# Optimization and Tuning Collectives in MVAPICH2

MVAPICH2 User Group (MUG) Meeting

by

**Hari Subramoni**

The Ohio State University

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

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

# Outline

- **Introduction**
- Collective Communication in MVAPICH2
- Tuning Collective communication operations in MVAPICH2
  - Benefits of using Hardware Multicast
  - Effects of  Shared-memory based communication
  - Tuning the performance of MPI_Allgather
  - Benefits of using kernel-assisted schemes for collectives
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- Summary

# MPI Collective Operations

- MPI Standard defines collective communication primitives to implement group communication operations
- Collectives such as MPI_Bcast, MPI_Allreduce are commonly used across parallel applications, owing to their ease of use and performance portability
- Processor and network architecture is constantly evolving – multi-core/many-core architectures, InfiniBand FDR, etc.
- Critical to design new algorithms to implement and optimize collective operations on emerging systems

# MPI Collective Operations

- Performance of collective operations is sensitive to various factors:
  - Message length
  - Number of processes in the communicator
  - Processor architecture
    - Number of CPU sockets per node
    - Number of cores per CPU socket
  - Network technology
    - InfiniBand DDR/QDR/FDR
    - Single-Rail, Multi-Rail
- Important to carefully tune various designs to implement collective operations efficiently

# Conventional Collective Communication Algorithms

- Basic collective communication algorithms:
  - Binomial Tree (MPI_Bcast, MPI_Reduce, MPI_Scatter, MPI_Gather)
  - Recursive Doubling (MPI_Allreduce, MPI_Allgather)
  - Ring Exchange (MPI_Allgather)
  - Bruck Algorithm, Pairwise Exchange (MPI_Alltoall)

- Combining basic algorithms to implement more complex schemes:
  - Scatter-Allgather (MPI_Bcast)
  - Reduce-Scatter-Allgather (MPI_Allreduce)
  - Reduce-Scatter-Gather (MPI_Reduce)

# Collective Communication Algorithms for Multicore Systems

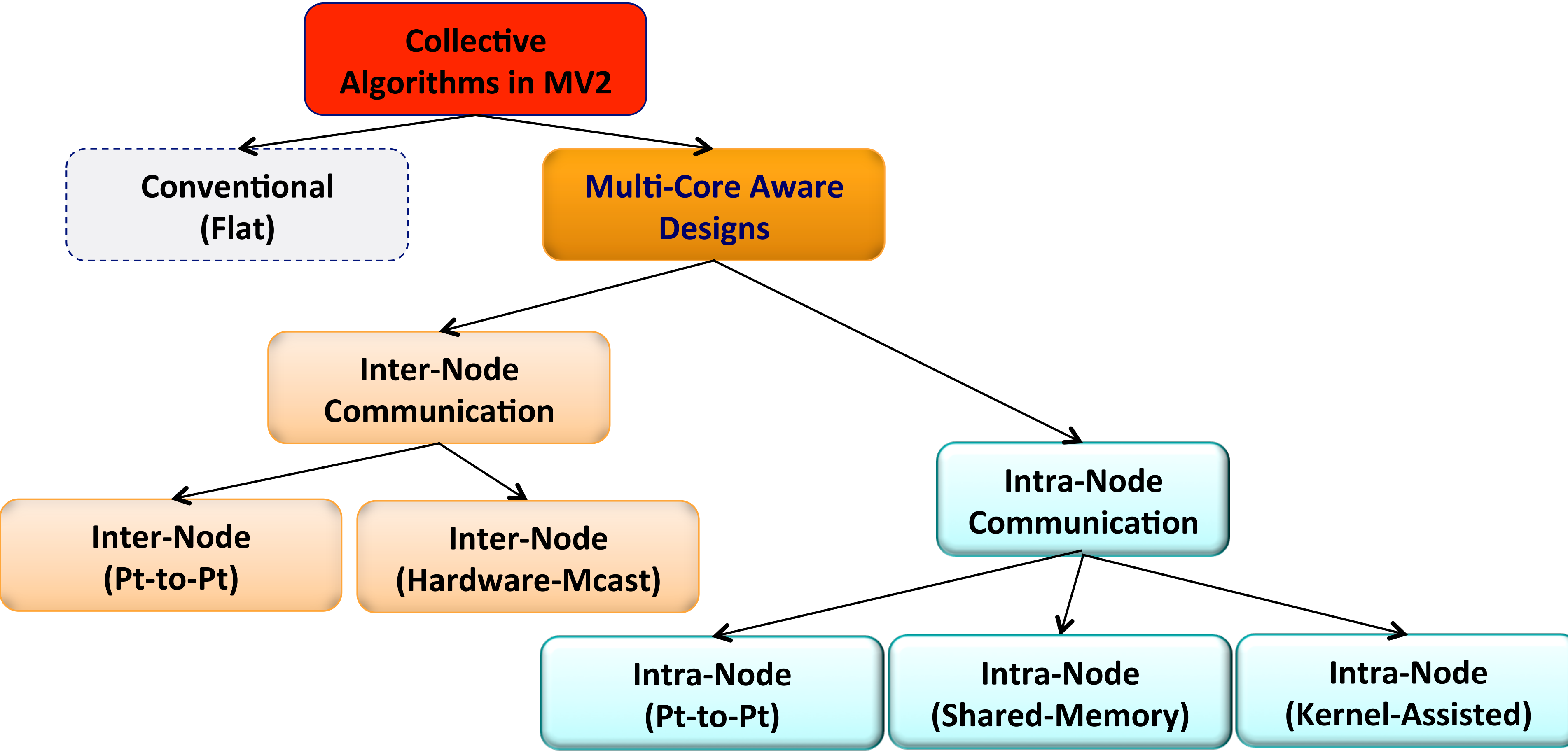
- Basic Multi-core Aware designs:
  - Hierarchical communicator system
  - Intra-node communicator
    - Includes all processes that share the same address space
    - Lowest rank process is the node-leader
  - Inter-leader communicator
  - MPICH implements many collectives in a hierarchical manner.

    Intra-node phases are implemented via simple point-to-point operations
  - Critical to design efficient schemes on emerging multi-/many- core systems

# Outline

- **Introduction**
- **Collective Communication in MVAPICH2**
- Tuning Collective communication operations in MVAPICH2
  - Benefits of using Hardware Multicast
  - Effects of  Shared-memory based communication
  - Tuning the performance of MPI_Allgather
  - Benefits of using kernel-assisted schemes for collectives
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- Summary

# 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)

# Collective Communication in MVAPICH2

- MVAPICH2 relies on highly optimized and tuned shared-memory based optimizations for several important collectives
  **MPI_Bcast, MPI_Reduce, MPI_Allreduce, MPI_Barrier**
- MVAPICH2 also uses a combination of hierarchical designs to optimize **MPI_Gather** and **MPI_Scatter**
- An optimized variant of recursive-doubling is used to improve the latency of **MPI_Allgather** collective
- MVAPICH2 also relies on hardware multicast based designs to optimize **MPI_Bcast** and **MPI_Scatter**
- Kernel-assisted mechanisms are also being used to optimize performance of collectives, such as **MPI_Allgather**, in MVAPICH2

# OSU MicroBenchmarks for Collectives

| Executable | Latency Benchmark |
|---|---|
| osu_allgather | MPI_Allgather |
| osu_allgatherv | MPI_Allgatherv |
| osu_allreduce | MPI_Allreduce |
| osu_alltoall | MPI_Alltoall |
| osu_alltoallv | MPI_Alltoallv |
| osu_barrier | MPI_Barrier |
| osu_bcast | MPI_Bcast |
| osu_gather | MPI_Gather |
| osu_gatherv | MPI_Gatherv |
| osu_reduce | MPI_Reduce |
| osu_reduce_scatter | MPI_Reduce_scatter |
| osu_scatter | MPI_Scatter |
| osu_scatterv | MPI_Scatterv |

# OSU MicroBenchmarks for Collectives

- Running OMB collective benchmarks:

  **mpirun_rsh –np # -hostfile hosts ./<benchmark name>**

  Reports the average time taken to complete the collective operation

  Latency averaged across all processes, across 1,000 iterations

- Additional Run-time options:
  - Users may choose to view additional latency statistics, such as Max and Min latency, by enabling the **``–f''** option

    **mpirun_rsh –np # -hostfile hosts ./<benchmark name> -f**
  - **``-m''** option allows users to specify the maximum amount of memory and the message lengths used by the benchmark

    **mpirun_rsh –np # -hostfile hosts ./<benchmark name> -m #**

    (Useful to prevent seg-faults with large messages, on systems with limited memory)

# Outline

- Introduction
- Collective Communication in MVAPICH2
- **Tuning Collective communication operations in MVAPICH2**
  - **Benefits of using Hardware Multicast**
  - Effects of  Shared-memory based communication
  - Tuning the performance of MPI_Allgather
  - Benefits of using kernel-assisted schemes for collectives
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- Summary

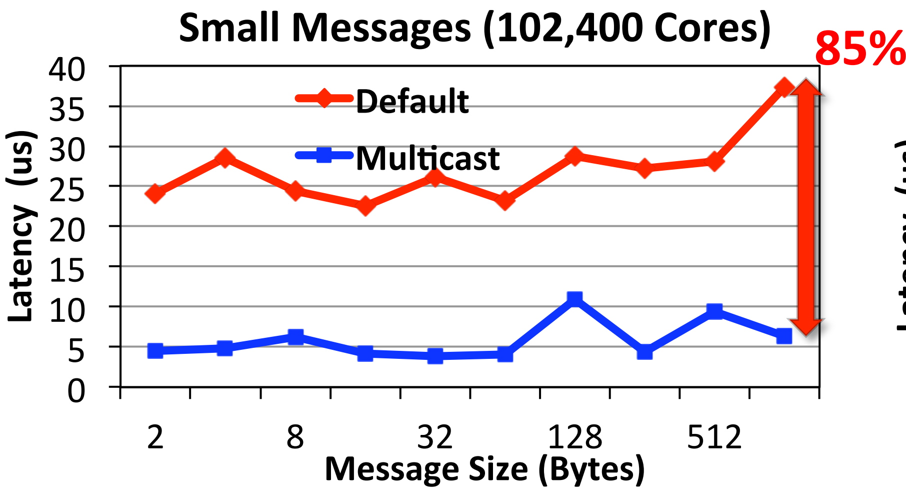
# Tuning Collective Operations in MVAPICH2

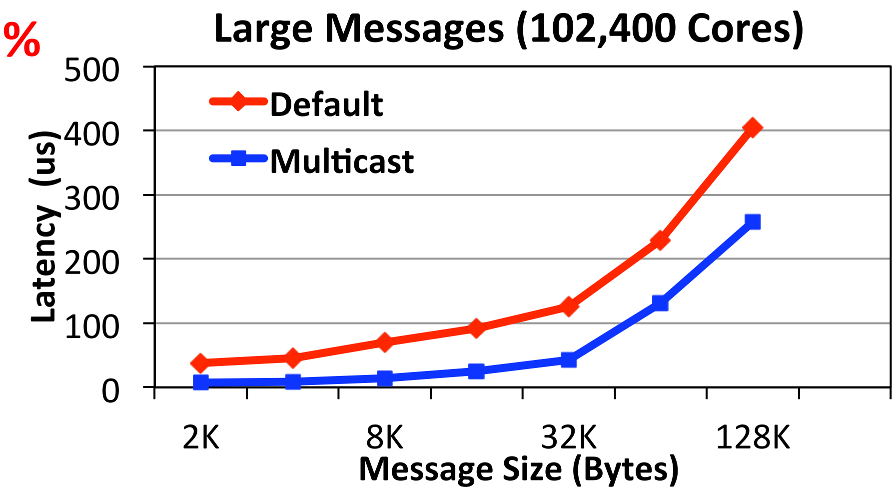
- MVAPICH2 relies on several shared-memory-based, hardware-multicast-based and kernel assisted designs to optimize collectives

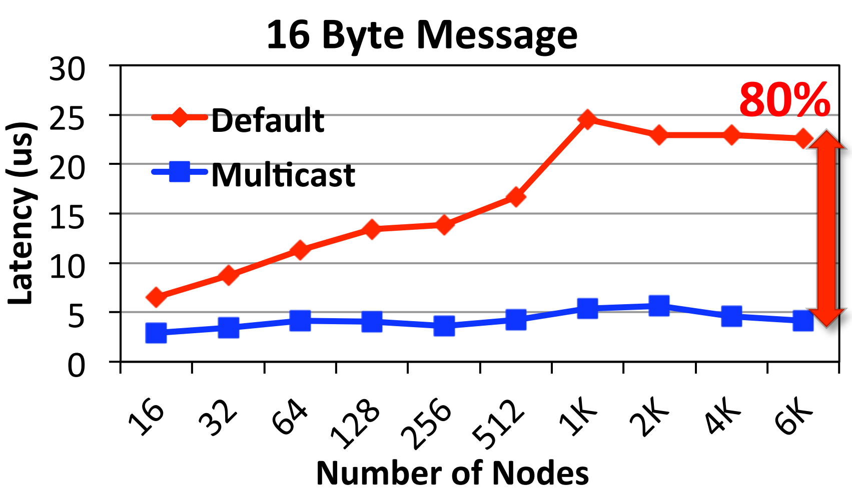
- Critical to tune such designs to ensure low communication latency for collective operations, across various message lengths, number of processes, different systems.

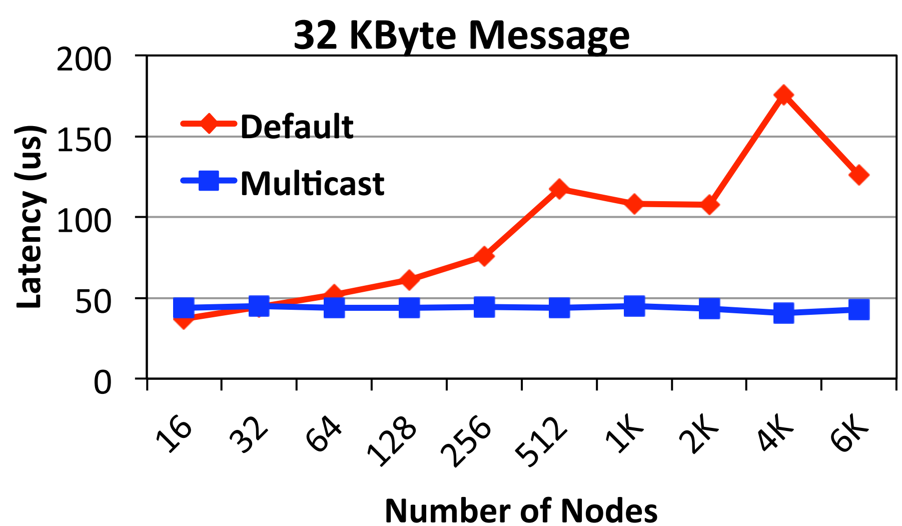
- Collective designs are broadly classified as:
  - **Communication algorithm** (Binomial, Recursive-Doubling, etc.)
  - **Communication mechanism** (Shared-memory, Limic, Mcast, etc.)

# Hardware Multicast-aware MPI_Bcast on TACC Stampede



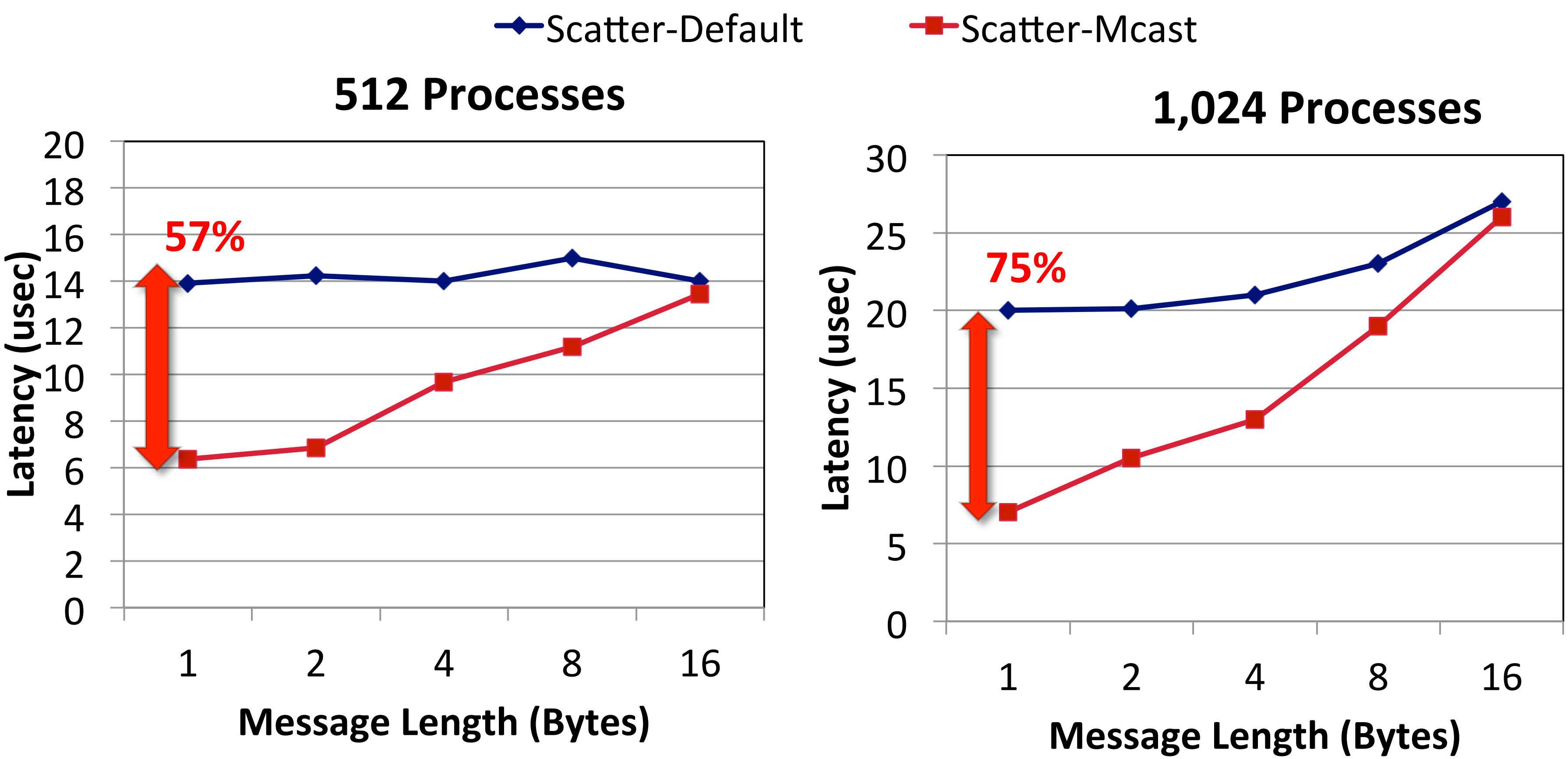

- MCAST-based designs improve latency of MPI_Bcast by up to **85%**
- Use MV2_USE_MCAST =1 to enable MCAST-based designs

# Enabling Hardware Multicast-aware

- Multicast is applicable to
  - MPI_Bcast
  - MPI_Scatter
  - MPI_Allreduce

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

# MPI_Scatter - Benefits of using Hardware-Mcast




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

# Outline

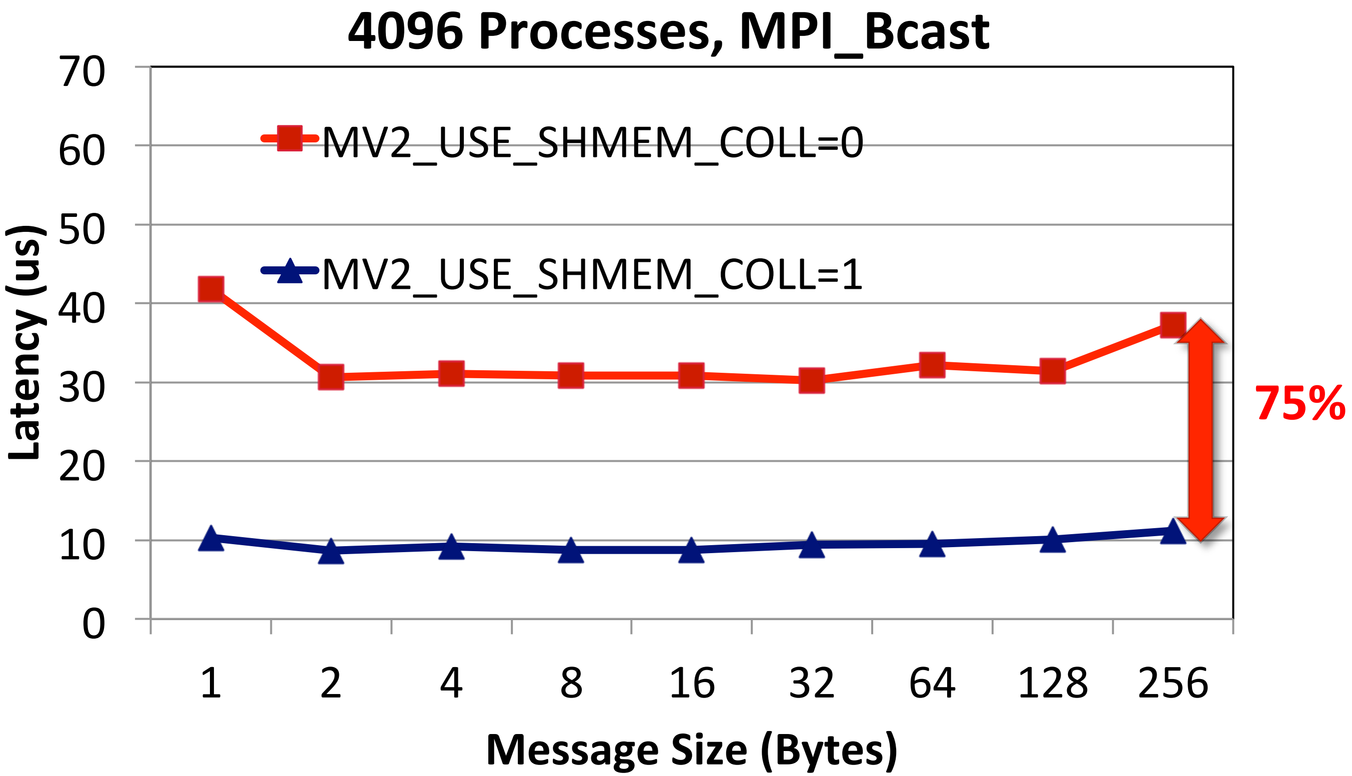
- Introduction
- Collective Communication in MVAPICH2
- **Tuning Collective communication operations in MVAPICH2**
  - Benefits of using Hardware Multicast
  - **Effects of Shared-memory based communication**
  - Tuning the performance of MPI_Allgather
  - Benefits of using kernel-assisted schemes for collectives
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- Summary

# Effects of Shared-memory based communication

- Shared-Memory specific data structures created to optimize collective communication
- Tunable parameter
- Applicable to all MPI Collectives (Except vector variants)
- For optimal performance this feature should be enabled

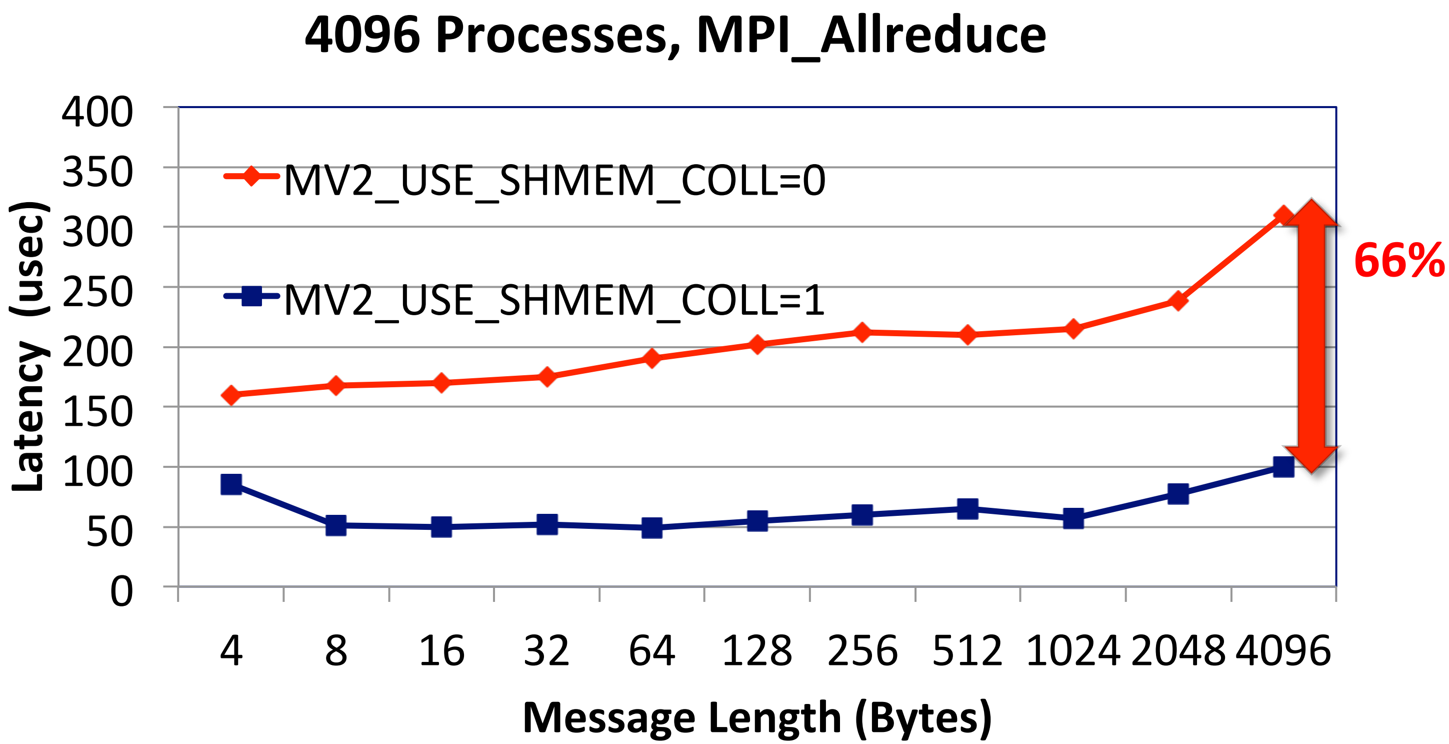
| Parameter | Description | Default Nature |
|---|---|---|
| MV2_USE_SHMEM_COLL = 1 | Enables shared memory optimizations for collectives | Enabled |

# Effects of Shared-memory based communication



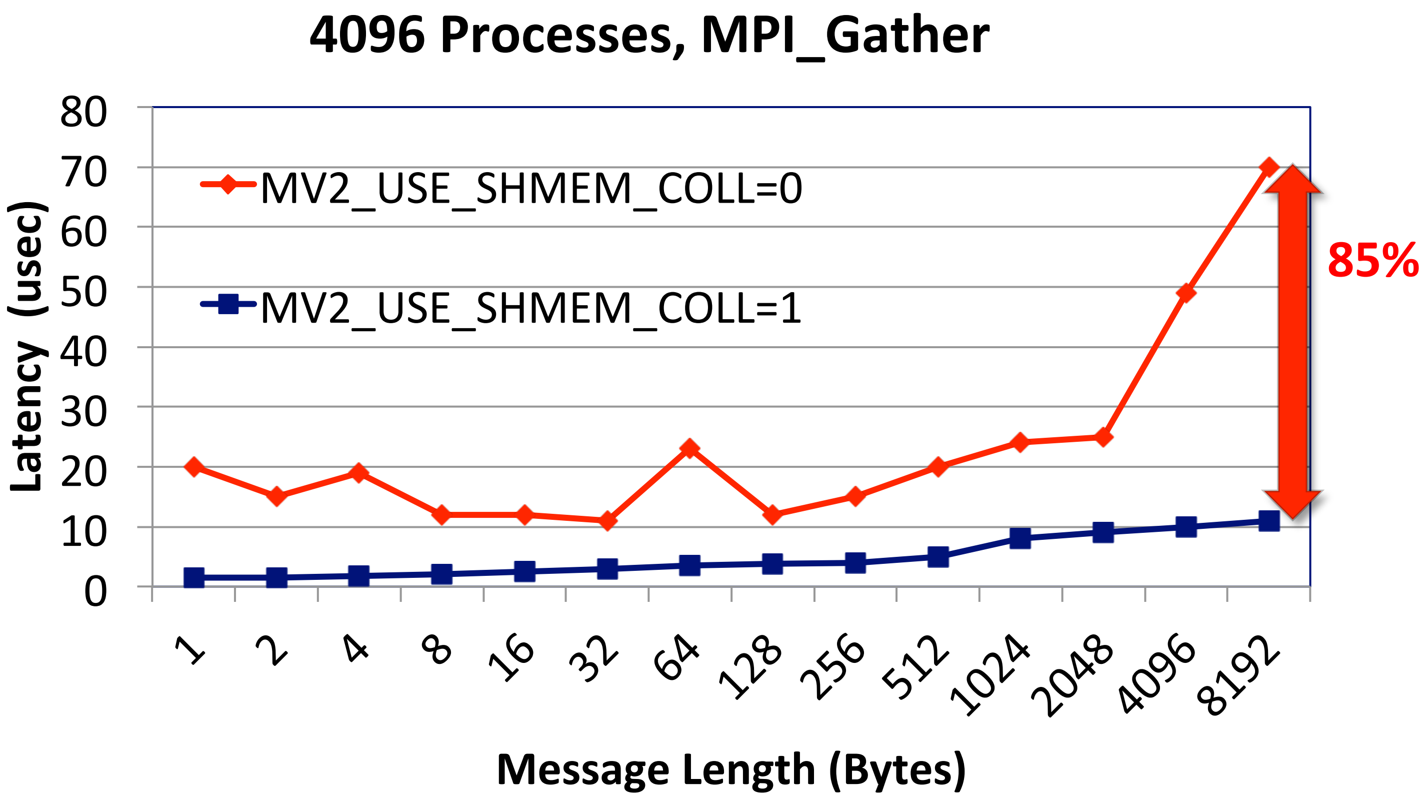

- MV2_USE_SHMEM_COLL=1 activates the use of shared-memory-based collective optimizations
- Latency of MPI_Bcast, with 4,096 MPI processes improves by up to 75% through optimized, tuned shared-memory-based designs

# Effects of Shared-memory based communication




- MV2_USE_SHMEM_COLL=1 activates the use of shared-memory-based collective optimizations
- Latency of MPI_Allreduce improves by up to **66%** through optimized, tuned shared-memory-based designs

# Effects of  Shared-memory based communication




- MV2_USE_SHMEM_COLL=1 activates the use of shared-memory-based  collective optimizations
- Latency of MPI_Gather  improves by up to **85%** through optimized, tuned shared-memory-based designs

# Outline

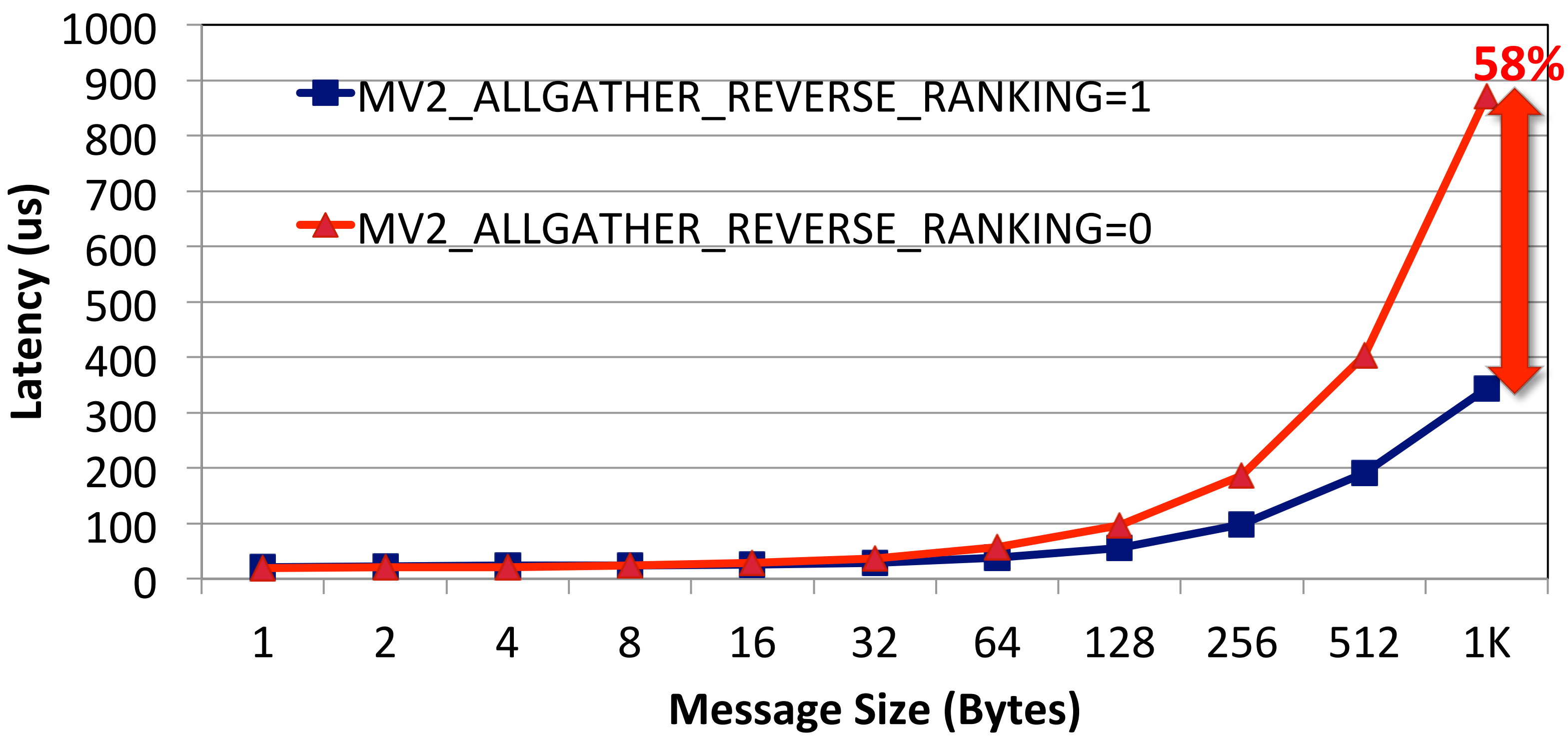
- Introduction
- Collective Communication in MVAPICH2
- Tuning Collective communication operations in MVAPICH2
  - Benefits of using Hardware Multicast
  - Effects of  Shared-memory based communication
  - **Tuning the performance of MPI_Allgather**
  - Benefits of using kernel-assisted schemes for collectives
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- Summary

# Tuning the performance of MPI_Allgather

- MV2_ALLGATHER_REVERSE_RANKING=1 enables optimizations for MPI_Allgather with small and medium length messages
- Latency of MPI_Allgather, with 2,048 MPI processes improves by up to **58%** through optimized and tuned designs

| Parameter | Description |
|---|---|
| MV2_ALLGATHER_REVERSE_RANKING=1 | Enables allgather reverse ranking optimization |

# Tuning the performance of MPI_Allgather




- MV2_ALLGATHER_REVERSE_RANKING=1 enables optimizations for MPI_Allgather with small and medium length messages
- Latency of MPI_Allgather, with 2,048 MPI processes improves by up to **58%** through optimized and tuned designs

# Outline

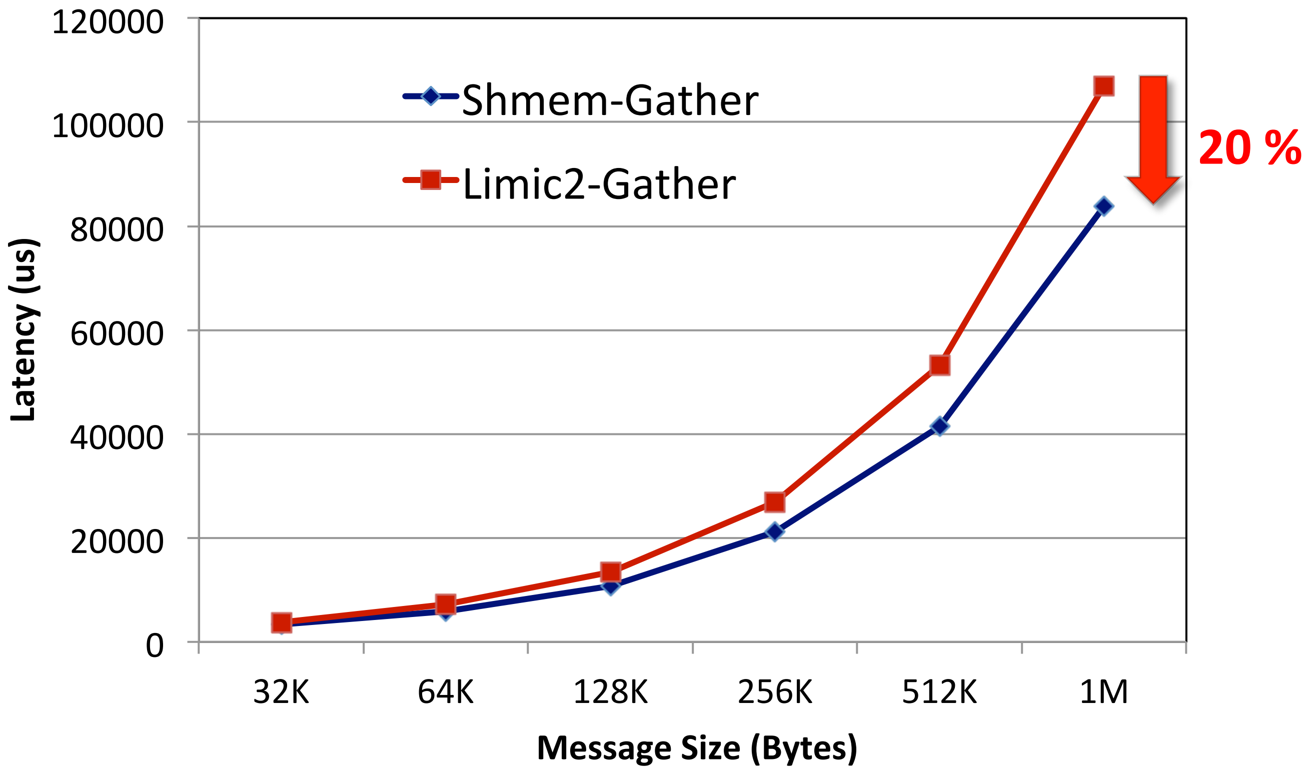
- Introduction
- Collective Communication in MVAPICH2
- Tuning Collective communication operations in MVAPICH2
  - Benefits of using Hardware Multicast
  - Effects of  Shared-memory based communication
  - Tuning the performance of MPI_Allgather
  - **Benefits of using kernel-assisted schemes for collectives**
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- Summary

# Collective Tuning – Using LiMIC2

- LIMIC2 module enables MVAPICH2 to use kernel-assisted data transfers
- MPI Jobs often run in fully subscribed mode and can involve significant intranode communication
- Such patterns are good indicators of enabling LIMIC2 to speedup intranode tranfers
- Collectives such as MPI_Allgather benefits from the use of LIMIC2

| Parameter | Description | Default Nature |
| --- | --- | --- |
| MV2_SMP_USE_LIMIC2 | Enables shared memory optimizations for collectives | Enabled |
| --with-limic2 | Configure flag | disabled |

# Collective Tuning – Using LiMIC2
# Allgather Case Study

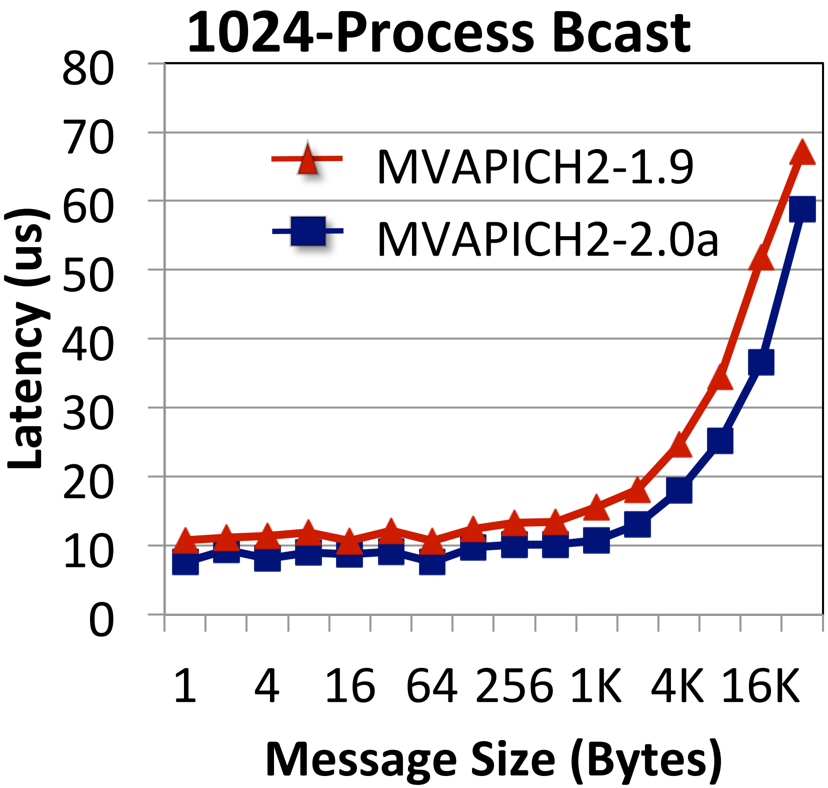


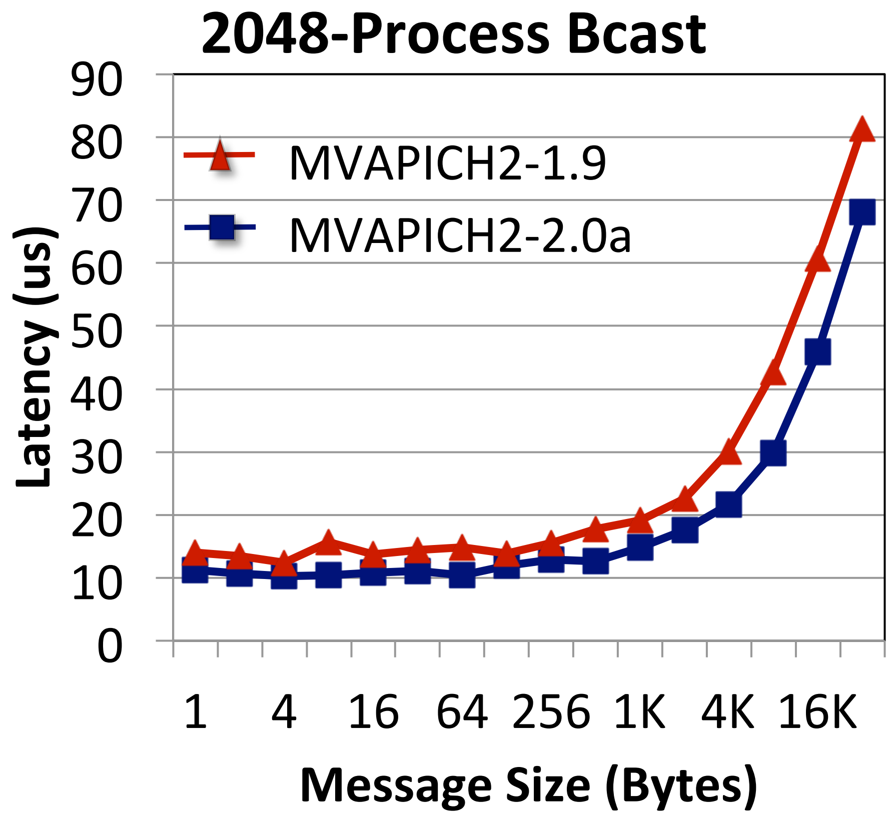

- MPI_Allgather relies on the ring-exchange pattern for large messages. Critical to optimize intra-node phases of the ring exchange on multi-core systems
- Zero-copy LiMIC transfers improve performance by up to **20%** for large message MPI_Allgather operations

# Outline

- Introduction
- Collective Communication in MVAPICH2
- Tuning Collective communication operations in MVAPICH2
  - Benefits of using Hardware Multicast
  - Effects of  Shared-memory based communication
  - Tuning the performance of MPI_Allgather
  - Benefits of using kernel-assisted schemes for collectives
- **Improved Bcast in MVAPICH2-2.0a**
- **Non-blocking collectives in MVAPICH2**
- Summary

# Improved Bcast in MVAPICH2-2.0a




- Average of **25%** improvement for 1024 processes and an average of **22%** improvement for 2048 processes

# Non-Blocking Collectives in MVAPICH2

- MVAPICH2 supports for MPI-3 Non-Blocking Collective communication and Neighborhood collective communication primitives
  - MVAPICH2 1.9 and MVPICH2 2.0a
- MPI-3 collectives in MVAPICH2 can use either the Gen2 or the nemesis interfaces, over InfiniBand
- MVAPICH2 implements non-blocking collectives either in a multi-core-aware hierarchical manner, or via a basic flat approach
- Application developers can use MPI-3 collectives to achieve computation/communication overlap
- Upcoming releases of MVAPICH2 will include support for non-blocking collectives based on Mellanox CORE-Direct interface

# Outline

- Introduction
- Collective Communication in MVAPICH2
- Tuning Collective communication operations in MVAPICH2
  - Benefits of using Hardware Multicast
  - Effects of Shared-memory based communication
  - Tuning the performance of MPI_Allgather
  - Benefits of using kernel-assisted schemes for collectives
- Improved Bcast in MVAPICH2-2.0a
- Non-blocking collectives in MVAPICH2
- **Summary**

# Summary

- Critical to optimize and MPI collective operations on emerging multi-core systems and high-speed networks
- Necessary to tune various collective designs to ensure low communication latency on modern commodity systems
- MVAPICH2 relies on several optimized and tuned designs to deliver low communication latency for collectives
- MVAPICH2 has been tuned across several architectures, considering various processor and network architectures

# Web Pointers

NOWLAB Web Page

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

MVAPICH Web Page

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

MVAPICH