How to Boost the Performance of Your HPC/AI Applications with MVAPICH2 Libraries?

A Tutorial at MUG’20

by

The MVAPICH Team
The Ohio State University

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

Latest version of the slides available at
http://cse.osu.edu/~subramon/mug20-mvapich2-tutorial.pdf
Overview of the MVAPICH2 Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), and AWS EFA
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD (upcoming))
- Started in 2001, first open-source version demonstrated at SC ‘02
- Supports the latest MPI-3.1 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA GPGPUs, since 2014
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- Tools:
  - OSU MPI Micro-Benchmarks (OMB), since 2003
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- Used by more than 3,100 organizations in 89 countries
- More than 820,000 (> 0.8 million) downloads from the OSU site directly
- Empowering many TOP500 clusters (June ‘20 ranking)
  - 4th, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 8th, 448,448 cores (Frontera) at TACC
  - 12th, 391,680 cores (ABCi) in Japan
  - 18th, 570,020 cores (Nurion) in South Korea and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 8th ranked TACC Frontera system
- Empowering Top500 systems for more than 15 years
Architecture of MVAPICH2 Software Family (for HPC and DL)

High Performance Parallel Programming Models

- **Message Passing Interface (MPI)**
- **PGAS (UPC, OpenSHMEM, CAF, UPC++)**
- **Hybrid --- MPI + X (MPI + PGAS + OpenMP/Cilk)**

High Performance and Scalable Communication Runtime

**Diverse APIs and Mechanisms**

- Point-to-point Primitives
- Collectives Algorithms
- Job Startup
- Energy-Awareness
- Remote Memory Access
- I/O and File Systems
- Fault Tolerance
- Virtualization
- Active Messages
- Introspection & Analysis

**Support for Modern Networking Technology**
(InfiniBand, iWARP, RoCE, Omni-Path)

- **Transport Protocols**
  - RC
  - XRC
  - UD
  - DC
- **Modern Features**
  - SHARP2*
  - ODP
  - SR-IOV
  - Multi Rail

**Support for Modern Multi-/Many-core Architectures**
(Intel-Xeon, OpenPower, Xeon-Phi, ARM, NVIDIA GPGPU)

- **Transport Mechanisms**
  - Shared Memory
  - CMA
  - IVSHMEM
  - XPMEM
- **Modern Features**
  - MCDRAM*
  - NVLink
  - CAPI*

* Upcoming
Production Quality Software Design, Development and Release

• Rigorous Q&A procedure before making a release
  – Exhaustive unit testing
  – Various test procedures on diverse range of platforms and interconnects
  – Test 19 different benchmarks and applications including, but not limited to
    • OMB, IMB, MPICH Test Suite, Intel Test Suite, NAS, Scalapak, and SPEC
  – Spend about 18,000 core hours per commit
  – Performance regression and tuning
  – Applications-based evaluation
  – Evaluation on large-scale systems

• All versions (alpha, beta, RC1 and RC2) go through the above testing
Automated Procedure for Testing Functionality

- Test OMB, IMB, MPICH Test Suite, Intel Test Suite, NAS, Scalapak, and SPEC
- Tests done for each build done build “buildbot”
- Test done for various different combinations of environment variables meant to trigger different communication paths in MVAPICH2
Scripts to Determine Performance Regression

- Automated method to identify performance regression between different commits
- Tests different MPI primitives
  - Point-to-point; Collectives; RMA
- Works with different
  - Job Launchers/Schedulers
    - SLURM, PBS/Torque, JSM
  - Works with different interconnects
- Works on multiple HPC systems
- Works on CPU-based and GPU-based systems
Designing (MPI+X) for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)

- Scalable Collective communication
  - Offloaded
  - Non-blocking
  - Topology-aware

- Balancing intra-node and inter-node communication for next generation multi-/many-core (128-1024 cores/node)
  - Multiple end-points per node

- Support for efficient multi-threading

- Integrated Support for GPGPUs and Accelerators

- Fault-tolerance/resiliency

- QoS support for communication and I/O

- Support for Hybrid MPI+PGAS programming
  - MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, MPI + UPC++...

- Virtualization

- Energy-Awareness
# MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)</strong></td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>Optimized Support for Microsoft Azure Platform with InfiniBand</td>
<td>MVAPICH2-Azure</td>
</tr>
<tr>
<td>Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis),</td>
<td>MVAPICH2-X</td>
</tr>
<tr>
<td>Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric Adapter (EFA)</td>
<td>MVAPICH2-X-AWS</td>
</tr>
<tr>
<td>Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications</td>
<td>MVAPICH2-GDR</td>
</tr>
<tr>
<td>Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)</td>
<td>MVAPICH2-EA</td>
</tr>
<tr>
<td>MPI Energy Monitoring Tool</td>
<td>OEMT</td>
</tr>
<tr>
<td>InfiniBand Network Analysis and Monitoring</td>
<td>OSU INAM</td>
</tr>
<tr>
<td>Microbenchmarks for Measuring MPI and PGAS Performance</td>
<td>OMB</td>
</tr>
</tbody>
</table>
Overview of MVAPICH2 Features

- Job start-up
- Transport Type Selection
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
Towards High Performance and Scalable Startup at Exascale

- Near-constant MPI and OpenSHMEM initialization time at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM respectively at 16,384 processes
- Memory consumption reduced for remote endpoint information by $O(\text{processes per node})$
- 1GB Memory saved per node with 1M processes and 16 processes per node

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


• MPI_Init takes 3.9 seconds on 57,344 processes on 1,024 nodes
• All numbers reported with 56 processes per node

New designs available since MVAPICH2-2.3.2
How to Get the Best Startup Performance with MVAPICH2?

• **MV2_HOMOGENEOUS_CLUSTER=1**
  //Set for homogenous clusters
• **MV2_ON_DEMAND_UD_INFO_EXCHANGE=1**
  //Enable UD based address exchange

Using SLURM as launcher

• Use PMI2
  - ./configure --with-pm=slurm --with-pmi=pmi2
  - srun --mpi=pmi2 ./a.out
• Use PMI Extensions
  - Patch for SLURM available at [http://mvapich.cse.ohio-state.edu/download/](http://mvapich.cse.ohio-state.edu/download/)
  - Patches available for SLURM 15, 16, and 17
  - PMI Extensions are automatically detected by MVAPICH2

Using mpirun_rsh as launcher

• **MV2_MT_DEGREE**
  - degree of the hierarchical tree used by mpirun_rsh
• **MV2_FASTSSH_THRESHOLD**
  - #nodes beyond which hierarchical-ssh scheme is used
• **MV2_NPROCS_THRESHOLD**
  - #nodes beyond which file-based communication is used for hierarchical-ssh during start up
Transport Protocol Selection 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

![Performance with HPCC Random Ring](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Significance</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_USE_UD_HYBRID</td>
<td>• Enable / Disable use of UD transport in Hybrid mode</td>
<td>Enabled</td>
<td>• Always Enable</td>
</tr>
<tr>
<td>MV2_HYBRID_ENABLE_THRESHOLD_SIZE</td>
<td>• Job size in number of processes beyond which hybrid mode will be enabled</td>
<td>1024</td>
<td>• Uses RC/XRC connection until job size &lt; threshold</td>
</tr>
<tr>
<td>MV2_HYBRID_MAX_RC_CONN</td>
<td>• Maximum number of RC or XRC connections created per process</td>
<td>64</td>
<td>• Prevents HCA QP cache thrashing</td>
</tr>
<tr>
<td></td>
<td>• Limits the amount of connection memory</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Refer to Running with Hybrid UD-RC/XRC section of MVAPICH2 user guide for more information
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3a-userguide.html#x1-690006.11](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3a-userguide.html#x1-690006.11)
Process Mapping support in MVAPICH2

(available since v1.4)

Preset Binding Policies

- bunch (Default)
- scatter
- hybrid

User-defined binding

MPI rank-to-core binding

MVAPICH2 detects processor architecture at job-launch
Preset Process-binding Policies – Bunch

- "Core" level "Bunch" mapping (Default)
  - MV2_CPU_BINDING_POLICY=bunch

- "Socket/Numanode" level "Bunch" mapping
  - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=bunch
Preset Process-binding Policies – Scatter

- "Core" level "Scatter" mapping
  - MV2_CPU_BINDING_POLICY=scatter

- "Socket/Numanode" level "Scatter" mapping
  - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=scatter
Process and thread binding policies in hybrid MPI+Threads

• A new process binding policy – “hybrid”
  – MV2_CPU_BINDING_POLICY = hybrid

• A new environment variable for co-locating Threads with MPI Processes
  – MV2_THREADS_PER_PROCESS = k
    – Automatically set to OMP_NUM_THREADS if OpenMP is being used
    – Provides a hint to the MPI runtime to spare resources for application threads.

• New variable for threads bindings with respect to parent process and architecture
  – MV2_HYBRID_BINDING_POLICY = {bunch|scatter|linear|compact|spread|numa}
    • Linear – binds MPI ranks and OpenMP threads sequentially (one after the other)
      – Recommended to be used on non-hyper threaded systems with MPI+OpenMP
    • Compact – binds MPI rank to physical-core and locates respective OpenMP threads on hardware threads
      – Recommended to be used on multi-/many-cores e.g., KNL, POWER8, and hyper-threaded Xeon, etc.
### Binding Example in Hybrid (MPI+Threads)

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

**Diagram:**

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

---

<table>
<thead>
<tr>
<th>Core0</th>
<th>HWT</th>
<th>Core1</th>
<th>HWT</th>
<th>Core2</th>
<th>HWT</th>
<th>Core3</th>
<th>HWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWT</td>
<td></td>
<td>HWT</td>
<td></td>
<td>HWT</td>
<td></td>
<td>HWT</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- Rank0
- Rank1
- Rank2
- Rank3
Binding Example in Hybrid (MPI+Threads) ---- Cont’d

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

- MPI Rank-0 with its 4-OpenMP threads gets bound on Core-0 through Core-3, and so on...
Binding Example in Hybrid (MPI+Threads) ---- Cont’d

- MPI Processes = 16
- Example: AMD EPYC 7551 processor with 8 NUMA domains
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_HYBRID_BINDING_POLICY = numa
User-Defined Process Mapping

- User has complete-control over process-mapping
- To run 4 processes on cores 0, 1, 4, 5:
  - $ mpirun_rsh -np 4 -hostfile hosts MV2_CPU_MAPPING=0:1:4:5 ./a.out
- Use ‘,’ or ‘-’ to bind to a set of cores:
  - $ mpirun_rsh -np 64 -hostfile hosts MV2_CPU_MAPPING=0,2-4:1:5:6 ./a.out
- Is process binding working as expected?
  - MV2_SHOW_CPU_BINDING=1
    - Display CPU binding information
    - Launcher independent
    - Example
      - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=scatter
        --------------CPU AFFINITY--------------
        RANK:0 CPU_SET: 0
        RANK:1 CPU_SET: 8

- Refer to Running with Efficient CPU (Core) Mapping section of MVAPICH2 user guide for more information
- http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3rc1-userguide.html#x1-600006.5
Collective Communication in MVAPICH2

Run-time flags:
- All shared-memory based collectives: MV2_USE_SHMEM_COLL (Default: ON)
- Hardware Mcast-based collectives: MV2_USE_MCAST (Default: OFF)
- CMA and XPMEM-based collectives are in MVAPICH2-X

Designed for Performance & Overlap
MCAST-based designs improve latency of MPI_Bcast by up to 85%.

Use MV2_USE_MCAST=1 to enable MCAST-based designs.
MPI_Scatter - Benefits of using Hardware-Mcast

- Enabling MCAST-based designs for MPI_Scatter improves small message up to 75%

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</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>
Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)

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

*More details in the tutorial "SHARPv2: In-Network Scalable Streaming Hierarchical Aggregation and Reduction Protocol" by Devendar Bureddy (NVIDIA/Mellanox)

* Bloch et al. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction
# Benefits of SHARP Allreduce at Application Level

## Avg DDOT Allreduce time of HPCG

![Chart showing Avg DDOT Allreduce time of HPCG with latency in seconds for different node and PPN combinations, comparing MVAPICH2 and MVAPICH2-SHArP, with 12% reduction in latency at (16,28) configuration.]

### SHARP support available since MVAPICH2 2.3a

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV2_ENABLE_SHARP=1</td>
<td>Enables SHARP-based collectives</td>
<td>Disabled</td>
</tr>
<tr>
<td>--enable-sharp</td>
<td>Configure flag to enable SHARP</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

- Refer to Running Collectives with Hardware based SHARP support section of MVAPICH2 user guide for more information
- [http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3-userguide.html#x1-990006.26](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3-userguide.html#x1-990006.26)

More details in the talk "Impact of SHARP and Adaptive Routing on Applications on Frontera" by John Cazes (TACC) on Wednesday (08/26/2020) from 12:00 PM - 12:30 PM EDT
# MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>Optimized Support for Microsoft Azure Platform with InfiniBand</td>
<td>MVAPICH2-Azure</td>
</tr>
<tr>
<td><strong>Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM),</strong></td>
<td>MVAPICH2-X</td>
</tr>
<tr>
<td><strong>OSU INAM (InfiniBand Network Monitoring and Analysis),</strong></td>
<td>MVAPICH2-X-AWS</td>
</tr>
<tr>
<td>Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric</td>
<td>MVAPICH2-X-AWS</td>
</tr>
<tr>
<td>Adapter (EFA)</td>
<td></td>
</tr>
<tr>
<td>Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning</td>
<td>MVAPICH2-GDR</td>
</tr>
<tr>
<td>Applications</td>
<td></td>
</tr>
<tr>
<td>Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and,</td>
<td>MVAPICH2-EA</td>
</tr>
<tr>
<td>RoCE (v1/v2)</td>
<td></td>
</tr>
<tr>
<td>MPI Energy Monitoring Tool</td>
<td>OEMT</td>
</tr>
<tr>
<td>InfiniBand Network Analysis and Monitoring</td>
<td>OSU INAM</td>
</tr>
<tr>
<td>Microbenchmarks for Measuring MPI and PGAS Performance</td>
<td>OMB</td>
</tr>
</tbody>
</table>
MVAPICH2-X for MPI and Hybrid MPI + PGAS Applications

<table>
<thead>
<tr>
<th>High Performance Parallel Programming Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI Message Passing Interface</td>
</tr>
<tr>
<td>PGAS (UPC, OpenSHMEM, CAF, UPC++)</td>
</tr>
<tr>
<td>Hybrid --- MPI + X (MPI + PGAS + OpenMP/Cilk)</td>
</tr>
</tbody>
</table>

High Performance and Scalable Unified Communication Runtime

- Optimized Point-to-point Primitives
- Remote Memory Access
- Active Messages
- Collectives Algorithms (Blocking and Non-Blocking)
- Scalable Job Startup
- Fault Tolerance
- Introspection & Analysis with OSU INAM

- Support for Modern Networking Technologies (InfiniBand, iWARP, RoCE, Omni-Path...)
- Support for Modern Multi-/Many-core Architectures (Intel-Xeon, OpenPower...)

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

- Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF
  - Available with since 2012 (starting with MVAPICH2-X 1.9)
  - [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)
MVAPICH2-X 2.3

- Released on 06/08/2020
- Major Features and Enhancements
  - **MPI Features**
    - Based on MVAPICH2 2.3.4
      - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
    - Enhanced point-to-point and collective tunings for AMD EPYC, Catalyst@EPCC, Mayer@Sandia, Auzre@Microsoft, AWS, and Frontera@TACC
  - **MPI (Advanced) Features**
    - Optimized support for large message MPI_Allreduce and MPI_Reduce
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Improved performance for communication using DC transport
      - OFA-IB-CH3 interface
    - Enhanced support for AWS EFA adapter and SRD transport protocol
      - OFA-IB-CH3 interface
    - Enhanced point-to-point and collective tuning for AWS EFA adapter and SRD transport protocol
      - OFA-IB-CH3 interface
    - Add multiple MPI_T PVARs and CVARs for point-to-point and collective operations
  - Tuning for MPI collective operations for Intel Broadwell, Intel CascadeLake, Azure HB (AMD EPYC), and Azure HC (Intel Skylake) systems
    - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interface
  - **Support for OSU InfiniBand Network Analysis and Management (OSU INAM) Tool v0.9.6**
  - **Unified Runtime Features**
    - Based on MVAPICH2 2.3.4 (OFA-IB-CH3 interface). All the runtime features enabled by default in OFA-IB-CH3 and OFA-IB-RoCE interface of MVAPICH2 2.3.4 are available in MVAPICH2-X 2.3
## MVAPICH2-X Feature Table

<table>
<thead>
<tr>
<th>Features for InfiniBand (OFA-IB-CH3) and RoCE (OFA-RoCE-CH3)</th>
<th>Basic</th>
<th>Basic-XPMEM</th>
<th>Intermediate</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture Specific Point-to-point and Collective Optimizations for x86, OpenPOWER, and ARM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Optimized Support for PGAS models (UPC, UPC++, OpenSHMEM, CAF) and Hybrid MPI+PGAS models</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CMA-Aware Collectives</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Optimized Asynchronous Progress*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>InfiniBand Hardware Multicast-based MPI_Bcast**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OSU InfiniBand Network Analysis and Monitoring (INAM)**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPMEM-based Point-to-Point and Collectives</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Direct Connected (DC) Transport Protocol**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>User mode Memory Registration (UMR)**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>On Demand Paging (ODP)**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Core-direct based Collective Offload*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SHARP-based Collective Offload**</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- * indicates disabled by default at runtime. Must use appropriate environment variable in MVAPICH2-X user guide to enable it.
- ** indicates features only tested with InfiniBand network

---

Network Based Computing Laboratory

MUG'20
Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards

- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards

- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards

- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards

- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
Impact of DC Transport Protocol on Neuron

- Up to 76% benefits over MVAPICH2 for Neuron using Direct Connected transport protocol at scale
  - VERSION 7.6.2 master (f5a1284) 2018-08-15
- Numbers taken on bbpv2.epfl.ch
  - Knights Landing nodes with 64 ppn
  - ./x86_64/special-mpi-c stop_time=2000 -c is_split=1 parinit.hoc
  - Used “runtime” reported by execution to measure performance
- Environment variables used
  - MV2_USE_DC=1
  - MV2_NUM_DC_TGT=64
  - MV2_SMALL_MSG_DC_POOL=96
  - MV2_LARGE_MSG_DC_POOL=96
  - MV2_USE_RDMA_CM=0

Available from MVAPICH2-X 2.3rc2 onwards

More details in talk "Building Brain Circuits: Experiences with shuffling terabytes of data over MPI" by Matthias Wolf, Blue Brain Project, EPFL, Switzerland on Wednesday (08/26/2020) from 1:30 PM - 2:00 PM EDT.
Optimized CMA-based Collectives for Large Messages

- Significant improvement over existing implementation for Scatter/Gather with 1MB messages (up to 4x on KNL, 2x on Broadwell, 14x on OpenPower)
- New two-level algorithms for better scalability
- Improved performance for other collectives (Bcast, Allgather, and Alltoall)


Available since MVAPICH2-X 2.3b
Benefits of the New Asynchronous Progress Design: Broadwell + InfiniBand

Up to 33% performance improvement in P3DFFT application with 448 processes
Up to 29% performance improvement in HPL application with 896 processes


Available since MVAPICH2-X 2.3rc1
Shared Address Space (XPMEM)-based Collectives Design

- "Shared Address Space"-based true **zero-copy** Reduction collective designs in MVAPICH2
- Offloaded computation/communication to peers ranks in reduction collective operation
- Up to **4X** improvement for 4MB Reduce and up to **1.8X** improvement for 4M AllReduce


Available since MVAPICH2-X 2.3rc1
Performance of Non-Reduction Collectives with XPMEM

- **28 MPI Processes** on single dual-socket Broadwell E5-2680v4, 2x14 core processor
- Used osu_bcast from OSU Microbenchmarks v5.5
Application Level Benefits of XPMEM-based Designs

CNTK AlexNet Training
(B.S=default, iteration=50, ppn=28)

MiniAMR (dual-socket, ppn=16)

- Intel XeonCPU E5-2687W v3 @ 3.10GHz (10-core, 2-socket)
- Up to 20% benefits over IMPI for CNTK DNN training using AllReduce
- Up to 27% benefits over IMPI and up to 15% improvement over MVAPICH2 for MiniAMR application kernel
## MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>Optimized Support for Microsoft Azure Platform with InfiniBand</td>
<td>MVAPICH2-Azure</td>
</tr>
<tr>
<td>Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM),</td>
<td>MVAPICH2-X</td>
</tr>
<tr>
<td>OSU INAM (InfiniBand Network Monitoring and Analysis),</td>
<td></td>
</tr>
<tr>
<td>Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric</td>
<td>MVAPICH2-X-AWS</td>
</tr>
<tr>
<td>Adapter (EFA)</td>
<td></td>
</tr>
<tr>
<td>Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning</td>
<td>MVAPICH2-GDR</td>
</tr>
<tr>
<td>Applications</td>
<td></td>
</tr>
<tr>
<td>Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and,</td>
<td>MVAPICH2-EA</td>
</tr>
<tr>
<td>RoCE (v1/v2)</td>
<td></td>
</tr>
<tr>
<td>MPI Energy Monitoring Tool</td>
<td>OEMT</td>
</tr>
<tr>
<td>InfiniBand Network Analysis and Monitoring</td>
<td>OSU INAM</td>
</tr>
<tr>
<td>Microbenchmarks for Measuring MPI and PGAS Performance</td>
<td>OMB</td>
</tr>
</tbody>
</table>
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

\[\text{At Sender:}\]

\[
\text{for } (j = 0; j < \text{pipeline}_\text{len}; j++) \\
\quad \text{cudaMemcpyAsync(s_hostbuf + j * blk, s_devbuf + j * blksz, \ldots);} \\
\text{for } (j = 0; j < \text{pipeline}_\text{len}; j++) \{} \\
\quad \text{while } (\text{result } \neq \text{cudaSuccess}) \{} \\
\quad \quad \text{result } = \text{cudaStreamQuery}(\ldots); \\
\quad \quad \text{if}(j > 0) \text{MPI_Test}(\ldots); \\
\quad \} \\
\text{MPI_Isend(s_hostbuf + j * block_sz, blksz \ldots);} \\
\} \\
\text{MPI_Waitall();}
\]

<<Similar at receiver>>

Low Productivity and High Performance
GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GPU

- 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: MVAPICHH2-GDR 1.8-2.3.4 Releases

- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers
- Unified memory
• MVAPICH2-GDR 2.3.4 requires the following software to be installed on your system:
  1. Mellanox OFED 3.2 and later
  2. NVIDIA Driver 367.48 or later
  3. NVIDIA CUDA Toolkit 7.5 and later
  4. NVIDIA Peer Memory (nv_peer_mem) module to enable GPUDirect RDMA (GDR) support

• Strongly Recommended for Best Performance
  5. GDRCOPY Library by NVIDIA: https://github.com/NVIDIA/gdrcopy

• Comprehensive Instructions can be seen from the MVAPICH2-GDR User Guide:
  – http://mvapich.cse.ohio-state.edu/userguide/gdr/
Simple Installation steps for both systems

Pick the right MVAPICH2-GDR RPM from Downloads page:

- [http://mvapich.cse.ohio-state.edu/downloads/](http://mvapich.cse.ohio-state.edu/downloads/)
- e.g. [http://mvapich.cse.ohio-state.edu/download/mvapich/gdr/2.3/mofed4.5/mvapich2-gdr-mcast.cuda10.0.mofed4.5.gnu4.8.5-2.3-1.el7.x86_64.rpm](http://mvapich.cse.ohio-state.edu/download/mvapich/gdr/2.3/mofed4.5/mvapich2-gdr-mcast.cuda10.0.mofed4.5.gnu4.8.5-2.3-1.el7.x86_64.rpm) (== <mv2-gdr-rpm-name>.rpm)

\$ wget http://mvapich.cse.ohio-state.edu/download/mvapich/gdr/2.3/<mv2-gdr-rpm-name>.rpm

**Root Users:**

\$ rpm -Uvh --nodeps <mv2-gdr-rpm-name>.rpm

**Non-Root Users:**

\$ rpm2cpio <mv2-gdr-rpm-name>.rpm | cpio – id

Contact MVAPICH help list with any questions related to the package

mvapich-help@cse.ohio-state.edu
ROCE and Optimized Collectives Support

- RoCE V1 and V2 support
- RDMA_CM connection support
- CUDA-Aware Collective Tuning
  - Point-point Tuning (available since MVAPICH2-GDR 2.0)
    - Tuned thresholds for the different communication patterns and features
    - Depending on the system configuration (CPU, HCA and GPU models)
  - Tuning Framework for GPU based collectives
    - Select the best algorithm depending on message size, system size and system configuration
    - Support for Bcast and Gather operations for different GDR-enabled systems
- Available since MVAPICH2-GDR 2.2RC1 release
MVAPICH2-GDR 2.3.4

- Released on 06/04/2020
- Major Features and Enhancements
  - Based on MVAPICH2 2.3.4
  - Enhanced MPI_Allreduce performance on DGX-2 systems
  - Enhanced MPI_Allreduce performance on POWER9 systems
  - Reduced the CUDA interception overhead for non-CUDA symbols
  - Enhanced performance for point-to-point and collective operations on Frontera’s RTX nodes
  - Add new runtime variable 'MV2_SUPPORT_DL' to replace 'MV2_SUPPORT_TENSOR_FLOW'
  - Added compilation and runtime methods for checking CUDA support
  - Enhanced GDR output for runtime variable MV2_SHOW_ENV_INFO
  - Tested with Horovod and common DL Frameworks (TensorFlow, PyTorch, and MXNet)
  - Tested with PyTorch Distributed
  - Support for CUDA 10.1
  - Support for PGI 20.x
  - Enhanced GPU communication support in MPI_THREAD_MULTIPLE mode
  - Enhanced performance of datatype support for GPU-resident data
    - Zero-copy transfer when P2P access is available between GPUs through NVLink/PCIe
  - Enhanced GPU-based point-to-point and collective tuning
    - OpenPOWER systems such as ORNL Summit and LLNL Sierra ABCI system @AIST, Owens and Pitzer systems @Ohio Supercomputer Center
  - Scaled Allreduce to 24,576 Volta GPUs on Summit
## Tuning GDRCOPY 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_GDRCOPY</td>
<td>• Enable / Disable GDRCOPY-based designs</td>
<td>1 (Enabled)</td>
<td>• Always enable</td>
</tr>
<tr>
<td>MV2_GDRCOPY_LIMIT</td>
<td>• Controls messages size until which GDRCOPY is used</td>
<td>8 KByte</td>
<td>• Tune for your system&lt;br&gt;• GPU type, host architecture. Impacts the eager performance</td>
</tr>
<tr>
<td>MV2_GPUDIRECT_GDRCOPY_LIB</td>
<td>• Path to the GDRCOPY library</td>
<td>Unset</td>
<td>• Always set</td>
</tr>
<tr>
<td>MV2_USE_GPUDIRECT_D2H_GDRCOPY_LIMIT</td>
<td>• Controls messages size until which GDRCOPY is used at sender</td>
<td>16Bytes</td>
<td>• Tune for your systems&lt;br&gt;• CPU and GPU type</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 Loopback 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_LOOPBACK</td>
<td>• Enable / Disable LOOPBACK-based designs</td>
<td>1 (Enabled)</td>
<td>• Always enable</td>
</tr>
<tr>
<td>MV2_GPUDIRECT_LOOPBACK_LIMIT</td>
<td>• Controls messages size until which LOOPBACK 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 HCA. Impacts the eager performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Sensitive to the P2P issue</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CUDA version: impact pipelining overheads and P2P bandwidth bottlenecks</td>
</tr>
<tr>
<td>MV2_USE_GPUDIRECT_RECEIVE_LIMIT</td>
<td>• Controls messages size until which 1 hop design is used (GDR Write at the receiver)</td>
<td>256 KBytes</td>
<td>• Tune for your system</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• GPU type, HCA type and configuration</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)
Application-Level Evaluation (HOOMD-blue)

64K Particles

- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomDBlue Version 1.0.5
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

256K Particles

Network Based Computing Laboratory
MPI Datatype support in MVAPICH2

• Datatypes support in MPI
  – Operate on customized datatypes to improve productivity
  – Enable MPI library to optimize non-contiguous data

```
At Sender:

MPI_Type_vector (n_blocks, n_elements, stride, old_type, &new_type);
MPI_Type_commit(&new_type);
...
MPI_Send(s_buf, size, new_type, dest, tag, MPI_COMM_WORLD);
```

• Inside MVAPICH2
  - Use datatype specific CUDA Kernels to pack data in chunks
  - Efficiently move data between nodes using RDMA
  - In progress - currently optimizes vector and hindexed datatypes
  - Transparent to the user

**MPI Datatype Processing (Computation Optimization)**

- **Comprehensive support**
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes
- **Separate non-blocking stream for kernels launched by MPI library**
  - Avoids stream conflicts with application kernels
- **Flexible set of parameters for users to tune kernels**
  - **Vector**
    - MV2_CUDA_KERNEL_VECTOR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_VECTOR_YSIZE
  - **Subarray**
    - MV2_CUDA_KERNEL_SUBARR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_SUBARR_XDIM
    - MV2_CUDA_KERNEL_SUBARR_YDIM
    - MV2_CUDA_KERNEL_SUBARR_ZDIM
  - **Indexed_block**
    - MV2_CUDA_KERNEL_IDXBLK_XDIM
Performance of Stencil3D (3D subarray)

Stencil3D communication kernel on 2 GPUs with various X, Y, Z dimensions using MPI_Isend/Irecv

- DT: Direct Transfer, TR: Targeted Kernel
- Optimized design gains up to 15%, 15% and 22% compared to TR, and more than 86% compared to DT on X, Y and Z respectively
Common Scenario

MPI_Isend (A,... Datatype,...)
MPI_Isend (B,... Datatype,...)
MPI_Isend (C,... Datatype,...)
MPI_Isend (D,... Datatype,...)
...
MPI_Waitall (...);

*A, B...contain non-contiguous MPI Datatype*
Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland

Wilkes GPU Cluster

- Default
- Callback-based
- Event-based

CSCS GPU cluster

- Default
- Callback-based
- Event-based

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

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


Cosmo model: http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/
**MVAPICH2-GDR: Enhanced Derived Datatype**

- Kernel-based and GDRCOPY-based one-shot packing for inter-socket and inter-node communication
- Zero-copy (packing-free) for GPUs with peer-to-peer direct access over PCIe/NVLink

**Graphs:***
- **GPU-based DDTBench mimics MILC communication kernel**
  - Speedup of MILC for different problem sizes.
  - Problem sizes are represented as [6, 8,8,8,8], [6, 8,8,8,16], [6, 8,8,16,16], [6, 16,16,16,16].
  - Speedup values are shown for OpenMPI 4.0.0, MVAPICH2-GDR 2.3.1, and MVAPICH2-GDR-Next.

- **Communication Kernel of COSMO Model**
  - Improved 3.4X
  - Improved 15X
  - Platform: Nvidia DGX-2 system (NVIDIA Volta GPUs connected with NVSwitch), CUDA 9.2
  - Platform: Cray CS-Storm (16 NVIDIA Tesla K80 GPUs per node), CUDA 8.0
  - Execution Time (s) for different number of GPUs: 16, 32, 64.

**Charts:**
- Speedup graph showing performance with different GPU configurations.
- Execution time graph showing comparison between OpenMPI 4.0.0, MVAPICH2-GDR 2.3.1, and MVAPICH2-GDR-Next on different GPU counts.

**Platforms:**
- Nvidia DGX-2 system
- Cray CS-Storm
Deep Learning: New Challenges for Runtimes

- **Scale-up**: Intra-node Communication
  - Many improvements like:
    - NVIDIA cuDNN, cuBLAS, NCCL, etc.
    - CUDA 9 Co-operative Groups

- **Scale-out**: Inter-node Communication
  - DL Frameworks – most are optimized for single-node only
  - Distributed (Parallel) Training is an emerging trend
    - **OSU-Caffe** – MPI-based
    - Microsoft CNTK – MPI/NCCL2
    - Google TensorFlow – gRPC-based/MPI/NCCL2
    - Facebook Caffe2 – Hybrid (NCCL2/Gloo/MPI)
    - PyTorch
MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training

ML/DL Applications

TensorFlow

PyTorch

MXNet

Horovod

MVAPICH2-X for CPU-Based Training

MVAPICH2-GDR for GPU-Based Training
Optimized designs in MVAPICH2-GDR offer better/comparable performance for most cases

MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 1 DGX-2 node (16 Volta GPUs)

Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 10.1

~2.5X better

~4.7X better

Optimized designs in MVAPICH2-GDR offer better performance for most cases.

MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) up to 1,536 GPUs

Platform: Dual-socket IBM POWER9 CPU, 6 NVIDIA Volta V100 GPUs, and 2-port InfiniBand EDR Interconnect

Distributed Training with TensorFlow and MVAPICH2-GDR

- ResNet-50 Training using TensorFlow benchmark on 1 DGX-2 node (16 Volta GPUs)

\[
\text{Scaling Efficiency} = \frac{\text{Actual throughput}}{\text{Ideal throughput at scale}} \times 100\%
\]

Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 9.2
Distributed Training with TensorFlow and MVAPICH2-GDR

- ResNet-50 Training using TensorFlow benchmark on SUMMIT -- 1536 Volta GPUs!
- 1,281,167 (1.2 mil.) images
- Time/epoch = 3.6 seconds
- Total Time (90 epochs) = 3.6 x 90 = 332 seconds = 5.5 minutes!

*We observed errors for NCCL2 beyond 96 GPUs

**ImageNet-1k has 1.2 million images**

**MVAPICH2-GDR reaching ~0.35 million images per second for ImageNet-1k!**

Platform: The Summit Supercomputer (#1 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 9.2
Distributed TensorFlow on TACC Frontera (2048 CPU nodes)

- Scaled TensorFlow to 2048 nodes on Frontera using MVAPICH2 and IntelMPI
- MVAPICH2 and IntelMPI give similar performance for DNN training
- Report a peak of 260,000 images/sec on 2048 nodes
- On 2048 nodes, ResNet-50 can be trained in 7 minutes!

Applications-Level Tuning: Compilation of Best Practices

- MPI runtime has many parameters
- Tuning a set of parameters can help you to extract higher performance
- Compiled a list of such contributions through the MVAPICH Website
  - [http://mvapich.cse.ohio-state.edu/best_practices/](http://mvapich.cse.ohio-state.edu/best_practices/)
- Initial list of applications
  - Amber
  - HoomDBlue
  - HPCG
  - Lulesh
  - MILC
  - Neuron
  - SMG2000
  - Cloverleaf
  - SPEC (LAMMPS, POP2, TERA_TF, WRF2)
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.
Amber: Impact of Tuning Eager Threshold

- Tuning the Eager threshold has a significant impact on application performance by avoiding the synchronization of rendezvous protocol and thus yielding better communication computation overlap
- 19% improvement in overall execution time at 256 processes
- Library Version: MVAPICH2 2.2
- MVAPICH Flags used
  - MV2_IBA_EAGER_THRESHOLD=131072
  - MV2_VBUF_TOTAL_SIZE=131072
- Input files used
  - Small: MDIN
  - Large: PMTOP

Data Submitted by: Dong Ju Choi @ UCSD
MiniAMR: Impact of Tuning Eager Threshold

- Tuning the Eager threshold has a significant impact on application performance by avoiding the synchronization of rendezvous protocol and thus yielding better communication computation overlap.

- 8% percent reduction in total communication time.

- Library Version: MVAPICH2 2.2.

- MVAPICH Flags used:
  - MV2_IBA_EAGER_THRESHOLD=32768
  - MV2_VBUF_TOTAL_SIZE=32768

Data Submitted by Karen Tomko @ OSC and Dong Ju Choi @ UCSD
SMG2000: Impact of Tuning Transport Protocol

- UD-based transport protocol selection benefits the SMG2000 application
- 22% and 6% on 1,024 and 4,096 cores, respectively
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - MV2_USE_ONLY_UD=1
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

Data Submitted by Jerome Vienne @ TACC
UD-based transport protocol selection benefits the SMG2000 application

- 15% and 27% improvement is seen for 768 and 1,024 processes respectively

- Library Version: MVAPICH2 2.2

- MVAPICH Flags used
  - MV2_USE_ONLY_UD=1

- Input File
  - YuEtAl2012

- System Details
  - Comet@SDSC
  - Haswell nodes with dual 12-cores socket per node and Mellanox FDR (56 Gbps) network.

Data Submitted by Mahidhar Tatineni @ SDSC
HPCG: Impact of Collective Tuning for MPI+OpenMP Programming Model

- Partial subscription nature of hybrid MPI+OpenMP programming requires a new level of collective tuning
  - For PPN=2 (Processes Per Node), the tuned version of MPI_Reduce shows 51% improvement on 2,048 cores
- 24% improvement on 512 cores
  - 8 OpenMP threads per MPI processes
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - The tuning parameters for hybrid MPI+OpenMP programming models is on by default from MVAPICH2-2.1 onward
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

Data Submitted by Jerome Vienne and Carlos Rosales-Fernandez @ TACC
HOOMD-blue: Impact of GPUDirect RDMA Based Tuning

- HOOMD-blue is a Molecular Dynamics simulation using a custom force field.
- GPUDirect specific features selection and tuning significantly benefit the HOOMD-blue application. We observe a factor of 2X improvement on 32 GPU nodes, with both 64K and 256K particles.
- Library Version: MVAPICH2-GDR 2.2
- MVAPICH-GDR Flags used
  - `MV2_USE_CUDA=1`
  - `MV2_USE_GPUDIRECT=1`
  - `MV2_GPUDIRECT_GDRCOPY=1`
- System Details
  - Wilkes@Cambridge
  - 128 Ivybridge nodes, each node is a dual 6-cores socket with Mellanox FDR

Data Submitted by Khaled Hamidouche @ OSU
Application Scalability on Skylake and KNL with Omni-Path

MiniFE (1300x1300x1300 ~ 910 GB)

NEURON (YuEtAl2012)

Cloverleaf (bm64) MPI+OpenMP, NUM_OMP_THREADS = 2

Runtime parameters: MV2_SMPI_LENGTH_QUEUE=524288 PSM2_MO_RNDV_SHM_THRESH=128K PSM2_MO_RNDV_HFI_THRESH=128K

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

Network Based Computing Laboratory
SPEC MPI 2007 Benchmarks: Broadwell + InfiniBand

MVAPICH2-X outperforms Intel MPI by up to 31%

Configuration: 448 processes on 16 Intel E5-2680v4 (Broadwell) nodes having 28 PPN and interconnected with 100Gbps Mellanox MT4115 EDR ConnectX-4 HCA
MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1-10M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF ...)
  - MPI + Task*
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features of Mellanox InfiniBand
  - Tag Matching*
  - Adapter Memory*
  - Bluefield based offload*
- Enhanced communication schemes for upcoming architectures
  - ROCm*
  - Intel Optane*
  - BlueField*
  - CAPI*
- Extended topology-aware collectives
- Extended Energy-aware designs and Virtualization Support
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended FT support
- Support for * features will be available in future MVAPICH2 Releases
For More Details on other MVAPICH2 Libraries/Features

- **MPI_T Support**
  - More details in the talk "Performance Engineering using MVAPICH and TAU" by Sameer Shende (Paratools/UO) on Tuesday (08/25/2020) from 3:30 PM - 4:00 PM EDT

- **MVAPICH2-Azure**
  - More details in the talk "MVAPICH2 on Microsoft Azure HPC" by Jithin Jose (Microsoft, Azure) on Tuesday (08/25/2020) from 1:30 PM - 2:00 PM EDT

- **MVAPICH2-AWS**
  - More details in the talk "Scaling Message Passing on Amazon Web Services with Elastic Fabric Adapter" by Raghunath Rajachandrasekar (AWS) on Tuesday (08/25/2020) from 1:00 PM - 1:30 PM EDT

- **Support for SHARP**
  - More details in the talk "Impact of SHARP and Adaptive Routing on Applications on Frontera" by John Cazes (TACC) on Wednesday (08/26/2020) from 12:00 PM - 12:30 PM EDT
Funding Acknowledgments

Funding Support by

[Logos of various sponsors]

Equipment Support by

[Logos of various sponsors]
Acknowledgments to all the Heroes (Past/Current Students and Staffs)

**Current Students (Graduate)**
- Q. Anthony (Ph.D.)
- M. Bayatpour (Ph.D.)
- C.-H. Chu (Ph.D.)
- A. Jain (Ph.D.)
- M. Kedia (M.S.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- N. S. Kumar (M.S.)
- B. Ramesh (Ph.D.)
- K. K. Suresh (Ph.D.)
- N. Sarkauskas (Ph.D.)
- S. Srivastava (M.S.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)

**Past Students**
- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kulkarni (M.S.)
- S. Krishna Mohanty (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- M. Li (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kulkarni (M.S.)
- S. Krishna Mohanty (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- M. Li (Ph.D.)

**Past Post-Docs**
- D. Banerjee
- X. Besseron
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potturi (Ph.D.)
- K. Raj (M.S.)
- S. Marcarelli
- A. Ruhela
- J. Vienne

**Current Research Scientists**
- A. Shafi
- H. Subramoni
- N. Sarkauskas (B.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)

**Current Senior Research Associate**
- J. Hashmi

**Current Research Specialist**
- A. Shafi
- J. Smith

**Current Software Engineer**
- J. Lin
- H. Wang

**Past Research Scientists**
- K. Hamidouche
- S. Sur
- X. Lu

**Past Programmers**
- D. Bureddy
- J. Perkins

**Past Research Specialist**
- M. Arnold

**Current Post-docs**
- M. S. Ghazimeersaeed
- K. Manian
Thank You!

panda@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/
Follow us on Twitter: @mvapich

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/