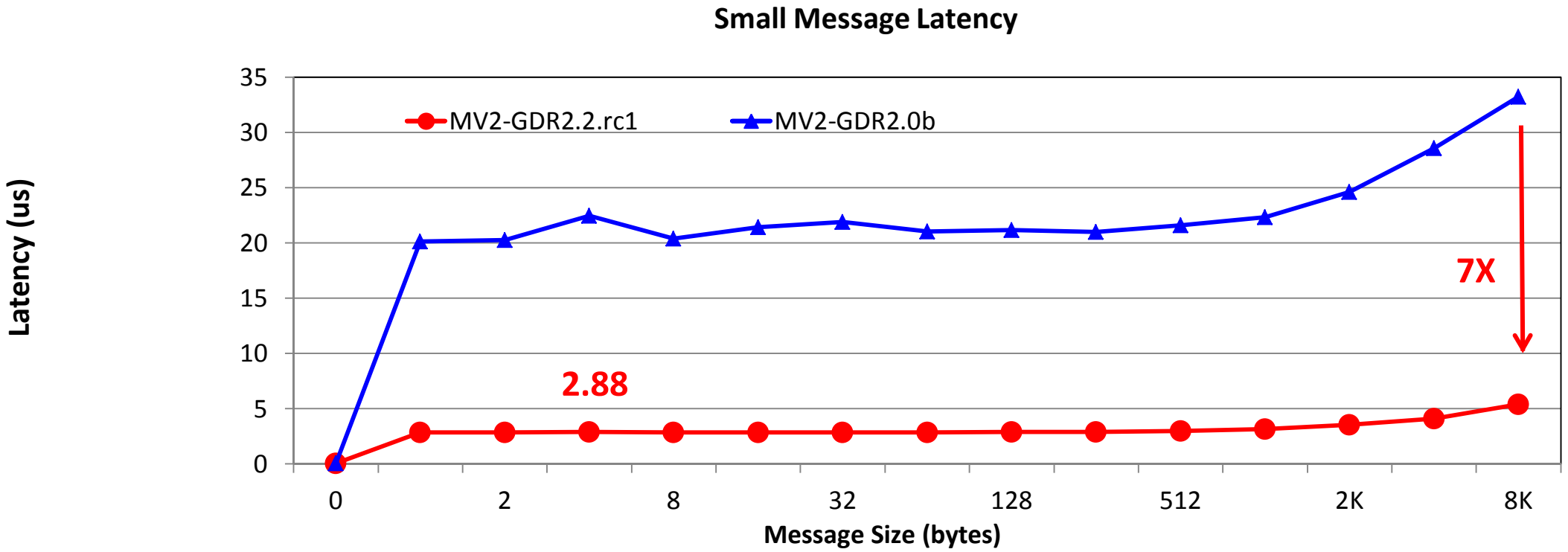


- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomdBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

Performance of MVAPICH2 with GPU-Direct-RDMA: MPI-3 RMA

GPU-GPU Internode MPI Put latency (RMA put operation Device to Device)

MPI-3 RMA provides flexible synchronization and completion primitives



MVAPICH2-GDR-2.2rc1
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-X4 EDR HCA
CUDA 7.5, Mellanox OFED 3.0 with GPU-Direct-RDMA

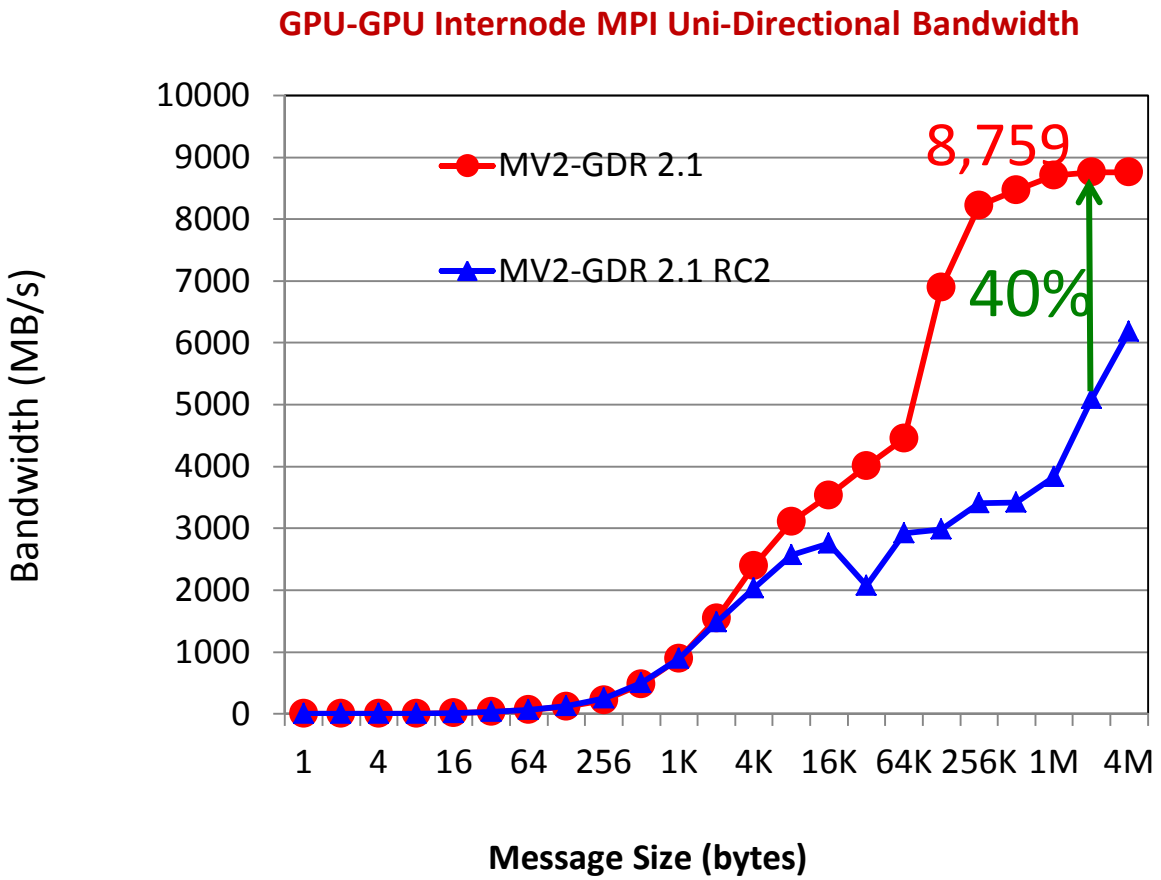
Tuning Multi-rail Support in MVAPICH2-GDR

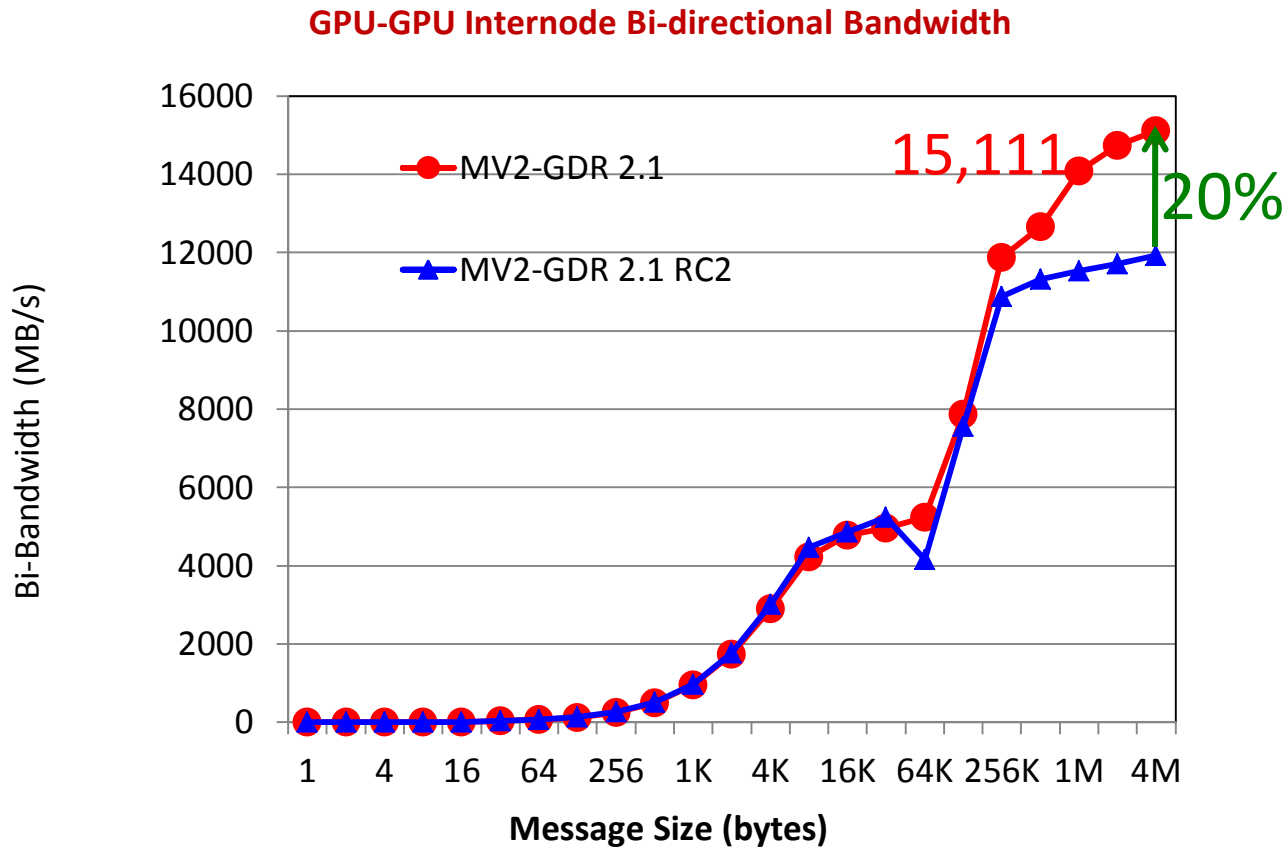
- Automatic rail and CPU binding depending on the GPU selection
  - User selects the GPU and MVAPICH2-GDR selects the best HCA (avoids the P2P bottleneck)
  - Multi-rail selection for large message size for better Bandwidth utilization (pipeline design)

Parameter	Significance	Default	Notes
MV2_RAIL_SHARING_POLICY	• How the Rails are bind/selected by processes	Shared	• Sharing gives the best performance for pipeline design
PROCESS_TO_RAIL_MAPPING	• Explicit binding of the HCAs to the CPU	First HCA	• Manually select if automatic mapping is leading to performance issues

- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

Performance of MVAPICH2-GDR with GPU-Direct RDMA and Multi-Rail Support





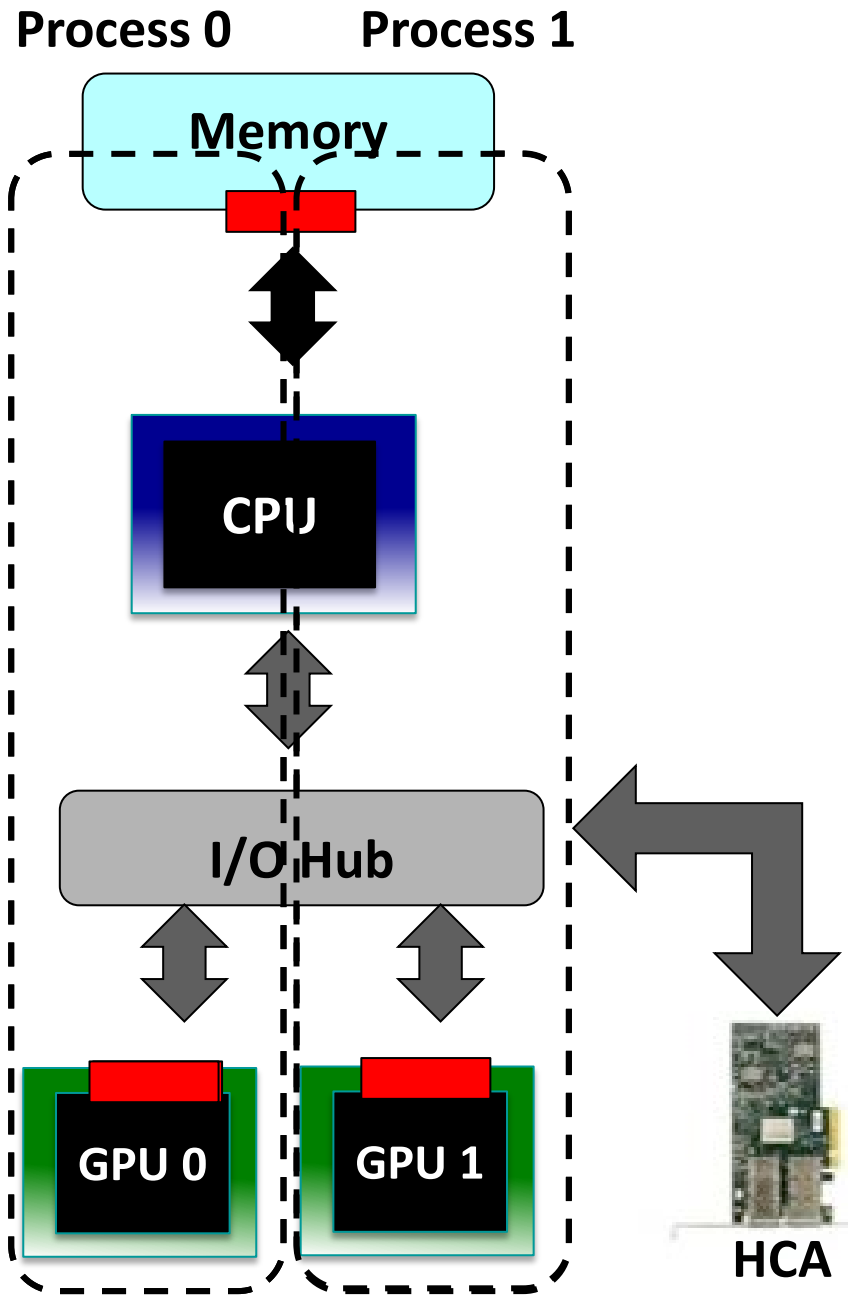
MVAPICH2-GDR-2.2rc1
Intel Ivy Bridge (E5-2680 v2) node - 20 cores, NVIDIA Tesla K40c GPU
Mellanox Connect-IB Dual-FDR HCA CUDA 7
Mellanox OFED 2.4 with GPU-Direct-RDMA

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - Basic CUDA-Aware Support
    - Advanced Support for GPUDirect RDMA
    - **Support for Efficient Intra-node Communication using CUDA IPC**
    - MPI Datatype Support
    - Support for OpenACC Constructs
    - CUDA Support in OMB
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

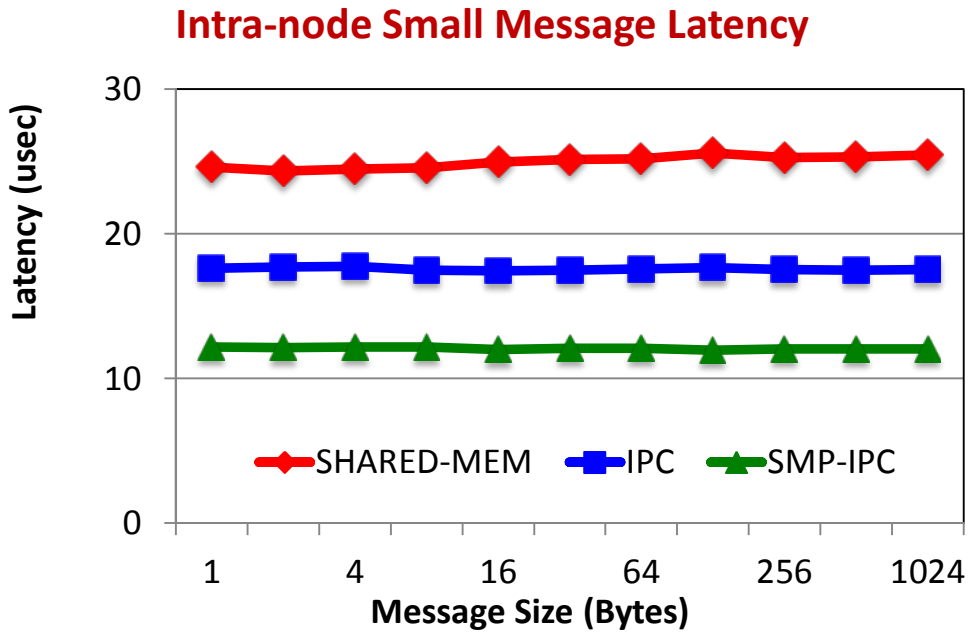
Multi-GPU Configurations

- Multi-GPU node architectures are becoming common
- Until CUDA 3.2
  - Communication between processes staged through the host
  - Shared Memory (pipelined)
  - Network Loopback [asynchronous)
- CUDA 4.0 and later
  - Inter-Process Communication (IPC)
  - Host bypass
  - Handled by a DMA Engine
  - Low latency and Asynchronous
  - Requires creation, exchange and mapping of memory handles
  - Overhead



Tuning IPC designs in MVAPICH2-GDR

- Works between GPUs within the same socket or IOH
- Leads to significant benefits in appropriate scenarios



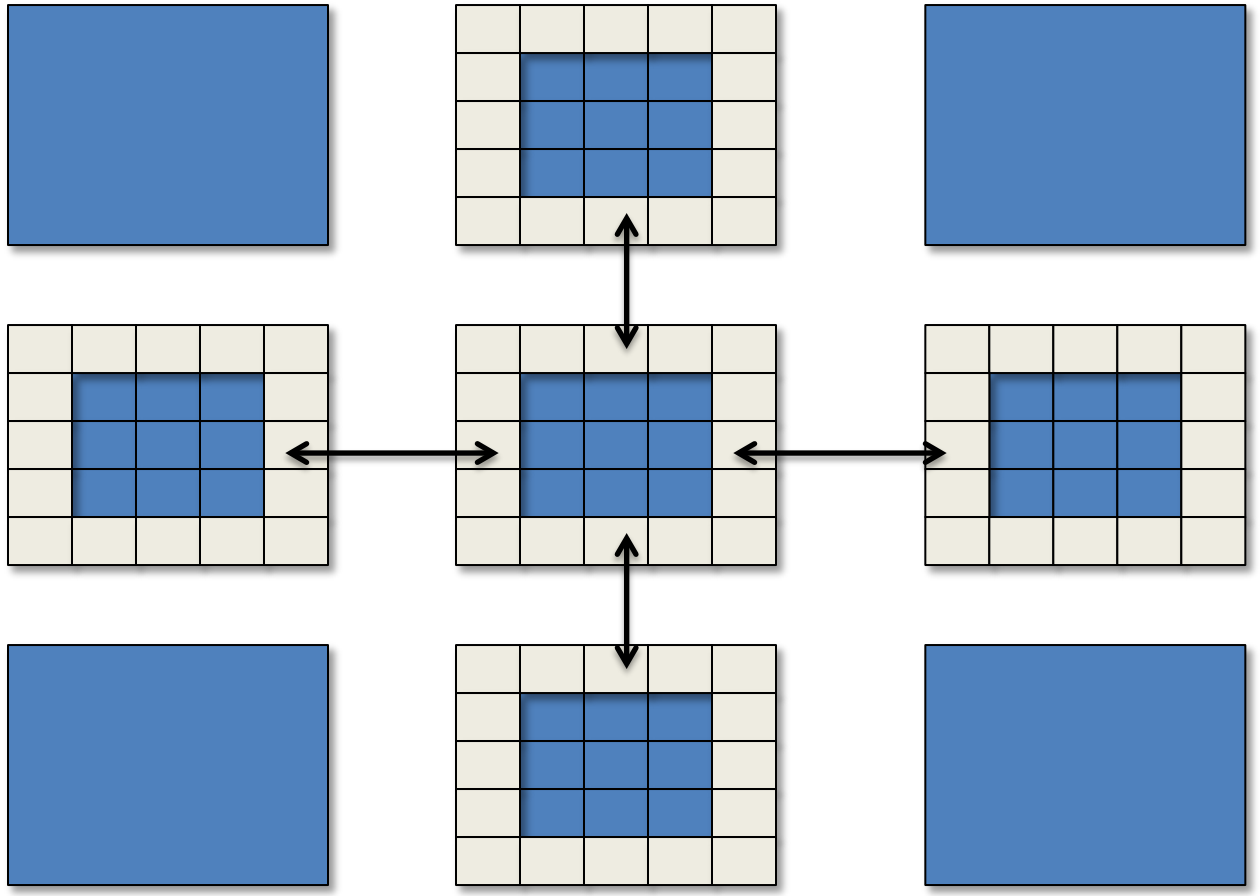
Parameter	Significance	Default	Notes
MV2_CUDA_IPC	• Enable / Disable CUDA IPC-based designs	1 (Enabled)	• Always leave set to 1
MV2_CUDA_SMP_IPC	• Enable / Disable CUDA IPC fastpath design for short messages	0 (Disabled)	• Benefits Device-to-Device transfers • Hurts Device-to-Host/Host-to-Device transfers • Always set to 1 if application involves only Device-to-Device transfers
MV2_IPC_THRESHOLD	• Message size where IPC code path will be used	16 KBytes	• Tune for your system

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - Basic CUDA-Aware Support
    - Advanced Support for GPUDirect RDMA
    - Support for Efficient Intra-node Communication using CUDA IPC
    - **MPI Datatype Support**
    - Support for OpenACC Constructs
    - CUDA Support in OMB
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

Non-contiguous Data Exchange

Halo data exchange



- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions
- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration

MPI Datatype support in MVAPICH2

- Datatypes support in MPI
  - Operate on customized datatypes to improve productivity
  - Enable MPI library to optimize non-contiguous data

At Sender:

```
MPI_Type_vector (n_blocks, n_elements, stride, old_type, &new_type);
MPI_Type_commit(&new_type);
…
MPI_Send(s_buf, size, new_type, dest, tag, MPI_COMM_WORLD);
```

- Inside MVAPICH2
  - Use datatype specific CUDA Kernels to pack data in chunks
  - Efficiently move data between nodes using RDMA
  - In progress - currently optimizes *vector* and *hindexed* datatypes
  - Transparent to the user

H. Wang, S. Potluri, D. Bureddy, C. Rosales and D. K. Panda, GPU-aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation, IEEE Transactions on Parallel and Distributed Systems, Accepted for Publication.

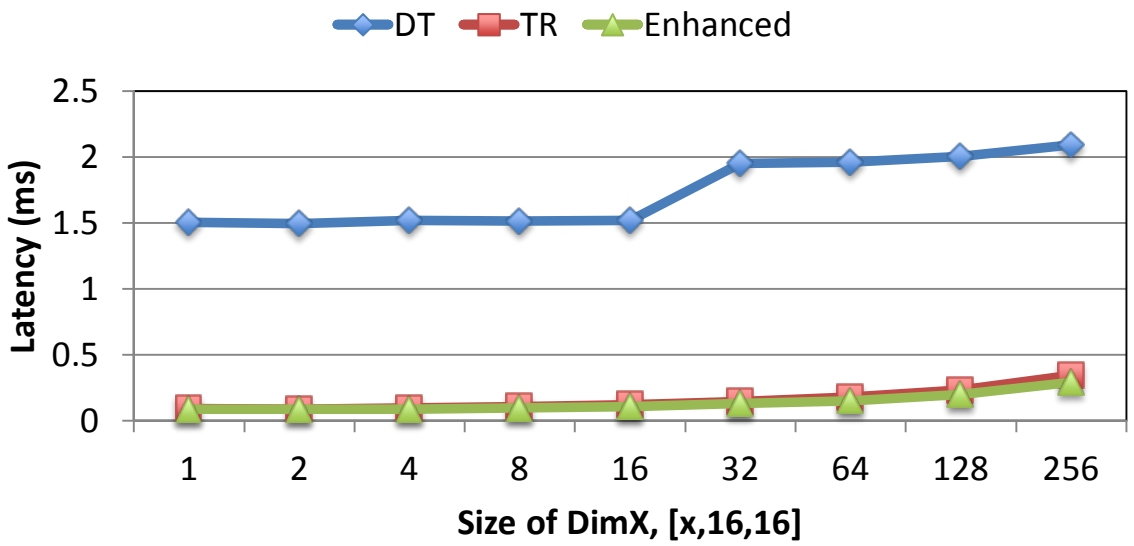
MPI Datatype Processing (Computation Optimization)

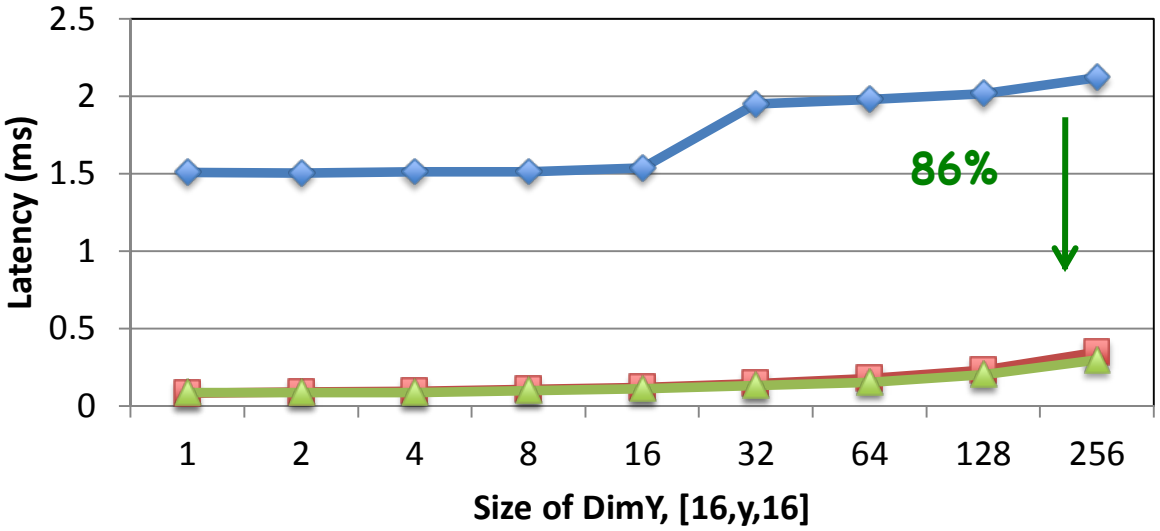
- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes
- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels
- Flexible set of parameters for users to tune kernels
  - Vector
    - MV2_CUDA_KERNEL_VECTOR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_VECTOR_YSIZE
  - Subarray
    - MV2_CUDA_KERNEL_SUBARR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_SUBARR_XDIM
    - MV2_CUDA_KERNEL_SUBARR_YDIM
    - MV2_CUDA_KERNEL_SUBARR_ZDIM
  - Indexed_block
    - MV2_CUDA_KERNEL_IDXBLK_XDIM

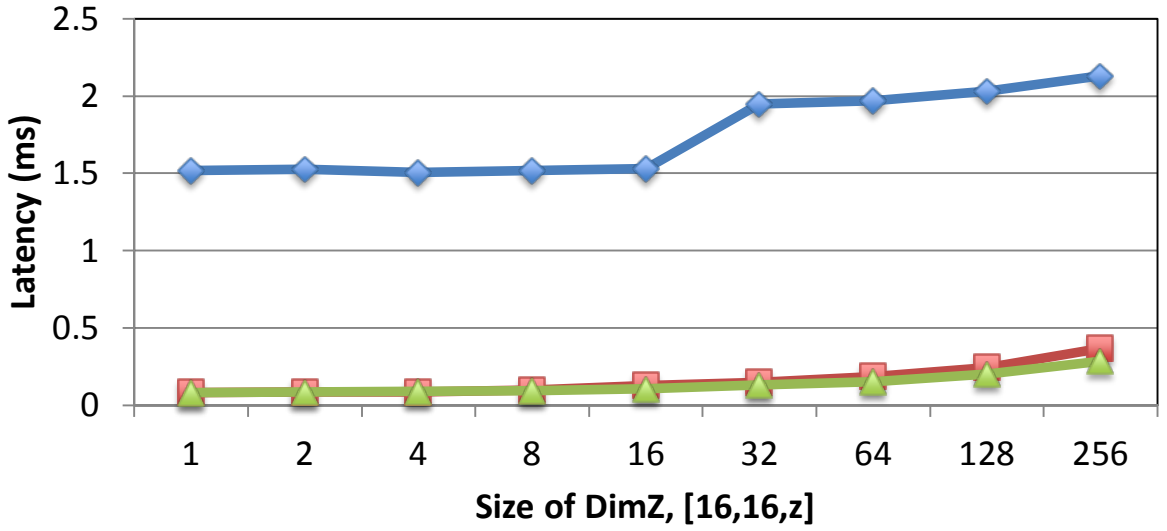
Performance of Stencil3D (3D subarray)

Stencil3D communication kernel on 2 GPUs with various X, Y, Z dimensions using MPI_Isend/Irecv

- DT: Direct Transfer, TR: Targeted Kernel
- Optimized design gains up to 15%, 15% and 22% compared to TR, and more than 86% compared to DT on X, Y and Z respectively







Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - Basic CUDA-Aware Support
    - Advanced Support for GPUDirect RDMA
    - Support for Efficient Intra-node Communication using CUDA IPC
    - MPI Datatype Support
    - **Support for OpenACC Constructs**
    - CUDA Support in OMB
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

OpenACC-Aware MPI

- acc_malloc to allocate device memory
  - No changes to MPI calls
  - MVAPICH2 detects the device pointer and optimizes data movement
- acc_deviceptr to get device pointer (in OpenACC 2.0)
  - Enables MPI communication from memory allocated by compiler when it is available in OpenACC 2.0 implementations
  - MVAPICH2 will detect the device pointer and optimize communication
- Delivers the same performance as with CUDA

```
A = acc_malloc(sizeof(int) * N);
……
#pragma acc parallel loop deviceptr(A) . . .
//compute for loop

MPI_Send (A, N, MPI_INT, 0, 1, MPI_COMM_WORLD);

……
acc_free(A);
```

```
A = malloc(sizeof(int) * N);
……
#pragma acc data copyin(A) . . .
{
#pragma acc parallel loop . . .
//compute for loop
MPI_Send(acc_deviceptr(A), N, MPI_INT, 0, 1,
MPI_COMM_WORLD);
}
```

Presentation Overview

- **Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - **MVAPICH2-GDR**
    - Basic CUDA-Aware Support
    - Advanced Support for GPUDirect RDMA
    - Support for Efficient Intra-node Communication using CUDA IPC
    - MPI Datatype Support
    - Support for OpenACC Constructs
    - **CUDA Support in OMB**
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

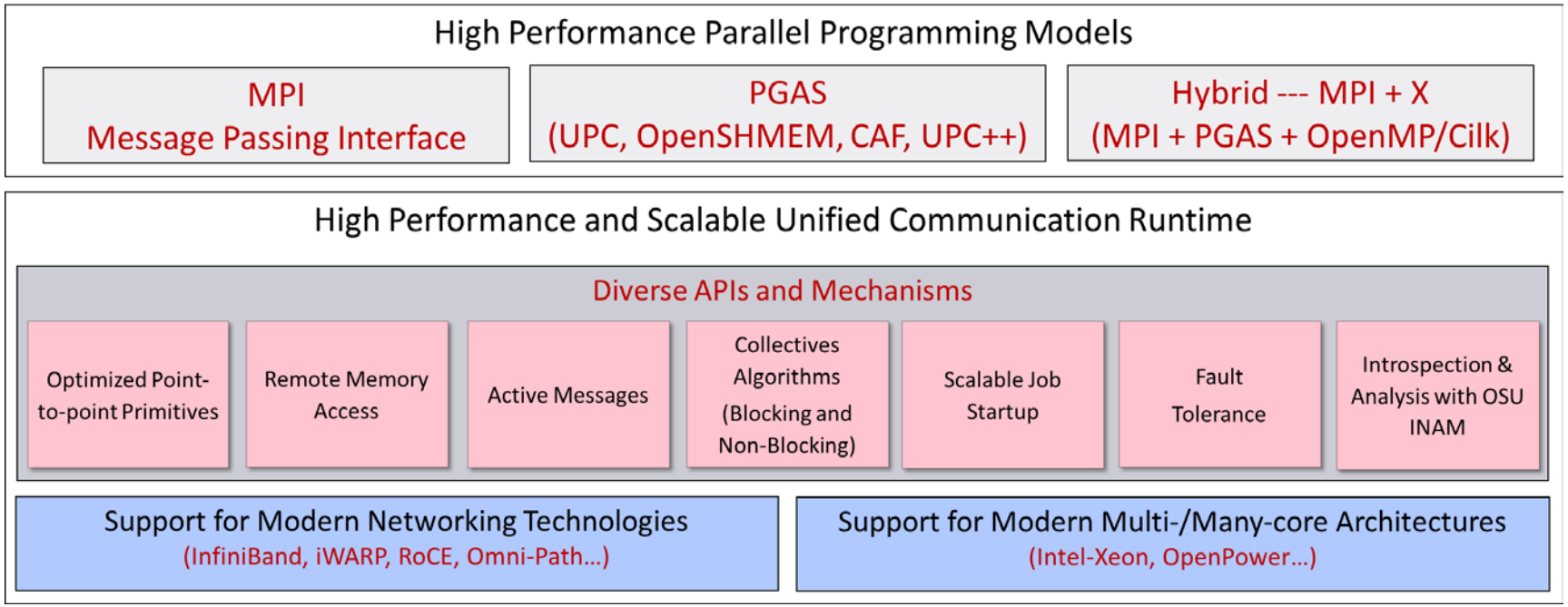
CUDA and OpenACC Extensions in OMB

- OSU Micro-benchmarks are widely used to compare performance of different MPI stacks and networks
- Enhancements to measure performance of MPI communication from GPU memory
  - Point-to-point: Latency, Bandwidth and Bi-directional Bandwidth
  - Collectives: Alltoall, Gather and Scatter
- Support for CUDA and OpenACC
- Flexible selection of data movement between CPU(H) and GPU(D): D->D, D->H and H->D
- Available from http://mvapich.cse.ohio-state.edu/benchmarks
- Available in an integrated manner with MVAPICH2-GDR stack
- Support for CUDA Managed Memory feature

Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - **MVAPICH2-X**
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

MVAPICH2-X for Hybrid MPI + PGAS Applications



- Current Model – Separate Runtimes for OpenSHMEM/UPC/UPC++/CAF and MPI
  - Possible deadlock if both runtimes are not progressed
  - Consumes more network resource
- Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF
  - Available with since 2012 (starting with MVAPICH2-X 1.9)
  - http://mvapich.cse.ohio-state.edu

MVAPICH2-X 2.2rc2

- Released on 08/08/2016
- Major Features and Enhancements
  - MPI Features
    - Based on MVAPICH2 2.2rc2 (OFA-IB-CH3 interface)
    - Efficient support for On Demand Paging (ODP) feature of Mellanox for point-to-point and RMA operations
    - Support for Intel Knights Landing architecture
  - UPC Features
    - Support for Intel Knights Landing architecture
  - UPC++ Features
    - Support for Intel Knights Landing architecture
  - OpenSHMEM Features
    - Support for Intel Knights Landing architecture
  - CAF Features
    - Support for Intel Knights Landing architecture
  - Hybrid Program Features
    - Support Intel Knights Landing architecture for hybrid MPI+PGAS applications
  - Unified Runtime Features
    - Based on MVAPICH2 2.2rc2 (OFA-IB-CH3 interface). All the runtime features enabled by default in OFA-IB-CH3 and OFA-IB-RoCE interface of MVAPICH2 2.2rc2 are available in MVAPICH2-X 2.2rc2

Compiling programs with MVAPICH2-X

- Compile MPI programs using mpicc
  - $ mpicc -o helloworld_mpi helloworld_mpi.c
- Compile UPC programs using upcc
  - $ upcc -o helloworld_upc helloworld_upc.c
- Compile OpenSHMEM programs using oshcc
  - $ oshcc -o helloworld_oshm helloworld_oshm.c
- Compile CAF programs using OpenUH CAF compiler
  - $ uhcaf **--layer=gasnet-mvapich2x** -o helloworld_caf helloworld_caft.caf
- Compile Hybrid MPI+UPC programs using upcc
  - $ upcc -o hybrid_mpi_upc hybrid_mpi_upc.c
- Compile Hybrid MPI+OpenSHMEM programs using oshcc
  - $ oshcc -o hybrid_mpi_oshm hybrid_mpi_oshm.c

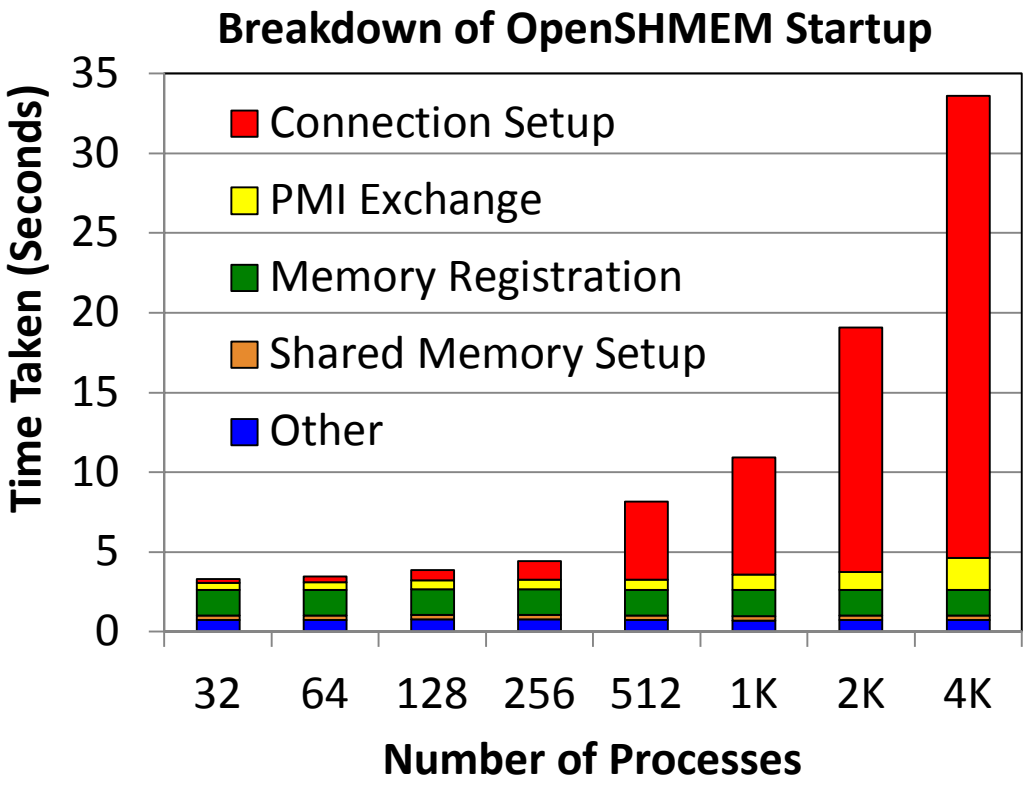
Running Programs with MVAPICH2-X

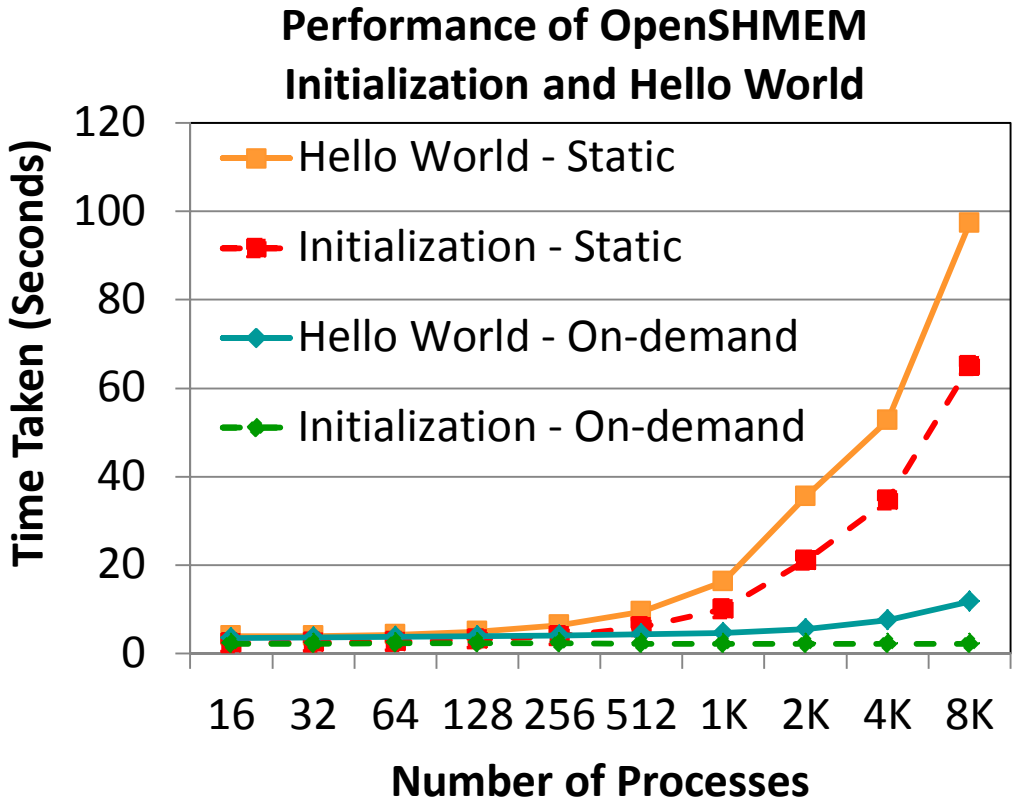
- MVAPICH2-X programs can be run using
  - mpirun_rsh and mpiexec.hydra (MPI, UPC, OpenSHMEM and hybrid)
  - upcrun (UPC)
  - oshrun (OpenSHMEM)
  - cafrun (CAF)
- Running using mpirun_rsh/mpiexec.hydra
  - $ mpirun rsh -np 4 -hostfile hosts ./test ---$ mpiexec -f hosts -n 2 ./test
- Running using upcrun
  - $ export MPIRUN CMD="<path-to-MVAPICH2-X-install>/bin/mpirun rsh -np %N -hostfile hosts %P %A"
  - $ upcrun -n 2 ./test
- Running using oshrun
  - $ oshrun -f hosts -np 2 ./test
- Running using cafrun
  - Export the PATH and LD_LIBRARY_PATH of the GNU version of MVAPICH2-X
  - $ cafrun –n 16 –v ./test

OSU Microbenchmarks – UPC, UPC++ and OpenSHMEM

- OpenSHMEM benchmarks
  - osu_oshm_put – Put latency
  - osu_oshm_get – Get latency
  - osu_oshm_put_mr – Put message rate
  - osu_oshm_atomics – Atomics latency
  - osu_oshm_collect – Collect latency
  - osu_oshm_broadcast – Broadcast latency
  - osu_oshm_reduce - Reduce latency
  - osu_oshm_barrier - Barrier latency
- UPC benchmarks
  - osu upc memput – Put latency
  - osu upc memget - Get latency
  - osu_upc_all_barrier – Barrier latency
  - osu_upc_all_broadcast – Broadcast latency
  - osu_upc_all_exchange – Exchange latency
  - osu_upc_all_gather_all – GatherAll latency
  - osu_upc_all_gather – Gather latency
  - osu_upc_all_reduce – Reduce latency
  - osu_upc_all_scatter – Scatter latency
- UPC++ benchmarks
  - osu_upcxx_async_copy_put – Put latency
  - osu_upcxx_async_copy_get – Get latency
  - osu_upcxx_allgather – Allgather latency
  - osu_upcxx_bcast – Broadcast latency
  - osu_upcxx_reduce – Reduce latency
  - osu_upcxx_alltoall – Alltoall latency
  - osu_upcxx_gather – Gather latency
  - osu_upcxx_scatter – Scatter latency

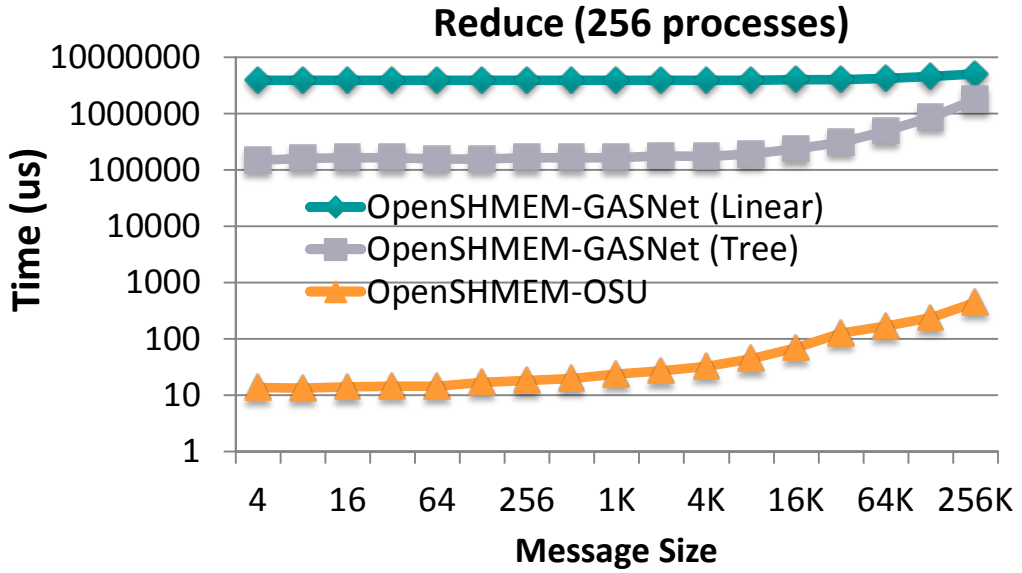
On-demand Connection Management for OpenSHMEM+MPI

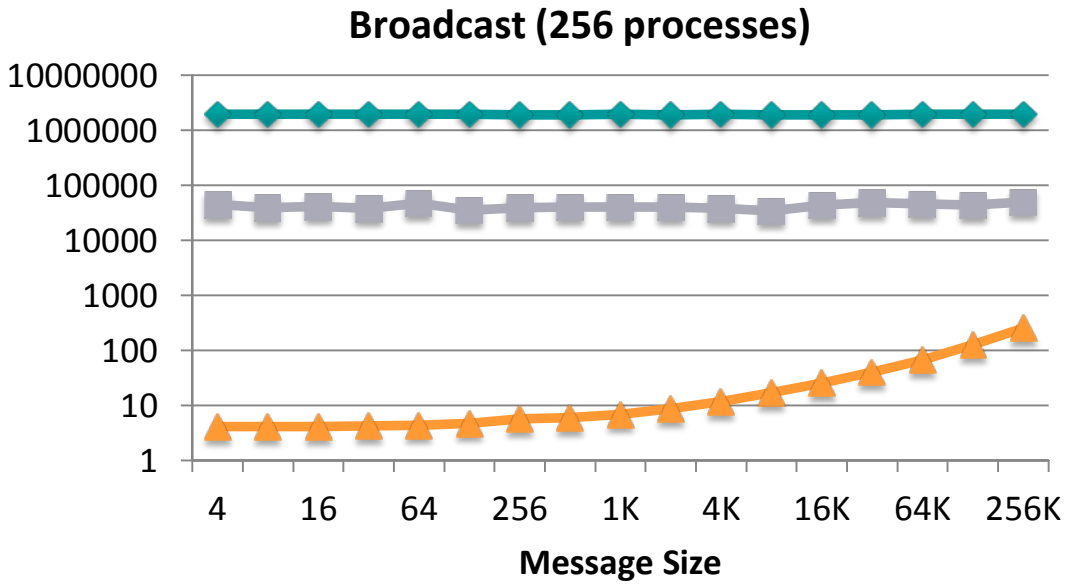


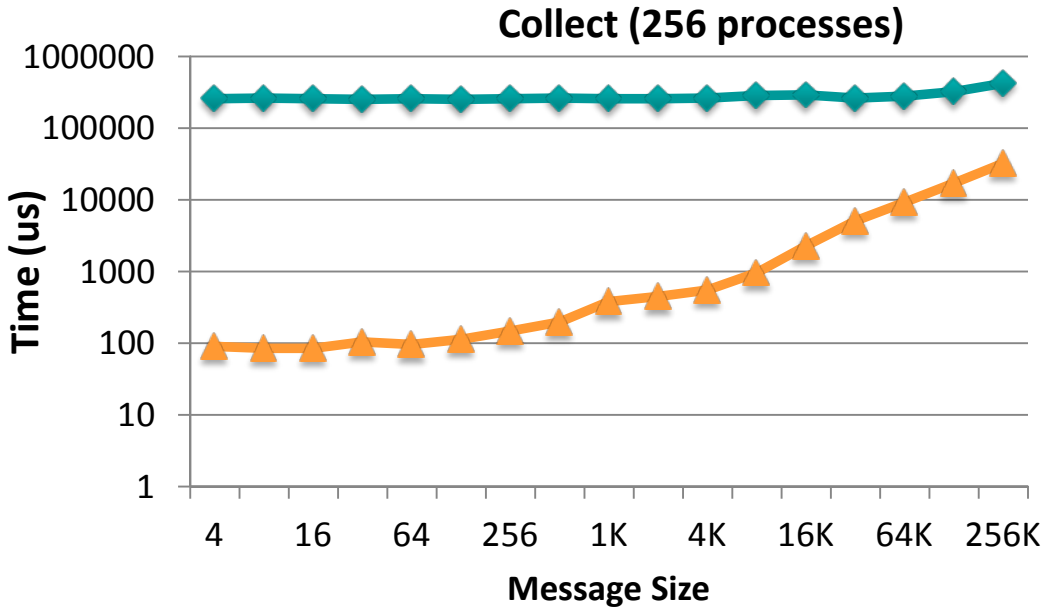


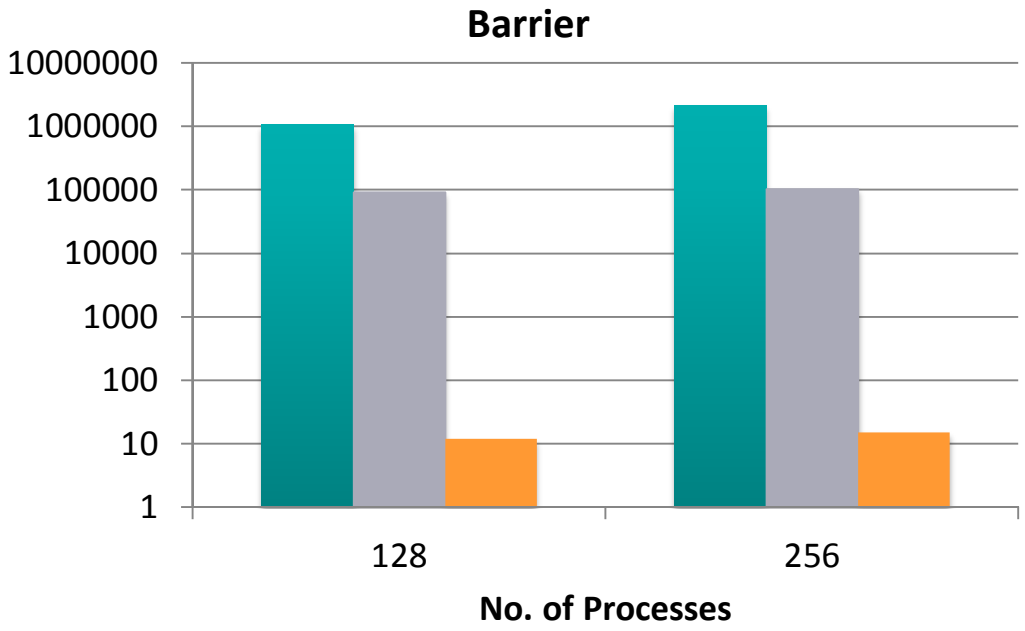
- Static connection establishment wastes memory and takes a lot of time
- On-demand connection management improves OpenSHMEM initialization time by 29.6 times
- Time taken for Hello World reduced by 8.31 times at 8,192 processes
- **Available since MVAPICH2-X 2.1rc1**

Microbenchmark Level Performance

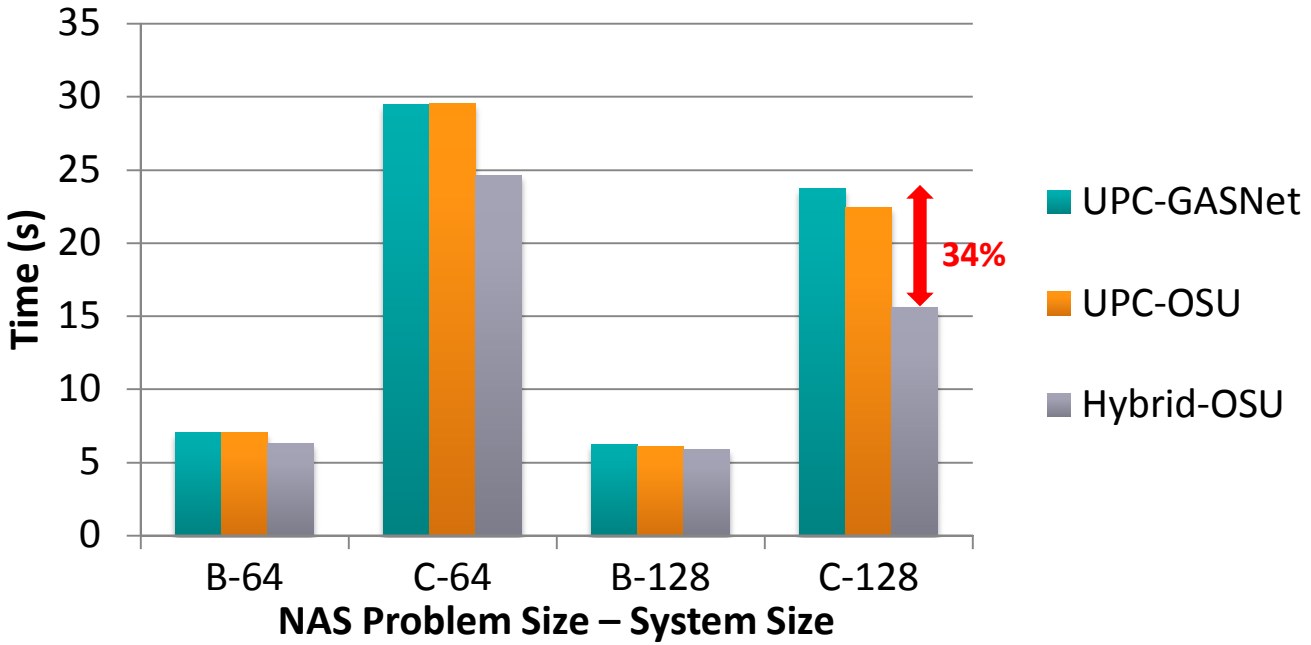








Hybrid MPI+UPC NAS-FT



- 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

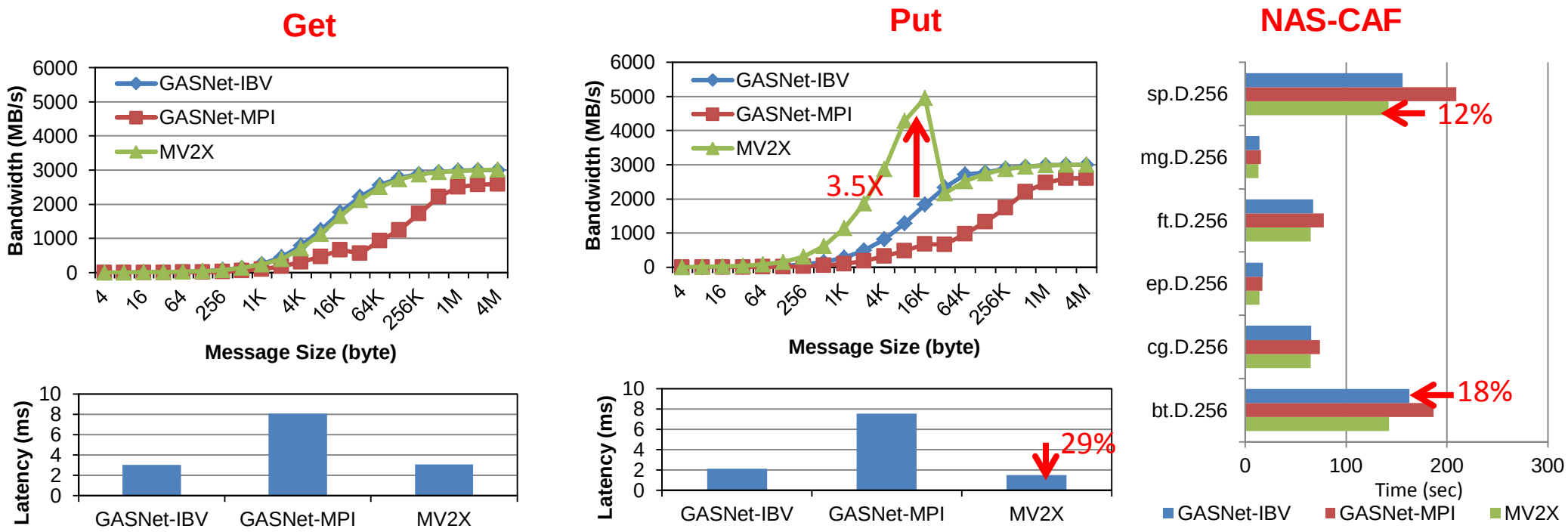
Hybrid MPI + UPC Support

Available since

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

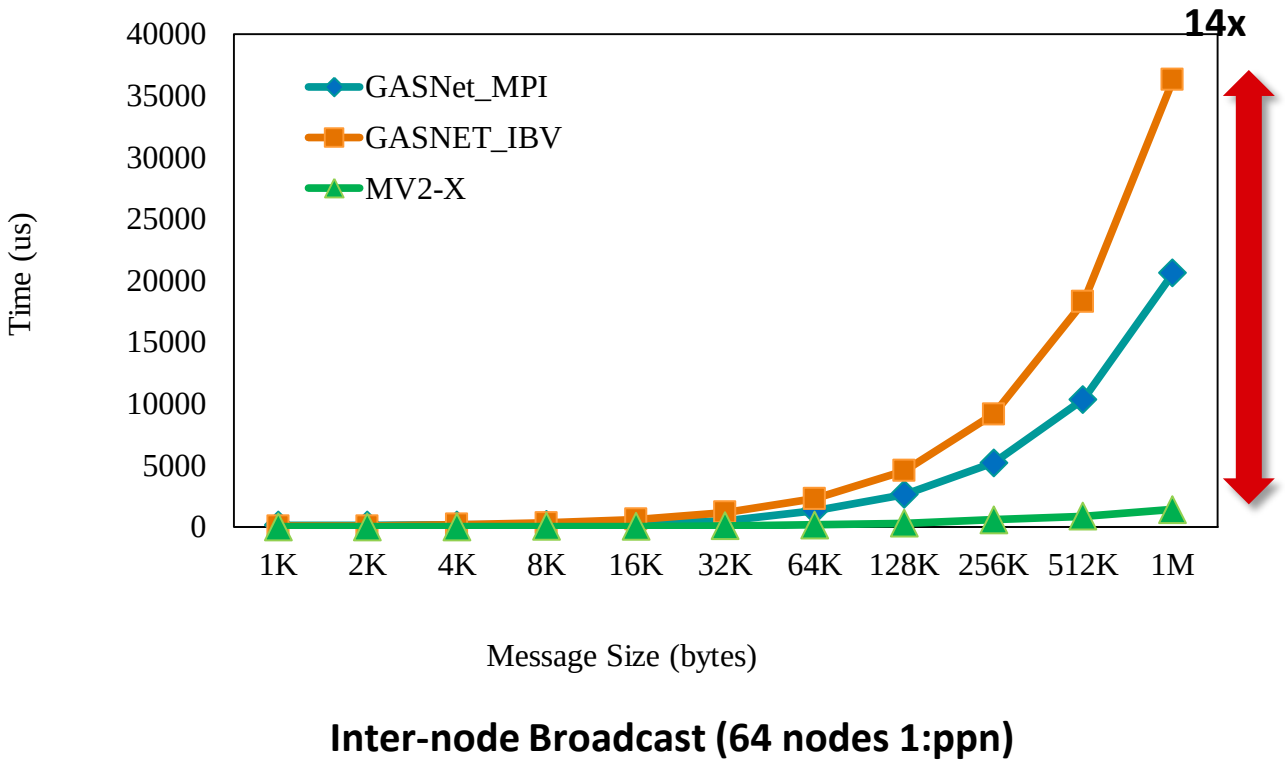
Performance Evaluations for One-sided Communication



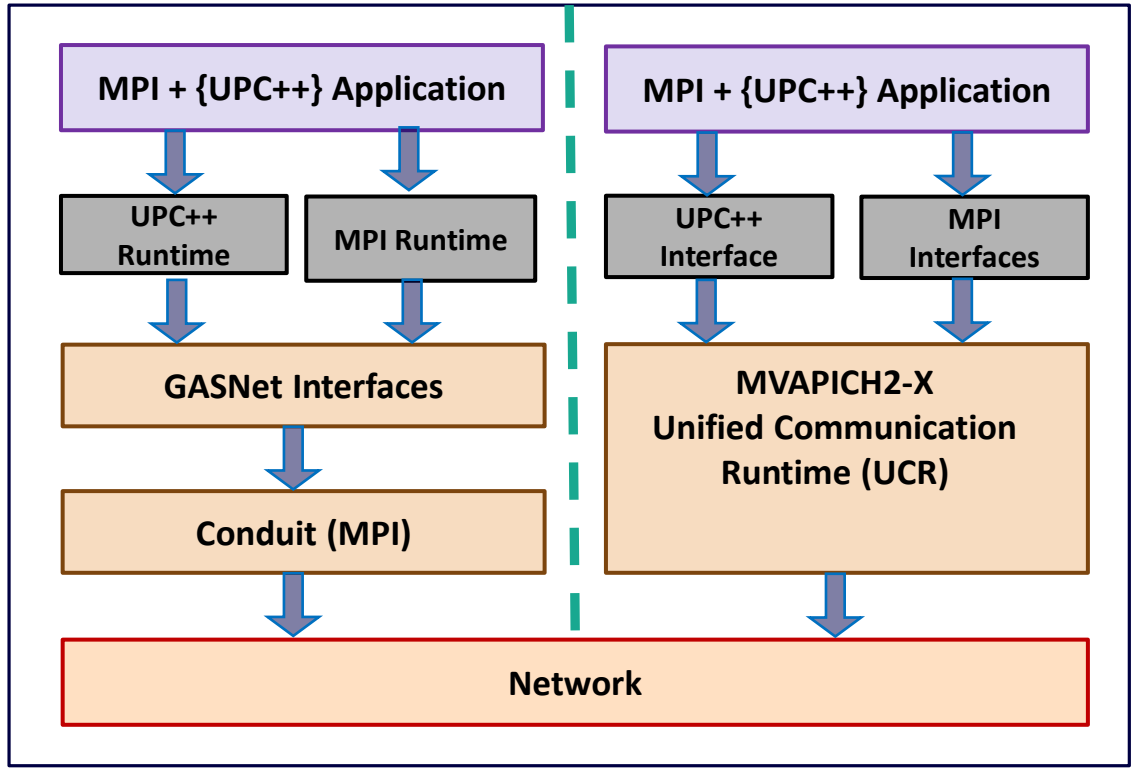
- Micro-benchmark improvement (MV2X vs. GASNet-IBV, UH CAF test-suite)
  - Put bandwidth: 3.5X improvement on 4KB; Put latency: reduce 29% on 4B
- Application performance improvement (NAS-CAF one-sided implementation)
  - Reduce the execution time by 12% (SP.D.256), 18% (BT.D.256)

J. Lin, K. Hamidouche, X. Lu, M. Li and D. K. Panda, High-performance Co-array Fortran support with MVAPICH2-X: Initial experience and evaluation, HIPS'15

UPC++ Support in MVAPICH2-X



Inter-node Broadcast (64 nodes 1:ppn)



- Full and native support for hybrid MPI + UPC++ applications
- Better performance compared to IBV and MPI conduits
- OSU Micro-benchmarks (OMB) support for UPC++
- Available since MVAPICH2-X (2.2rc1)

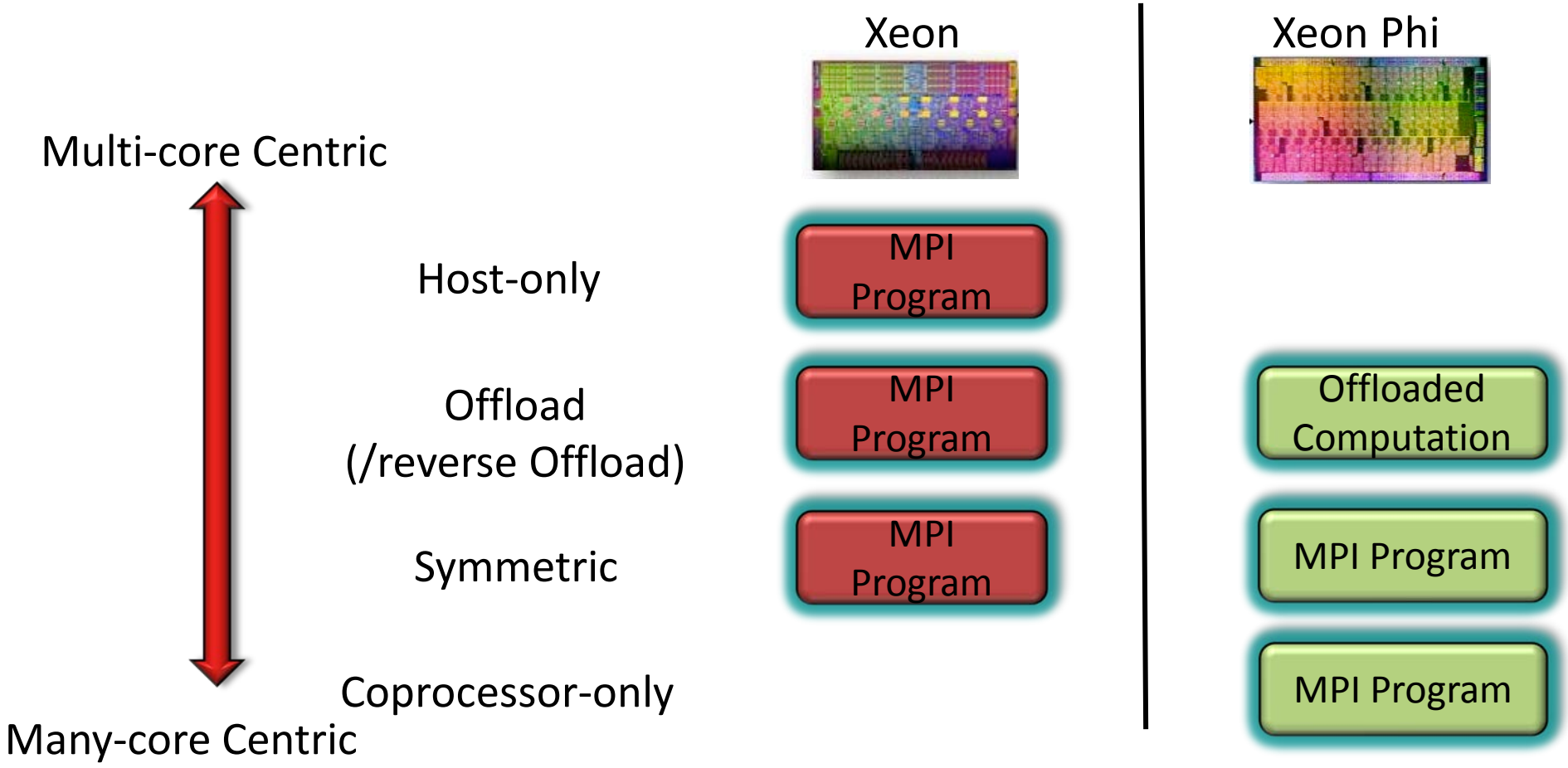
More Details in Student Poster Presentation

Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - MVAPICH2-X
  - **MVAPICH2-MIC**
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

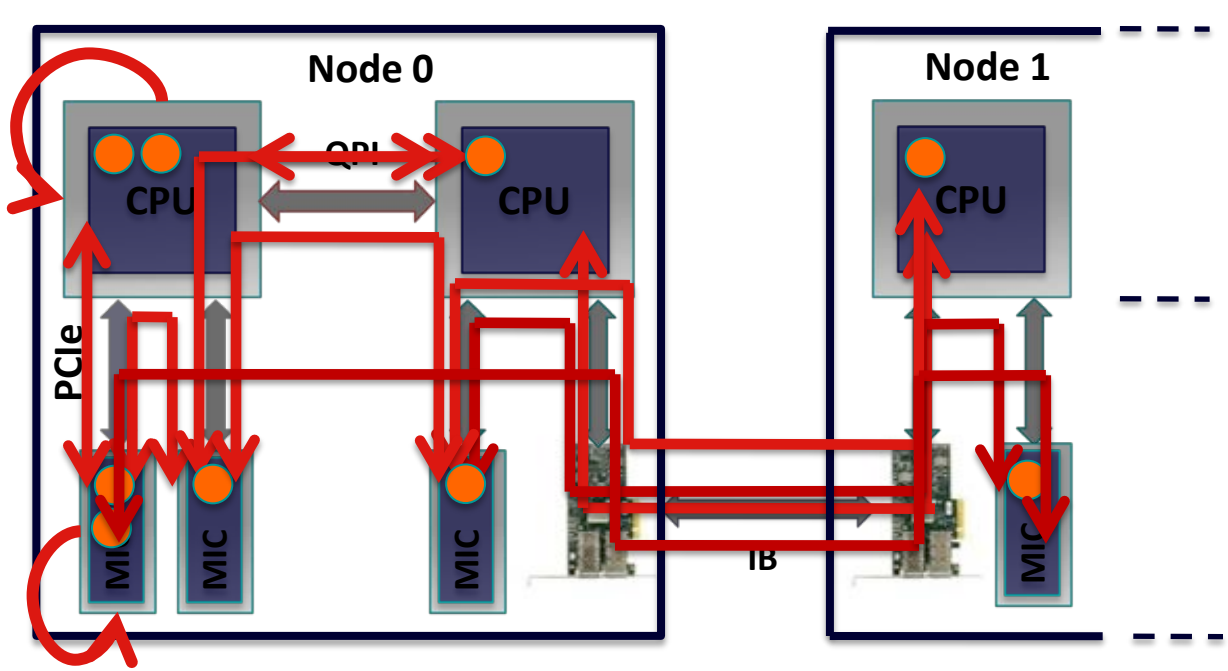
MPI Applications on MIC Clusters

- Flexibility in launching MPI jobs on clusters with Xeon Phi



Data Movement on Intel Xeon Phi Clusters

• Connected as PCIe devices – Flexibility but Complexity



MPI Process

1. Intra-Socket
2. Inter-Socket
3. Inter-Node
4. Intra-MIC
5. Intra-Socket MIC-MIC
6. Inter-Socket MIC-MIC
7. Inter-Node MIC-MIC
8. Intra-Socket MIC-Host
9. Inter-Socket MIC-Host
10. Inter-Node MIC-Host

11. Inter-Node MIC-MIC with IB adapter on remote socket and more . . .

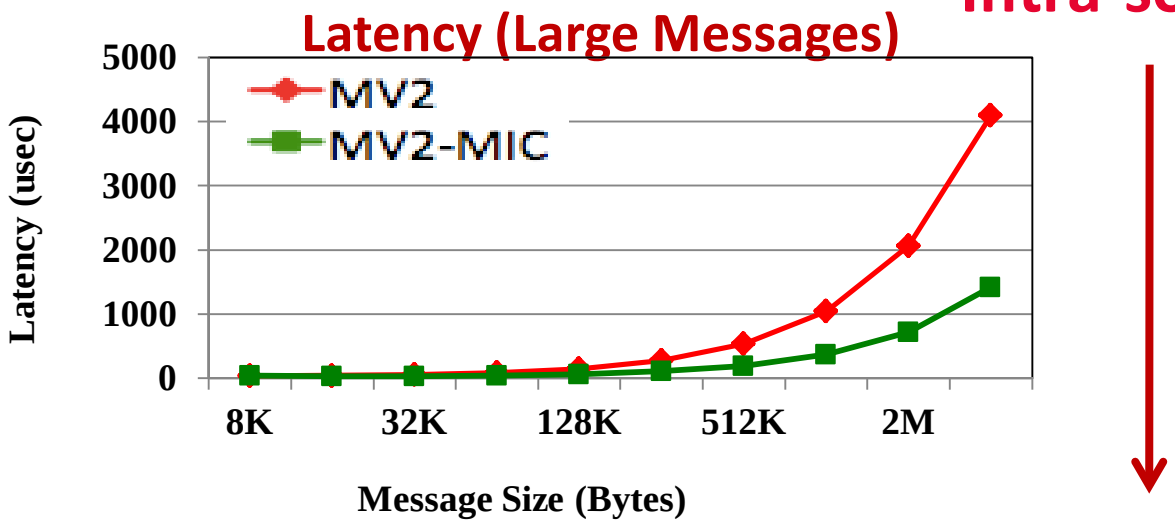
• Critical for runtimes to optimize data movement, hiding the complexity

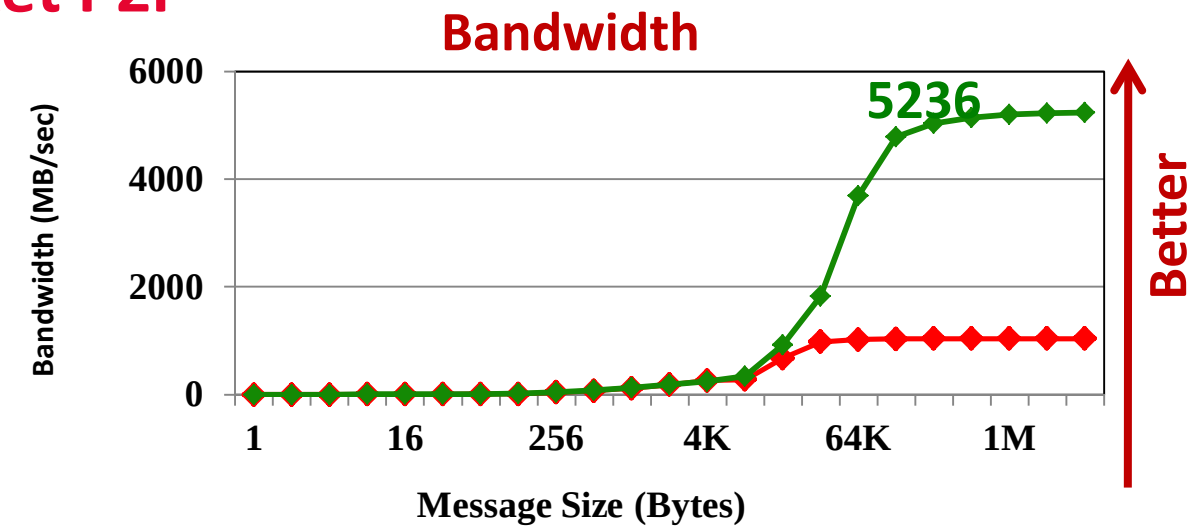
MVAPICH2-MIC 2.0 Design for Clusters with IB and MIC

- Offload Mode
- Intranode Communication
  - Coprocessor-only and Symmetric Mode
- Internode Communication
  - Coprocessors-only and Symmetric Mode
- Multi-MIC Node Configurations
- Running on three major systems
  - Stampede, Blueridge (Virginia Tech) and Beacon (UTK)

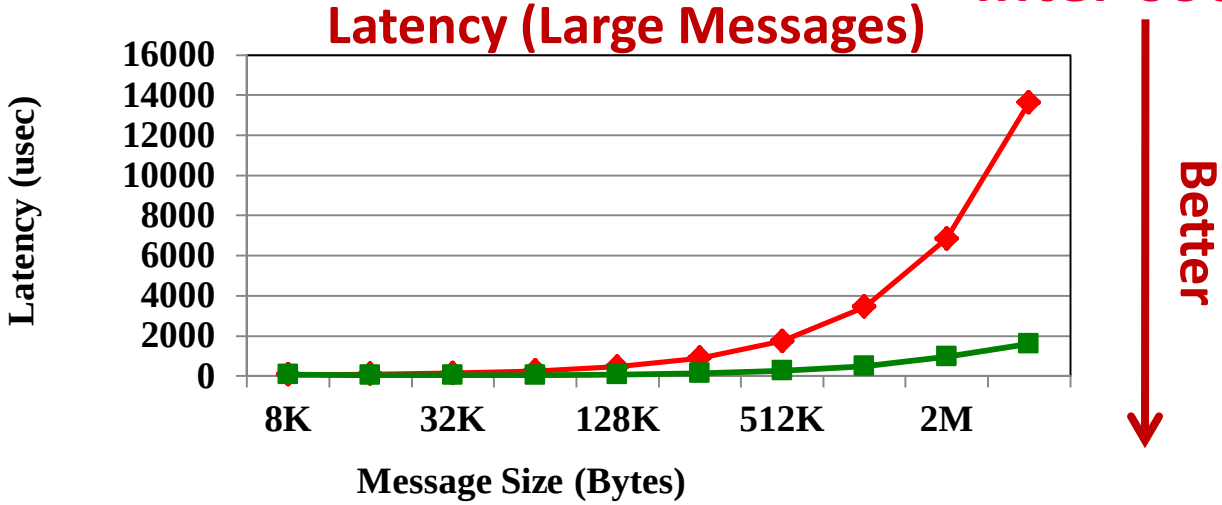
MIC-Remote-MIC P2P Communication with Proxy-based Communication

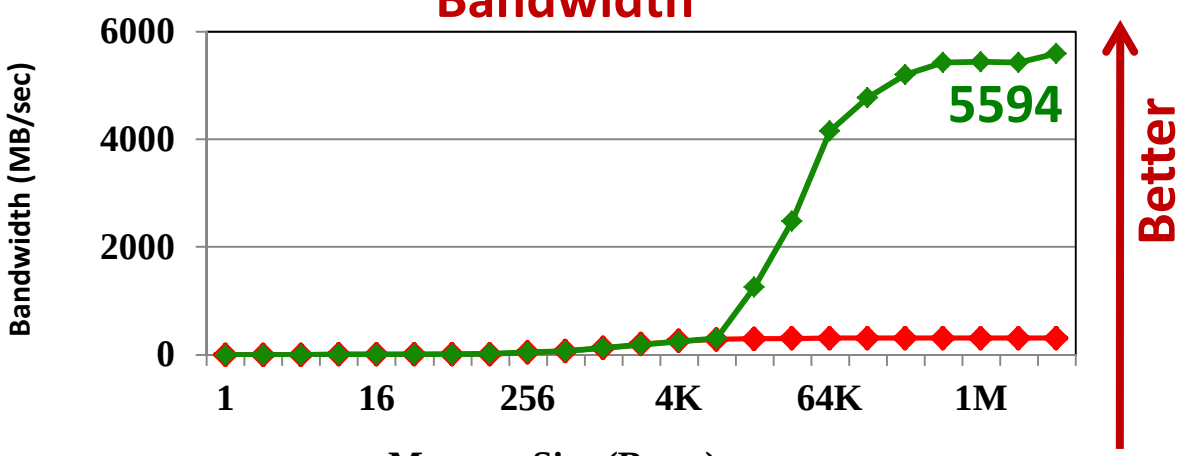
Intra-socket P2P



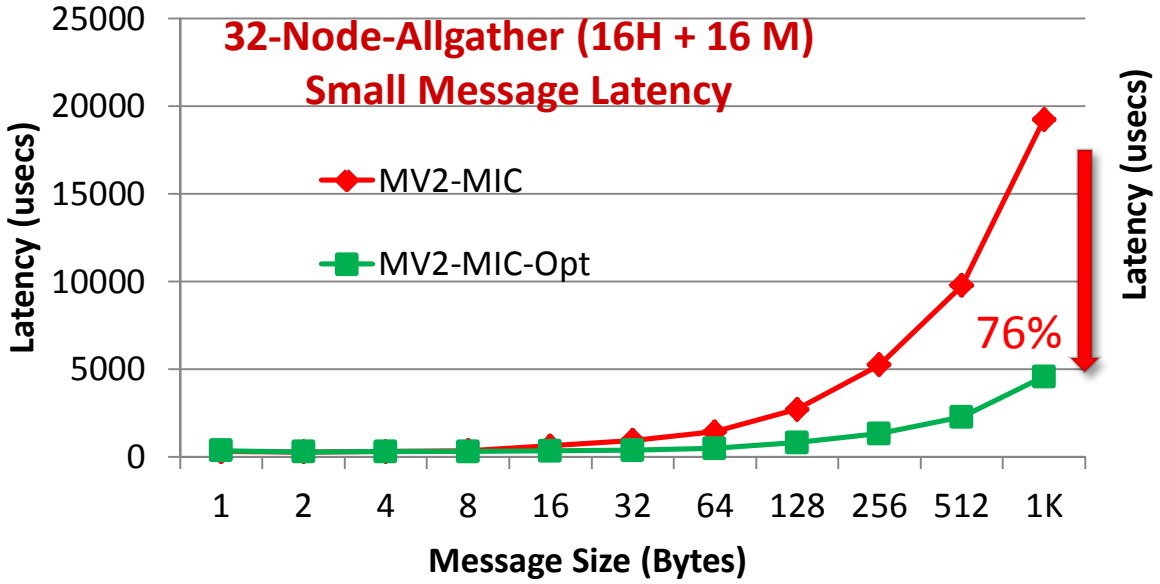


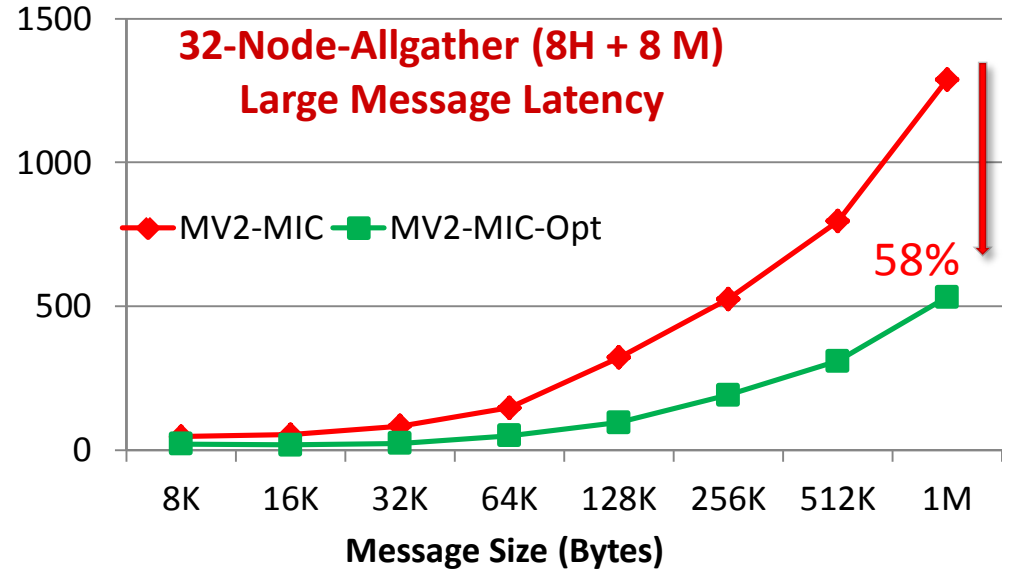
Inter-socket P2P

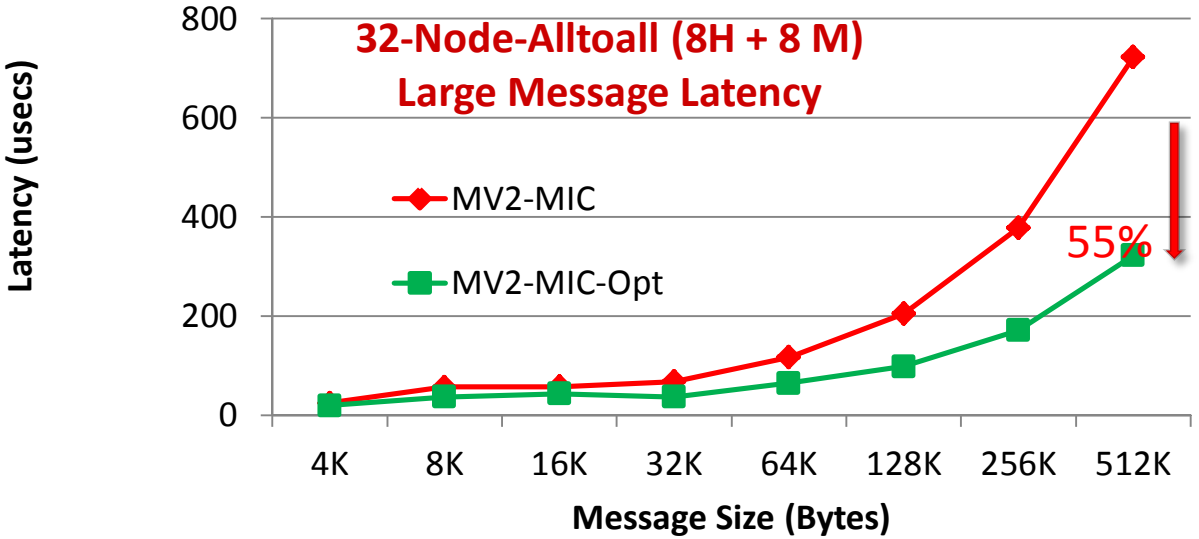


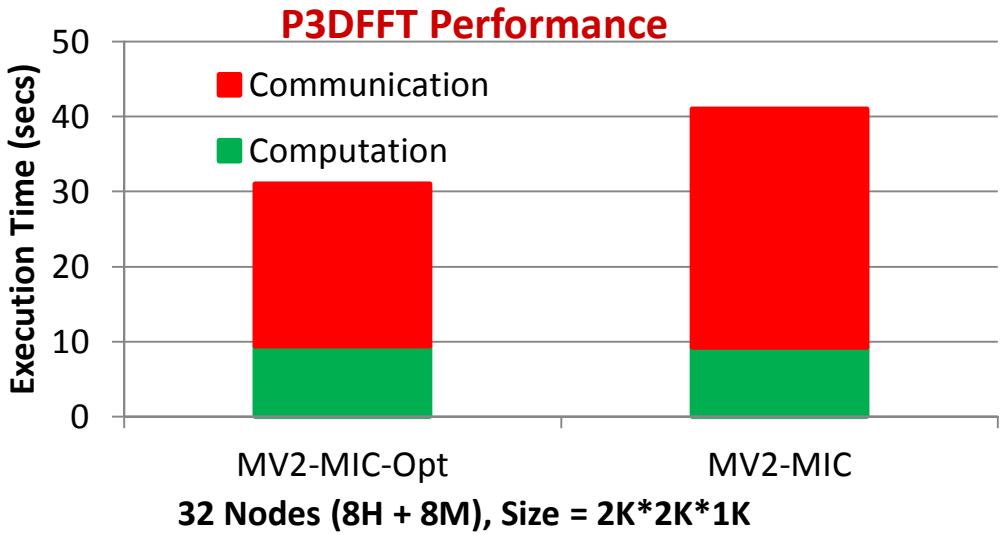


Optimized MPI Collectives for MIC Clusters (Allgather & Alltoall)









A. Venkatesh, S. Potluri, R. Rajachandrasekar, M. Luo, K. Hamidouche and D. K. Panda - High Performance Alltoall and Allgather designs for InfiniBand MIC Clusters; IPDPS'14, May 2014

Latest Status on MVAPICH2-MIC

- Running on three major systems
- Public version of MVAPICH2-MIC 2.0 is released (12/02/14)
  - http://mvapich.cse.ohio-state.edu/downloads/#mv2mic
- Enhanced version with support for KNL coming soon!

Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - **MVAPICH2-EA**
  - MVAPICH2-Virt
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

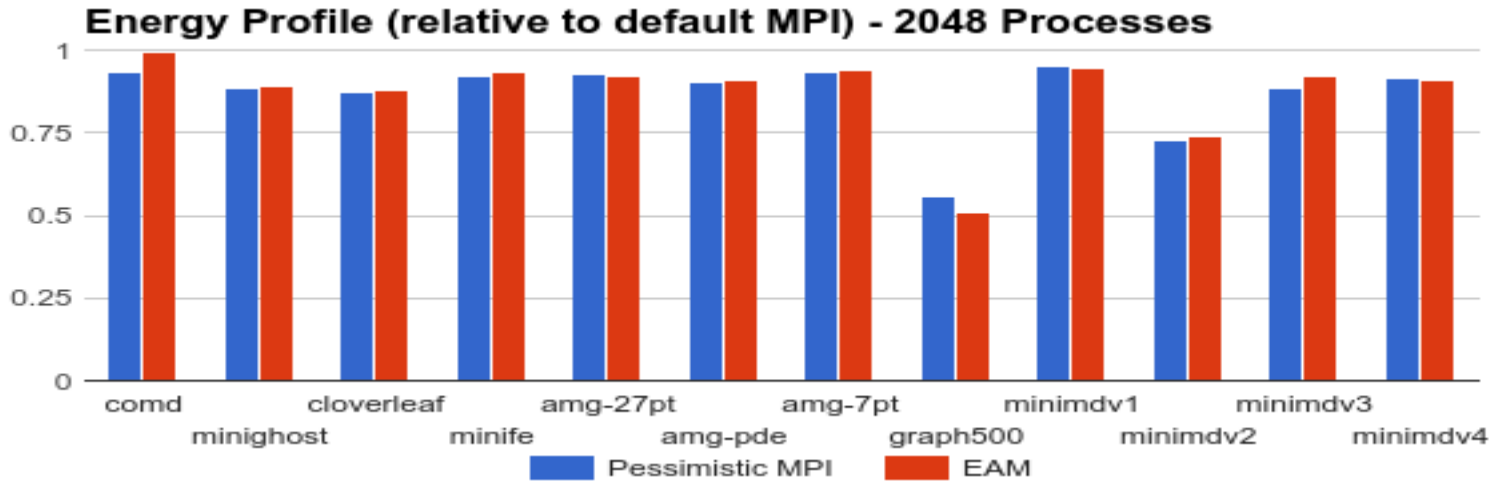
Energy-Aware MVAPICH2 & OSU Energy Management Tool (OEMT)

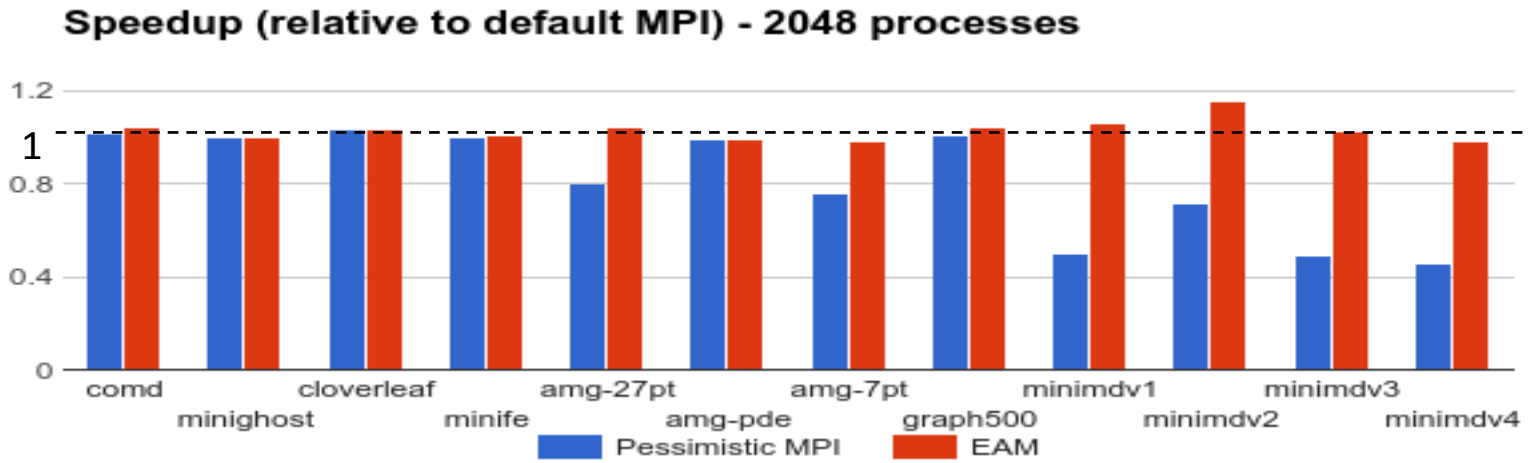
- MVAPICH2-EA 2.1 (Energy-Aware)
  - A white-box approach
  - New Energy-Efficient communication protocols for pt-pt and collective operations
  - Intelligently apply the appropriate Energy saving techniques
  - Application oblivious energy saving

- OEMT
  - A library utility to measure energy consumption for MPI applications
  - Works with all MPI runtimes
  - PRELOAD option for precompiled applications
  - Does not require ROOT permission:
    - A safe kernel module to read only a subset of MSRs

MVAPICH2-EA: Application Oblivious Energy-Aware-MPI (EAM)

- An energy efficient runtime that provides energy savings without application knowledge
- Uses automatically and transparently the best energy lever
- Provides guarantees on maximum degradation with 5-41% savings at <= 5% degradation
- Pessimistic MPI applies energy reduction lever to each MPI call





A Case for Application-Oblivious Energy-Efficient MPI Runtime A. Venkatesh, A. Vishnu, K. Hamidouche, N. Tallent, D. K. Panda, D. Kerbyson, and A. Hoise, Supercomputing '15, Nov 2015 *[Best Student Paper Finalist]*

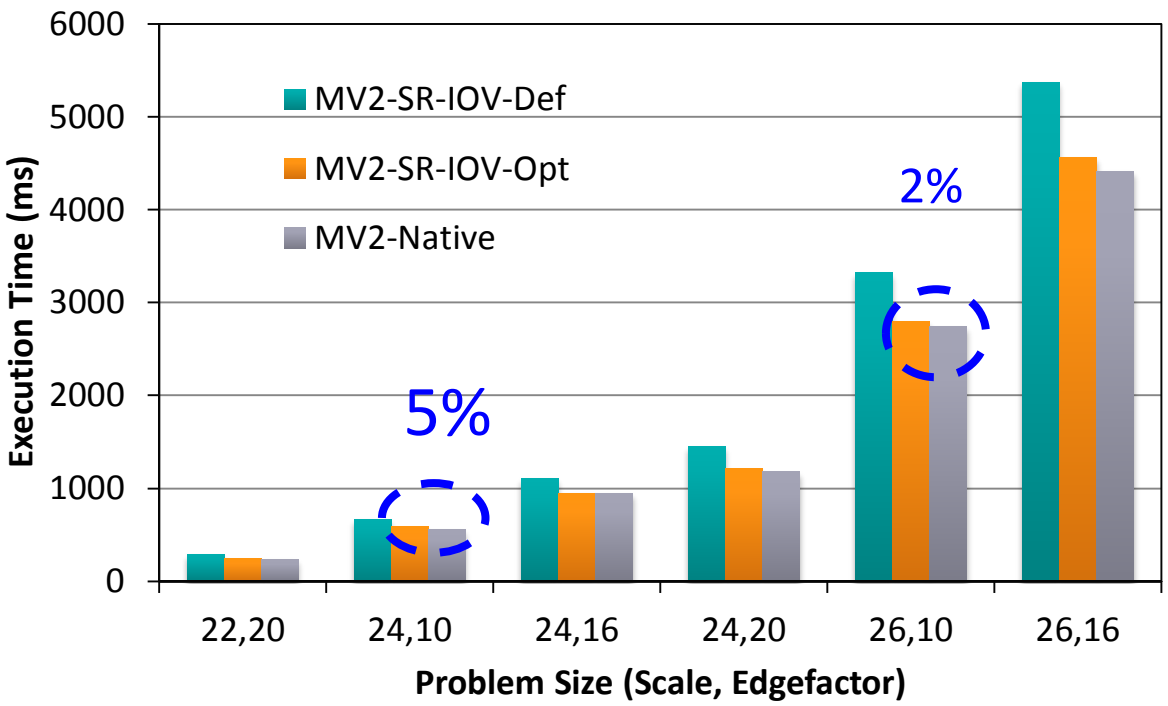
Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - **MVAPICH2-Virt**
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

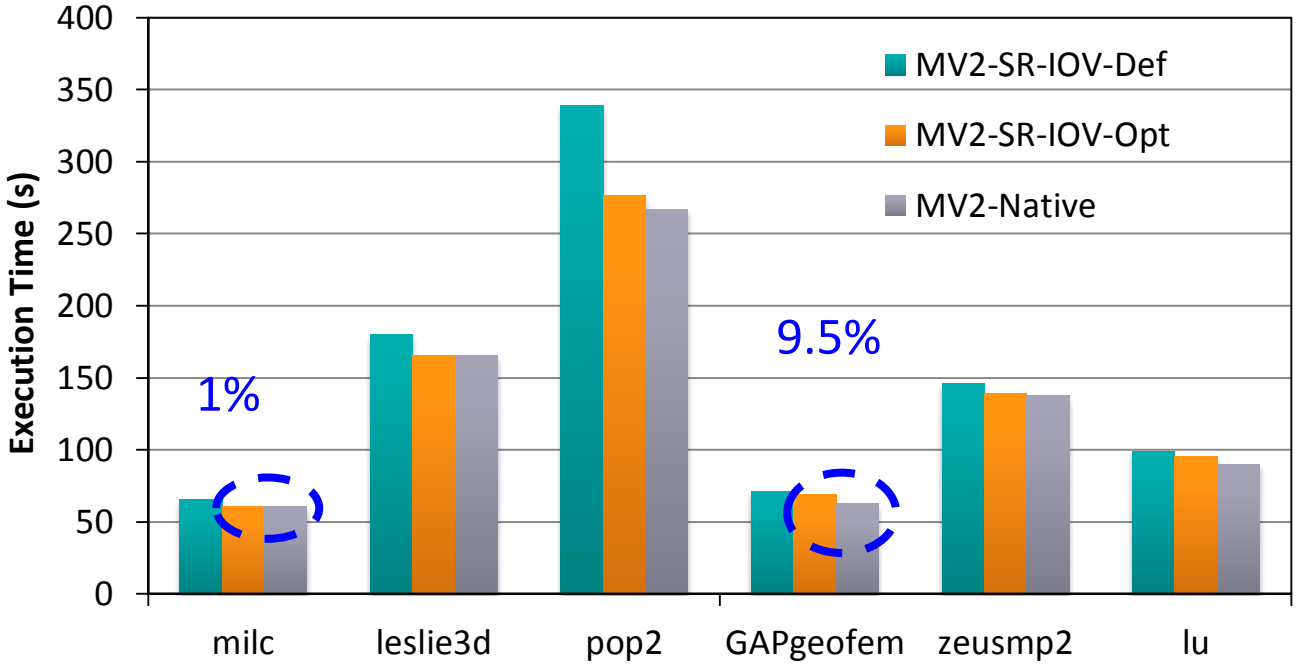
MVAPICH2-Virt 2.2rc1

- Released on 07/12/2016
- Major Features and Enhancements
  - Based on MVAPICH2 2.2rc1
  - High-performance and locality-aware MPI communication with IPC-SHM and CMA for containers
  - Support for locality auto-detection in containers
  - Automatic communication channel selection among IPC-SHM, CMA, and HCA
  - Support for easy configuration through runtime parameters
  - Tested with
    - Docker 1.9.1 and 1.10.3
    - Mellanox InfiniBand adapters (ConnectX-3 (56Gbps))

Application-Level Performance on Chameleon (SR-IOV Support)



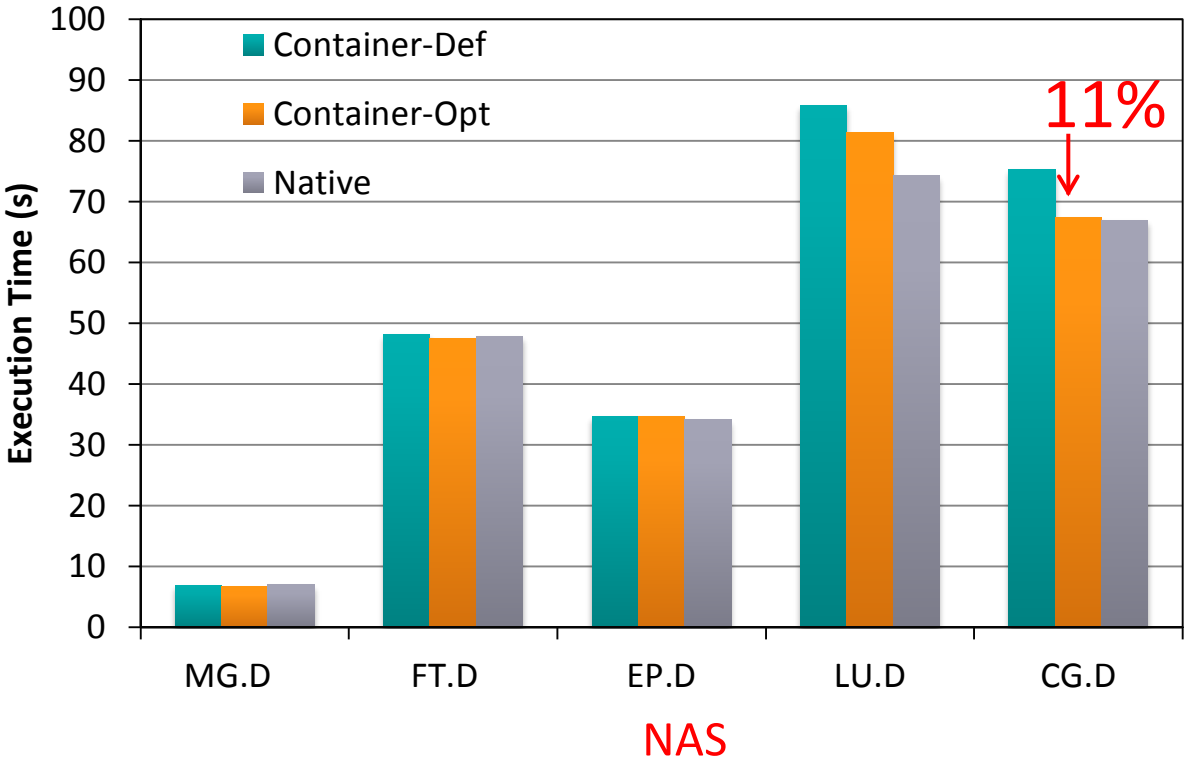
Graph500



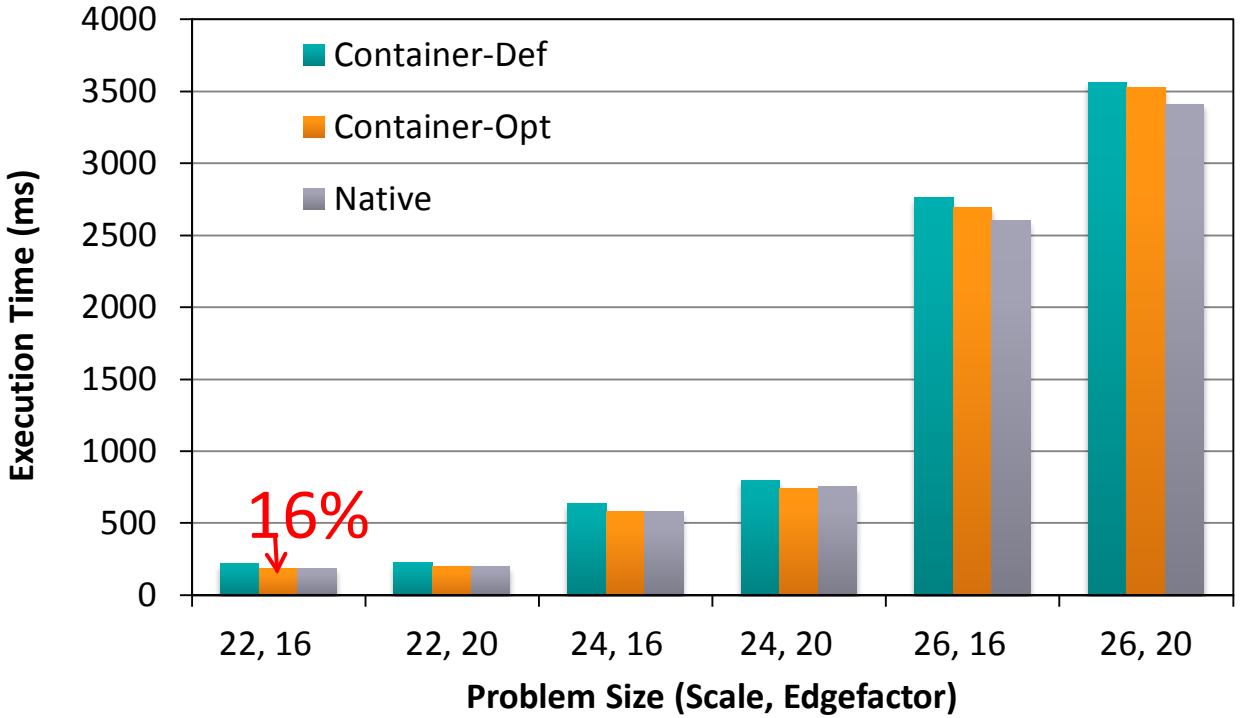
SPEC MPI2007

- 32 VMs, 6 Core/VM
- Compared to Native, 2-5% overhead for Graph500 with 128 Procs
- Compared to Native, 1-9.5% overhead for SPEC MPI2007 with 128 Procs

Application-Level Performance on Chameleon (Containers Support)



NAS



Graph 500

- 64 Containers across 16 nodes, pining 4 Cores per Container
- Compared to Container-Def, up to 11% and 16% of execution time reduction for NAS and Graph 500
- Compared to Native, less than 9 % and 4% overhead for NAS and Graph 500

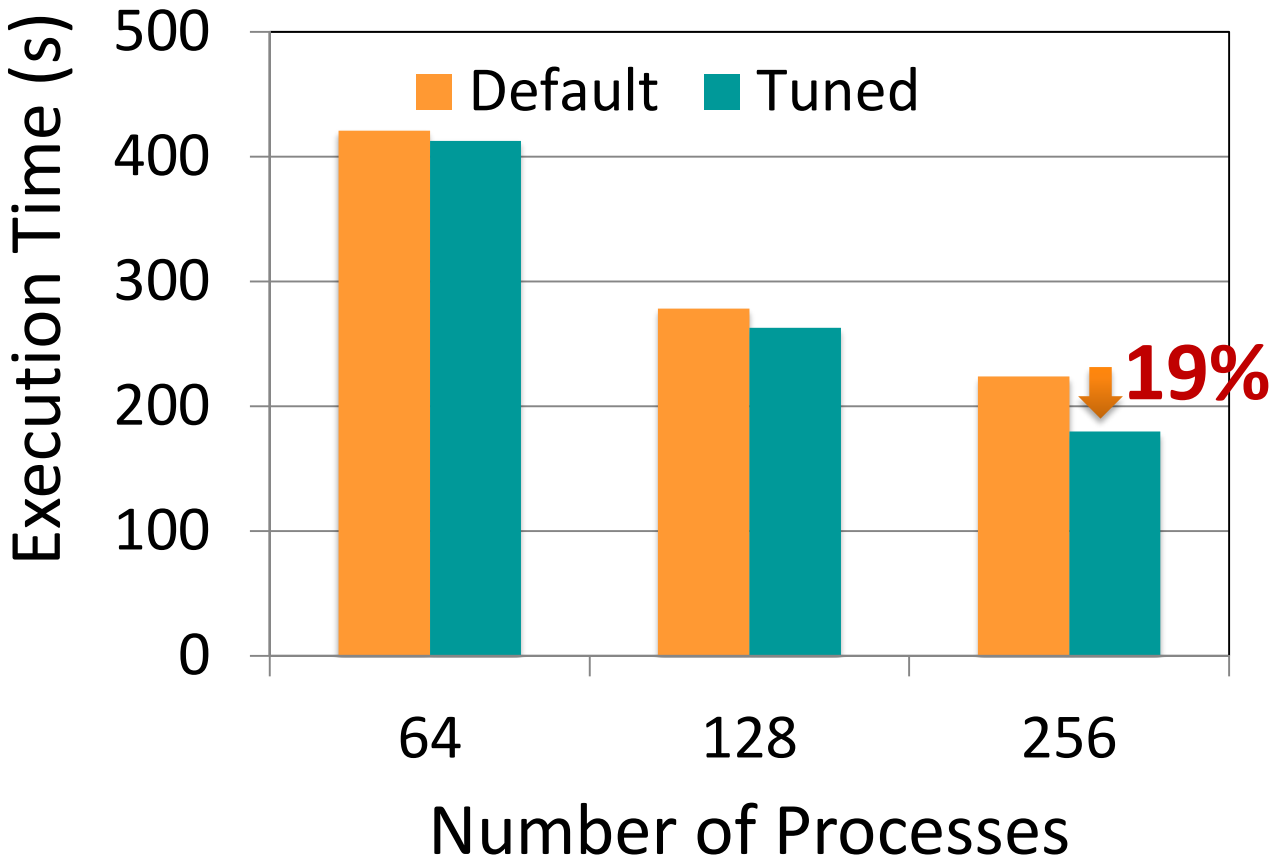
Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- **Application Best Practices**
  - **Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue**
- Overview of Configuration and Debugging Support
- Conclusions and Final Q&A

Applications-Level Tuning: Compilation of Best Practices

- MPI runtime has many parameters
- Tuning a set of parameters can help you to extract higher performance
- Compiled a list of such contributions through the MVAPICH Website
  - http://mvapich.cse.ohio-state.edu/best_practices/
- Initial list of applications
  - Amber
  - HoomDBlue
  - HPCG
  - Lulesh
  - MILC
  - Neuron
  - SMG2000
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.

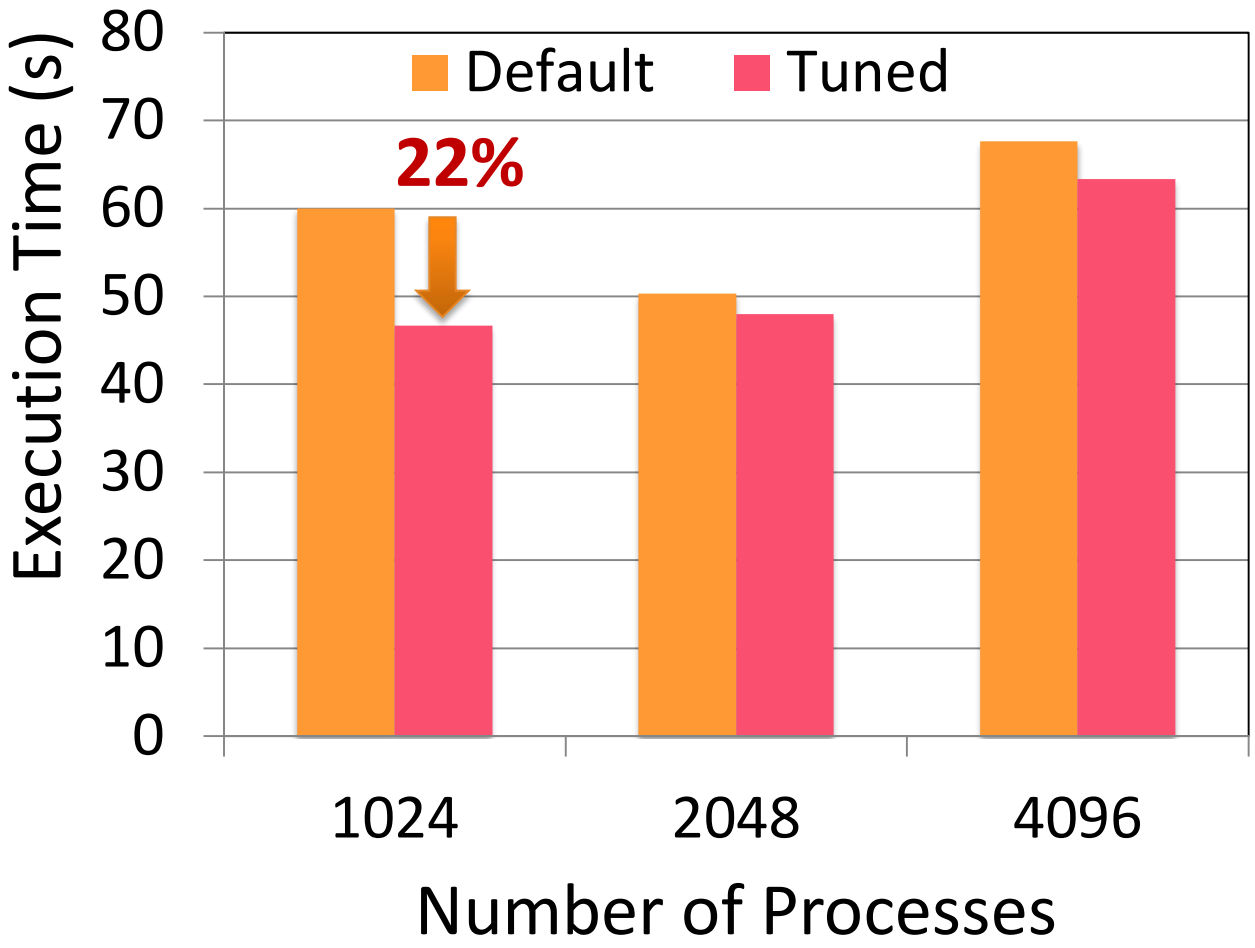
Amber: Impact of Tuning Eager Threshold



Data Submitted by: Dong Ju Choi @ UCSD

- Tuning the Eager threshold has a significant impact on application performance by avoiding the synchronization of rendezvous protocol and thus yielding better communication computation overlap
- 19% improvement in overall execution time at 256 processes
- Library Version: MVAPICH2 2.2b
- MVAPICH Flags used
  - MV2_IBA_EAGER_THRESHOLD=131072
  - MV2_VBUF_TOTAL_SIZE=131072
- Input files used
  - Small: MDIN
  - Large: PMTOP

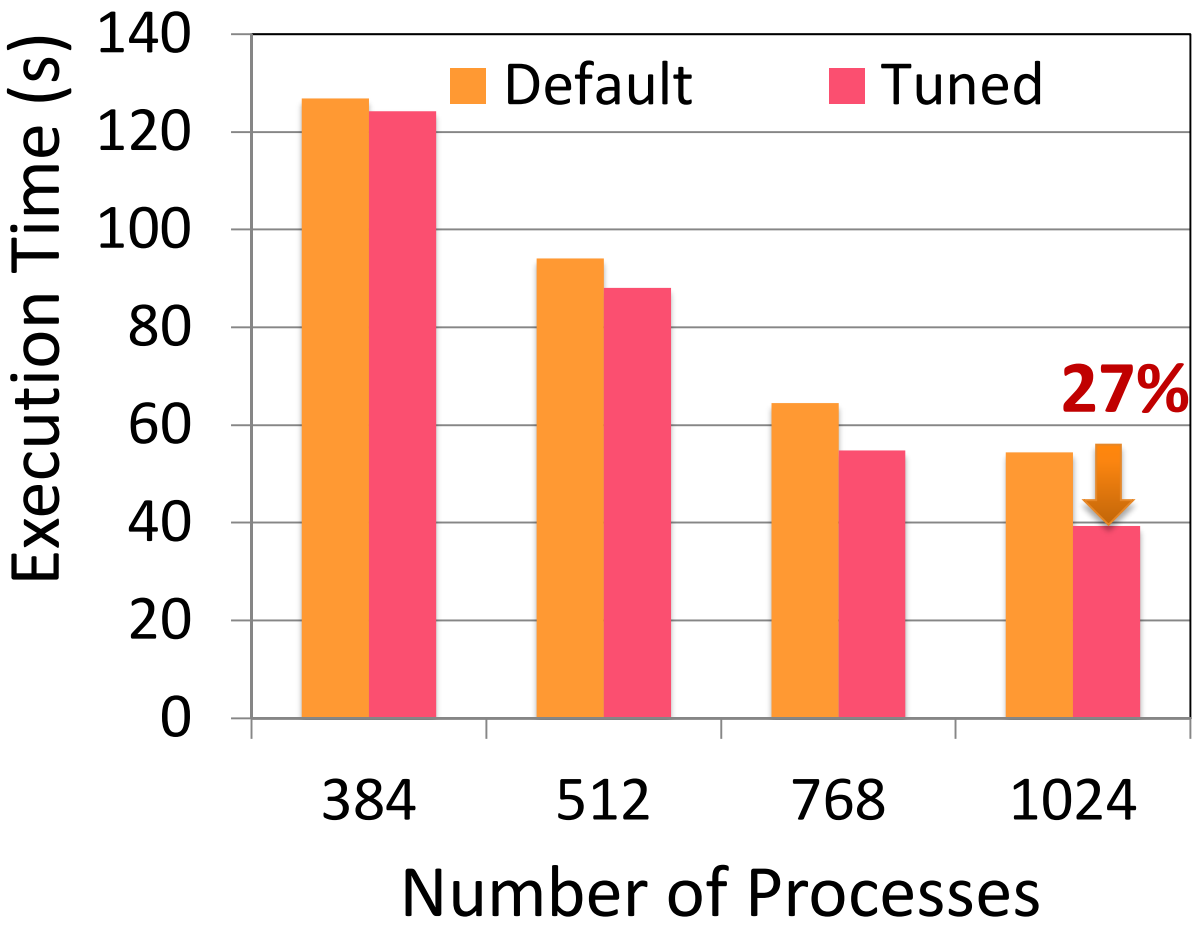
SMG2000: Impact of Tuning Transport Protocol



Data Submitted by Jerome Vienne @ TACC

- UD-based transport protocol selection benefits the SMG2000 application
- 22% and 6% on 1,024 and 4,096 cores, respectively
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - MV2_USE_ONLY_UD=1
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

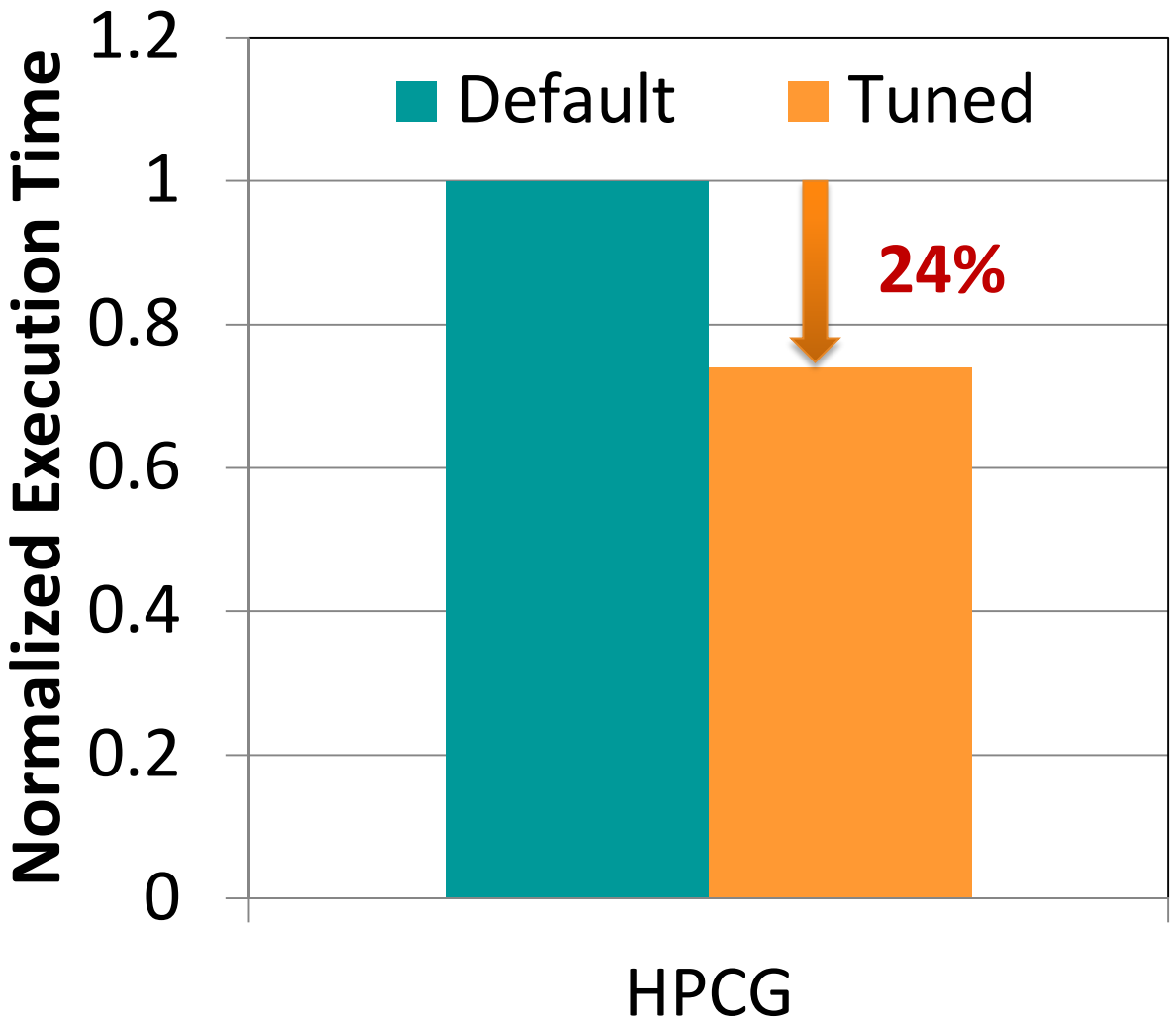
Neuron: Impact of Tuning Transport Protocol



Data Submitted by Mahidhar Tatineni @ SDSC

- UD-based transport protocol selection benefits the Neuron application
- 15% and 27% improvement is seen for 768 and 1,024 processes respectively
- Library Version: MVAPICH2 2.2b
- MVAPICH Flags used
  - MV2_USE_ONLY_UD=1
- Input File
  - YuEtAl2012
- System Details
  - Comet@SDSC
  - Haswell nodes with dual 12-cores socket per node and Mellanox FDR (56 Gbps) network.

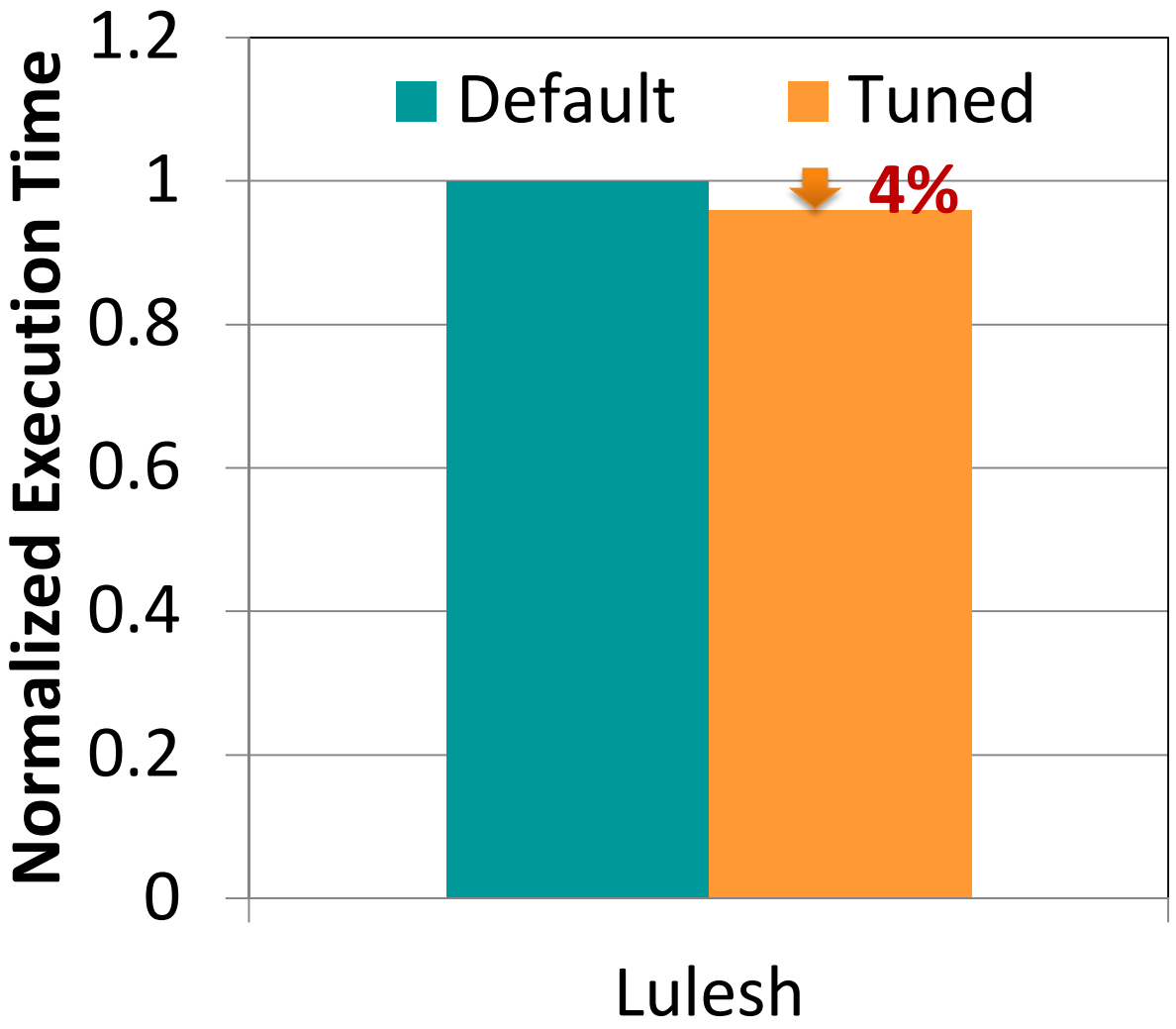
HPCG: Impact of Collective Tuning for MPI+OpenMP Programming Model



- Partial subscription nature of hybrid MPI+OpenMP programming requires a new level of collective tuning
  - For PPN=2 (Processes Per Node), the tuned version of MPI_Reduce shows 51% improvement on 2,048 cores
- 24% improvement on 512 cores
  - 8 OpenMP threads per MPI processes
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - The tuning parameters for hybrid MPI+OpenMP programming models is on by default from MVAPICH2-2.1 onward
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

Data Submitted by Jerome Vienne and Carlos Rosales-Fernandez @ TACC

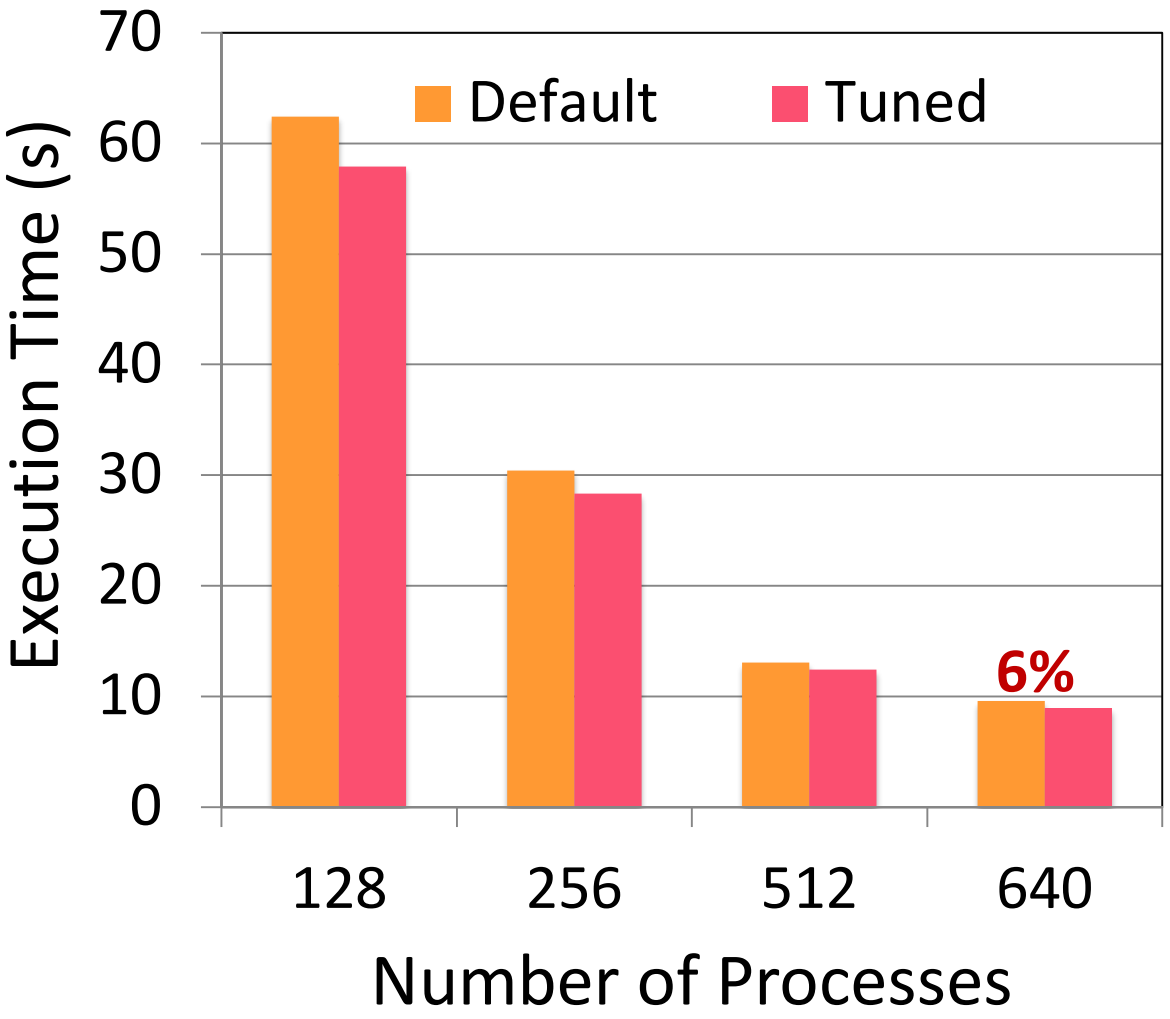
LULESH: Impact of Collective Tuning for MPI+OpenMP Programming Model



- Partial subscription nature of hybrid MPI+OpenMP programming requires a new level of collective tuning
  - For PPN=2 (Processes Per Node), the tuned version of MPI_Reduce shows 51% improvement on 2,048 cores
- 4% improvement on 512 cores
  - 8 OpenMP threads per MPI processes
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - The tuning parameters for hybrid MPI+OpenMP programming models is on by default from MVAPICH2-2.1 onward
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

Data Submitted by Jerome Vienne and Carlos Rosales-Fernandez @ TACC

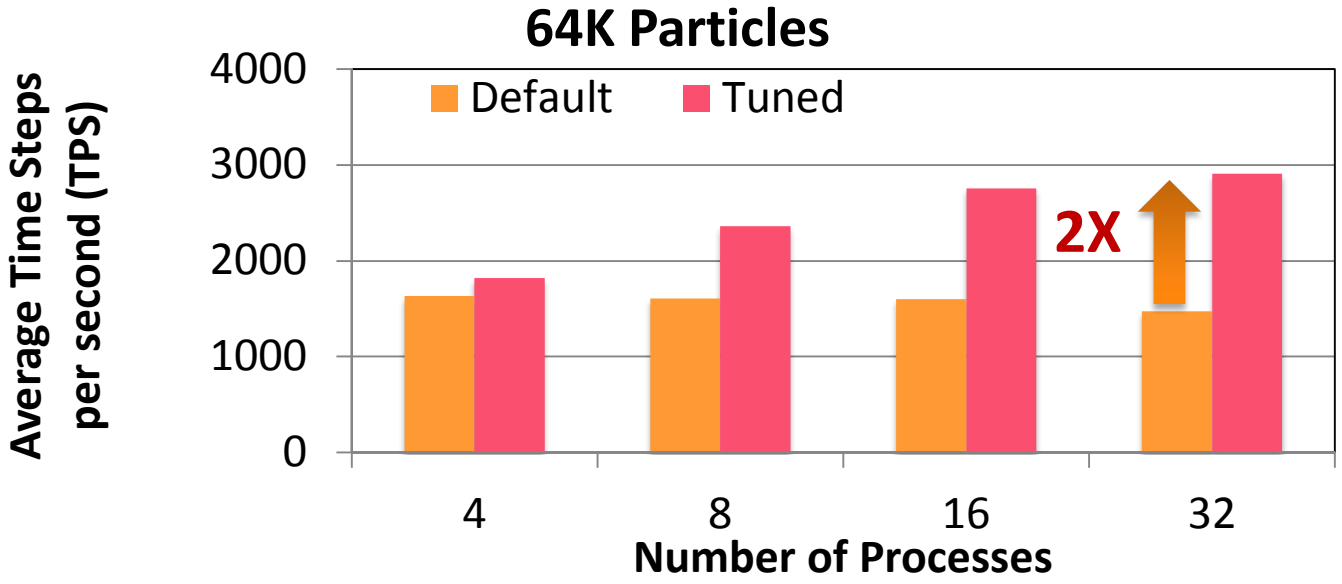
MILC: Impact of User-mode Memory Registration (UMR) based tuning

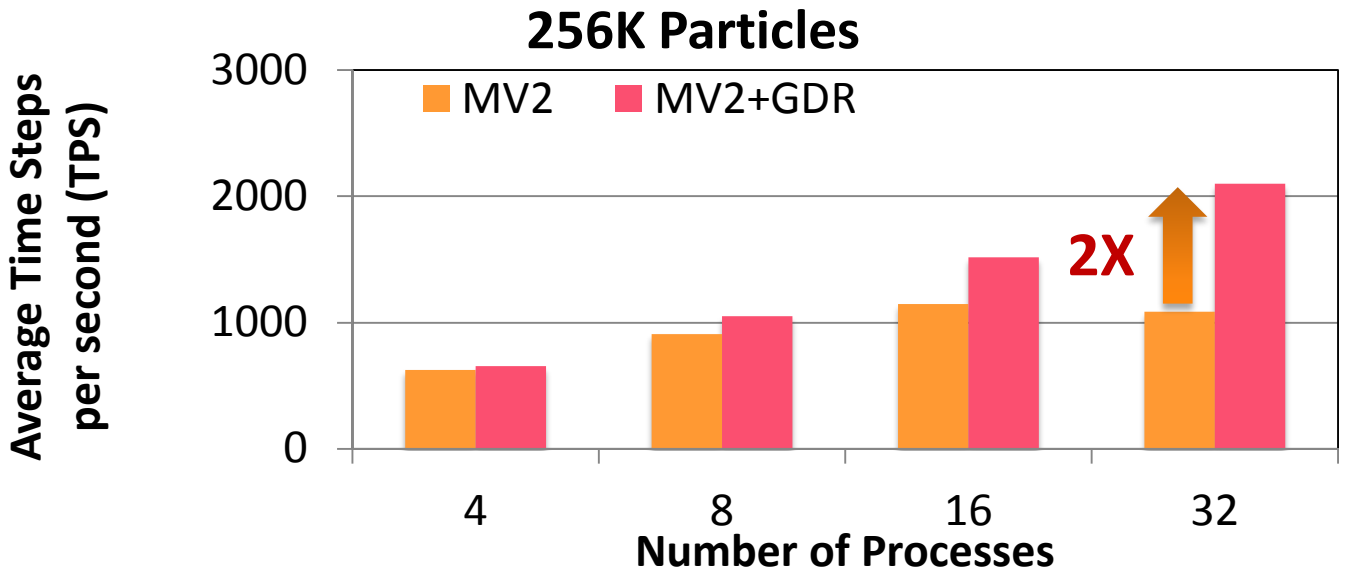


Data Submitted by Mingzhe Li @ OSU

- Non-contiguous data processing is very common on HPC applications. MVAPICH2 offers efficient designs for MPI Datatype support using novel hardware features such as UMR
- UMR-based protocol selection benefits the MILC application.
  - 4% and 6% improvement in execution time at 512 and 640 processors, respectively
- Library Version: MVAPICH2-X 2.2b
- MVAPICH Flags used
  - MV2_USE_UMR=1
- System Details
  - The experimental cluster consists of 32 Ivy Bridge Compute nodes interconnected by Mellanox FDR.
  - The Intel Ivy Bridge processors consist of Xeon dual ten-core sockets operating at 2.80GHz with 32GB RAM and Mellanox OFED version 3.2-1.0.1.1.

HOOMD-blue: Impact of GPUDirect RDMA Based Tuning





Data Submitted by Khaled Hamidouche @ OSU

- HOOMD-blue is a Molecular Dynamics simulation using a custom force field.
- GPUDirect specific features selection and tuning significantly benefit the HOOMD-blue application. We observe a factor of 2X improvement on 32 GPU nodes, with both 64K and 256K particles
- Library Version: MVAPICH2-GDR 2.2b
- MVAPICH-GDR Flags used
  - MV2_USE_CUDA=1
  - MV2_USE_GPUDIRECT=1
  - MV2_GPUDIRECT_GDRCOPY=1
- System Details
  - Wilkes@Cambridge
  - 128 Ivybridge nodes, each node is a dual 6-cores socket with Mellanox FDR

Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- **Overview of Configuration and Debugging Support**
  - **User Resources**
  - **Getting help and Bug report detail**
- Conclusions and Final Q&A

User Resources

- MVAPICH2 Quick Start Guide
- MVAPICH2 User Guide
  - Long and very detailed
  - FAQ
- MVAPICH2 Web-Site
  - Overview and Features
  - Reference performance
  - Publications
- Mailing List Support
  - mvapich-discuss
- Mailing List Archives
- All above resources accessible from: http://mvapich.cse.ohio-state.edu/

Getting Help

- Check the MVAPICH2 FAQ
- Check the Mailing List Archives
- Basic System Diagnostics
  - ibv_devinfo - at least one port should be PORT_ACTIVE
  - ulimit -l - should be “unlimited” on all compute nodes
  - host resolution: DNS or /etc/hosts
  - password-less ssh login
  - run IB perf tests for all the message sizes(-a option)
    - Ib_send_lat, ib_send_bw
  - run system program (like hostname) and MPI hello world program

Getting Help (Cont.)

- More diagnostics
  - Already fixed issue: always try with latest release
  - Regression: verifying with previous release
  - Application issue: verify with other MPI libraries
  - Launcher issue: verifying with multiple launchers (mpirun_rsh, mpiexec.hydra)
  - Debug mode
  - Compiler optimization issues: try with different compiler

Submitting Bug Report

- Subscribe to mvapich-discuss and send problem report
- Include as much information as possible
- Run-time issues
  - Config flags (“mpiname –a” output)
  - Exact command used to run the application
  - Run-rime parameters in the environment
  - Standalone reproducer program
  - Information about the IB network
    - OFED version
    - ibv_devinfo
  - Remote system access

Submitting Bug Report (Cont.)

- Build and Installation issues
  - MVAPICH2 version
  - Compiler version
  - Platform details (OS, kernel version..etc)
  - Configure flags
  - Attach Config.log file
  - Attach configure, make and make install step output
    - ./configure {–flags} 2>&1 | tee config.out
    - Make 2>&1 | tee make.out
    - Make install 2>&1 | tee install.out

Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-GDR
  - MVAPICH2-X
  - MVAPICH2-MIC
  - MVAPICH2-EA
  - MVAPICH2-Virt
- Application Best Practices
- Overview of Configuration and Debugging Support
- **Conclusions and Final Q&A**

Looking into the Future ….

- Exascale systems will be constrained by
  - Power
  - Memory per core
  - Data movement cost
  - Faults
- Programming Models and Runtimes for HPC need to be designed for
  - Scalability
  - Performance
  - Fault-resilience
  - Energy-awareness
  - Programmability
  - Productivity
- Highlighted some of the issues and challenges
- Need continuous innovation on all these fronts

MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF …)
  - MPI + Task*
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features of Mellanox InfiniBand
  - Switch-IB2 SHArP*
  - GID-based support*
- Enhanced communication schemes for upcoming architectures
  - Knights Landing with MCDRAM*
  - NVLINK*
  - CAPI*
- Extended topology-aware collectives
- Extended Energy-aware designs and Virtualization Support
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended Checkpoint-Restart and migration support with SCR
- Support for * features will be available in future MVAPICH2 Releases

Concluding Remarks

- Provided an overview of the MVAPICH2 software libraries
- Presented in-depth details on configuration and runtime parameters, optimizations and their impacts
- Provided an overview of debugging support
- Summarized the impact of optimization and tuning by creating a set of "Application Best Practices"
- Demonstrated how users can take advantage of these optimization techniques to extract performance and scalability while using various MVAPICH2 software libraries

International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)

ESPM2 2016 will be held with the Supercomputing Conference (SC '16), at Salt Lake City, Utah, on Friday, November 18th, 2016

http://web.cse.ohio-state.edu/~hamidouc/ESPM2/espm2_16.html#program

In Cooperation with ACM SIGHPC

Paper Submission Deadline: August 26th, 2016

Author Notification: September 30th, 2016

Camera Ready: October 7th, 2016

ESPM2 2015 was held with the Supercomputing Conference (SC '15), at Austin, Texas, on Sunday, November 15th, 2015

http://web.cse.ohio-state.edu/~hamidouc/ESPM2/espm2.html#program

Funding Acknowledgments

Funding Support by

Equipment Support by

Personnel Acknowledgments

Current Students

- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Gugnani (Ph.D.)
- J. Hashimi (Ph.D.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- K. Kulkarni (M.S.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Past Students

- P. Balaji (Ph.D.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- 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.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)

Past Post-Docs

- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne
- D. Banerjee
- J. Lin

Current Research Scientists

- K. Hamidouche
- X. Lu
- H. Subramoni

Current Research Specialist

- M. Arnold
- J. Perkins

- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)

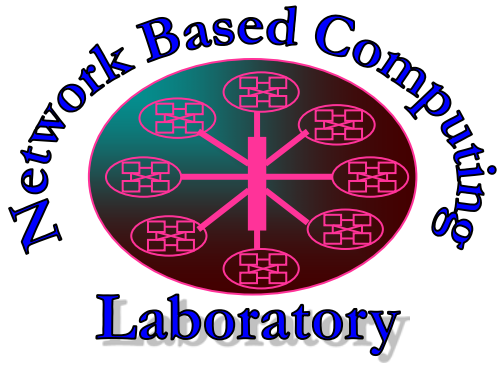
Past Research Scientist

- S. Sur

Past Programmers

- D. Bureddy

Thank You!



Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH Project
http://mvapich.cse.ohio-state.edu/

panda@cse.ohio-state.edu, subramon@cse.ohio-state.edu, hamidouc@cse.ohio-state.edu