



**MVAPICH**

MPI, PGAS and Hybrid MPI+PGAS Library

# How to Tune and Extract Higher Performance with MVAPICH2 Libraries?

**An XSEDE ECSS webinar**

by

**Dhabaleswar K. (DK) Panda**

The Ohio State University

E-mail: [panda@cse.ohio-state.edu](mailto:panda@cse.ohio-state.edu)

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

# Drivers of Modern HPC Cluster Architectures



Multi-core Processors

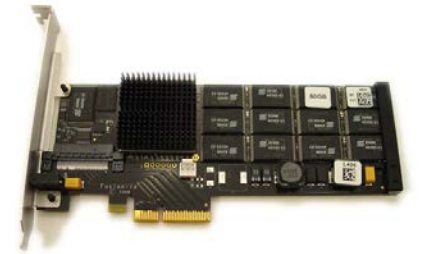


High Performance Interconnects -  
InfiniBand

<1usec latency, 100Gbps Bandwidth>



Accelerators / Coprocessors  
high compute density, high  
performance/watt  
>1 TFlop DP on a chip



SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs and Intel Xeon Phi)



*Tianhe – 2*



*Titan*

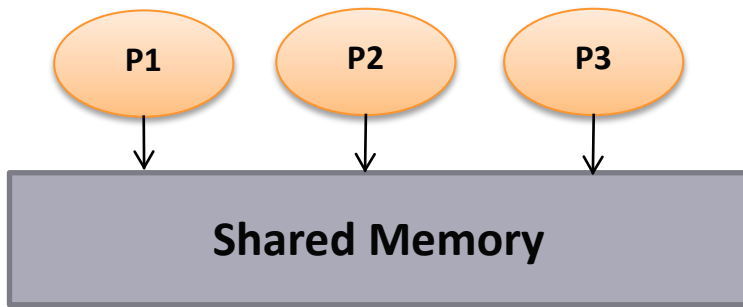


*Stampede*



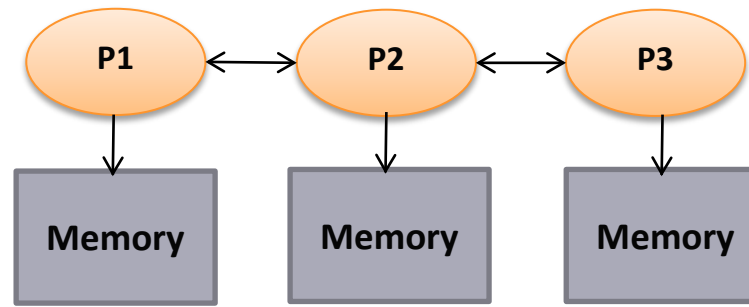
*Tianhe – 1A*

# Parallel Programming Models Overview



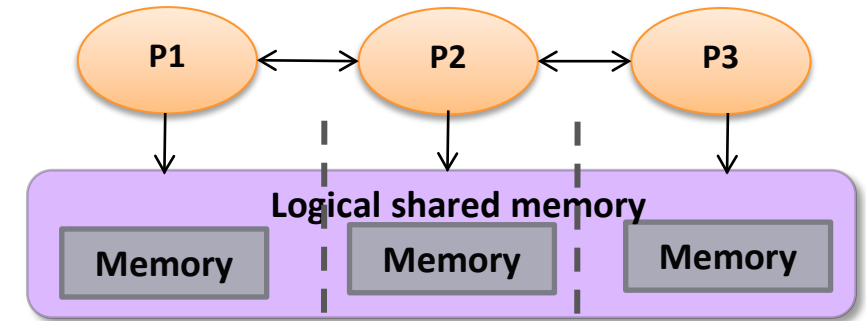
Shared Memory Model

SHMEM, DSM



Distributed Memory Model

MPI (Message Passing Interface)



Partitioned Global Address Space (PGAS)

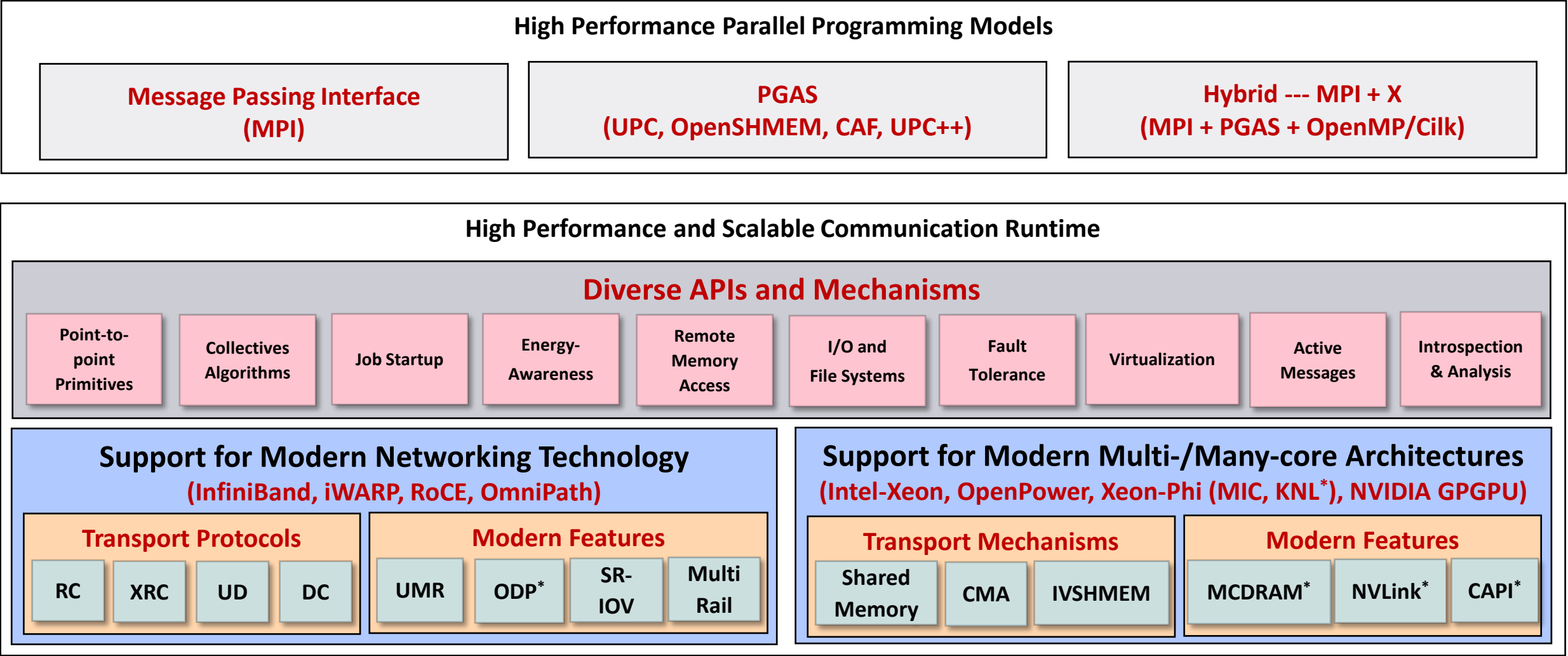
Global Arrays, UPC, Chapel, X10, CAF, ...

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

# Overview of the MVAPICH2 Project

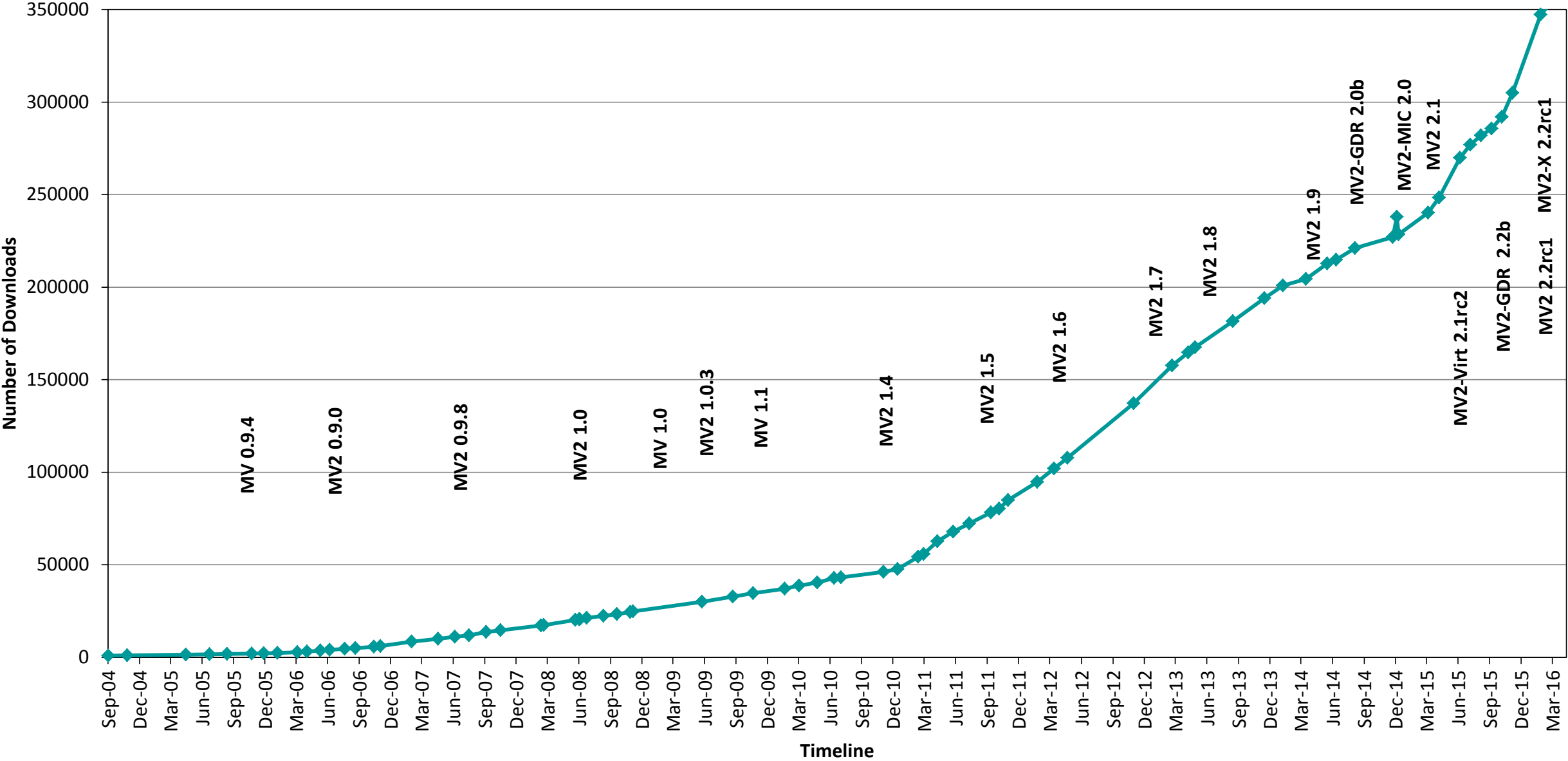
- High Performance open-source MPI Library for InfiniBand, 10-40Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - **Used by more than 2,550 organizations in 79 countries**
  - **More than 363,000 (> 0.36 million) downloads from the OSU site directly**
  - Empowering many TOP500 clusters (Nov '15 ranking)
    - 10<sup>th</sup> ranked 519,640-core cluster (Stampede) at TACC
    - 13<sup>th</sup> ranked 185,344-core cluster (Pleiades) at NASA
    - 25<sup>th</sup> ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - <http://mvapich.cse.ohio-state.edu>
- Empowering Top500 systems for over a decade
  - System-X from Virginia Tech (3<sup>rd</sup> in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  - Stampede at TACC (10<sup>th</sup> in Nov'15, 519,640 cores, 5.168 Plops)

# MVAPICH2 Architecture



\* Upcoming

# MVAPICH/MVAPICH2 Release Timeline and Downloads



# MVAPICH2 Software Family



Requirements	MVAPICH2 Library to use
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

# Strong Procedure for Design, Development and Release

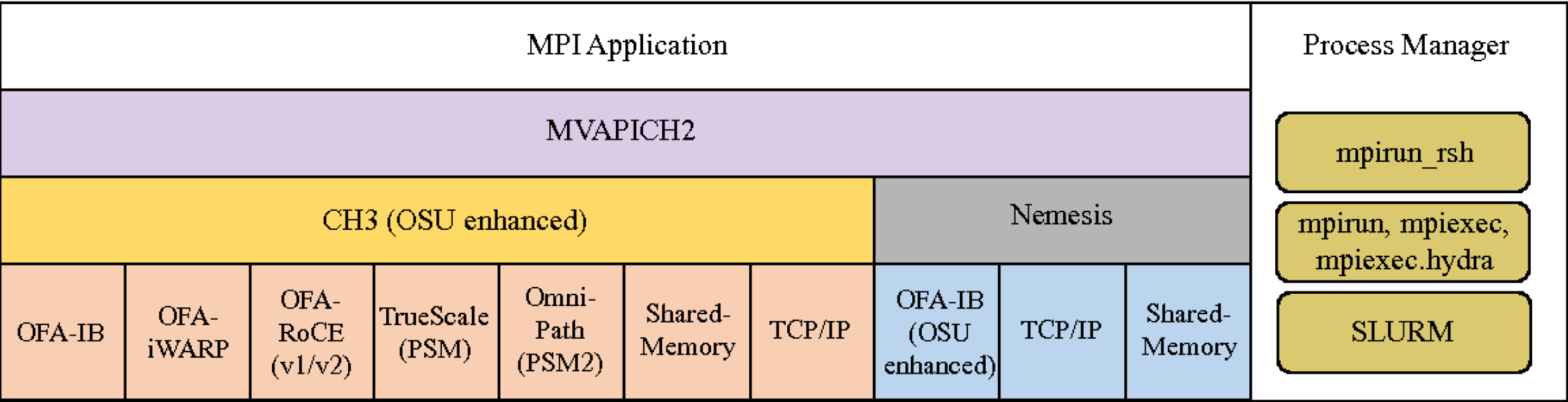
- Research is done for exploring new designs
- Designs are first presented to conference/journal publications
- Best performing designs are incorporated into the codebase
- Rigorous Q&A procedure before making a release
  - Exhaustive unit testing
  - Various test procedures on diverse range of platforms and interconnects
  - Performance tuning
  - Applications-based evaluation
  - Evaluation on large-scale systems
- Even alpha and beta versions go through the above testing



# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Coda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBLue
- Conclusions and Final Q&A

# MVAPICH2 Interfaces (Latest Release 2.2rc1)



All Different PCI interfaces

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

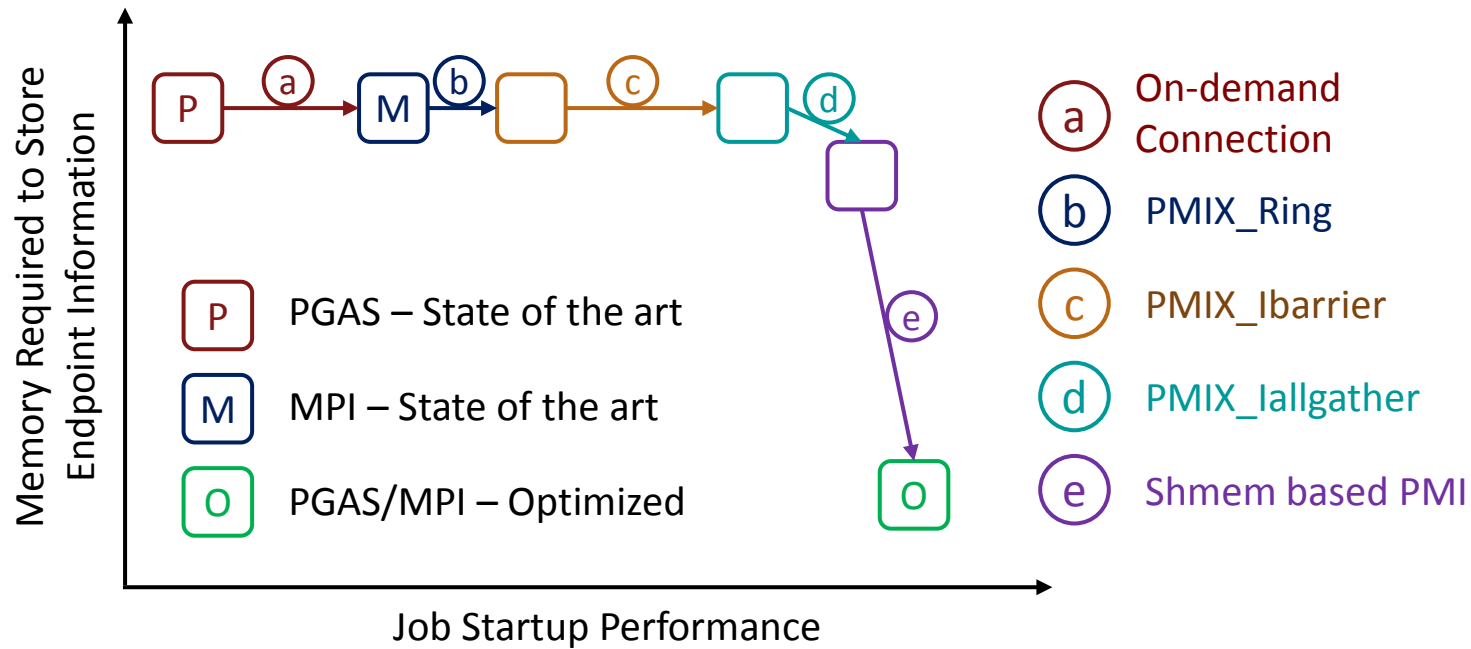
# Features of MVAPICH2 2.2rc1

- Released on 03/30/2016
- Major Features and Enhancements
  - Based on MPICH-3.1.4
  - Support for OpenPower architecture
    - Optimized inter-node and intra-node communication
  - Support for Intel Omni-Path architecture
    - Thanks to Intel for contributing the patch
    - Introduction of a new PSM2 channel for Omni-Path
  - Support for RoCEv2
  - Architecture detection for PSC Bridges system with Omni-Path
  - Enhanced startup performance and reduced memory footprint for storing InfiniBand end-point information with SLURM
    - Support for shared memory based PMI operations
    - Availability of an updated patch from the MVAPICH project website with this support for SLURM installations
  - Optimized pt-to-pt and collective tuning for Chameleon InfiniBand systems at TACC/UoC
  - Enable affinity by default for TrueScale(PSM) and Omni-Path(PSM2) channels
  - Enhanced tuning for shared-memory based MPI\_Bcast
  - Enhanced debugging support and error messages
  - Update to hwloc version 1.11.2

# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Conclusions and Final Q&A

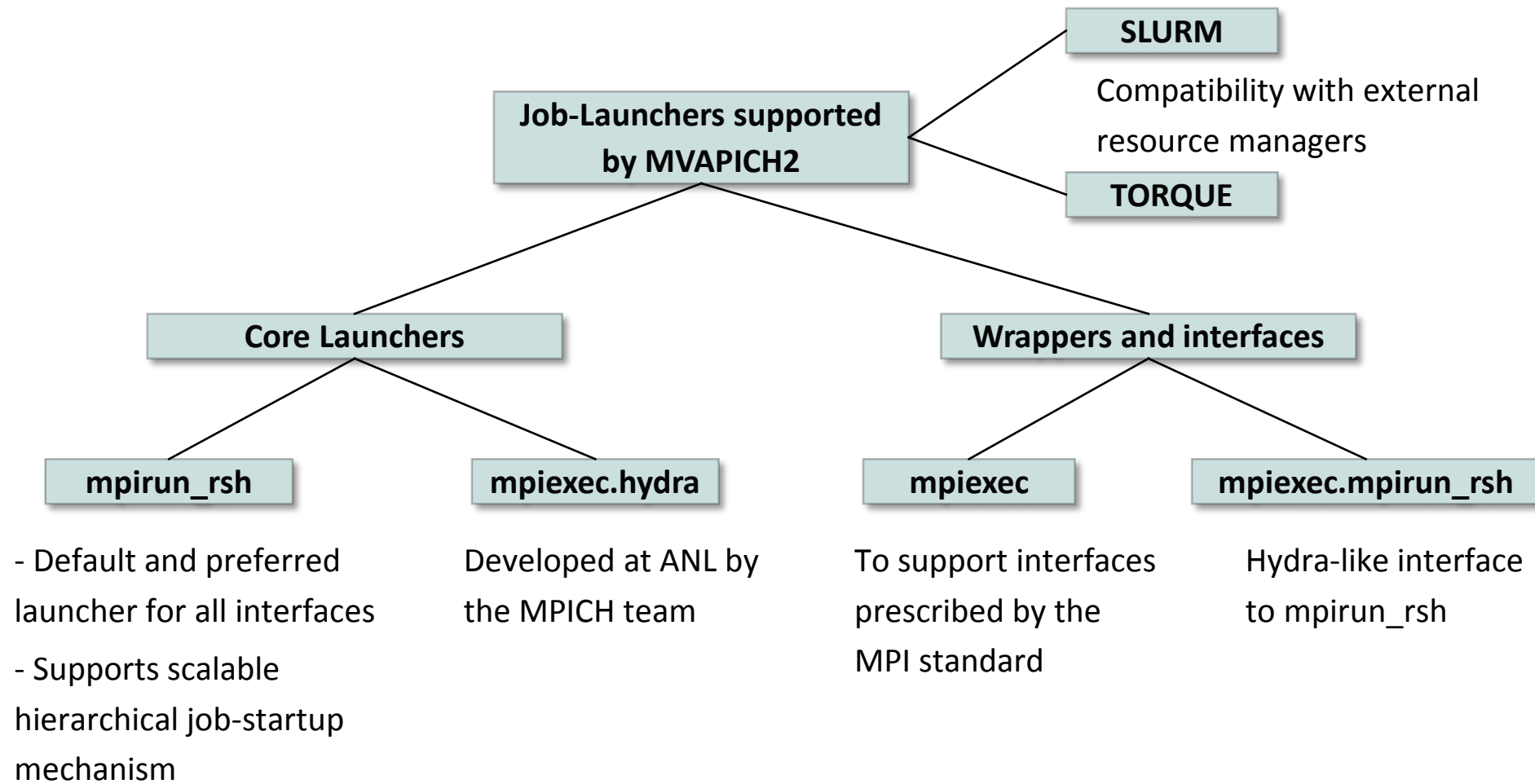
# Towards High Performance and Scalable Startup at Exascale



- 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  $O(\text{processes per node})$
- 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*

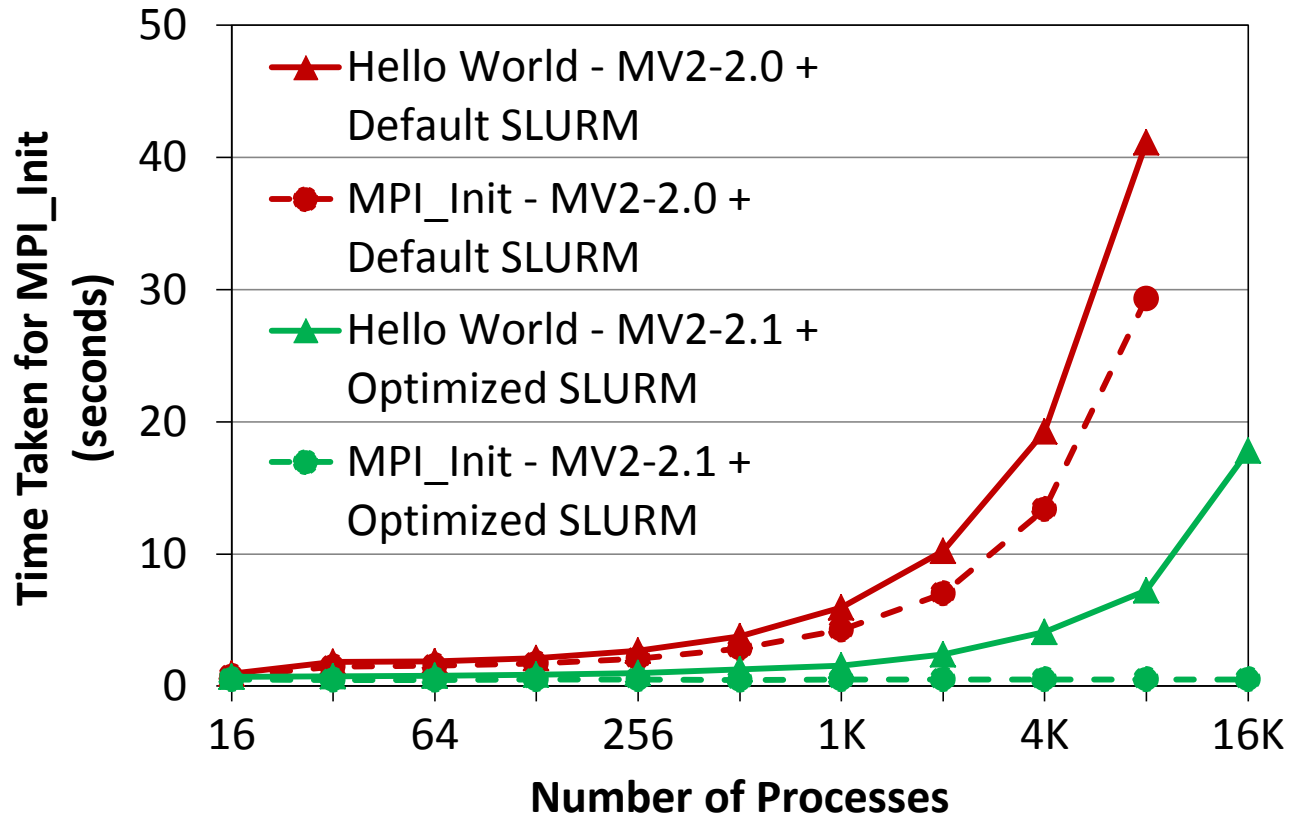
# Job-Launchers supported by MVAPICH2



# Tuning Job-Launch with mpirun\_rsh

- 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
- MV2\_HOMOGENEOUS\_CLUSTER
  - Setting it optimizes startup for homogeneous clusters
- MV2\_ON\_DEMAND\_UD\_INFO\_EXCHANGE
  - To optimize start-up by exchanging UD connection info on-demand

# Performance of MPI\_Init and Hello World at Large Scale on TACC Stampede



- Near-constant MPI\_Init at any scale
- PMI Exchange costs overlapped with application computation
- 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
- New designs show good scaling with 16K processes and above
- Available since MVAPICH2-2.1 and as patch for SLURM

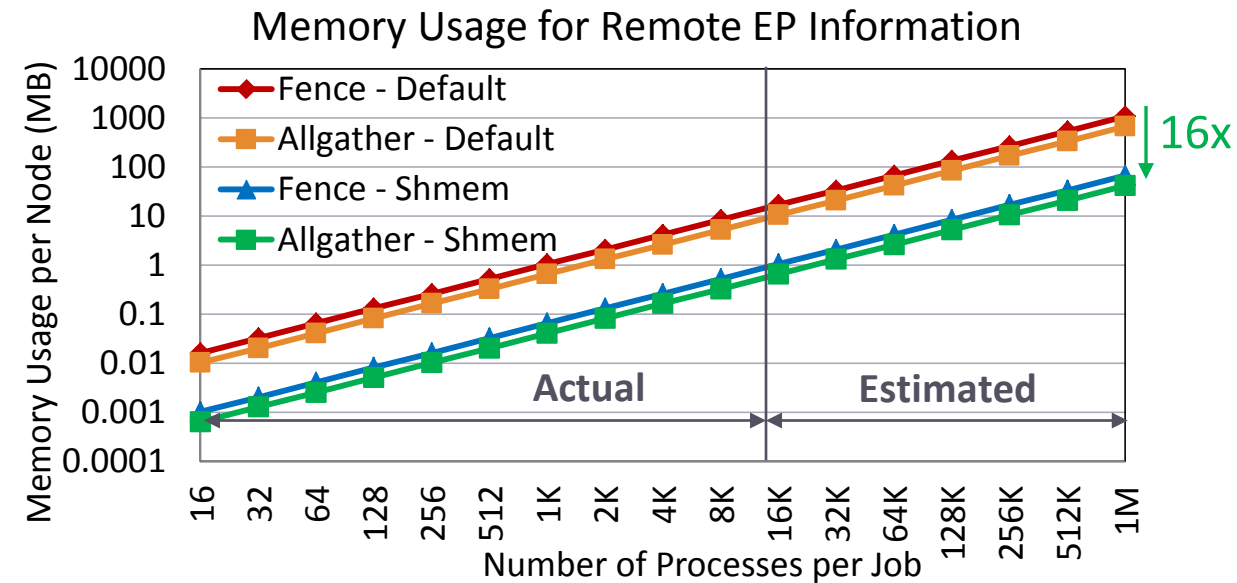
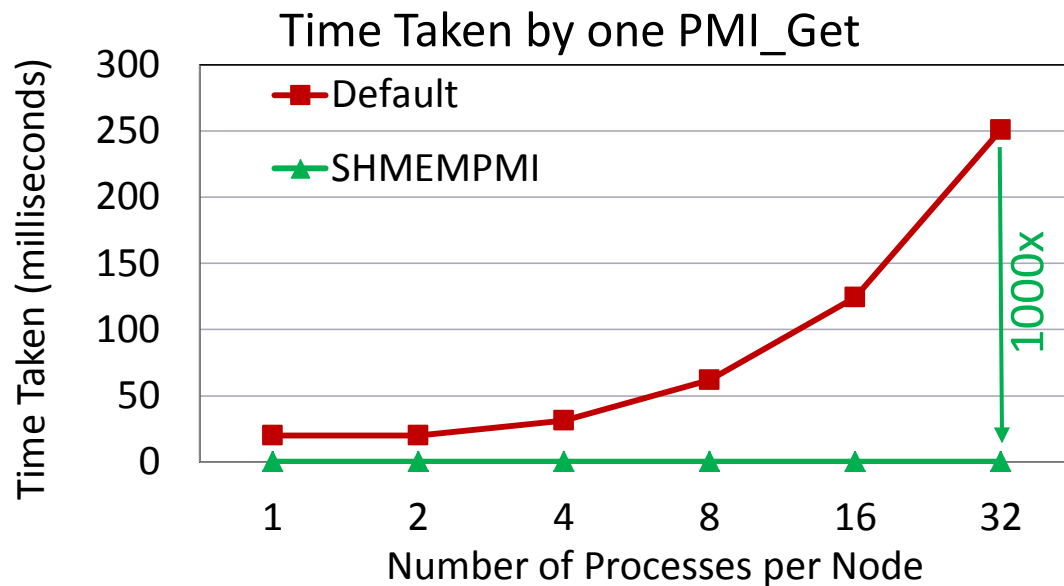
TACC Stampede - Connect-IB (54 Gbps): 2.6 GHz Quad Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR

**“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)



# Process Management Interface 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 -  $O(\text{processes per node})$  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

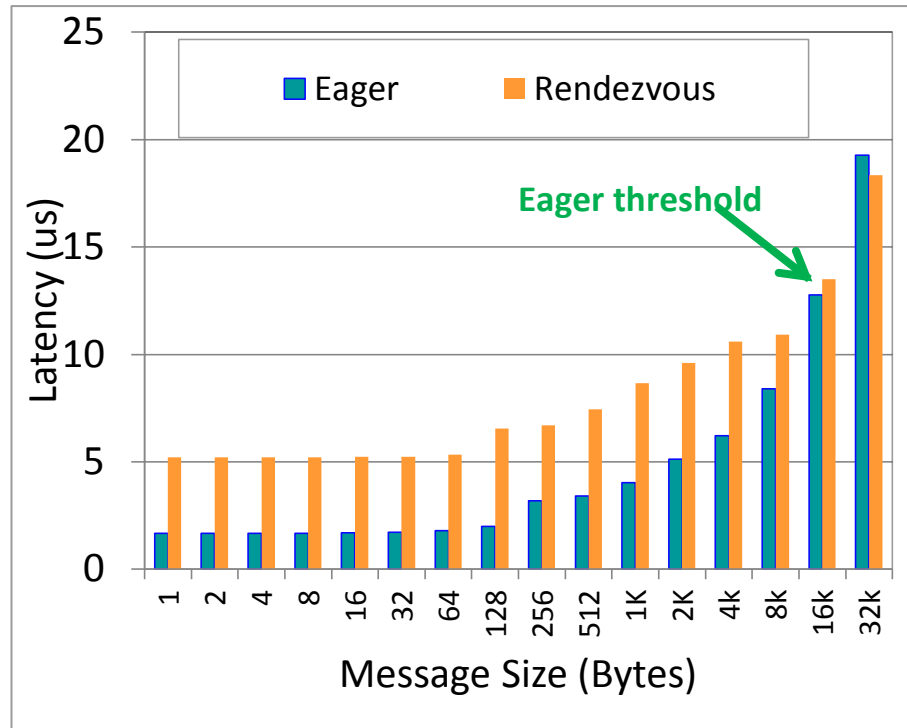


“SHMEMPMI – Shared Memory Based PMI for Performance and Scalability” S. Chakraborty, H. Subramoni, J. Perkins, and D.K. Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '16), Accepted for publication

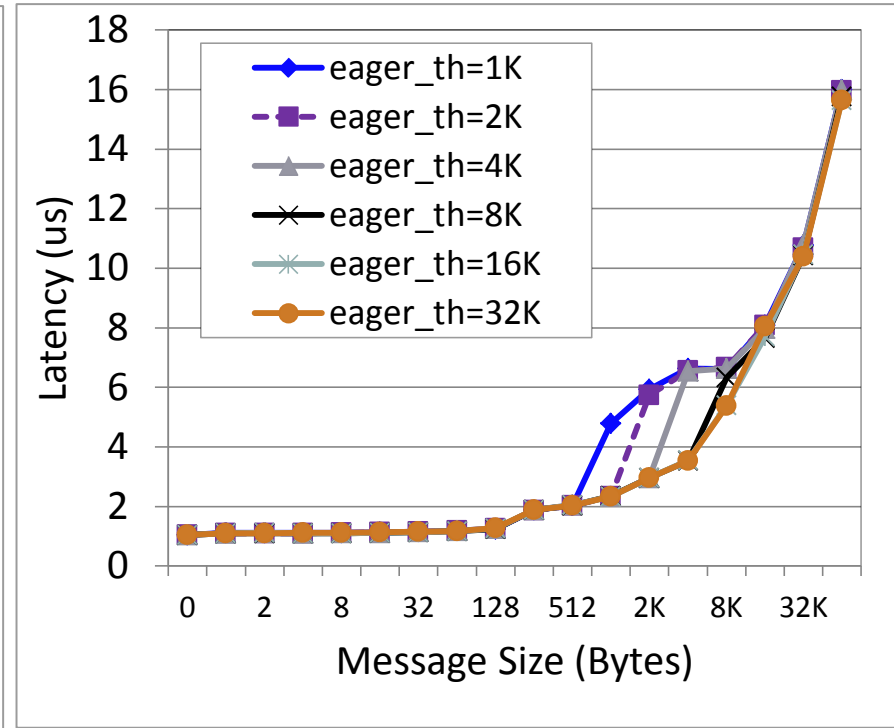
# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Conclusions and Final Q&A

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



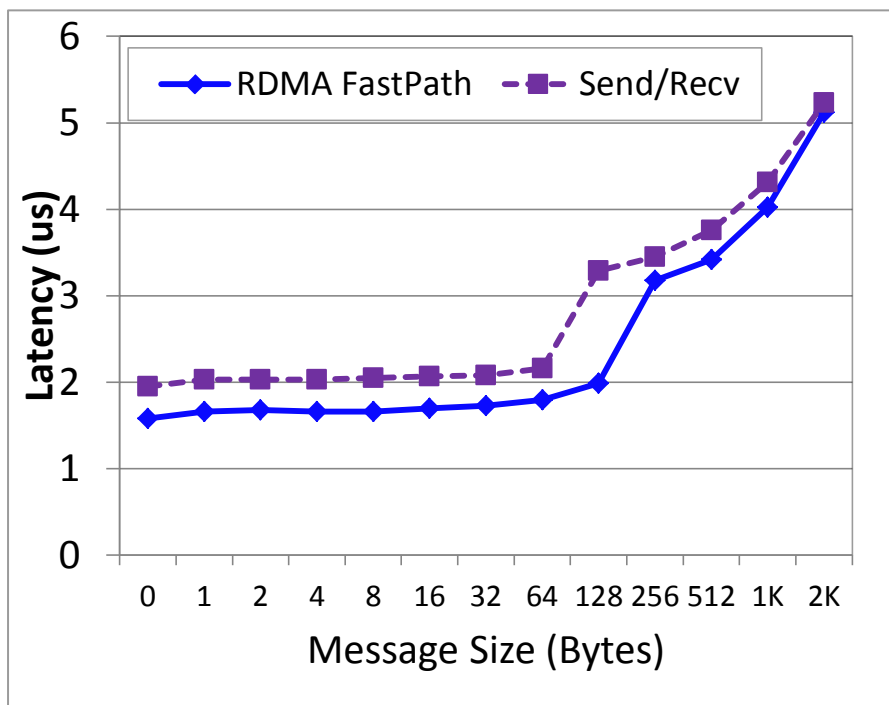
Eager vs Rendezvous



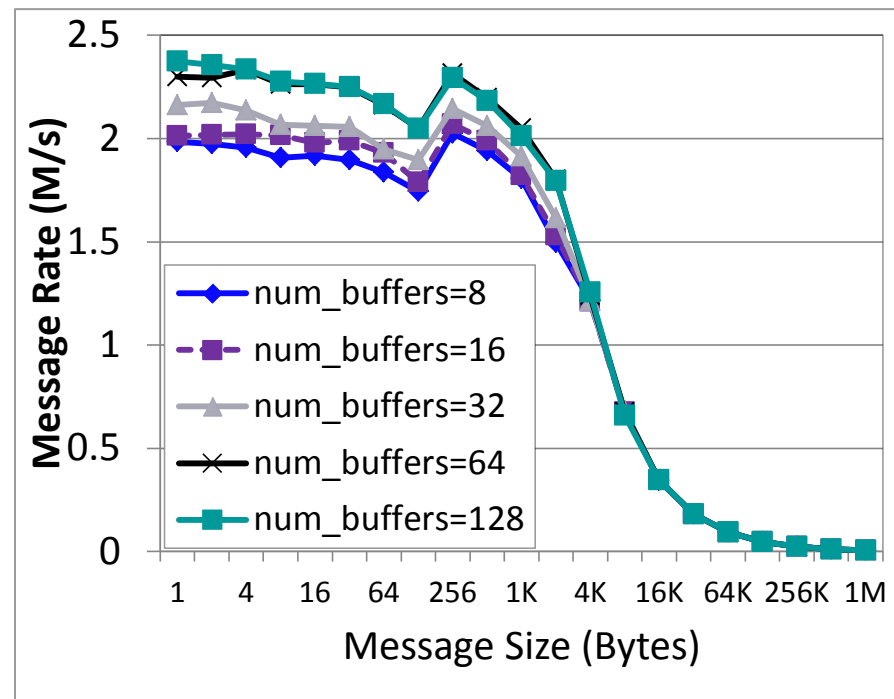
Impact of Eager Threshold

- 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 -f 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



Eager: Send/Recv vs RDMA FP



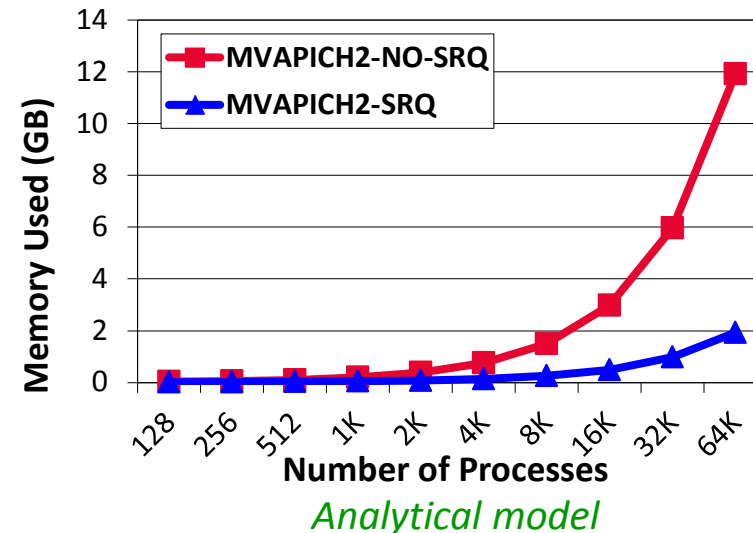
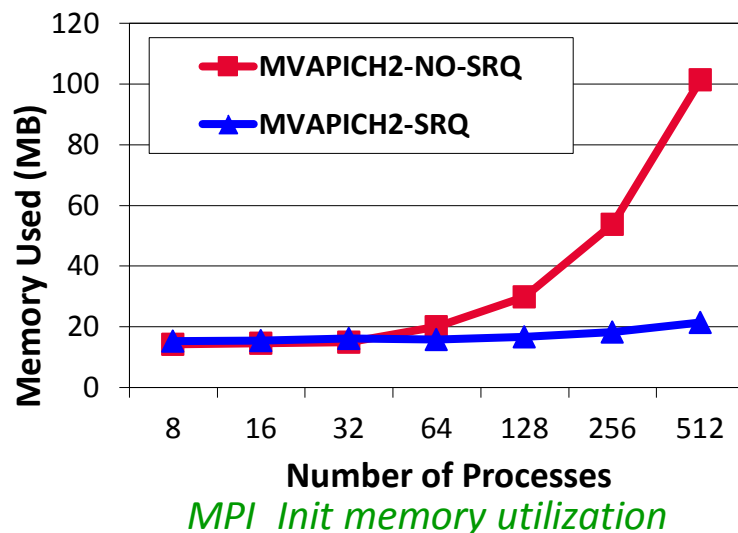
Impact of RDMA FP buffers

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

# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBLue
- Conclusions and Final Q&A

# Using Shared Receive Queues with MVAPICH2

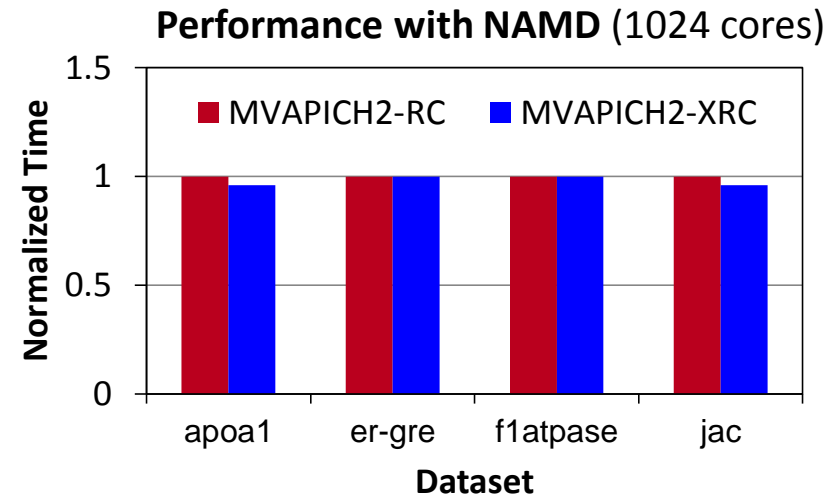
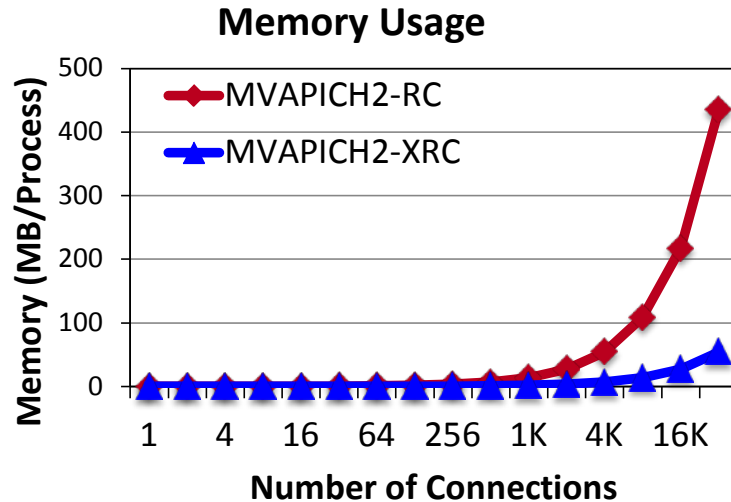


- SRQ reduces the memory used **by 1/6<sup>th</sup>** at 64,000 processes

Parameter	Significance	Default	Notes
MV2_USE_SRQ	<ul style="list-style-type: none"> <li>Enable / Disable use of SRQ in MVAPICH2</li> </ul>	Enabled	<ul style="list-style-type: none"> <li>Always Enable</li> </ul>
MV2_SRQ_MAX_SIZE	<ul style="list-style-type: none"> <li>Limits the maximum size of the SRQ</li> <li>Places upper bound on amount of memory used for SRQ</li> </ul>	4096	<ul style="list-style-type: none"> <li>Increase to 8192 for large scale runs</li> </ul>
MV2_SRQ_SIZE	<ul style="list-style-type: none"> <li>Number of buffers posted to the SRQ</li> <li>Automatically doubled by MVAPICH2 on receiving SRQ LIMIT EVENT from IB HCA</li> </ul>	256	<ul style="list-style-type: none"> <li>Upper Bound: MV2_SRQ_MAX_SIZE</li> </ul>

- 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



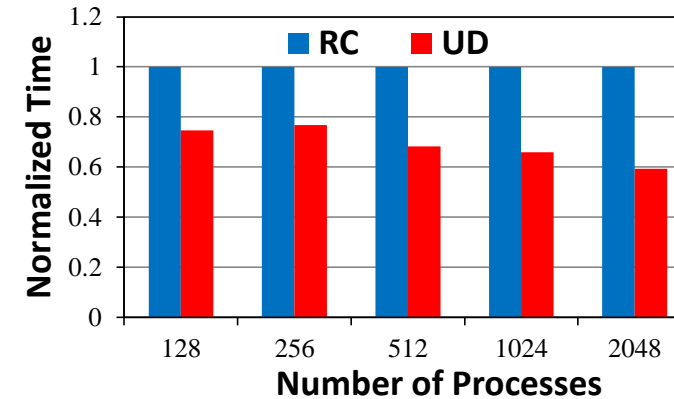
- Memory usage for 32K processes with 8-cores per node can be **54 MB/process** (for connections)
- NAMD performance improves when there is frequent communication to many peers
- Enabled by setting **MV2\_USE\_XRC** to 1 (Default: Disabled)
- Requires OFED version > 1.3
  - Unsupported in earlier versions (< 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



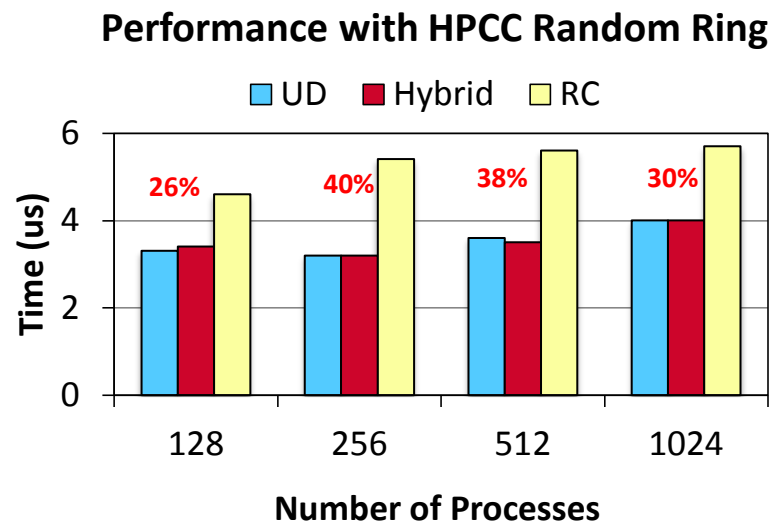
- Can use UD transport by configuring MVAPICH2 with the **-enable-hybrid**
  - Reduces QP cache trashing and memory footprint at large scale

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

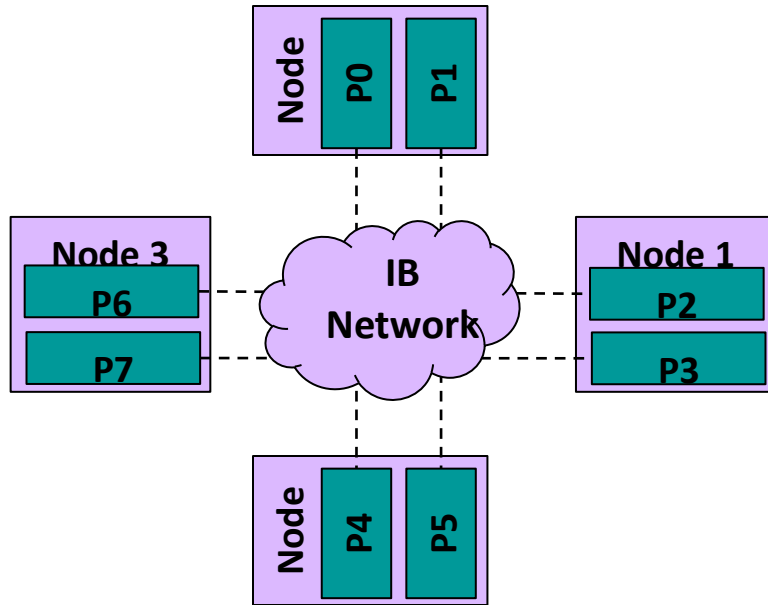


- Both UD and RC/XRC have benefits
  - Hybrid for the best of both
- Enabled by configuring MVAPICH2 with the `-enable-hybrid`
- Available since MVAPICH2 1.7 as integrated interface

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

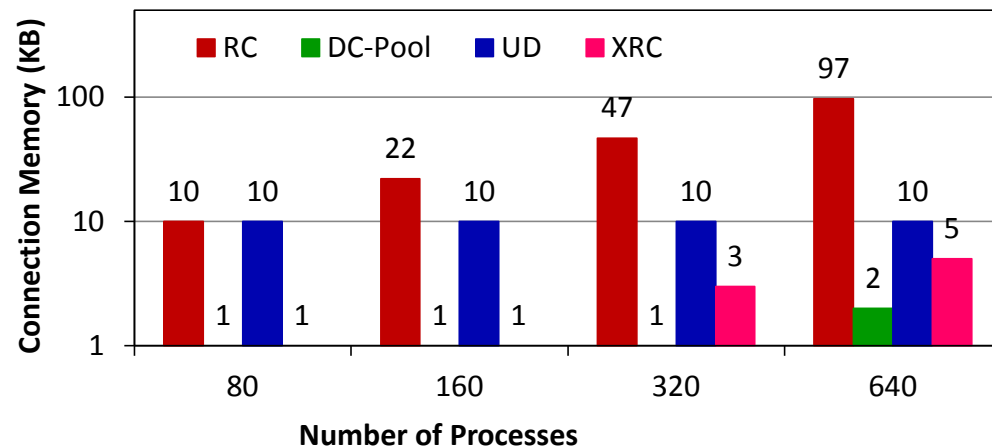
- Refer to **Running with Hybrid UD-RC/XRC** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-650006.11>

# Minimizing Memory Footprint by Direct Connect (DC) Transport

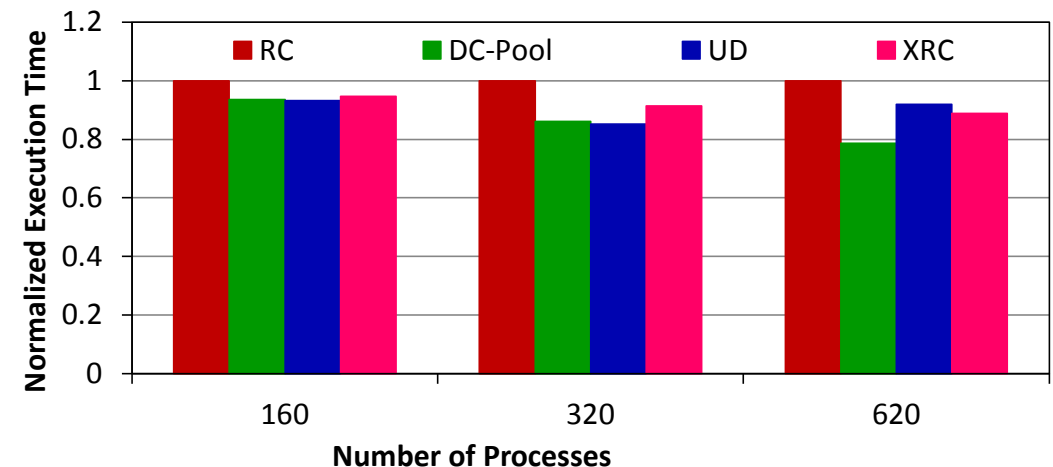


- Constant connection cost (*One QP for any peer*)
- Full Feature Set (RDMA, Atomics etc)
- Separate objects for send (DC Initiator) and receive (DC Target)
  - DC Target identified by “DCT Number”
  - Messages routed with (DCT Number, LID)
  - Requires same “DC Key” to enable communication
- Available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

# Presentation Overview

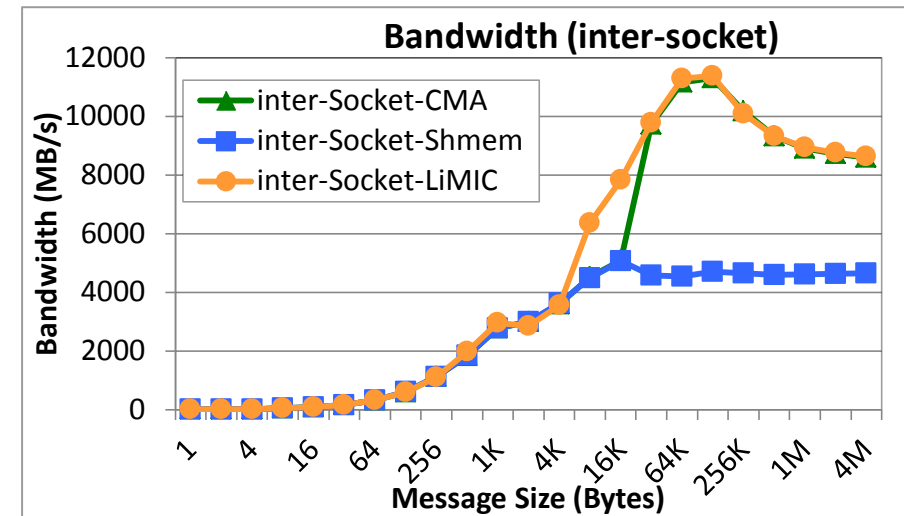
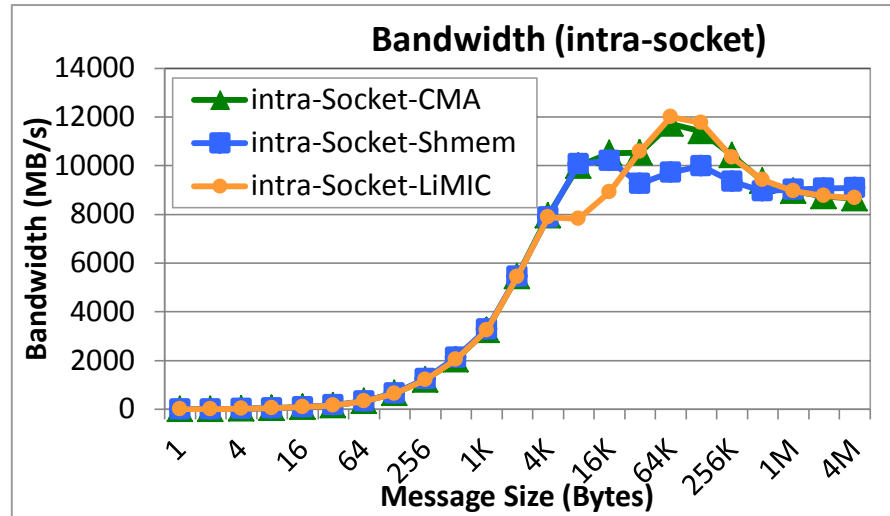
- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- Conclusions and Final Q&A

## 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

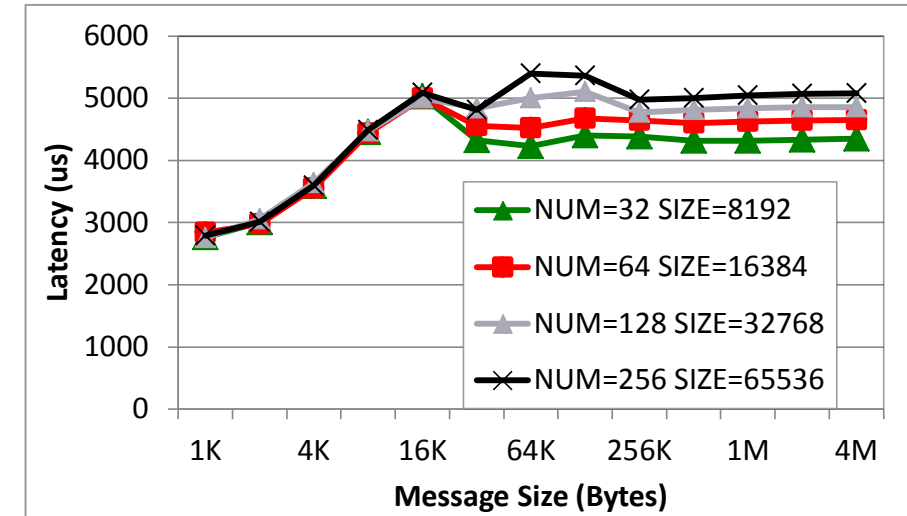
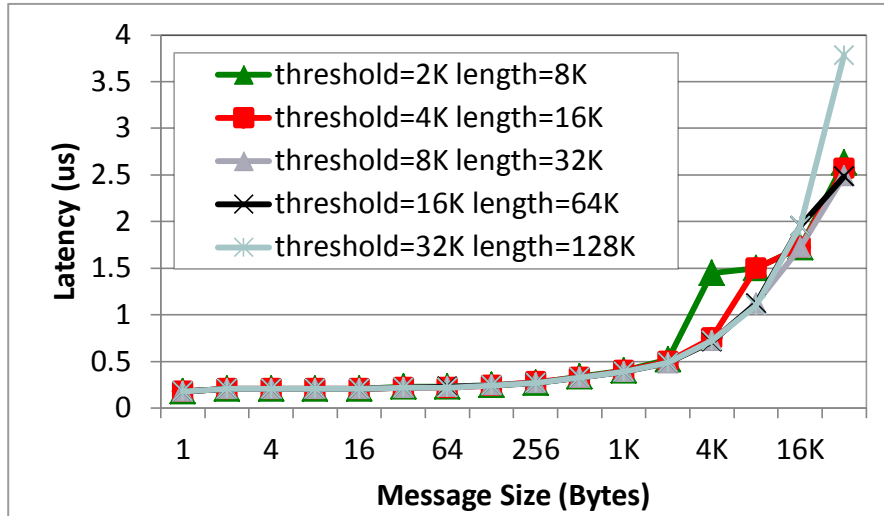
# MVAPICH2 Two-Sided Intra-Node Tuning:

## Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA)



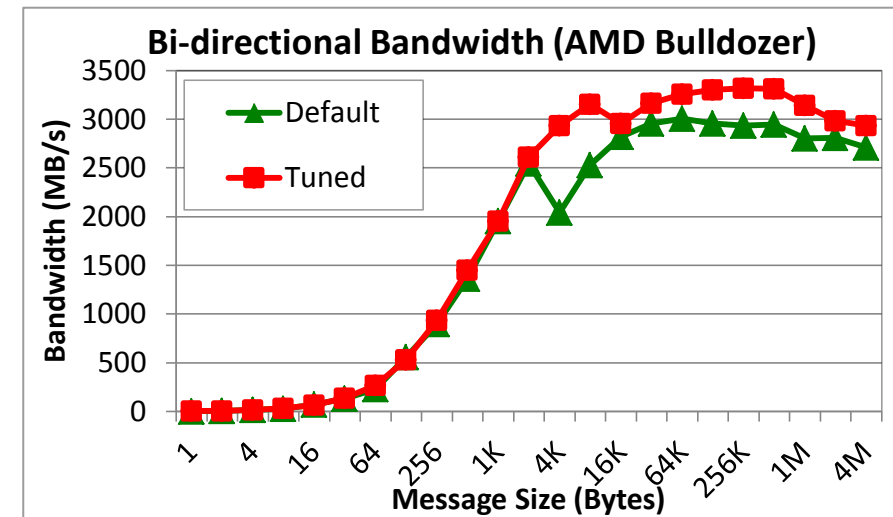
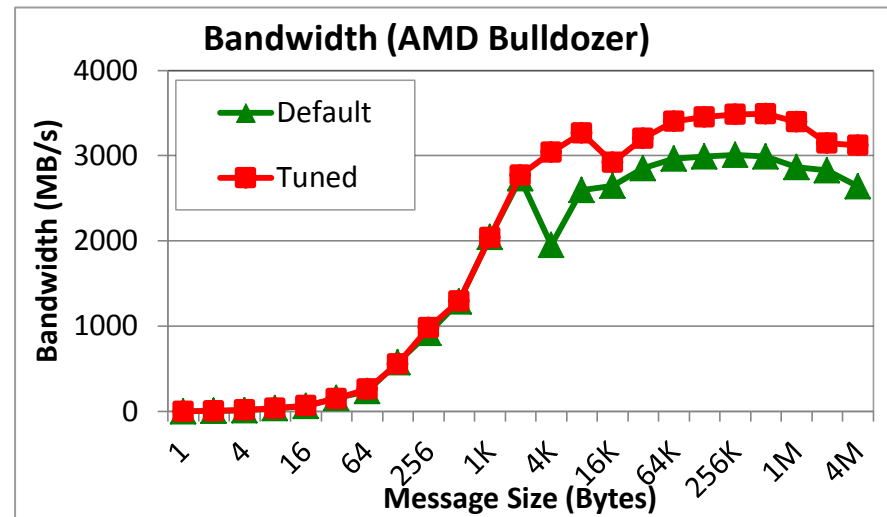
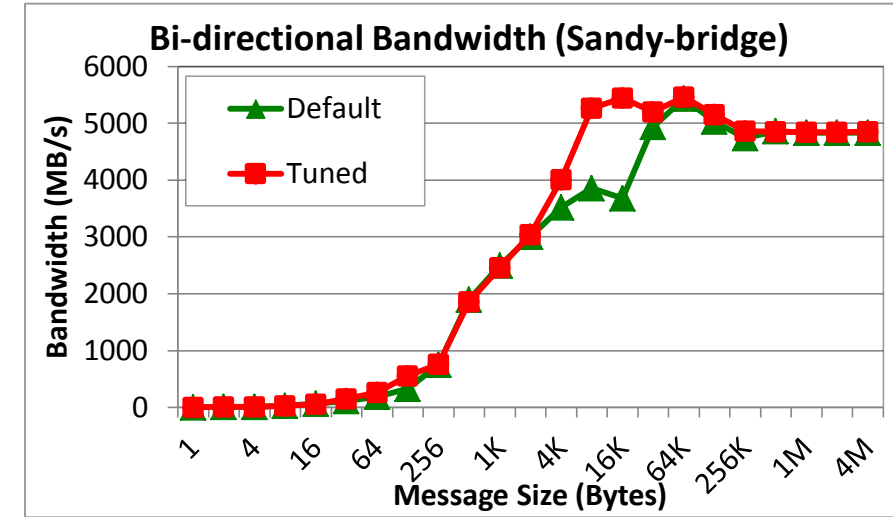
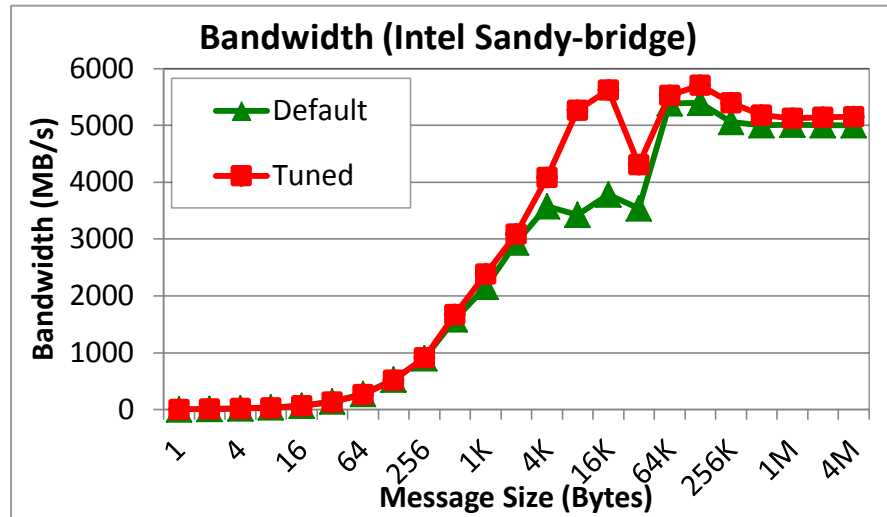
- LiMIC2:
  - configure the library with '--with-limic2'
  - mpirun\_rsh -np 2 -f hostfile a.out (To disable: MV2\_SMP\_USE\_LIMIC2=0)
- CMA:
  - configure the library with '--with-cma'
  - mpirun\_rsh -np 2 -f hostfile a.out (To disable: MV2\_SMP\_USE\_CMA=0)
- When both '--with-limic2' and '--with-cma' 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 -f 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 -f 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



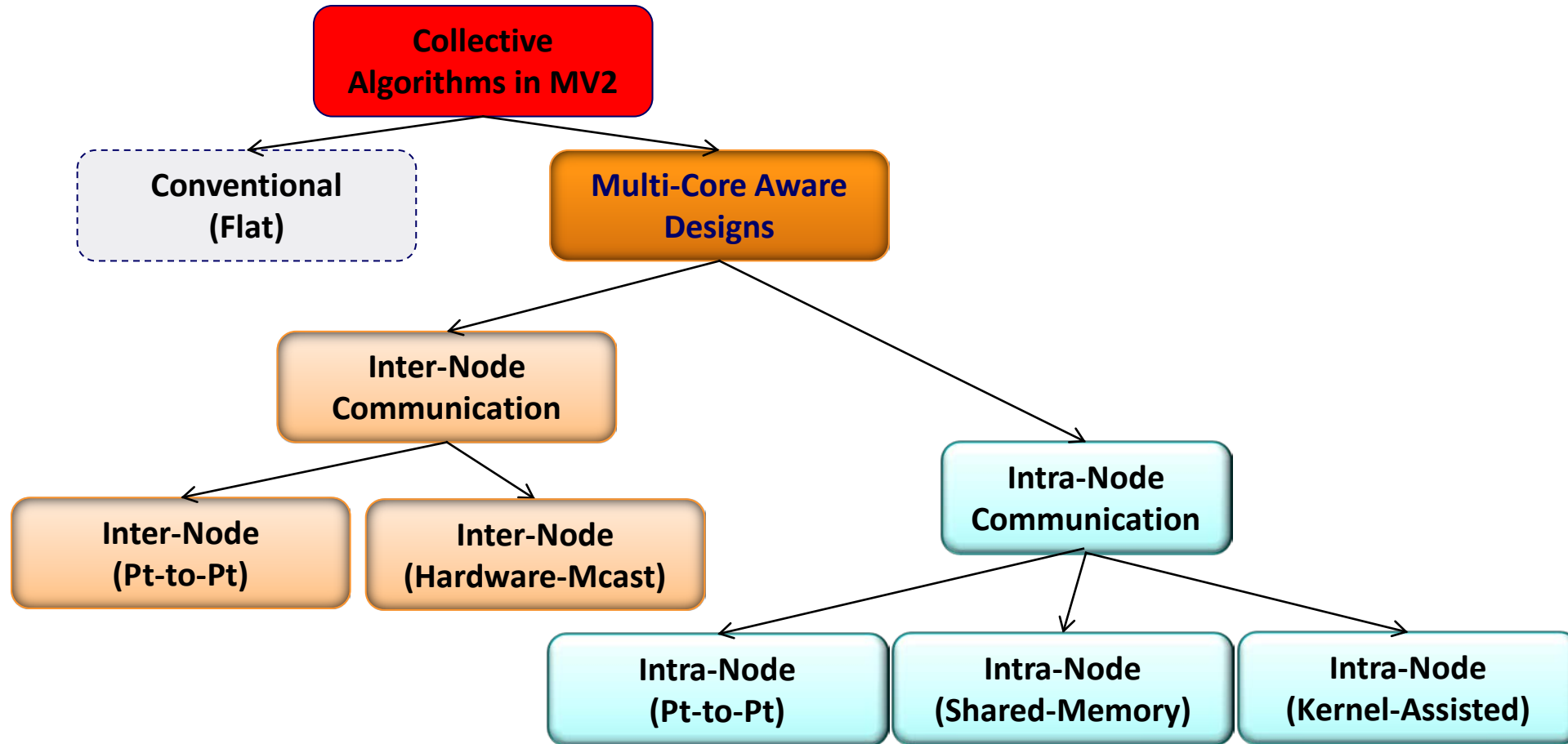
- Architecture-specific tuning is executed for new architectures and new designs introduced into MVAPICH2
- **MV2\_SMP\_EAGERSIZE** and **MV2\_SMP\_SEND\_BUFF\_SIZE** are updated from Default to Tuned

# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBLue
- Conclusions and Final Q&A



# Collective Communication in MVAPICH2

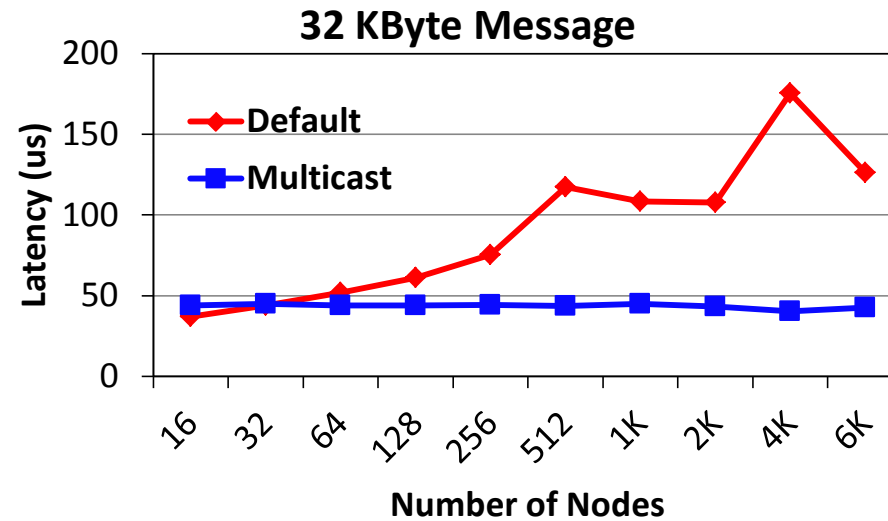
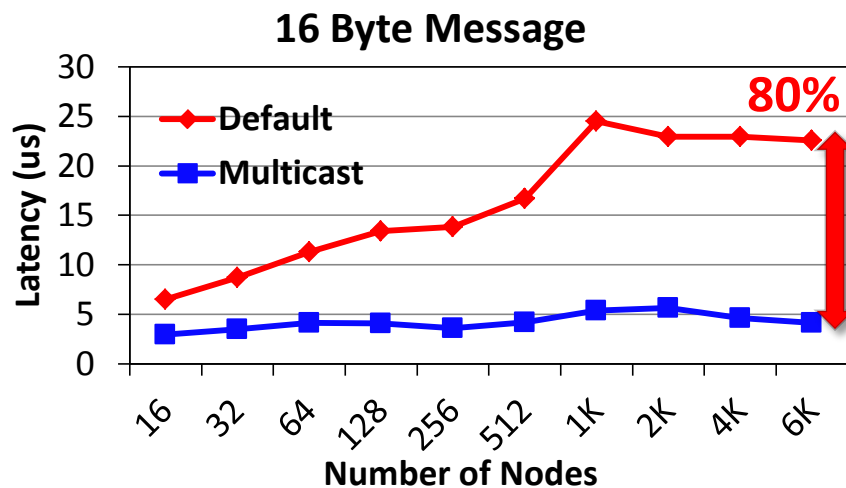
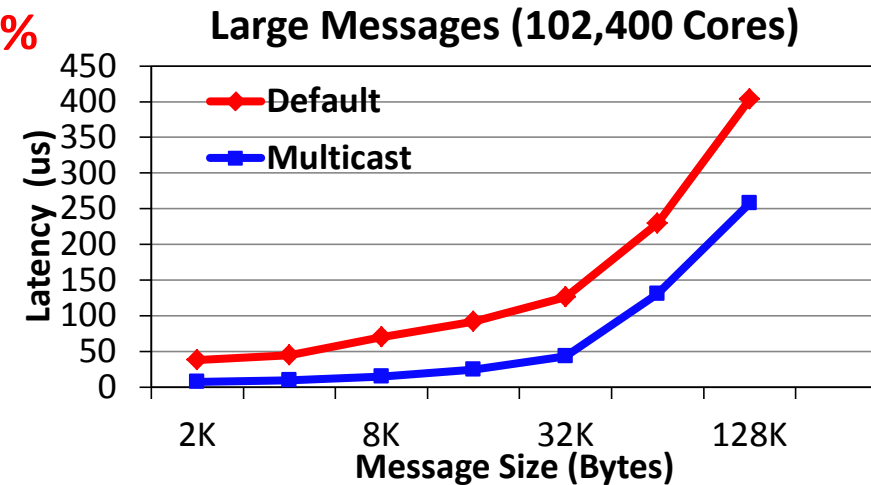
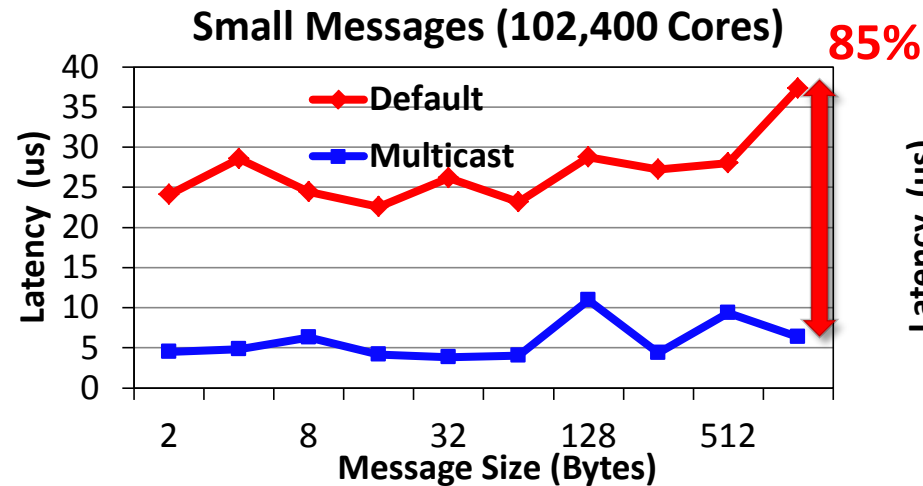


Run-time flags:

All shared-memory based collectives: MV2\_USE\_SHMEM\_COLL (Default: ON)

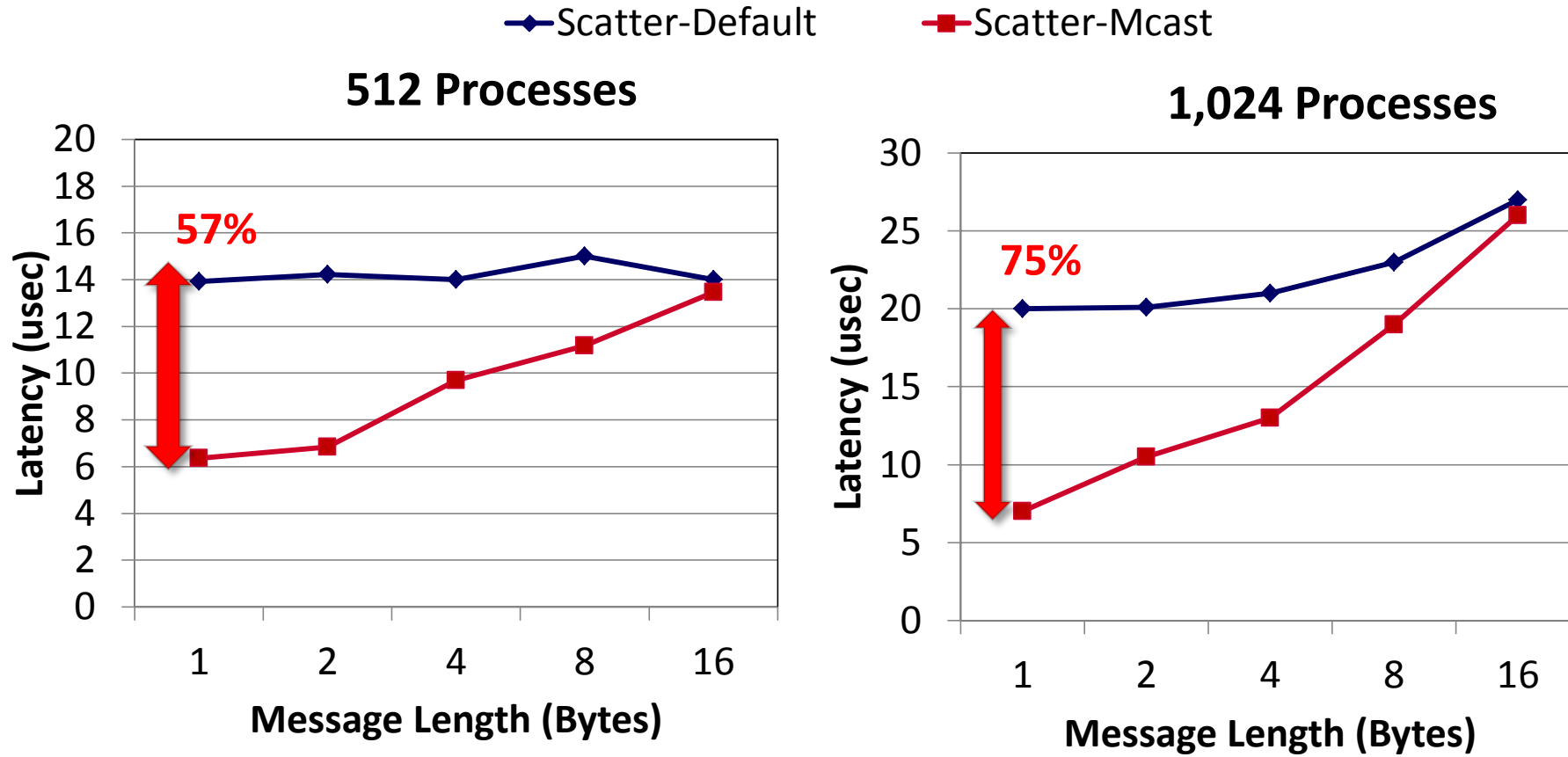
Hardware Mcast-based collectives: MV2\_USE\_MCAST (Default : OFF)

# 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

## Enabling Hardware Multicast-aware

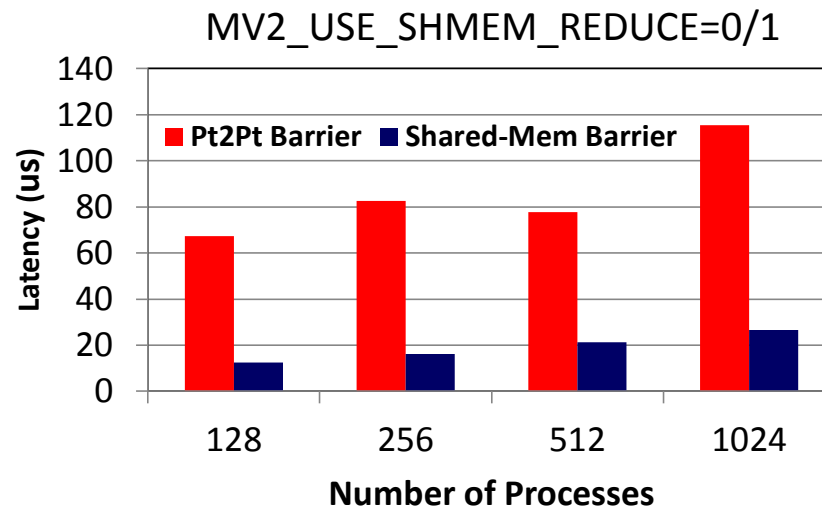
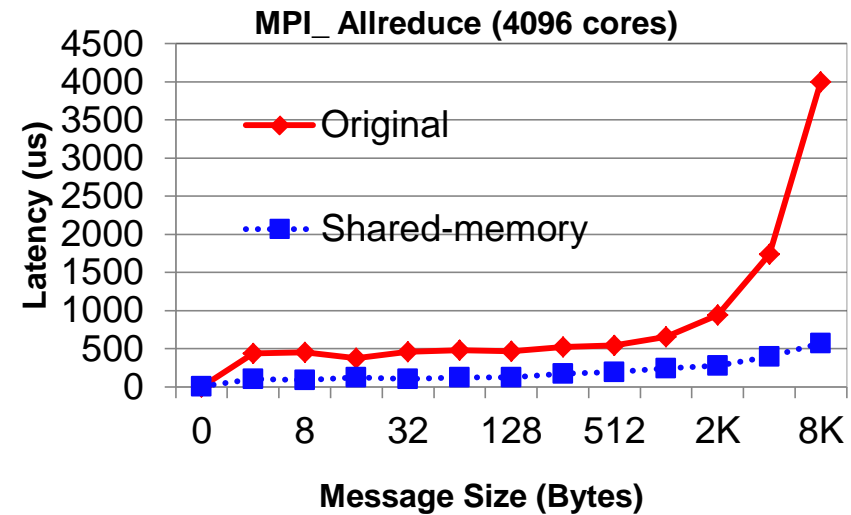
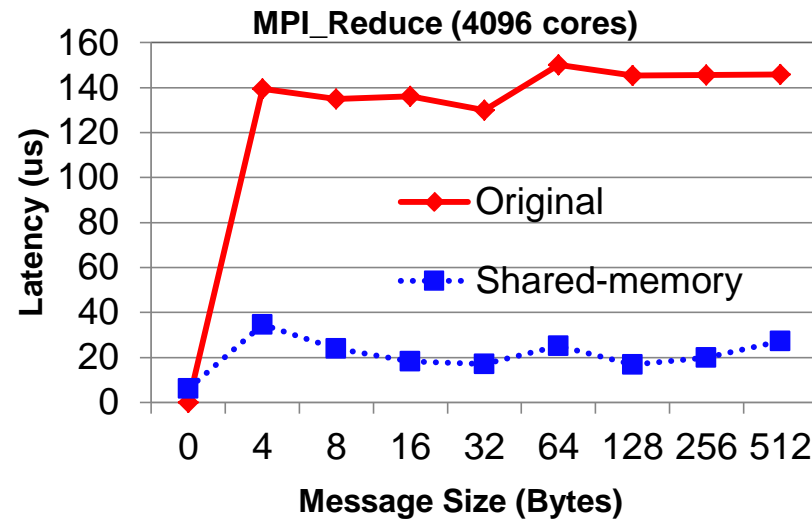
- Multicast is applicable to
  - MPI\_Bcast
  - MPI\_Scatter
  - MPI\_Allreduce

Parameter	Description	Default Nature
MV2_USE_MCAST = 1	Enables hardware Multicast features	Disabled
--enable-mcast	Configure flag to enable	Enabled

- Refer to **Running Collectives with Hardware based Multicast support** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2rc1-userguide.html#x1-620006.8>

# Shared-memory Aware Collectives

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



MV2\_USE\_SHMEM\_ALLREDUCE=0/1

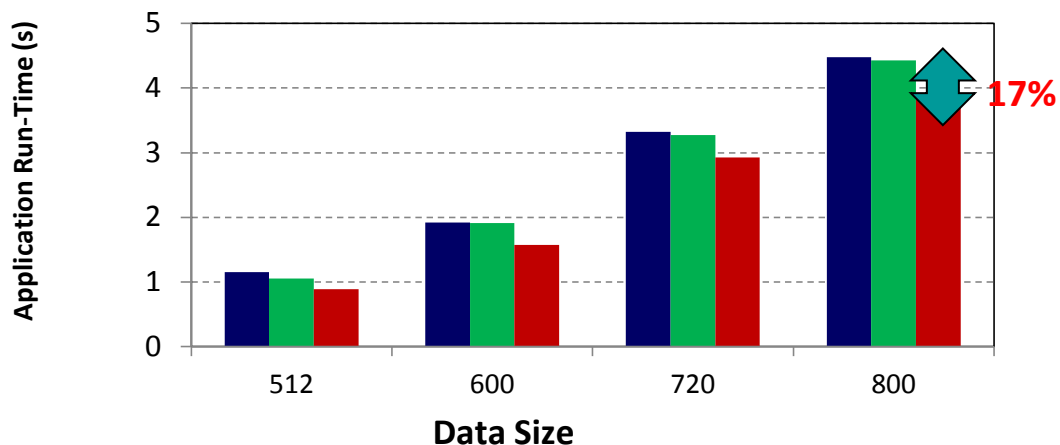
- MVAPICH2 Barrier with 1K Intel Westmere cores, QDR IB

MV2\_USE\_SHMEM\_BARRIER=0/1

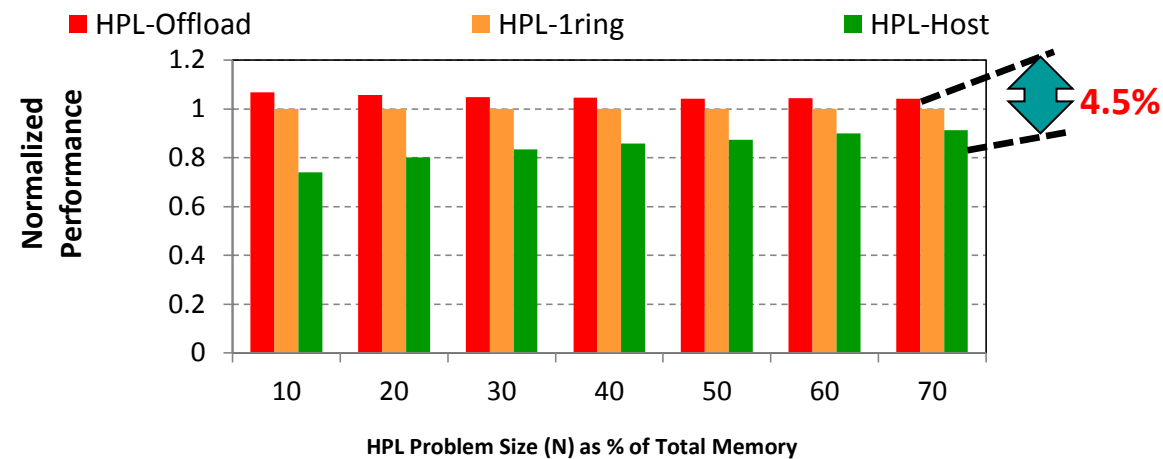
## Non-Blocking Collectives in MVAPICH2

- MVAPICH2 supports for MPI-3 Non-Blocking Collective communication and Neighborhood collective communication primitives
  - MVAPICH2 1.9 to MVAPICH2 2.2
- MPI-3 collectives in MVAPICH2 can use either the Gen2 or the nemesis interfaces, over InfiniBand
- MVAPICH2 implements non-blocking collectives either in a multi-core-aware hierarchical manner, or via a basic flat approach
- Application developers can use MPI-3 collectives to achieve computation/communication overlap

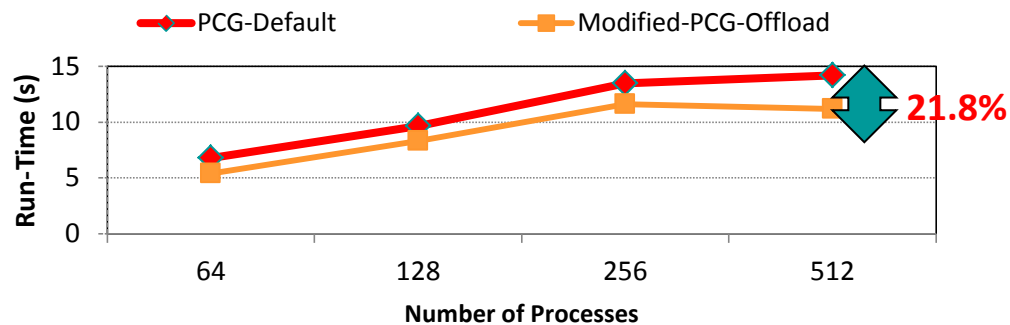
# Co-Design with MPI-3 Non-Blocking Collectives and Collective Offload Co-Direct Hardware (Available since MVAPICH2-X 2.2a)



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



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



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

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

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

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

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

# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- 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 ( $\geq$  CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

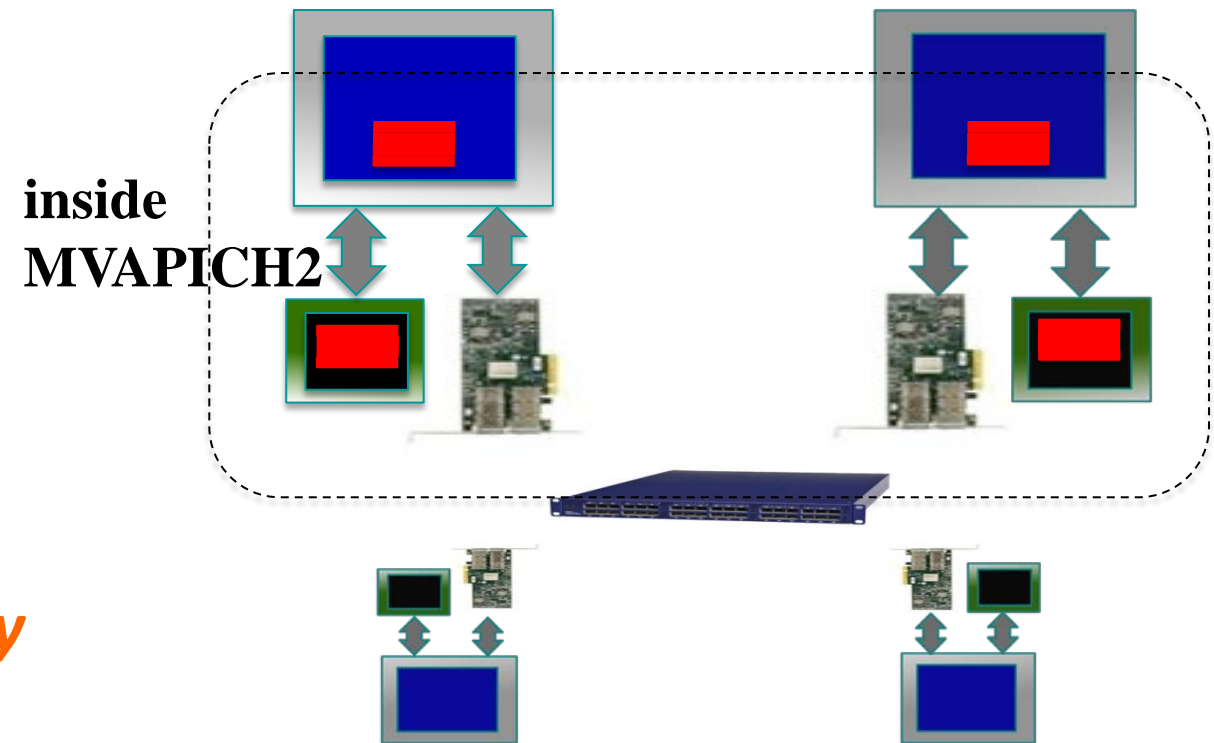
## At Sender:

```
MPI_Send(s_devbuf, size, ...);
```

## At Receiver:

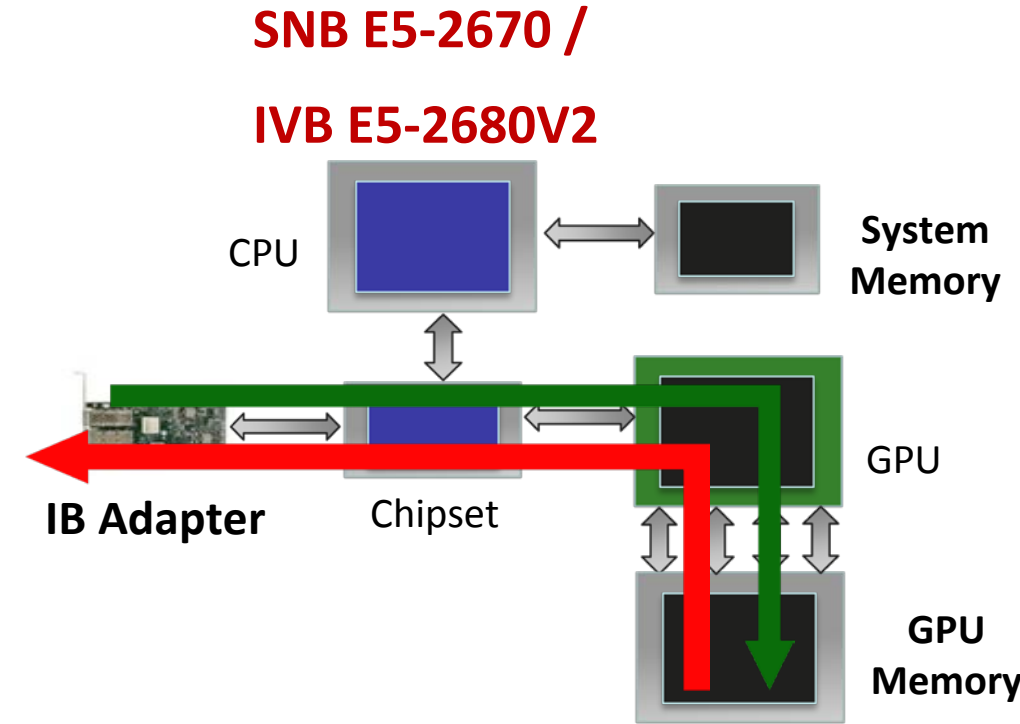
```
MPI_Recv(r_devbuf, size, ...);
```

*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**

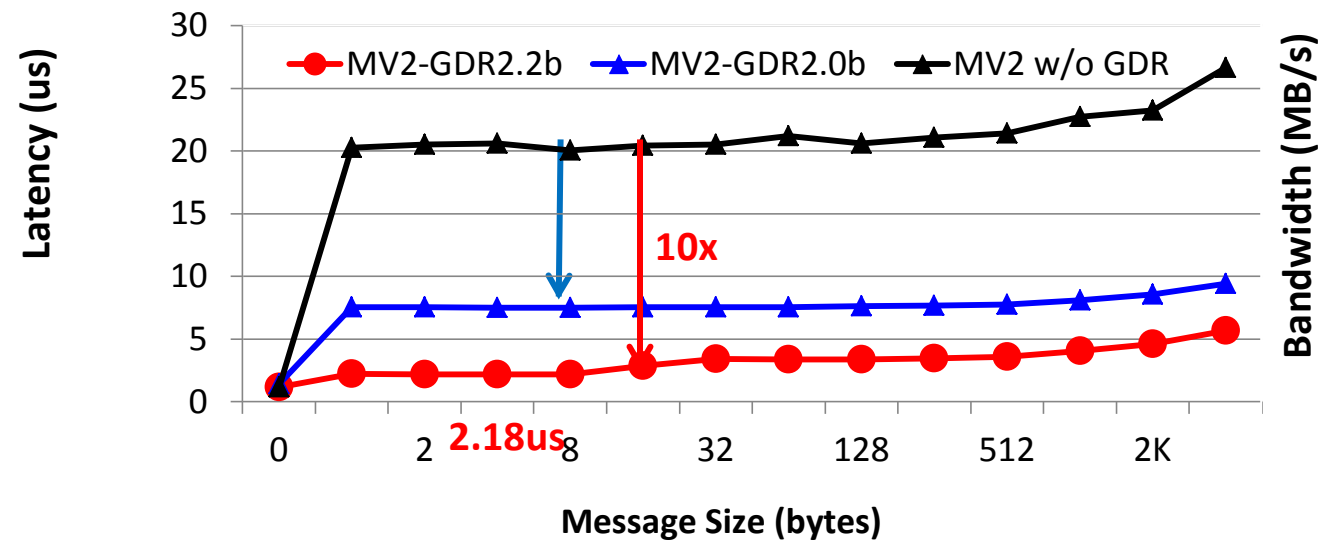
**IVB E5-2680V2**  
**P2P write: 6.4 GB/s**  
**P2P read: 3.5 GB/s**

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

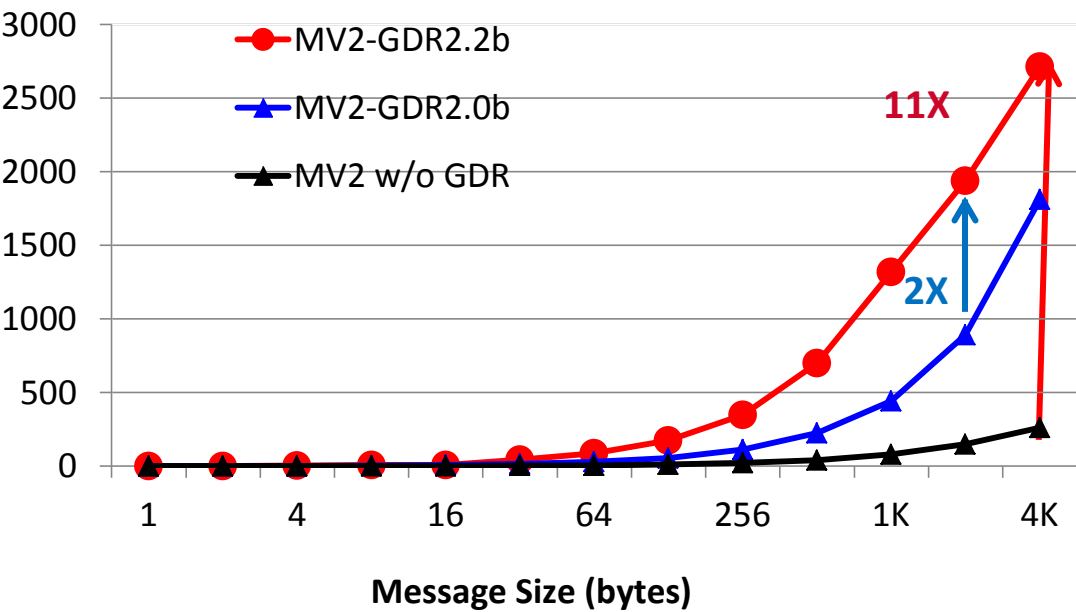
- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage 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

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

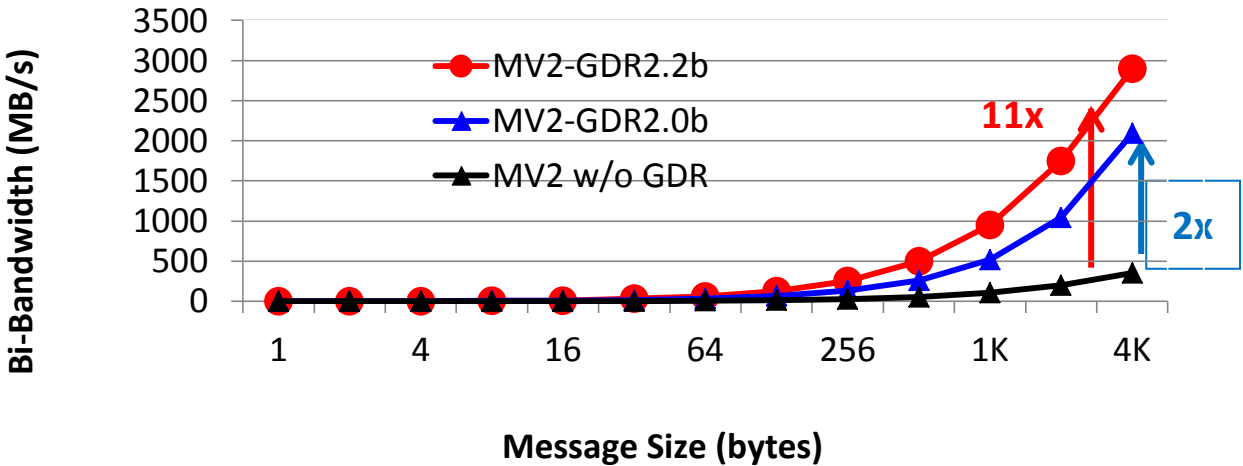
GPU-GPU internode latency



GPU-GPU Internode Bandwidth



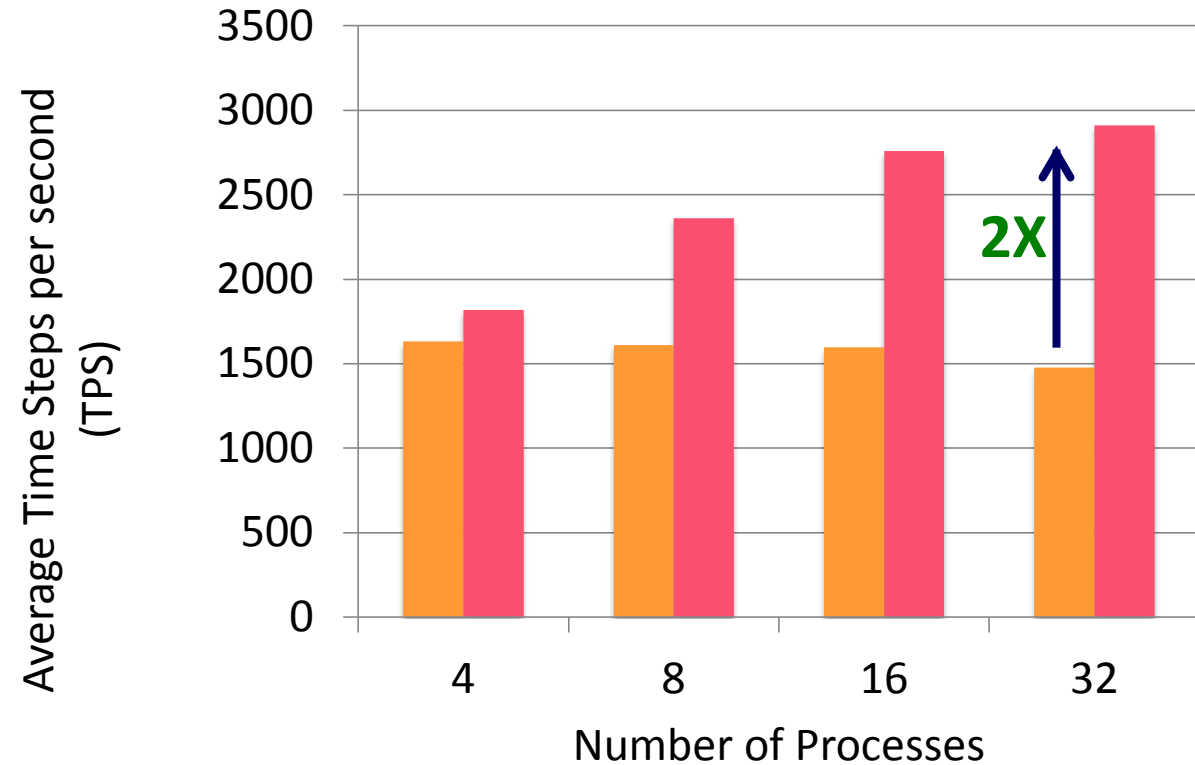
GPU-GPU Internode Bi-Bandwidth



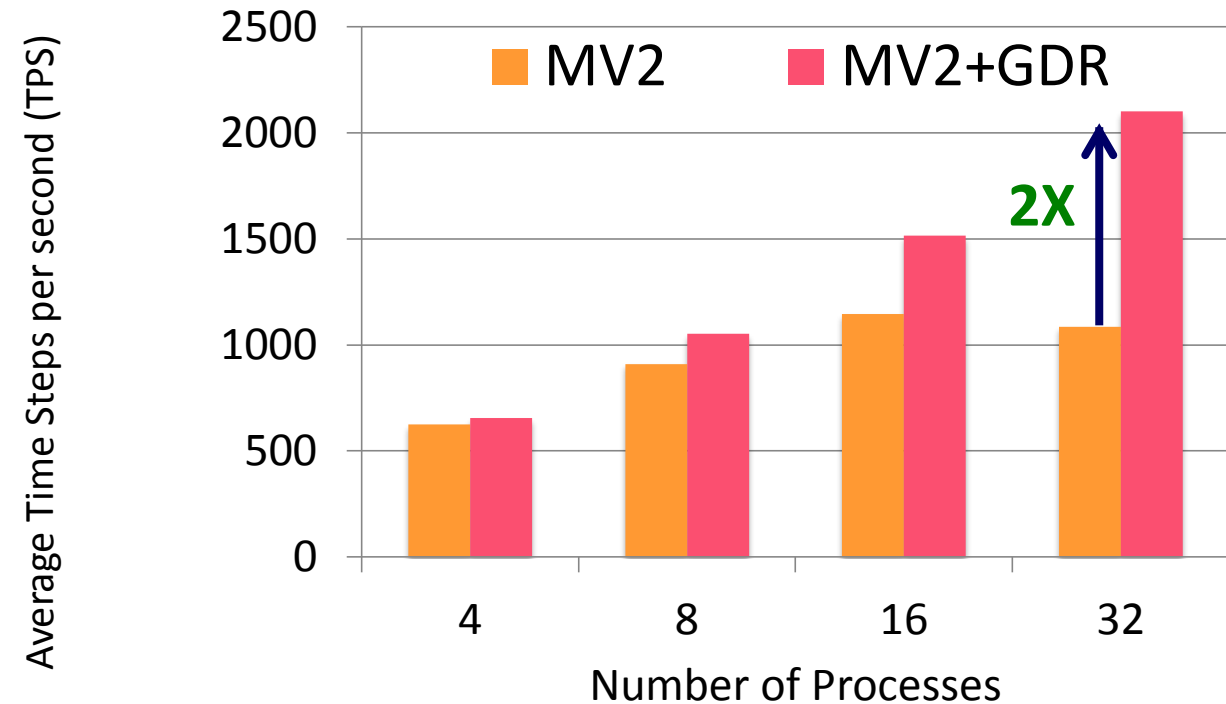
**MVAPICH2-GDR-2.2b**  
**Intel Ivy Bridge (E5-2680 v2) node - 20 cores**  
**NVIDIA Tesla K40c GPU**  
**Mellanox Connect-IB Dual-FDR HCA**  
**CUDA 7**  
**Mellanox OFED 2.4 with GPU-Direct-RDMA**

# Application-Level Evaluation (HOOMD-blue)

## 64K Particles

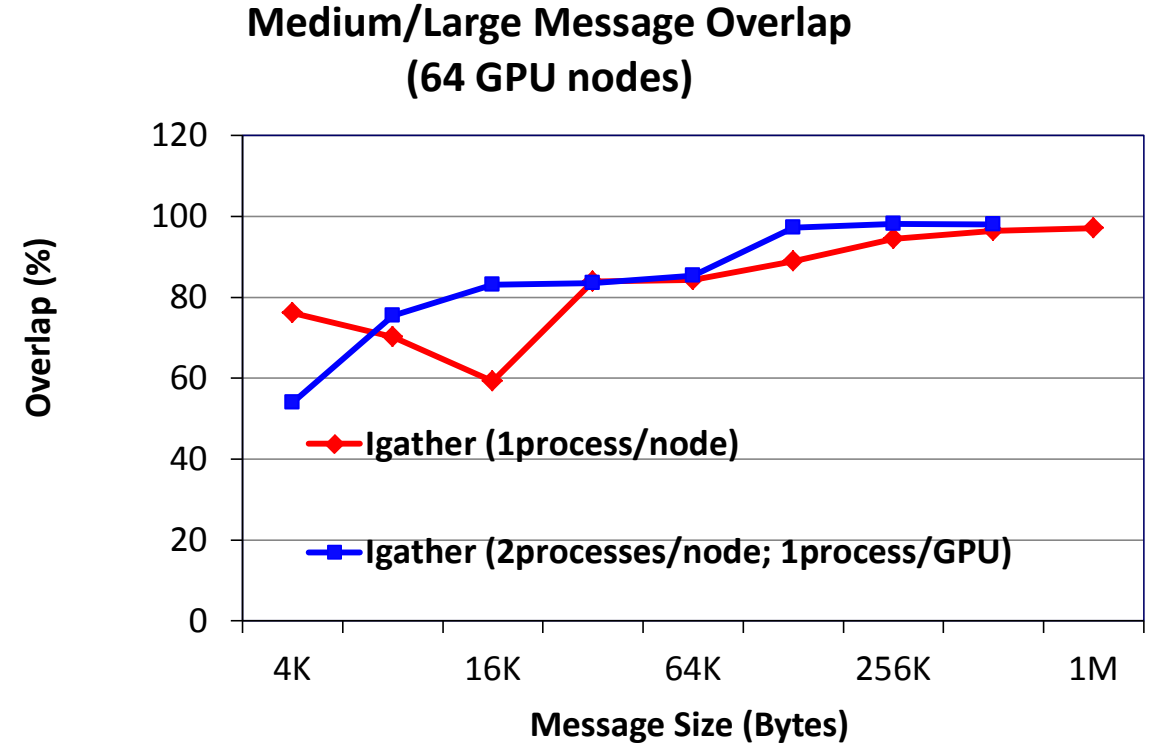
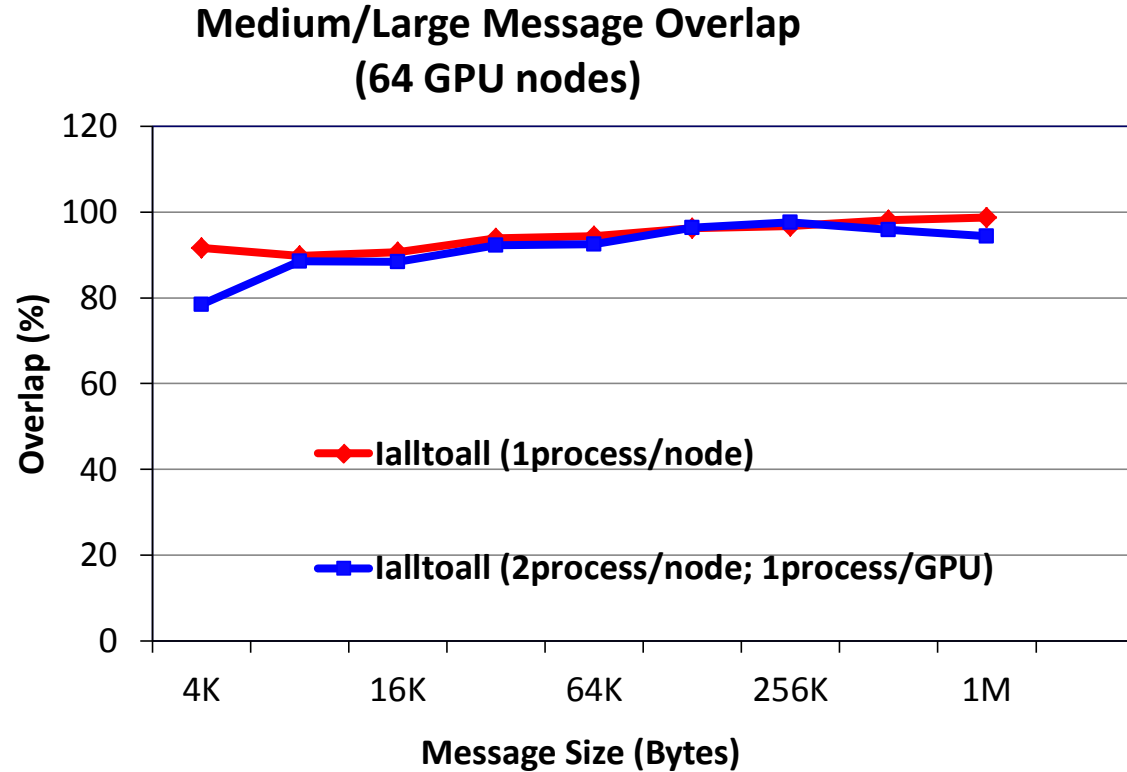


## 256K Particles



- 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

# CUDA-Aware Non-Blocking Collectives



A. Venkatesh, K. Hamidouche, H. Subramoni, and D. K. Panda, Offloaded GPU Collectives using CORE-Direct and CUDA Capabilities on IB Clusters, HIPC, 2015

Platform: Wilkes: Intel Ivy Bridge  
NVIDIA Tesla K20c + Mellanox Connect-IB  
Available since MVAPICH2-GDR 2.2a

## Using MVAPICH2-GDR Version

- MVAPICH2-GDR 2.2b with GDR support can be downloaded from <https://mvapich.cse.ohio-state.edu/download/#mv2gdr/>
- System software requirements
  - Mellanox OFED 2.1 or later
  - NVIDIA Driver 331.20 or later
  - NVIDIA CUDA Toolkit 6.0 or later
  - Plugin for GPUDirect RDMA
    - [http://www.mellanox.com/page/products\\_dyn?product\\_family=116](http://www.mellanox.com/page/products_dyn?product_family=116)
    - **Strongly Recommended** : use the new GDRCOPY module from NVIDIA
      - <https://github.com/NVIDIA/gdrcopy>
- Has optimized designs for point-to-point communication using GDR
- Can also be run without GDR (have better performance benefits compared to MVAPICH2)
- Contact MVAPICH help list with any questions related to the package  
[mvapich-help@cse.ohio-state.edu](mailto:mvapich-help@cse.ohio-state.edu)

# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBLue
- Conclusions and Final Q&A



# Pipelined Data Movement in MVAPICH2-GDR: Tuning

Parameter	Significance	Default	Notes
MV2_USE_CUDA	<ul style="list-style-type: none"><li>• Enable / Disable GPU designs</li></ul>	0 (Disabled)	<ul style="list-style-type: none"><li>• Disabled to avoid pointer checking overheads for host communication</li><li>• Always enable to support MPI communication from GPU Memory</li></ul>
MV2_CUDA_BLOCK_SIZE	<ul style="list-style-type: none"><li>• Controls the pipeline blocksize</li></ul>	256 KByte	<ul style="list-style-type: none"><li>• Tune for your system and application</li><li>• Varies based on<ul style="list-style-type: none"><li>- CPU Platform, IB HCA and GPU</li><li>- CUDA driver version</li><li>- Communication pattern (latency/bandwidth)</li></ul></li></ul>

- 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](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)

## Tuning GPUDirect RDMA (GDR) Designs in MVAPICH2-GDR

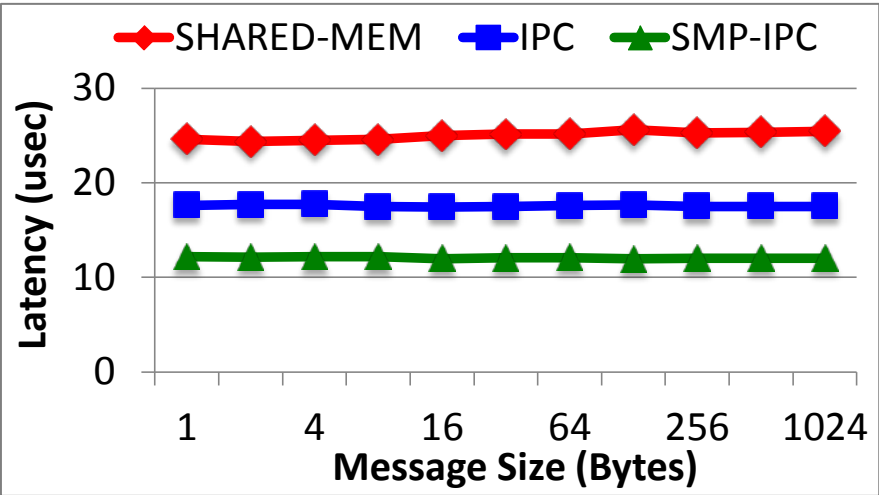
Parameter	Significance	Default	Notes
MV2_USE_GPUDIRECT	<ul style="list-style-type: none"><li>• Enable / Disable GDR-based designs</li></ul>	1 (Enabled)	<ul style="list-style-type: none"><li>• Always enable</li></ul>
MV2_GPUDIRECT_LIMIT	<ul style="list-style-type: none"><li>• Controls messages size until which GPUDirect RDMA is used</li></ul>	8 KByte	<ul style="list-style-type: none"><li>• Tune for your system</li><li>• GPU type, host architecture and CUDA version: impact pipelining overheads and P2P bandwidth bottlenecks</li></ul>

- 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](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)

# Runtime Parameters

- Works between GPUs within the same socket or IOH

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



Intranode osu\_latency small

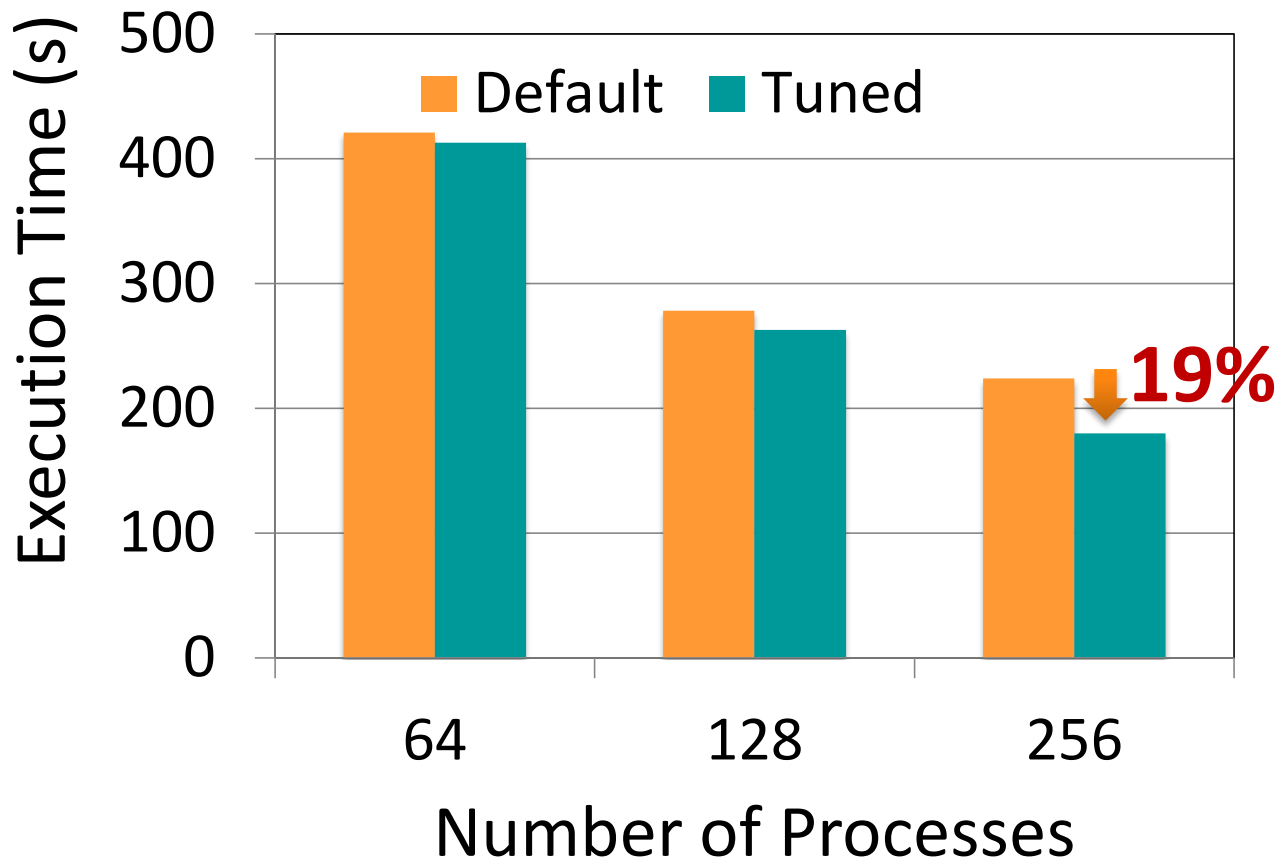
# Presentation Overview

- Runtime Optimization and Tuning Flexibility in
  - MVAPICH2 / MVAPICH2-X
    - Job start-up
    - Point-to-point Inter-node Protocol
    - Transport Type Selection
    - Point-to-point Intra-node Protocol and Scheme
    - Collectives
  - MVAPICH2-GDR
    - Cuda-aware MPI
    - GPUDirect RDMA (GDR) Features
    - Tuning and Optimizations
- Application Best Practices
  - Amber, MiniAMR, SMG2000, Neuron, HPCCG, LULESH, MILC and HoomDBlue
- 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/](http://mvapich.cse.ohio-state.edu/best_practices/)
- Initial list of applications
  - Amber
  - HoomdBlue
  - HPCG
  - Lulesh
  - MILC
  - MiniAMR
  - 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.

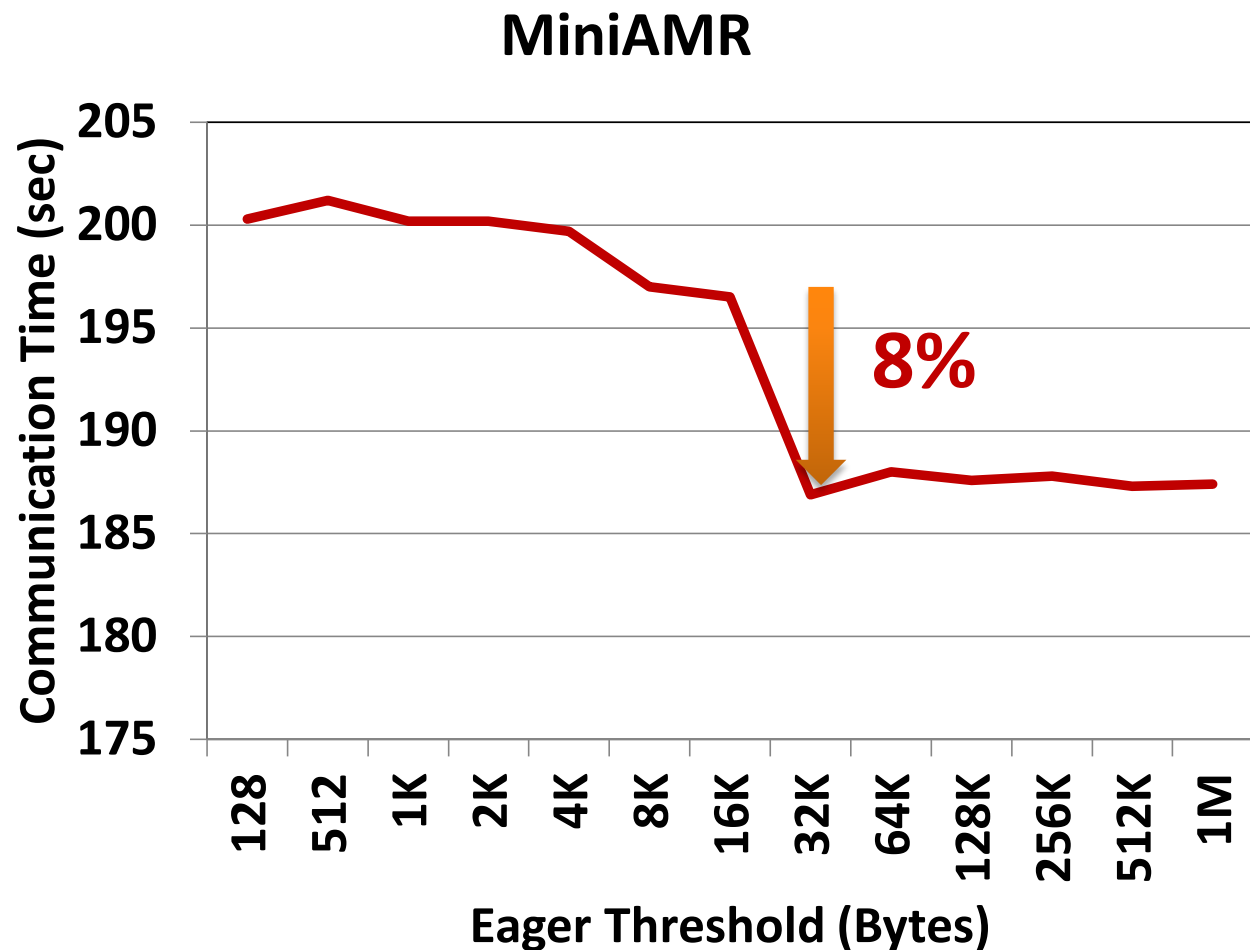
# Amber: Impact of Tuning Eager Threshold



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](#)

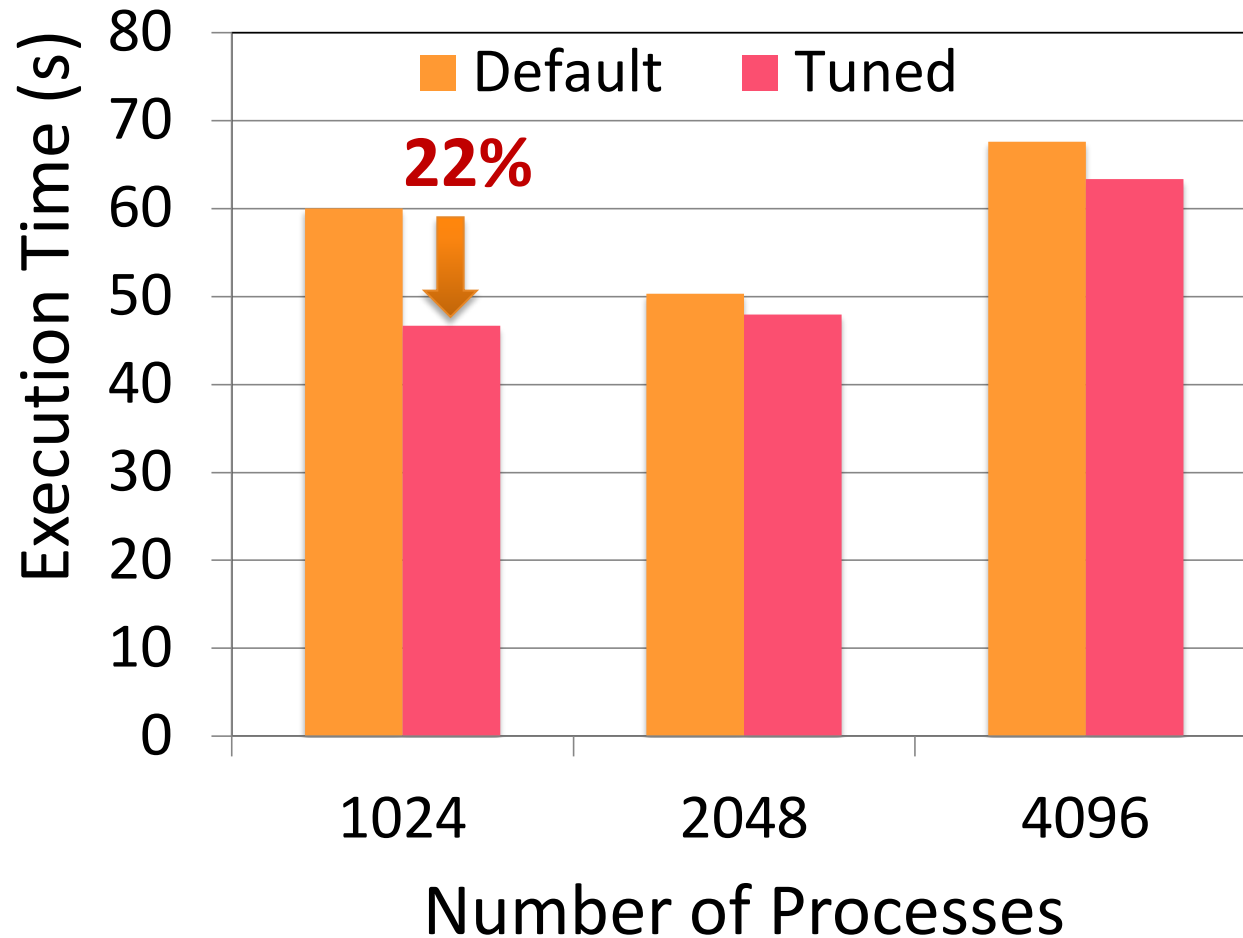
# MiniAMR: Impact of Tuning Eager Threshold



- 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
- 8% percent reduction in total communication time
- Library Version: MVAPICH2 2.2b
- MVAPICH Flags used
  - `MV2_IBA_EAGER_THRESHOLD=32768`
  - `MV2_VBUF_TOTAL_SIZE=32768`

Data Submitted by Karen Tomko @ OSC and Dong Ju Choi @ UCSD

## SMG2000: Impact of Tuning Transport Protocol

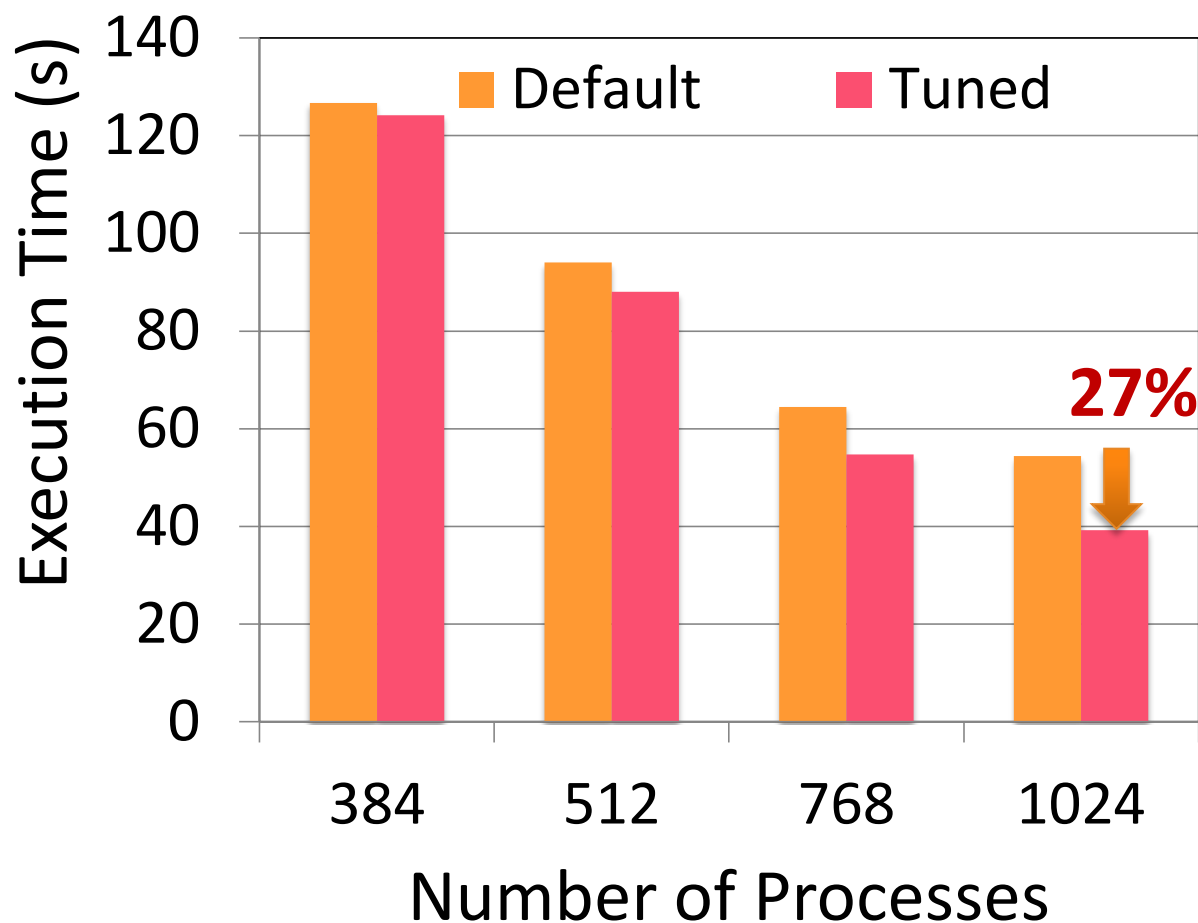


Data Submitted by Jerome Vienne @ TACC

- 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



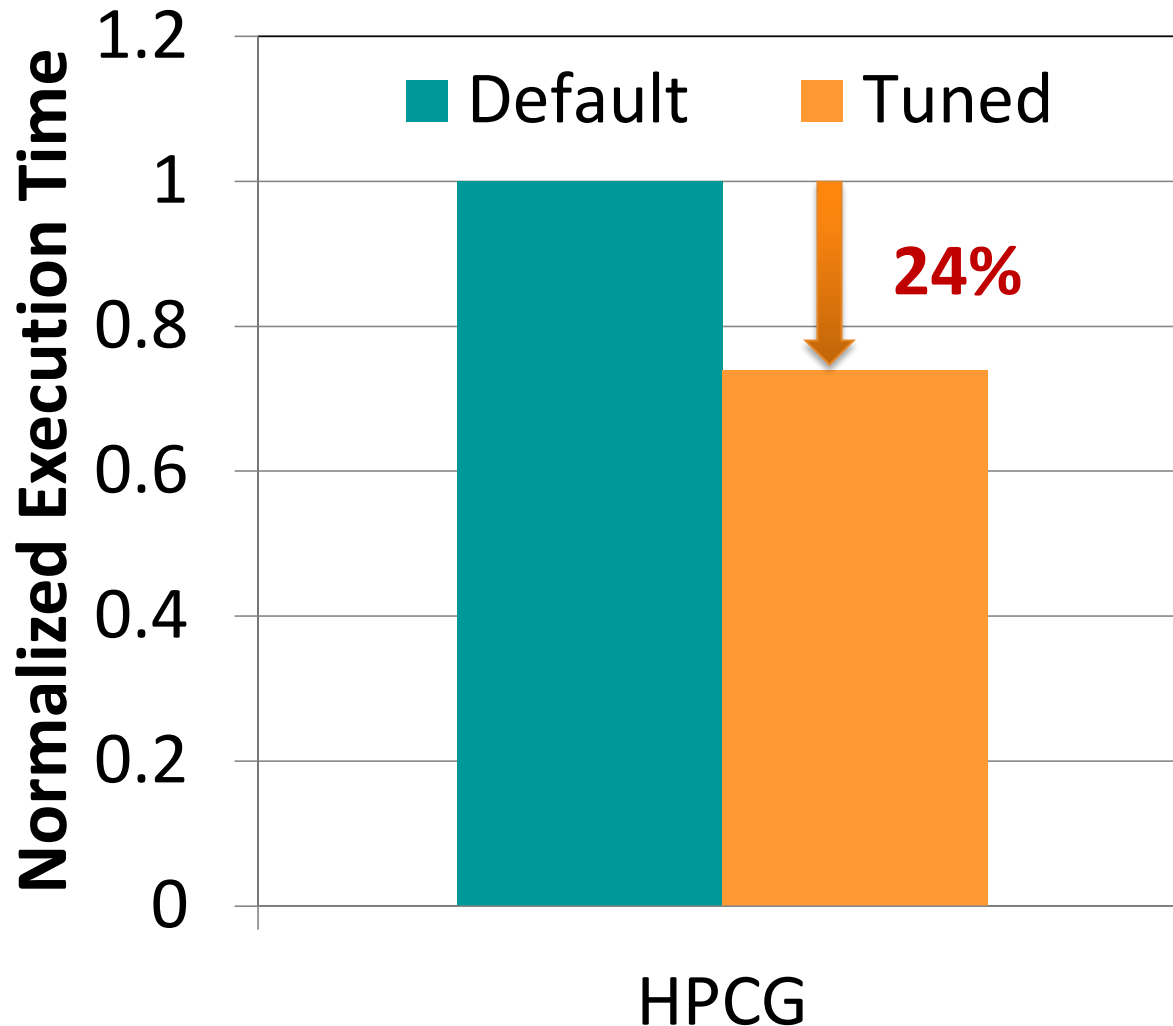
## Neuron: Impact of Tuning Transport Protocol



Data Submitted by Mahidhar Tatineni @ SDSC

- UD-based transport protocol selection benefits the SMG2000 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.

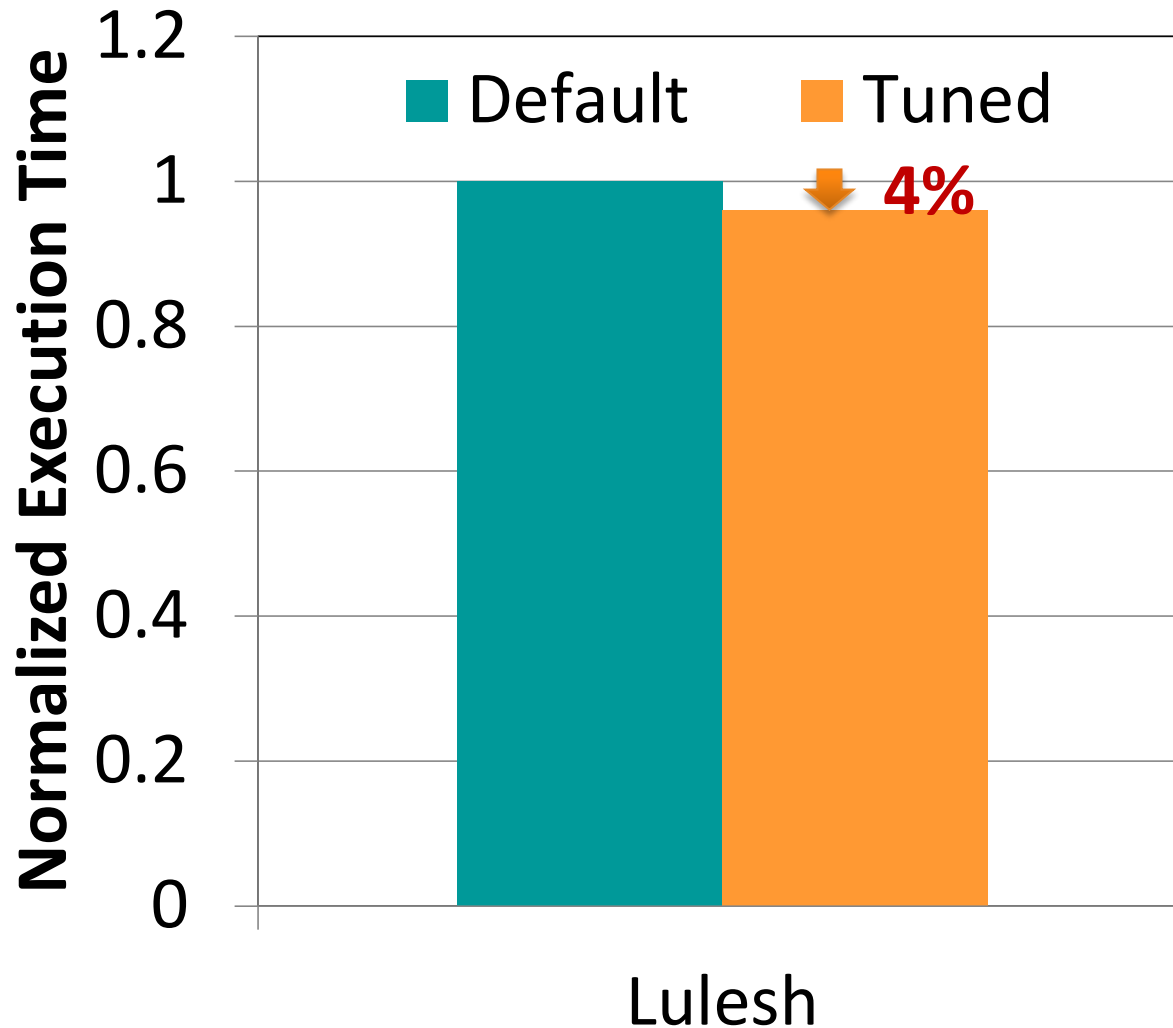
# HPCG: Impact of Collective Tuning for MPI+OpenMP Programming Model



- 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

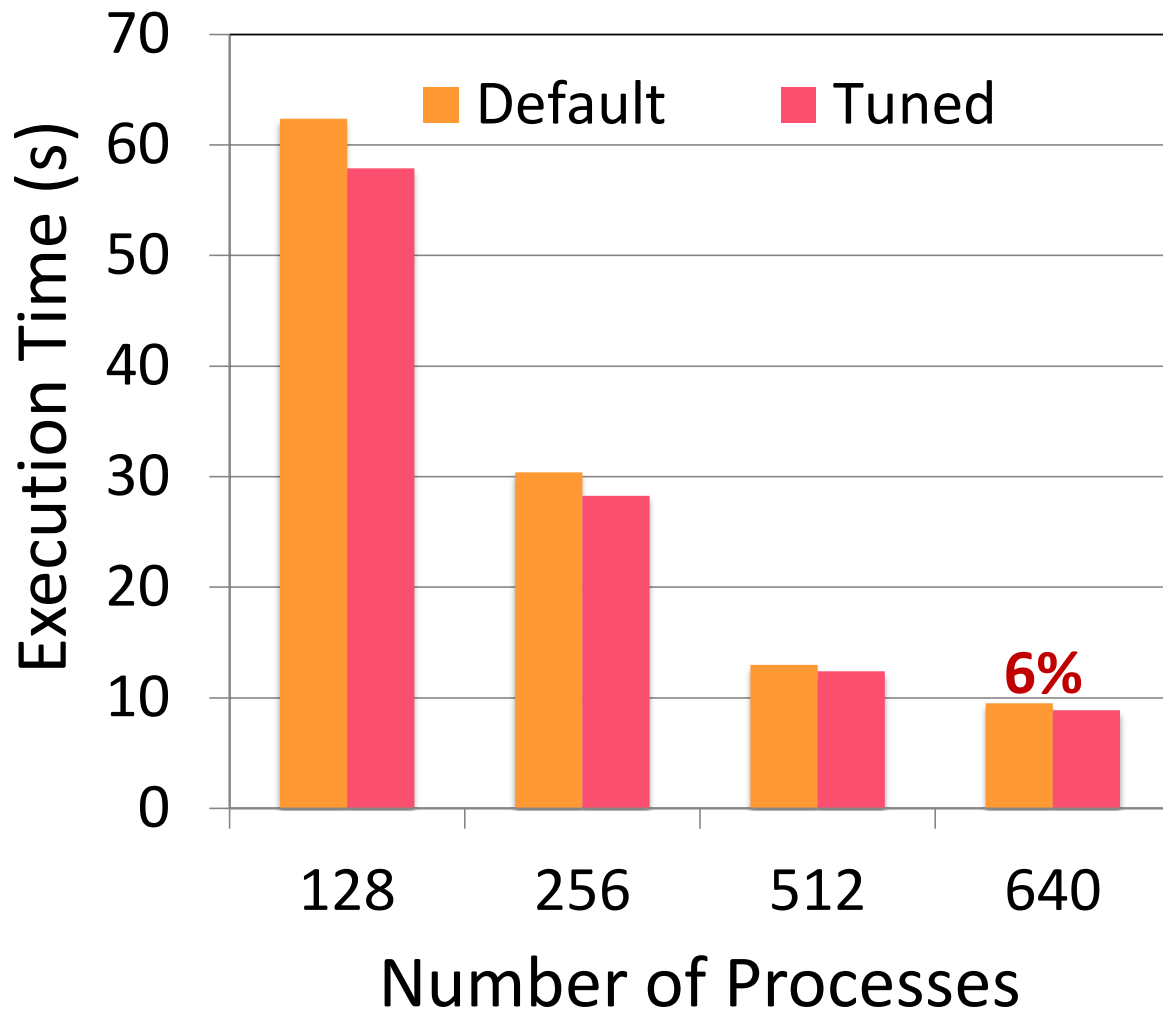
# LULESH: Impact of Collective Tuning for MPI+OpenMP Programming Model



- 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

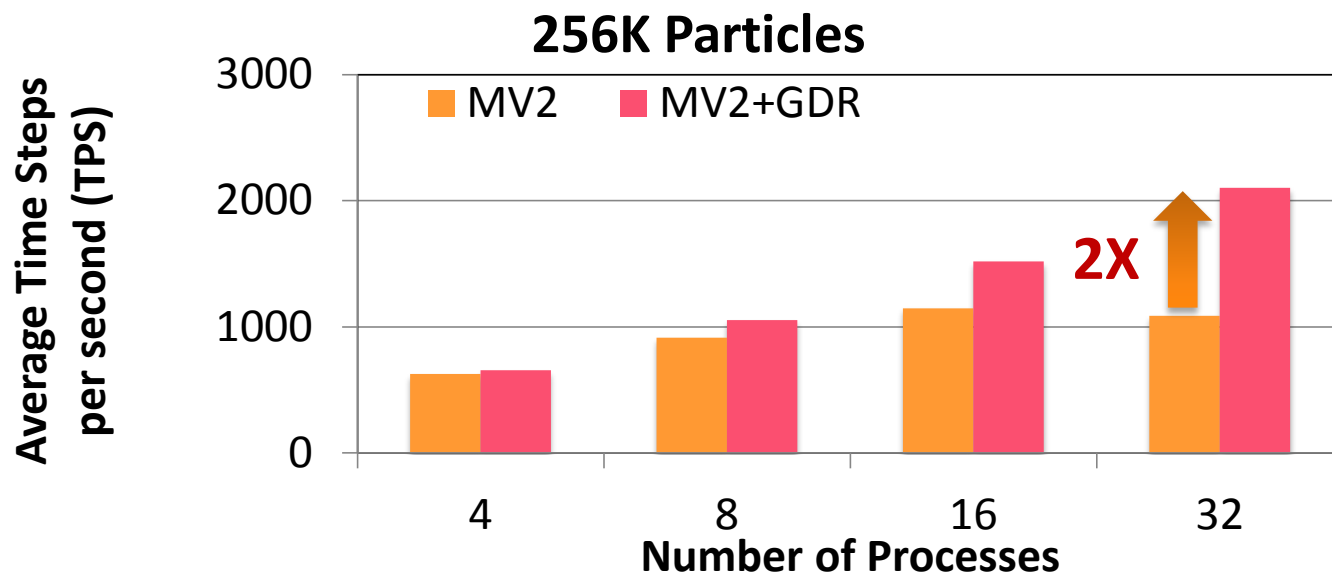
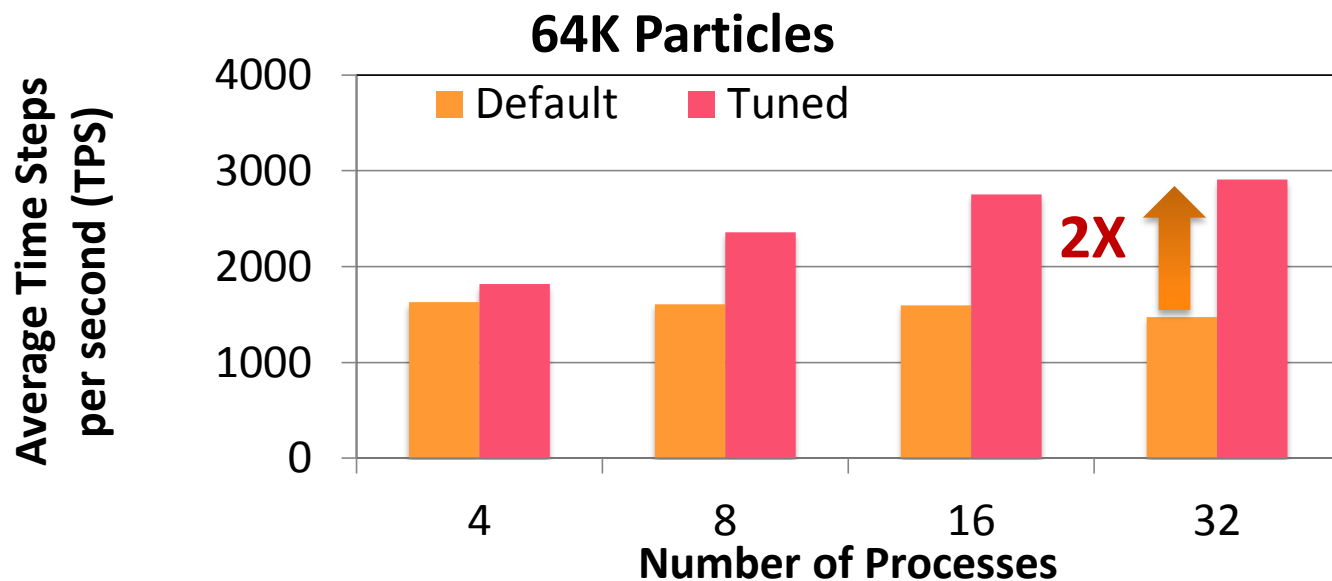
# MILC: Impact of User-mode Memory Registration (UMR) based tuning



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

- HoomdBlue is a Molecular Dynamics simulation using a custom force field.
- GPUDirect specific features selection and tuning significantly benefit the HoomdBlue 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

## Concluding Remarks

- Provided an overview of the different MVAPICH2 software libraries
- Presented details on configuration and runtime parameters, optimizations and their impacts for MVAPICH2 and MVAPICH2-GDR libraries
- Demonstrated best practices for several commonly used applications

# MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + UPC++, MPI + CAF ...)
  - Support for task-based parallelism (UPC++)
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features
  - User Mode Memory Registration (UMR)
  - On-demand Paging (ODP)
- Enhanced Inter-node and Intra-node communication schemes for upcoming OmniPath and Knights Landing architectures
- Extended RMA support (as in MPI 3.0)
- Extended topology-aware collectives
- Energy-aware point-to-point (one-sided and two-sided) and collectives
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended Checkpoint-Restart and migration support with SCR

# Funding Acknowledgments

## Funding Support by



## Equipment Support by





# Personnel Acknowledgments

## ***Current Students***

- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- S. Chakraborty (Ph.D.)
- C.-H. Chu (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

## ***Current Research Scientists*** ***Current Senior Research Associate***

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

## ***Current Post-Doc***

- J. Lin
- D. Banerjee

## ***Current Programmer***

- J. Perkins

## ***Current Research Specialist***

- M. Arnold

- 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***

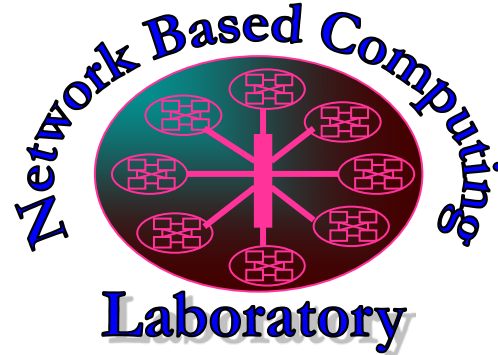
- S. Sur

## ***Past Programmers***

- D. Bureddy

# Thank You!

[panda@cse.ohio-state.edu](mailto:panda@cse.ohio-state.edu)



Network-Based Computing Laboratory

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



## MVAPICH

The MVAPICH2 Project

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