Optimization and Tuning of Job Start-up, Process Mapping, and Point-to-Point Communication in MVAPICH2

MVAPICH2 User Group (MUG) Meeting

by

Hari Subramoni & Sreeram Potluri

The Ohio State University

E-mail: {subramon.potluri}@cse.ohio-state.edu

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

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

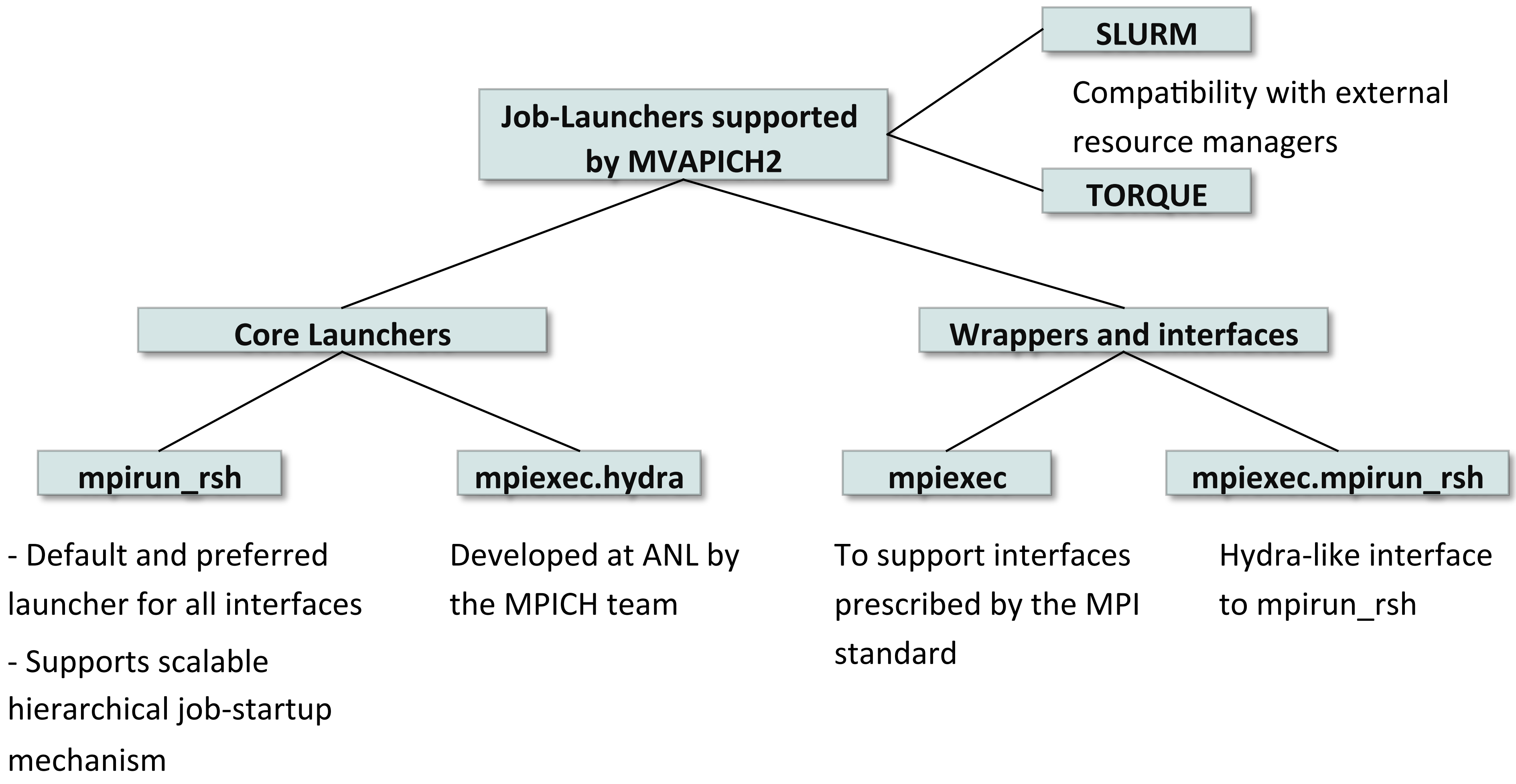
Outline

- Job Startup
- Process Mapping
  - Preset process-binding policies
  - User-defined binding
- Pt-to-pt Inter-node Communication
  - Eager and Rendezvous Protocols
  - RDMA Fast Path
- Pt-to-pt Intra-node Communication
  - Shared-memory and LiMIC2/CMA based Communication
  - Architecture-based Tuning
  - Improved Multi-Pair Latency and Message Rate in MV2 2.0a
- One-sided Communication

Job-Launch Mechanisms for MPI

- HPC clusters continue to grow with fatter compute nodes
- Increased focus on the scalability of programming models and libraries
- Job launch mechanisms have not received enough attention and have scaled poorly over the last few years
- “mpirun_rsh” designed to be a Scalable and Extensible Launching Architecture (ScELA) for MPI applications

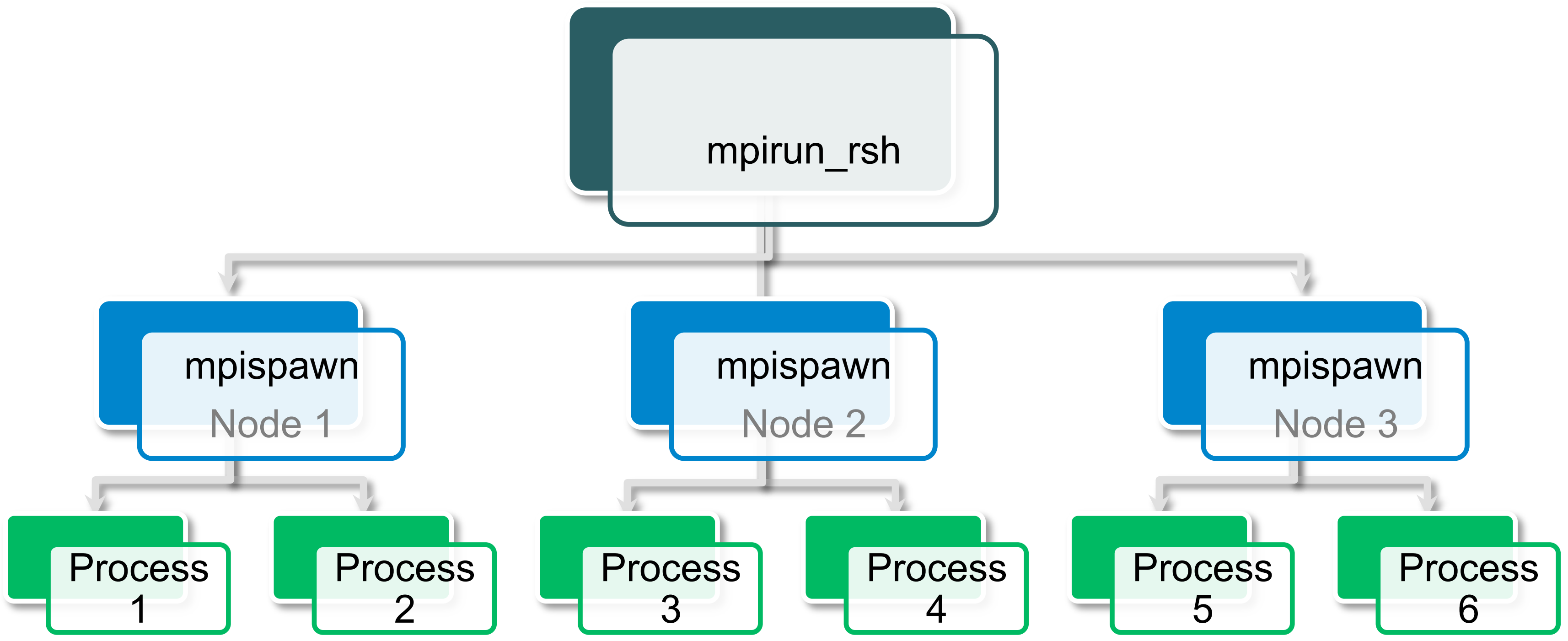
Job-Launchers supported by MVAPICH2


Job-Launchers supported by MVAPICH2
SLURM
Compatibility with external resource managers
TORQUE
Core Launchers
Wrappers and interfaces
mpirun_rsh
mpiexec.hydra
mpiexec
mpiexec.mpirun_rsh
- Default and preferred launcher for all interfaces
- Supports scalable hierarchical job-startup mechanism
Developed at ANL by the MPICH team
To support interfaces prescribed by the MPI standard
Hydra-like interface to mpirun_rsh

Scalable Job-Launching with mpirun_rsh (ScELA)

- Suggested for all interfaces
  - CH3, Nemesis-IB, iWARP, RoCE, PSM, uDAPL, Shared memory-CH3, TCP/IP-CH3 and TCP/IP-Nemesis
- Tuned for scalability
  - Multi-core aware
  - PMI aggregation and information caching
  - Low-latency communication primitives
  - On-demand QP exchanges
  - File-based communication scheme for very-large jobs
  - Hierarchical job-startup
  - Ample scope for tuning based on system characteristics

Scalable Job-Launching with mpirun_rsh (ScELA)

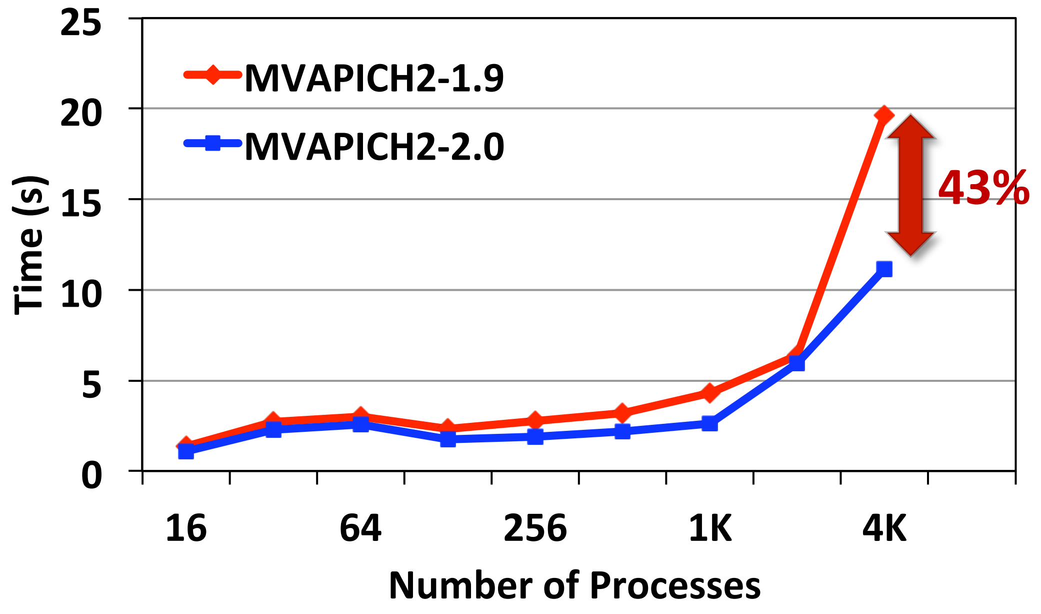


- Hierarchical launch
  - **‘mpirun_rsh’** launches ‘**mpispawn’** on compute nodes
  - mpispawns launch application processes on processor cores
- **mpispawns** interconnect to form a k-ary tree to facilitate communication
- Common communication primitives built on mpispawn tree

Tuning Job-Launch with mpirun_rsh

- MV2_MT_DEGREE
  - degree of the hierarchical tree used by mpirun_rsh
- MV2_FASTSSH_THRESHOLD
  - #nodes beyond which hierarchical-ssh scheme is used
- MV2_NPROCS_THRESHOLD
  - #nodes beyond which file-based communication is used for hierarchical-ssh during start up
- MV2_HOMOGENEOUS_CLUSTER
  - Setting it optimizes startup for homogeneous clusters
- MV2_ON_DEMAND_UD_INFO_EXCHANGE
  - To optimize start-up by exchanging UD connection info on-demand

Performance of Job Launcher on Stampede@TACC



- Several optimizations to enhance job startup performance
    - **On-demand exchange of startup related information**
- **45% reduction in time for MPI hello world program at *4K cores***
    - **Run with 16 processes per node**

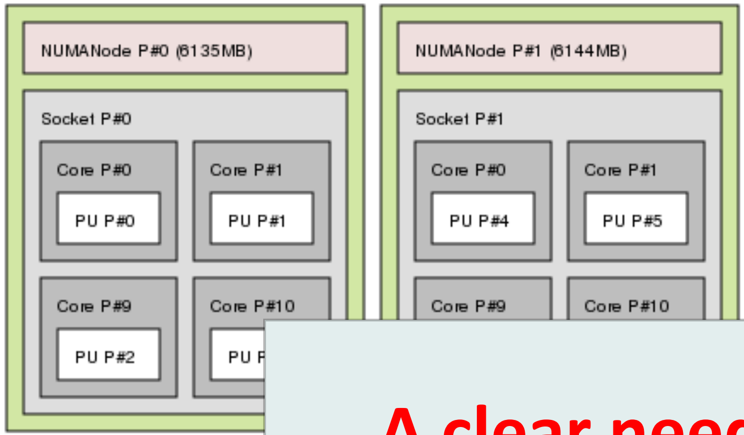
ConnectX-3-FDR (54 Gbps): 2.7 GHz Dual Octa-core (SandyBridge) Intel PCI Gen3 with Mellanox IB FDR switch

Outline

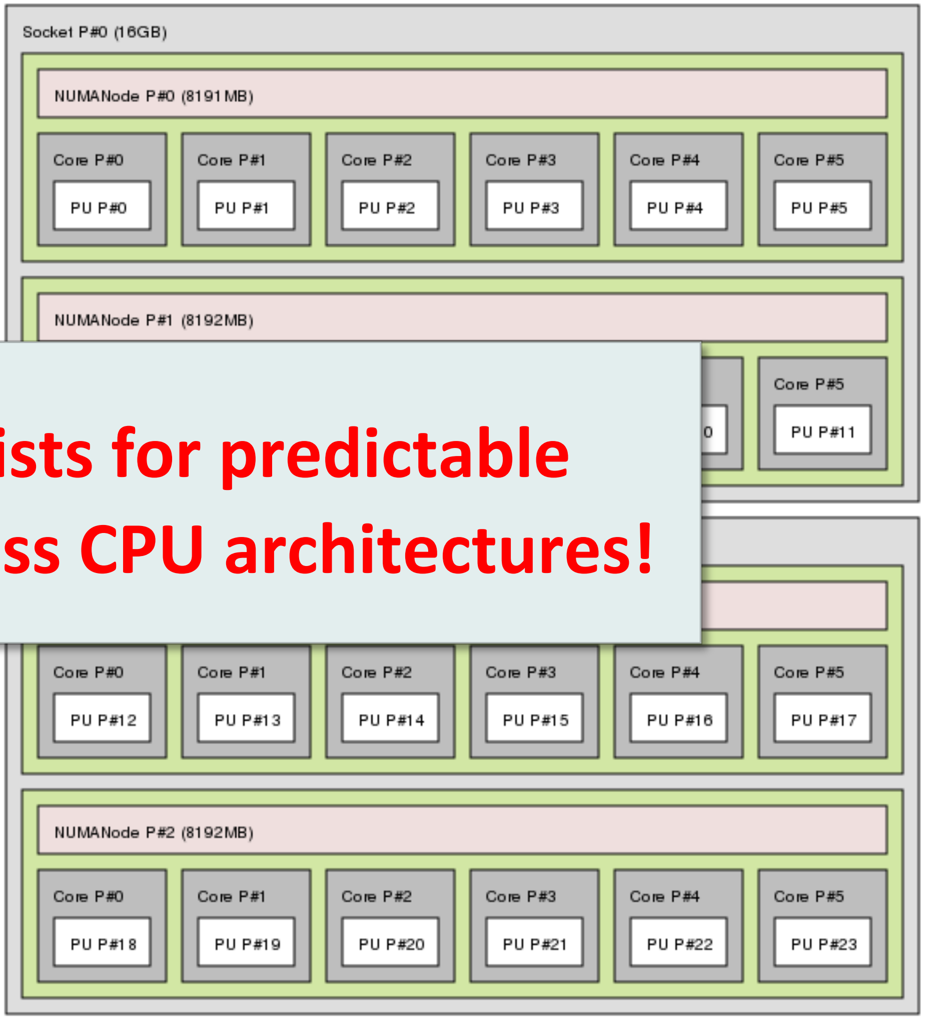
- Job Startup
- Process Mapping
  - Preset process-binding policies
  - User-defined binding
- Pt-to-pt Inter-node Communication
  - Eager and Rendezvous Protocols
  - RDMA Fast Path
- Pt-to-pt Intra-node Communication
  - Shared-memory and LiMIC2/CMA based Communication
  - Architecture-based Tuning
  - Improved Multi-Pair Latency and Message Rate in MV2 2.0a
- One-sided Communication

Plethora of CPU architectures

Intel Westmere CPU

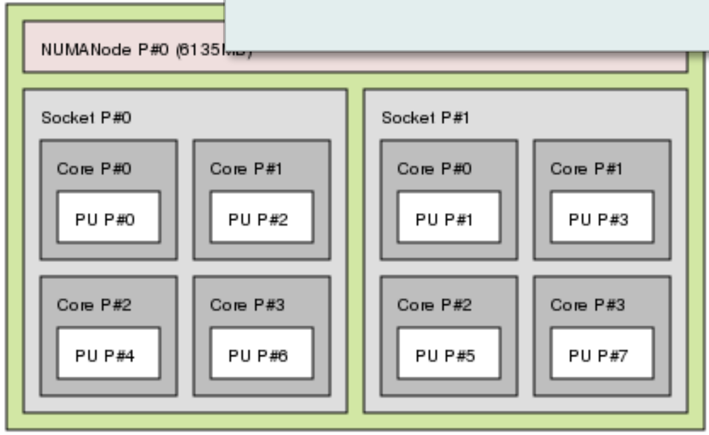


AMD Magny-Cours CPU



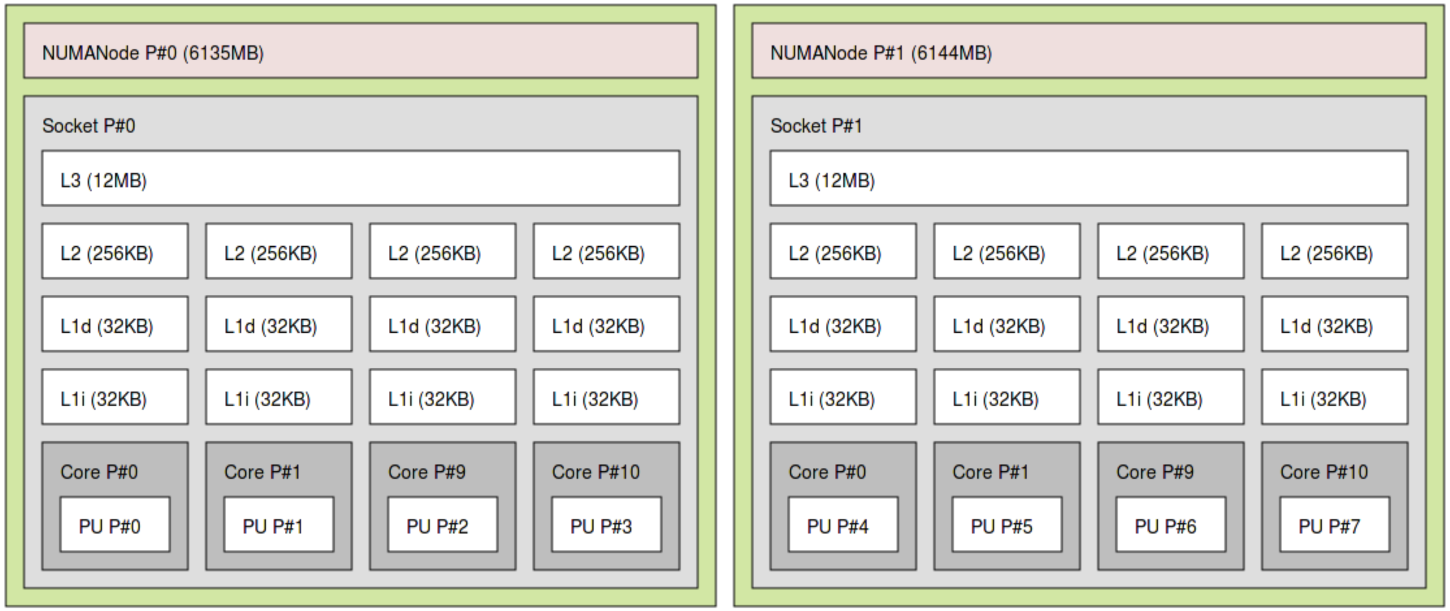
A clear need exists for predictable performance across CPU architectures!

Inte



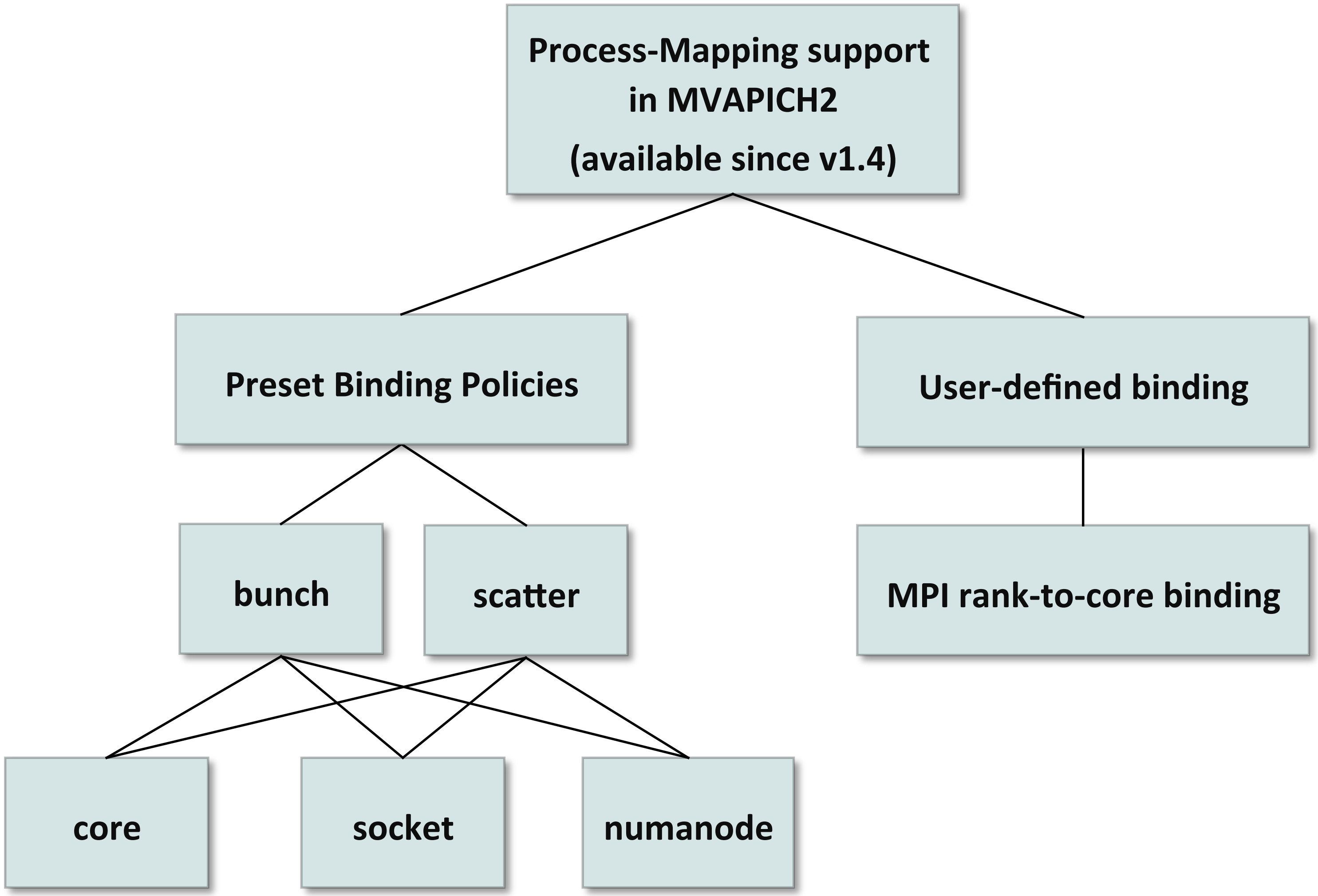
Performance Impact of Core selection

Intel Westmere (8 cores/node)



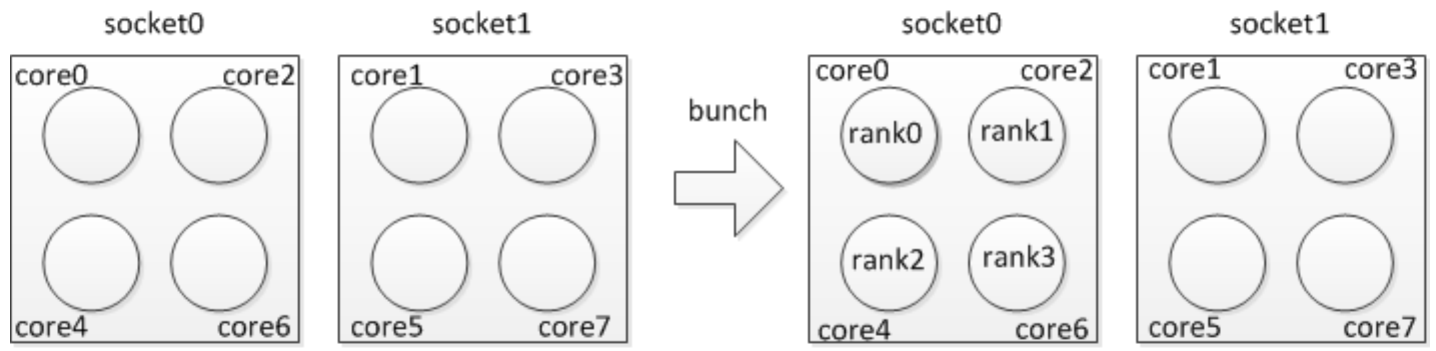
	Message Latency		Observation
Core Pair	0-byte	8K-byte	
1-2	0.17us	1.83us	Same socket, shared L3, best performance
0-1	0.17us	1.87us	Same socket, shared L3, but core0 handles interrupts
1-5	0.41us	3.16us	Different sockets, does not share L3
0-4	0.42us	3.17us	Different sockets, does not share L3, core0 handles interrupts

Process Mapping support in MVAPICH2



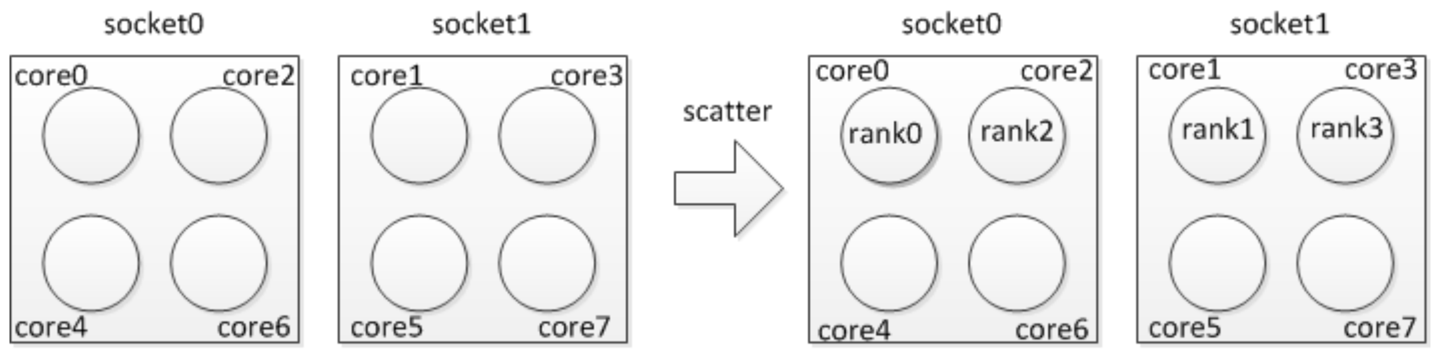
Preset process-binding policies

- MVAPICH2 detects processor architecture at job-launch
- "Bunch" mapping
  - MV2_CPU_BINDING_POLICY=bunch
  - The default mapping policy



Preset process-binding policies

- MVAPICH2 detects processor architecture at job-launch
- "Scatter" mapping
  - MV2_CPU_BINDING_POLICY=scatter

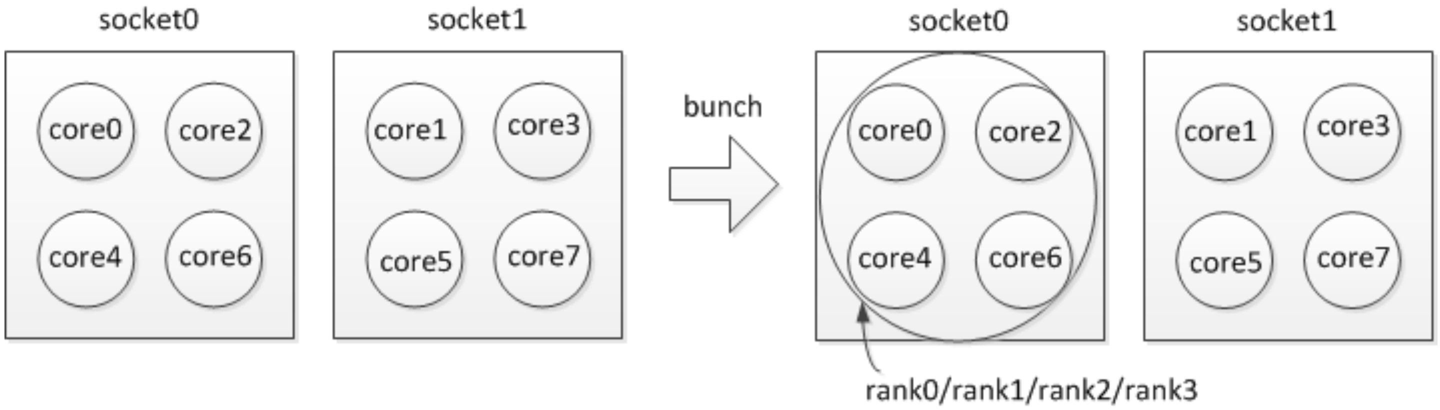


Preset process-binding policies

- Three binding levels
  - Core
  - Socket
  - NUMA-node

MV2_CPU_BINDING_LEVEL = socket

MV2_CPU_BINDING_POLICY = bunch

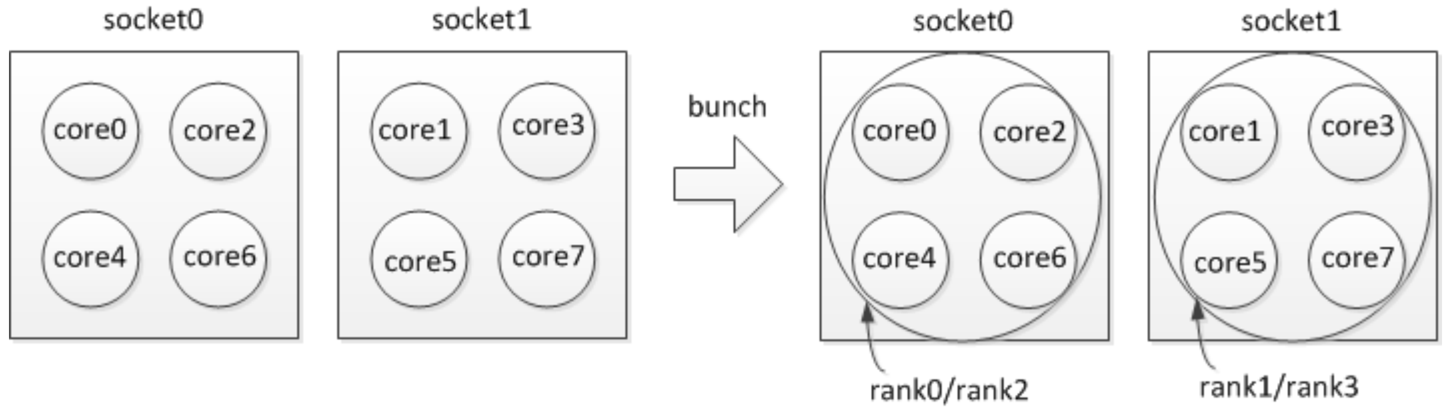


Preset process-binding policies

- Three binding levels
  - Core
  - Socket
  - NUMA-node

MV2_CPU_BINDING_LEVEL = socket

MV2_CPU_BINDING_POLICY = scatter



User-Defined Process Mapping

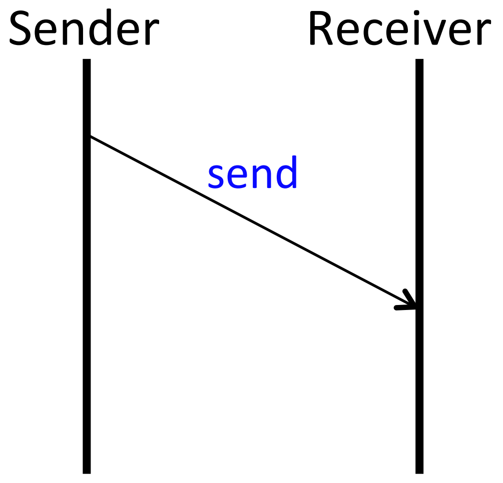
- User has complete-control over process-mapping
- To run 4 processes on cores 0, 1, 4, 5:
  - $ mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0:1:4:5 ./a.out
- Use ‘,’ or ‘-’ to bind to a set of cores:
  - $mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0,2-4:1:5:6 ./a.out
- Useful when a single rank process spawns multiple threads and sets thread binding

Outline

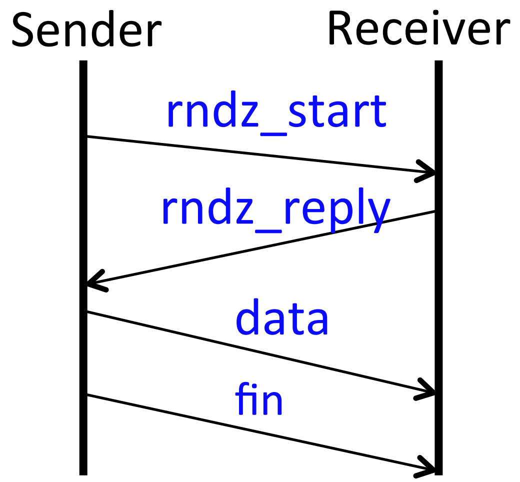
- Job Startup
- Process Mapping
  - Preset process-binding policies
  - User-defined binding
- Pt-to-pt Inter-node Communication
  - Eager and Rendezvous Protocols
  - RDMA Fast Path
- Pt-to-pt Intra-node Communication
  - Shared-memory and LiMIC2/CMA based Communication
  - Architecture-based Tuning
  - Improved Multi-Pair Latency and Message Rate in MV2 2.0a
- One-sided Communication

Inter-node Point-to-Point Communication

- EAGER (buffered, used for small messages)
  - RDMA Fast Path
  - Send/Recv
- RENDEZVOUS (un-buffered, used for large messages)
  - Reduces memory requirement by MPI library
  - Zero-Copy
  - No remote side involvement
  - **Protocols**
    - **RPUT** (RDMA Write)
    - **RGET** (RDMA Read)
    - **R3** (Send/Recv with Packetized Send)

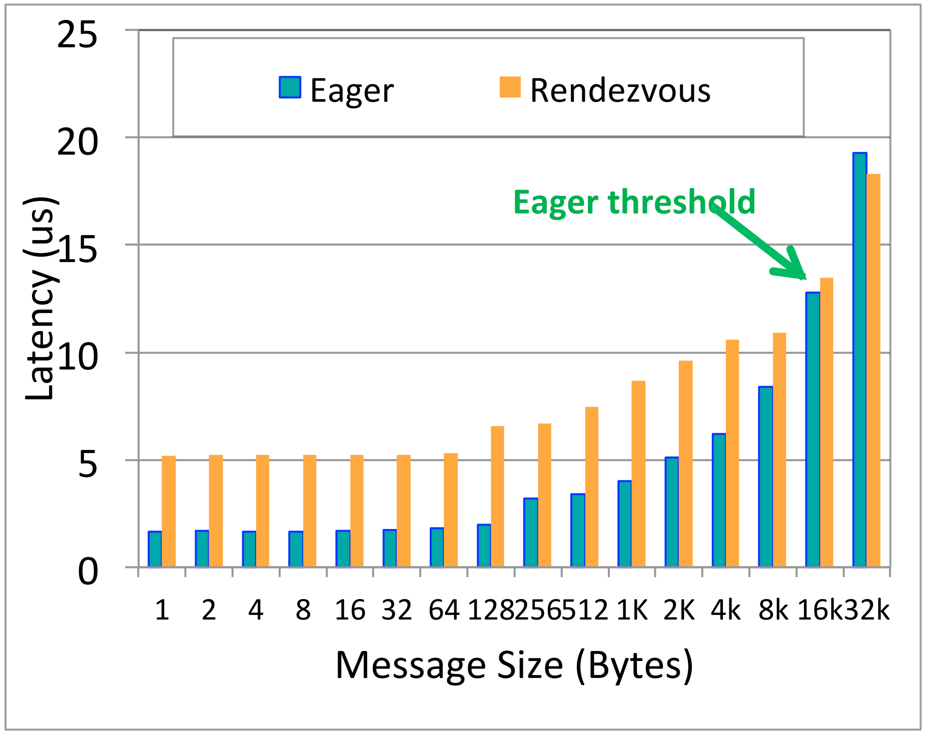


Eager Protocol

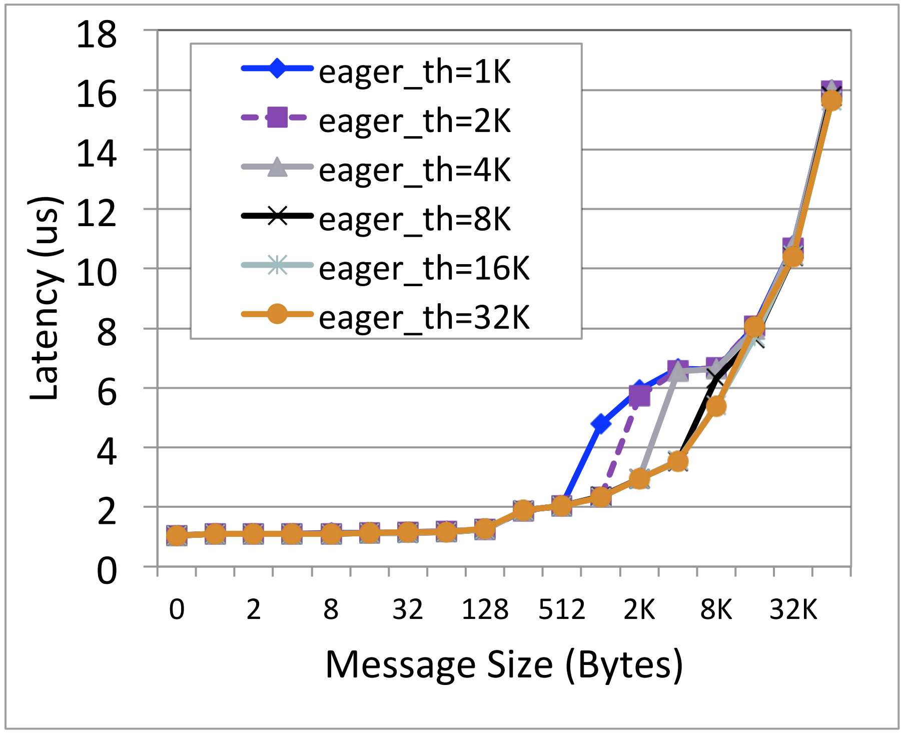


Rendezvous Protocol

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



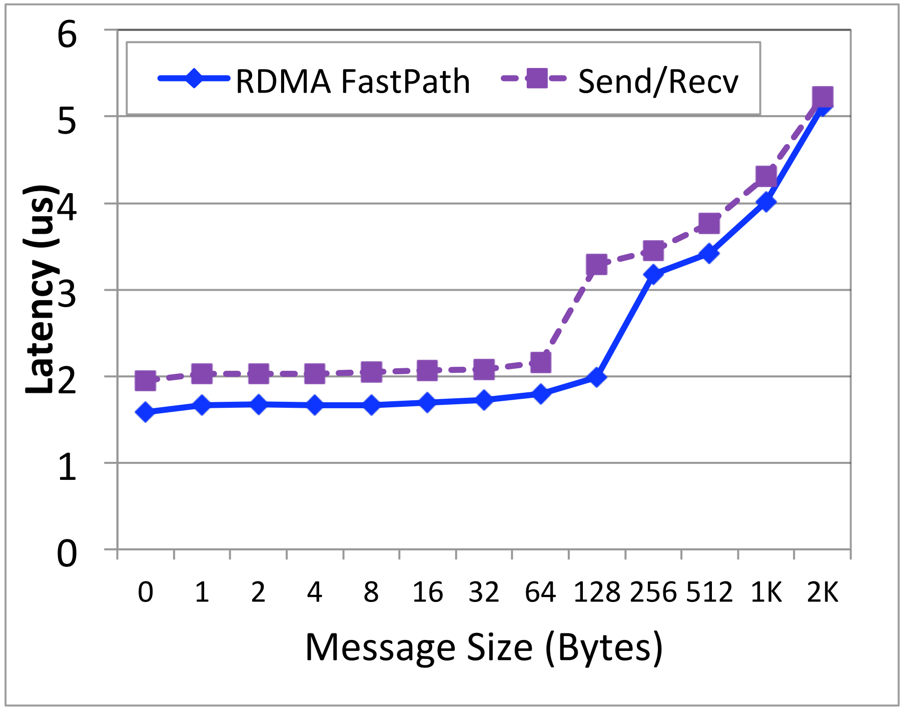
Eager vs Rendezvous



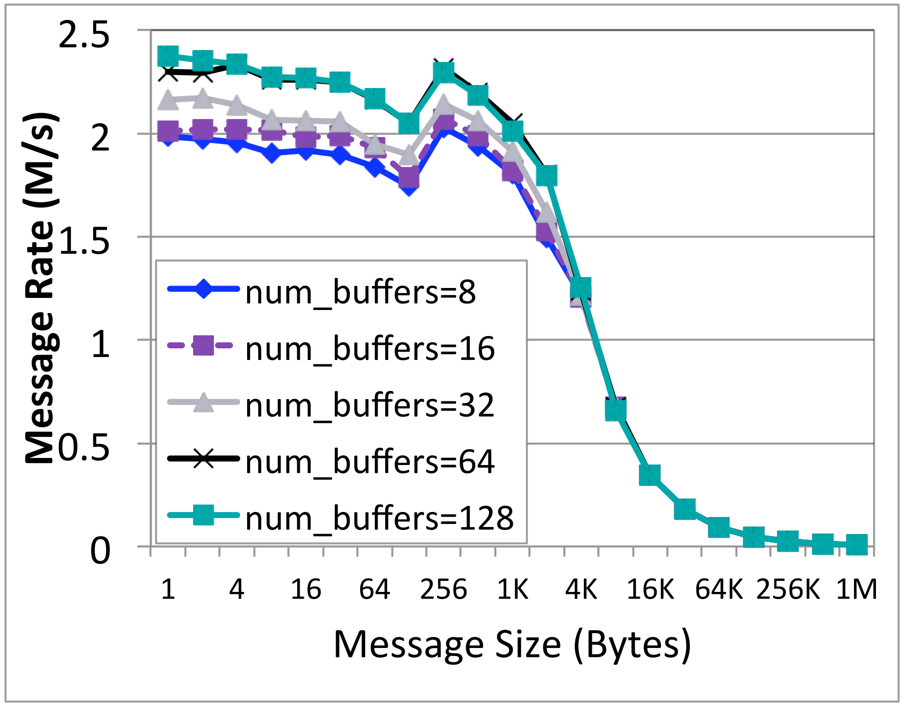
Impact of Eager Threshold

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

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



Eager: Send/Recv vs RDMA FP



Impact of RDMA FP buffers

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

Outline

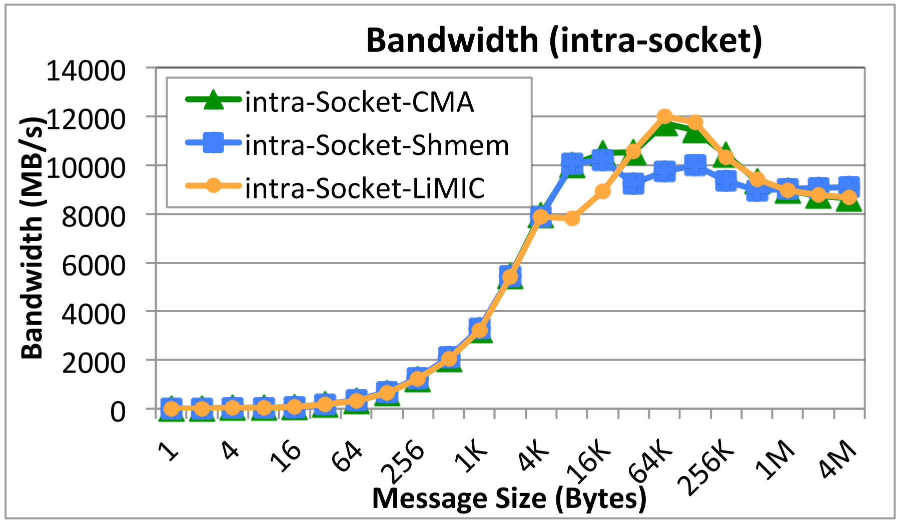
- Job Startup
- Process Mapping
  - Preset process-binding policies
  - User-defined binding
- Pt-to-pt Inter-node Communication
  - Eager and Rendezvous Protocols
  - RDMA Fast Path
- Pt-to-pt Intra-node Communication
  - Shared-memory and LiMIC2/CMA based Communication
  - Architecture-based Tuning
  - Improved Multi-Pair Latency and Message Rate in MV2 2.0a
- One-sided Communication

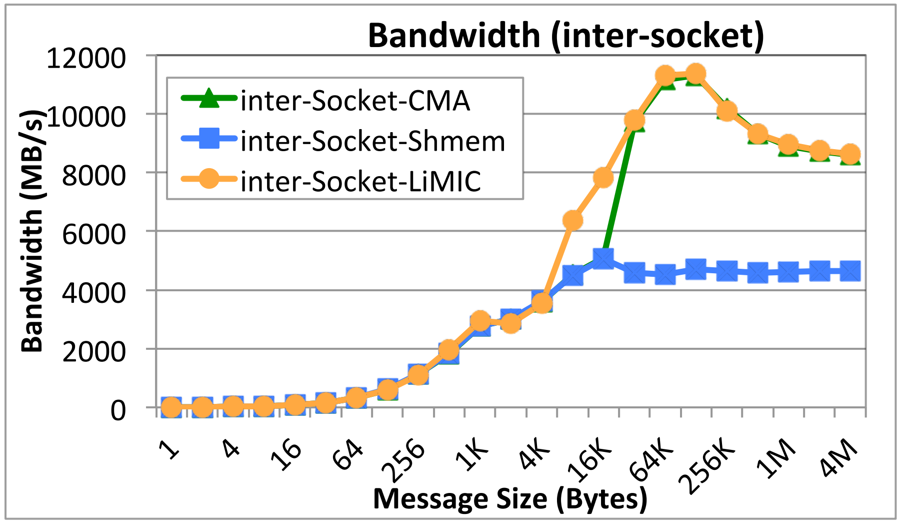
Intra-node Communication Support in MVAPICH2

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

MVAPICH2 Two-Sided Intra-Node Tuning:

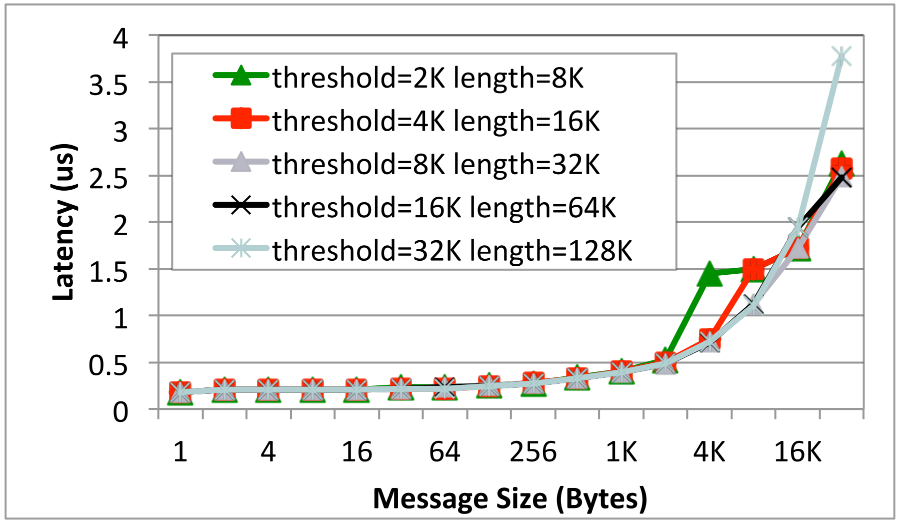
Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA)

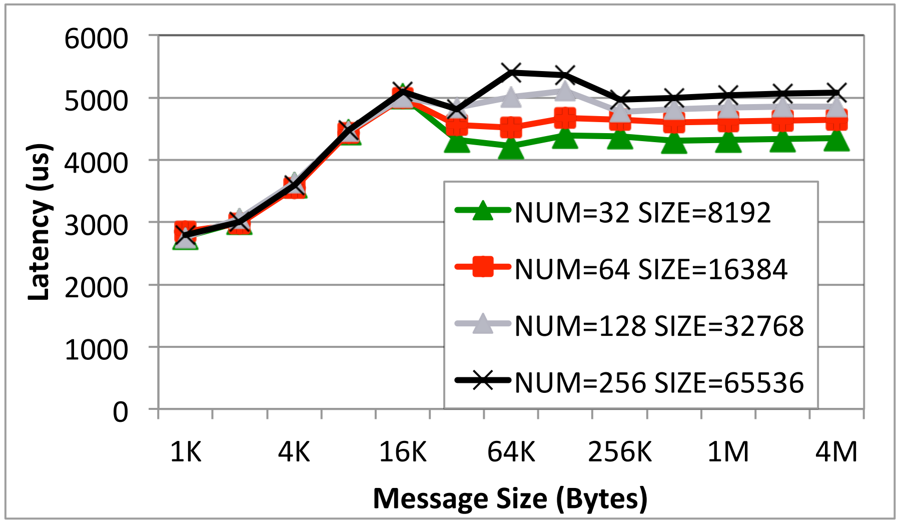




- LiMIC2:
  - configure the library with '--with-limic2'
  - mpirun_rsh –np 2 –f hostfile a.out (To disable: MV2_SMP_USE_LIMIC2=0)
- CMA:
  - configure the library with '--with-cma'
  - mpirun_rsh –np 2 –f hostfile a.out (To disable: MV2_SMP_USE_CMA=0)
- When both '--with-limic2' and '--with-cma' are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

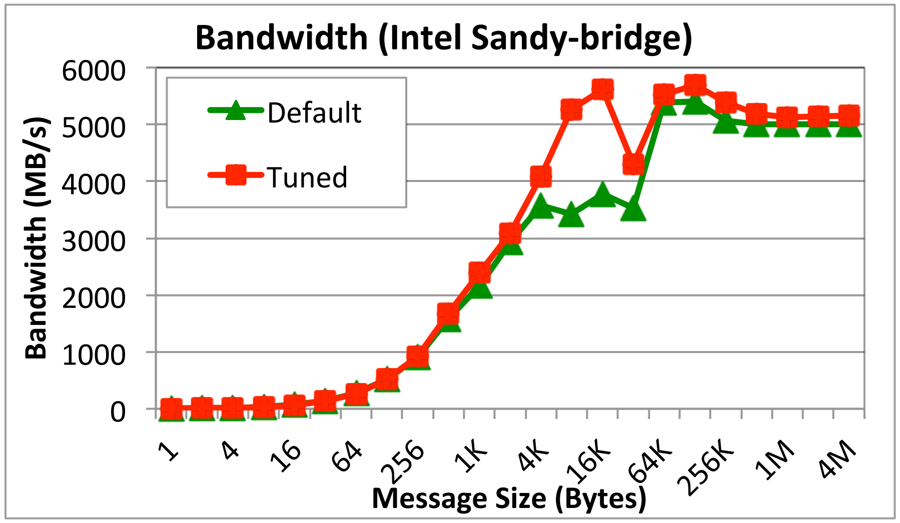
MVAPICH2 Two-Sided Intra-Node Tuning:
Shared-Memory based Runtime Parameters

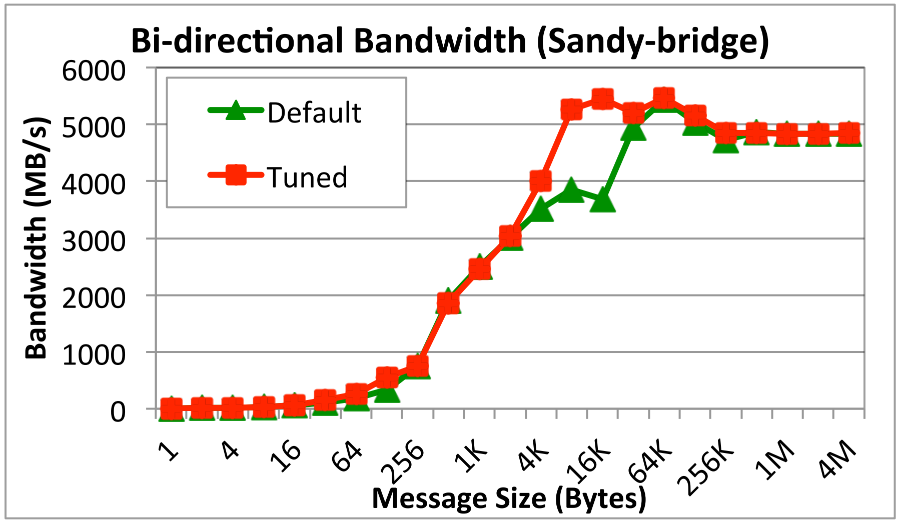


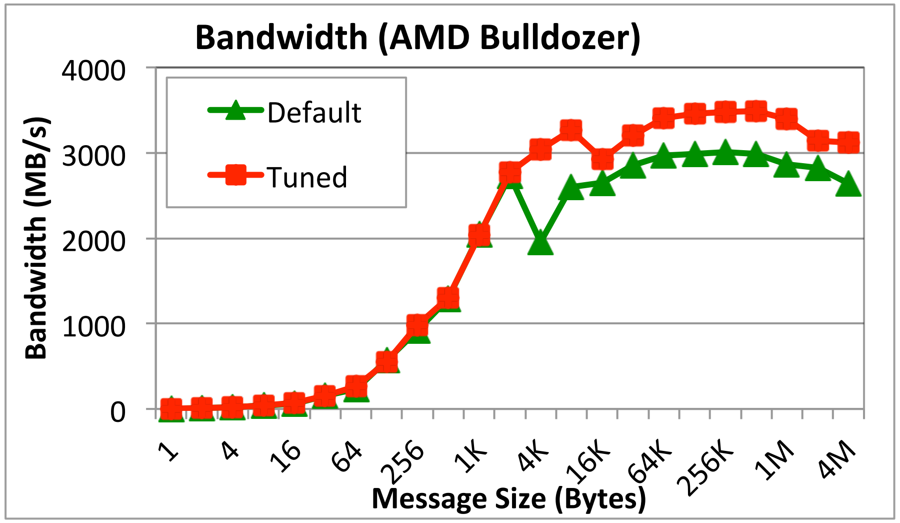


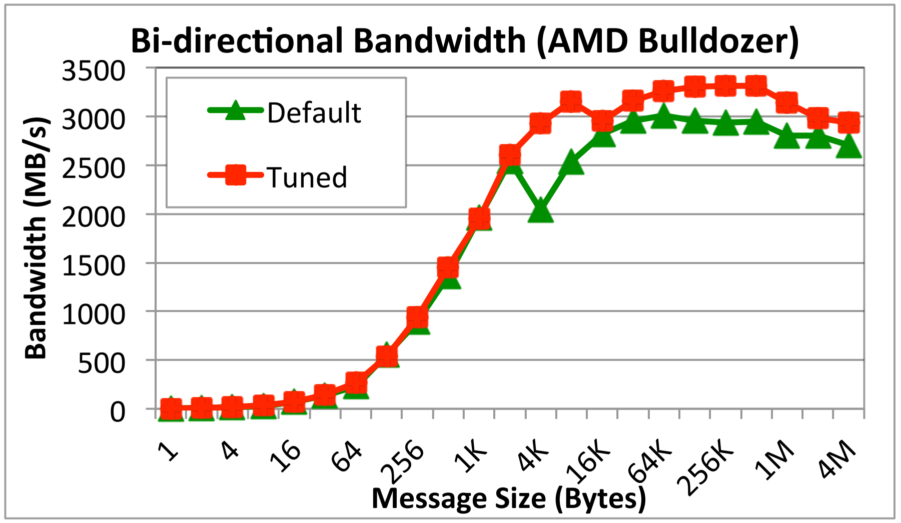
- Adjust eager threshold and eager buffer size:
  - mpirun_rsh –np 2 –f hostfile MV2_SMP_EAGERSIZE=16K MV2_SMPI_LENGTH_QUEUE=64 a.out
  - Will affect the performance of small messages and memory footprint
- Adjust number of buffers and buffer size for shared-memory based Rendezvous protocol:
  - mpirun_rsh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

Impact of Architecture-Specific Tuning



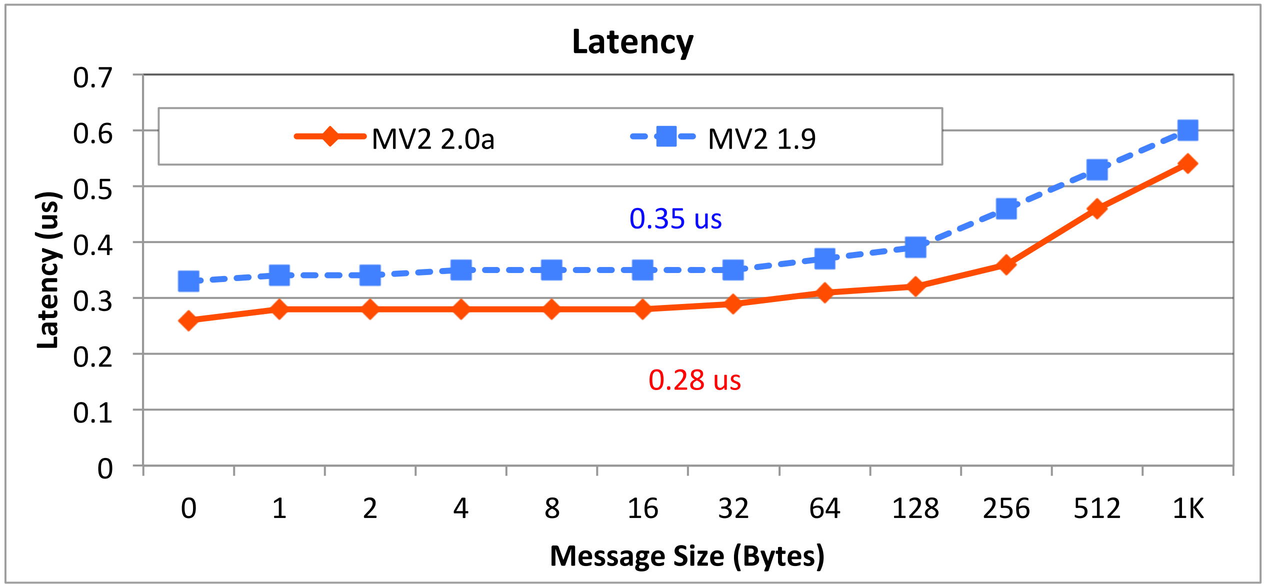






- Architecture-specific tuning is executed for new architectures and new designs introduced into MV2
- MV2_SMP_EAGERSIZE and MV2_SMP_SEND_BUFF_SIZE are updated from Default (1.8) to Tuned (1.9)

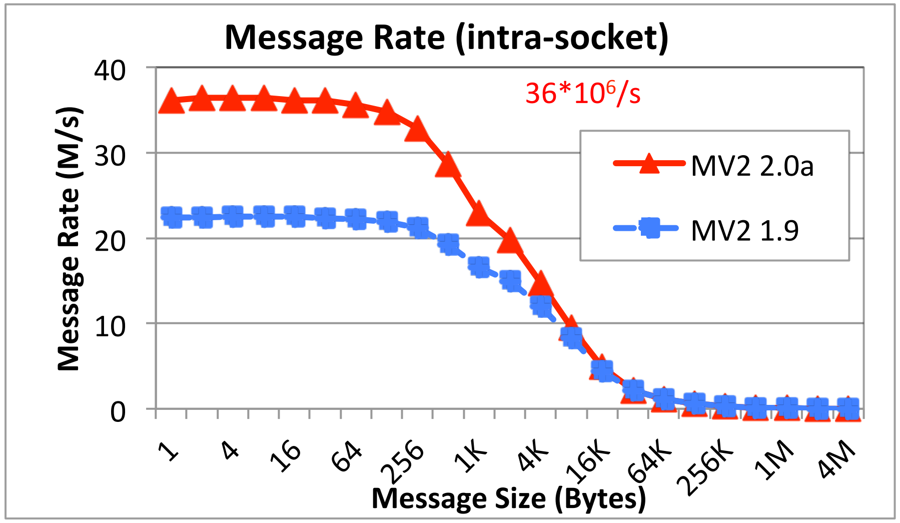
Improved Intra-node Multi-pair Latency and Message Rate

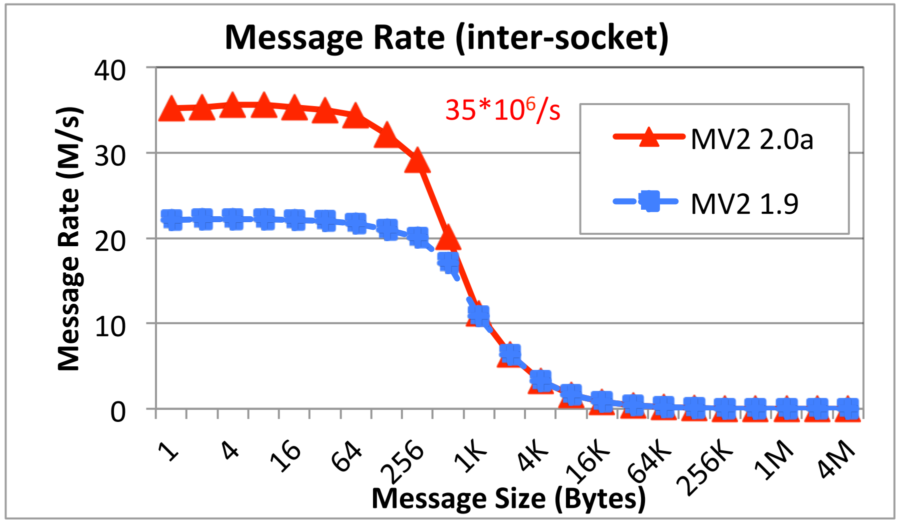


MV2 2.0a

Intel Sandy-bridge

8 pairs of processes





Outline

- Job Startup
- Process Mapping
- Pt-to-pt Inter-node Communication
- Pt-to-pt Intra-node Communication
  - Shared-memory and LiMIC2/CMA based Communication
  - Architecture-based Tuning
  - Improved Multi-Pair Latency and Message Rate in MV2 2.0a
- One-sided Communication
  - InterNode Communication
  - IntraNode Communication
  - MPI-3 RMA Model

Internode One-sided Communication: Direct RDMA-based Designs

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

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

Intranode One-sided Communication

- MVAPICH2 provides truly one-sided implementation of RMA synchronization and communication within a node
  - Shared Memory Backed Windows
  - LiMIC Kernel Module

Parameter	Significance	Default	Notes
MV2_USE_SHM_ONE_SIDED	• Enable / disable shared memory backed windows	0 (Disabled)	• Enable when using one-sided communication • Requires window memory to be allocated using MPI_Alloc_mem • Can also be selectively enabled by passing an info argument to MPI_Alloc_mem
MV2_USE_LIMIC_ONE_SIDED	• Enable / disable LiMIC based one-sided	1 (Enabled)	• Enabled when library is configures with LiMIC

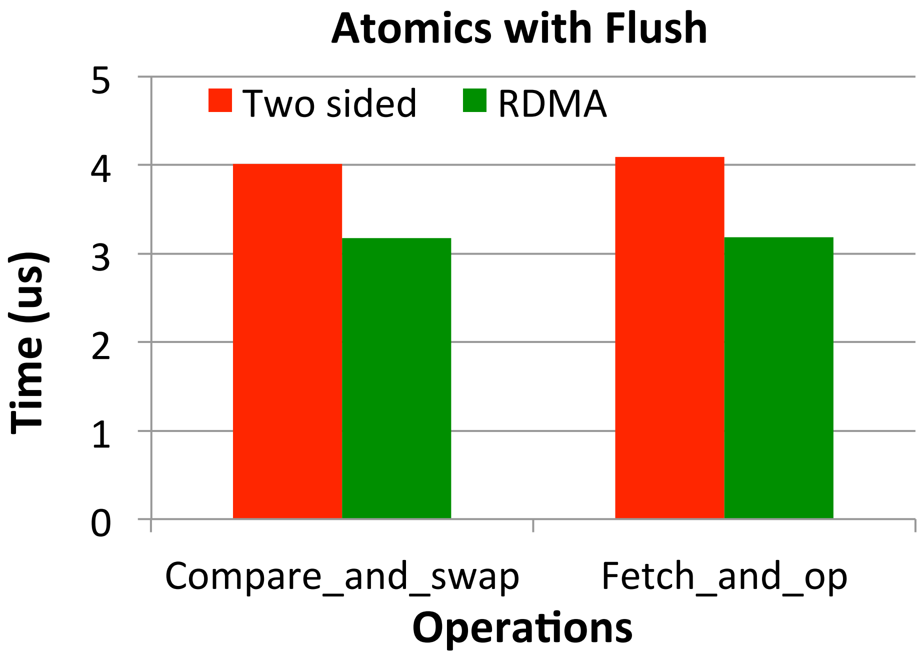
- More information in Sections 6.6 and 6.7 of the userguide:

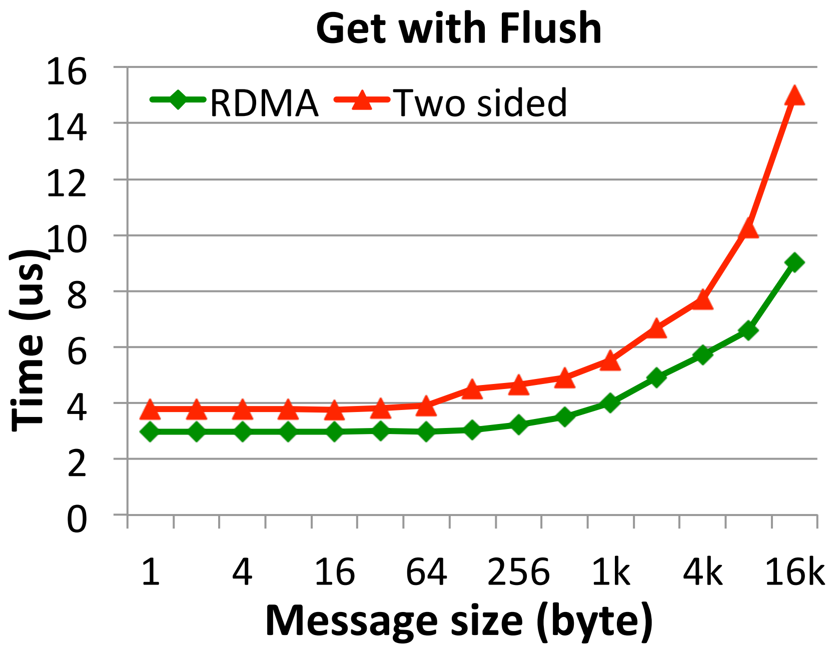
  http://mvapich.cse.ohio-state.edu/support/user_guide_mvapich2-2.0a.html#x1-590006.6

  http://mvapich.cse.ohio-state.edu/support/user_guide_mvapich2-2.0a.html#x1-600006.7

MPI-3 RMA Model: Performance

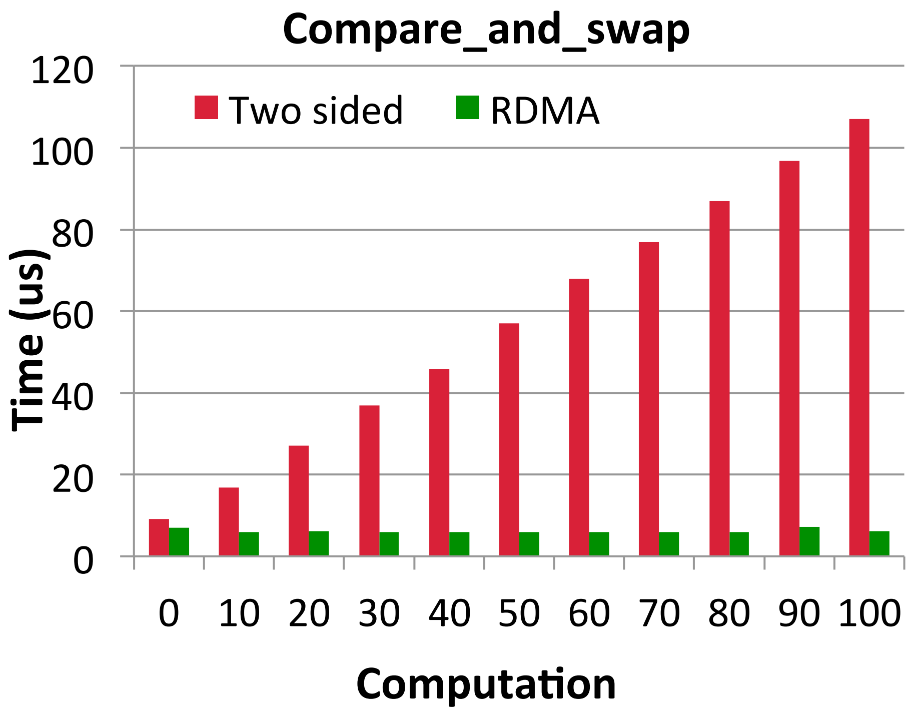
- RDMA-based and truly 1-sided implementation of MPI-3 RMA in progress

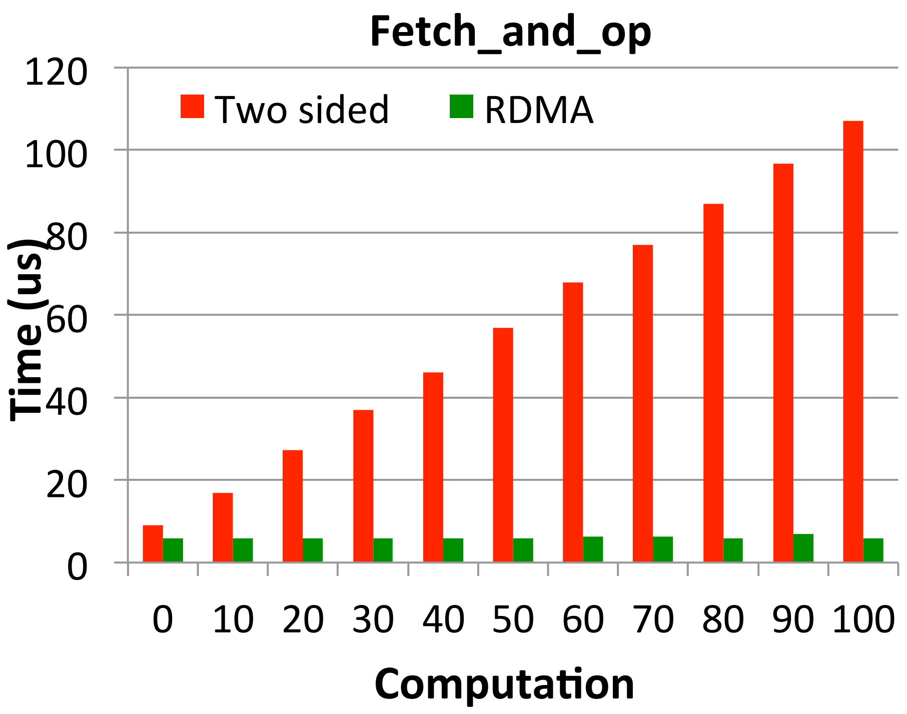




- MVAPICH2-2.0a and OSU micro-benchmarks (OMB v4.1)
- Better performance for MPI_Compare_and_swap and MPI_Fetch_and_op and MPI_Get performance with RDMA-based design

MPI-3 RMA Model: Overlap





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

Web Pointers

NOWLAB Web Page

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

MVAPICH Web Page

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

MVAPICH