All You Want to Know about MVAPICH2 Libraries and Much More...

A Tutorial at MUG’15

by

The MVAPICH Team

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

http://www.cse.ohio-state.edu/~panda
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)
Drivers of Modern HPC Cluster Architectures

- Multi-core processors are ubiquitous
- InfiniBand very popular in HPC clusters
- Accelerators/Coprocessors becoming common in high-end systems
- Pushing the envelope for Exascale computing

**Tianhe – 2 (1)**

**Titan (2)**

**Stampede (8)**

**Tianhe – 1A (24)**
Parallel Programming Models Overview

- 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

Application Kernels/Applications

Middleware

Programming Models
MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenACC, Cilk, Hadoop, MapReduce, etc.

Communication Library or Runtime for Programming Models

Point-to-point Communication (two-sided & one-sided)

Collective Communication

Synchronization & Locks

I/O & File Systems

Fault Tolerance

Networking Technologies
(InfiniBand, 10/40GigE, Aries, BlueGene, OmniPath)

Multi/Many-core Architectures

Accelerators (NVIDIA and MIC)

Co-Design Opportunities and Challenges across Various Layers
Designing (MPI+X) at Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
  - Extremely minimum memory footprint
- Hybrid programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, ...)
- Balancing intra-node and inter-node communication for next generation multi-core (128-1024 cores/node)
  - Multiple end-points per node
- Support for efficient multi-threading
- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
  - Power-aware
- Support for MPI-3 RMA Model
- Support for GPGPUs and Accelerators
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Virtualization Support
The MVAPICH2 Software Family

- High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP, RDMA over Converged Enhanced Ethernet (RoCE), and virtualized clusters.
  - MVAPICH (MPI-1), Available since 2002
  - MVAPICH2 (MPI-2.2, MPI-3.0 and MPI-3.1), Available since 2004
  - MVAPICH2-X (Advanced MPI + PGAS), Available since 2012
  - Support for GPGPUs (MVAPICH2-GDR), Available since 2014
  - Support for MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
- [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)
MVAPICH Project Timeline

- Oct-02: MVAPICH
- Jan-04: MVAPICH
- Nov-04: MVAPICH
- Jan-10: MVAPICH
- Nov-12: OMB
- Aug-14: EOL
- Apr-15: MVAPICH2-GDR
- Jul-15: MVAPICH2-MIC
- Jul-15: MVAPICH2-X
- Jul-15: MVAPICH2-Virt
- Jul-15: MUG'15
The MVAPICH2 Software Family (Cont.)

• Empowering many TOP500 clusters (Jun ‘15 ranking)
  – 8th ranked 519,640-core cluster (Stampede) at TACC
  – 11th ranked 185,344-core cluster (Pleiades) at NASA
  – 22nd ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others

• Used by more than 2,450 organizations in 76 countries

• More than 281,000 downloads from the OSU site directly

• Available with software stacks of many IB, HSE, and server vendors including Linux Distros (RedHat and SuSE)

• Empowering Top500 systems for over a decade
  – System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  – Stampede at TACC (8th in Jun’15, 462,462 cores, 5.168 Plops)
# Usage Guidelines for the MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>Requirements</th>
<th>MVAPICH2 Library to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI with IB, iWARP and RoCE</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>Advanced MPI, PGAS and MPI+PGAS with IB and RoCE</td>
<td>MVAPICH2-X</td>
</tr>
<tr>
<td>MPI with IB &amp; GPU</td>
<td>MVAPICH2-GDR</td>
</tr>
<tr>
<td>MPI with IB &amp; MIC</td>
<td>MVAPICH2-MIC</td>
</tr>
<tr>
<td>HPC Cloud with MPI &amp; IB</td>
<td>MVAPICH2-Virt</td>
</tr>
</tbody>
</table>
Strong Procedure for Design, Development and Release

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

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in MVAPICH2
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases

• Overview of MPIT, Configuration and Debugging Support

• Conclusions and Final Q&A
MVAPICH2 Architecture (Latest Release 2.2a)

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

All Different PCI interfaces
MVAPICH2 2.2a

• Released on 08/18/2015

• Major Features and Enhancements
  – Based on MPICH-3.1.4
  – Support for backing on-demand UD CM information with shared memory for minimizing memory footprint
  – Dynamic identification of maximum read/atomic operations supported by HCA
  – Enabling support for intra-node communications in RoCE mode without shared memory
  – Updated to hwloc 1.11.0
  – Updated to sm_20 kernel optimizations for MPI Datatypes
  – Automatic detection and tuning for 24-core Haswell architecture
  – Enhanced startup performance
    • Support for PMI-2 based startup with SLURM
  – Checkpoint-Restart Support with DMTCP (Distributed MultiThreaded Checkpointing)
  – Enhanced communication performance for small/medium message sizes
  – Support for linking Intel Trace Analyzer and Collector
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  – MVAPICH2
    • Job start-up
    • Point-to-point Inter-node Protocol
    • Transport Type Selection
    • Point-to-point Intra-node Protocol and Scheme
    • MPI-3 RMA
    • Collectives
    • Fault-tolerance
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases
• Overview of MPIT, Configuration and Debugging Support
• Conclusions and Final Q&A
Job-Launchers supported by MVAPICH2

Core Launchers
- mpirun_rsh
  - Default and preferred launcher for all interfaces
  - Supports scalable hierarchical job-startup mechanism
- mpiexec.hydra
  - Developed at ANL by the MPICH team

Wrappers and interfaces
- mpiexec
  - To support interfaces prescribed by the MPI standard
- mpiexec.mpirun_rsh
  - Hydra-like interface to mpirun_rsh

SLURM
  - Compatibility with external resource managers
TORQUE

MUG'15
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
**MPI_Init Performance on TACC Stampede**

- Near-constant MPI_Init performance
- 59 times improvement at 8,192 processes (512 nodes)
- New designs show good scaling with 16K processes and above

"Non-blocking PMI Extensions for Fast MPI Startup"


Int'l Symposium on Cluster, Cloud, and Grid Computing (CCGrid ‘15)
MPI Hello World Performance on TACC Stampede

- PMI Exchange costs overlapped with application computation
- 5.7 times improvement at 8,192 processes (512 nodes)
- New designs to be available as part of upcoming releases

"Non-blocking PMI Extensions for Fast MPI Startup"
Int'l Symposium on Cluster, Cloud, and Grid Computing (CCGrid '15)
Presentation Overview

- Installation Guidelines, Runtime Optimization and Tuning Flexibility in MVAPICH2
  - Job start-up
  - Point-to-point Inter-node Protocol
    - Eager and Rendezvous Protocols
    - RDMA Fast Path
  - Transport Type Selection
  - Point-to-point Intra-node Protocol and Scheme
  - MPI-3 RMA
  - Collectives
  - Fault-tolerance
  - MVAPICH2-X
  - MVAPICH2-GDR
  - MVAPICH2-MIC
  - MVAPICH2-Virt
  - OMB
- Preview of Upcoming Releases
- Overview of MPIT, Configuration and Debugging Support
- Conclusions and Final Q&A
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)*
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 -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

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

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in MVAPICH2
  – Job start-up
  – Point-to-point Inter-node Protocol
  • Transport Type Selection
    – Shared Receive Queue
    – eXtended Reliable Connect transport protocol
    – UD transport protocol and Hybrid
  • Point-to-point Intra-node Protocol and Scheme
  • MPI-3 RMA
  • Collectives
  • Fault-tolerance
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases
• Overview of MPIT, Configuration and Debugging Support
• Conclusions and Final Q&A
SRQ is a hardware mechanism for a process to share receive resources (memory) across multiple connections

- Introduced in specification v1.2

\[
0 < Q \ll P^*((M*N)-1)
\]
Using Shared Receive Queues with MVAPICH2

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

**MPI_Init memory utilization**

**Analytical model**

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

- Refer to Shared Receive Queue (SRQ) Tuning section of MVAPICH2 user guide for more information
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1000008.5](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1000008.5)
Using eXtended Reliable Connection (XRC) in MVAPICH2

- Memory usage for 32K processes with 8-cores per node can be **54 MB/process** (for connections)
- NAMD performance improves when there is frequent communication to many peers
- Enabled by setting **MV2_USE_XRC** to 1 (Default: Disabled)
- Requires OFED version > 1.3
  - Unsupported in earlier versions (< 1.3), OFED-3.x and MLNX_OFED-2.0
  - MVAPICH2 build process will automatically disable XRC if unsupported by OFED
- Automatically enables SRQ and ON-DEMAND connection establishment
- Refer to [eXtended Reliable Connection (XRC)](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1010008.6) section of MVAPICH2 user guide for more information
- **MVAPICH2-RC** vs **MVAPICH2-XRC**

**Memory Usage**

<table>
<thead>
<tr>
<th>Number of Connections</th>
<th>Memory (MB/Process)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>16</td>
<td>600</td>
</tr>
<tr>
<td>64</td>
<td>1600</td>
</tr>
<tr>
<td>256</td>
<td>54 MB/Process</td>
</tr>
<tr>
<td>1K</td>
<td>100 MB/Process</td>
</tr>
<tr>
<td>4K</td>
<td>300 MB/Process</td>
</tr>
<tr>
<td>16K</td>
<td>600 MB/Process</td>
</tr>
</tbody>
</table>

**Performance with NAMD (1024 cores)**

- Dataset: apoa1, er-gre, f1atpase, jac
- **Normalized Time**

Using eXtended Reliable Connection (XRC) in MVAPICH2

**Memory Usage**

<table>
<thead>
<tr>
<th>Number of Connections</th>
<th>Memory (MB/Process)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>16</td>
<td>600</td>
</tr>
<tr>
<td>64</td>
<td>1600</td>
</tr>
<tr>
<td>256</td>
<td>54 MB/Process</td>
</tr>
<tr>
<td>1K</td>
<td>100 MB/Process</td>
</tr>
<tr>
<td>4K</td>
<td>300 MB/Process</td>
</tr>
<tr>
<td>16K</td>
<td>600 MB/Process</td>
</tr>
</tbody>
</table>

**Performance with NAMD (1024 cores)**

- Dataset: apoa1, er-gre, f1atpase, jac
- **Normalized Time**

MUG'15
Using UD Transport with MVAPICH2

### Memory Footprint of MVAPICH2

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>RC (MVAPICH2 2.0b)</th>
<th>UD (MVAPICH2 2.0b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conn.</td>
<td>Buffers</td>
</tr>
<tr>
<td>512</td>
<td>22.9</td>
<td>24</td>
</tr>
<tr>
<td>1024</td>
<td>29.5</td>
<td>24</td>
</tr>
<tr>
<td>2048</td>
<td>42.4</td>
<td>24</td>
</tr>
</tbody>
</table>

### Performance with SMG2000

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

### Parameter Significance

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

- Refer to Running with scalable UD transport section of MVAPICH2 user guide for more information
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-620006.10](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-620006.10)
Hybrid (UD/RC/XRC) Mode in MVAPICH2

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

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/mvapich2-2.2a-userguide.html#x1-630006.11

**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
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  – MVAPICH2
    • Job start-up
    • Point-to-point Inter-node Protocol
    • Transport Type Selection
    • Point-to-point Intra-node Protocol and Scheme
      – Shared-memory and LiMIC2/CMA based Communication
      – Architecture-based Tuning
    • MPI-3 RMA
    • Collectives
    • Fault-tolerance
      – MVAPICH2-X
      – MVAPICH2-GDR
      – MVAPICH2-MIC
      – MVAPICH2-Virt
      – OMB
• Preview of Upcoming Releases
• Overview of MPIT, Configuration and Debugging Support
• Conclusions and Final Q&A
Intra-node Communication Support in MVAPICH2

• Shared-Memory based two-copy intra-node communication
  – Copy from the sender’s user buffer to the shared buffer
  – Copy from the shared buffer to the receiver’s user buffer

• LiMIC2 on modern multi-core platforms
  – Kernel-level module for achieving single copy intra-node communication
  – LiMIC2 is used for rendezvous protocol message size
  – LiMIC2 module is required

• CMA (Cross Memory Attach) support
  – Single copy intra-node communication through Linux syscalls
  – Available from Linux kernel 3.2
MVAPICH2 Two-Sided Intra-Node Tuning: Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA)

- LiMIC2:
  - configure the library with ‘--with-limic2’
  - `mpirun_rsh -np 2 -f hostfile a.out (To disable: MV2_SMP_USE_LIMIC2=0)
- CMA:
  - configure the library with ‘--with-cma’
  - `mpirun_rsh -np 2 -f hostfile a.out (To disable: MV2_SMP_USE_CMA=0)
- When both ‘--with-limic2’ and ‘--with-cma’ are included at the same time, LiMIC2 is chosen by default
- When neither ‘--with-limic2’ or ‘--with-cma’ is used during in configuration, shared-memory based design is chosen
MVAPICH2 Two-Sided Intra-Node Tuning: Shared-Memory based Runtime Parameters

- Adjust eager threshold and eager buffer size:
  - `mpirun_rsh -np 2 -f hostfile MV2_SMP_EAGERSIZE=16K MV2_SMPI_LENGTH_QUEUE=64 a.out`
  - Will affect the performance of small messages and memory footprint

- Adjust number of buffers and buffer size for shared-memory based Rendezvous protocol:
  - `mpirun_rsh -np 2 -f hostfile MV2_SMP_SEND_BUFFER=32 MV2_SMP_SEND_BUFF_SIZE=8192 a.out`
  - Will affect the performance of large messages and memory footprint
Impact of Architecture-Specific Tuning

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

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  • MVAPICH2
    • Job start-up
    • Point-to-point Inter-node Protocol
    • Transport Type Selection
    • Point-to-point Intra-node Protocol and Scheme
  • MPI-3 RMA
    • InterNode Communication
    • IntraNode Communication
    • MPI-3 RMA Model
  • Collectives
  • Fault-tolerance
    • MVAPICH2-X
    • MVAPICH2-GDR
    • MVAPICH2-MIC
    • MVAPICH2-Virt
    • OMB

• Preview of Upcoming Releases

• Overview of MPIT, Configuration and Debugging Support

• Conclusions and Final Q&A
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_USE_RDMA_ONE_SIDED</td>
<td>• Enable / Disable RDMA-based</td>
<td>1 (Enabled)</td>
<td>• Disable only for debugging purposes</td>
</tr>
<tr>
<td></td>
<td>designs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

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

- Refer to Running with LiMIC2 section of MVAPICH2 user guide for more information
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-580006.6](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-580006.6)
MPI-3 RMA Model: Performance

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

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

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in MVAPICH2
  – Job start-up
  – Point-to-point Inter-node Protocol
  – Transport Type Selection
  – Point-to-point Intra-node Protocol and Scheme
  – MPI-3 RMA
• Collectives
  – Improved Hardware Based Collectives in MVAPICH2
  – Tuning Collective Communication Operations in MVAPICH2
  – Non-blocking Collectives in MVAPICH2
• Fault-tolerance
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB
• Preview of Upcoming Releases
• Overview of MPIT, Configuration and Debugging Support
• Conclusions and Final Q&A
Collective Communication in MVAPICH2

Run-time flags:
All shared-memory based collectives:  **MV2_USE_SHMEM_COLL** (Default: **ON**)
Hardware Mcast-based collectives:  **MV2_USE_MCAST** (Default: **OFF**)

MUG'15
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

MUG'15
Enabling Hardware Multicast-aware

• Multicast is applicable to
  • MPI_Bcast
  • MPI_Scatter
  • MPI_Allreduce

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Nature</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_USE_MCAST = 1</td>
<td>Enables hardware Multicast features</td>
<td>Disabled</td>
</tr>
<tr>
<td>--enable-mcast</td>
<td>Configure flag to enable</td>
<td>Enabled</td>
</tr>
</tbody>
</table>
MPI_Scatter - Benefits of using Hardware-Mcast

- 57% improvement in small message latency for 512 processes
- 75% improvement in small message latency for 1,024 processes

Use MV2_USE_MCAST=1 to enable MCAST-based designs
Shared-memory Aware Collectives

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

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

MV2_USE_SHMEM_BARRIER=0/1
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 intra node communication
- Such patterns are good indicators of enabling LiMIC2 to speedup intra node transfers
- Collectives such as MPI_Allgather benefits from the use of LiMIC2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Nature</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_SMP_USE_LIMIC2</td>
<td>Enables shared memory optimizations for collectives</td>
<td>Enabled</td>
</tr>
<tr>
<td>--with-limic2</td>
<td>Configure flag</td>
<td>disabled</td>
</tr>
</tbody>
</table>
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
Presentation Overview

- Installation Guidelines, Runtime Optimization and Tuning Flexibility in MVAPICH2
  - Job start-up
  - Point-to-point Inter-node Protocol
  - Transport Type Selection
  - Point-to-point Intra-node Protocol and Scheme
  - MPI-3 RMA
  - Collectives
  - Fault-tolerance
    - Checkpoint-Restart Schemes
    - Process-Migration Schemes
    - Automatic Path Migration
    - Run Through Stabilization
  - MVAPICH2-X
  - MVAPICH2-GDR
  - MVAPICH2-MIC
  - MVAPICH2-Virt
  - OMB

- Preview of Upcoming Releases
- Overview of MPIT, Configuration and Debugging Support
- Conclusions and Final Q&A
## Fault-Tolerance in MVAPICH2

### System-Level
- **Checkpoint-Restart**
  - Process-migration
  - DMTCP
- **Automatic Path Migration**

### Application-aware
- **App-aware CR**
- **Run-through Stabilization**

<table>
<thead>
<tr>
<th>Feature</th>
<th>MVAPICH2 version</th>
<th>Release Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLCR-based system-level MPI application Checkpointing</td>
<td>0.9.8</td>
<td>2006</td>
</tr>
<tr>
<td>FTB-enabled Checkpoint-Restart</td>
<td>1.4</td>
<td>2008</td>
</tr>
<tr>
<td>FUSE-assisted Write-Aggregation Scheme</td>
<td>1.6</td>
<td>2010</td>
</tr>
<tr>
<td>Basic File-Copy based Process Migration</td>
<td>1.6</td>
<td>2010</td>
</tr>
<tr>
<td>Pipelined Process Migration using RDMA</td>
<td>1.7</td>
<td>2011</td>
</tr>
<tr>
<td>Checkpoint-Restart support for the Nemesis-IB channel</td>
<td>1.8</td>
<td>2012</td>
</tr>
<tr>
<td>Scalable Multi-level Checkpointing using SCR</td>
<td>1.9</td>
<td>2013</td>
</tr>
<tr>
<td>DMTCP Support</td>
<td>2.1</td>
<td>2014</td>
</tr>
<tr>
<td>More features under development</td>
<td>2.2x</td>
<td>2015</td>
</tr>
</tbody>
</table>

*FTB* - Fault Tolerance for Big Data applications, *FUSE* - Filesystem User Space Export

**MUG'15**
Using the Checkpoint-Restart Feature

- Requires Berkeley Lab Checkpoint-Restart (BLCR) library
- Build with CR support: `--enable-ckpt` (or) `--with-blcr=PATH_TO_BLCR_INSTALLATION`
- Launching the job:
  ```
  $mpirun_rsh -np 2 -hostfile ./hfile
  MV2_CKPT_FILE = /pfs/ckpt/app1
  MV2_CKPT_MAX_SAVE_CKPTS = 3
  MV2_CKPT_NO_SYNC = 0 ./a.out
  ```
- Triggering a checkpoint:
  - `cr_checkpoint -p <PID of mpirun_rsh>`
  - Run `$MV2_INSTALL_DIR/bin/mv2_checkpoint` and select the job to checkpoint
  - Call `MVAPICH2_Sync_Checkpoint()` from within the application
  - Set `MV2_CKPT_INTERVAL = 30` for automated checkpointing
- Restarting from a checkpoint:
  - `cr_restart /pfs/ckpt/context.<pid>`

Refer to System-Level Checkpoint/Restart section of MVAPICH2 user guide for more information

http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-670006.14.1
Using the Process-Migration Feature

- Requires BLCR, Fault-Tolerance Backplane (FTB), and FUSE (RDMA-based)
- Build with Migration support: --enable-ckpt-migration
- Setup FTB and launch the job:
  
  ```sh
  $mpirun_rsh -np 4 -hostfile ./hosts -sparehosts ./spares ./a.out
  ```

- Triggering a migration:
  - Send `SIGUSR2` signal to `mpispawn` on source/failing node
  - `$MV2_INSTALL_PATH/bin/mv2_trigger <hostname_of_source_node>`
  - Automatically triggered by FTB-IPMI available at
    
    **http://nowlab.cse.ohio-state.edu/projects/ftb-ib/#FTB-IPMI**

- Refer to Job Pause-Migration-Restart Support section of MVAPICH2 user guide for more information
  
  **http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-750006.14.3**
Network-Level FT with Automatic Path Migration (APM)

- Allows recovery from network faults in the presence of multiple paths
- Enabled by the LID-Mask Count (LMC) mechanism
- Run with APM support:
  - `$ mpirun_rsh -np 2 host1 host2 MV2_USE_APM=1 ./a.out`
- Test APM in the absence of actual network faults:
  - `$ mpirun_rsh -np 2 host1 host2 MV2_USE_APM=1 MV2_USE_APM_TEST=1 ./a.out`
  - Periodically migrates between primary and alternate paths

- Refer to Network Fault Tolerance with Automatic Path Migration section of MVAPICH2 user guide
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-770006.14.5](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-770006.14.5)
Run-Through Stabilization

- Proposal made to the MPI Forum’s FT working group pre-3.0
- Communication failures not treated as fatal errors
- Return error code on process failure to user-set handler
- Outstanding send/recv/wild-card recv (with MPI_ANY_SOURCE) posted to failed communicator returns error code
- Supported in the Nemesis-IB channel (--with-device=ch3:nemesis:ib)
- Run with mpiexec.hydra
  - Set `MV2_RUN THROUGH_STABILIZATION = 1`
  - Add --disable-auto-cleanup flag
- Query list of failed processes from application:
  - `MPI_Comm_get_attr(MPI_COMM_WORLD, MPICH_ATTR_FAILED_PROCESSES, &failed_procs, &flag);`

- Refer to Run-Through Stabilization section of MVAPICH2 user guide for more information
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2.2.2a-userguide.html#x1-760006.14.4](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2.2.2a-userguide.html#x1-760006.14.4)
Presentation Overview

- Installation Guidelines, Runtime Optimization and Tuning Flexibility in MVAPICH2
- MVAPICH2-X
  - MVAPICH2-X, Feature highlights
  - Advanced Features (DC and Core-Direct)
  - Usage instructions
  - Performance highlights
  - Application-level Case Studies and Evaluation
  - FAQs
- MVAPICH2-GDR
- MVAPICH2-MIC
- MVAPICH2-Virt
- OMB

- Preview of Upcoming Releases
- Overview of MPIT, Configuration and Debugging Support
- Conclusions and Final Q&A
MVAPICH2-X for Advanced MPI, Hybrid MPI + PGAS Applications

- Current Model – Separate Runtimes for OpenSHMEM/UPC/CAF and MPI
  - Possible deadlock if both runtimes are not progressed
  - Consumes more network resource

- Unified communication runtime for MPI, UPC, OpenSHMEM, CAF available with MVAPICH2-X 1.9 onwards!
  - [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)
MVAPICH2-X 2.2a

- Released on 08/18/2015
- MVAPICH2-X 2.2a Feature Highlights
  - Based on MVAPICH2 2.2a including MPI-3 features
    - Compliant with UPC 2.20.2, OpenSHMEM v1.0h and CAF 3.0.39
  - MPI (Advanced) Features
    - Support for Dynamically Connected (DC) transport protocol
      - Available for pt-to-pt, RMA and collectives - Support for Hybrid mode with RC/DC/UD/XRC
    - Support for Core-Direct based Non-blocking collectives
      - Available for Ibcast, Ibarrier, Iscatter, Igather, Ialltoall and Iallgather
  - OpenSHMEM Features
    - Support for RoCE - Support for Dynamically Connected (DC) transport protocol
  - UPC Features
    - Based on Berkeley UPC 2.20.2 (contains changes/additions in preparation for upcoming UPC 1.3 specification)
    - Support for RoCE - Support for Dynamically Connected (DC) transport protocol
  - CAF Features
    - Support for RoCE
    - Support for Dynamically Connected (DC) transport protocol
Minimizing Memory Footprint further by 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 with MVAPICH2-X 2.2a

### Memory Footprint for Alltoall

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>RC</th>
<th>DC-Pool</th>
<th>UD</th>
<th>XRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>160</td>
<td>22</td>
<td>10</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>320</td>
<td>97</td>
<td>10</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>640</td>
<td>100</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

### Normalized Execution Time

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>RC</th>
<th>DC-Pool</th>
<th>UD</th>
<th>XRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>320</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>620</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>


MUG’15
Support for Dynamic Connected Transport in MVAPICH2-X

- Requirements for Dynamic Connected
  - Mellanox ConnectIB (Dual-FDR) and ConnectX-4 (EDR) adapters
  - Mellanox OFED >= 2.4
  - Shared Receive Queue support
- This command launches test on nodes n0 and n1, two processes per node with support for Dynamic Connected Transport.
  - $ mpirun_rsh -rsh -np 4 n0 n0 n1 n1 MV2_USE_DC=1 ./test

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_USE_DC</td>
<td>• Enable use of the DC InfiniBand transport.</td>
<td>Disabled</td>
<td>• RC/XRC not used</td>
</tr>
<tr>
<td>MV2_NUM_DC_TGT</td>
<td>• Controls the number of DC receive communication objects</td>
<td>1</td>
<td>• Tuned for system size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Only change if necessary</td>
</tr>
<tr>
<td>MV2_SMALL_MSG_DC_POOL</td>
<td>• Controls the number of DC send communication objects used for transmitting small messages</td>
<td>8</td>
<td>• Tuned for system size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Only change if necessary</td>
</tr>
<tr>
<td>MV2_LARGE_MSG_DC_POOL</td>
<td>• Controls the number of DC send communication objects used for transmitting large messages</td>
<td>8</td>
<td>• Tuned for system size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Only change if necessary</td>
</tr>
</tbody>
</table>

- Refer to Support for Dynamic Connected Transport section of MVAPICH2-X user guide for more information
- http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-x-2.2a-userguide.html#x1-230005.1
Non-Blocking Collectives in MVAPICH2-X

- MVAPICH2 supports for MPI-3 Non-Blocking Collective communication and Neighborhood collective communication primitives
  - MVAPICH2 1.9 to MVAPICH2 2.2a
- 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
**Support for Core-Direct Based Non-Blocking Collectives**

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

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

K. Kandalla, et al. High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A Study with Parallel 3D FFT, ISC 2011

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

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


Modified HPL with Offload-Bcast does up to 4.5% better than default version (512 Processes)
Using Core-Direct Enabled Non-Blocking Collectives with MVAPICH2-X

- Requirements for Core Direct
  - Mellanox ConnectX(2/3/4) or ConnectIB (Dual-FDR) adapters
  - Mellanox OFED >= 2.4
- The Core-Direct feature and can be enabled or disabled globally by the use of the environment variable MV2_USE_CORE_DIRECT.
- This command launches test on nodes n0 and n1, two processes per node with support for Core-Direct based non-blocking collectives.
  - $ mpirun_rsh -rsh -np 4 n0 n0 n1 n1 MV2_USE_CORE_DIRECT=1 ./test

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_USE_CORE_DIRECT</td>
<td>• Enable core-direct support</td>
<td>Disabled</td>
<td>• Enable for best overlap</td>
</tr>
<tr>
<td>MV2_USE_CD_IALLGATHER</td>
<td>• Enable the use of core-direct for MPI_Iallgather</td>
<td>1</td>
<td>• Enable for best overlap</td>
</tr>
<tr>
<td>MV2_USE_CD_IALLTOALL</td>
<td>• Enable the use of core-direct for MPI_Ialltoall</td>
<td>1</td>
<td>• Enable for best overlap</td>
</tr>
<tr>
<td>MV2_USE_CD_IBARRIER</td>
<td>• Enable the use of core-direct for MPI_Ibarrier</td>
<td>1</td>
<td>• Enable for best overlap</td>
</tr>
<tr>
<td>MV2_USE_CD_IBCAST</td>
<td>• Enable the use of core-direct for MPI_Ibcast</td>
<td>1</td>
<td>• Enable for best overlap</td>
</tr>
<tr>
<td>MV2_USE_CD_IGATHER</td>
<td>• Enable the use of core-direct for MPI_Igather</td>
<td>1</td>
<td>• Enable for best overlap</td>
</tr>
<tr>
<td>MV2_USE_CD_ISCATTER</td>
<td>• Enable the use of core-direct for MPI_Isscatter</td>
<td>1</td>
<td>• Enable for best overlap</td>
</tr>
</tbody>
</table>

- Refer to Support for Core-Direct Based Non-Blocking Collectives section of MVAPICH2-X user guide
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-x-2.2a-userguide.html#x1-240005.2](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-x-2.2a-userguide.html#x1-240005.2)

MUG'15
MVAPICH2-X RPMs

- MVAPICH2-X RPMs available for:
  - RedHat Enterprise Linux 6 (OFED 1.5.4)
    - GCC 4.4.7 & INTEL 14.0.3
    - Compatible with OFED 1.5.4.1 and MLNX_OFED 1.5.3
  - RedHat Enterprise Linux 6 (Stock InfiniBand Packages)
    - GCC 4.4.7 & INTEL 14.0.3
    - Compatible with OFED 3.5 and RHEL6 InfiniBand packages
  - RedHat Enterprise Linux 6 (MLNX-OFED)
    - GCC 4.4.7 & INTEL 14.0.3
    - Compatible with MLNX_OFED 2.2 & 2.3
- Advanced MVAPICH2-X RPMs available for
  - Support for DC, Core-Direct and OSU INAM
  - RedHat Enterprise Linux 6 (MLNX-OFED)
    - GCC 4.4.7 & INTEL 14.0.3
    - Compatible with MLNX_OFED 2.4
- Please contact us at mvapich-help@cse.ohio-state.edu for other platforms
- RHEL RPMs are compatible with CentOS as well
- MVAPICH2-X RPMs are relocatable
Downloading and Installing MVAPICH2-X

• Downloading MVAPICH2-X RPMs
  – `wget http://mvapich.cse.ohio-state.edu/download/mvapich/mv2x/mvapich2-x-2.2a-1.rhel6.ofed-1.5.4.tar.gz`

• Tarball contents:
  – GNU and Intel RPMs for MVAPICH2-X, OpenSHMEM, and UPC
  – OpenUH based RPM for CAF

• Install using rpm command
  – `rpm --Uvh [--prefix=install-path] *.rpm --force --nodeps`
  – Default installation location is `/opt/mvapich2-x`
Compiling programs with MVAPICH2-X

- Compile MPI programs using mpicc
  
  - $ mpicc -o helloworld_mpi helloworld_mpi.c

- Compile UPC programs using upcc
  
  - $ upcc -o helloworld_upc helloworld_upc.c

- Compile OpenSHMEM programs using oshcc
  
  - $ oshcc -o helloworld_oshm helloworld_oshm.c

- Compile CAF programs using OpenUH CAF compiler
  
  - $ uhcaf --layer=gasnet-mvapich2x -o helloworld_caf helloworld_caf.t.caf

- Compile Hybrid MPI+UPC programs using upcc
  
  - $ upcc -o hybrid_mpi_upc hybrid_mpi_upc.c

- Compile Hybrid MPI+OpenSHMEM programs using oshcc
  
  - $ oshcc -o hybrid_mpi_oshm hybrid_mpi_oshm.c
Running Programs with MVAPICH2-X

- **MVAPICH2-X programs** can be run using
  - `mpirun_rsh` and `mpiexec.hydra` (MPI, UPC, OpenSHMEM and hybrid)
  - `upcrun` (UPC)
  - `oshrun` (OpenSHMEM)
  - `cafrun` (CAF)

- **Running using mpirun_rsh/mpiexec.hydra**
  - `$ mpirun rsh -np 4 -hostfile hosts ./test` --- `$ mpiexec -f hosts -n 2 ./test`

- **Running using upcrun**
  - `$ export MPIRUN_CMD="<path-to-MVAPICH2-X-install>/bin/mpiexec_rsh -np %N -hostfile hosts %P %A"`
  - `$ upcrun -n 2 ./test`

- **Running using oshrun**
  - `$ oshrun -f hosts -np 2 ./test`

- **Running using cafrun**
  - Export the PATH and LD_LIBRARY_PATH of the GNU version of MVAPICH2-X
  - `$ cafrun -n 16 -v ./test`
OpenSHMEM Startup Performance on TACC Stampede

- Near-constant OpenSHMEM initialization cost
- 89 times improvement at 8,192 processes (512 nodes)
- Designs already publicly available in MVAPICH2-X 2.1rc1
OpenSHMEM Hello World Performance on TACC Stampede

- PMI Exchange and Connection setup costs can be overlapped with computation
- 8.31 times improvement at 8,192 processes (512 nodes)

“On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI”
S. Chakraborty, H. Subramoni, J. Perkins, A. Awan and D. K. Panda
The 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS ‘15)
Microbenchmark Level Performance

Reduce (256 processes)

Broadcast (256 processes)

Collect (256 processes)

Barrier

Message Size

OpenSHMEM-GASNet (Linear)
OpenSHMEM-GASNet (Tree)
OpenSHMEM-OSU
Application Level Performance with Graph500 and Sort

Graph500 Execution Time

- Performance of Hybrid (MPI+OpenSHMEM) Graph500 Design
  - 8,192 processes
    - 2.4X improvement over MPI-CSR
    - 7.6X improvement over MPI-Simple
  - 16,384 processes
    - 1.5X improvement over MPI-CSR
    - 13X improvement over MPI-Simple

Sort Execution Time

- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - 4,096 processes, 4 TB Input Size
    - MPI – 2408 sec; 0.16 TB/min
    - Hybrid – 1172 sec; 0.36 TB/min
    - 51% improvement over MPI-design

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC’13), June 2013
Hybrid MPI+OpenSHMEM Sort Application

• Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  • Execution Time (seconds)
    - 1TB Input size at 8,192 cores: MPI – 164, Hybrid-SR (Simple Read) – 92.5, Hybrid-ER (Eager Read) - 90.36
    - 45% improvement over MPI-based design
  • Weak Scalability (configuration: input size of 1TB per 512 cores)
    - At 4,096 cores: MPI – 0.25 TB/min, Hybrid-SR – 0.34 TB/min, Hybrid-SR – 0.38 TB/min
    - 38% improvement over MPI based design

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC’13), June 2013

MUG'15
MiniMD – Total Execution Time

- Hybrid design performs better than MPI implementation
- 1,024 processes
  - 17% improvement over MPI version
- Strong Scaling
  Input size: 128 * 128 * 128

M. Li, J. Lin, X. Lu, K. Hamidouche, K. Tomko and D. K. Panda, Scalable MiniMD Design with Hybrid MPI and OpenSHMEM, OpenSHMEM User Group Meeting (OUG ‘14), held in conjunction with 8th International Conference on Partitioned Global Address Space Programming Models, (PGAS 14).
Hybrid MPI+UPC NAS-FT

- Modified NAS FT UPC all-to-all pattern using MPI_Alltoall
- Truly hybrid program
- For FT (Class C, 128 processes)
  - 34% improvement over UPC-GASNet
  - 30% improvement over UPC-OSU

Hybrid MPI + UPC Support

Available since MVAPICH2-X 1.9

J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, Fourth Conference on Partitioned Global Address Space Programming Model (PGAS ’10), October 2010
Performance Evaluations for One-sided Communication

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

MVAPICH2-X: FAQs

• Inadequate shared heap size
  – Set appropriate heap size

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPC_SHARED_HEAP_SIZE</td>
<td>Set UPC shared heap size</td>
<td>64M</td>
</tr>
<tr>
<td>OOSHM_USE_SHARED_HEAP_SIZE</td>
<td>Set OpenSHMEM symmetric heap size</td>
<td>512M</td>
</tr>
</tbody>
</table>

• Can’t install mvapich2-x rpm in /opt
  – Use “—prefix” option when installing via rpm command

• Refer to OpenSHMEM Runtime Parameters section of MVAPICH2 user guide for more information
• http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-x-2.2a-userguide.html#x1-500008.4
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-X
  - MVAPICH2-GDR
    • Designs: Overview, Performance and Tuning
    • Usage: Build, Initialization and Cleanup
    • Support for MPI + OpenACC
  - MVAPICH2-MIC
  - MVAPICH2-Virt
  - OMB

• Preview of Upcoming Releases

• Overview of MPIT, Configuration and Debugging Support

• Conclusions and Final Q&A
MPI + CUDA - Naive

- Data movement in applications with standard MPI and CUDA interfaces

At Sender:

cudaMemcpy(s_hostbuf, s_devbuf, ...);
MPI_Send(s_hostbuf, size, ...);

At Receiver:

MPI_Recv(r_hostbuf, size, ...);
cudaMemcpy(r_devbuf, r_hostbuf, ...);

High Productivity and Low Performance
**MPI + CUDA - Advanced**

- Pipelining at user level with non-blocking MPI and CUDA interfaces

**At Sender:**

```c
for (j = 0; j < pipeline_len; j++)
    cudaMemcpyAsync(s_hostbuf + j * blk, s_devbuf + j * blksz, …);
for (j = 0; j < pipeline_len; j++) {
    while (result != cudaSuccess) {
        result = cudaStreamQuery(…);
        if(j > 0) MPI_Test(…);
    }
    MPI_Isend(s_hostbuf + j * block_sz, blksz …);
} MPI_Waitall();

<<Similar at receiver>>
```

*Low Productivity and High Performance*
GPU-Aware MPI Library: MVAPICH2-GDR

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

At Sender:
MPI_Send(s_devbuf, size, …);

At Receiver:
MPI_Recv(r_devbuf, size, …);

High Performance and High Productivity
CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.2 Releases

- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers
GPU-Direct RDMA (GDR) with CUDA

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

SNB E5-2670 / IVB E5-2680V2

SNB E5-2670
- P2P write: 5.2 GB/s
- P2P read: < 1.0 GB/s

IVB E5-2680V2
- P2P write: 6.4 GB/s
- P2P read: 3.5 GB/s
MVAPICH2-GDR 2.1rc2

- Released on 06/24/2015
- Major Features and Enhancements
  - Based on MVAPICH2-2.1rc2
  - CUDA 7.0 compatibility
  - CUDA-Aware support for MPI_Rsend and MPI_Irsend primitives
  - Parallel intranode communication channels (shared memory for H-H and GDR for D-D)
  - Optimized H-H, H-D and D-H communication
  - Optimized intranode D-D communication
  - Optimization and tuning for point-point and collective operations
  - Update to sm_20 kernel optimization for Datatype processing
  - Optimized design for GPU based small message transfers
  - Adding R3 support for GPU based packetized transfer
  - Enhanced performance for small message host-to-device transfers
  - Support for MPI_Scan and MPI_Exscan collective operations from GPU buffers
  - Optimization of collectives with new copy designs
Performance of MVAPICH2-GDR with GPU-Direct-RDMA

**GPU-GPU Internode Small Message Latency**

- MV2-GDR2.1RC2
- MV2-GDR2.0b
- MV2 w/o GDR

Latency (us) vs Message Size (bytes)

- 9.3X
- 3.5X
- 2.15 usec

**GPU-GPU Internode MPI Uni-Directional Bandwidth**

- MV2-GDR2.1RC2
- MV2-GDR2.0b
- MV2 w/o GDR

Bandwidth (MB/s) vs Message Size (bytes)

- 11x
- 2x

**GPU-GPU Internode Bi-directional Bandwidth**

- MV2-GDR2.1RC2
- MV2-GDR2.0b
- MV2 w/o GDR

Bandwidth (MB/s) vs Message Size (bytes)

- 11x
- 2x

MVAPICH2-GDR-2.1RC2
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
• 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
MPI3-RMA Performance of MVAPICH2-GDR with GPU-Direct-RDMA

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

MPI-3 RMA provides flexible synchronization and completion primitives

![Small Message Latency Chart](chart.png)

**Small Message Latency**

- MV2-GDR2.1RC2
- MV2-GDR2.0b

**Message Size (bytes)**

**Latency (us)**

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

• MVAPICH2-GDR 2.1RC2 with GDR support can be downloaded from https://mvapich.cse.ohio-state.edu/download

• System software requirements
  • Mellanox OFED 2.1 or later
  • NVIDIA Driver 331.20 or later
  • NVIDIA CUDA Toolkit 6.0 or later
  • Plugin for GPUDirect RDMA
    – Strongly Recommended: use the new GDRCOPY module from NVIDIA
      • https://github.com/NVIDIA/gdrcopy
  • Has optimized designs for point-to-point communication using GDR
  • Contact MVAPICH help list with any questions related to the package
    mvapich-help@cse.ohio-state.edu
## Pipelined Data Movement in MVAPICH2-GDR: Tuning

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

- 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_USE_GPUDIRECT</td>
<td>• Enable / Disable GDR-based designs</td>
<td>1 (Enabled)</td>
<td>• Always enable</td>
</tr>
<tr>
<td>MV2_GPUDIRECT_LIMIT</td>
<td>• Controls messages size until which GPUDirect RDMA is used</td>
<td>8 KByte</td>
<td>• Tune for your system</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• GPU type, host architecture and CUDA version: impact pipelining overheads and P2P bandwidth bottlenecks</td>
</tr>
</tbody>
</table>

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

- Works between GPUs within the same socket or IOH

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_CUDA_IPC</td>
<td>• Enable / Disable CUDA IPC-based designs</td>
<td>1 (Enabled)</td>
<td>• Always leave set to 1</td>
</tr>
<tr>
<td>MV2_CUDA_SMP_IPC</td>
<td>• Enable / Disable CUDA IPC fastpath design for short messages</td>
<td>0 (Disabled)</td>
<td>• Benefits Device-to-Device transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Hurts Device-to-Host/Host-to-Device transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Always set to 1 if application involves only Device-to-Device transfers</td>
</tr>
</tbody>
</table>

Intranode osu_latency small

MV2_CUDA_IPC

MV2_CUDA_SMP_IPC
Enabling Support for MPI Communication from GPU Memory

• Configuring the support
  – --enable-cuda --with-cuda=<path-to-cuda-installation>
  – With PGI compilers
    • PGI does not handle an asm string leading to linker issues with CUDA kernels in MVAPICH2 library
    • --enable-cuda=basic --with-cuda=<path-to-cuda-installation>

• Enabling the support at runtime
  – Disabled by default to avoid pointer checking overheads for Host communication
  – Set MV2_USE_CUDA=1 to enable support of communication from GPU memory
GPU Device Selection, Initialization and Cleanup

- MVAPICH2-GDR 1.9 and earlier versions require GPU device to be selected before MPI_Init
  - To allocate and initialize required GPU resources in MPI_Init
  - Node rank information is exposed by the launchers (mpirun_rsh or hydra) as MV2_COMM_WORLD_LOCAL_RANK
    
    ```
    int local_rank = atoi(getenv("MV2_COMM_WORLD_LOCAL_RANK"));
    cudaSetDevice (local_rank% num_devices)
    ```

- Restriction removed from MVAPICH2-GDR 2.0 onwards
  - Does GPU resource initialization dynamically
  - Applications can select the device before or after MPI_Init

- CUDA allocates resources like shared memory files which require explicit cleanup
  - cudaDeviceReset or cuCtxDestroy
  - Applications have to do this after MPI_Finalize to allow MVAPICH2 runtime to deallocate resources
OpenACC

- OpenACC is gaining popularity
- A set of compiler directives (#pragma)
- Offload specific loops or parallelizable sections in code onto accelerators
  
  ```c
  #pragma acc region
  {
      for(i = 0; i < size; i++) {
          A[i] = B[i] + C[i];
      }
  }
  ```

- Routines to allocate/free memory on accelerators
  ```c
  buffer = acc_malloc(MYBUFSIZE);
  acc_free(buffer);
  ```

- Supported for C, C++ and Fortran
- Huge list of modifiers – `copy, copyout, private, independent, etc..`
Using MVAPICH2-GDR with OpenACC 1.0

- `acc_malloc` to allocate device memory
  - No changes to MPI calls
  - MVAPICH2 detects the device pointer and optimizes data movement
  - Delivers the same performance as with CUDA

```c
A = acc_malloc(sizeof(int) * N);

......

#pragma acc parallel loop deviceptr(A) . . .
//compute for loop

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

......

acc_free(A);
```
Using MVAPICH2-GDR with OpenACC 2.0

- `acc_deviceptr` to get device pointer (in OpenACC 2.0)
  - Enables MPI communication from memory allocated by compiler when it is available in OpenACC 2.0 implementations
  - MVAPICH2 will detect the device pointer and optimize communication
  - Delivers the same performance as with CUDA

```c
A = malloc(sizeof(int) * N);

......

#pragma acc data copyin(A) . . .
{

#pragma acc parallel loop . . .
//compute for loop
MPI_Send(acc_deviceptr(A), N, MPI_INT, 0, 1, MPI_COMM_WORLD);

}

......
free(A);
```
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  – MVAPICH2
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases

• Overview of MPIT, Configuration and Debugging Support

• Conclusions and Final Q&A
MPI Applications on MIC Clusters

• Flexibility in launching MPI jobs on clusters with Xeon Phi

Multi-core Centric

Host-only

Offload (/reverse Offload)

Symmetric

Coprocessor-only

Many-core Centric

Xeon

Host-only

MPI Program

MPI Program

MPI Program

Offloaded Computation

Xeon Phi

Host-only

MPI Program

MPI Program

MPI Program
Data Movement on Intel Xeon Phi Clusters

• Connected as PCIe devices – Flexibility but Complexity

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

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

• Critical for runtimes to optimize data movement, hiding the complexity
MVAPICH2-MIC Design for Clusters with IB and MIC

- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only
  - Symmetric Mode
- Multi-MIC Node Configurations
MVAPICH2-MIC 2.0

- Released on 12/02/2014

- Major Features and Enhancements
  - Based on MVAPICH2 2.0.1
  - Support for native, symmetric and offload modes of MIC usage
  - Optimized intra-MIC communication using SCIF and shared-memory channels
  - Optimized intra-Node Host-to-MIC communication using SCIF and IB channels
  - Enhanced mpirun_rsh to launch jobs in symmetric mode from the host
  - Support for proxy-based communication for inter-node transfers
    - Active-proxy, 1-hop and 2-hop designs (actively using host CPU)
    - Passive-proxy (passively using host CPU)
  - Support for MIC-aware MPI_Bcast()
  - Improved SCIF performance for pipelined communication
  - Optimized shared-memory communication performance for single-MIC jobs
  - Supports an explicit CPU-binding mechanism for MIC processes
  - Tuned pt-to-pt intra-MIC, intra-node, and inter-node transfers
  - Supports hwloc v1.9

- Running on three major systems
  - Stampede
  - Blueridge (Virginia Tech)
  - Beacon (UTK)
MIC-Remote-MIC P2P Communication with Proxy-based Communication

Intra-socket P2P

Latency (Large Messages)

- MV2
- MV2-MIC

Bandwidth

Better

Inter-socket P2P

Latency (Large Messages)

- MV2
- MV2-MIC

Bandwidth

Better

Latency (usec)

Message Size (Bytes)

0 8K 32K 128K 512K 2M

0 1000 2000 3000 4000 5000

0 16 256 4K 64K 1M

0 2000 4000 6000

0 1 16 256 4K 64K 1M

0 5236

Better

MUG'15
Optimized MPI Collectives for MIC Clusters (Allgather & Alltoall)


MUG’15
Latest Status on MVAPICH2-MIC

• Running on three major systems
  – Stampede
  – Blueridge (Virginia Tech)
  – Beacon (UTK)

• Public version of MVAPICH2-MIC 2.0 is released (12/02/14)

• Try it out!!
  – http://mvapich.cse.ohio-state.edu/downloads/#mv2mic
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  – MVAPICH2
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases

• Overview of MPIT, Configuration and Debugging Support

• Conclusions and Final Q&A
Can HPC and Virtualization be Combined?

• Virtualization has many benefits
  – Job migration
  – Compaction

• Not very popular in HPC due to overhead associated with Virtualization

• New SR-IOV (Single Root – IO Virtualization) support available with Mellanox InfiniBand adapters

• Enhanced MVAPICH2 support for SR-IOV

• Publicly available as MVAPICH2-Virt library

J. Zhang, X. Lu, J. Jose, R. Shi and D. K. Panda, Can Inter-VM Shmem Benefit MPI Applications on SR-IOV based Virtualized InfiniBand Clusters? EuroPar’14
J. Zhang, X. Lu, J. Jose, M. Li, R. Shi and D.K. Panda, High Performance MPI Libray over SR-IOV enabled InfiniBand Clusters, HiPC’14
• Enables high-performance communication in virtualized environments
  – SR-IOV based communication for Inter-Node MPI communication
  – Inter-VM Shared Memory (IVSHMEM) based communication for Intra-Node-Inter-VM MPI communication
  – http://mvapich.cse.ohio-state.edu
MVAPICH2-Virt 2.1rc2

• Released on 06/26/2015

• Major Features and Enhancements
  – Based on MVAPICH2 2.1rc2
  – Support for efficient MPI communication over SR-IOV enabled InfiniBand network
  – High-performance and locality-aware MPI communication with IVSHMEM
  – Support for IVSHMEM device auto-detection in virtual machine
  – Automatic communication channel selection among SR-IOV, IVSHMEM, and CMA/LiMIC2
  – Support for easy configuration through runtime parameters
  – Tested with - Mellanox InfiniBand adapters (ConnectX-3 (56Gbps))
• Running the following rpm command will install the software in /opt/mvapich2/virt/2.1/ directory.
  – $ rpm -Uvh mvapich2-virt-2.1-0.1.rc2.el6.x86_64.rpm --force --nodeps

• The RPMs contained in our packages are relocatable and can be installed using a prefix other than the default of /opt/mvapich2/virt/ used by the package in the previous example.

• Install package specifying custom prefix
  – $ rpm --prefix /custom/install/prefix -Uvh --nodeps <package>.rpm

• If you do not have root permission you can use rpm2cpio to extract the package.

• Use rpm2cpio to extract the package
  – $ rpm2cpio <package>.rpm | cpio -id

• Refer to Installing MVAPICH2-Virt Package section of MVAPICH2-Virt user guide for more information
  • http://mvapich.cse.ohio-state.edu/userguide/virt/#_installing_mvapich2_virt_package
Compiling and Running Programs with MVAPICH2-Virt

• Provides variety of MPI compilers
  – mpicc, mpif77, mpiCC, or mpif90

• Prerequisites for running
  – Either ssh or rsh should be enabled between the front nodes/VMs and the computing VMs. In addition to this setup, you should be able to login to the remote VMs without any password prompts.
  – All host names should resolve to the same IP address on all machines. For instance, if a machine’s host names resolves to 127.0.0.1 due to the default /etc/hosts on some Linux distributions it leads to incorrect behavior of the library.

• This command launches cpi on VMs vm0 and vm1, two processes per node. By default ssh is used.
  – $ mpirun_rsh -rsh -np 4 vm0 vm0 vm1 vm1 ./cpi

• Refer to Basic Usage Instructions section of MVAPICH2-Virt user guide for more information
  • http://mvapich.cse.ohio-state.edu/userguide/virt/#_basic_usage_instructions
### Advanced Usage Instructions

- Provides different parameters for enabling/disabling advanced features

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_VIRT_USE_IVSHMEM</td>
<td>• Enable / Disable IVSHMEM-based optimizations</td>
<td>Enabled</td>
<td>• Always Enable</td>
</tr>
<tr>
<td>MV2_VIRT_IVSHMEM_DEVICE</td>
<td>• Identify an IVSHMEM device if multiple devices are available</td>
<td>Auto select</td>
<td>• None</td>
</tr>
</tbody>
</table>

- Refer to [Advanced Usage Instructions](http://mvapich.cse.ohio-state.edu/userguide/virt/#_advanced_usage_instructions) section of MVAPICH2-Virt user guide for more information
IB-Level Performance – Inter-node Pt2Pt Latency & Bandwidth

- Native vs. SR-IOV at basic IB level
- Compared to native latency, 0-24% overhead
- Compared to native bandwidth, 0-17% overhead
EC2 C3.2xlarge instances

Similar performance with MV2-SR-IOV-Def

Compared to Native, similar overhead as basic IB level

Compared to EC2, up to 29X and 16X performance speedup on Latency & Bandwidth
EC2 C3.2xlarge instances

- Compared to MV2-SR-IOV-Def, up to 84% and 158% performance improvement on Latency & Bandwidth
- Compared to Native, 3-7% overhead for Latency, 3-8% overhead for Bandwidth
- Compared to EC2, up to 160X and 28X performance speedup on Latency & Bandwidth
Application-Level Performance (8 VM * 8 Core/VM)

- Compared to Native, 1-9% overhead for NAS
- Compared to Native, 4-9% overhead for Graph500
NSF Chameleon Cloud: A Powerful and Flexible Experimental Instrument

- Large-scale instrument
  - Targeting Big Data, Big Compute, Big Instrument research
  - ~650 nodes (~14,500 cores), 5 PB disk over two sites, 2 sites connected with 100G network

- Reconfigurable instrument
  - Bare metal reconfiguration, operated as single instrument, graduated approach for ease-of-use

- Connected instrument
  - Workload and Trace Archive
  - Partnerships with production clouds: CERN, OSDC, Rackspace, Google, and others
  - Partnerships with users

- Complementary instrument
  - Complementing GENI, Grid’5000, and other testbeds

- Sustainable instrument
  - Industry connections

http://www.chameleoncloud.org/
Presentation Overview

- Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-X
  - MVAPICH2-GDR
  - MVAPICH2-MIC
  - MVAPICH2-Virt
  - OMB

- Preview of Upcoming Releases

- Overview of MPIT, Configuration and Debugging Support

- Conclusions and Final Q&A
OSU Microbenchmarks v5.0

- OSU Micro-Benchmarks 5.0 (08/18/15)
- Non-blocking collective benchmarks
  - osu_iallgather, osu_ialltoall, osu.ibarrier, osu_ibcast, osu_igather, and osu_iscatter
  - Benchmarks can display the amount of achievable overlap
    - Overlap defined as the amount of computation that can be performed while the communication progresses in the background
    - Have the additional option: "-t" set the number of MPI_Test() calls during the dummy computation
      - set CALLS to 100, 1000, or any number > 0
- Startup benchmarks
  - osu_init
    - Measures the time taken for an MPI to complete MPI_Init
  - osu_hello
    - Measures the time taken for an MPI library to complete a simple hello world MPI program
Non-blocking Collective Benchmarks

• Objective
  • Estimation of overlap potential available to the MPI application as function of system size and message size
  • Indication of the latency in executing a non-blocking collective

• Breakdown
  • Initiation of NBC
  • Compute time selection
  • Pure communication time
  • Wait time
OSU Microbenchmarks – UPC and OpenSHMEM

- OpenSHMEM benchmarks
  - osu_oshm_put – Put latency
  - osu_oshm_get – Get latency
  - osu_oshm_put_mr – Put message rate
  - osu_oshm_atomics – Atomics latency
  - osu_oshm_collect – Collect latency
  - osu_oshm_broadcast – Broadcast latency
  - osu_oshm_reduce - Reduce latency
  - osu_oshm_barrier - Barrier latency

- UPC benchmarks
  - osu upc memput – Put latency
  - osu upc memget - Get latency
CUDA and OpenACC Extensions in OMB

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

• Configuring with GPU support
  – Enabling CUDA support: “--enable-cuda --with-cuda-include=<path-to-CUDA-headers> --with-cuda-libraries=<path-to-CUDA-libraries>”
  – Enabling OpenACC support: “--enable-openacc”
  – Both enabled in integrated version when library is build with CUDA support

• Running the benchmarks
  – “pt2pt-benchmark [options] [RANK0 RANK1]”
  – “collective-benchmark [options]”

• Option to select between CUDA and OpenACC: “-d [cuda|openacc]”

• Parameters to select location of buffers
  – Pt2pt benchmarks support selection at each rank: “D D, D H, H D, H H”
  – The –d option enables use of device buffers in collective benchmarks
Examples

Consider two GPU nodes: \( n1 \) and \( n2 \) each with two GPUs

Measure internode GPU-to-GPU latency using CUDA

\[
\text{mpirun-rsh} \ --np \ 2 \ n1 \ n2 \ MV2\_USE\_CUDA=1 \ ./\text{get}\_local\_rank./osu\_latency \ D \ D
\]

\[
\text{mpirun-rsh} \ --np \ 2 \ n1 \ n2 \ MV2\_USE\_CUDA=1 \ ./\text{get}\_local\_rank./osu\_latency \ --d \ \text{cuda} \ D \ D
\]

Measure internode GPU-to-GPU latency using OpenACC

\[
\text{mpirun-rsh} \ --np \ 2 \ n1 \ n2 \ MV2\_USE\_CUDA=1 \ ./\text{get}\_local\_rank./osu\_latency \ --d \ \text{openacc} \ D \ D
\]

Measure internode GPU-to-Host latency using CUDA

\[
\text{mpirun-rsh} \ --np \ 2 \ n1 \ n2 \ MV2\_USE\_CUDA=1 \ ./\text{get}\_local\_rank./osu\_latency \ D \ H
\]

Measure intra node GPU-to-GPU latency

\[
\text{mpirun-rsh} \ --np \ 2 \ n1 \ n1 \ MV2\_USE\_CUDA=1 \ ./\text{get}\_local\_rank./osu\_latency \ D \ D
\]

Measure MPI_Alltoall latency between GPUs with CUDA

\[
\text{mpirun-rsh} \ --np \ 4 \ n1 \ n1 \ n2 \ n2 \ MV2\_USE\_CUDA=1 \ ./\text{get}\_local\_rank./osu\_alltoall \ --d \ \text{cuda}
\]
Presentation Overview

• **Installation Guidelines, Runtime Optimization and Tuning Flexibility in**
  - MVAPICH2
  - MVAPICH2-X
  - MVAPICH2-GDR
  - MVAPICH2-MIC
  - MVAPICH2-Virt
  - OMB

• **Preview of Upcoming Releases**
  - OSU INAM
    - Energy-aware MVAPICH2 and Tools

• **Overview of MPIT, Configuration and Debugging Support**

• **Conclusions and Final Q&A**
Overview of OSU INAM

- OSU INAM monitors IB clusters in real time by querying various subnet management entities in the network.
- Major features of the OSU INAM tool include:
  - Analyze and profile network-level activities with many parameters (data and errors) at user specified granularity.
  - Capability to analyze and profile node-level, job-level and process-level activities for MPI communication (pt-to-pt, collectives and RMA).
  - Remotely monitor CPU utilization of MPI processes at user specified granularity.
  - Visualize the data transfer happening in a "live" fashion - Live View for
    - Entire Network - Live Network Level View
    - Particular Job - Live Job Level View
    - One or multiple Nodes - Live Node Level View
  - Capability to visualize data transfer that happened in the network at a time duration in the past - Historical View for
    - Entire Network - Historical Network Level View
    - Particular Job - Historical Job Level View
    - One or multiple Nodes - Historical Node Level View
OSU INAM – Network Level View

- 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

Full Network (152 nodes)

Zoomed-in View of the Network
OSU INAM – Job and Node Level Views

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 provides 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

Finding Routes Between Nodes
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  – MVAPICH2
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases
  – OSU INAM
  – Energy-aware MVAPICH2 and Tools
    • MVAPICH2-EA
    • OMET

• Overview of MPIT, Configuration and Debugging Support

• Conclusions and Final Q&A
**MVAPICH2-EA & OEMT**

- MVAPICH2-EA (Energy-Aware)
  - A white-box approach
  - New Energy-Efficient communication protocols for pt-pt and collective operations
  - Intelligently apply the appropriate Energy saving techniques
  - Application oblivious energy saving
- OEMT
  - A library utility to measure energy consumption for MPI applications
  - Works with all MPI runtimes
  - PRELOAD option for precompiled applications
  - Does not require ROOT permission:
    - A safe kernel module to read only a subset of MSRs
MV2-EA : Application Oblivious Energy-Aware-MPI (EAM)

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

OSU Energy Monitoring Tool (OEMT)

- **The library: liboemt.so**
  - MPI aware Energy monitoring utility
  - Preload option: the most generic and works even with a pre-compiled application that you do not have code
  - Linking to application during the compilation time
  - Reports the energy consumed at the end of the application
  - Different options about the energy domain (counter) to report

- **The kernel module: koemt**
  - Simple and safe kernel that enables the read of the MSR without ROOT
  - Tight interaction with liboemt.so
Example of OEMT Usage

$ $MV2_PATH/bin/mpirun_rsh -n 2 hostA hostB OEMT_MSR_DOMAIN=3 OEMT_MSR_VERBOSITY=2
LD_PRELOAD=$MV2_PATH/lib/liboemt.so ./osu_latency

# OSU MPI Bandwidth Test
# Size        Bandwidth (MB/s)
1            0.84
2            1.75
4            3.52

........
2097152      5566.22
4194304      5672.54

<0> (PP0) = 16.310346 Joules
<0> (PKG) = 49.491838 Joules
<1> (PP0) = 18.169216 Joules
<0> (DRAM) = 4.375097 Joules
<1> (PKG) = 50.048855 Joules
<1> (DRAM) = 4.609476 Joules
Presentation Overview

• Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  – MVAPICH2
  – MVAPICH2-X
  – MVAPICH2-GDR
  – MVAPICH2-MIC
  – MVAPICH2-Virt
  – OMB

• Preview of Upcoming Releases

• Overview of MPIT, Configuration and Debugging Support
  – Advanced Optimization and Tuning of MPI Applications using MPI-T
  – User Resources
  – Frequently reported issues and Common mistakes
  – Useful Diagnostics
  – Performance Troubleshooting
  – Getting help and Bug report detail

• Conclusions and Final Q&A
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
• Control Variables (CVARS) and Performance Variables (PVARs)
• More about the interface: mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

(PMPI Approach)

(MPI-T Approach)
MPI-T in MVAPICH2: Control Variables (CVARS)

- Typically used to configure and tune MPI internals
- Environment variables, configuration parameters and toggles
MPI-T in MVAPICH2: Performance Variables (PVARS)

• Insights into performance of the MPI library

• Highly-implementation specific

• Memory consumption, timing information, resource-usage, data transmission info.

• Per-call basis or an entire MPI job

Memory Usage:
- current level
- maximum watermark

InfiniBand N/W:
- #control packets
- #out-of-order packets

Pt-to-pt messages:
- unexpected queue length
- unexp. match attempts
- recvq. length

Registration cache:
- hits
- misses

Shared-memory:
- limic/ CMA
- buffer pool size & usage

Collective ops:
- comm. creation
- #algorithm invocations
[Bcast – 8; Gather – 10]

...
MPI-T workflows supported in MVAPICH2

- Several workflows based on role: End Users / Performance Tuners / MPI Implementers

- Two main workflows for Users and Tuners to improve application performance
  
  - Transparently using MPIT-Aware external tools
  
  - Co-designing applications and MPI-libraries using MPI-T

To use the MPI-T interface with MVAPICH2, add the following configure flag:

```
--enable-mpit-pvars=mv2
```
Case Study: Reducing memory footprint with UH3D

- Load-imbalance in application
- Memory usage increases drastically with scale as more VBUFs are used
- MPI-T assisted analysis attributed memory usage to statically-established RC connections
- Optimal selection of UD threshold leads to **7.5X reduction in memory usage**
- Total execution time with tuned version was consistently equal to, or lesser than, default

User Resources

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

- Job Startup issues
- MPI_Init and Other MPI errors
- Creation of CQ or QP failure
- Failed to register memory with InfiniBand HCA
- Multicast group creation failed
- InfiniBand setup issues
- MVAPICH2 over RoCE issues
- MPI + OpenMP, Multi-threaded MPI shows bad performance
Job Startup Issues

• **Symptoms**
  • [mpirun_rsh][child_handler] Error in init phase, aborting! (0/2 mpispawn connections)

• **Cause**
  • Host file is not correct
  • SSH issues

• **Troubleshooting**
  • Verify host file
  • Password less ssh
  • DNS or /etc/hosts
**MPI_Init and Other MPI errors**

- **Symptoms**
  - "Fatal error in MPI_Init: Other MPI error"

- **Cause**
  - Could be because of multiple reasons

- **Troubleshooting**
  - Reconfigure with `–enable-g=dbg –enable fast=none` to better understand the problem

```text
[cli_0]: aborting job:
Fatal error in MPI_Init:
Other MPI error, error stack:
MPIR_init_thread(408).......:
MPID_Init(308)..............: channel initialization failed
MPIDI_CH3_Init(283).........:
MPIDI_CH3I_RDMA_init(171)...:
rdma_setup_startup_ring(389): cannot open hca device
```
Creation of CQ or QP failure

• **Symptoms**
  • **libibverbs**: Warning: RLIMIT_MEMLOCK is 32768 bytes.  
    This will severely limit memory registrations.  
    Other MPI error, error stack:  
    MPIR_Init_thread(449)........:
    MPID_Init(365)..................: channel initialization failed
    MPIDI_CH3_Init(313)..........:
    MPIDI_CH3I_RDMA_init(170)....:
    rdma_setup_startup_ring(416): **cannot create cq**

• **Cause**
  • Memory buffers used in verbs operations and ib context uses pinned memory  
  • Inability to pin the required memory

• **Troubleshooting**
  • Make sure enough memory set for “max locked memory” (limit –l)  
  • recommended “unlimited” on all compute nodes

• Refer to Creation of CQ or QP failure section of MVAPICH2 user guide for more information  
• [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1380009.4.3](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1380009.4.3)
Failed to register memory with InfiniBand HCA

- **Symptoms**
  - “Cannot register vbuf region”
  - “Abort: vbuf pool allocation failed”
  - QP errors, node failures

- **Cause**
  - Limited registered (pinned) memory

- **Troubleshooting**
  - OFED parameters: log_num_mtt, log_mttts_per_seg
  - max_reg_mem = \((2^{\text{log\_num\_mtt}}) \times (2^{\text{log\_mttts\_per\_seg}}) \times \text{PAGE\_SIZE}\)
  - Some OFED default values are too low (< 2GB)
  - clusters with large physical memory (> 64)
  - **Recommendation**: increase log_num_mtt value
    - max_reg_mem = \((2^{24}) \times (2^{1}) \times (4 \text{ kB}) = 128 \text{ GB}\)
  - Refer to MVAPICH2 failed to register memory section of MVAPICH2 user guide for more information
  - [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1150009.1.4](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-1150009.1.4)
Multicast group creation failed

**Symptoms**

- [host1:mpi_rank_0][create_2level_comm] Warning: Multicast group setup failed. Not using any multicast features

**Cause**

- Umad device permission
- OpenSM issues

**Troubleshooting**

- Check umad device user permissions
  
  $ ls -l /dev/InfiniBand/umad0
  
  crw-rw-rw- 1 root root 231, 0 Aug 9 02:04 /dev/InfiniBand/umad0

- Slow opensm response
  - MV2_MCAST_COMM_INIT_TIMEOUT
- Maximum multicast groups reached (very unlikely). Check OpenSM logs

- Refer to Running Collectives with Hardware based Multicast section of MVAPICH2 user guide
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-600006.8](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-600006.8)
InfiniBand setup issues

- **Symptoms**
  - [0->6150] send desc error, wc_opcode=0
  - [0->6150] wc.status=12, wc.opcode=0, vbuf->phead->type=25 = XXXX
  - [4979] Abort: [] Got completion with error 12, vendor code=0x81, dest rank=6150
  - wc.status : 12 (IBV_WC_RETRY_EXC_ERR), 13 (IBV_WC_RNR_RETRY_EXC_ERR)

- **Cause**
  - Bad QP attributes
  - Loose cable, bad HCA or a bad switch blade
  - Remote side is in a bad state
  - Heavy congestion in the network

- **Troubleshooting**
  - MV2_DEFAULT_RETRY_COUNT
  - Map src, dest ranks to host file and check those specific nodes
MVAPICH2 over RoCE issues

• **Symptoms**
  • Intermittent hangs

• **Cause**
  • Most likely setup issues

• **Troubleshooting**
  • Requires loss-less Ethernet fabric
  • Configure Ethernet switch to treat RoCE traffic as loss-less
  • Create a separate VLAN interface
  • All VLAN interfaces appear as additional GID index
  • Select non-default GID index with MV2_DEFAULT_GID_INDEX
  • Use VLAN IP addresses in `/etc/mv2.conf` in RDMA CM mode

• Refer to [Run with mpirun_rsh using OFA-RoCE section of MVAPICH2 user guide for more information](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-370005.2.7)
MPI + OpenMP, Multi-threaded MPI shows bad performance

• **Symptoms**
  Poor performance, hangs

• **Cause**
  CPU affinity enabled by default
  All OpenMP, pthreads in the application process bind to same core

• **Troubleshooting**
  • Turn off affinity
    MV2_ENABLE_AFFINITY = 0
  • Choose binding level
    MV2_CPU_BINDING_LEVEL=socket

• Refer to Running with Efficient CPU (Core) Mapping section of MVAPICH2 user guide for information
  • [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-540006.5](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.2a-userguide.html#x1-540006.5)
Useful Diagnostics

- What parameters are being used by my job?
- Where is the segmentation fault?
- What is the peak memory used by my app?
- Is process binding working as expected?
What parameters are being used by my job?

- **MV2_SHOW_ENV_INFO**
  - Show values of the run time parameters
  - 1 (short list), 2 (full list)

- **Example**

  ```
  $ mpirun_rsh -np 2 --hostfile hfile MV2_SHOW_ENV_INFO=1 ./exec
  ```

MVAPICH2-2.0b Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESSOR_ARCH_NAME</td>
<td>MV2_ARCH_INTEL_XEON_E5_2680_16</td>
</tr>
<tr>
<td>HCA_NAME</td>
<td>MV2_HCA_MLX_CX_FDR</td>
</tr>
<tr>
<td>HETEROGENEOUS</td>
<td>NO</td>
</tr>
<tr>
<td>MV2_VBUF_TOTAL_SIZE</td>
<td>17408</td>
</tr>
<tr>
<td>MV2_IBA_EAGER_THRESHOLD</td>
<td>17408</td>
</tr>
<tr>
<td>MV2_RDMA_FAST_PATH_BUF_SIZE</td>
<td>5120</td>
</tr>
<tr>
<td>MV2_EAGERSIZE_1SC</td>
<td>8192</td>
</tr>
<tr>
<td>MV2_PUT_FALLBACK_THRESHOLD</td>
<td>8192</td>
</tr>
<tr>
<td>MV2_GET_FALLBACK_THRESHOLD</td>
<td>0</td>
</tr>
<tr>
<td>MV2_SMP_EAGERSIZE</td>
<td>8193</td>
</tr>
<tr>
<td>MV2_SMP_EAGERSIZE_1SC</td>
<td>8193</td>
</tr>
<tr>
<td>MV2_SMP_LENGTH_QUEUE</td>
<td>524288</td>
</tr>
<tr>
<td>MV2_SMP_NUM_SEND_BUFFER</td>
<td>16</td>
</tr>
<tr>
<td>MV2_SMP_BATCH_SIZE</td>
<td>8</td>
</tr>
</tbody>
</table>
Where is the segmentation fault?

- **MV2_DEBUG_SHOW_BACKTRACE**
  - Shows backtrace with debug builds (--enable-g=dbg, --enable-fast=none)

- **Example**
  - segmentation fault report with out much information
    ```
    [host1:mpi_rank_0][error_sighandler] Caught error: Segmentation fault (signal 11)
    ```
  - mpirun_rsh -np 2 --hostfile hfile MV2_DEBUGSHOW_BACKTRACE=1 ./exec
    ```
    [error_sighandler] Caught error: Segmentation fault (signal 11)
    ```
    ```
    [print_backtrace] 0: libmpich.so.10(print_backtrace+0x22) [0x2af447e29d9a]
    [print_backtrace] 1: libmpich.so.10(error_sighandler+0x7c) [0x2af447e29ef2]
    [print_backtrace] 2: libmpich.so.10(allocate_vbufs+0x71) [0x2af447de6d9f]
    [print_backtrace] 3: libmpich.so.10(rdma_iba_allocate_memory+0x101) [0x2af447dd5ca2]
    [print_backtrace] 4: libmpich.so.10(MPIDI_CH3_RDMA_init+0x1569) [0x2af447dce9f1]
    [print_backtrace] 5: libmpich.so.10(MPIDI_CH3_Init+0x406) [0x2af447da32f4]
    [print_backtrace] 6: libmpich.so.10(MPID_Init+0x31f) [0x2af447d8a91b]
    [print_backtrace] 7: libmpich.so.10(MPIR_Init_thread+0x3e0) [0x2af447f90aca]
    [print_backtrace] 8: libmpich.so.10(MPI_Init+0x1de) [0x2af447f8f645]
    [print_backtrace] 9: ./mpi_hello() [0x400746]
    ```

Where is the segmentation fault?
What is the peak memory used by my app?

- **MV2_DEBUG_MEM_USAGE_VERBOSE**
  - Show memory usage statistics
  - 1 (rank 0 usage), 2 (all ranks)

- **Example**

```bash
$ mpirun_rsh -np 2 -hostfile hfile MV2_DEBUG_MEM_USAGEVERBOSE=1 ./exec
```

```
[mv2_print_mem_usage] VmPeak: 79508 kB VmHWM: 16340 kB
[mv2_print_vbuf_usage_usage] RC VBUFS:512 UD VBUFS:0 TOT MEM:8828 kB
```
Is process binding working as expected?

- **MV2_SHOW_CPU_BINDING**
  - Display CPU binding information
  - Launcher independent

- **Examples**
  - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=scatter
    
    
    CPU AFFINITY
    ------------
    RANK:0 CPU_SET: 0
    RANK:1 CPU_SET: 8

  - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=core
    
    
    CPU AFFINITY
    ------------
    RANK:0 CPU_SET: 0
    RANK:1 CPU_SET: 1

  - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=scatter MV2_CPU_BINDING_LEVEL=socket
    
    
    CPU AFFINITY
    ------------
    RANK:0 CPU_SET: 0 1 2 3 4 5 6 7
    RANK:1 CPU_SET: 8 9 10 11 12 13 14 15

  - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=bunch MV2_CPU_BINDING_LEVEL=socket
    
    
    CPU AFFINITY
    ------------
    RANK:0 CPU_SET: 0 1 2 3 4 5 6 7
    RANK:1 CPU_SET: 0 1 2 3 4 5 6 7
    
    -------------------------------------
Performance Troubleshooting

- Check “active_speed” in “ibv_devinfo –v” output
- Check OFED memory registration limits (log_num_mtt, log_mtt_per_seg)
- Increase registration cache size
  - MV2_NDREG_ENTRIES, MV2_NDREG_ENTRIES_MAX
- Are huge pages configured?
- SMP copy block size: MV2_SMP_SEND_BUF_SIZE
- Small message performance
  - RDMA fast path thresholds
    - MV2_NUM_RDMA_BUFFER, MV2_RDMA_FAST_PATH_BUF_SIZE
  - Eager thresholds
    - MV2_IBA_EAGER_THRESHOLD, MV2_VBUF_TOTAL_SIZE
- Large message performance
  - RNDV protocols: MV2_RNDV_PROTOCOL
- Collectives
  - Try different algorithms, change algorithm specific parameters
  - More in later talks
Getting Help

• Check the MVAPICH2 FAQ

• Check the Mailing List Archives

• Basic System Diagnostics
  – `ibv_devinfo` - at least one port should be `PORT_ACTIVE`
  – `ulimit -l` - should be “unlimited” on all compute nodes
  – host resolution: DNS or `/etc/hosts`
  – password-less ssh login
  – run IB perf tests for all the message sizes (-a option)
    • `ib_send_lat`, `ib_send_bw`
  – run system program (like `hostname`) and MPI hello world program
Getting Help (Cont.)

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

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

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

- Installation Guidelines, Runtime Optimization and Tuning Flexibility in
  - MVAPICH2
  - MVAPICH2-X
  - MVAPICH2-GDR
  - MVAPICH2-MIC
  - MVAPICH2-Virt
  - OMB

- Preview of Upcoming Releases

- Overview of MPIT, Configuration and Debugging Support
  - Advanced Optimization and Tuning of MPI Applications using MPI-T
  - User Resources
  - Frequently reported issues and Common mistakes
  - Useful Diagnostics
  - Performance Troubleshooting
  - Getting help and Bug report detail

- Conclusions and Final Q&A
MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 500K-1M cores
  - Dynamically Connected Transport (DCT) service with Connect-IB
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF ...)
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features
  - User Mode Memory Registration (UMR)
  - On-demand Paging
- Enhanced Inter-node and Intra-node communication schemes for upcoming OmniPath enabled Knights Landing architectures
- Extended RMA support (as in MPI 3.0)
- Extended topology-aware collectives
- Power-aware point-to-point (one-sided and two-sided) and collectives
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended Checkpoint-Restart and migration support with SCR
Concluding Remarks

- Provided an overview of the different MVAPICH2 software libraries
- Presented in-depth details on configuration and runtime parameters, optimizations and their impacts
- Provided an overview of debugging support
- Demonstrated how MPI and PGAS users can use these optimization techniques to extract performance and scalability while using various MVAPICH2 software libraries
Funding Acknowledgments

Funding Support by

[Logos of different organizations]

Equipment Support by

[Logos of different organizations]
Personnel Acknowledgments

**Current Students**
- A. Augustine (M.S.)
- A. Awan (Ph.D.)
- A. Bhat (M.S.)
- S. Chakraborty (Ph.D.)
- C.-H. Chu (Ph.D.)
- N. Islam (Ph.D.)
- K. Kulkarni (M.S.)
- M. Li (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

**Past Students**
- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharpapp (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandallia (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrarekar (Ph.D.)

**Past Post-Docs**
- H. Wang
- X. Besseron
- H.-W. Jin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne

**Current Senior Research Associates**
- K. Hamidouche
- X. Lu

**Current Post-Doc**
- J. Lin
- D. Shankar

**Current Programmer**
- J. Perkins

**Current Research Specialist**
- M. Arnold

**Past Research Scientist**
- S. Sur

**Past Programmers**
- D. Bureddy
Web Pointers

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

MVAPICH Web Page
http://mvapich.cse.ohio-state.edu

panda@cse.ohio-state.edu