

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

A Tutorial at the
MVAPICH User Group (MUG) Meeting '18
by

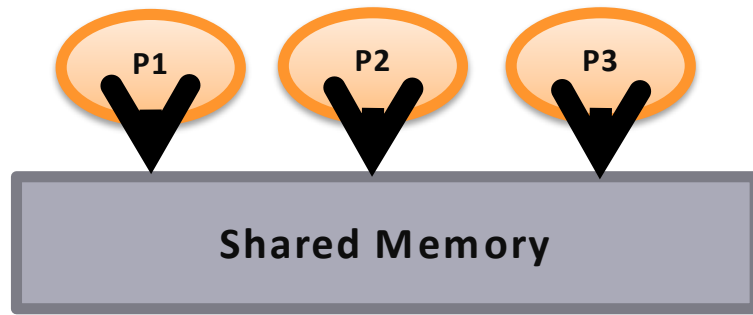
The MVAPICH Team

The Ohio State University

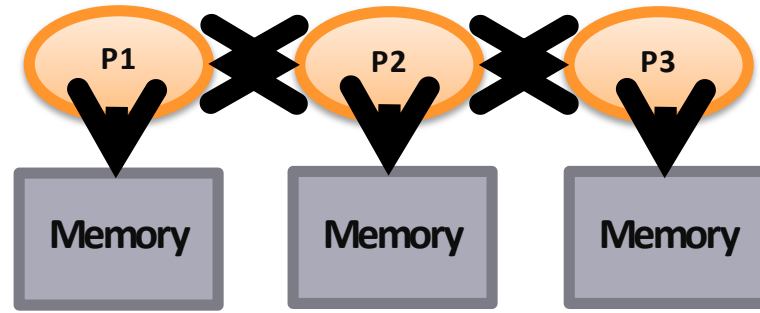
E-mail: panda@cse.ohio-state.edu

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

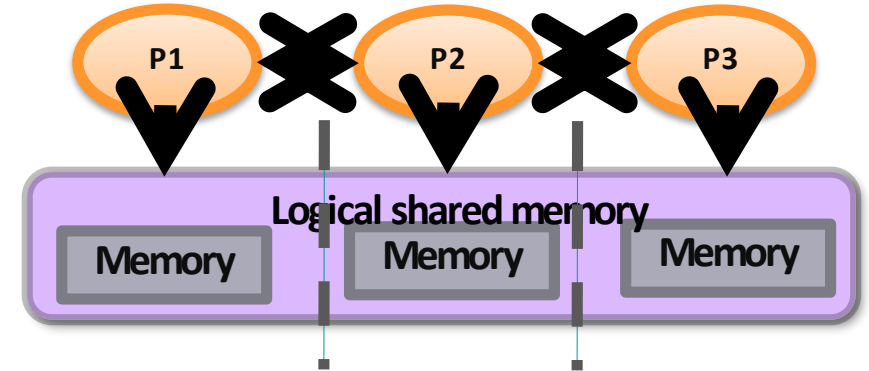
Parallel Programming Models Overview



Shared Memory Model
SHMEM, DSM



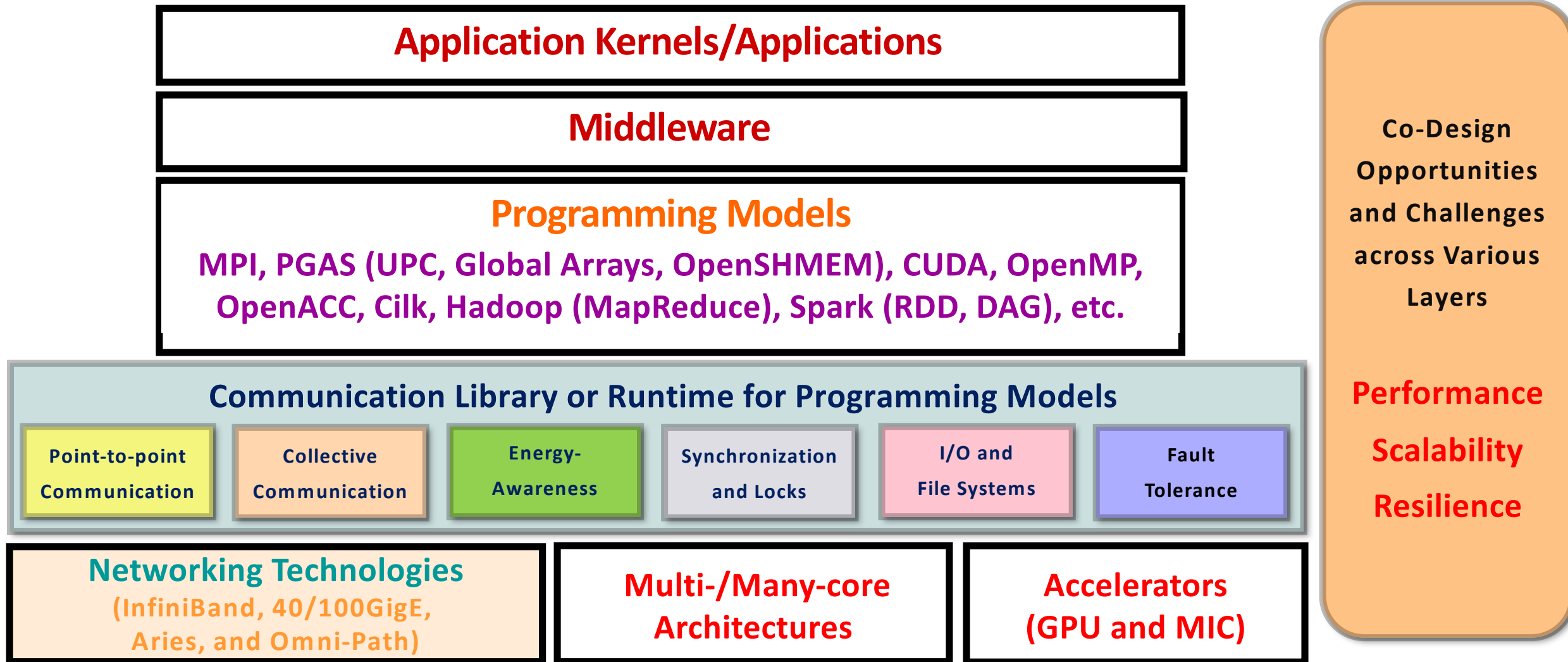
Distributed Memory Model
MPI (Message Passing Interface)



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

Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges



Designing (MPI+X) for Exascale

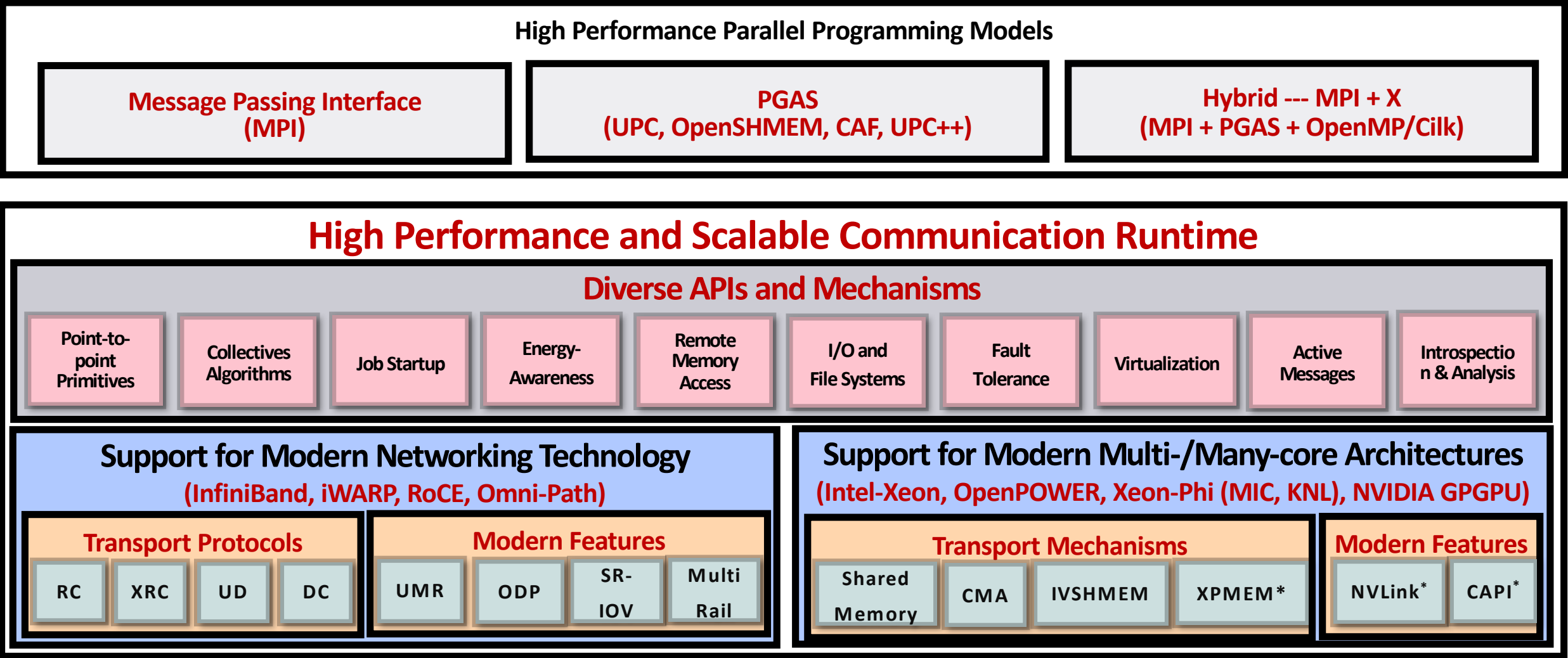
- Scalability for million to billion processors
 - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
- Scalable Collective communication
 - Offloaded
 - Non-blocking
 - Topology-aware
- Balancing intra-node and inter-node communication for next generation multi-/many-core (128-1024 cores/node)
 - Multiple end-points per node
- Support for efficient multi-threading
- Integrated Support for GPGPUs and Accelerators
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming
 - MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, MPI + UPC++...
- Virtualization
- Energy-Awareness

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
 - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 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
 - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
 - **Used by more than 2,925 organizations in 86 countries**
 - **More than 484,000 (> 0.48 million) downloads from the OSU site directly**
 - Empowering many TOP500 clusters (Jul '18 ranking)
 - 2nd ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China
 - 12th, 556,104 cores (Oakforest-PACS) in Japan
 - 15th, 367,024 cores (Stampede2) at TACC
 - 24th, 241,108-core (Pleiades) at NASA and many others
 - Available with software stacks of many vendors and Linux Distro (RedHat and SuSE)
 - <http://mvapich.cse.ohio-state.edu>
- Empowering Top500 systems for over a decade



Architecture of MVAPICH2 Software Family



* Upcoming

MVAPICH2 Software Family

Requirements	Library
MPI with IB, iWARP, Omni-Path, and RoCE	MVAPICH2
MPI with IB, RoCE & GPU and Support for Deep Learning	MVAPICH2-GDR
InfiniBand Network Analysis and Monitoring	OSU INAM
Microbenchmarks for Measuring MPI and PGAS Performance	OMB

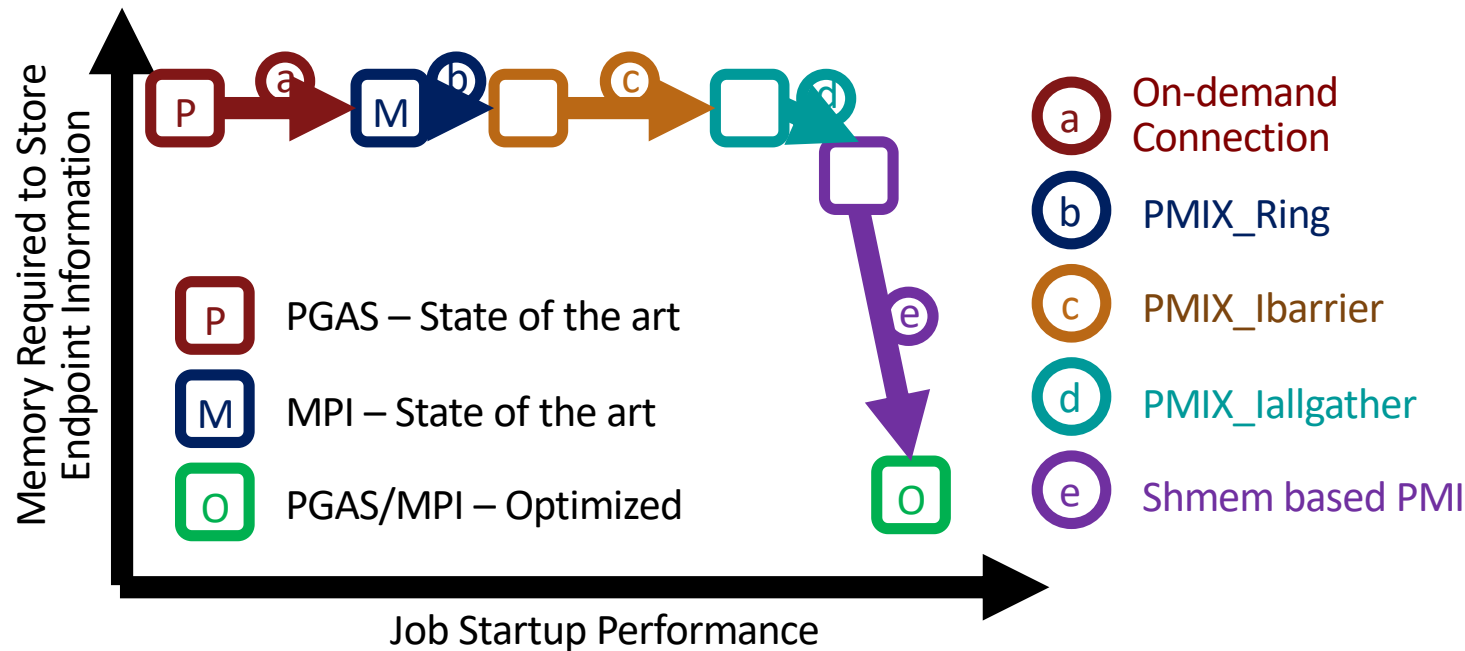
MVAPICH2 2.3-GA

- Released on 07/23/2018
- Major Features and Enhancements
 - Based on MPICH v3.2.1
 - Introduce basic support for executing MPI jobs in Singularity
 - Improve performance for MPI-3 RMA operations
 - **Enhancements for Job Startup**
 - Improved job startup time for OFA-IB-CH3, PSM-CH3, and PSM2-CH3
 - On-demand connection management for PSM-CH3 and PSM2-CH3 channels
 - Enhance PSM-CH3 and PSM2-CH3 job startup to use non-blocking PMI calls
 - Introduce capability to run MPI jobs across multiple InfiniBand subnets
 - **Enhancements to point-to-point operations**
 - Enhance performance of point-to-point operations for CH3-Gen2 (InfiniBand), CH3-PSM, and CH3-PSM2 (Omni-Path) channels
 - Improve performance for Intra- and Inter-node communication for OpenPOWER architecture
 - Enhanced tuning for OpenPOWER, Intel Skylake and Cavium ARM (ThunderX) systems
 - Improve performance for host-based transfers when CUDA is enabled
 - Improve support for large processes per node and hugepages on SMP systems
 - **Enhancements to collective operations**
 - Enhanced performance for Allreduce, Reduce_scatter_block, Allgather, Allgatherv
 - Thanks to Danielle Sikich and Adam Moody @ LLNL for the patch
 - Add support for non-blocking Allreduce using Mellanox SHARP
 - Enhance tuning framework for Allreduce using SHARP
 - Enhanced collective tuning for IBM POWER8, IBM POWER9, Intel Skylake, Intel KNL, Intel Broadwell
 - **Enhancements to process mapping strategies and automatic architecture/network detection**
 - Improve performance of architecture detection on high core-count systems
 - Enhanced architecture detection for OpenPOWER, Intel Skylake and Cavium ARM (ThunderX) systems
 - New environment variable MV2_THREADS_BINDING_POLICY for multi-threaded MPI and MPI+OpenMP applications
 - Support 'spread', 'bunch', 'scatter', 'linear' and 'compact' placement of threads
 - Warn user if oversubscription of core is detected
 - Enhance MV2_SHOW_CPU_BINDING to enable display of CPU bindings on all nodes
 - Added support for MV2_SHOW_CPU_BINDING to display number of OMP threads
 - Added logic to detect heterogeneous CPU/HFI configurations in PSM-CH3 and PSM2-CH3 channels
 - Thanks to Matias Cabral@Intel for the report
 - Enhanced HFI selection logic for systems with multiple Omni-Path HFIs
 - Introduce run time parameter MV2_SHOW_HCA_BINDING to show process to HCA bindings
 - **Miscellaneous enhancements and improved debugging and tools support**
 - Enhance support for MPI_T PVARs and CVARs
 - Enhance debugging support for PSM-CH3 and PSM2-CH3 channels
 - Update to hwloc version 1.11.9
 - Tested with CLANG v5.0.0

Presentation Overview

- **Job start-up**
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

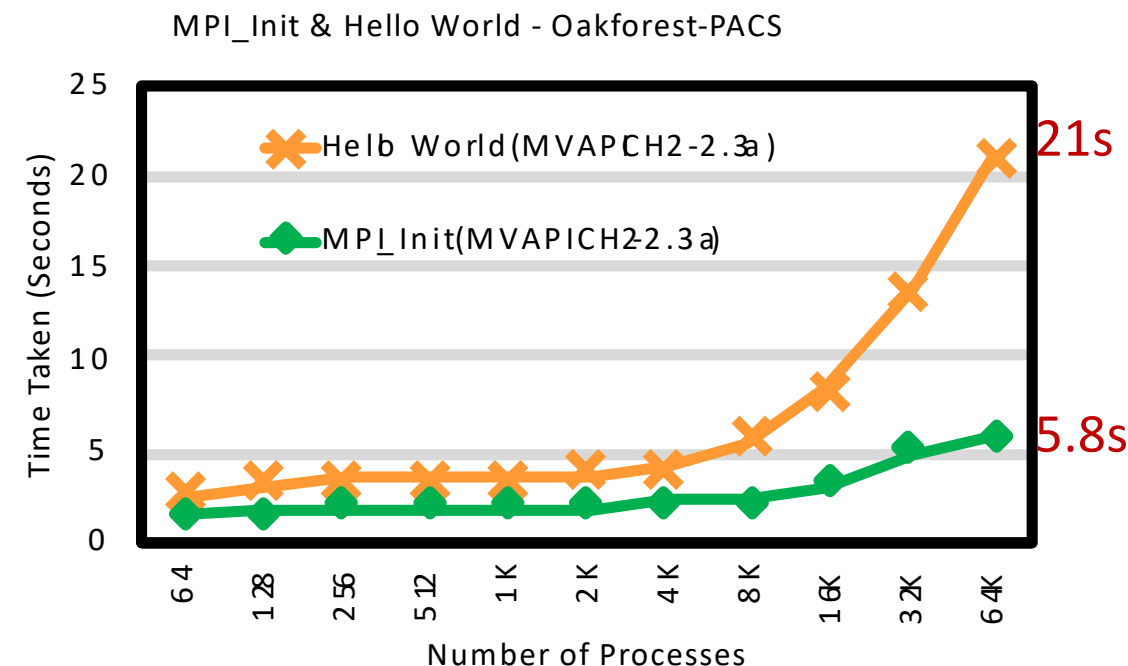
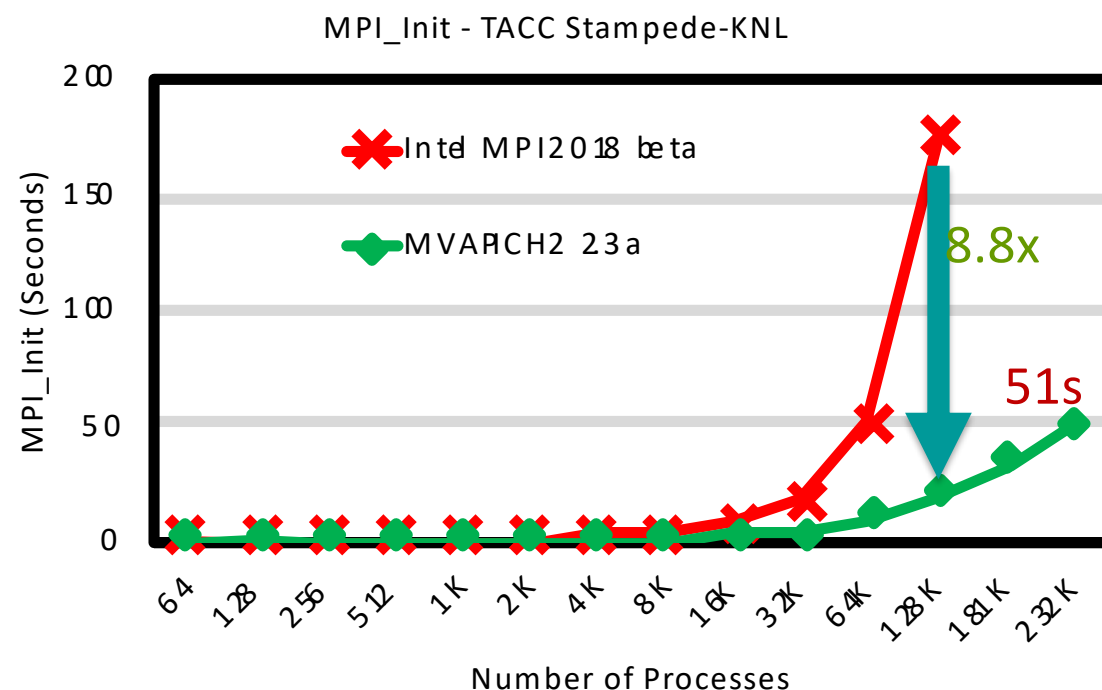
Towards High Performance and Scalable Startup at Exascale



- 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)

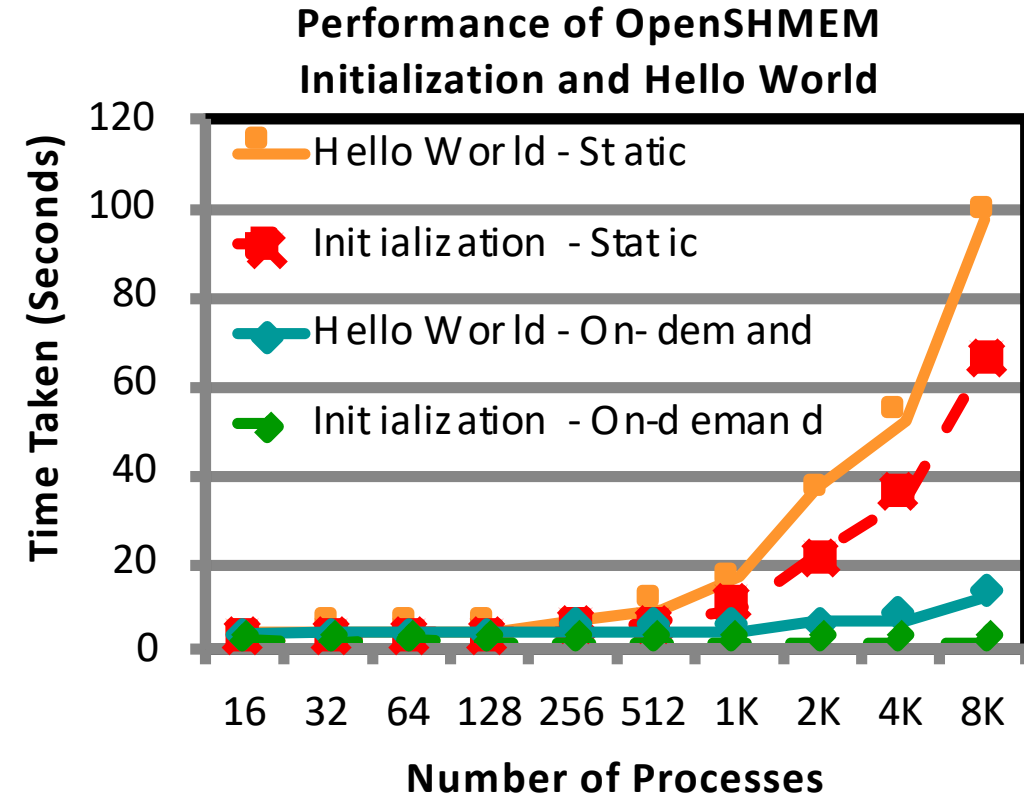
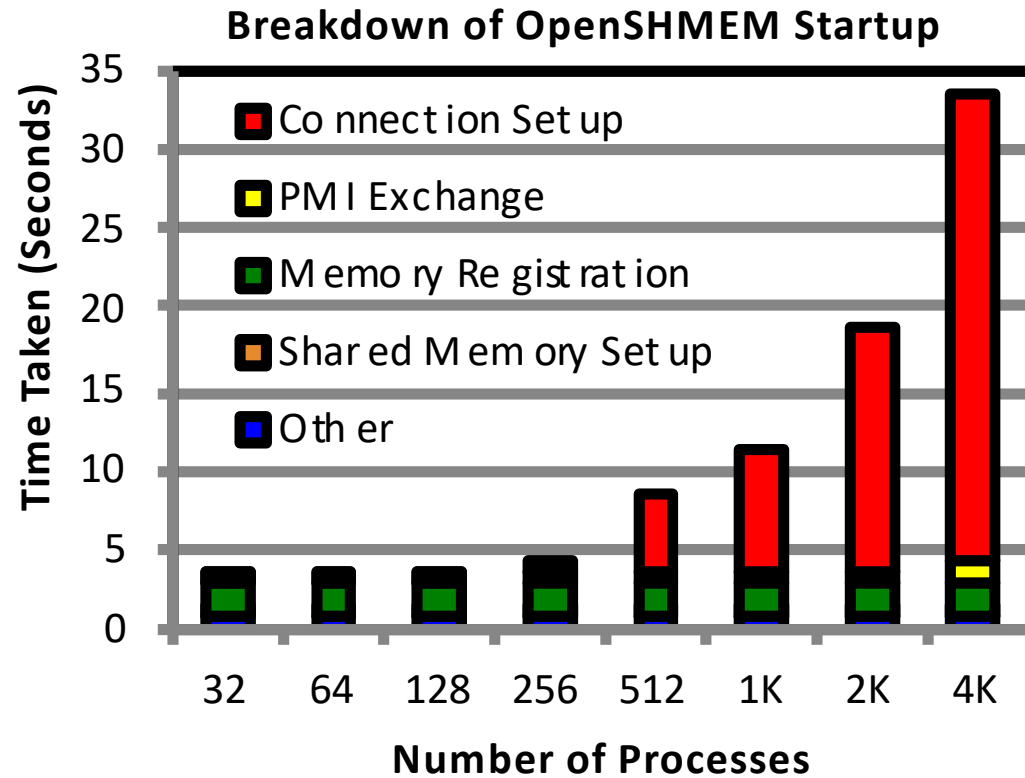
Startup Performance on KNL + Omni-Path



- MPI_Init takes 51 seconds on 231,956 processes on 3,624 KNL nodes (Stampede – Full scale)
- 8.8 times faster than Intel MPI at 128K processes (Courtesy: TACC)
- At 64K processes, MPI_Init and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
- All numbers reported with 64 processes per node

New designs available in MVAPICH2-2.3a and as patch for SLURM-15.08.8 and SLURM-16.05.1

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**

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 available at <http://mvapich.cse.ohio-state.edu/download/>
 - Patches available for SLURM 15, 16, and 17
 - PMI Extensions are automatically detected by MVAPICH2

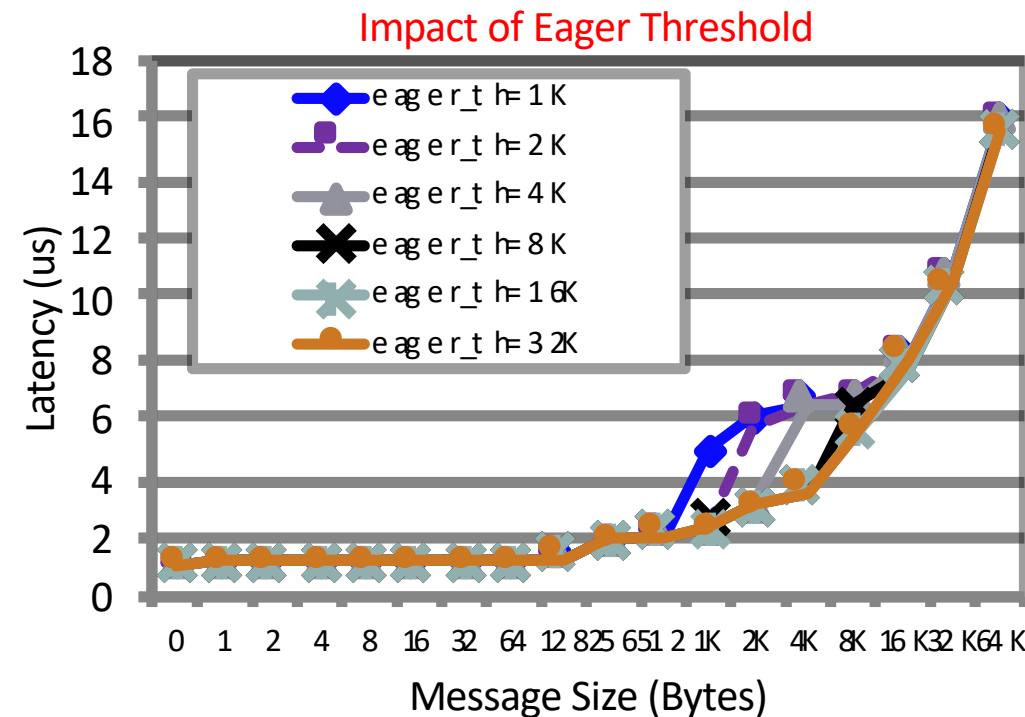
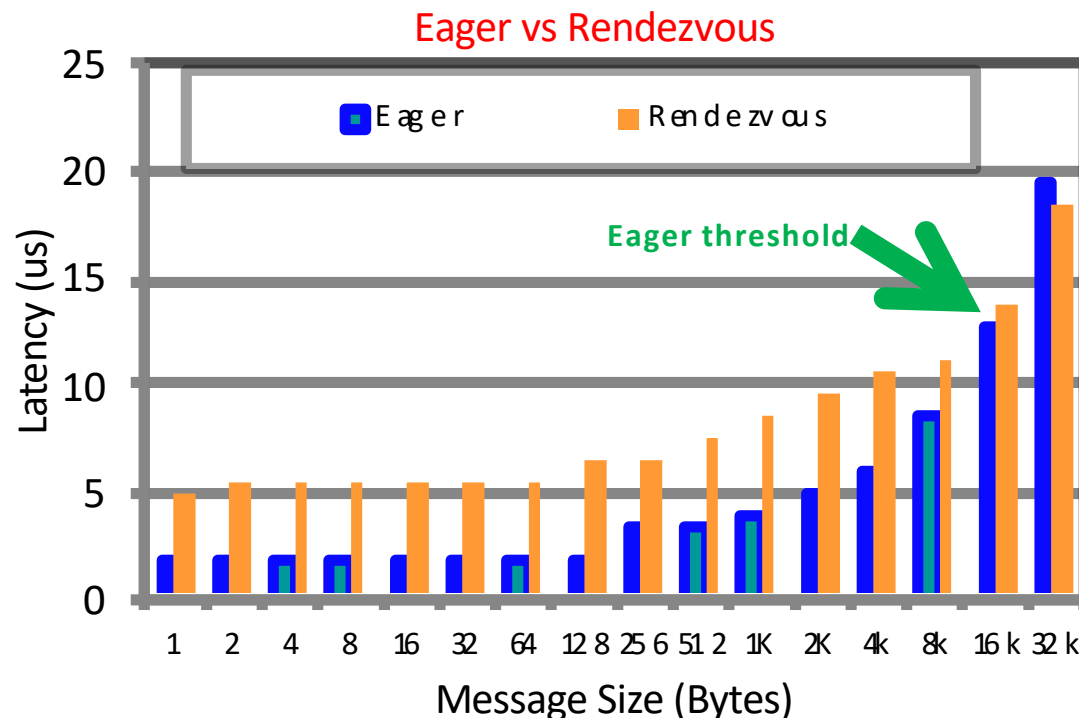
Using mpirun_rsh as launcher

- **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

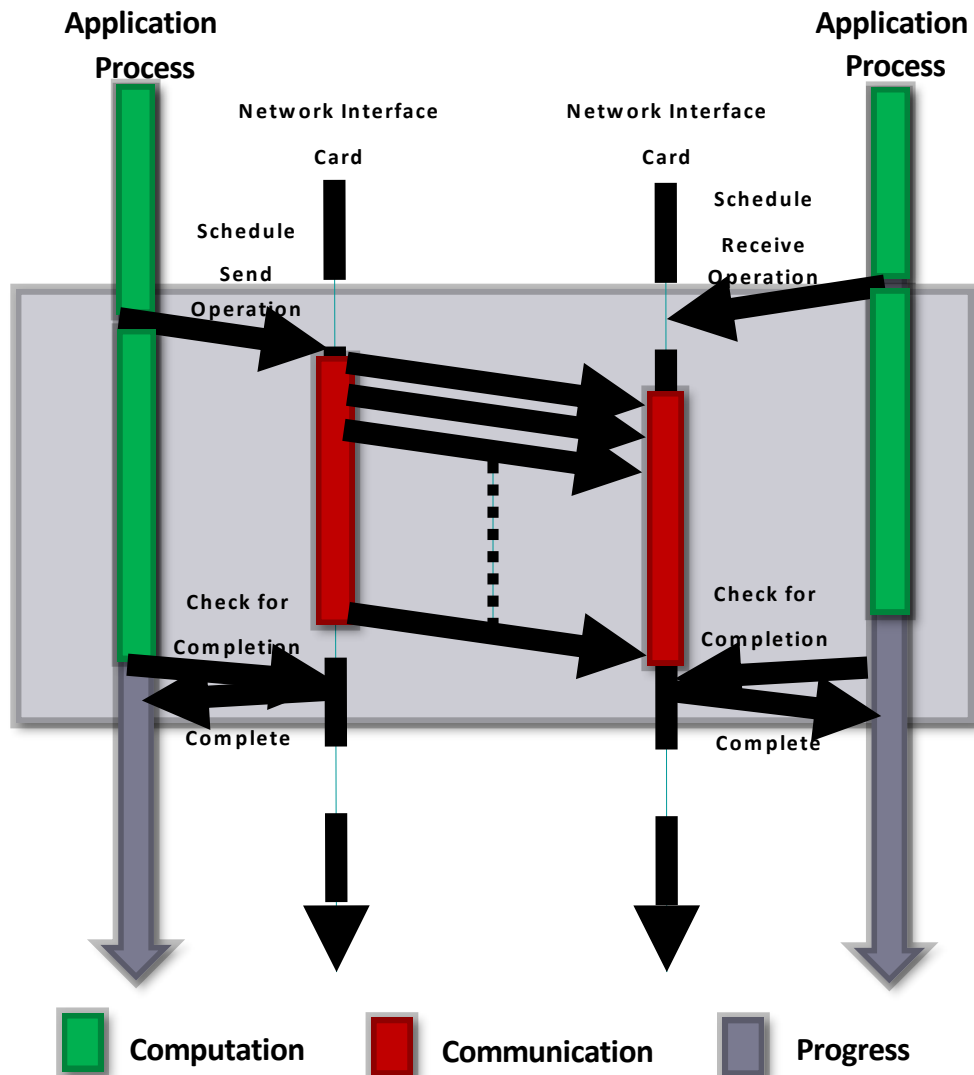
- Job start-up
- **Point-to-point Inter-node Protocol**
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

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

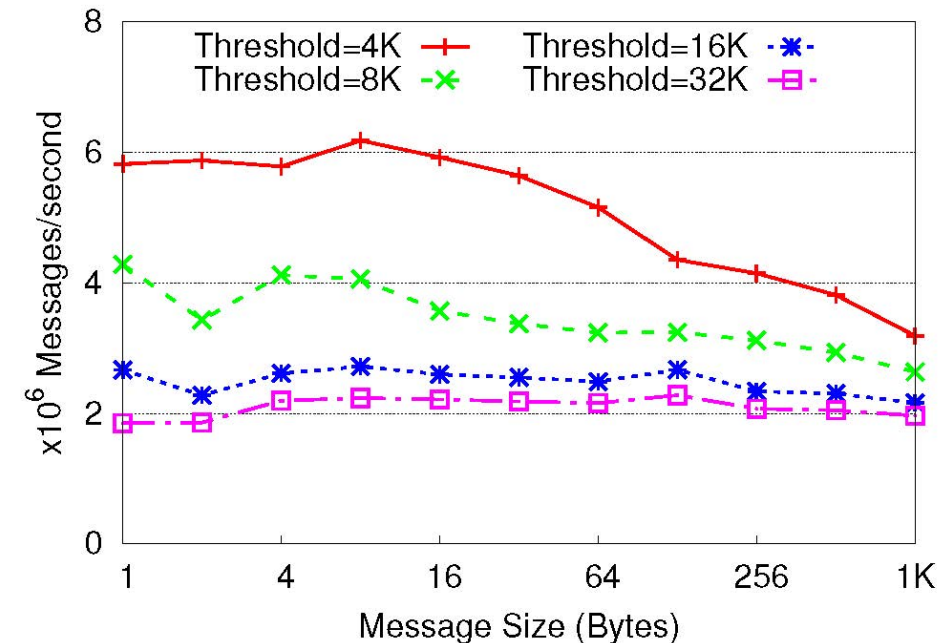


- Switching Eager to Rendezvous transfer
 - Default: Architecture dependent on common platforms, in order to achieve both best performance and memory footprint
- Threshold can be modified by users to get smooth performance across message sizes
 - `mpirun_rsh -np 2 -hostfile hostfile MV2_IBA_EAGER_THRESHOLD=32K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

Analyzing Overlap Potential of Eager Protocol

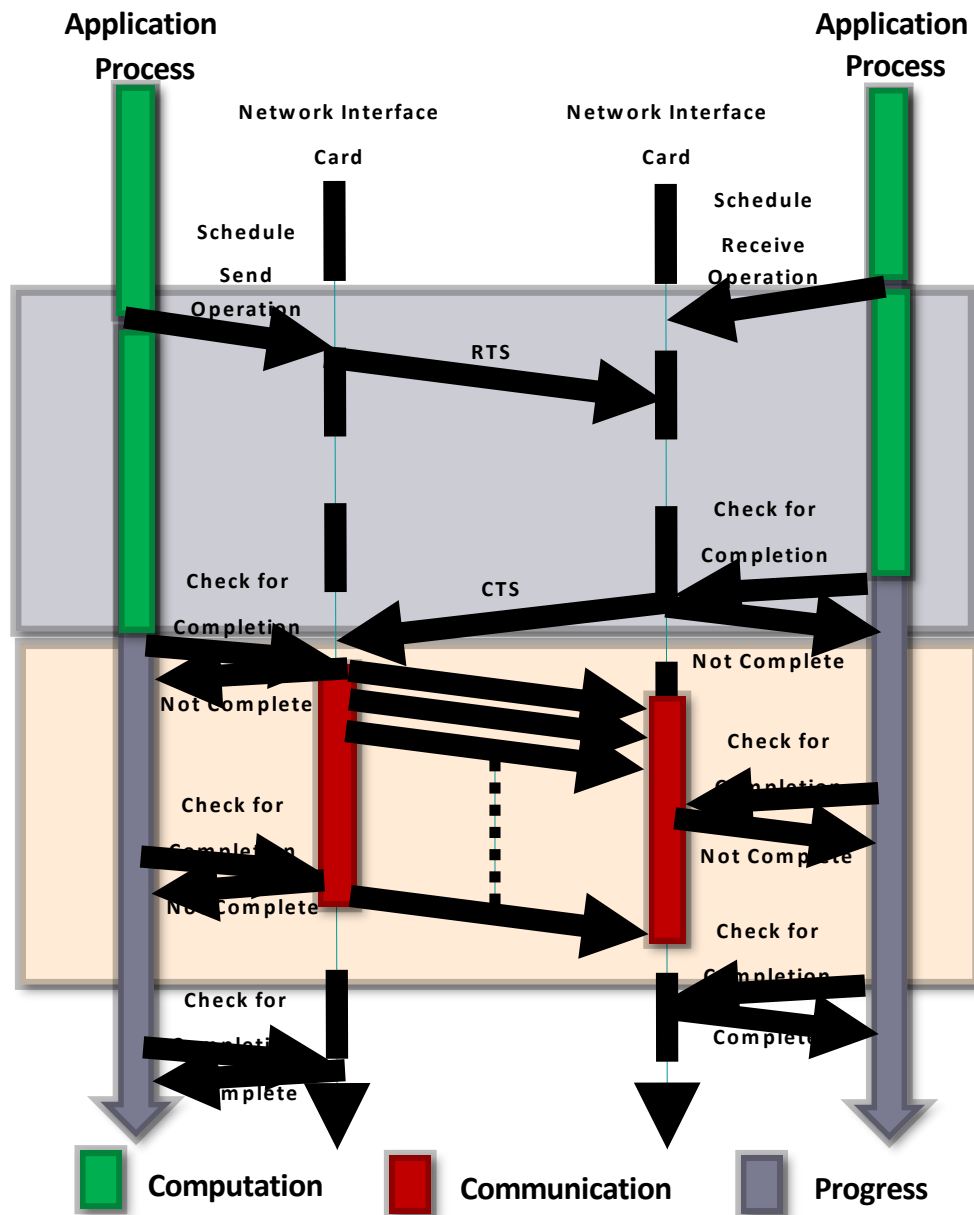


- Application processes schedule communication operation
- Network adapter progresses communication in the background
- Application process free to perform useful compute in the foreground
- **Overlap of computation and communication => Better Overall Application Performance**
- **Increased buffer requirement**
- **Poor communication performance if used for all types of communication operations**



Impact of changing Eager Threshold on performance of multi-pair message-rate benchmark with 32 processes on Stampede

Analyzing Overlap Potential of Rendezvous Protocol



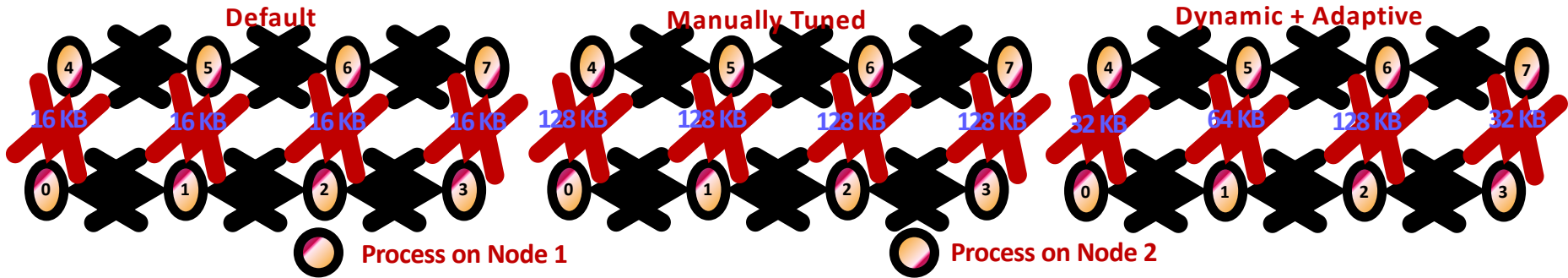
- Application processes schedule communication operation
- Application process free to perform useful compute in the foreground
- Little communication progress in the background
- All communication takes place at final synchronization
- **Reduced buffer requirement**
- **Good communication performance if used for large message sizes and operations where communication library is progressed frequently**
- **Poor overlap of computation and communication => Poor Overall Application Performance**

Dynamic and Adaptive MPI Point-to-point Communication Protocols

Desired Eager Threshold

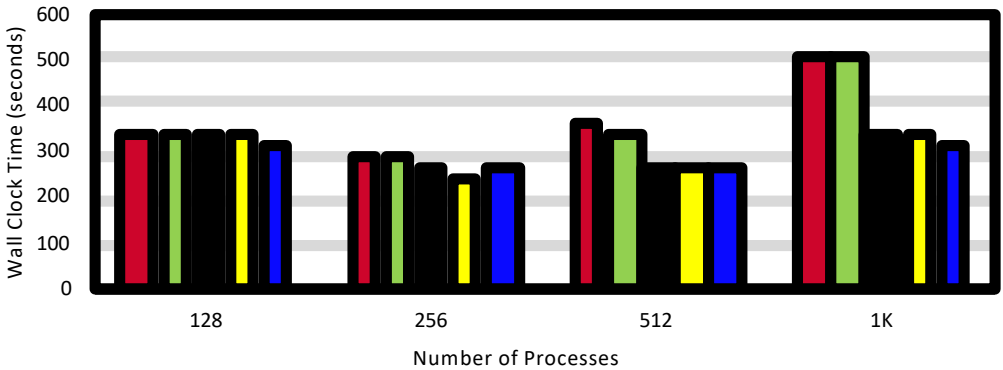
Process Pair	Eager Threshold (KB)
0 – 4	32
1 – 5	64
2 – 6	128
3 – 7	32

Eager Threshold for Example Communication Pattern with Different Designs

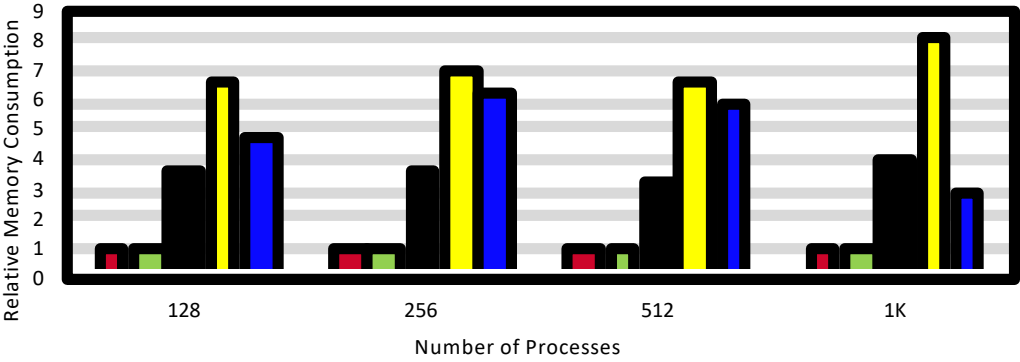


Design	Metrics: Overlap & Memory Requirement	Metrics: Performance & Productivity
Default	Poor overlap; Low memory requirement	Low Performance; High Productivity
Manually Tuned	Good overlap; High memory requirement	High Performance; Low Productivity
Dynamic + Adaptive	Good overlap; Optimal memory requirement	High Performance; High Productivity

Execution Time of Amber



Relative Memory Consumption of Amber



Default Threshold d=17K Threshold d=64K Threshold d=128K Dynamic Threshold

Default Threshold d=17K Threshold d=64K Threshold d=128K Dynamic Threshold

Dynamic and Adaptive Tag Matching

Challenge

Tag matching is a significant overhead for receivers

Existing Solutions are

- Static and do not adapt dynamically to communication pattern
- Do not consider memory overhead

Solution

A new tag matching design

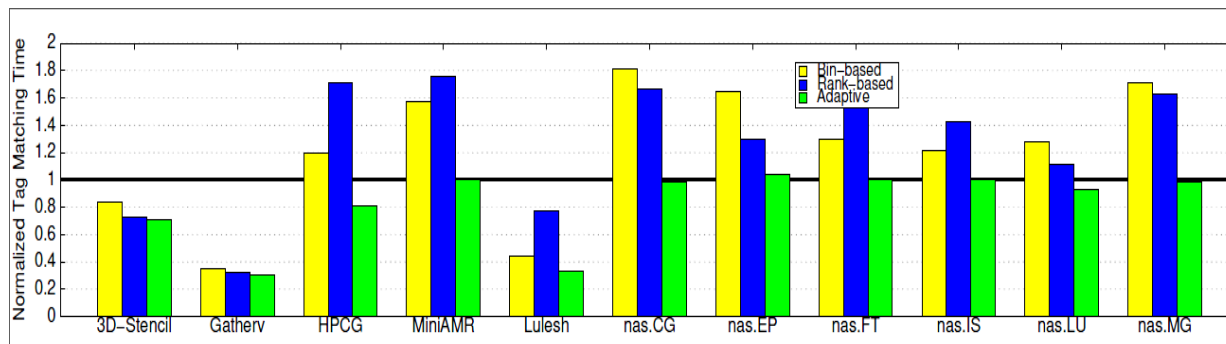
- Dynamically adapt to communication patterns
- Use different strategies for different ranks
- Decisions are based on the number of request object that must be traversed before hitting on the required one

Results

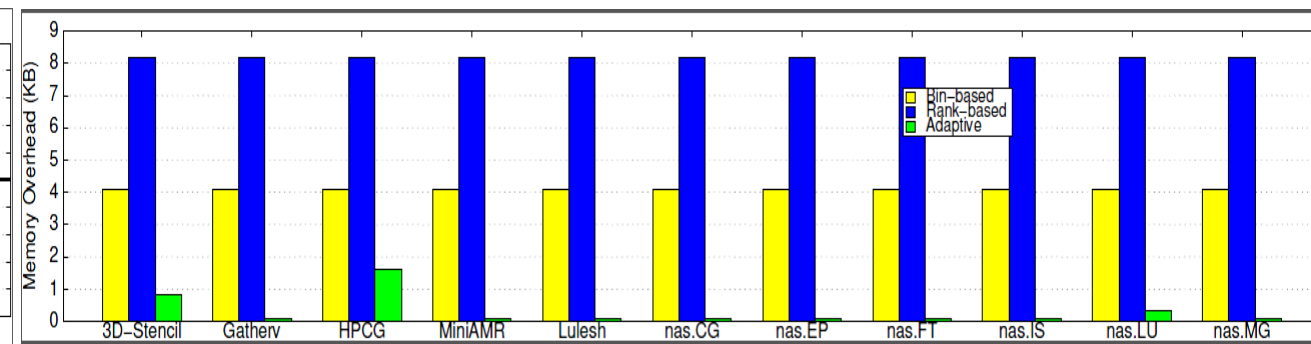
Better performance than other state-of-the-art tag-matching schemes

Minimum memory consumption

Will be available in future MVAPICH2 releases



Normalized Total Tag Matching Time at 512 Processes
Normalized to Default (Lower is Better)

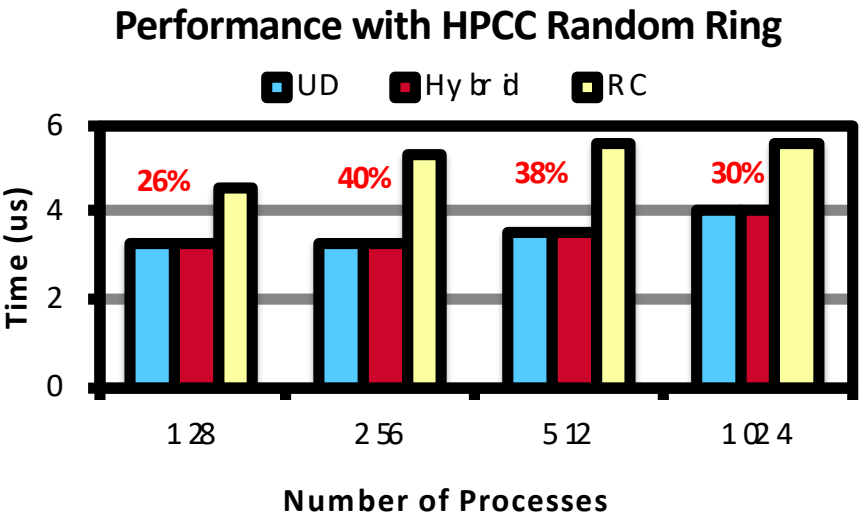


Normalized Memory Overhead per Process at 512 Processes
Compared to Default (Lower is Better)

Presentation Overview

- Job start-up
- Point-to-point Inter-node Protocol
- **Transport Type Selection**
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

Hybrid (UD/RC/XRC) Mode in MVAPICH2

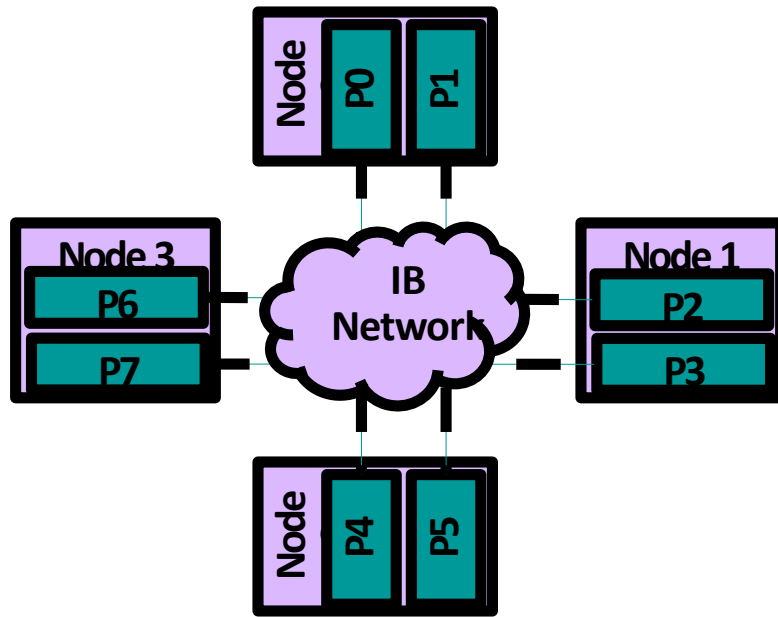


- Both UD and RC/XRC have benefits
 - Hybrid for the best of both
- Enabled by configuring MVAPICH2 with the `-enable-hybrid`
- Available since MVAPICH2 1.7 as integrated interface

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

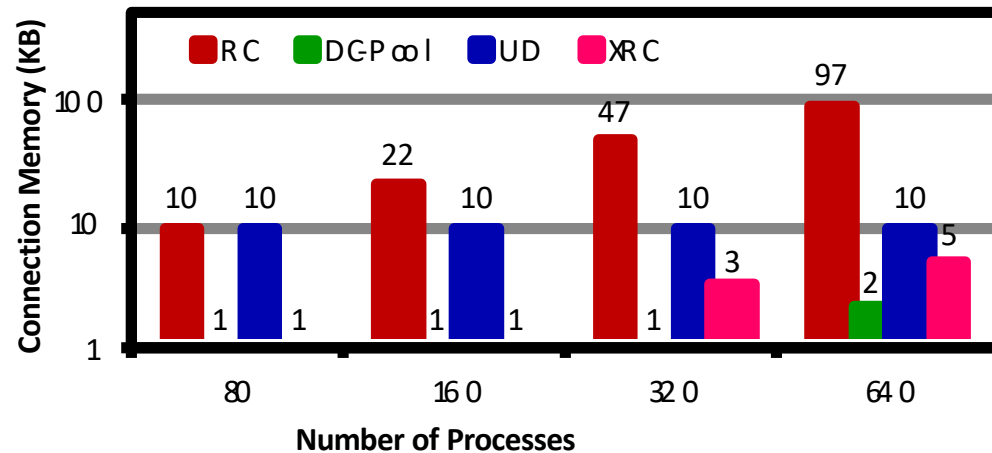
- Refer to **Running with Hybrid UD-RC/XRC** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3a-userguide.html#x1-690006.11>

Minimizing Memory Footprint by Direct Connect (DC) Transport

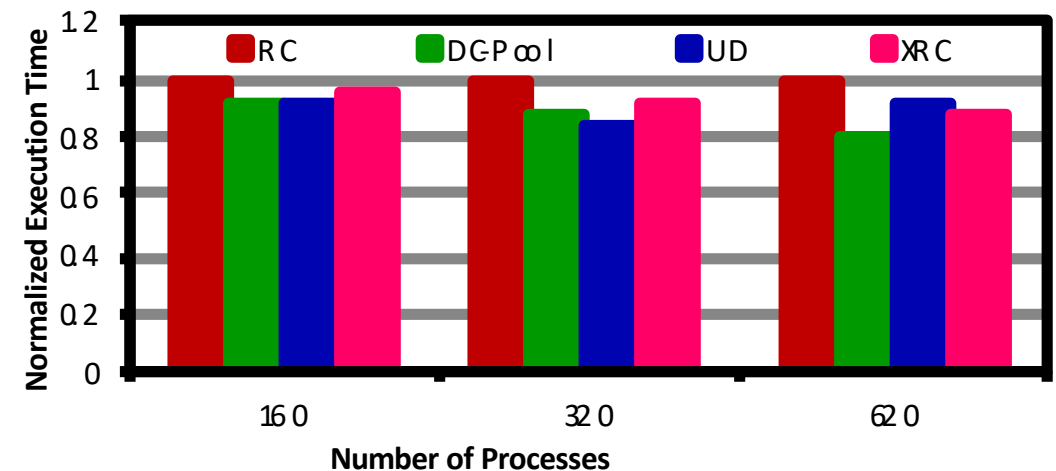


- Constant connection cost (*One QP for any peer*)
- Full Feature Set (RDMA, Atomics etc)
- Separate objects for send (DC Initiator) and receive (DC Target)
 - DC Target identified by “DCT Number”
 - Messages routed with (DCT Number, LID)
 - Requires same “DC Key” to enable communication
- Available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

Memory Footprint for Alltoall



NAMD - Apoa1: Large data set

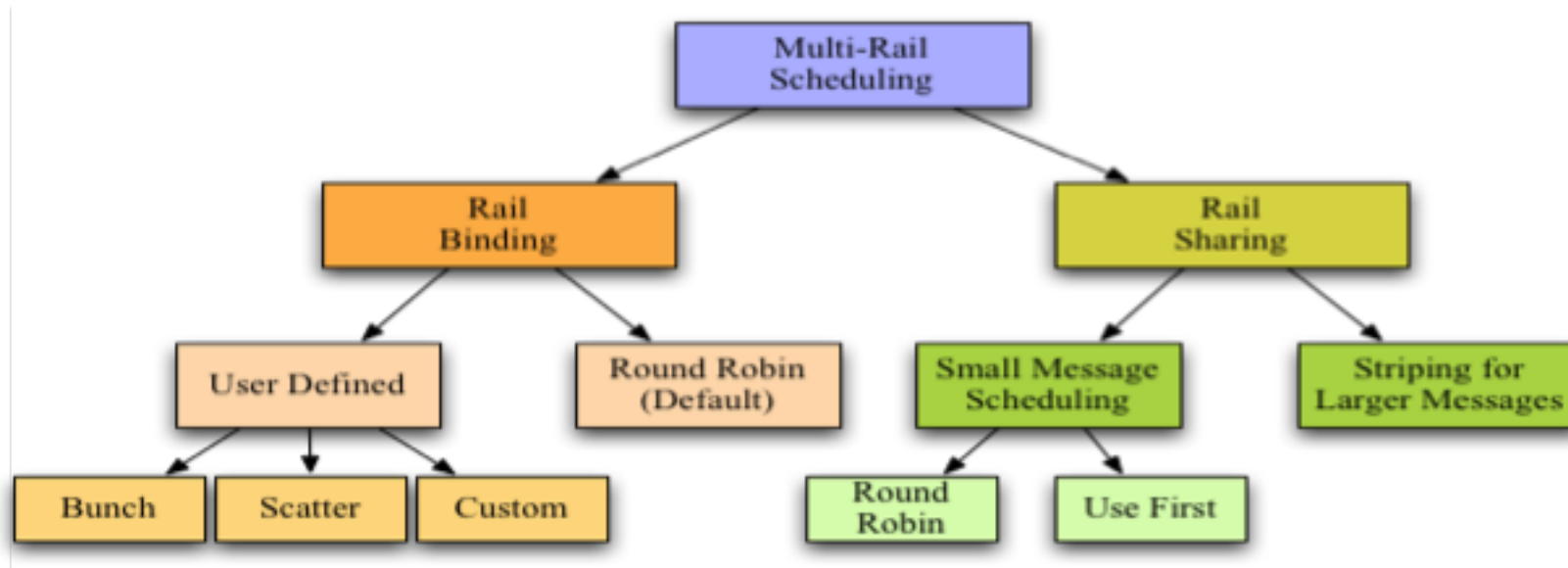


H. Subramoni, K. Hamidouche, A. Venkatesh, S. Chakraborty and D. K. Panda, Designing MPI Library with Dynamic Connected Transport (DCT) of InfiniBand : Early Experiences. IEEE International Supercomputing Conference (ISC '14)

Presentation Overview

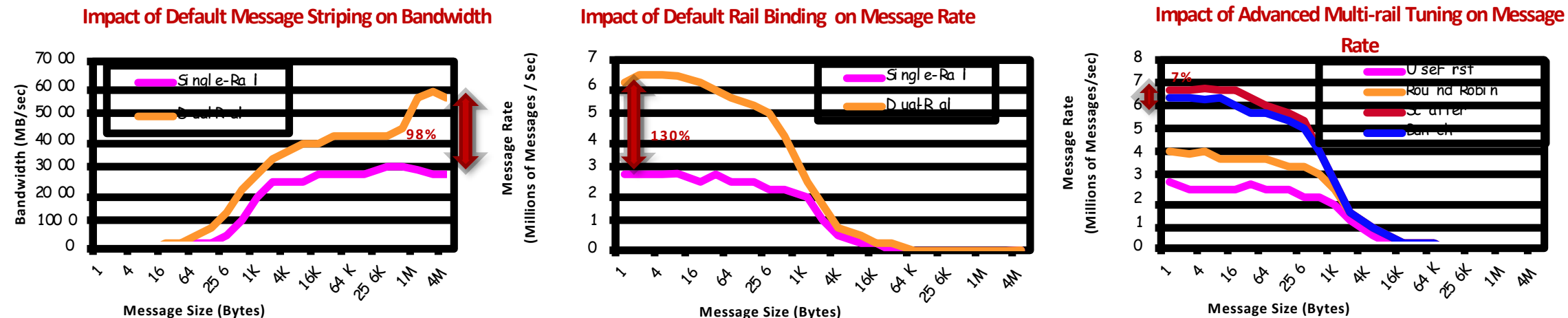
- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- **Multi-rail**
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

MVAPICH2 Multi-Rail Design



- What is a rail?
 - HCA, Port, Queue Pair
- Automatically detects and uses all active HCAs in a system
 - Automatically handles heterogeneity
- Supports multiple rail usage policies
 - Rail Sharing – Processes share all available rails
 - Rail Binding – Specific processes are bound to specific rails

Performance Tuning on Multi-Rail Clusters



Two 24-core Magny Cours nodes with two Mellanox ConnectX QDR adapters
Six pairs with OSU Multi-Pair bandwidth and messaging rate benchmark

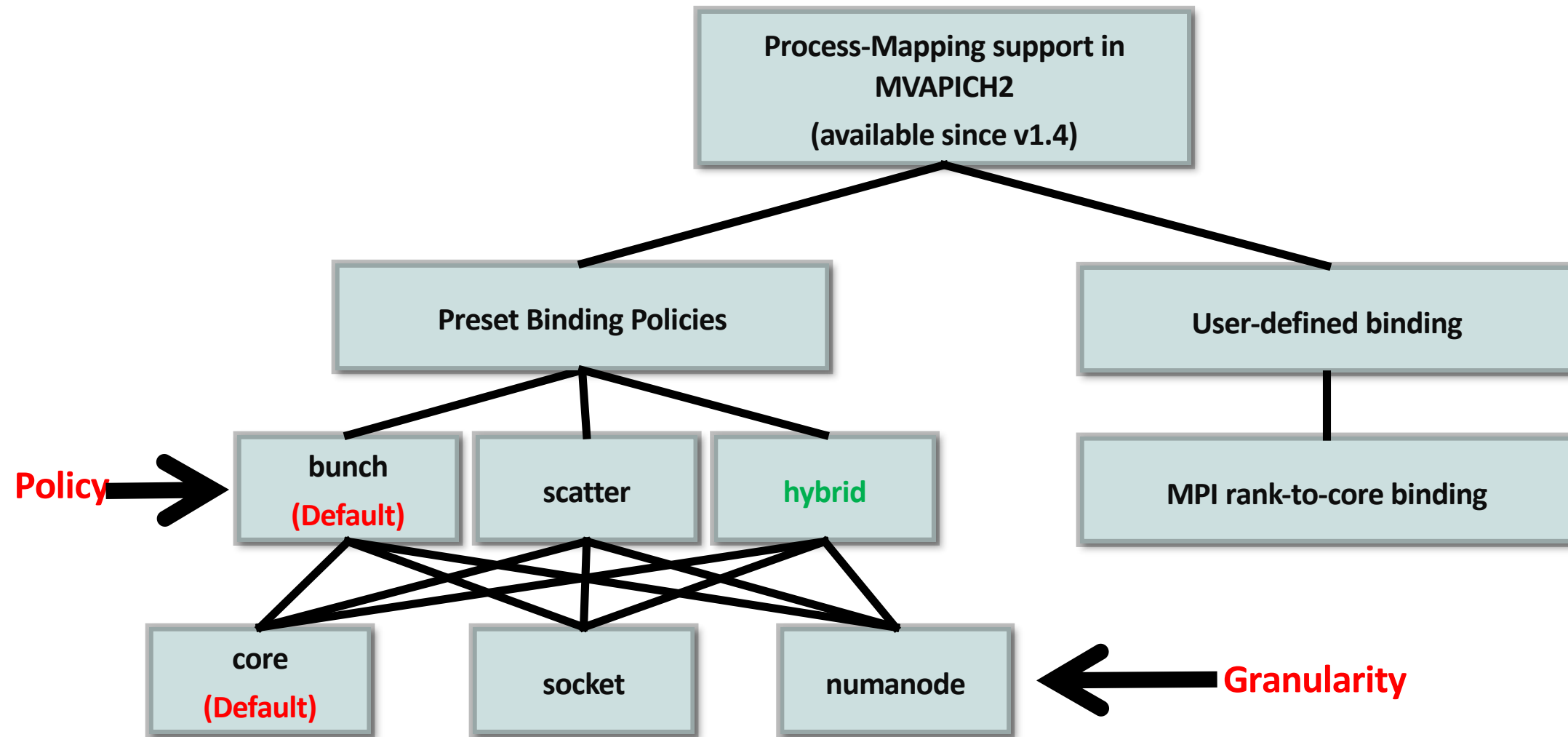
Parameter	Significance	Default	Notes
MV2_IBA_HCA	• Manually set the HCA to be used	Unset	• To get names of HCA ibstat grep “^CA”
MV2_DEFAULT_PORT	• Select the port to use on a active multi port HCA	0	• Set to use different port
MV2_RAIL_SHARING_LARGE_MSG_THRESHOLD	• Threshold beyond which striping will take place	16 Kbyte	
MV2_RAIL_SHARING_POLICY	• Choose multi-rail rail sharing / binding policy • For Rail Sharing set to USE_FIRST or ROUND_ROBIN • Set to FIXED_MAPPING for advanced rail binding options	Rail Binding in Round Robin mode	• Advanced tuning can result in better performance
MV2_PROCESS_TO_RAIL_MAPPING	• Determines how HCAs will be mapped to the rails	BUNCH	• Options: SCATTER and custom list

- Refer to **Enhanced design for Multiple-Rail** section of MVAPICH2 user guide for more information
- <http://mvanich.cse.ohio-state.edu/static/media/mvanich/mvanich2-2.3a-userguide.html#x1-700006.12>

Presentation Overview

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- **Process Mapping and Point-to-point Intra-node Protocols**
- Collectives
- MPI_T Support

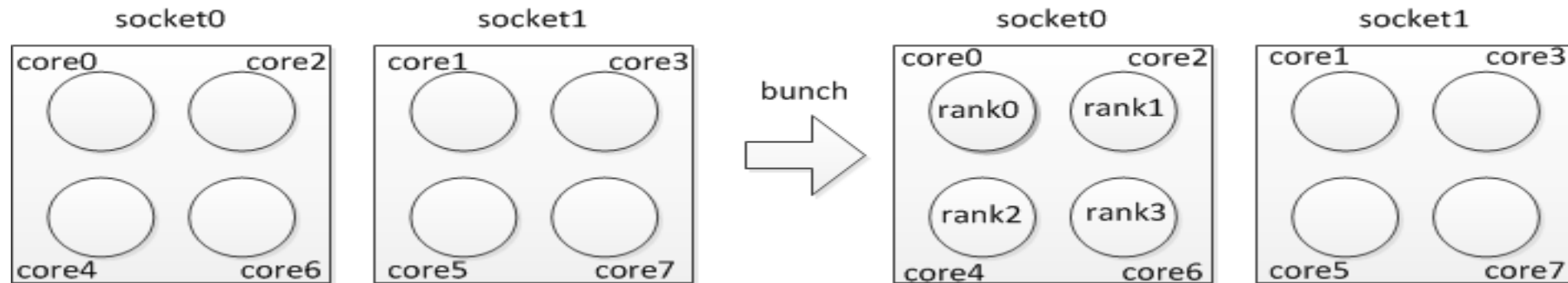
Process Mapping support in MVAPICH2



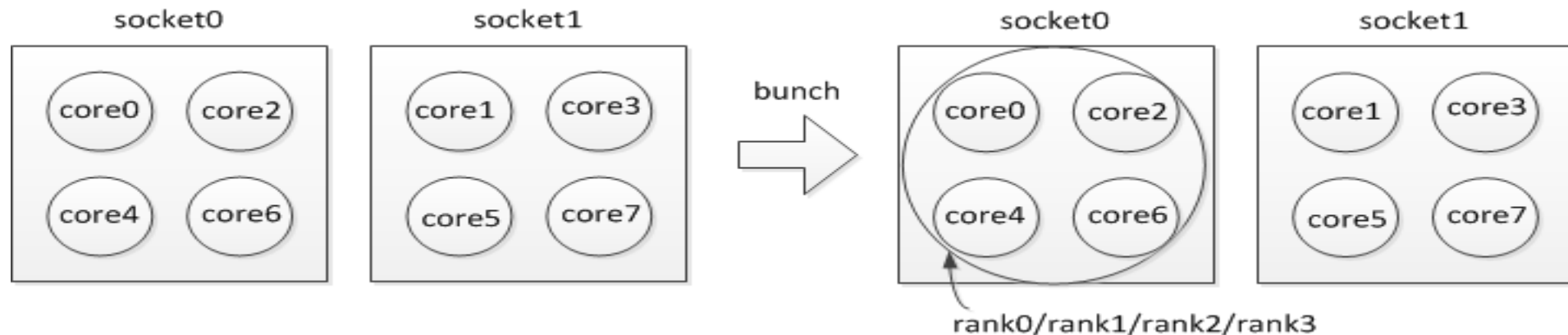
- MVAPICH2 detects processor architecture at job-launch

Preset Process-binding Policies – Bunch

- “Core” level “Bunch” mapping (Default)
 - MV2_CPU_BINDING_POLICY=bunch

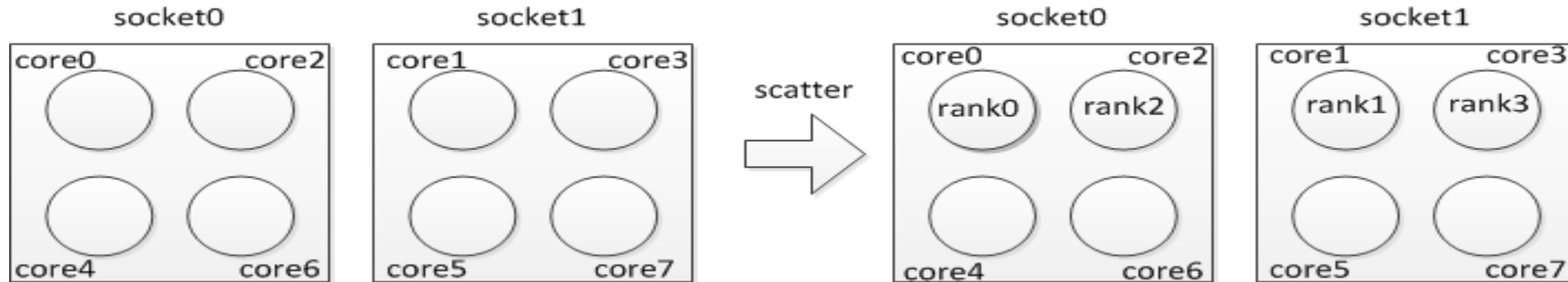


- “Socket/Numanode” level “Bunch” mapping
 - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=bunch

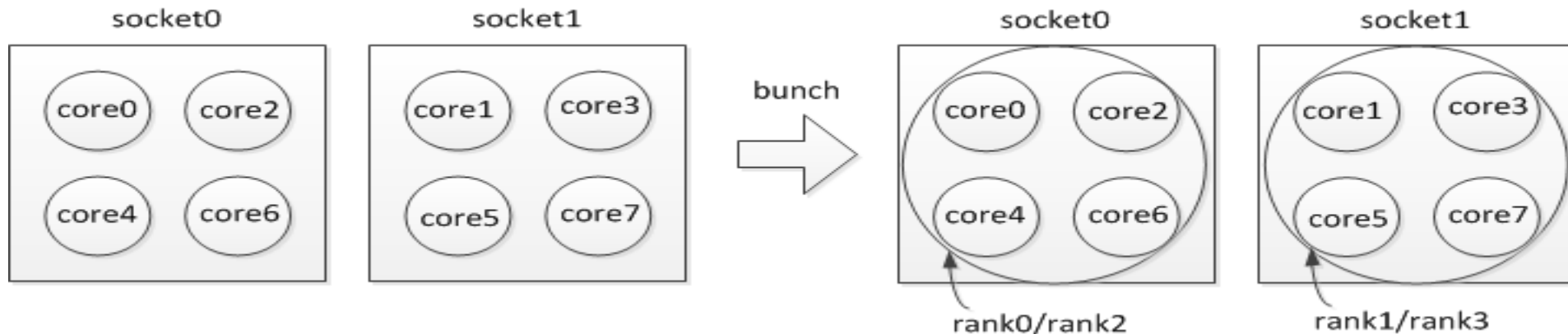


Preset Process-binding Policies – Scatter

- “Core” level “Scatter” mapping
 - MV2_CPU_BINDING_POLICY=scatter



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

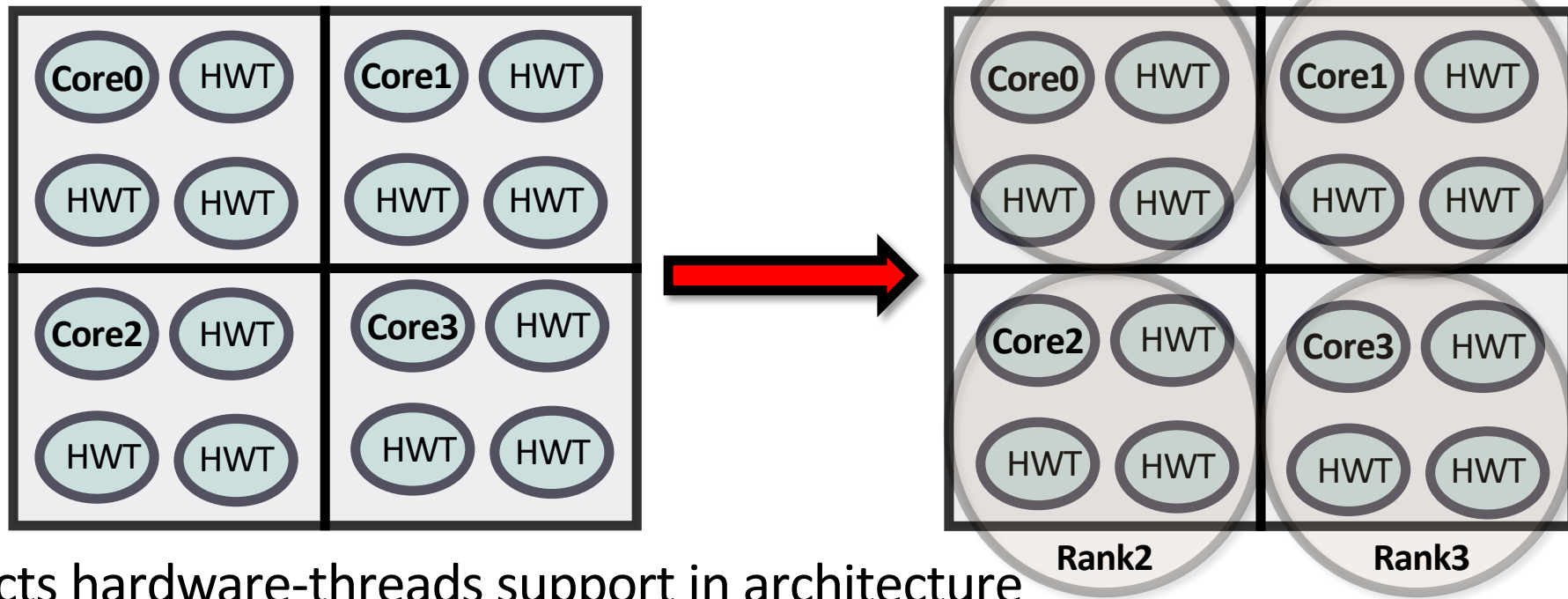


Process and thread binding policies in hybrid MPI+Threads

- A new process binding policy – “hybrid”
 - `MV2_CPU_BINDING_POLICY` = hybrid
- A new environment variable for co-locating Threads with MPI Processes
 - `MV2_THREADS_PER_PROCESS` = k
 - Automatically set to `OMP_NUM_THREADS` if OpenMP is being used
 - Provides a hint to the MPI runtime to spare resources for application threads.
- New variable for threads bindings with respect to parent process and architecture
 - `MV2_THREADS_BINDING_POLICY` = {linear | compact}
 - Linear – binds MPI ranks and OpenMP threads sequentially (one after the other)
 - Recommended to be used on non-hyper threaded systems with MPI+OpenMP
 - Compact – binds MPI rank to physical-core and locates respective OpenMP threads on hardware threads
 - Recommended to be used on multi-/many-cores e.g., KNL, POWER8, and hyper-threaded Xeon, etc.

Binding Example in Hybrid (MPI+Threads)

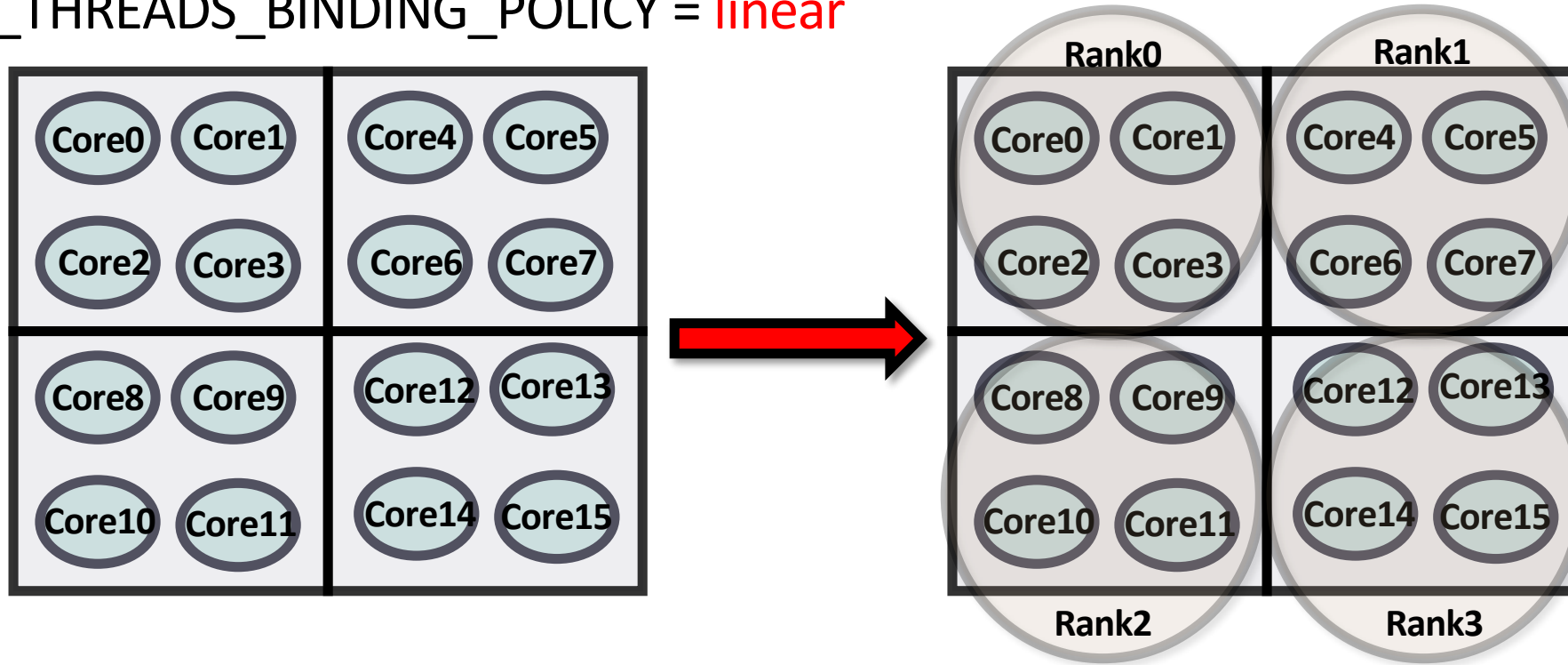
- MPI Processes = 4, OpenMP Threads per Process = 4
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_THREADS_PER_PROCESS = 4
- MV2_THREADS_BINDING_POLICY = compact



- Detects hardware-threads support in architecture
- Assigns MPI ranks to physical cores and respective OpenMP Threads to HW threads

Binding Example in Hybrid (MPI+Threads) ---- Cont'd

- MPI Processes = 4, OpenMP Threads per Process = 4
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_THREADS_PER_PROCESS = 4
- MV2_THREADS_BINDING_POLICY = **linear**



- MPI Rank-0 with its 4-OpenMP threads gets bound on Core-0 through Core-3, and so on

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-----

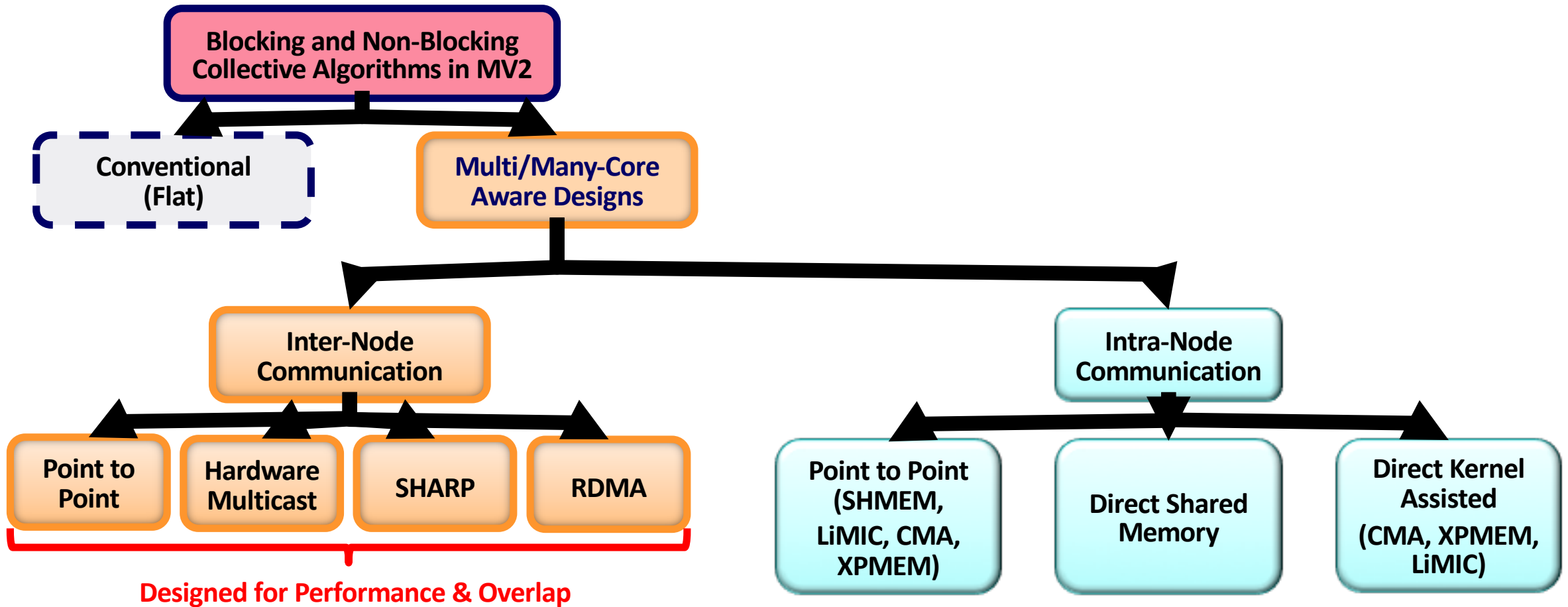
RANK:0 CPU_SET: 0

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/mvapich2-2.3rc1-userguide.html#x1-600006.5>

# Presentation Overview

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- **Collectives**
- MPI\_T Support

# Collective Communication in MVAPICH2



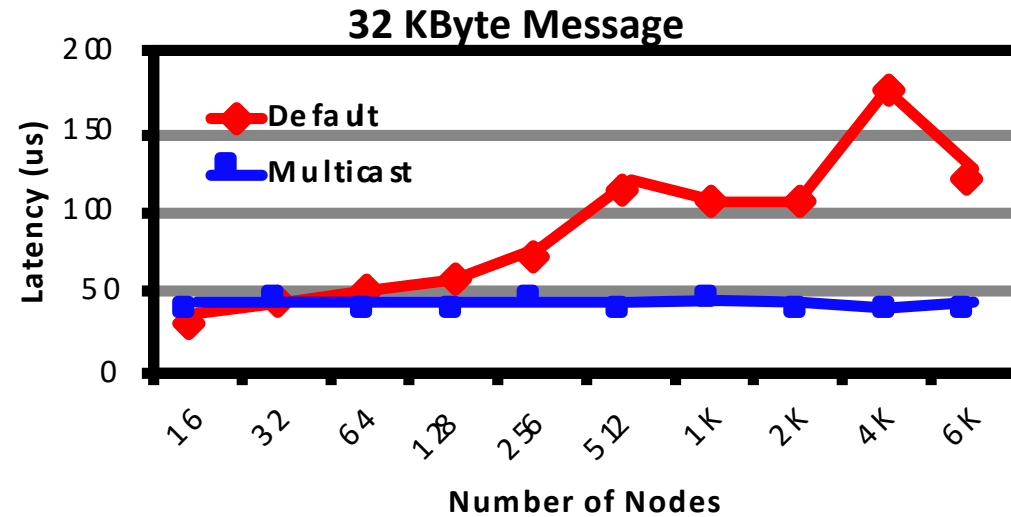
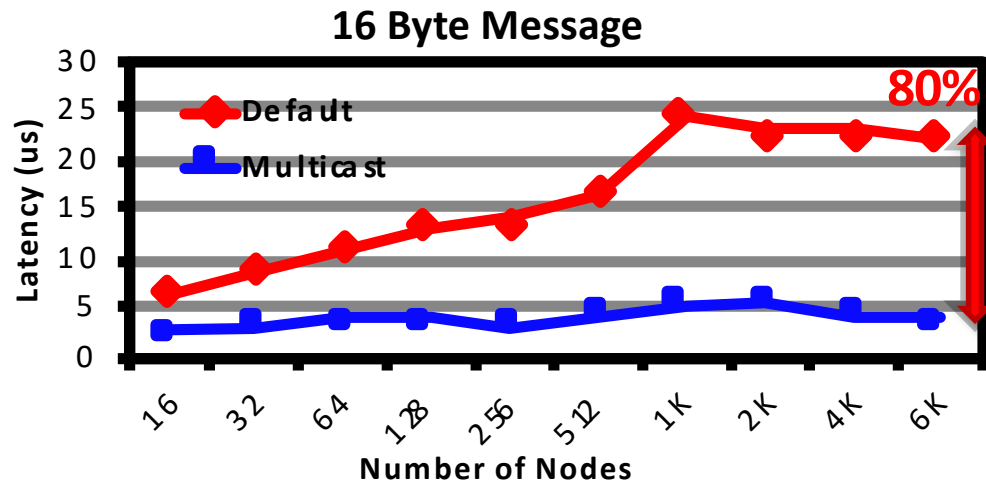
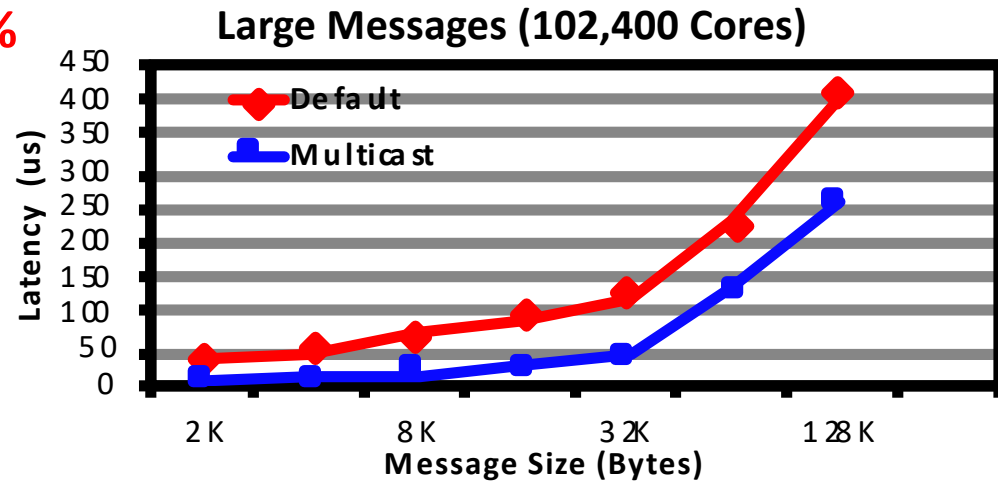
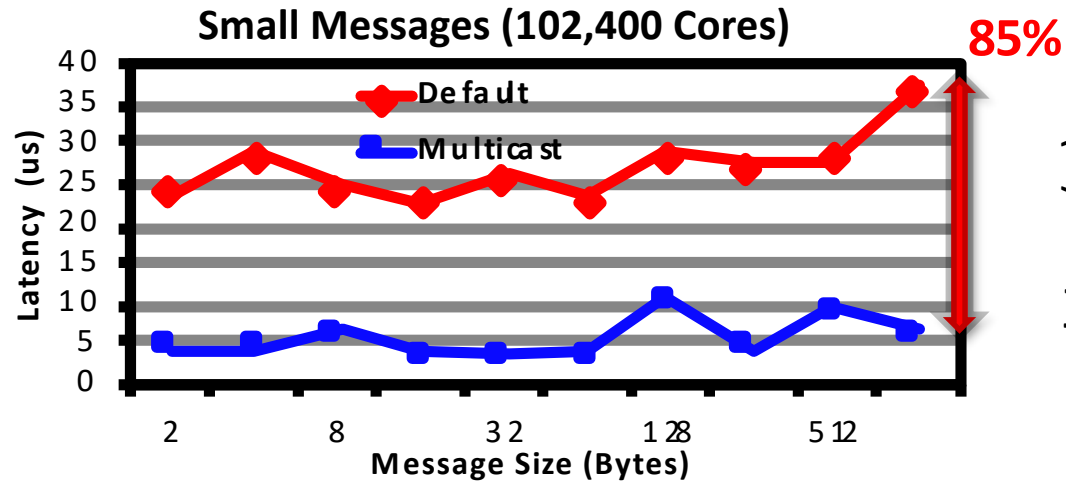
Run-time flags:

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

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

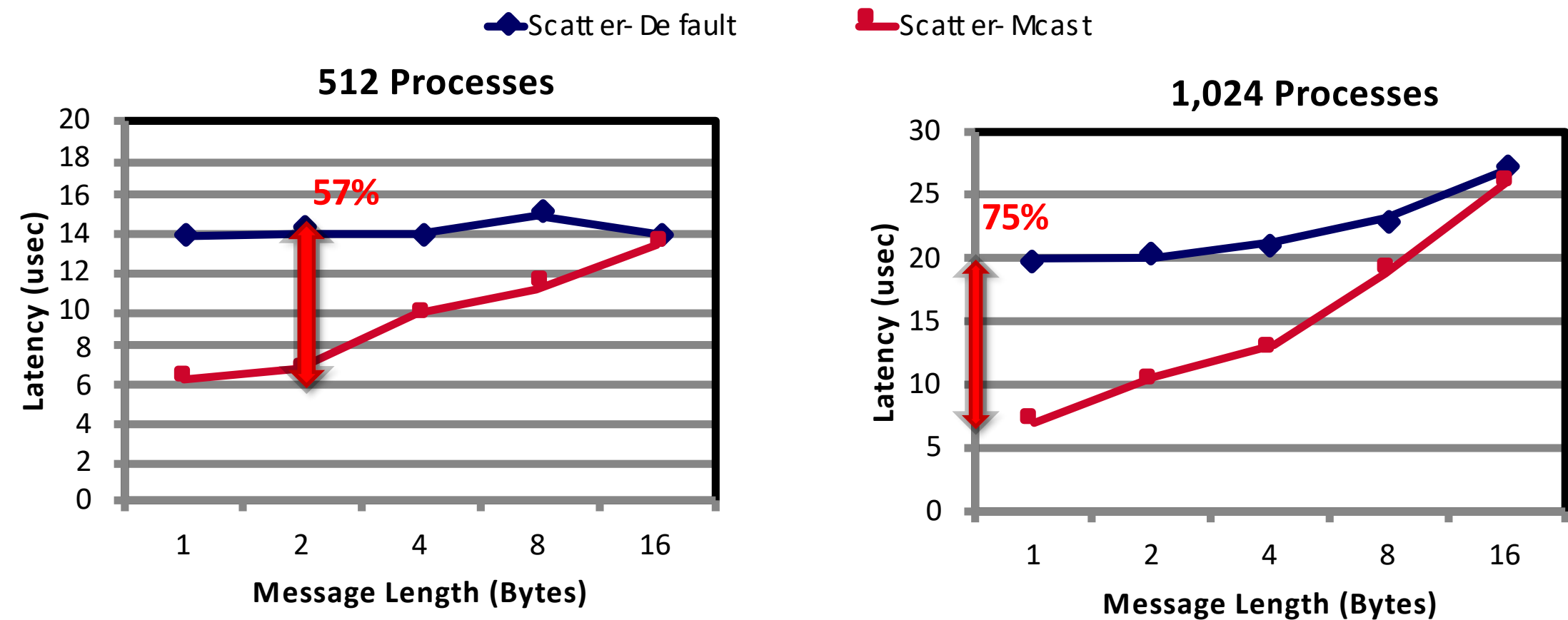
CMA-based collectives : MV2\_USE\_CMA\_COLL (Default : ON)

# 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

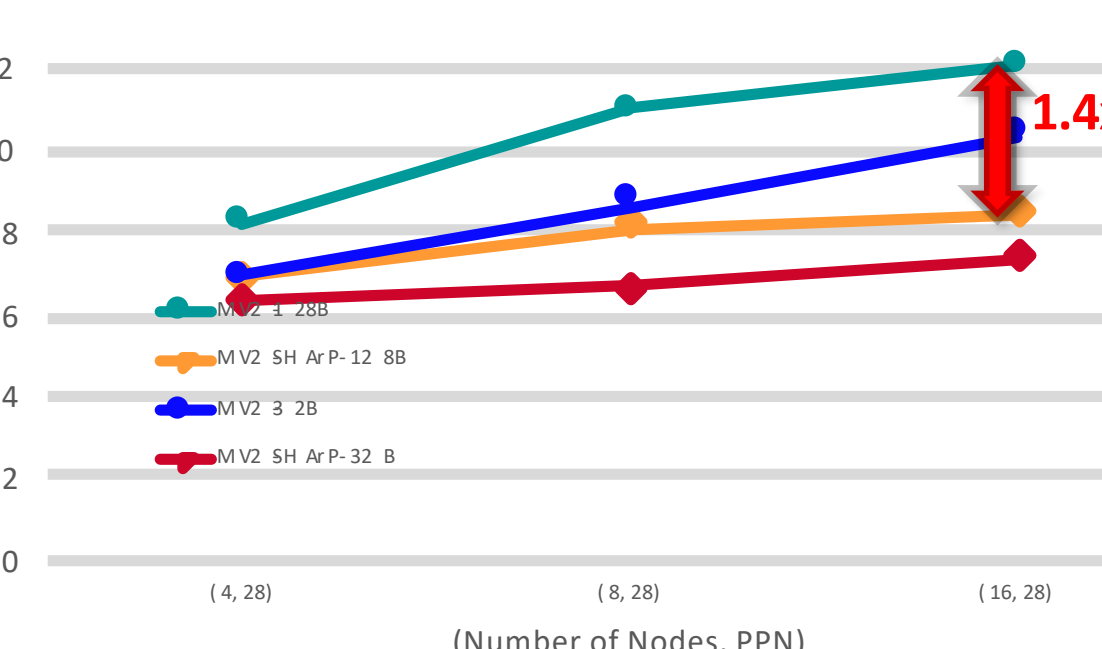
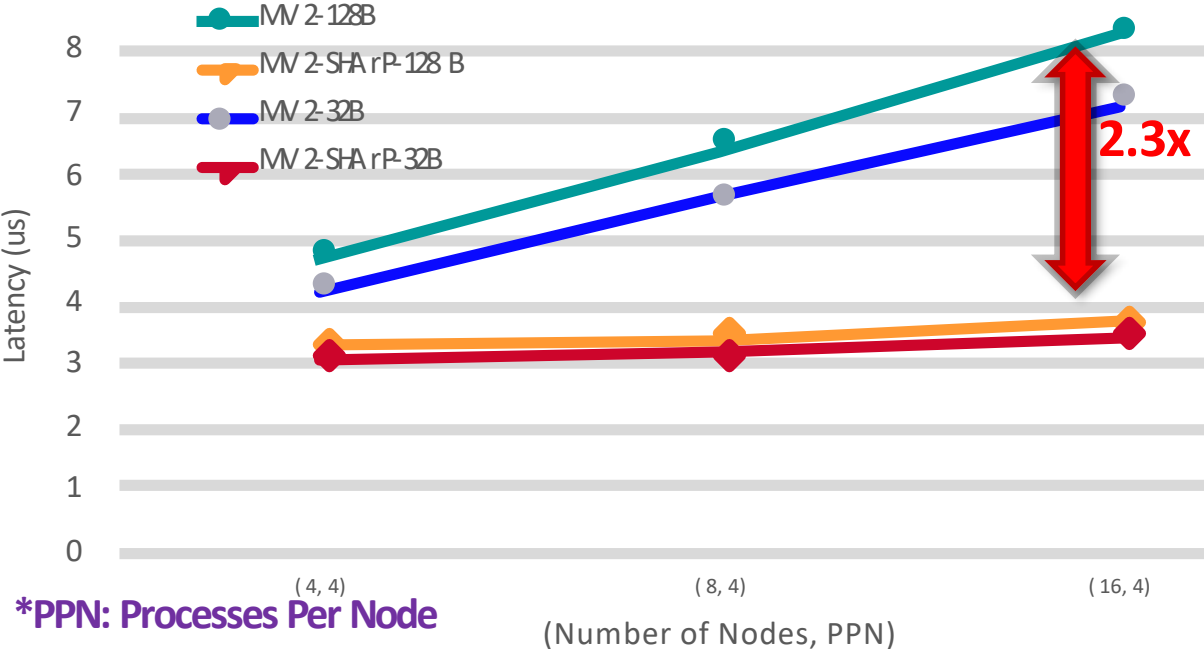
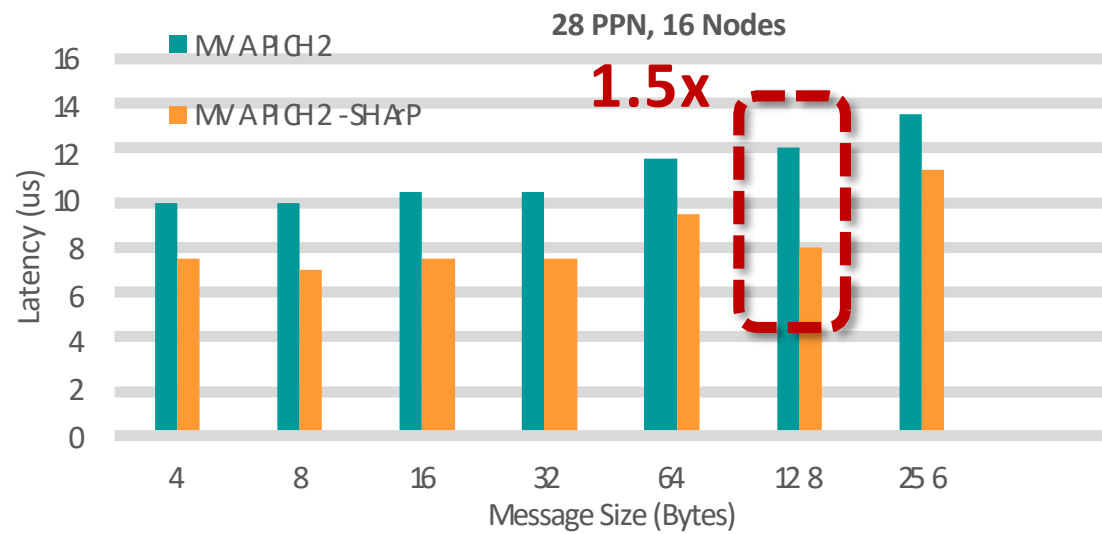
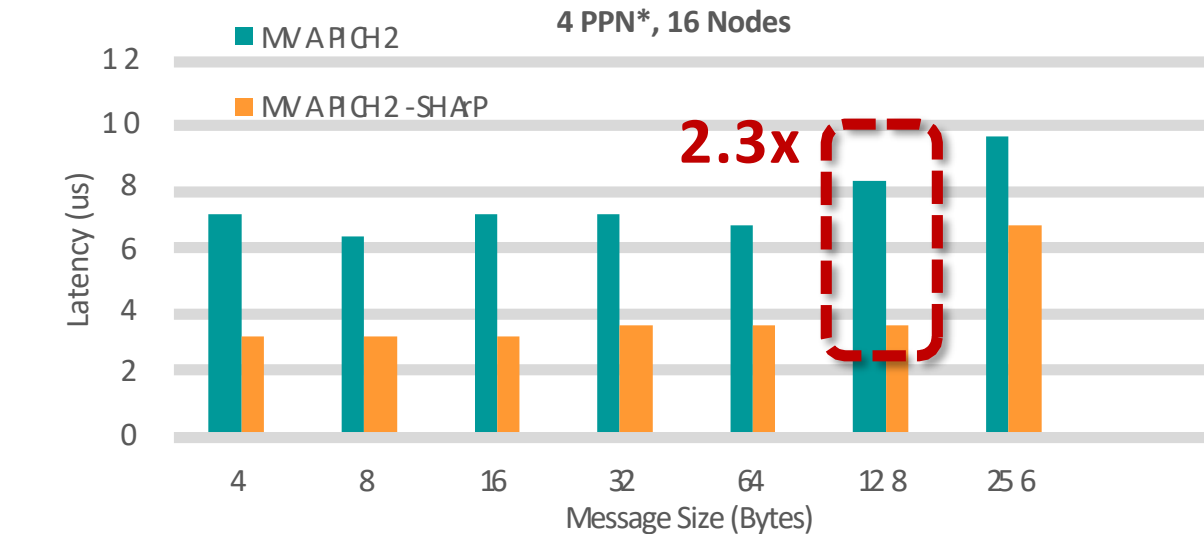


- Enabling MCAST-based designs for MPI\_Scatter improves small message up to **75%**

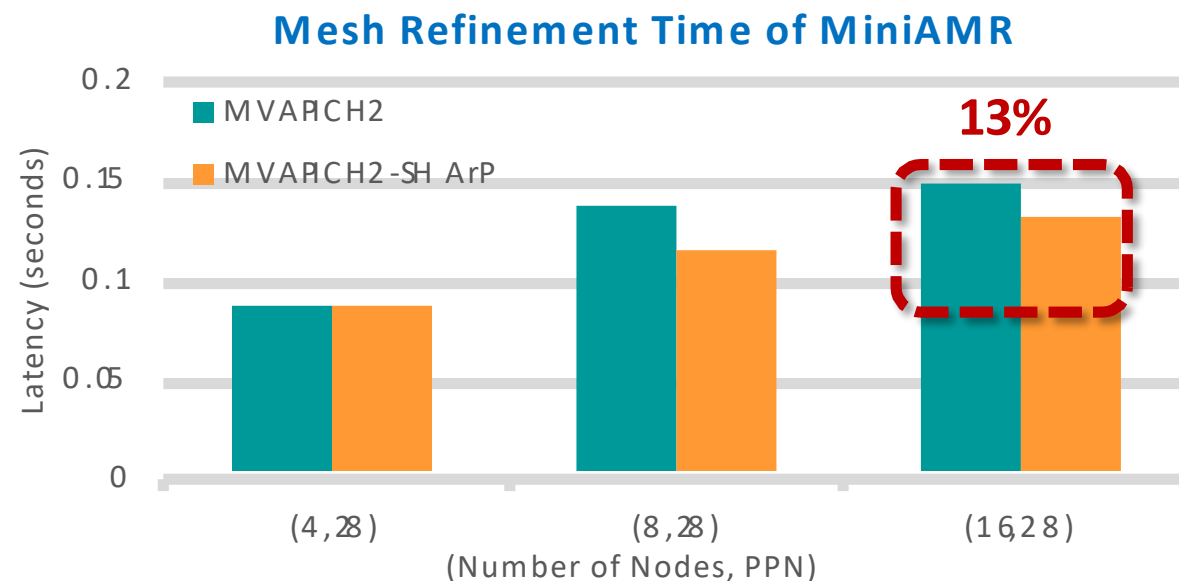
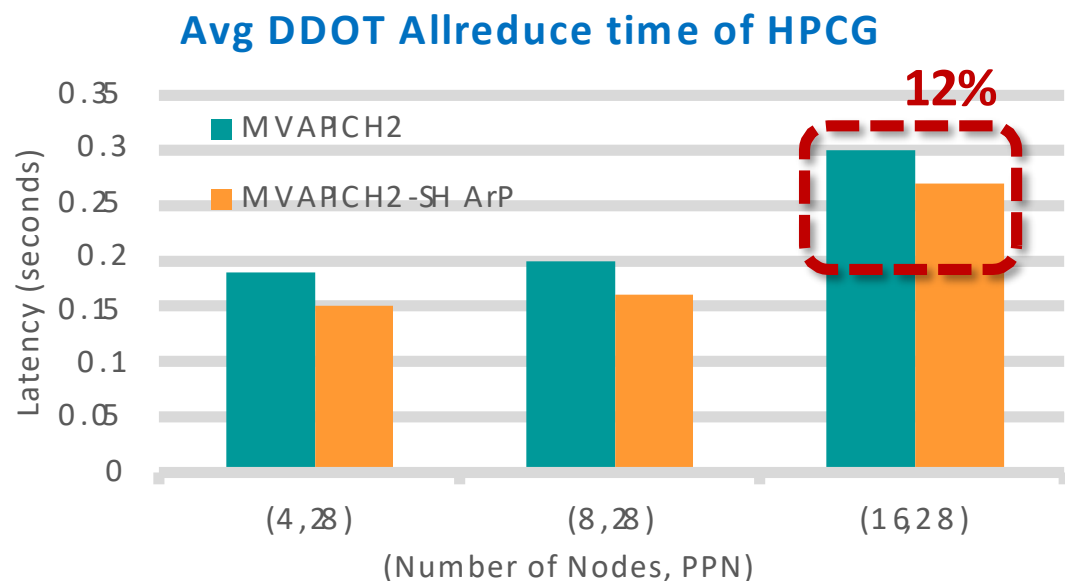
| Parameter         | Description                         | Default  |
|-------------------|-------------------------------------|----------|
| MV2_USE_MCAST = 1 | Enables hardware Multicast features | Disabled |
| --enable-mcast    | Configure flag to enable            | Enabled  |

# Advanced Allreduce Collective Designs Using SHArP

osu\_allreduce (OSU Micro Benchmark) using MVAPICH2 2.3b



# Benefits of SHARP at Application Level

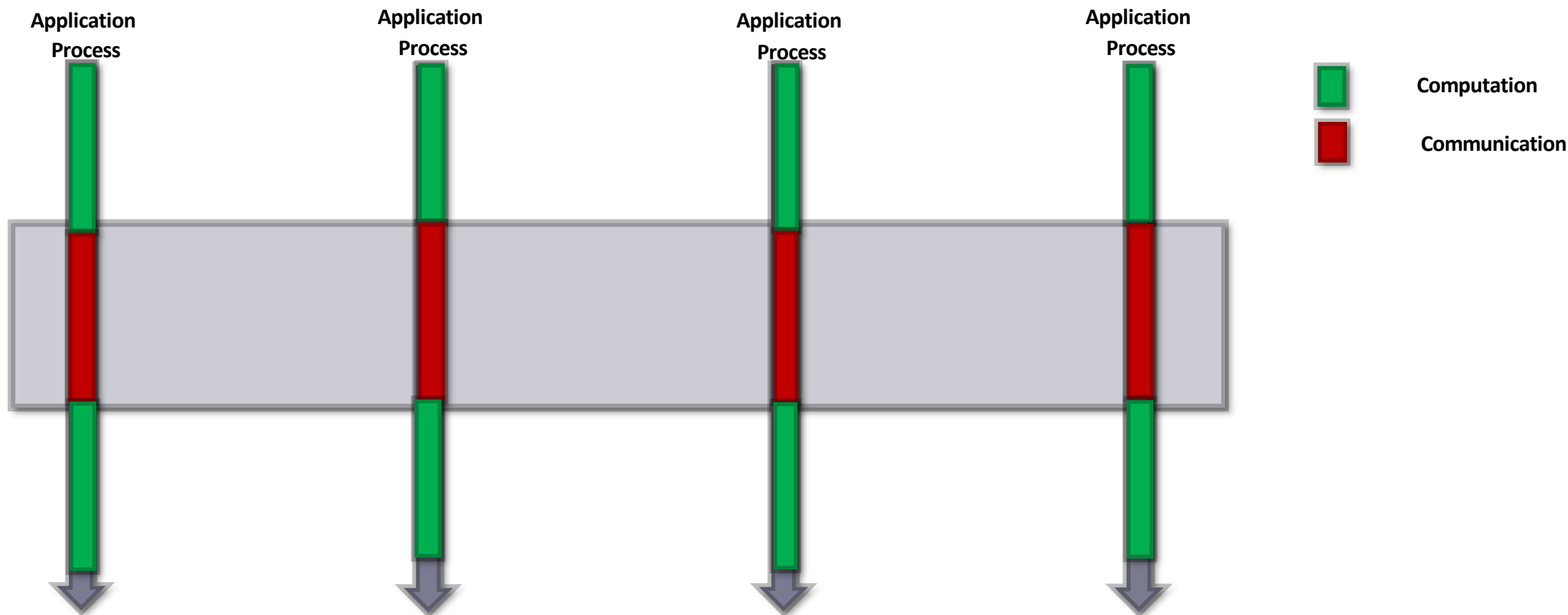


SHARP support available since MVAPICH2 2.3a

| Parameter          | Description                     | Default  |
|--------------------|---------------------------------|----------|
| MV2_ENABLE_SHARP=1 | Enables SHARP-based collectives | Disabled |
| --enable-sharp     | Configure flag to enable SHARP  | Disabled |

- Refer to **Running Collectives with Hardware based SHArP support** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3b-userguide.html#x1-990006.26>

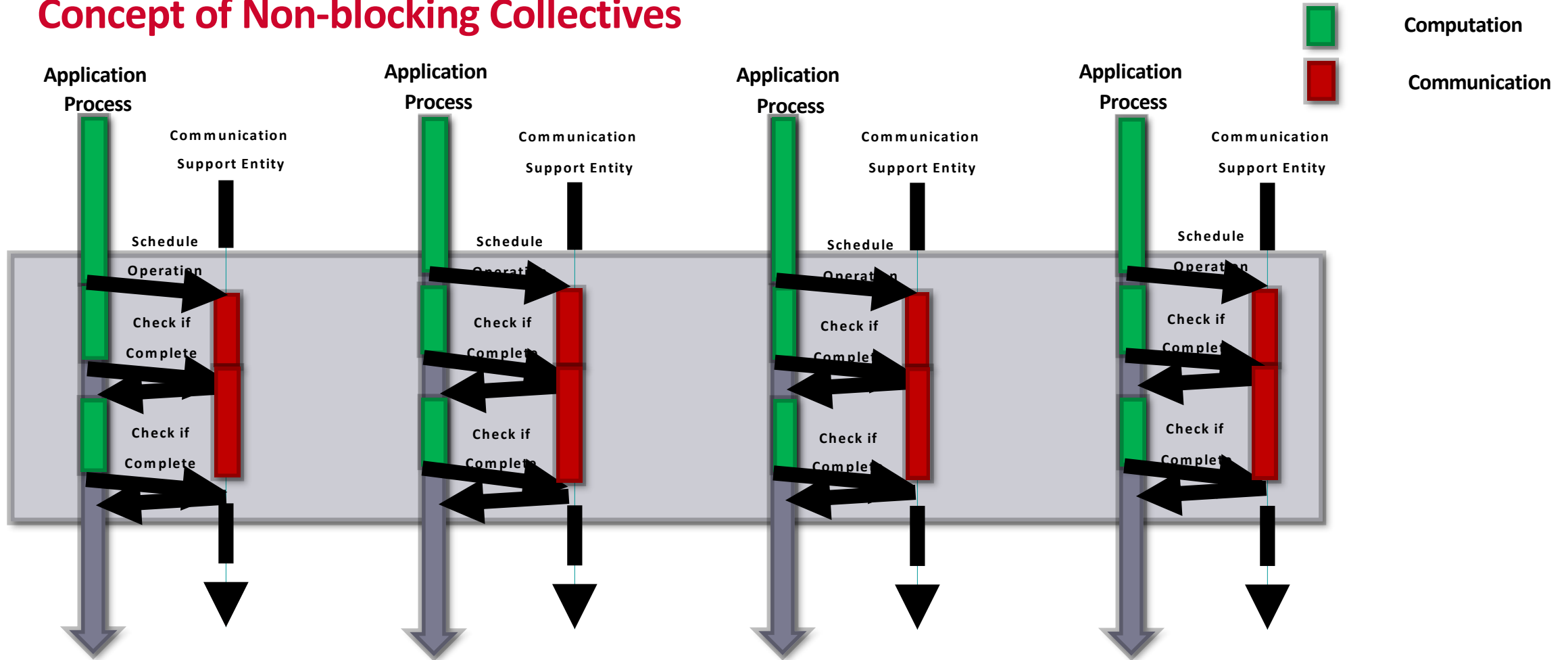
# Problems with Blocking Collective Operations



- Communication time cannot be used for compute
  - No overlap of computation and communication
  - Inefficient



# 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

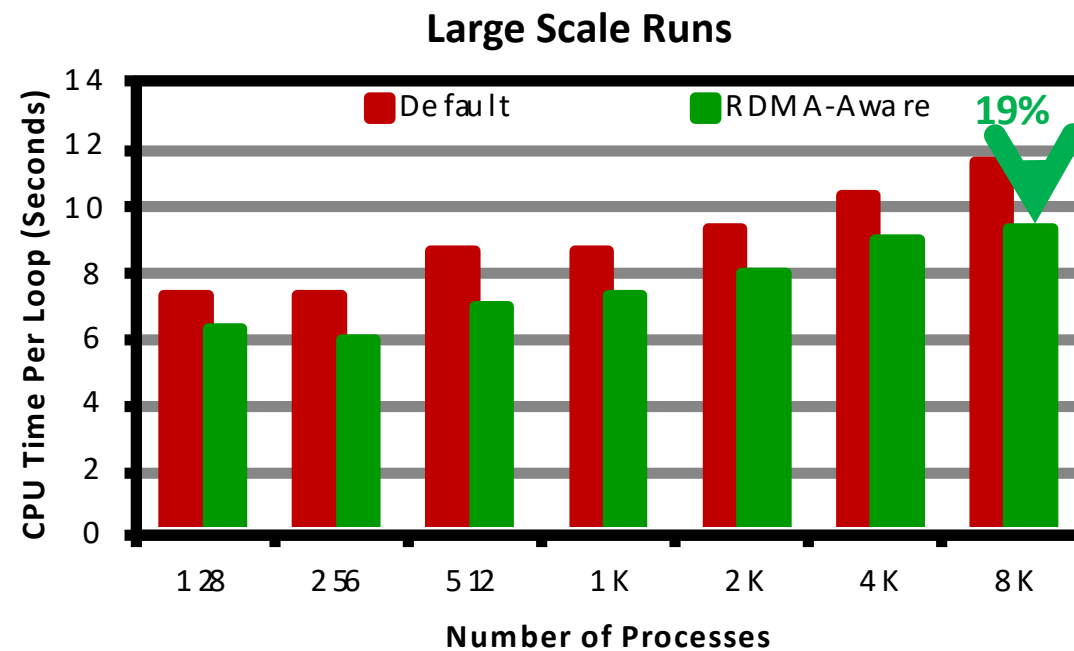
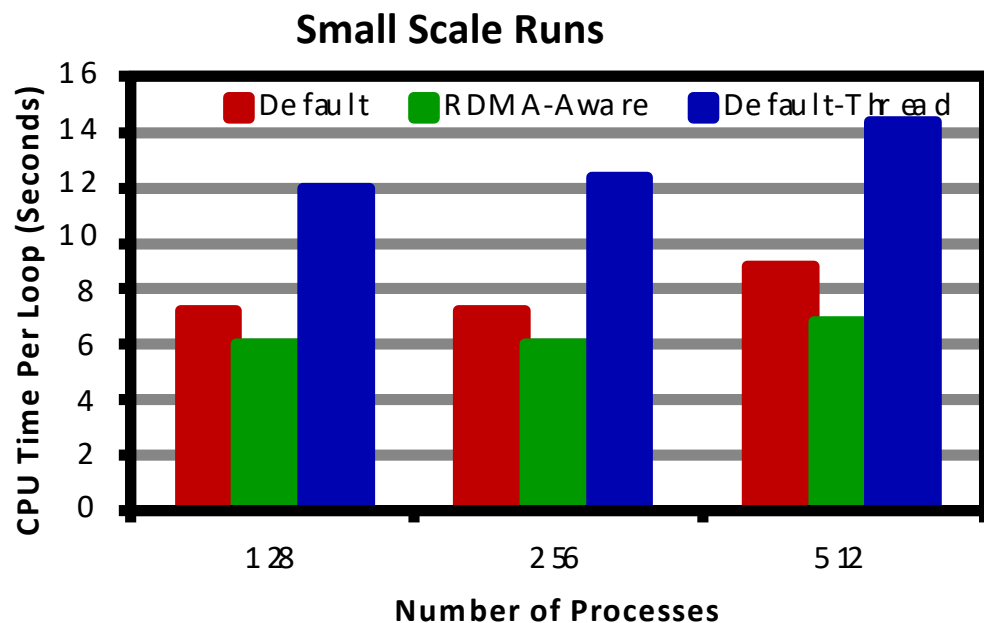
- Enables overlap of computation with communication
- Non-blocking calls do not match blocking collective calls
  - MPI may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: Optimized for latency
  - Non-blocking collectives: Optimized for overlap
- A process 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

## 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()
}
```

# P3DFFT Performance with Non-Blocking Alltoall using RDMA Primitives

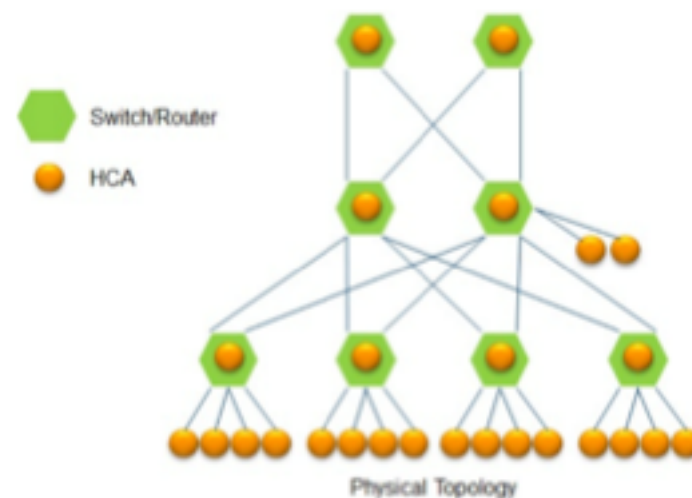


- Weak scaling experiments; problem size increases with job size
- RDMA-Aware delivers 19% improvement over Default @ 8,192 procs
- Default-Thread exhibits worst performance
  - Possibly because threads steal CPU cycles from P3DFFT
  - Do not consider for large scale experiments

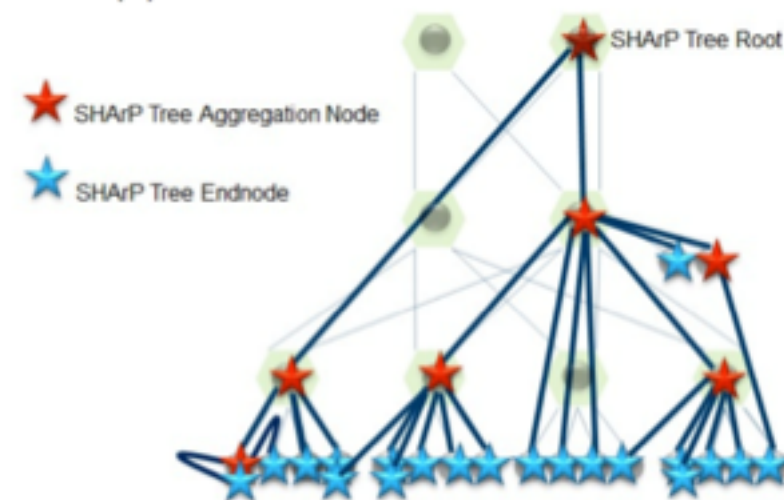
**Will be available in future**

# Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)

- Management and execution of MPI operations in the network by using SHArP
  - Manipulation of data while it is being transferred in the switch network
- SHArP provides an abstraction to realize the reduction operation
  - Defines Aggregation Nodes (AN), Aggregation Tree, and Aggregation Groups
  - AN logic is implemented as an InfiniBand Target Channel Adapter (TCA) integrated into the switch ASIC \*
  - Uses RC for communication between ANs and between AN and hosts in the Aggregation Tree \*



Physical Network Topology\*

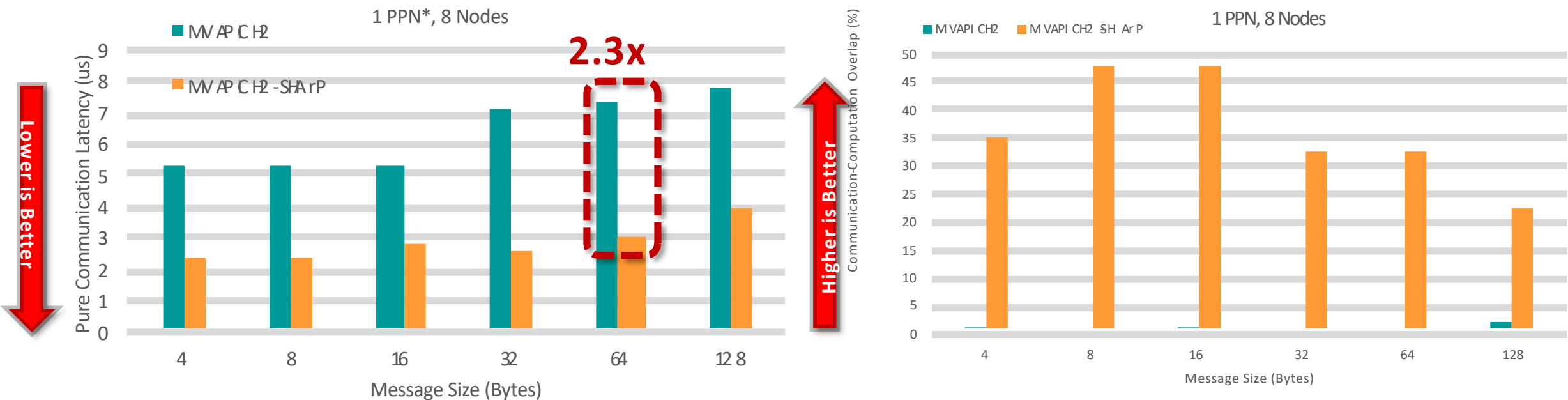


Logical SHArP Tree\*

\* Bloch et al. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction

# Evaluation of SHArP based Non Blocking Allreduce

## MPI\_Allreduce Benchmark



- Complete offload of Allreduce collective operation to Switch helps to have much higher overlap of communication and computation

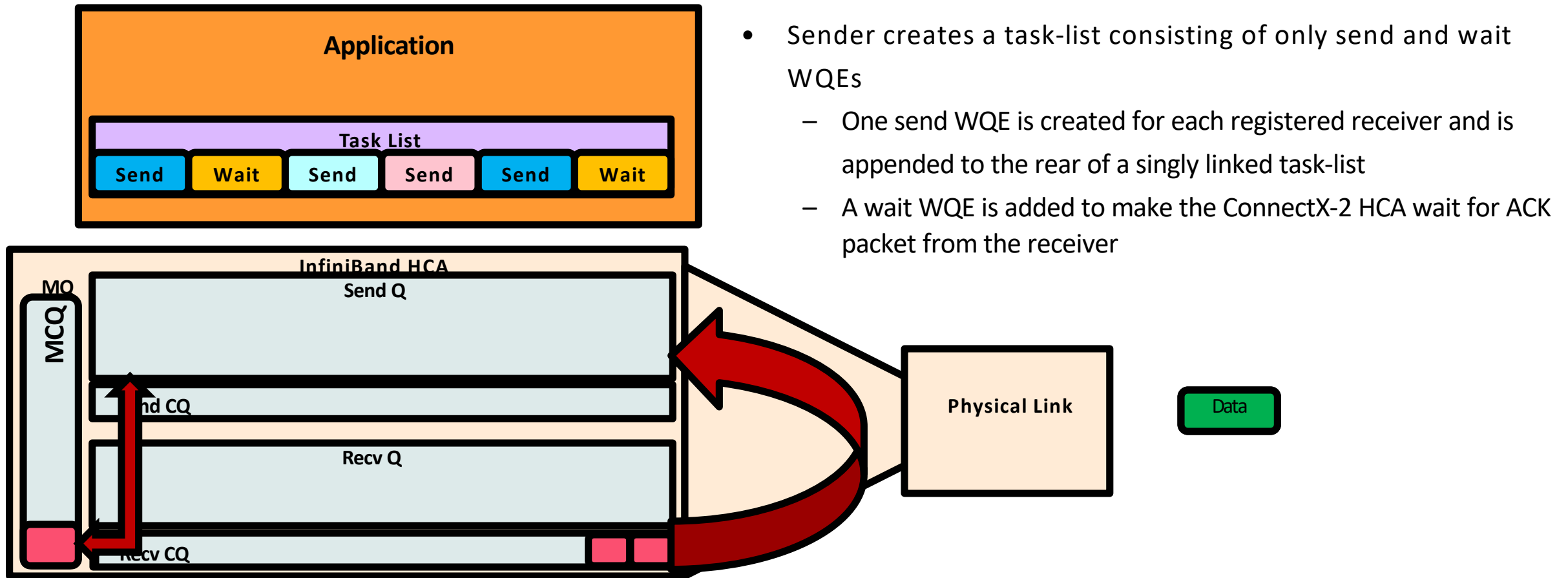
Available since MVAPICH2 2.3a

\*PPN: Processes Per Node

## Collective Offload in ConnectX-2, ConnectX-3, Connect-IB and ConnectX-4, ConnectX-5

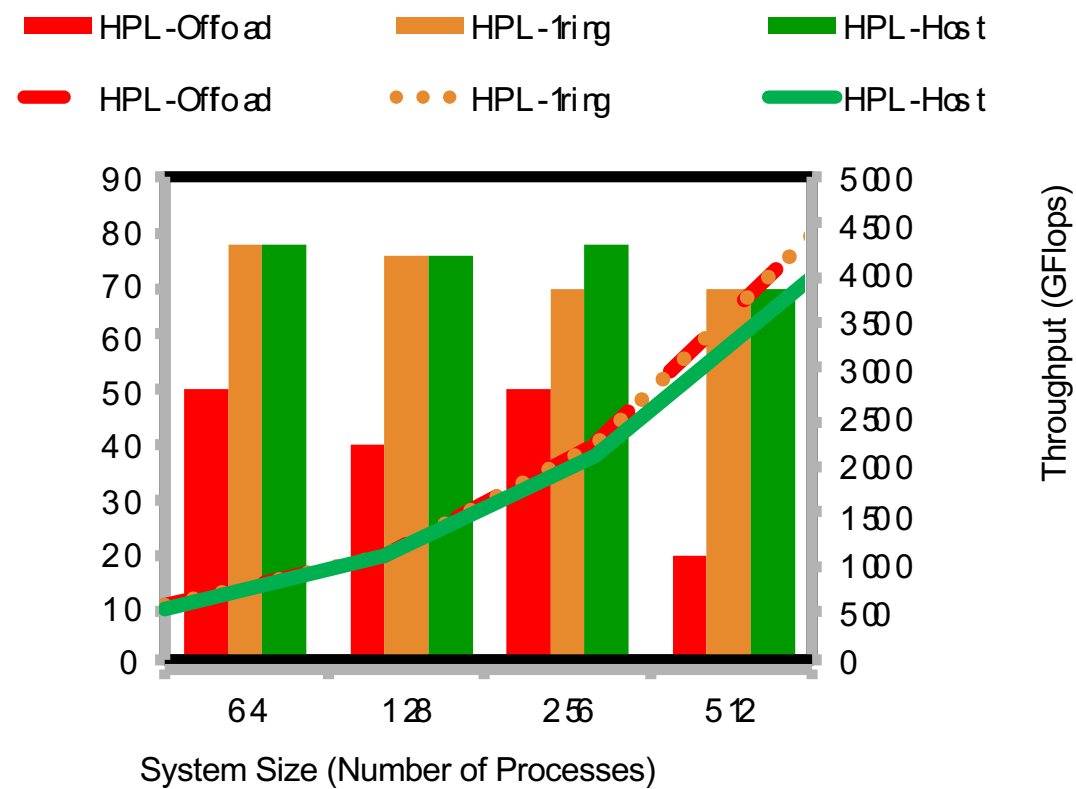
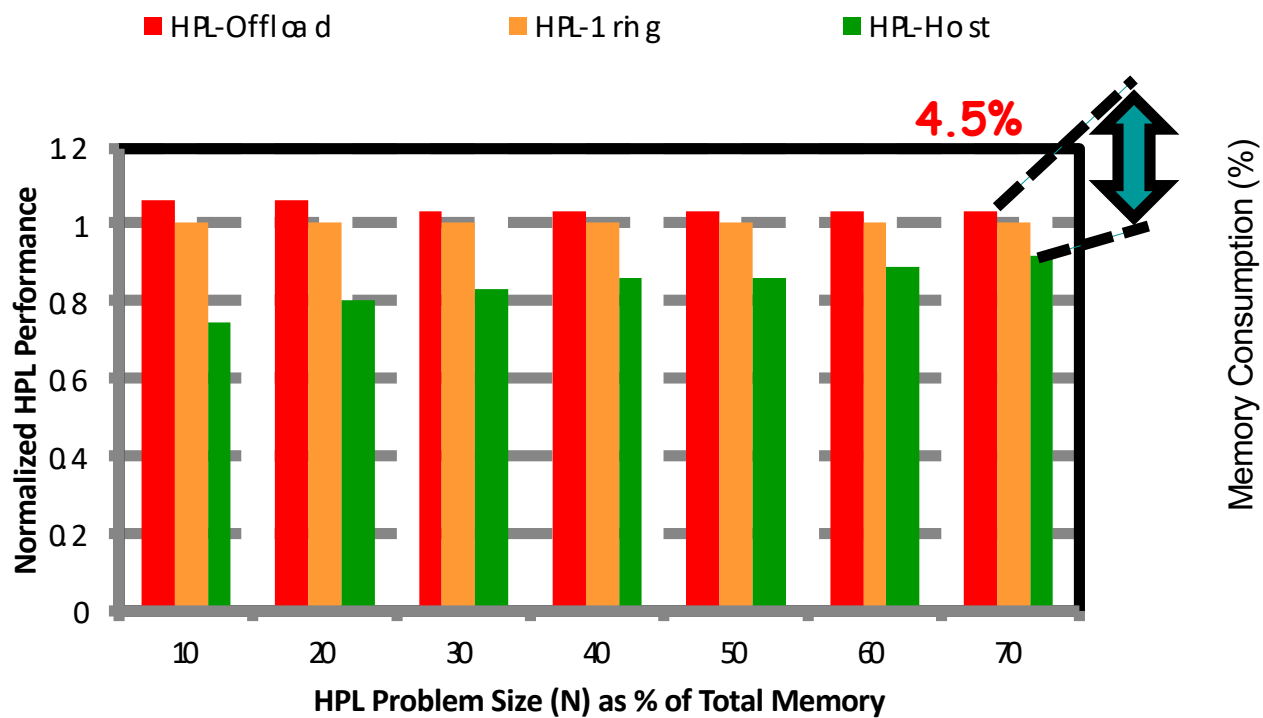
- Mellanox's ConnectX-2, ConnectX-3, ConnectIB, ConnectX-4, and ConnectX-5 adapters feature "task-list" offload interface
  - Extension to existing InfiniBand APIs
- Collective communication with 'blocking' feature is usually a scaling bottleneck
  - Matches with the need for non-blocking collective in MPI
- Accordingly MPI software stacks need to be re-designed to leverage offload in a comprehensive manner
- Can applications be modified to take advantage of non-blocking collectives and what will be the benefits?

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





# Co-designing HPL with Core-Direct and Performance Benefits



Available in MVAPICH2-X

## HPL Performance Comparison with 512 Processes

HPL-Offload consistently offers higher throughput than HPL-1ring and HPL-Host. Improves peak throughput by up to 4.5 % for large problem sizes

HPL-Offload surpasses the peak throughput of HPL-1ring with significantly smaller problem sizes and run-times!

K. Kandalla, H. Subramoni, J. Vienne, S. Pai Raikar, K. Tomko, S. Sur, and D K Panda,  
Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, (HOTI 2011)

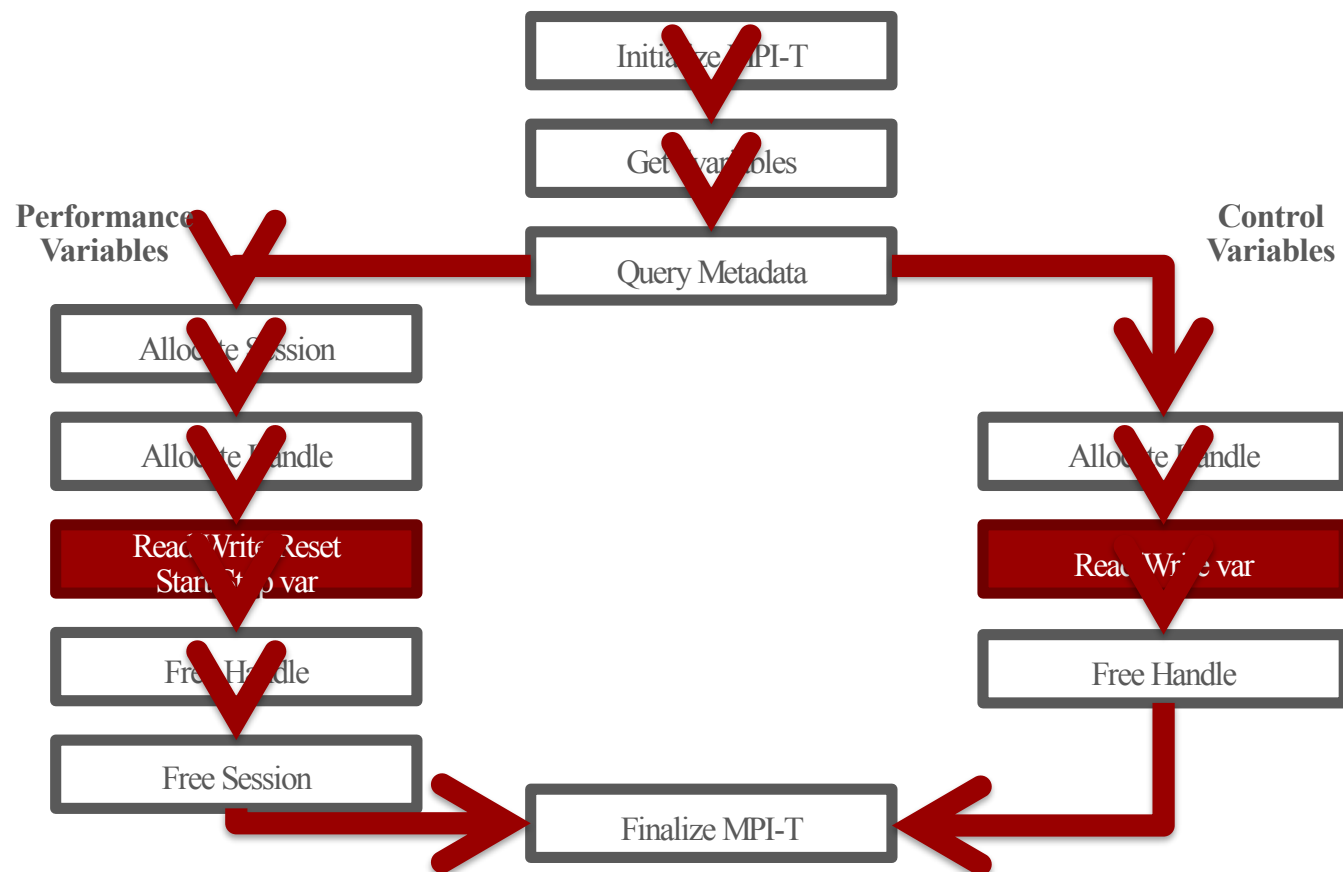
# Presentation Overview

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- **MPI\_T Support**

## MPI Tools Information Interface (MPI\_T)

- Introduced in MPI 3.0 standard to expose internals of MPI to tools and applications
- Generalized interface – no defined variables in the standard
- Variables can differ between
  - MPI implementations
  - Compilations of same MPI library (production vs debug)
  - Executions of the same application/MPI library
  - There could be no variables provided
- **Control Variables (CVARS) and Performance Variables (PVARs)**
- More about the interface: [mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf](https://mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf)

## MPI\_T usage semantics



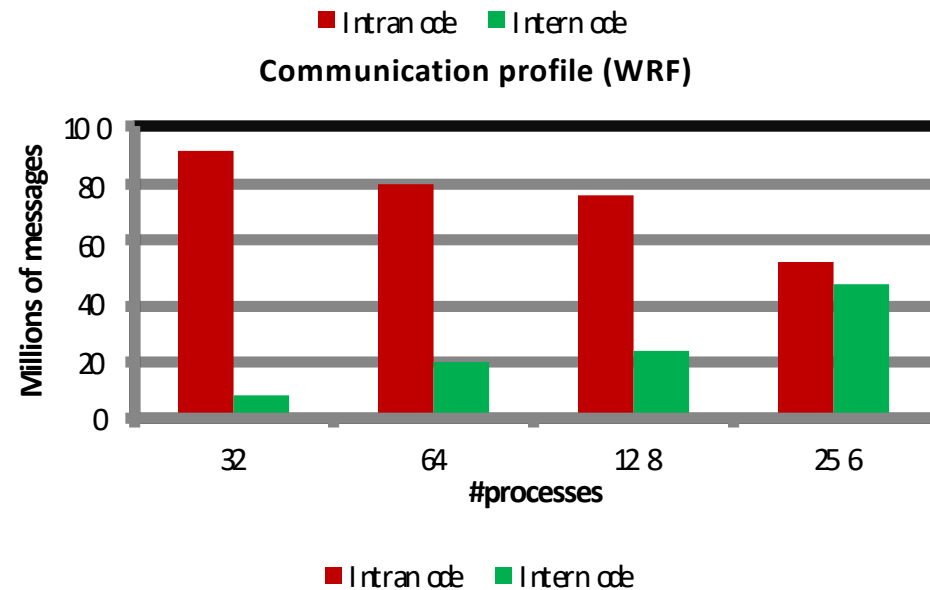
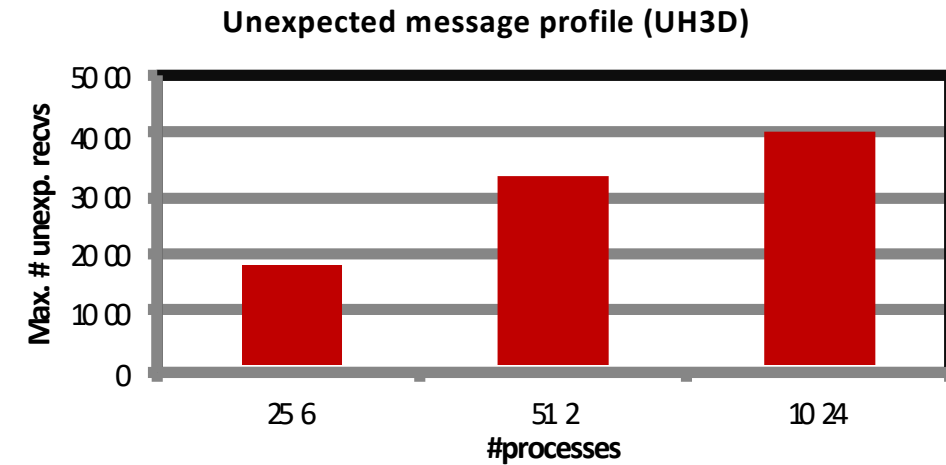
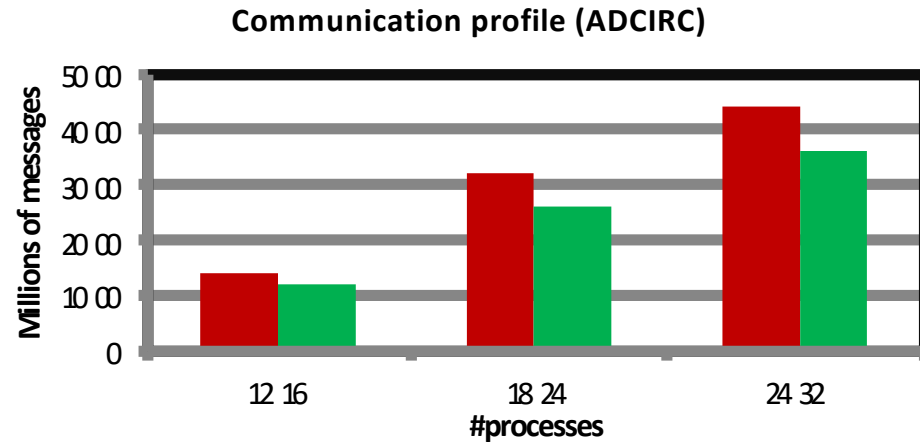
```
int MPI_T_cvar_get_info(int cvar_index, char *name, int *name_len, int *verbosity,
int MPI_T_pvar_start(MPI_T_pvar_session session, MPI_T_pvar_handle handle,
int MPI_T_pvar_handle_alloc(MPI_T_pvar_session session, int pvar_index,
int MPI_T_pvar_read_data_type(void *obj, MPI_T_pvar_handle handle, void **priviledged);
int MPI_T_pvar_reset(MPI_T_pvar_session session, MPI_T_pvar_handle handle, int count);
char *desc, int desc_len, int bind, int *scope);
```

# Co-designing Applications to use MPI-T

Example Pseudo-code: Optimizing the eager limit dynamically:

```
MPI_T_init_thread(..)
MPI_T_cvar_get_info(MV2_EAGER_THRESHOLD)
if (msg_size < MV2_EAGER_THRESHOLD + 1KB)
 MPI_T_cvar_write(MV2_EAGER_THRESHOLD, +1024)
MPI_Send(..)
MPI_T_finalize(..)
```

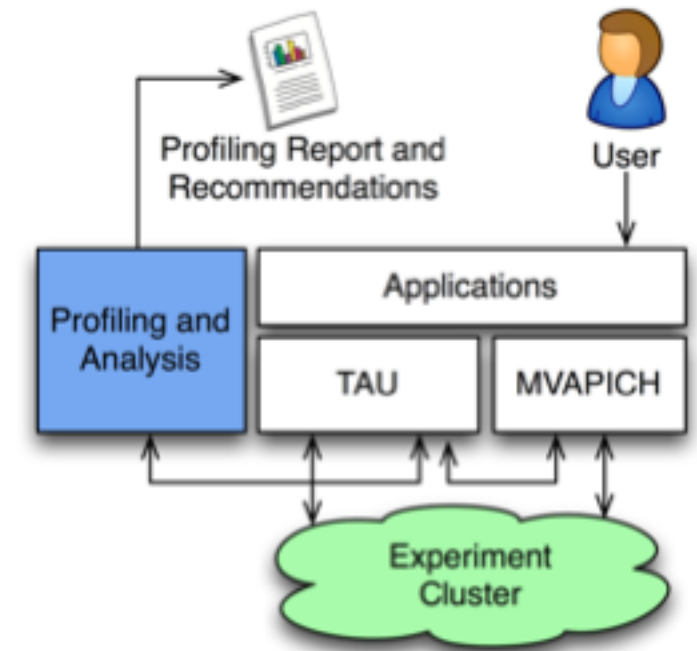
# Evaluating Applications with MPI-T



- Users can gain insights into application communication characteristics!

# Performance Engineering Applications using MVAPICH2 and TAU

- Enhance existing support for MPI\_T in MVAPICH2 to expose a richer set of performance and control variables
- Get and display MPI Performance Variables (PVARs) made available by the runtime in TAU
- Control the runtime's behavior via MPI Control Variables (CVARs)
- Introduced support for new MPI\_T based CVARs to MVAPICH2
  - MPIR\_CVAR\_MAX\_INLINE\_MSG\_SZ, MPIR\_CVAR\_VBUF\_POOL\_SIZE, MPIR\_CVAR\_VBUF\_SECONDARY\_POOL\_SIZE
- TAU enhanced with support for setting MPI\_T CVARs in a non-interactive mode for uninstrumented applications
- S. Ramesh, A. Maheo, S. Shende, A. Malony, H. Subramoni, and D. K. Panda, *MPI Performance Engineering with the MPI Tool Interface: the Integration of MVAPICH and TAU*, *EuroMPI/USA '17, Best Paper Finalist*



Available in MVAPICH2

**VBUF usage without CVAR based tuning as displayed by ParaProf**

| Name                                                                   | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamples | Total     |
|------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs) | 3,313,056 | 3,313,056 | 3,313,056 | 0         | 1          | 3,313,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                         | 320       | 320       | 320       | 0         | 1          | 320       |
| mv2_vbuf_available (Number of VBUFs available)                         | 255       | 255       | 255       | 0         | 1          | 255       |
| mv2_vbuf_freed (Number of VBUFs freed)                                 | 25,545    | 25,545    | 25,545    | 0         | 1          | 25,545    |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                 | 65        | 65        | 65        | 0         | 1          | 65        |
| mv2_vbuf_max_use (Maximum number of VBUFs used)                        | 65        | 65        | 65        | 0         | 1          | 65        |
| num_calloc_calls (Number of MPIT_calloc calls)                         | 89        | 89        | 89        | 0         | 1          | 89        |

**VBUF usage with CVAR based tuning as displayed by ParaProf**

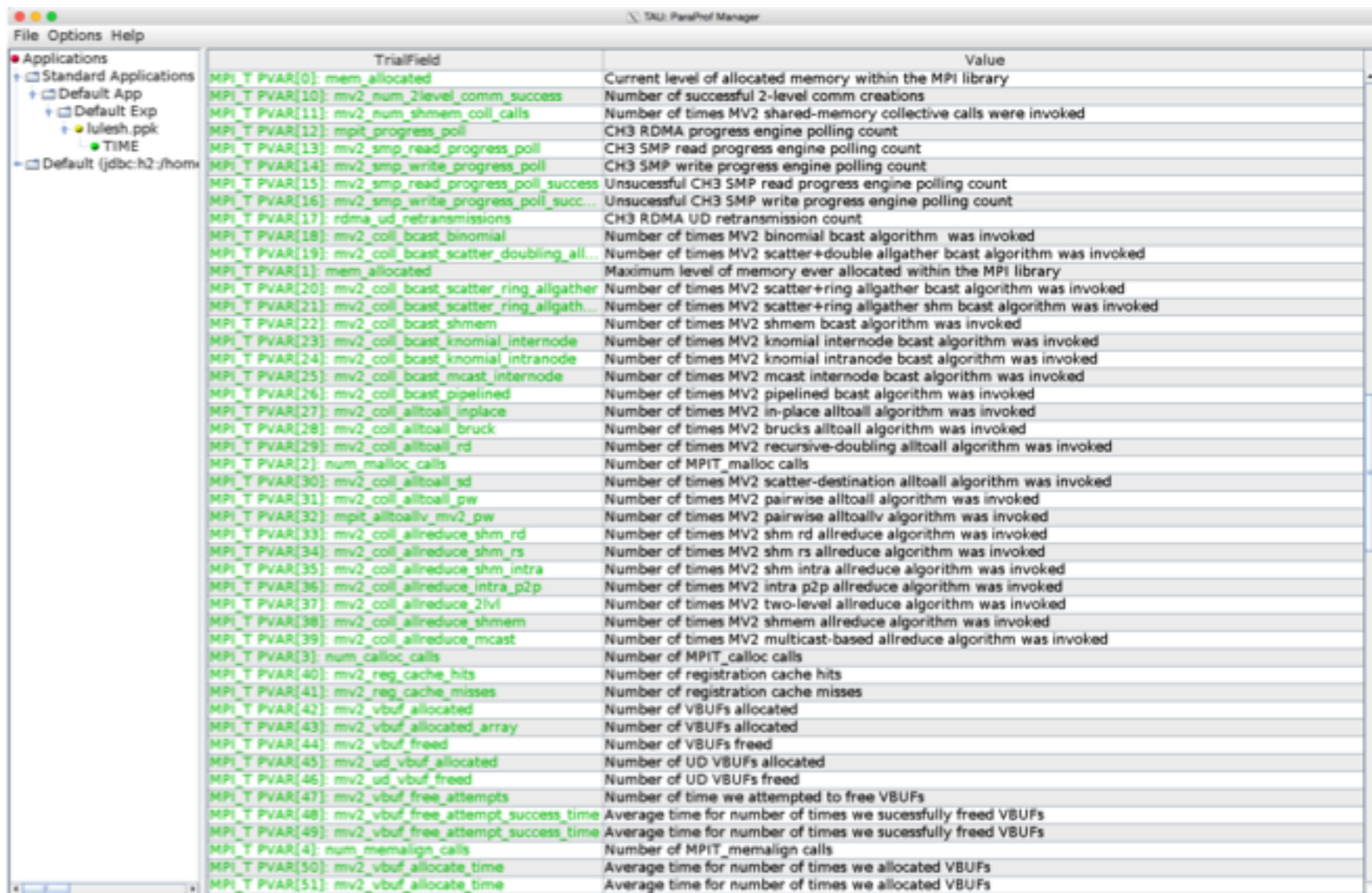
| Name                                                                   | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamples | Total     |
|------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs) | 1,815,056 | 1,815,056 | 1,815,056 | 0         | 1          | 1,815,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                         | 160       | 160       | 160       | 0         | 1          | 160       |
| mv2_vbuf_available (Number of VBUFs available)                         | 94        | 94        | 94        | 0         | 1          | 94        |
| mv2_vbuf_freed (Number of VBUFs freed)                                 | 5,479     | 5,479     | 5,479     | 0         | 1          | 5,479     |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                 | 66        | 66        | 66        | 0         | 1          | 66        |

# Enhancing MPI\_T Support

- Introduced support for new MPI\_T based CVARs to MVAPICH2
  - `MPIR_CVAR_MAX_INLINE_MSG_SZ`
    - Controls the message size up to which “inline” transmission of data is supported by MVAPICH2
  - `MPIR_CVAR_VBUF_POOL_SIZE`
    - Controls the number of internal communication buffers (VBUFs) MVAPICH2 allocates initially. Also, `MPIR_CVAR_VBUF_POOL_REDUCED_VALUE[1] ([2...n])`
  - `MPIR_CVAR_VBUF_SECONDARY_POOL_SIZE`
    - Controls the number of VBUFs MVAPICH2 allocates when there are no more free VBUFs available
  - `MPIR_CVAR_IBA_EAGER_THRESHOLD`
    - Controls the message size where MVAPICH2 switches from eager to rendezvous protocol for large messages
- TAU enhanced with support for setting MPI\_T CVARs in a non-interactive mode for uninstrumented applications



# PVARs Exposed by MVAPICH2



The screenshot shows the TAU: ParaProf Manager interface. On the left is a sidebar with a tree view containing 'Applications', 'Standard Applications', 'Default App', 'Default Exp', 'lulesh.ppk', 'TIME', and 'Default (jdb:h2:/home)'. The main window displays a table of MPI PVARs. The table has two columns: 'TrialField' and 'Value'. The 'TrialField' column lists MPI PVARs with their names and values, such as 'MPI\_T\_PVAR[0]: mem\_allocated' and 'MPI\_T\_PVAR[10]: mv2\_num\_2level\_comm\_success'. The 'Value' column provides a brief description of each PVAR, such as 'Current level of allocated memory within the MPI library' and 'Number of successful 2-level comm creations'.

| TrialField                                             | Value                                                                      |
|--------------------------------------------------------|----------------------------------------------------------------------------|
| MPI_T_PVAR[0]: mem_allocated                           | Current level of allocated memory within the MPI library                   |
| MPI_T_PVAR[10]: mv2_num_2level_comm_success            | Number of successful 2-level comm creations                                |
| MPI_T_PVAR[11]: mv2_num_shmem_coll_calls               | Number of times MV2 shared-memory collective calls were invoked            |
| MPI_T_PVAR[12]: mpit_progress_poll                     | CH3 RDMA progress engine polling count                                     |
| MPI_T_PVAR[13]: mv2_smp_read_progress_poll             | CH3 SMP read progress engine polling count                                 |
| MPI_T_PVAR[14]: mv2_smp_write_progress_poll            | CH3 SMP write progress engine polling count                                |
| MPI_T_PVAR[15]: mv2_smp_read_progress_poll_success     | Unsuccessful CH3 SMP read progress engine polling count                    |
| MPI_T_PVAR[16]: mv2_smp_write_progress_poll_success    | Unsuccessful CH3 SMP write progress engine polling count                   |
| MPI_T_PVAR[17]: rdma_ud_retransmissions                | CH3 RDMA UD retransmission count                                           |
| MPI_T_PVAR[18]: mv2_coll_bcast_binomial                | Number of times MV2 binomial bcast algorithm was invoked                   |
| MPI_T_PVAR[19]: mv2_coll_bcast_scatter_doubling_all... | Number of times MV2 scatter+double allgather bcast algorithm was invoked   |
| MPI_T_PVAR[1]: mem_allocated                           | Maximum level of memory ever allocated within the MPI library              |
| MPI_T_PVAR[20]: mv2_coll_bcast_scatter_ring_allgather  | Number of times MV2 scatter+ring allgather bcast algorithm was invoked     |
| MPI_T_PVAR[21]: mv2_coll_bcast_scatter_ring_allgath... | Number of times MV2 scatter+ring allgather shm bcast algorithm was invoked |
| MPI_T_PVAR[22]: mv2_coll_bcast_shmem                   | Number of times MV2 shm bcast algorithm was invoked                        |
| MPI_T_PVAR[23]: mv2_coll_bcast_knomial_intranode       | Number of times MV2 knomial intranode bcast algorithm was invoked          |
| MPI_T_PVAR[24]: mv2_coll_bcast_knomial_intranode       | Number of times MV2 knomial intranode bcast algorithm was invoked          |
| MPI_T_PVAR[25]: mv2_coll_bcast_mcast_intranode         | Number of times MV2 mcast intranode bcast algorithm was invoked            |
| MPI_T_PVAR[26]: mv2_coll_bcast_pipelined               | Number of times MV2 pipelined bcast algorithm was invoked                  |
| MPI_T_PVAR[27]: mv2_coll_alltoall_inplace              | Number of times MV2 in-place alltoall algorithm was invoked                |
| MPI_T_PVAR[28]: mv2_coll_alltoall_bruck                | Number of times MV2 brucks alltoall algorithm was invoked                  |
| MPI_T_PVAR[29]: mv2_coll_alltoall_rd                   | Number of times MV2 recursive-doubling alltoall algorithm was invoked      |
| MPI_T_PVAR[2]: num_malloc_calls                        | Number of MPIT_malloc calls                                                |
| MPI_T_PVAR[30]: mv2_coll_alltoall_sd                   | Number of times MV2 scatter-destination alltoall algorithm was invoked     |
| MPI_T_PVAR[31]: mv2_coll_alltoall_pw                   | Number of times MV2 pairwise alltoall algorithm was invoked                |
| MPI_T_PVAR[32]: mpit_alltoall_mv2_pw                   | Number of times MV2 pairwise alltoallv algorithm was invoked               |
| MPI_T_PVAR[33]: mv2_coll_allreduce_shm_rd              | Number of times MV2 shm rd allreduce algorithm was invoked                 |
| MPI_T_PVAR[34]: mv2_coll_allreduce_shm_rs              | Number of times MV2 shm rs allreduce algorithm was invoked                 |
| MPI_T_PVAR[35]: mv2_coll_allreduce_shm_intra           | Number of times MV2 shm intra allreduce algorithm was invoked              |
| MPI_T_PVAR[36]: mv2_coll_allreduce_intra_p2p           | Number of times MV2 intra p2p allreduce algorithm was invoked              |
| MPI_T_PVAR[37]: mv2_coll_allreduce_2lvl                | Number of times MV2 two-level allreduce algorithm was invoked              |
| MPI_T_PVAR[38]: mv2_coll_allreduce_shmem               | Number of times MV2 shm allreduce algorithm was invoked                    |
| MPI_T_PVAR[39]: mv2_coll_allreduce_mcast               | Number of times MV2 multicast-based allreduce algorithm was invoked        |
| MPI_T_PVAR[3]: num_malloc_calls                        | Number of MPIT_malloc calls                                                |
| MPI_T_PVAR[40]: mv2_reg_cache_hits                     | Number of registration cache hits                                          |
| MPI_T_PVAR[41]: mv2_reg_cache_misses                   | Number of registration cache misses                                        |
| MPI_T_PVAR[42]: mv2_vbuf_allocated                     | Number of VBUFs allocated                                                  |
| MPI_T_PVAR[43]: mv2_vbuf_allocated_array               | Number of VBUFs allocated                                                  |
| MPI_T_PVAR[44]: mv2_vbuf_freed                         | Number of VBUFs freed                                                      |
| MPI_T_PVAR[45]: mv2_ud_vbuf_allocated                  | Number of UD VBUFs allocated                                               |
| MPI_T_PVAR[46]: mv2_ud_vbuf_freed                      | Number of UD VBUFs freed                                                   |
| MPI_T_PVAR[47]: mv2_vbuf_free_attempts                 | Number of time we attempted to free VBUFs                                  |
| MPI_T_PVAR[48]: mv2_vbuf_free_attempt_success_time     | Average time for number of times we successfully freed VBUFs               |
| MPI_T_PVAR[49]: mv2_vbuf_free_attempt_success_time     | Average time for number of times we successfully freed VBUFs               |
| MPI_T_PVAR[4]: num_malloc_calls                        | Number of MPIT_malloc calls                                                |
| MPI_T_PVAR[50]: mv2_vbuf_allocate_time                 | Average time for number of times we allocated VBUFs                        |
| MPI_T_PVAR[51]: mv2_vbuf_allocate_time                 | Average time for number of times we allocated VBUFs                        |

Courtesy: The TAU Team

# CVARs Exposed by MVAPICH2

| TAU ParaProf Manager   |                                          |                                                                                                                |
|------------------------|------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| File Options Help      | TrialField                               | Value                                                                                                          |
| Applications           | Local Time                               | 2016-08-16T10:11:04-07:00                                                                                      |
| Standard Applications  | MPI Processor Name                       | cerberus.nic.uoregon.edu                                                                                       |
| Default App            | MPID_CVAR_ABORT_ON_LEAKED_HANDLES        | If true, MPI will call MPI_Abort at MPI_Finalize if any MPI object handles have been leaked. For example...    |
| Default Exp            | MPID_CVAR_ALLGATHERV_PIPELINE_MSG_SIZE   | The smallest message size that will be used for the pipelined, large-message, ring algorithm in the MPI...     |
| lulesh.ppk             | MPID_CVAR_ALLGATHER_LONG_MSG_SIZE        | For MPI_Allgather and MPI_Allgatherv, the long message algorithm will be used if the send buffer size is ...   |
| TIME                   | MPID_CVAR_ALLGATHER_SHORT_MSG_SIZE       | For MPI_Allgather and MPI_Allgatherv, the short message algorithm will be used if the send buffer size is...   |
| Default (jdbch2:/home) | MPID_CVAR_ALLREDUCE_SHORT_MSG_SIZE       | the short message algorithm will be used if the send buffer size is <= this value (in bytes)                   |
|                        | MPID_CVAR_ALLTOALL_MEDIUM_MSG_SIZE       | the medium message algorithm will be used if the per-destination message size (sendcount*size(sendtyp...       |
|                        | MPID_CVAR_ALLTOALL_SHORT_MSG_SIZE        | the short message algorithm will be used if the per-destination message size (sendcount*size(sendtype)) ...    |
|                        | MPID_CVAR_ALLTOALL_THROTTLE              | max no. of irecv/isends posted at a time in some alltoall algorithms. Setting it to 0 causes all irecv/isen... |
|                        | MPID_CVAR_ASYNC_PROGRESS                 | If set to true, MPICH will initiate an additional thread to make asynchronous progress on all communicati...   |
|                        | MPID_CVAR_BCAST_LONG_MSG_SIZE            | Let's define short messages as messages with size < MPID_CVAR_BCAST_SHORT_MSG_SIZE, and mediu...               |
|                        | MPID_CVAR_BCAST_MIN_PROCS                | Let's define short messages as messages with size < MPID_CVAR_BCAST_SHORT_MSG_SIZE, and mediu...               |
|                        | MPID_CVAR_BCAST_SHORT_MSG_SIZE           | Let's define short messages as messages with size < MPID_CVAR_BCAST_SHORT_MSG_SIZE, and mediu...               |
|                        | MPID_CVAR_CH3_EAGER_MAX_MSG_SIZE         | This cvar controls the message size at which CH3 switches from eager to rendezvous mode.                       |
|                        | MPID_CVAR_CH3_ENABLE_HCOLL               | If true, enable HCOLL collectives.                                                                             |
|                        | MPID_CVAR_CH3_INTERFACE_HOSTNAME         | If non-NULL, this cvar specifies the IP address that other processes should use when connecting to this pr...  |
|                        | MPID_CVAR_CH3_NOLOCAL                    | If true, force all processes to operate as though all processes are located on another node. For example...    |
|                        | MPID_CVAR_CH3_ODD_EVEN_CLIQUES           | If true, odd procs on a node are seen as local to each other, and even procs on a node are seen as local t...  |
|                        | MPID_CVAR_CH3_PORT_RANGE                 | The MPID_CVAR_CH3_PORT_RANGE environment variable allows you to specify the range of TCP ports ...             |
|                        | MPID_CVAR_CH3_RMA_ACC_IMMED              | Use the immediate accumulate optimization                                                                      |
|                        | MPID_CVAR_CH3_RMA_GC_NUM_COMPLETED       | Threshold for the number of completed requests the runtime finds before it stops trying to find more co...     |
|                        | MPID_CVAR_CH3_RMA_GC_NUM_TESTED          | Threshold for the number of RMA requests the runtime tests before it stops trying to check more reques...      |
|                        | MPID_CVAR_CH3_RMA_LOCK_IMMED             | Issue a request for the passive target RMA lock immediately. Default behavior is to defer the lock reque...    |
|                        | MPID_CVAR_CH3_RMA_MERGE_LOCK_OP_UNLOCK   | Enable/disable an optimization that merges lock, op, and unlock messages, for single-operation passive ta...   |
|                        | MPID_CVAR_CH3_RMA_NREQUEST_NEW_THRESHOLD | Threshold for the number of new requests since the last attempt to complete pending requests. Higher ...       |
|                        | MPID_CVAR_CH3_RMA_NREQUEST_THRESHOLD     | Threshold at which the RMA implementation attempts to complete requests while completing RMA oper...           |
|                        | MPID_CVAR_CHOP_ERROR_STACK               | If >0, truncate error stack output lines this many characters wide. If 0, do not truncate, and if <0 use a ... |
|                        | MPID_CVAR_COLL_ALIAS_CHECK               | Enable checking of aliasing in collective operations                                                           |
|                        | MPID_CVAR_COMM_SPLIT_USE_QSORT           | Use qsort(3) in the implementation of MPI_Comm_split instead of bubble sort.                                   |
|                        | MPID_CVAR_CTXID_EAGER_SIZE               | The MPID_CVAR_CTXID_EAGER_SIZE environment variable allows you to specify how many words in th...              |
|                        | MPID_CVAR_DEBUG_HOLD                     | If true, causes processes to wait in MPI_Init and MPI_Initthread for a debugger to be attached. Once the ...   |
|                        | MPID_CVAR_DEFAULT_THREAD_LEVEL           | Sets the default thread level to use when using MPI_INIT.                                                      |
|                        | MPID_CVAR_DUMP_PROVIDERS                 | If true, dump provider information at init                                                                     |
|                        | MPID_CVAR_ENABLE_COLL_FT_RET             | DEPRECATED! Will be removed in MPICH-3.2 Collectives called on a communicator with a failed process...         |
|                        | MPID_CVAR_ENABLE_SMP_ALLREDUCE           | Enable SMP aware allreduce.                                                                                    |
|                        | MPID_CVAR_ENABLE_SMP_BARRIER             | Enable SMP aware barrier.                                                                                      |
|                        | MPID_CVAR_ENABLE_SMP_BCAST               | Enable SMP aware broadcast (See also: MPID_CVAR_MAX_SMP_BCAST_MSG_SIZE)                                        |
|                        | MPID_CVAR_ENABLE_SMP_COLLECTIVES         | Enable SMP aware collective communication.                                                                     |
|                        | MPID_CVAR_ENABLE_SMP_REDUCE              | Enable SMP aware reduce.                                                                                       |
|                        | MPID_CVAR_ERROR_CHECKING                 | If true, perform checks for errors, typically to verify valid inputs to MPI routines. Only effective when M... |
|                        | MPID_CVAR_GATHERV_INTER_SSEND_MIN_PROCS  | Use Ssend (synchronous send) for intercommunicator MPI_Gatherv if the "group B" size is >= this value...       |
|                        | MPID_CVAR_GATHER_INTER_SHORT_MSG_SIZE    | use the short message algorithm for intercommunicator MPI_Gather if the send buffer size is < this value...    |
|                        | MPID_CVAR_GATHER_VSMALL_MSG_SIZE         | use a temporary buffer for intracommunicator MPI_Gather if the send buffer size is < this value (in bytes...   |
|                        | MPID_CVAR_IBA_EAGER_THRESHOLD            | 0 (old) -> 204800 (new). This set the switch point between eager and rendezvous protocol                       |
|                        | MPID_CVAR_MAX_INLINE_SIZE                | This set the maximum inline size for data transfer                                                             |
|                        | MPID_CVAR_MAX_SMP_ALLREDUCE_MSG_SIZE     | Maximum message size for which SMP-aware allreduce is used. A value of '0' uses SMP-aware allreduce ...        |

Courtesy: The TAU Team

# Using MVAPICH2 and TAU

- To set CVARs or read PVARs using TAU for an uninstrumented binary:


```
% export TAU_TRACK_MPI_T_PVARS=1
% export TAU_MPI_T_CVAR_METRICS=
 MPIR_CVAR_VBUF_POOL_REDUCED_VALUE[1],
 MPIR_CVAR_IBA_EAGER_THRESHOLD
% export TAU_MPI_T_CVAR_VALUES=32,64000
% export PATH=/path/to/tau/x86_64/bin:$PATH
% mpirun -np 1024 tau_exec -T mvapich2,mpit ./a.out
% paraprof
```

Courtesy: The TAU Team



# VBUF usage without CVARs

TAU: ParaProf: Context Events for: node 0 - mpit\_withoutcvar\_bt.C.1k.ppk

| Name  | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamples | Total     |
|----------------------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs)                 | 3,313,056 | 3,313,056 | 3,313,056 | 0         | 1          | 3,313,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                                         | 320       | 320       | 320       | 0         | 1          | 320       |
| mv2_vbuf_available (Number of VBUFs available)                                         | 255       | 255       | 255       | 0         | 1          | 255       |
| mv2_vbuf_freed (Number of VBUFs freed)                                                 | 25,545    | 25,545    | 25,545    | 0         | 1          | 25,545    |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                                 | 65        | 65        | 65        | 0         | 1          | 65        |
| mv2_vbuf_max_use (Maximum number of VBUFs used)                                        | 65        | 65        | 65        | 0         | 1          | 65        |
| num_calloc_calls (Number of MPIT_calloc calls)                                         | 89        | 89        | 89        | 0         | 1          | 89        |
| num_free_calls (Number of MPIT_free calls)                                             | 47,801    | 47,801    | 47,801    | 0         | 1          | 47,801    |
| num_malloc_calls (Number of MPIT_malloc calls)                                         | 49,258    | 49,258    | 49,258    | 0         | 1          | 49,258    |
| num_memalign_calls (Number of MPIT_memalign calls)                                     | 34        | 34        | 34        | 0         | 1          | 34        |
| num_memalign_free_calls (Number of MPIT_memalign_free calls)                           | 0         | 0         | 0         | 0         | 0          | 0         |

Courtesy: The TAU Team

# VBUF usage with CVARs

TAU: ParaProf: Context Events for: node 0 - bt-mz.E.vbuf\_pool\_16.1k.ppk

| Name                                                                   | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamp... | Total     |
|------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs) | 1,815,056 | 1,815,056 | 1,815,056 | 0         | 1          | 1,815,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                         | 160       | 160       | 160       | 0         | 1          | 160       |
| mv2_vbuf_available (Number of VBUFs available)                         | 94        | 94        | 94        | 0         | 1          | 94        |
| mv2_vbuf_freed (Number of VBUFs freed)                                 | 5,479     | 5,479     | 5,479     | 0         | 1          | 5,479     |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                 | 66        | 66        | 66        | 0         | 1          | 66        |
| mv2_vbuf_max_use (Maximum number of VBUFs used)                        | 66        | 66        | 66        | 0         | 1          | 66        |
| num_calloc_calls (Number of MPIT_calloc calls)                         | 89        | 89        | 89        | 0         | 1          | 89        |
| num_free_calls (Number of MPIT_free calls)                             | 130       | 130       | 130       | 0         | 1          | 130       |
| num_malloc_calls (Number of MPIT_malloc calls)                         | 1,625     | 1,625     | 1,625     | 0         | 1          | 1,625     |
| num_memalign_calls (Number of MPIT_memalign calls)                     | 56        | 56        | 56        | 0         | 1          | 56        |
| num_memalign_free_calls (Number of MPIT_memalign_free calls)           | 0         | 0         | 0         | 0         | 0          | 0         |

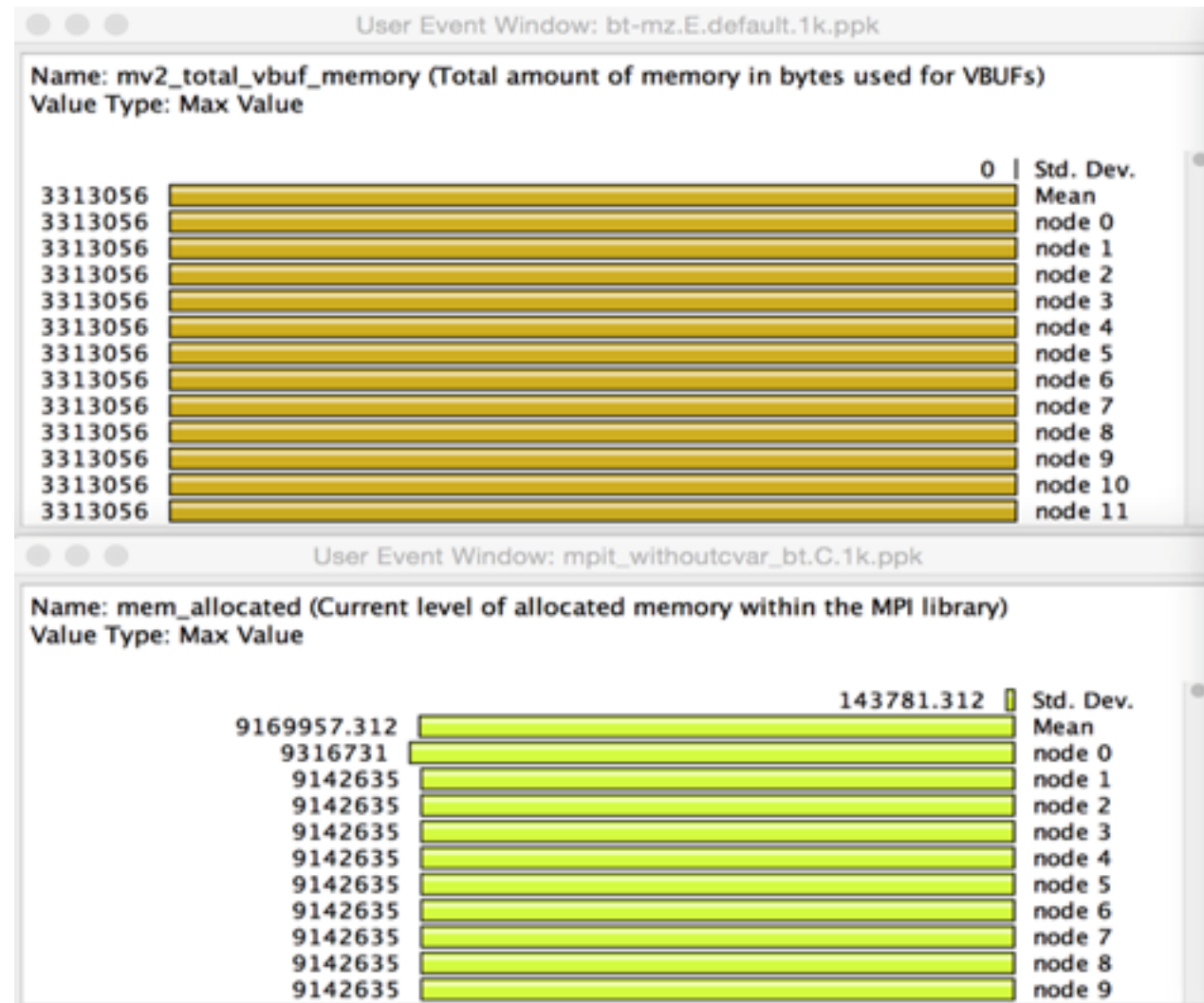
TAU: ParaProf Manager

| Applications               | TrialField               | Value                                                   |
|----------------------------|--------------------------|---------------------------------------------------------|
| Standard Applications      | MPI Processor Name       | c526-502.stampede.tacc.utexas.edu                       |
| Default App                | MPIR_CVAR_VBUF_POOL_SIZE | 0 (old) -> 16 (new), This set the size of the VBUF pool |
| Default Exp                |                          |                                                         |
| bt-mz.E.vbuf_pool_16.1k.pp |                          |                                                         |
| TIME                       |                          |                                                         |

Total memory used by VBUFs is reduced from 3,313,056 to 1,815,056

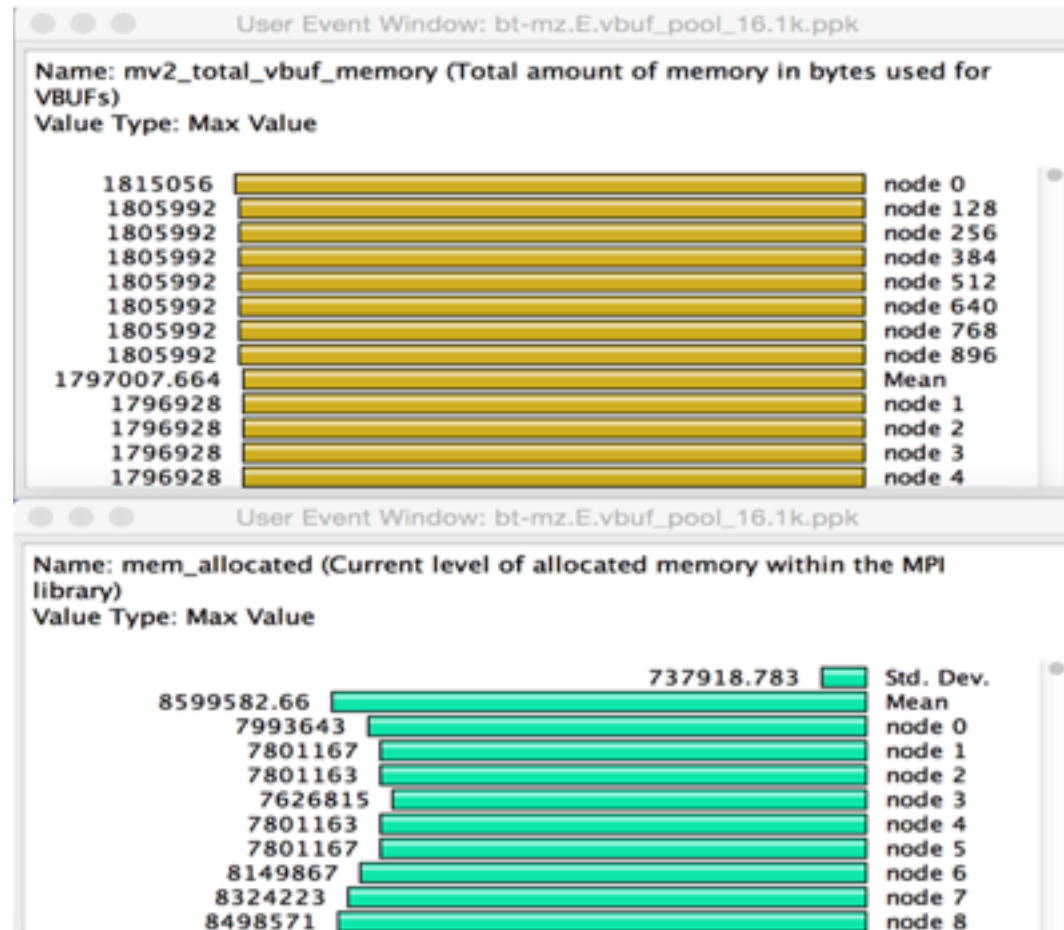
Courtesy: The TAU Team

# VBUF Memory Usage Without CVAR



Courtesy: The TAU Team

# VBUF Memory Usage With CVAR



```
% export TAU_TRACK_MPI_T_PVARS=1
```

```
% export TAU_MPI_T_CVAR_METRICS=MPIR_CVAR_VBUF_POOL_SIZE
```

```
% export TAU_MPI_T_CVAR_VALUES=16
```

```
% mpirun -np 1024 tau_exec -T mvapich2 ./a.out
```

Courtesy: The TAU Team

# MVAPICH2 Software Family

| Requirements                                           | Library      |
|--------------------------------------------------------|--------------|
| MPI with IB, iWARP, Omni-Path, and RoCE                | MVAPICH2     |
| MPI with IB, RoCE & GPU and Support for Deep Learning  | MVAPICH2-GDR |
| InfiniBand Network Analysis and Monitoring             | OSU INAM     |
| Microbenchmarks for Measuring MPI and PGAS Performance | OMB          |



# GPU-Aware (CUDA-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

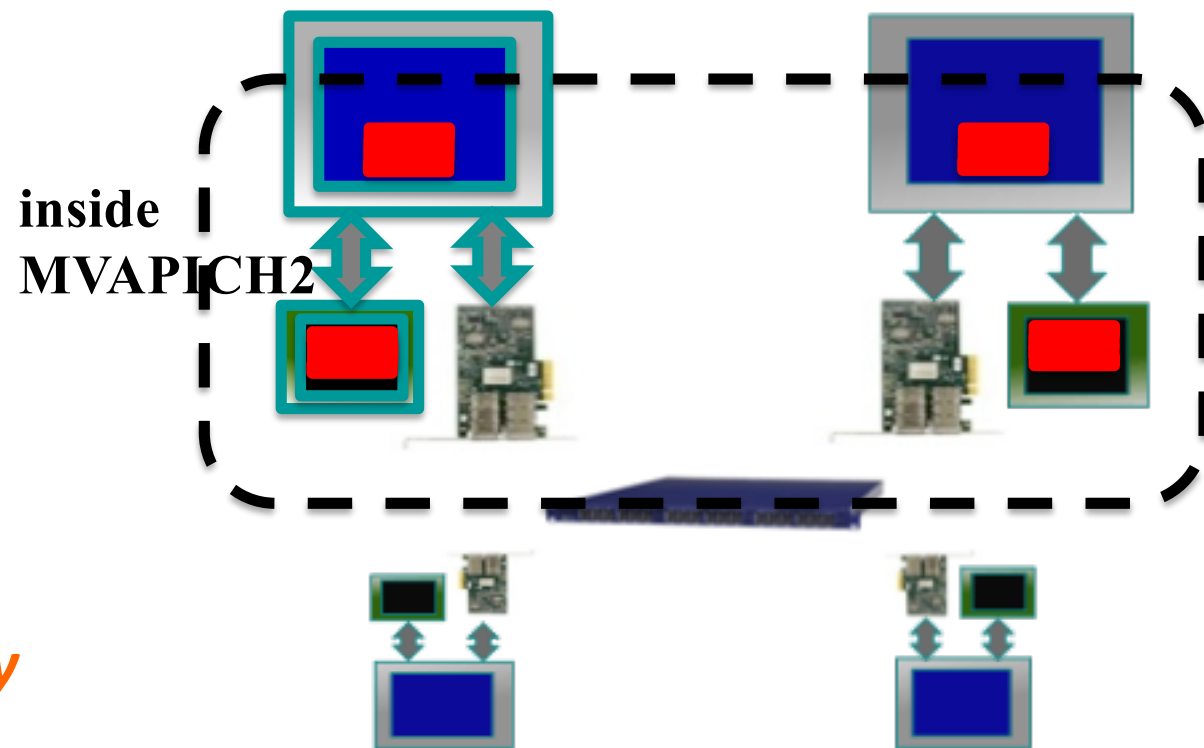
## At Sender:

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

## At Receiver:

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

*High Performance and High Productivity*



## CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.3 Releases

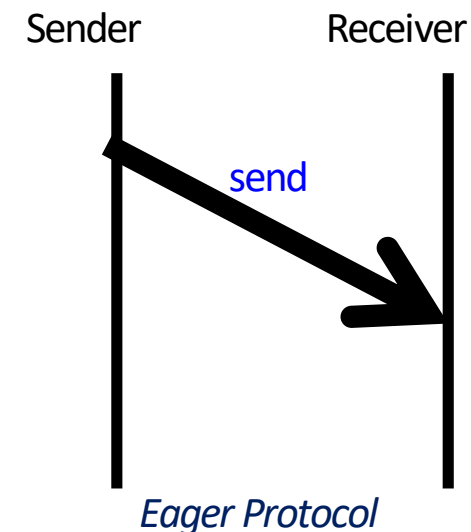
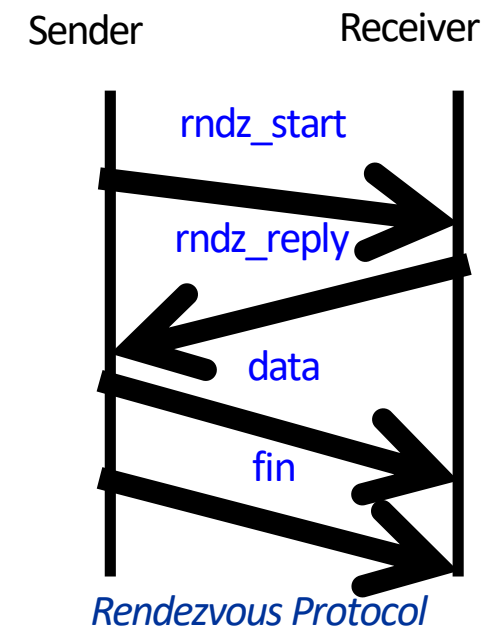
- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage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

# Presentation Overview

- **Support for Efficient Small Message Communication with GPUDirect RDMA**
- Multi-rail Support
- Support for Efficient Intra-node Communication using CUDA IPC
- MPI Datatype Support

# Enhanced MPI Design with GPUDirect RDMA

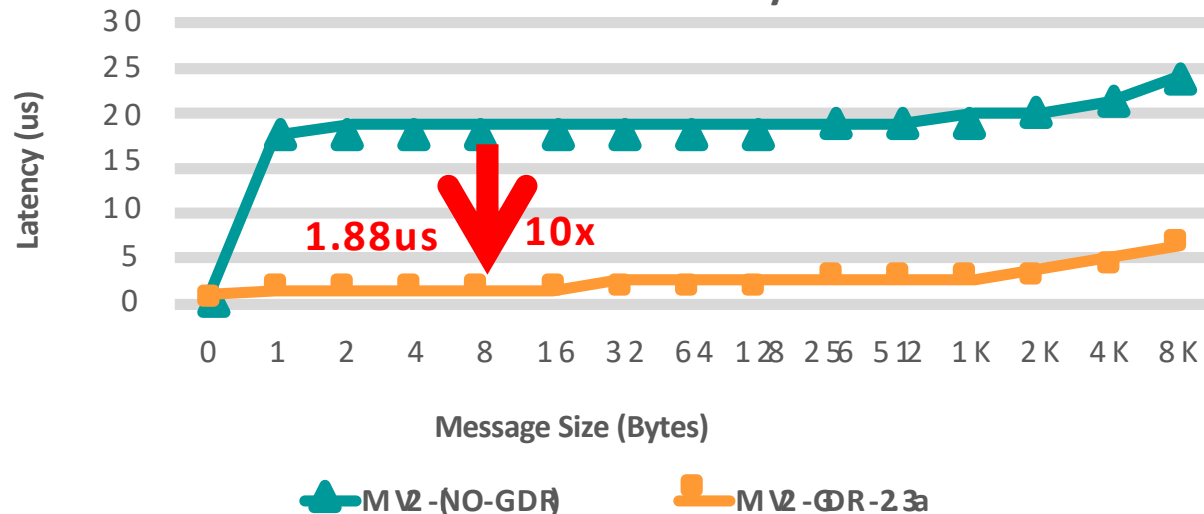
- Current MPI design using GPUDirect RDMA uses Rendezvous protocol
  - Has higher latency for small messages
- Can eager protocol be supported to improve performance for small messages?
- Two schemes proposed and used
  - **Loopback** (using network adapter to copy data)
  - **Fastcopy/GDRCOPY** (using CPU to copy data)



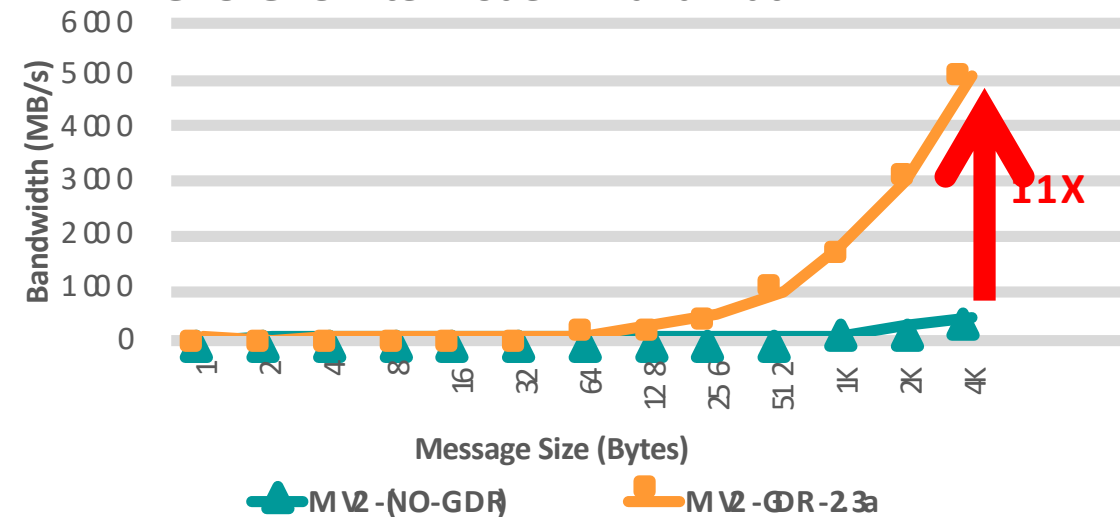
R. Shi, S. Potluri, K. Hamidouche M. Li, J. Perkins D. Rossetti and D. K. Panda, Designing Efficient Small Message Transfer Mechanism for Inter-node MPI Communication on InfiniBand GPU Clusters IEEE International Conference on High Performance Computing (HiPC'2014)

# Optimized MVAPICH2-GDR Design

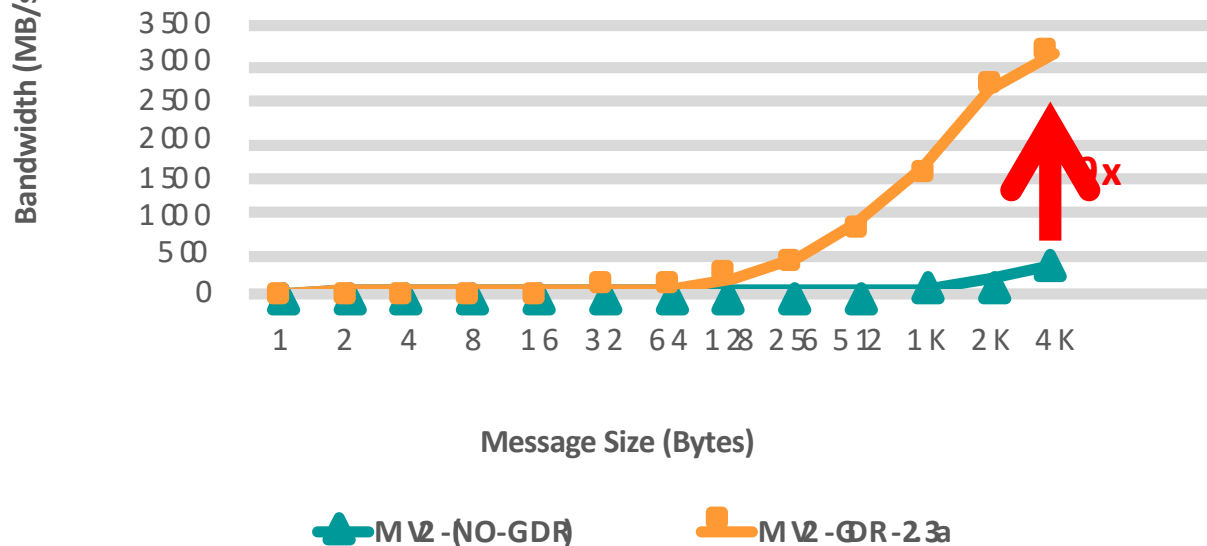
## GPU-GPU Inter-node Latency



## GPU-GPU Inter-node Bi-Bandwidth

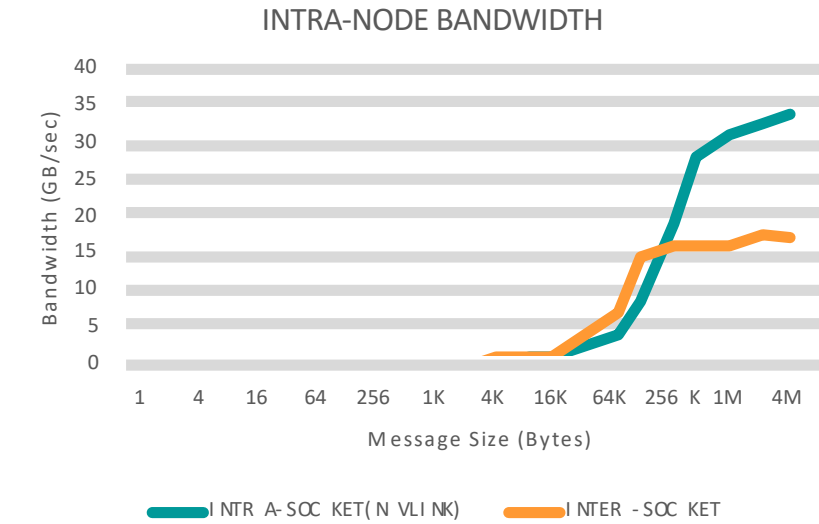
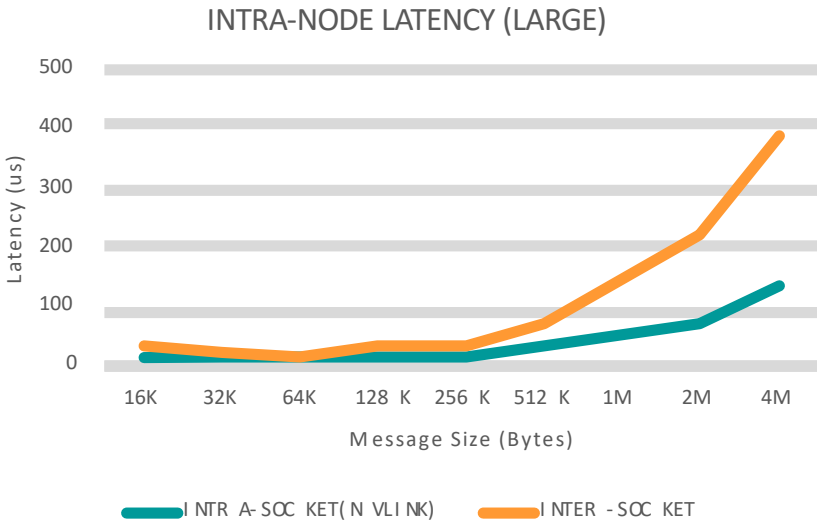
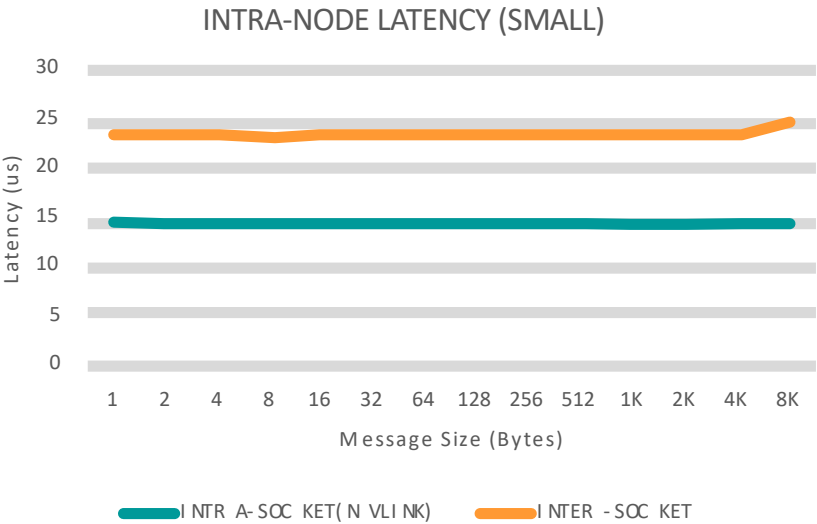


## GPU-GPU Inter-node Bandwidth



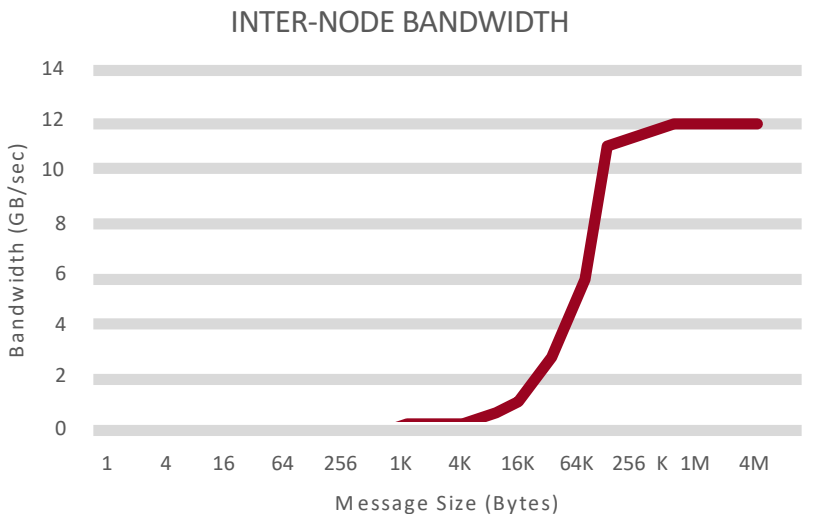
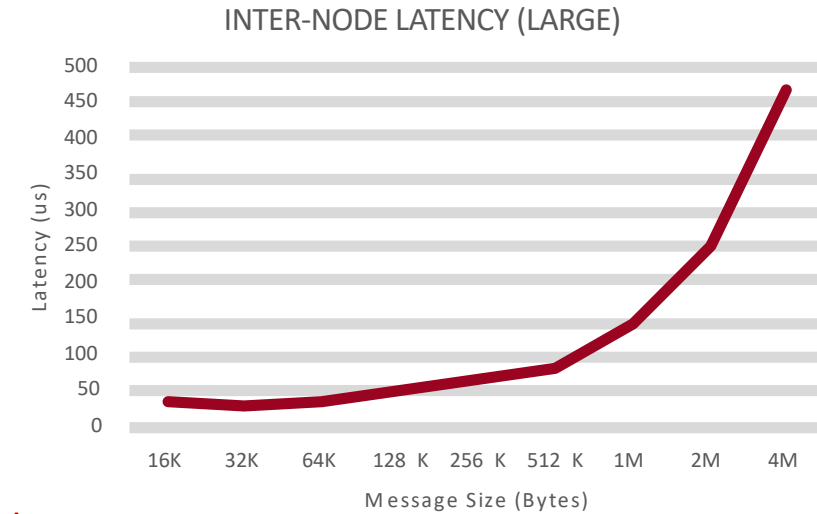
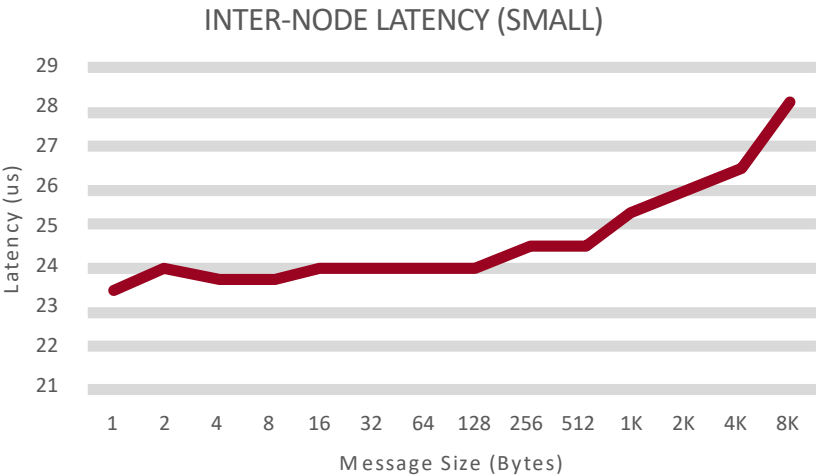
**MVAPICH2-GDR-2.3a**  
**Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores**  
**NVIDIA Volta V100 GPU**  
**Mellanox Connect-X4 EDR HCA**  
**CUDA 9.0**  
**Mellanox OFED 4.0 with GPU-Direct-RDMA**

# MVAPICH2-GDR: Performance on OpenPOWER (NVLink + Pascal)



**Intra-node Latency: 14.6 us (without GPUDirectRDMA)**

**Intra-node Bandwidth: 33.9 GB/sec (NVLINK)**



**Inter-node Latency: 23.8 us (without GPUDirectRDMA)**

**Available in MVAPICH2-GDR 2.3a**

**Inter-node Bandwidth: 11.9 GB/sec (EDR)**

**Platform: OpenPOWER (ppc64le) nodes equipped with a dual-socket CPU, 4 Pascal P100-SXM GPUs, and EDR InfiniBand Inter-connect**

## Tuning GDRCOPY Designs in MVAPICH2-GDR

| Parameter                           | Significance                                                   | Default        | Notes                                                                                  |
|-------------------------------------|----------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------|
| MV2_USE_GPUDIRECT_GDRCOPY           | • Enable / Disable GDRCOPY-based designs                       | 1<br>(Enabled) | • Always enable                                                                        |
| MV2_GPUDIRECT_GDRCOPY_LIMIT         | • Controls messages size until which GDRCOPY is used           | 8 KByte        | • Tune for your system<br>• 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<br>• 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](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<br>(Enabled) | • Always enable                                                                                                                |
| MV2_GPUDIRECT_LOOPBACK_LIMIT | • Controls messages size until which LOOPBACK is used | 8 KByte        | • Tune for your system<br>• GPU type, host architecture and HCA. Impacts the eager performance<br>• 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](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)



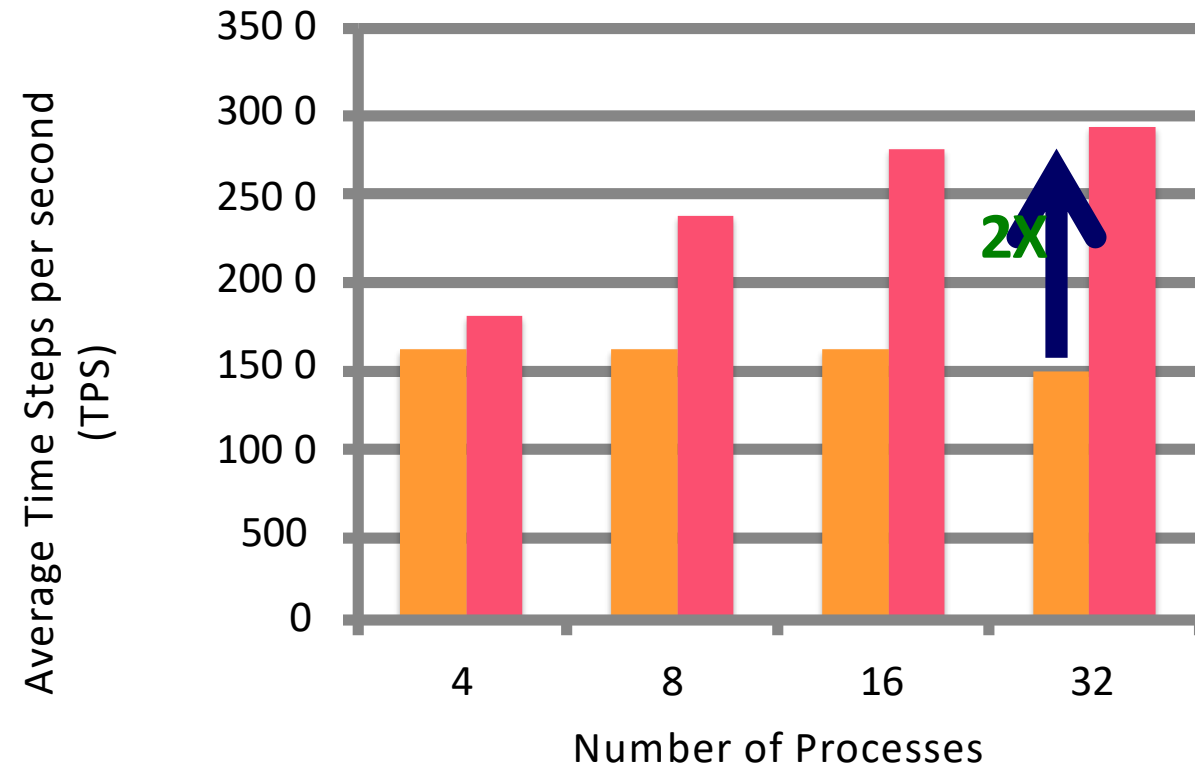
## Tuning GPUDirect RDMA (GDR) Designs in MVAPICH2-GDR

| Parameter                       | Significance                                                                          | Default        | Notes                                                                                                                                  |
|---------------------------------|---------------------------------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------|
| MV2_USE_GPUDIRECT               | • Enable / Disable GDR-based designs                                                  | 1<br>(Enabled) | • Always enable                                                                                                                        |
| MV2_GPUDIRECT_LIMIT             | • Controls messages size until which GPUDirect RDMA is used                           | 8 KByte        | • Tune for your system<br>• GPU type, host architecture and<br>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<br>• GPU type, HCA type and configuration                                                                       |

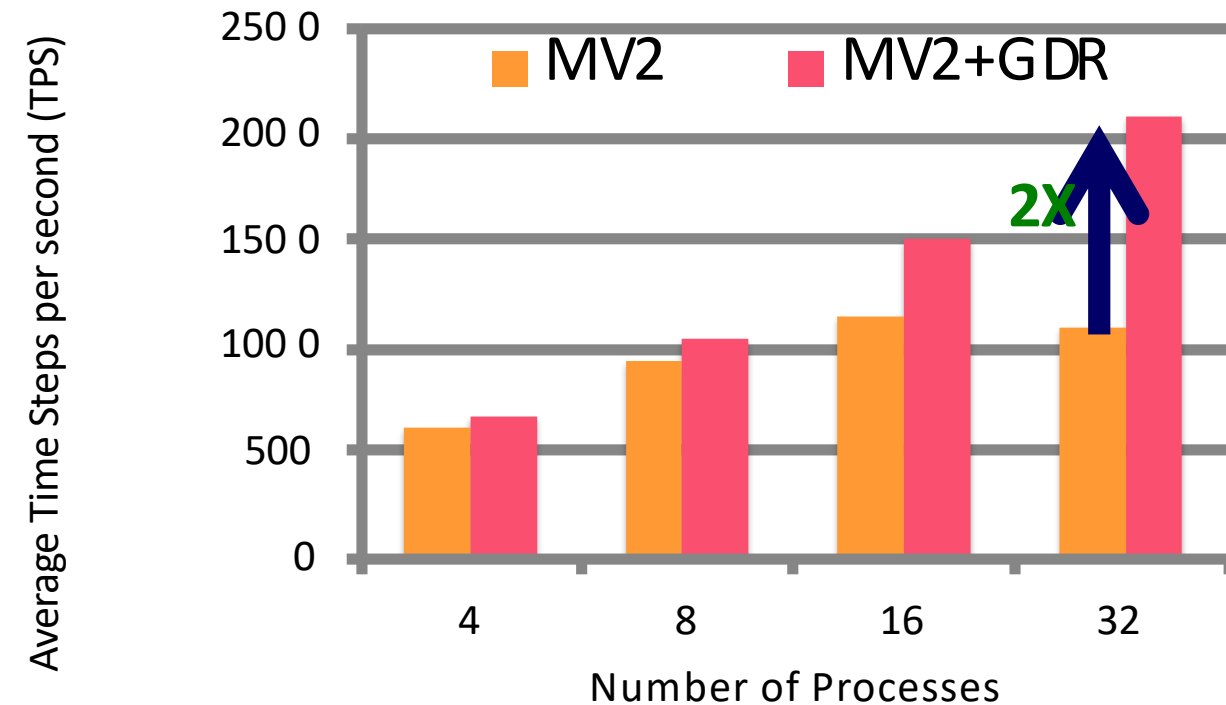
- Refer to **Tuning and Usage Parameters** section of MVAPICH2-GDR user guide for more information
- [http://mvapich.cse.ohio-state.edu/userguide/gdr/#\\_tuning\\_and\\_usage\\_parameters](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)

# 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

# Presentation Overview

- Support for Efficient Small Message Communication with GPUDirect RDMA
- **Multi-rail Support**
- Support for Efficient Intra-node Communication using CUDA IPC
- MPI Datatype Support

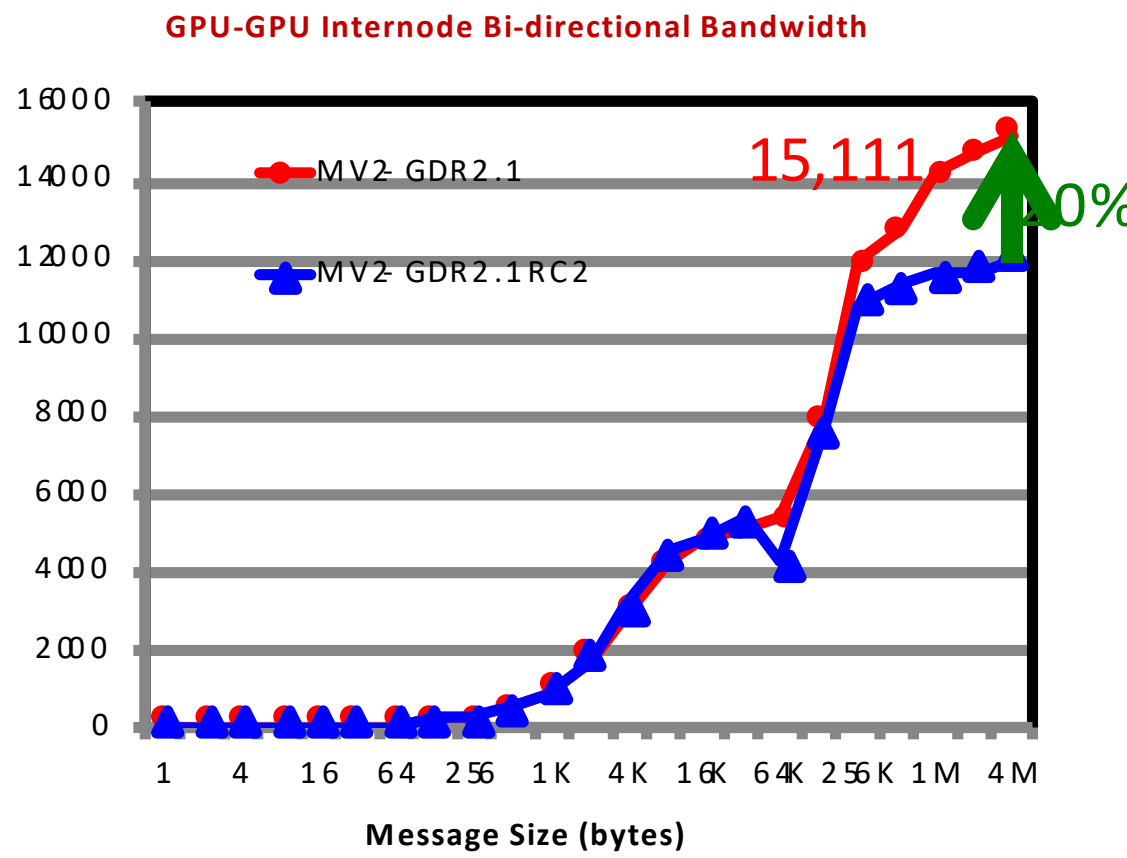
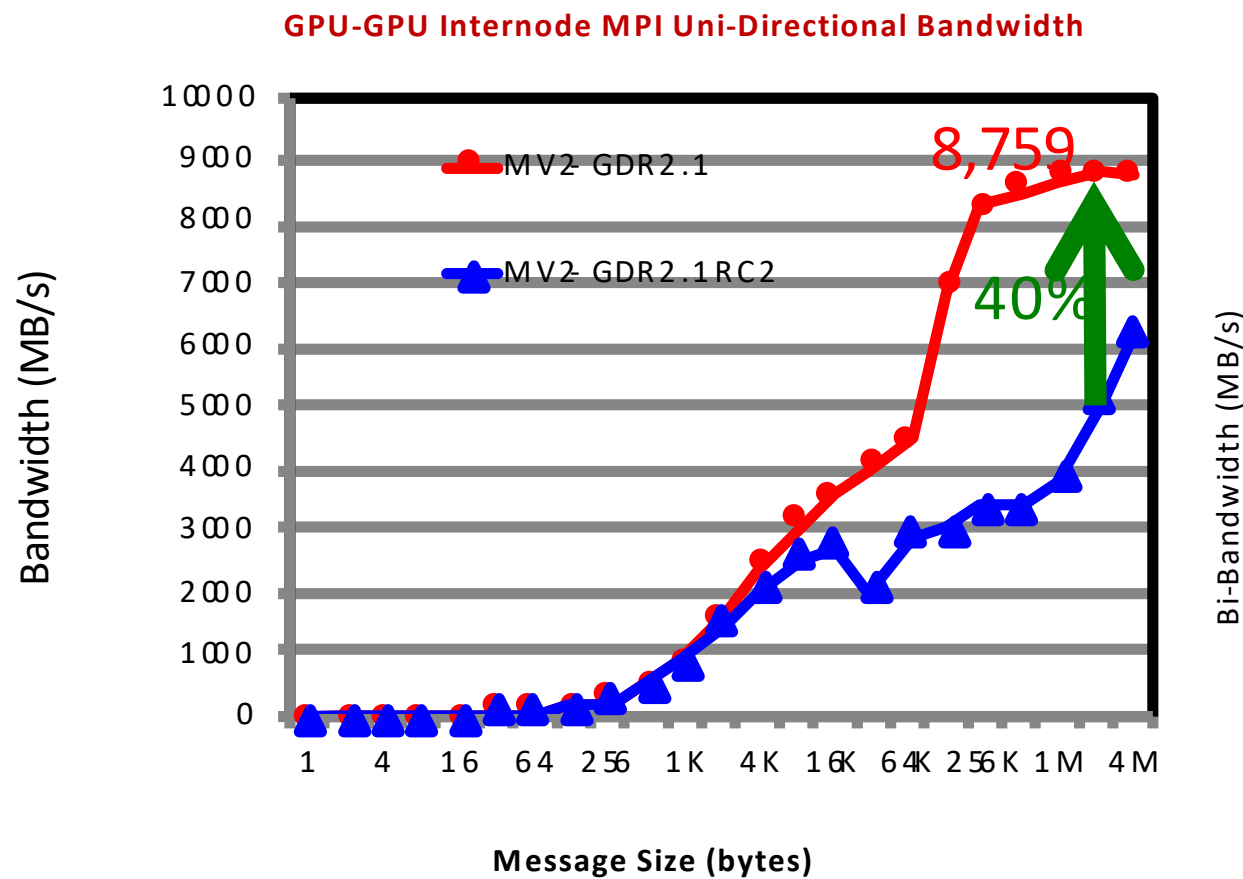
# Tuning Multi-rail Support in MVAPICH2-GDR

- 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                                                      |
| MV2_PROCESS_TO_RAIL_MAPPING | • Explicit binding of the HCAs to the CPU      | First HCA | • Use this parameter to manually select a different parameter only if default binding seems to perform poorly |

- 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)

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



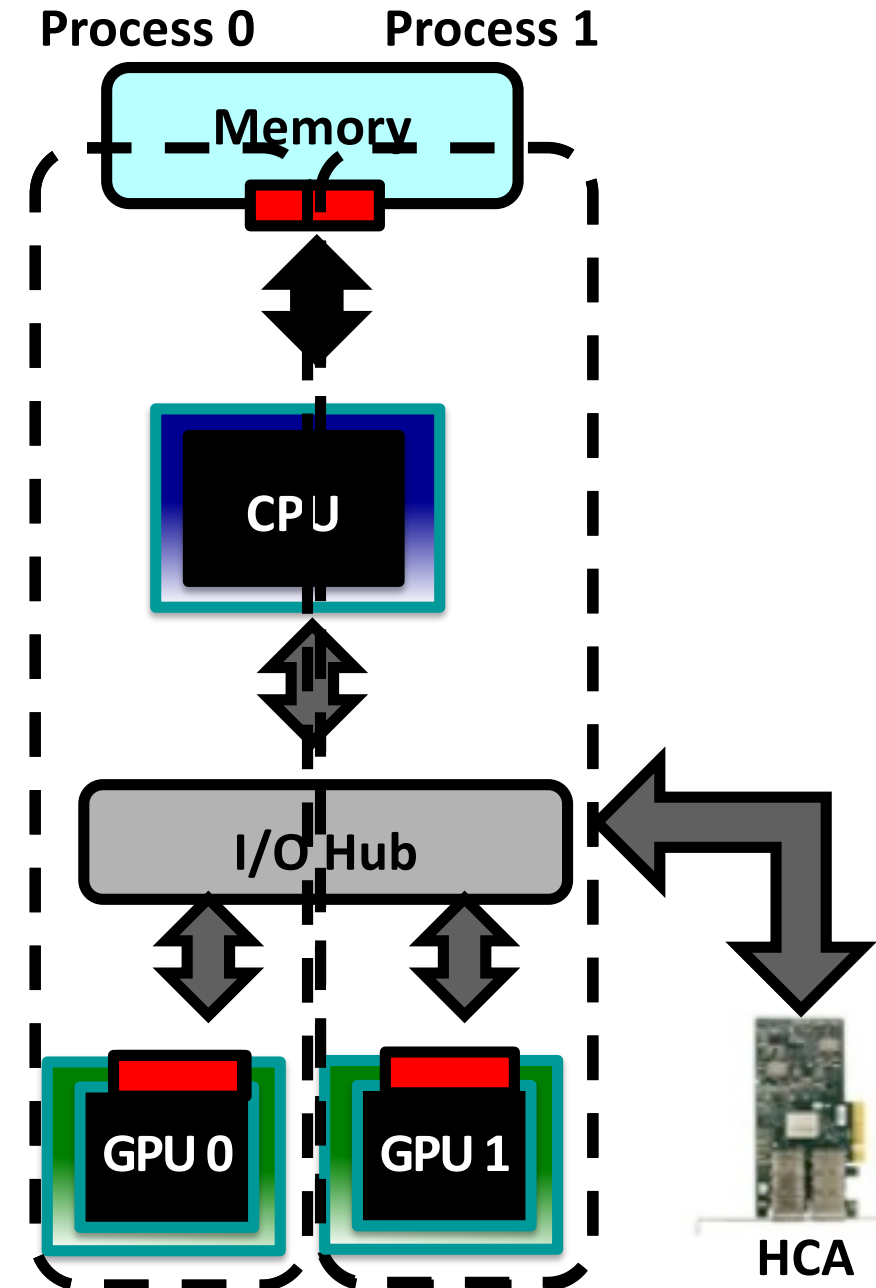
MVAPICH2-GDR-2.1 and MVAPICH2-GDR 2.1 RC2  
Intel Ivy Bridge (E5-2680 v2) node - 20 cores, NVIDIA Tesla K40c GPU  
Mellanox Connect-IB Dual-FDR HCA CUDA 7  
Mellanox OFED 2.4 with GPU-Direct-RDMA

# Presentation Overview

- Support for Efficient Small Message Communication with GPUDirect RDMA
- Multi-rail Support
- **Support for Efficient Intra-node Communication using CUDA IPC**
- MPI Datatype Support

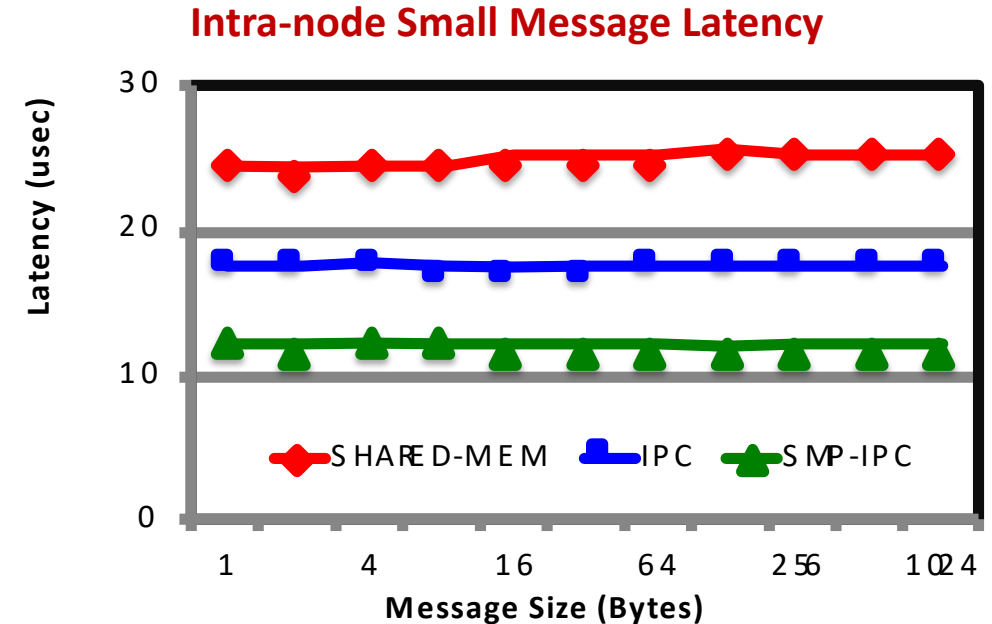
# Multi-GPU Configurations

- 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) | <ul style="list-style-type: none"> <li>• Benefits Device-to-Device transfers</li> <li>• Hurts Device-to-Host/Host-to-Device transfers</li> <li>• Always set to 1 if application involves only Device-to-Device transfers</li> </ul> |
| MV2_IPC_THRESHOLD | • Message size where IPC code path will be used                | 16 KBytes    | • Tune for your system                                                                                                                                                                                                              |



## Alternative Designs

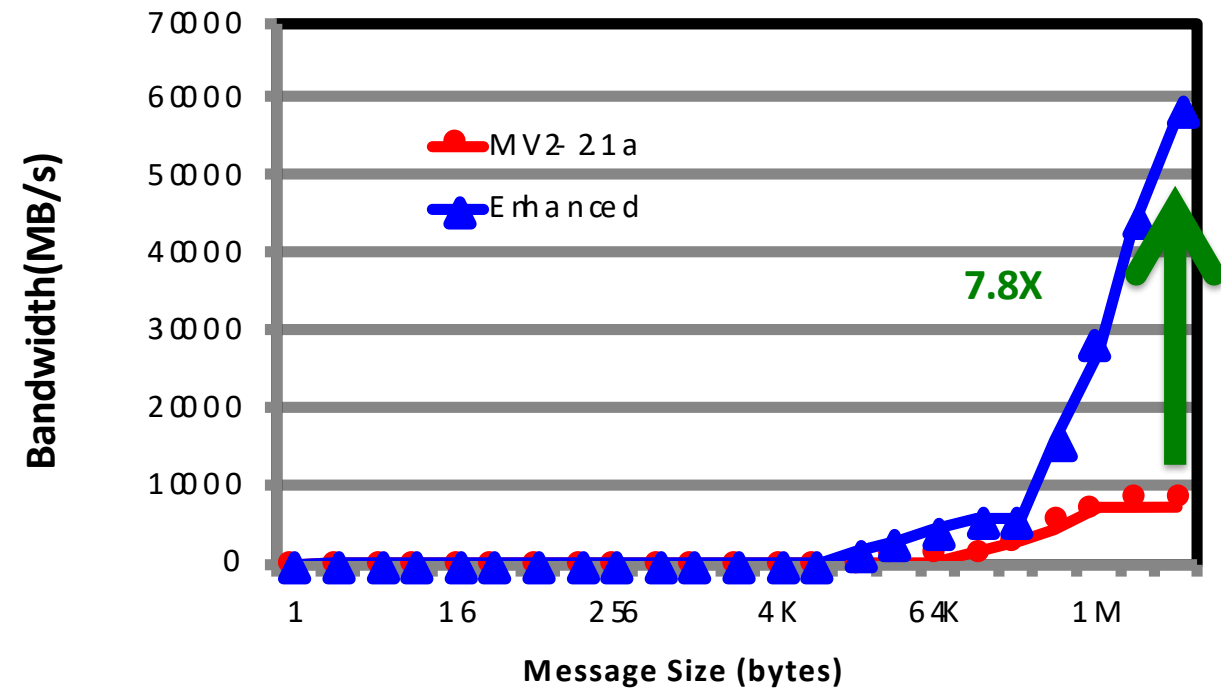
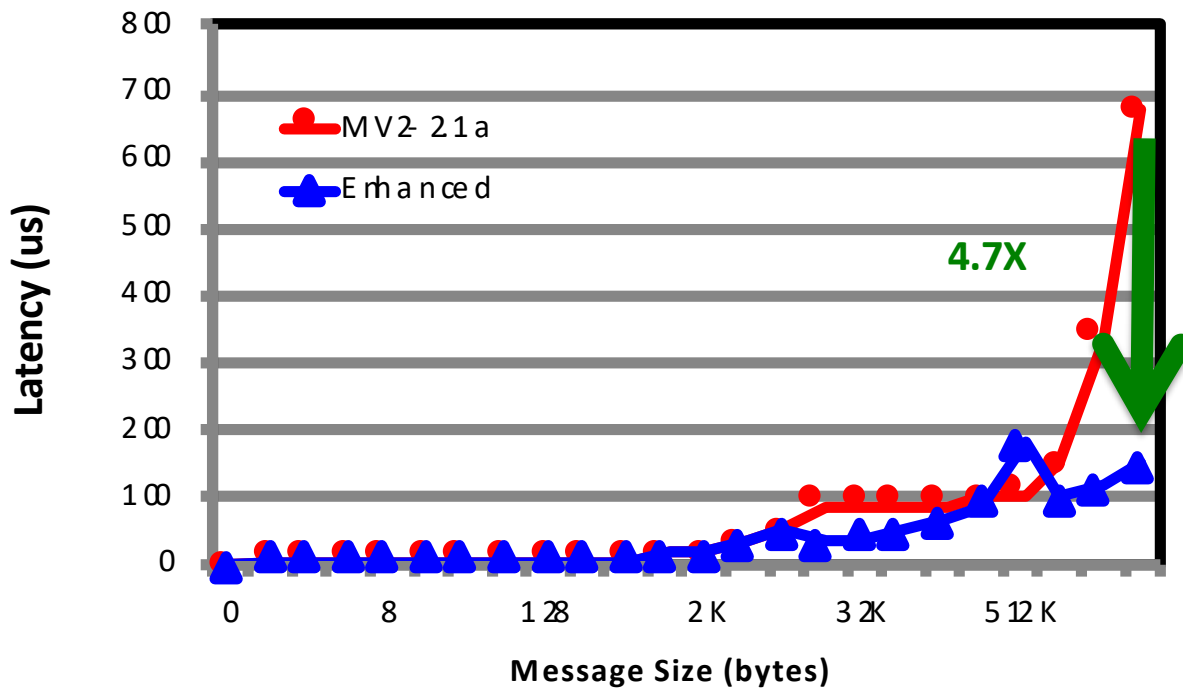
- Double Buffering schemes
  - Uses intermediate buffers (IPC Pinned)
  - Control information through Host memories
    - Exchange the handlers through the host for IPC completion
  - Works for all CUDA versions (Since 5.5)
  - Memory Overhead
- Cache based design
  - Rendezvous based design
  - Cache the IPC handlers at the source and destination (through the control messages)
  - With Cache hit => direct data movement
  - Requires CUDA 6.5 and onwards
  - High Performance and memory efficiency

## Tuning IPC designs in MVAPICH2-GDR

- Works between GPUs within the same socket or IOH

| Parameter                      | Significance                                                      | Default        | Notes                                                                                                                                           |
|--------------------------------|-------------------------------------------------------------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| MV2_CUDA_ENABLE_IPC_CACHE      | • Enable / Disable CUDA IPC_CACHE-based designs                   | 1<br>(Enabled) | • Always leave set to 1<br>• Best performance<br>• Enables one-sided semantics                                                                  |
| MV2_CUDA_IPC_BUFFERED          | • Enable / Disable CUDA IPC_BUFFERED design                       | 1<br>(Enabled) | • Used for subset of operations<br>• Backup for the IPC-Cache design<br>• Uses double buffering schemes<br>• Used for efficient Managed support |
| MV2_CUDA_IPC_MAX_CACHE_ENTRIES | • Number of entries in the cache                                  | 64             | • Tuned for your application<br>• Depends on the communication patterns<br>• Increase the value for irregular applications                      |
| MV2_CUDA_IPC_STAGE_BUF_SIZE    | • The size of the staging buffers in the double buffering schemes |                | • Tune this value only if degradation is observed with IPC transfers                                                                            |

# IPC-Cache Communication Enhancement



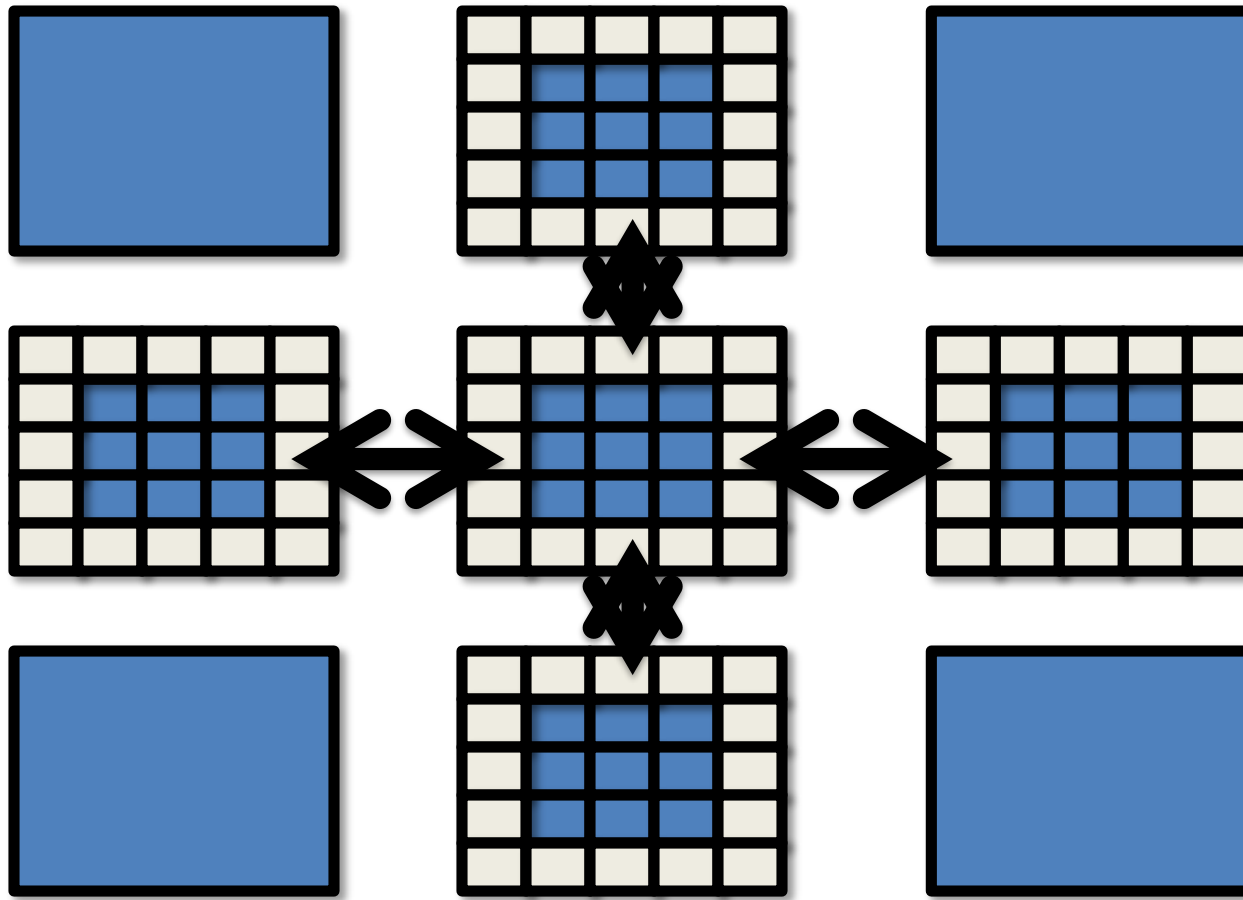
- Two Processes sharing the same **K80 GPUs**
- Proposed designs achieve **4.7X** improvement in latency
- **7.8X** improvement is delivered for Bandwidth
- Available with the latest release of **MVAPICH2-GDR 2.2**

# Presentation Overview

- Support for Efficient Small Message Communication with GPUDirect RDMA
- Multi-rail Support
- Support for Efficient Intra-node Communication using CUDA IPC
- **MPI Datatype Support**

# 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.*

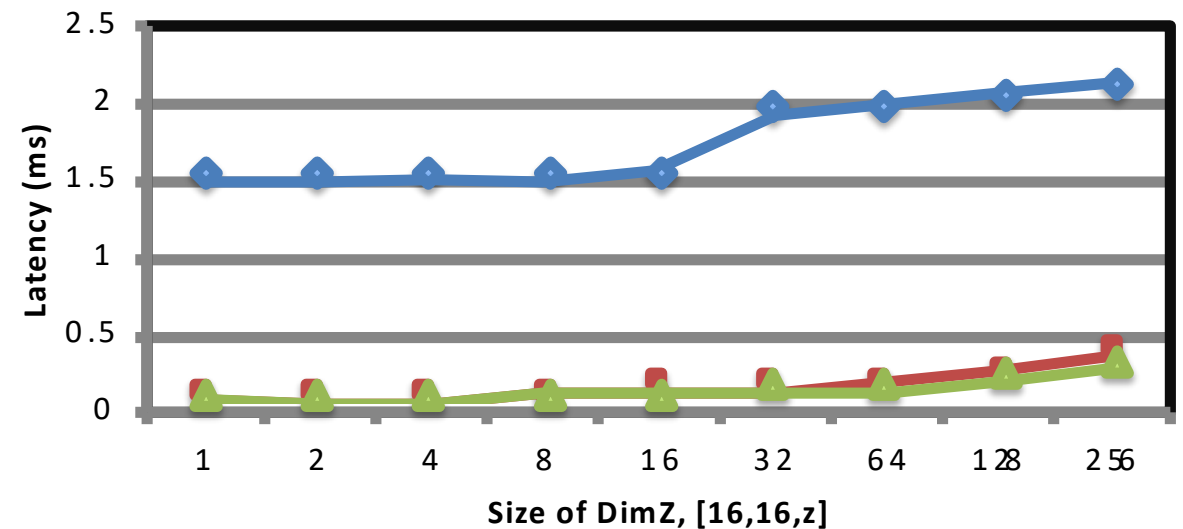
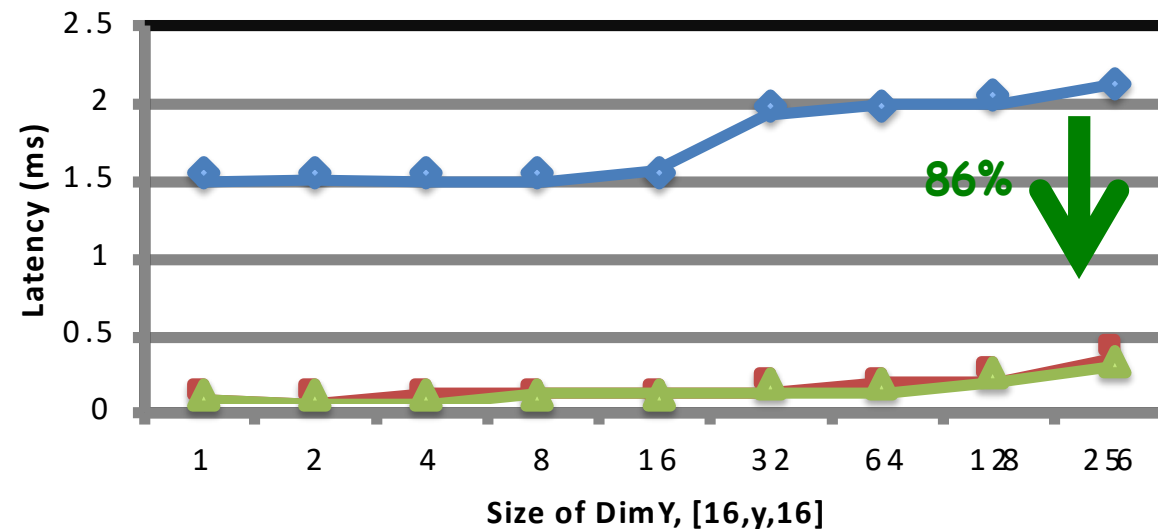
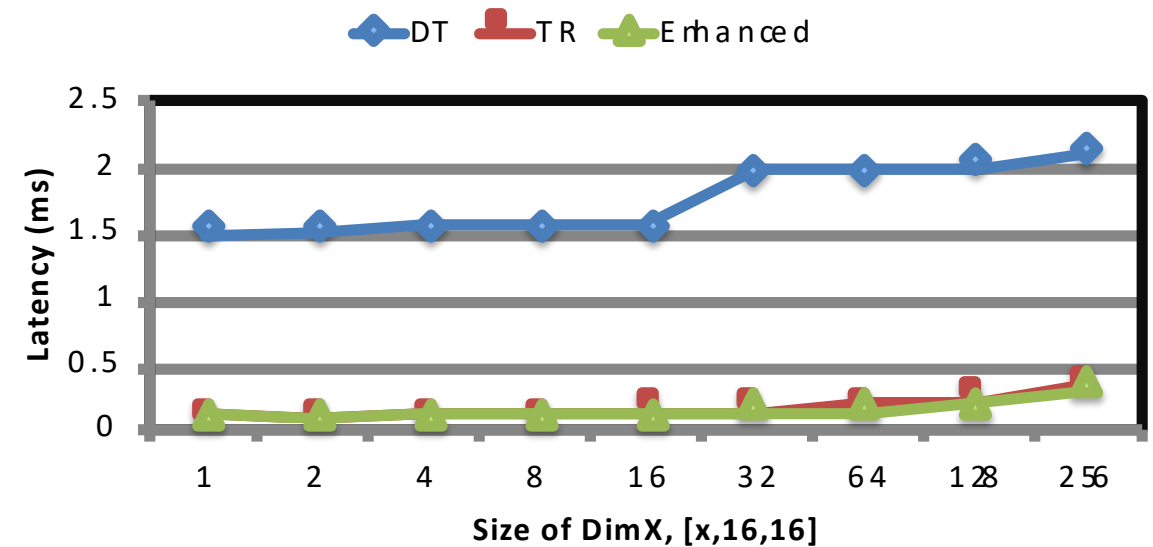
# MPI Datatype Processing (Computation Optimization )

- 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





# MPI Datatype Processing (Communication Optimization )

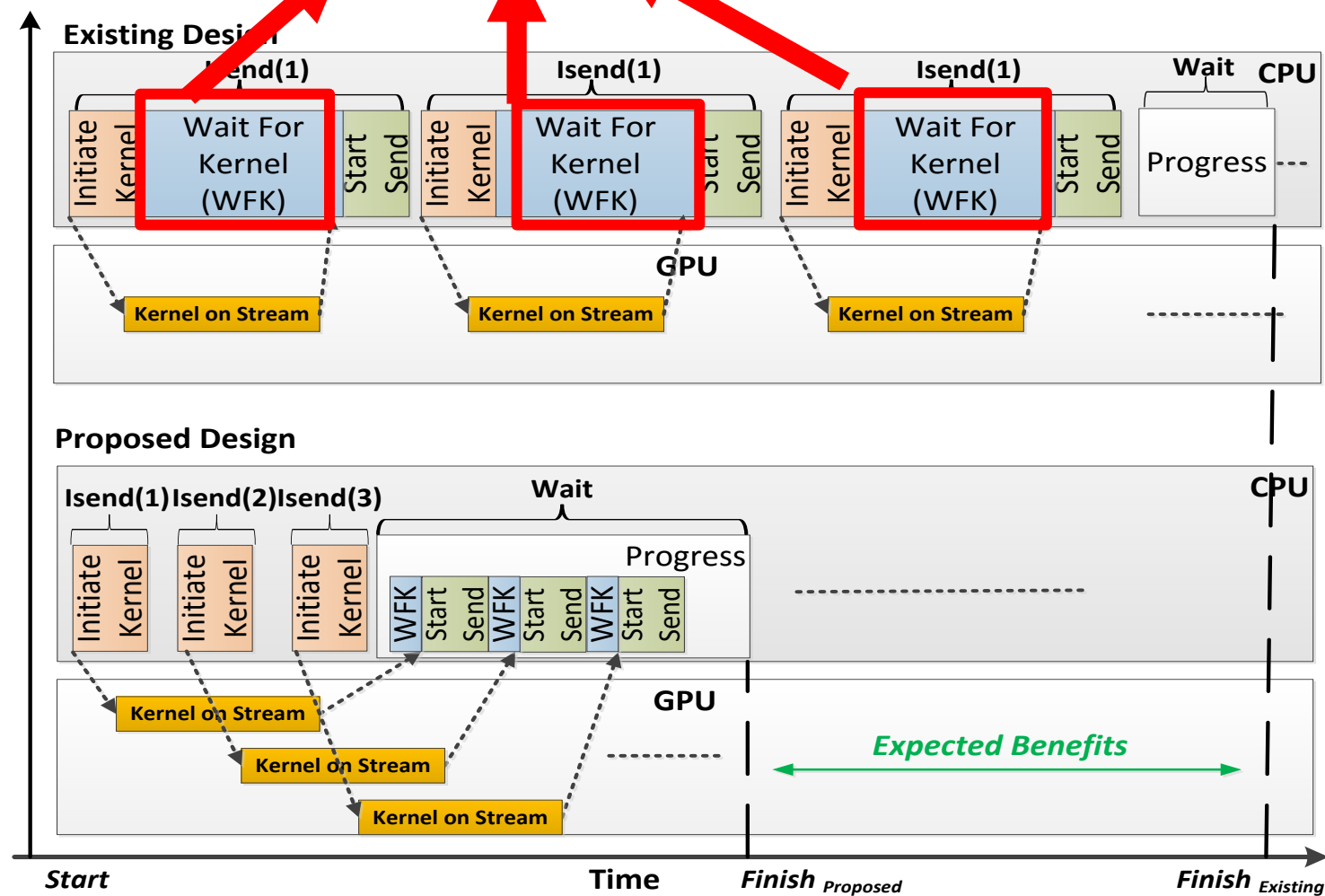
## Common Scenario

```
MPI_Isend (A,.. Datatype,...)
MPI_Isend (B,.. Datatype,...)
MPI_Isend (C,.. Datatype,...)
MPI_Isend (D,.. Datatype,...)
...
```

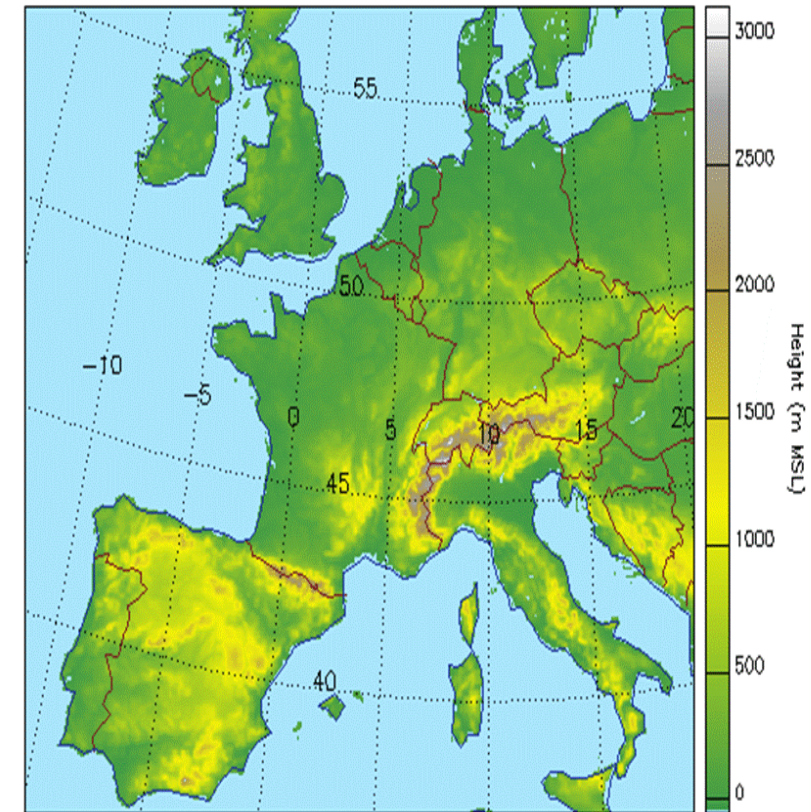
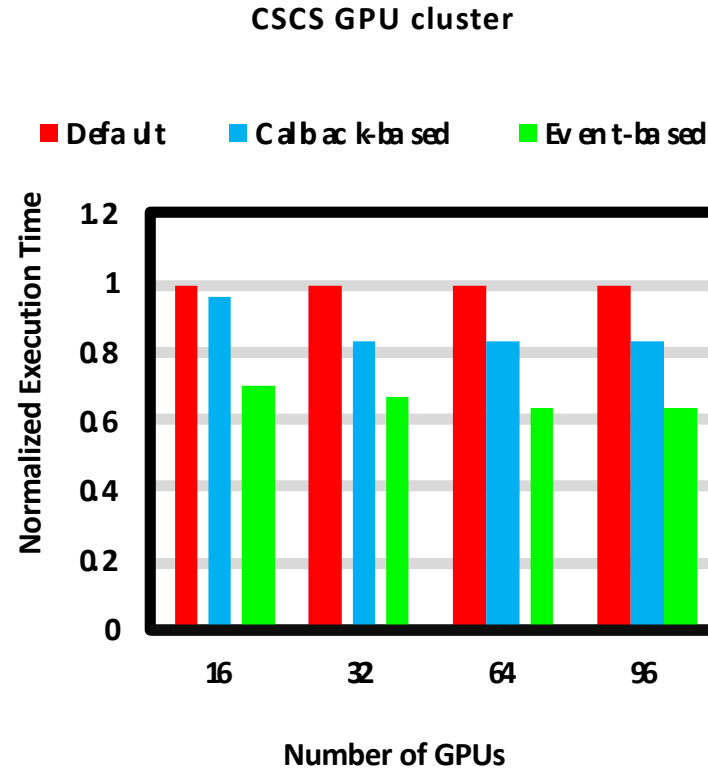
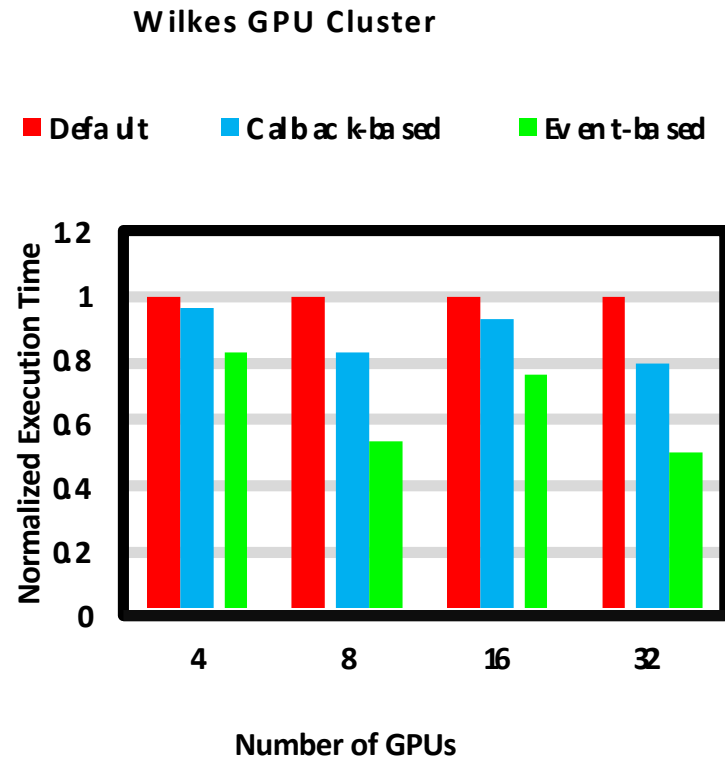
```
MPI_Waitall (...);
```

\*A, B...contain non-contiguous MPI Datatype

## Waste of computing resources on CPU and GPU



# Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland



- 2X improvement on 32 GPUs nodes
- 30% improvement on 96 GPU nodes (8 GPUs/node)

Cosmo model: <http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/>

On-going collaboration with CSCS and MeteoSwiss (Switzerland) in co-designing MV2-GDR and Cosmo Application

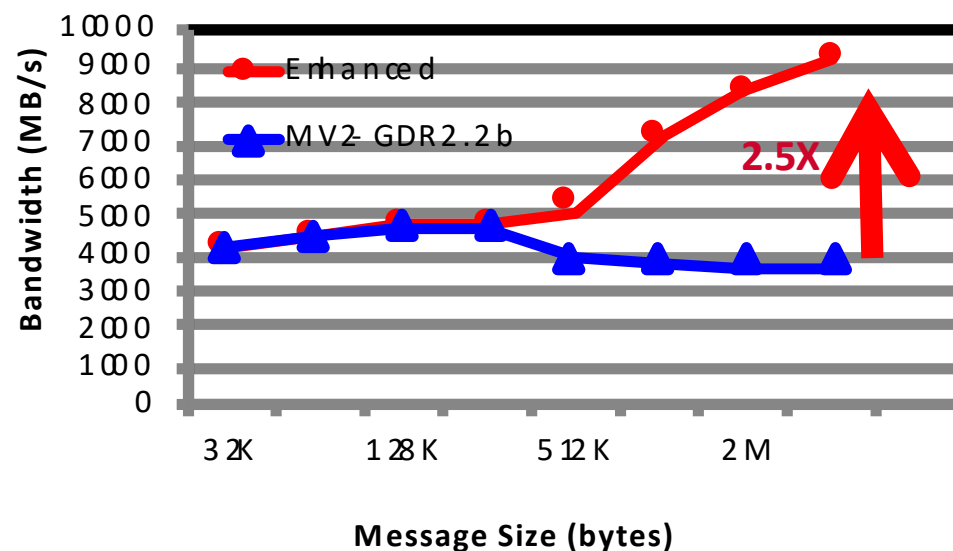
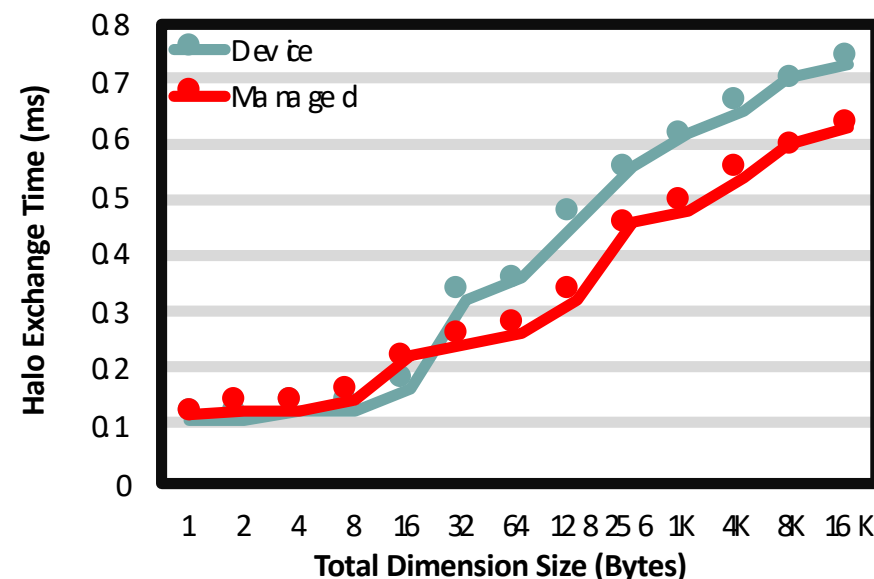
C. Chu, K. Hamidouche, A. Venkatesh, D. Banerjee, H. Subramoni, and D. K. Panda, Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-enabled Systems, IPDPS'16

# Enhanced Support for GPU Managed Memory

- CUDA Managed => no memory pin down
  - No IPC support for intranode communication
  - No GDR support for Internode communication
- Significant productivity benefits due to abstraction of explicit allocation and *cudaMemcpy()*
- Initial and basic support in MVAPICH2-GDR
  - For both intra- and inter-nodes use “pipeline through” host memory
- Enhance intranode managed memory to use IPC
  - Double buffering pair-wise IPC-based scheme
  - Brings IPC performance to Managed memory
  - High performance and high productivity
  - 2.5 X improvement in bandwidth
- OMB extended to evaluate the performance of point-to-point and collective communications using managed buffers

D. S. Banerjee, K Hamidouche, and D. K Panda, Designing High Performance Communication Runtime for GPUManaged Memory: Early Experiences, GPGPU-9 Workshop, held in conjunction with PPOPP '16

2D Stencil Performance for Halowidth=1



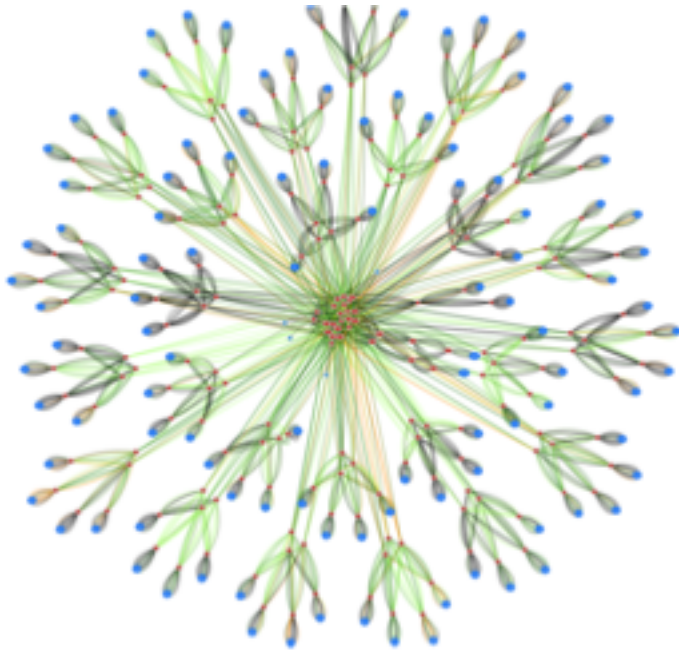
# MVAPICH2 Software Family

| Requirements                                           | Library      |
|--------------------------------------------------------|--------------|
| MPI with IB, iWARP, Omni-Path, and RoCE                | MVAPICH2     |
| MPI with IB, RoCE & GPU and Support for Deep Learning  | MVAPICH2-GDR |
| InfiniBand Network Analysis and Monitoring             | OSU INAM     |
| Microbenchmarks for Measuring MPI and PGAS Performance | OMB          |

# Overview of OSU INAM

- A network monitoring and analysis tool that is capable of analyzing traffic on the InfiniBand network with inputs from the MPI runtime
  - <http://mvapich.cse.ohio-state.edu/tools/osu-inam/>
- Monitors IB clusters in real time by querying various subnet management entities and gathering input from the MPI runtimes
- OSU INAM v0.9.3 released on 03/06/18
  - Enhance INAMD to query end nodes based on command line option
  - Add a web page to display size of the database in real-time
  - Enhance interaction between the web application and SLURM job launcher for increased portability
  - Improve packaging of web application and daemon to ease installation
  - Enhance web interface to improve the user experience
  - Improve debugging and logging support in daemon and web application
- Significant enhancements to user interface to enable scaling to clusters with thousands of nodes
- Improve database insert times by using 'bulk inserts'
- Capability to look up list of nodes communicating through a network link
- Capability to classify data flowing over a network link at job level and process level granularity in conjunction with MVAPICH2-X 2.2rc1
- "Best practices " guidelines for deploying OSU INAM on different clusters
- Capability to analyze and profile node-level, job-level and process-level activities for MPI communication
  - Point-to-Point, Collectives and RMA
- Ability to filter data based on type of counters using "drop down" list
- Remotely monitor various metrics of MPI processes at user specified granularity
- "Job Page" to display jobs in ascending/descending order of various performance metrics in conjunction with MVAPICH2-X
- Visualize the data transfer happening in a "live" or "historical" fashion for entire network, job or set of nodes

# OSU INAM Features



Comet@SDSC --- Clustered View

(1,879 nodes, 212 switches, 4,377 network links)

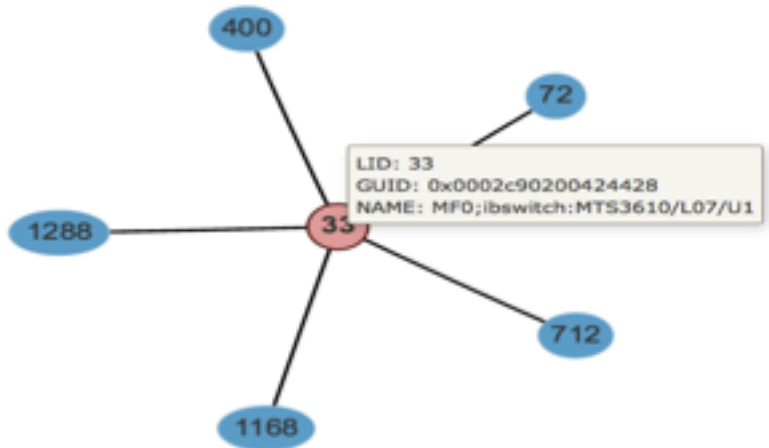


Finding Routes Between Nodes

- Show network topology of large clusters
- Visualize traffic pattern on different links
- Quickly identify congested links/links in error state
- See the history unfold – play back historical state of the network



## OSU INAM Features (Cont.)



Visualizing a Job (5 Nodes)

- Job level view
  - Show different network metrics (load, error, etc.) for any live job
  - Play back historical data for completed jobs to identify bottlenecks
- Node level view - details per process or per node
  - CPU utilization for each rank/node
  - Bytes sent/received for MPI operations (pt-to-pt, collective, RMA)
  - Network metrics (e.g. XmitDiscard, RcvError) per rank/node



Estimated Process Level Link Utilization

- Estimated Link Utilization view
  - Classify data flowing over a network link at different granularity in conjunction with MVAPICH2-X 2.2rc1
    - Job level and
    - Process level

# OSU Microbenchmarks

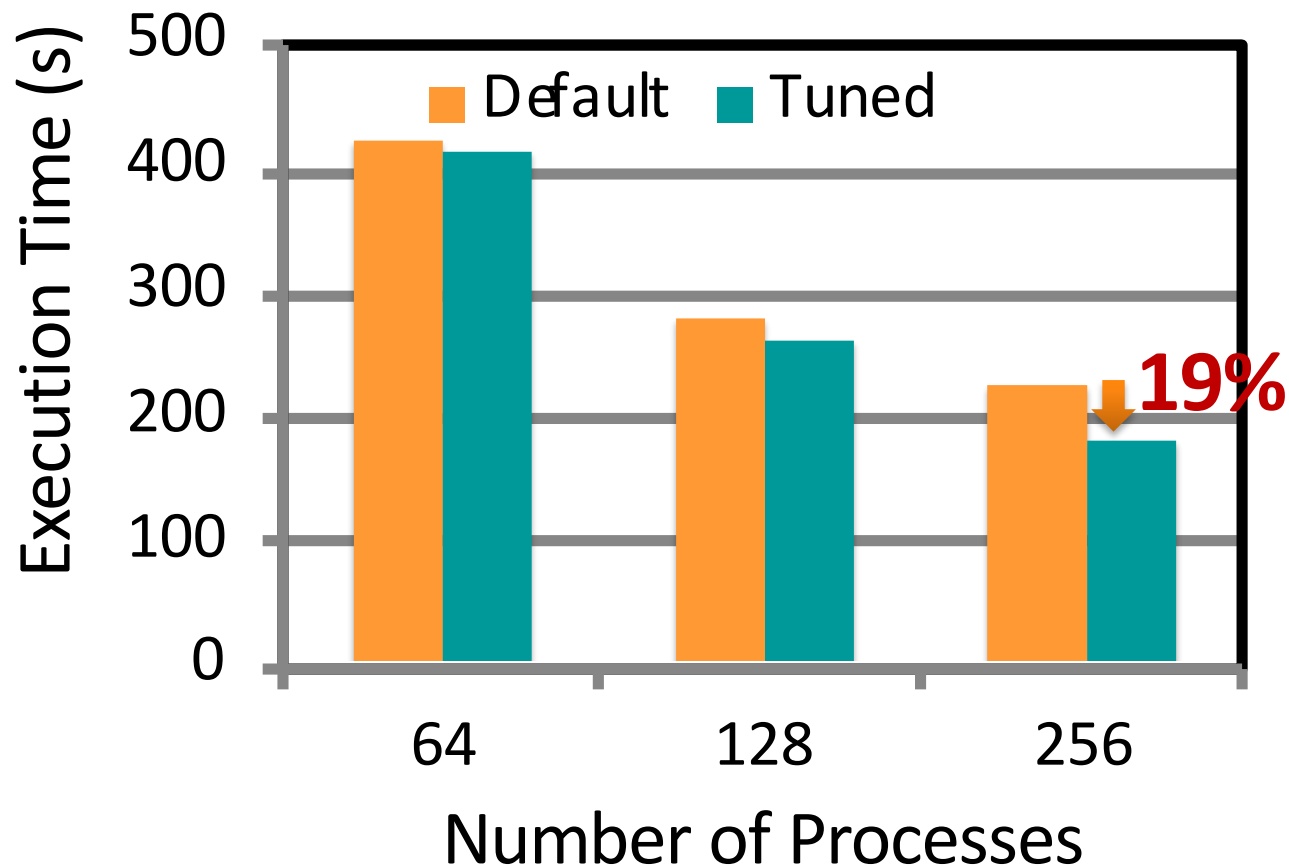
- Available since 2004
- Suite of microbenchmarks to study communication performance of various programming models
- Benchmarks available for the following programming models
  - Message Passing Interface (MPI)
  - Partitioned Global Address Space (PGAS)
    - Unified Parallel C (UPC)
    - Unified Parallel 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
- Continuing to add support for newer primitives and features
- Please visit the following link for more information
  - <http://mvapich.cse.ohio-state.edu/benchmarks/>



# 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
  - Neuron
  - SMG2000
  - Cloverleaf
  - SPEC (LAMMPS, POP2, TERA\_TF, WRF2)
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.

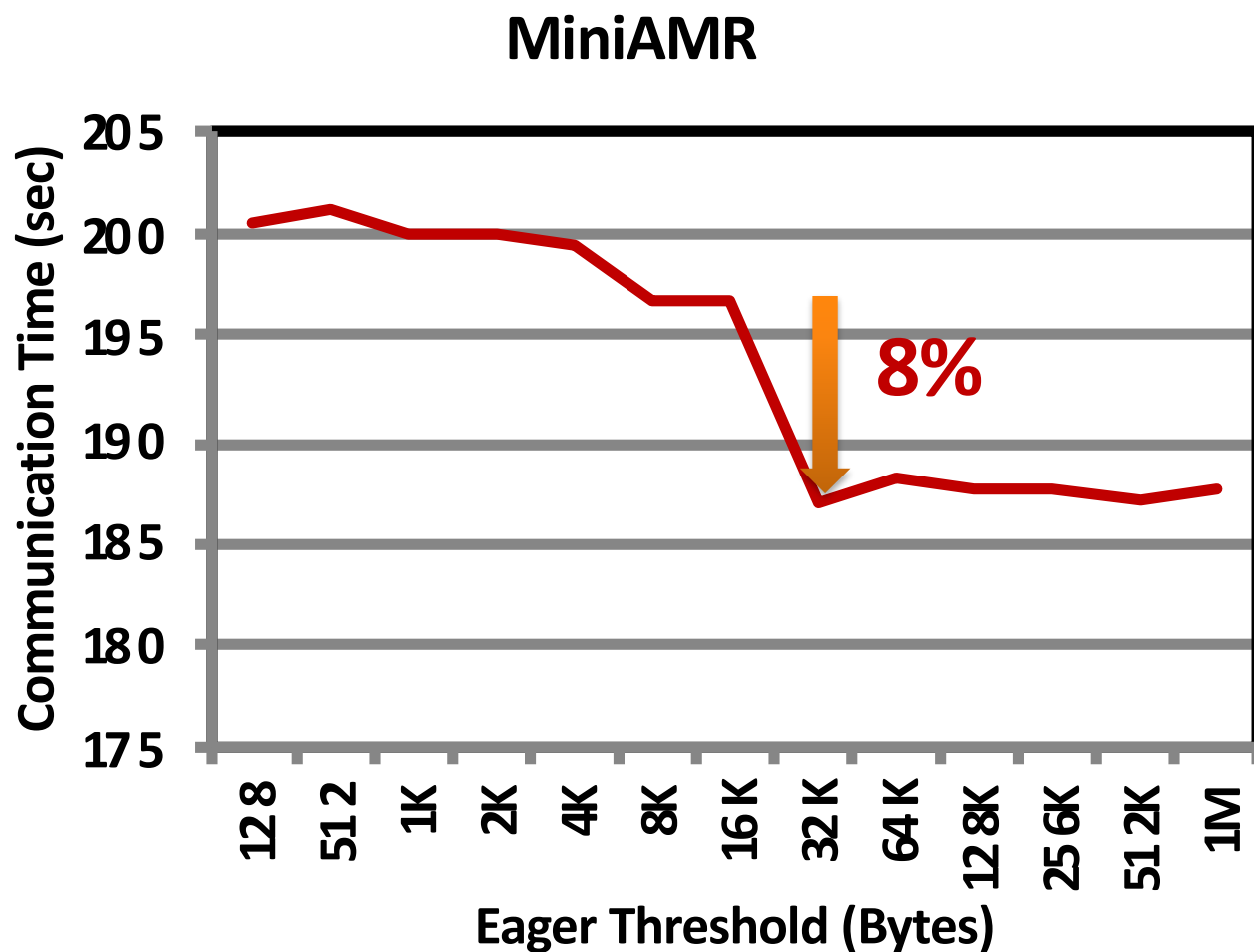
# Amber: Impact of Tuning Eager Threshold



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 2.2
- MVAPICH Flags used
  - `MV2_IBA_EAGER_THRESHOLD=131072`
  - `MV2_VBUF_TOTAL_SIZE=131072`
- Input files used
  - Small: [MDIN](#)
  - Large: [PMTOP](#)

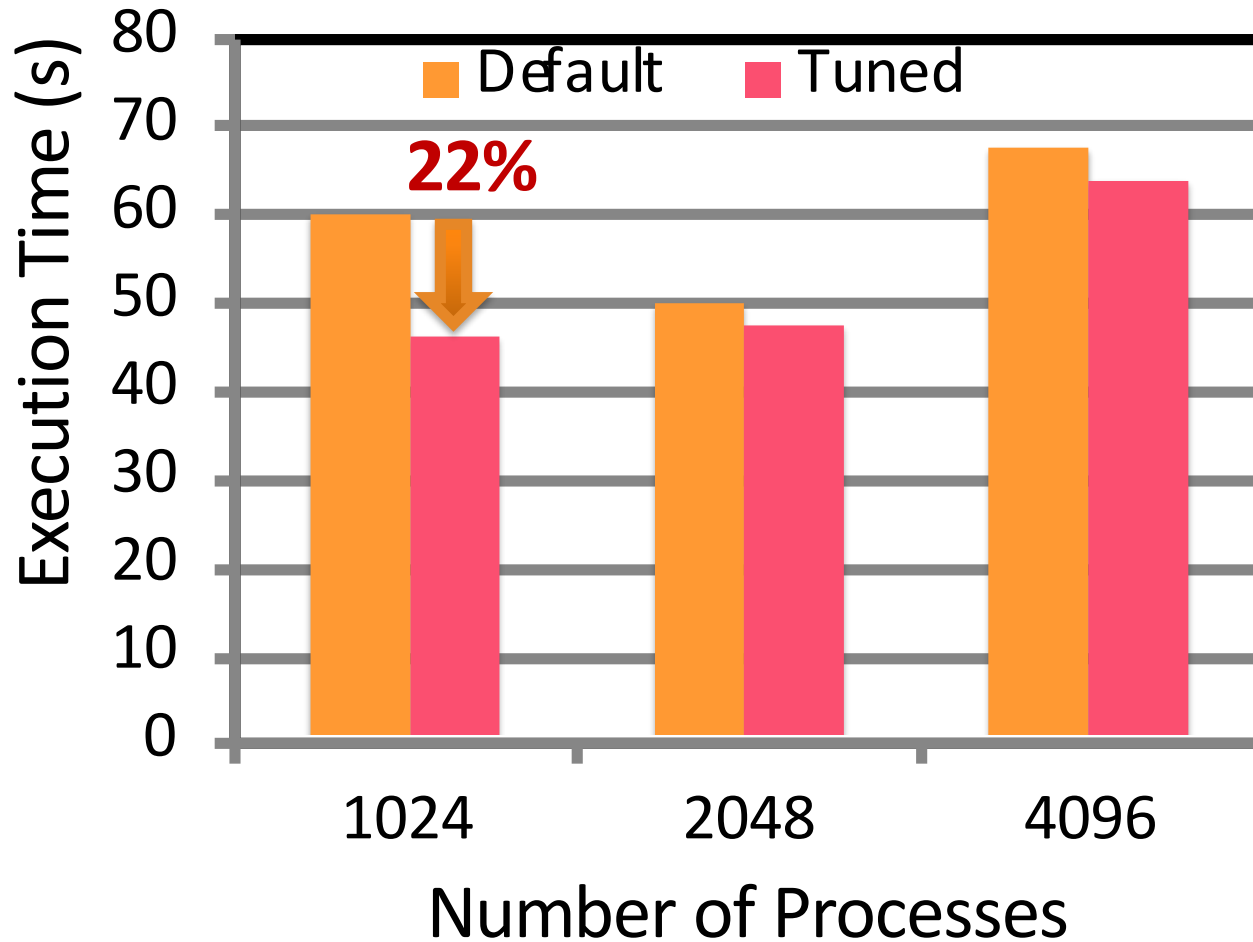
# MiniAMR: Impact of Tuning Eager Threshold



- 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 2.2
- MVAPICH Flags used
  - `MV2_IBA_EAGER_THRESHOLD=32768`
  - `MV2_VBUF_TOTAL_SIZE=32768`

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

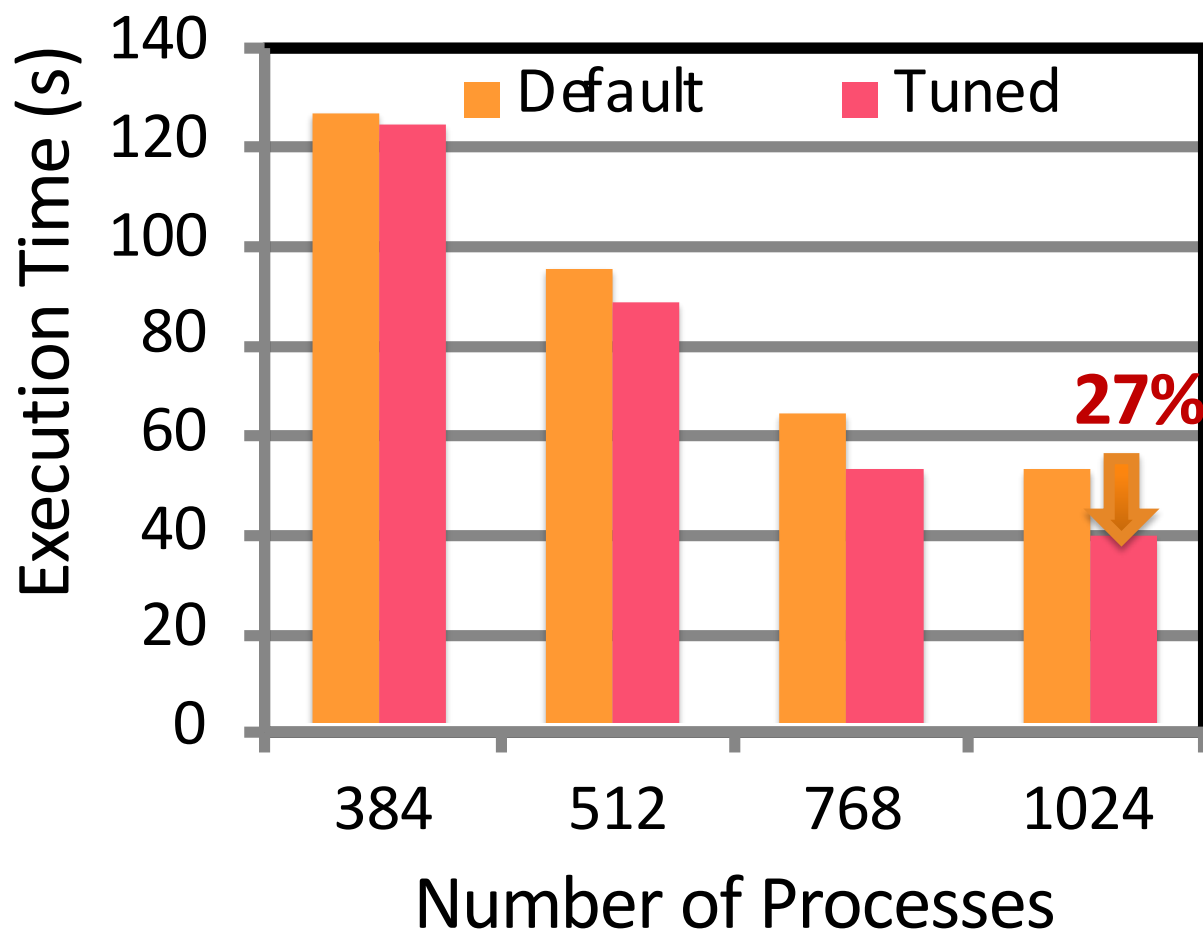
## SMG2000: Impact of Tuning Transport Protocol



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

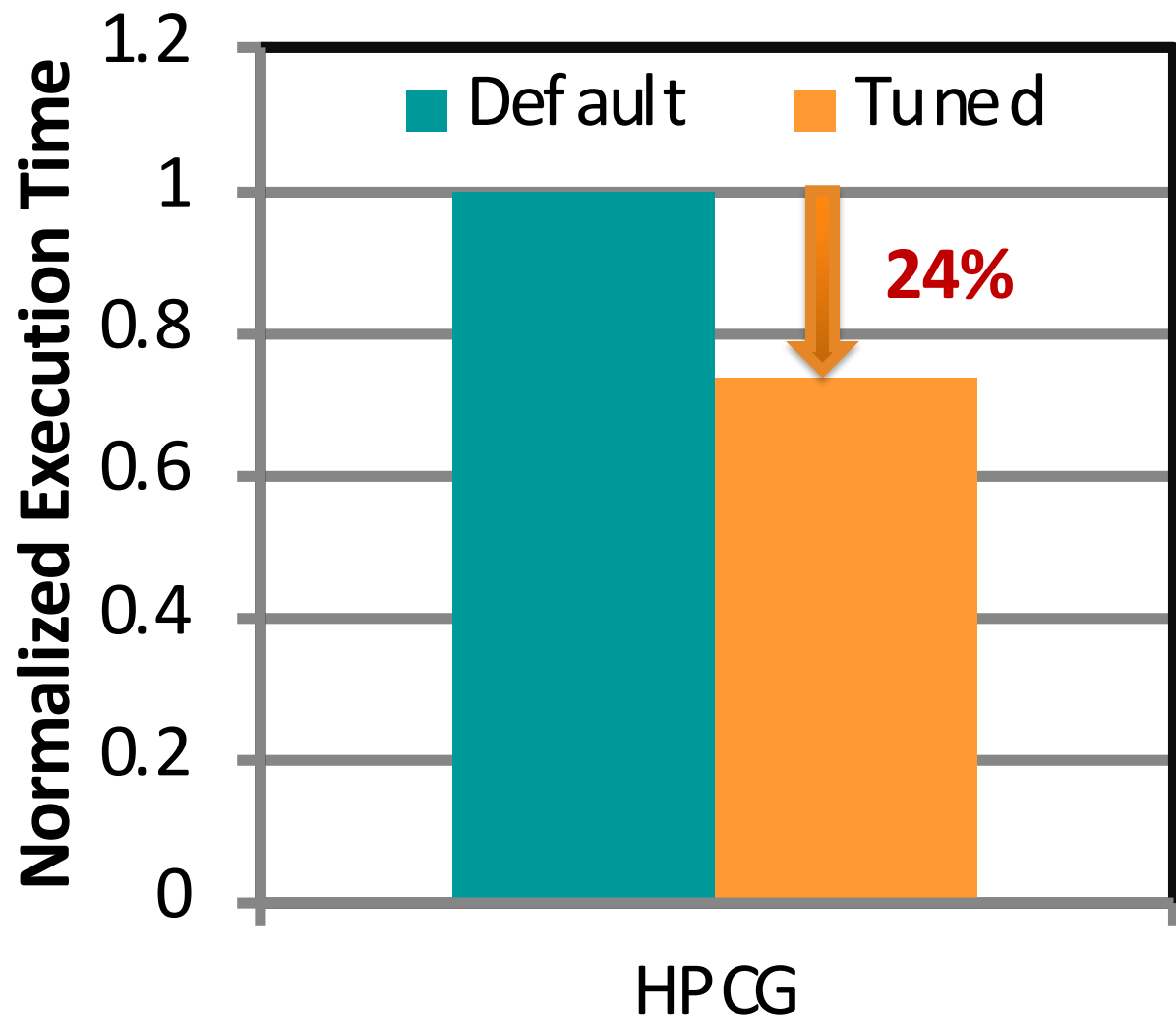
## Neuron: Impact of Tuning Transport Protocol



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 2.2
- 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.

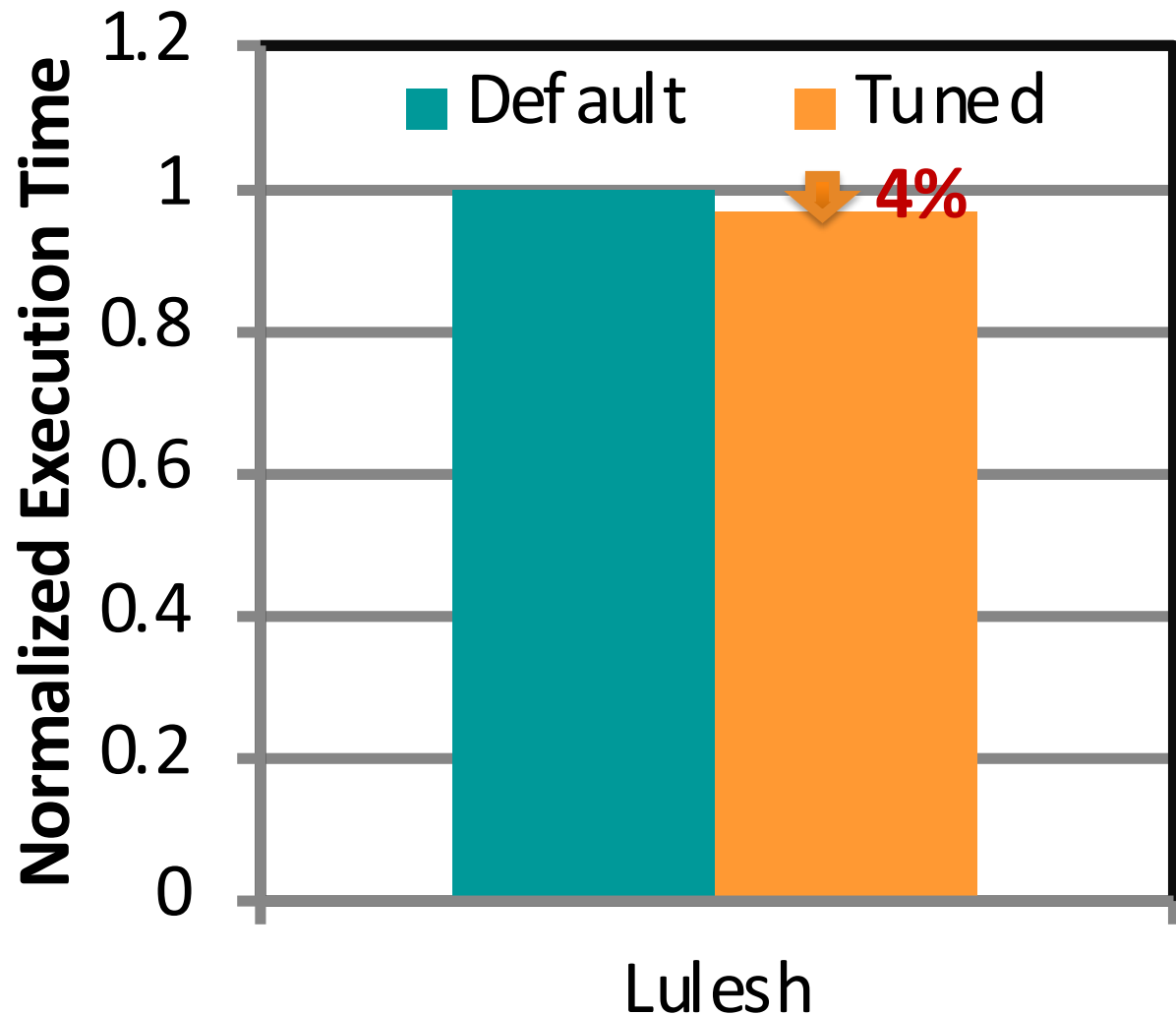
# HPCG: Impact of Collective Tuning for MPI+OpenMP Programming Model



- 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

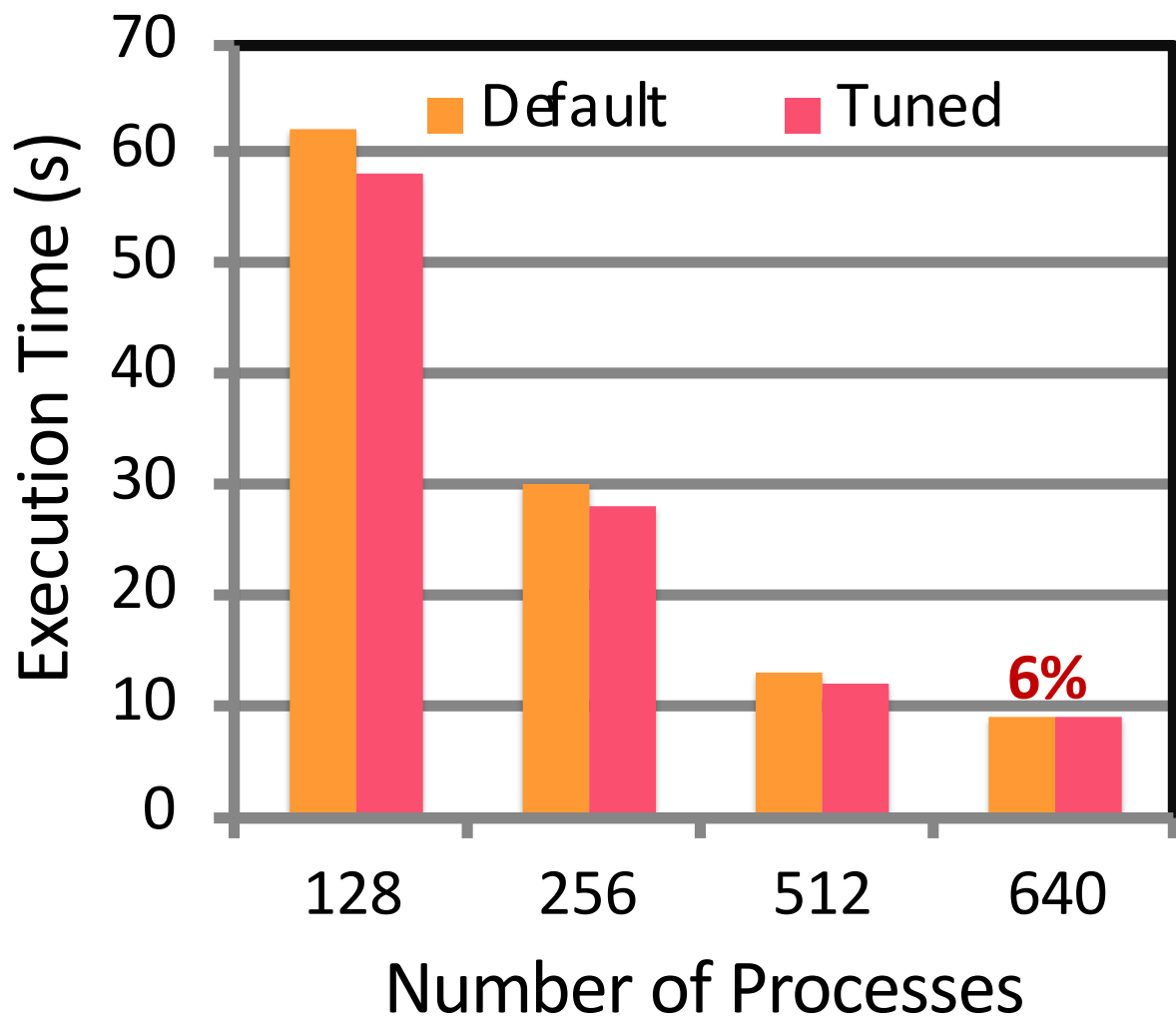
# LULESH: Impact of Collective Tuning for MPI+OpenMP Programming Model



- 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

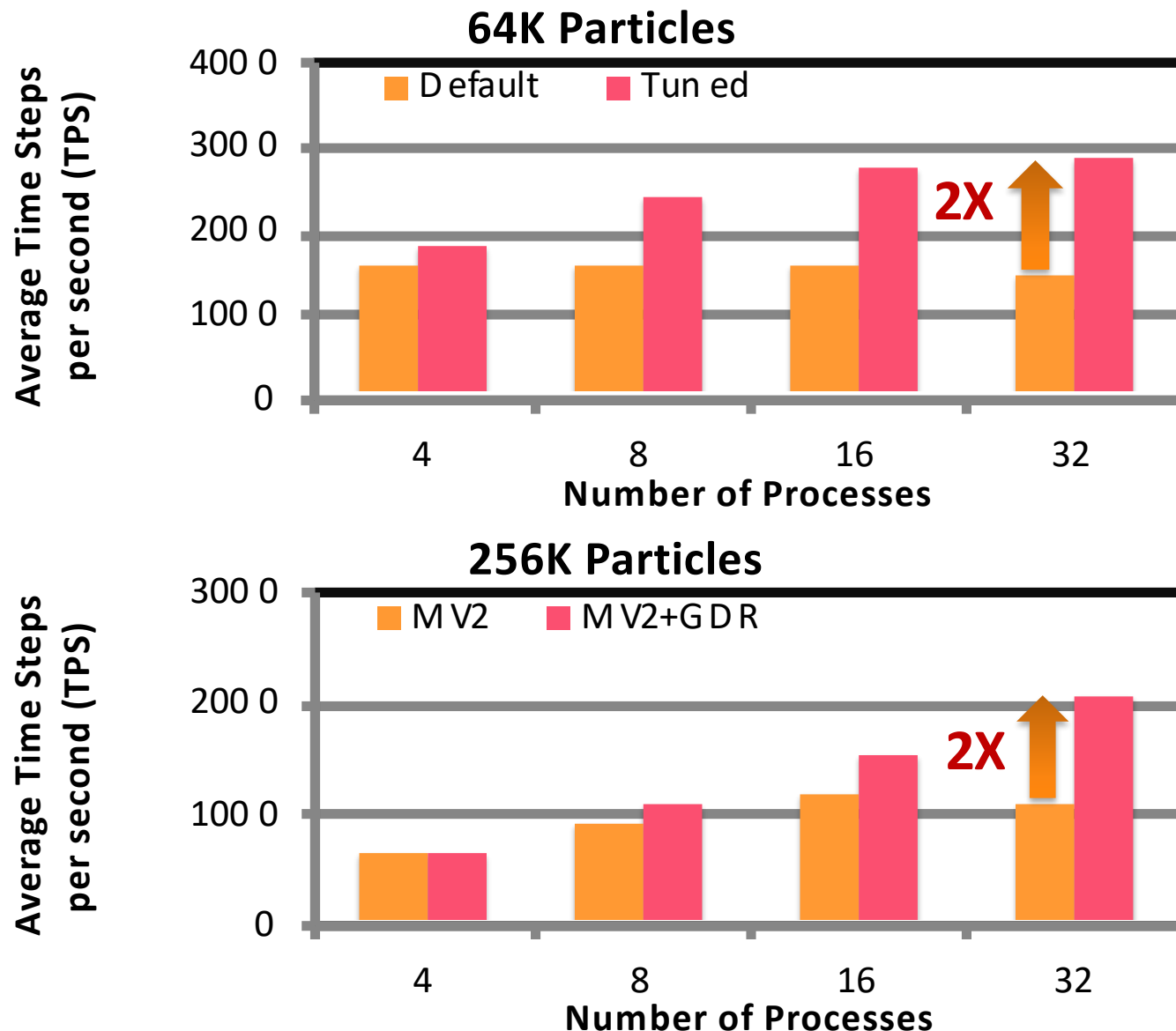


Data Submitted by Mingzhe Li @ OSU

- 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 2.2
- 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.



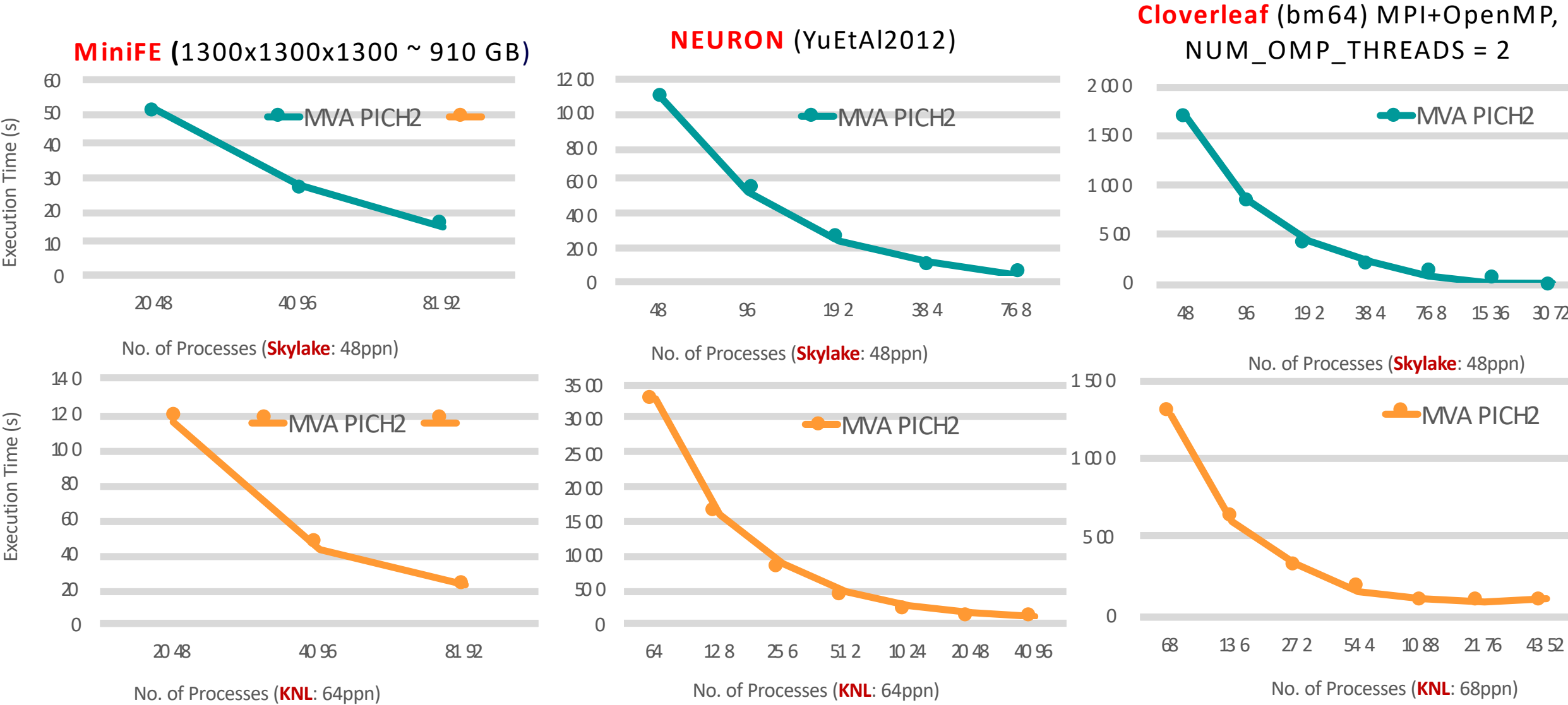
# HOOMD-blue: Impact of GPUDirect RDMA Based Tuning



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 2.2
- 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

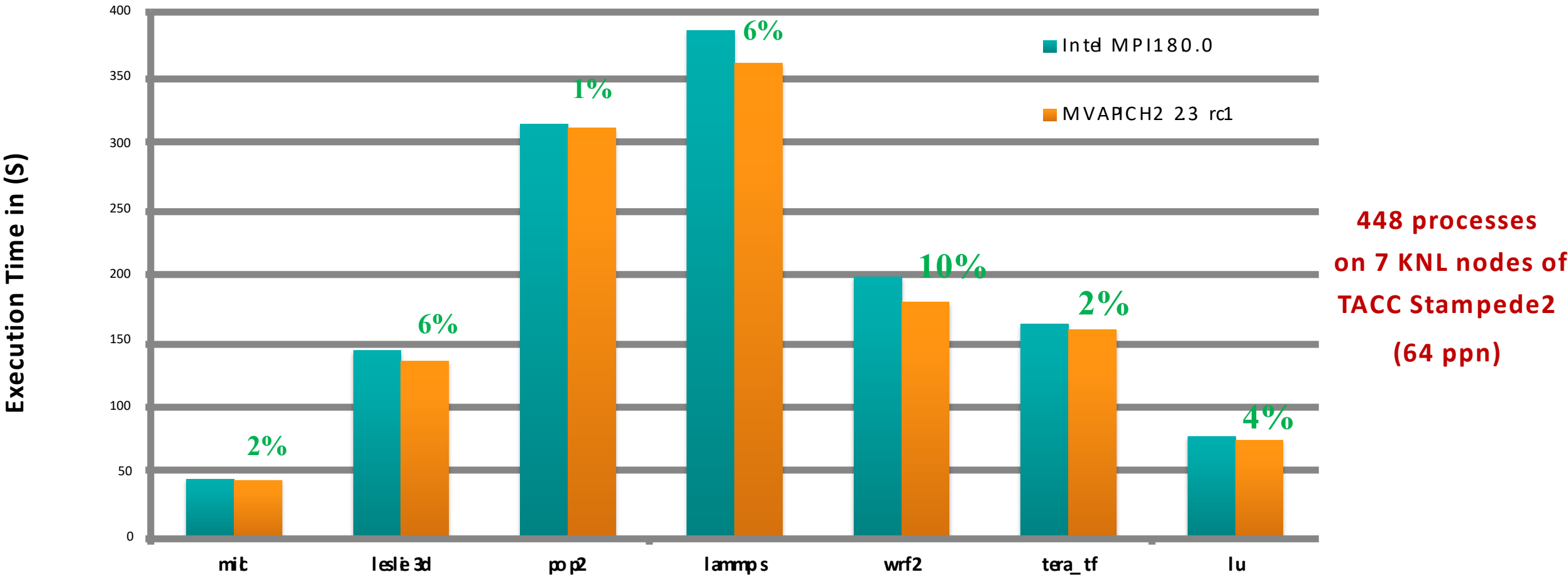
# Application Scalability on Skylake and KNL with Omni-Path



Courtesy: Mahidhar Tatineni @SDSC, Dong Ju (DJ) Choi@SDSC, and Samuel Khuvis@OSC ---- Testbed: TACC Stampede2 using MVAPICH2-2.3b

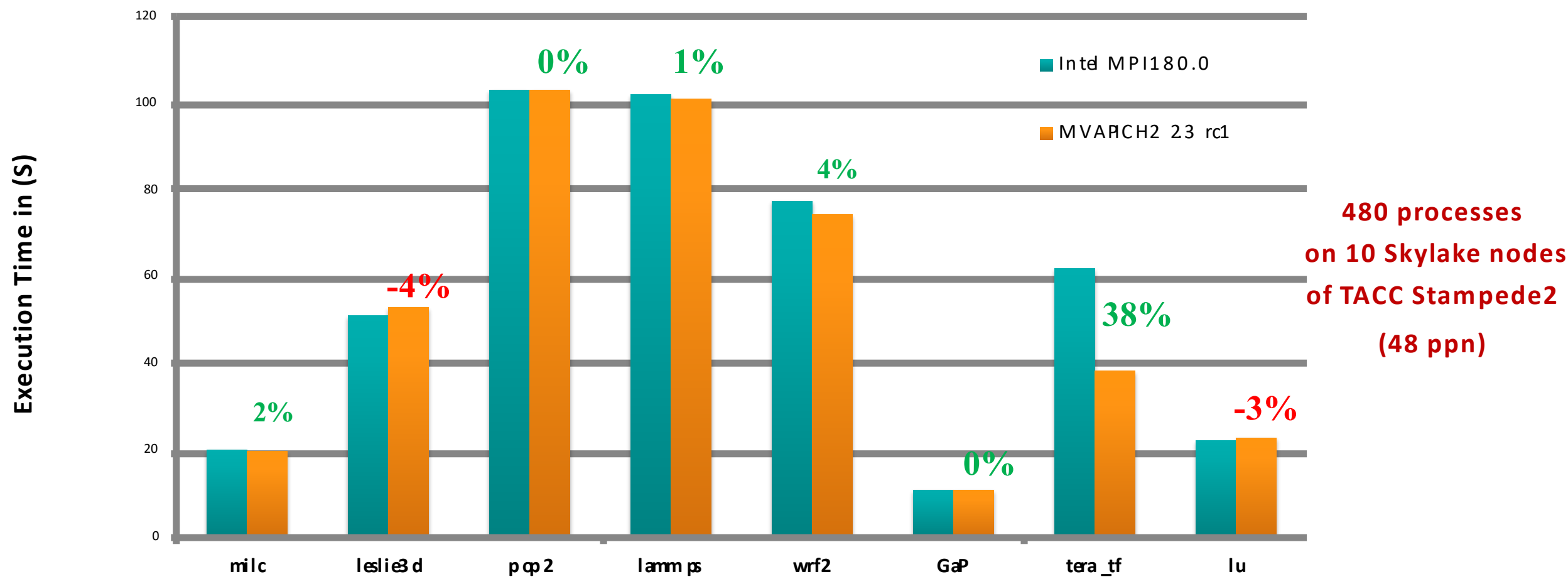
Runtime parameters: MV2\_SMPI\_LENGTH\_QUEUE=524288 PSM2\_MQ\_RNDV\_SHM\_THRESH=128K PSM2\_MQ\_RNDV\_HFI\_THRESH=128K

# Performance of SPEC MPI 2007 Benchmarks (KNL + Omni-Path)



Mvapich2 outperforms Intel MPI by up to 10%

# Performance of SPEC MPI 2007 Benchmarks (Skylake + Omni-Path)



MVAPICH2 outperforms Intel MPI by up to 38%

# MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1-10M 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
  - Tag Matching\*
  - Adapter Memory\*
- Enhanced communication schemes for upcoming architectures
  - 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 FT support
- Support for \* features will be available in future MVAPICH2 Releases

# Thank You!



Network-Based Computing Laboratory

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



The MVAPICH2 Project

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