Overview of the MVAPICH Project: Latest Status and Future Roadmap

MVAPICH2 User Group (MUG) Meeting

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda
High-End Computing (HEC): PetaFlop to ExaFlop

100 PFlops in 2017

143 PFlops in 2018

1 EFlops in 2020-2021?

Expected to have an ExaFlop system in 2020-2021!
Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges

Application Kernels/Applications (HPC and DL)

Middleware

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

Communication Library or Runtime for Programming Models

- Point-to-point Communication
- Collective Communication
- Energy-Awareness
- Synchronization and Locks
- I/O and File Systems
- Fault Tolerance

Networking Technologies
(InfiniBand, 40/100/200GigE, Aries, and Omni-Path)

Multi-/Many-core Architectures

Accelerators (GPU and FPGA)

Co-Design Opportunities and Challenges across Various Layers

Performance
Scalability
Resilience

Network Based Computing Laboratory
MVAPICH User Group Meeting (MUG) 2019
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)
  - Scalable job start-up
  - Low memory footprint
- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
- Balancing intra-node and inter-node communication for next generation nodes (128-1024 cores)
  - Multiple end-points per node
- Support for efficient multi-threading
- Integrated Support for Accelerators (GPGPUs and FPGAs)
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, MPI+UPC++, CAF, ...)
- Virtualization
- Energy-Awareness
Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - Used by more than 3,025 organizations in 89 countries
  - More than 564,000 (> 0.5 million) downloads from the OSU site directly
  - Empowering many TOP500 clusters (Nov ‘18 ranking)
    - 3rd, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China
    - 5th, 448,448 cores (Frontera) at TACC
    - 8th, 391,680 cores (ABCI) in Japan
    - 15th, 570,020 cores (Neurion) in South Korea and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, and OpenHPC)

- [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu) Partner in the TACC Frontera System

- Empowering Top500 systems for over a decade
**MVAPICH Project Timeline**

- **MVAPICH**
  - Oct-02
  - EOL

- **MVAPICH2**
  - Jan-04
  - Nov-04

- **MVAPICH2-X**
  - Aug-14
  - Aug-15

- **MVAPICH2-EA**
  - Sep-15

- **MVAPICH2-Virt**
  - Jul-15

- **MVAPICH2-MIC**
  - Aug-15

- **MVAPICH2-GDR**
  - Sep-15

- **OSU-INAM**
MVAPICH2 Release Timeline and Downloads

Network Based Computing Laboratory
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
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
  - 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
<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), 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>
MVAPICH2 2.3.2

- Released on 08/09/2019
- Major Features and Enhancements
  - Improved performance for inter-node communication
  - Improved performance for Gather, Reduce, and Allreduce with cyclic hostfile
    - Thanks to X-ScaleSolutions for the patch
  - Improved performance for intra-node point-to-point communication
  - Add support for Mellanox HDR adapters
  - Add support for Cascade lake systems
  - Add support for Microsoft Azure platform
    - Enhanced point-to-point and collective tuning for Microsoft Azure
  - Add support for new NUMA-aware hybrid binding policy
  - Add support for AMD EPYC Rome architecture
  - Improved multi-rail selection logic
  - Enhanced heterogeneity detection logic
  - Enhanced point-to-point and collective tuning for AMD EPYC Rome, Frontera@TACC, Mayer@Sandia, Pitzer@OSC, Summit@ORNL, Lassen@LLNL, and Sierra@LLNL systems
  - Add multiple PVARs and CVARs for point-to-point and collective operations
Highlights of MVAPICH2 2.3.2-GA Release

- Support for highly-efficient inter-node and intra-node communication
- Scalable Start-up
- Optimized Collectives using SHArP and Multi-Leaders
- Support for OpenPOWER and ARM architectures
- Performance Engineering with MPI-T
- Application Scalability and Best Practices
One-way Latency: MPI over IB with MVAPICH2

**Small Message Latency**

- TrueScale-QDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
- ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch
- ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
- ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch
- Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch
- ConnectX-6-HDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch

**Large Message Latency**
Bandwidth: MPI over IB with MVAPICH2

Unidirectional Bandwidth

Bidirectional Bandwidth

TrueScale-QDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch
ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch
Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch
ConnectX-6-HDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch
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)

**PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia ‘14)


**Start-up Performance on KNL + Omni-Path**

- **MPI_Init** takes 22 seconds on 229,376 processes on 3,584 KNL nodes (Stampede2 – Full scale)
- 8.8 times faster than Intel MPI at 128K processes (Courtesy: TACC)
- At 64K processes, **MPI_Init** and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
- All numbers reported with 64 processes per node

**New designs available since MVAPICH2-2.3a and as patch for SLURM-15.08.8 and SLURM-16.05.1**
Startup Performance on TACC Frontera

MPI_Init on Frontera

- 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 in MVAPICH2-2.3.2
MVAPICH2 2.3.2 significantly improves performance on top of MVAPICH2 2.3.1

All numbers reported with 56 processes per node
Benefits of SHARP Allreduce at Application Level

**Avg DDOT Allreduce time of HPCG**

- **MVAPICH2**
- **MVAPICH2-SHArP**

<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)
Intra-node Point-to-Point Performance on OpenPower

**Intra-Socket Small Message Latency**

- **MVAPICH2-2.3.1**
- **SpectrumMPI-2019.02.07**

**0.22us**

**Intra-Socket Large Message Latency**

- **MVAPICH2-2.3.1**
- **SpectrumMPI-2019.02.07**

**Intra-Socket Bandwidth**

- **MVAPICH2-2.3.1**
- **SpectrumMPI-2019.02.07**

**Intra-Socket Bi-directional Bandwidth**

- **MVAPICH2-2.3.1**
- **SpectrumMPI-2019.02.07**

**Platform:** Two nodes of OpenPOWER (POWER9-ppc64le) CPU using Mellanox EDR (MT4121) HCA
Intra-node Point-to-point Performance on ARM Cortex-A72

**Small Message Latency**

0.27 micro-second
(1 bytes)

**Large Message Latency**

**Bandwidth**

**Bi-directional Bandwidth**

Platform: ARM Cortex A72 (aarch64) processor with 64 cores dual-socket CPU. Each socket contains 32 cores.
Performance Engineering Applications using MVAPICH2 and TAU

- Enhance existing support for MPI_T in MVAPICH2 to expose a richer set of performance and control variables
- Get and display MPI Performance Variables (PVARs) made available by the runtime in TAU
- Control the runtime’s behavior via MPI Control Variables (CVARs)
- Introduced support for new MPI_T based CVARs to MVAPICH2
  - `MPIR_CVAR_MAX_INLINE_MSG_SZ`, `MPIR_CVAR_VBUF_POOL_SIZE`, `MPIR_CVAR_VBUF_SECONDARY_POOL_SIZE`
- TAU enhanced with support for setting MPI_T CVARs in a non-interactive mode for uninstrumented applications
- S. Ramesh, A. Maheo, S. Shende, A. Malony, H. Subramoni, and D. K. Panda, MPI Performance Engineering with the MPI Tool Interface: the Integration of MVAPICH and TAU, EuroMPI/USA ‘17, Best Paper Finalist
- More details in Sameer Shende’s talk today and poster presentations

VBUF usage without CVAR based tuning as displayed by ParaProf

<table>
<thead>
<tr>
<th>Name</th>
<th>Min.</th>
<th>Max.</th>
<th>Mean.</th>
<th>Std. Dev.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg_send_vbuf_memory_usage</code></td>
<td>3,133,016</td>
<td>3,133,016</td>
<td>3,133,016</td>
<td>0</td>
<td>3,133,016</td>
</tr>
<tr>
<td><code>msg_send_vbuf_allocated</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_available</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_free</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_max_use</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_allocated_max</code></td>
<td>320</td>
<td>320</td>
<td>320</td>
<td>0</td>
<td>320</td>
</tr>
<tr>
<td><code>msg_send_vbuf_available_max</code></td>
<td>215</td>
<td>215</td>
<td>215</td>
<td>0</td>
<td>215</td>
</tr>
<tr>
<td><code>msg_send_vbuf_free_max</code></td>
<td>21,345</td>
<td>21,345</td>
<td>21,345</td>
<td>0</td>
<td>21,345</td>
</tr>
<tr>
<td><code>msg_send_vbuf_max_use_max</code></td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td><code>msg_send_vbuf_max_use_max</code></td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>0</td>
<td>80</td>
</tr>
</tbody>
</table>

VBUF usage with CVAR based tuning as displayed by ParaProf

<table>
<thead>
<tr>
<th>Name</th>
<th>Min.</th>
<th>Max.</th>
<th>Mean.</th>
<th>Std. Dev.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg_send_vbuf_memory_usage</code></td>
<td>1,813,150</td>
<td>1,813,150</td>
<td>1,813,150</td>
<td>0</td>
<td>1,813,150</td>
</tr>
<tr>
<td><code>msg_send_vbuf_allocated</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_available</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_free</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_max_use</code></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><code>msg_send_vbuf_allocated_max</code></td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td><code>msg_send_vbuf_available_max</code></td>
<td>94</td>
<td>94</td>
<td>94</td>
<td>0</td>
<td>94</td>
</tr>
<tr>
<td><code>msg_send_vbuf_free_max</code></td>
<td>5,470</td>
<td>5,470</td>
<td>5,470</td>
<td>0</td>
<td>5,470</td>
</tr>
<tr>
<td><code>msg_send_vbuf_max_use_max</code></td>
<td>56</td>
<td>56</td>
<td>56</td>
<td>0</td>
<td>56</td>
</tr>
</tbody>
</table>
Application Scalability on Skylake and KNL

MiniFE (1300x1300x1300 ~ 910 GB)

NEURON (YuEtAl2012)

Cloverleaf (bm64) MPI+OpenMP, NUM_OMP_THREADS = 2

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

Runtime parameters: MV2_SMPI_LENGTH_QUEUE=524288 PSM2_MQ_RNDV_SHM_THRESH=128K PSM2_MQ_RNDV_HFI_THRESH=128K
MVAPICH2-Azure 2.3.2

- Released on 08/16/2019
- Major Features and Enhancements
  - Based on MVAPICH2-2.3.2
  - Enhanced tuning for point-to-point and collective operations
  - Targeted for Azure HB & HC virtual machine instances
  - Flexibility for 'one-click' deployment
  - Tested with Azure HB & HC VM instances
Performance of Radix

Total Execution Time on HC (Lower is better)

- MVAPICH2-X
- HPCx

Number of Processes (Nodes X PPN)

Total Execution Time on HB (Lower is better)

- MVAPICH2-X
- HPCx

38% faster

3x faster
Performance of FDS (HC)

Part of input parameter: MESH IJK=5,5,5, XB=-1.0,0.0,-1.0,0.0,0.0,1.0, MULT_ID='mesh array'

1.11x better
MVAPICH2 Upcoming Features

• Integration of SHARP2 and associated Collective Optimizations
• Communication optimizations on upcoming architectures
  – Intel Cooper Lake
  – AMD Rome
  – ARM
• Dynamic and Adaptive Communication Protocols
Dynamic and Adaptive MPI Point-to-point Communication Protocols

Desired Eager Threshold

<table>
<thead>
<tr>
<th>Process Pair</th>
<th>Eager Threshold (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 4</td>
<td>32</td>
</tr>
<tr>
<td>1 – 5</td>
<td>64</td>
</tr>
<tr>
<td>2 – 6</td>
<td>128</td>
</tr>
<tr>
<td>3 – 7</td>
<td>32</td>
</tr>
</tbody>
</table>

Eager Threshold for Example Communication Pattern with Different Designs

- **Default**
  - Poor overlap; Low memory requirement
  - Low Performance; High Productivity

- **Manually Tuned**
  - Good overlap; High memory requirement
  - High Performance; Low Productivity

- **Dynamic + Adaptive**
  - Good overlap; Optimal memory requirement
  - High Performance; High Productivity

**Execution Time of Amber**

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Default</th>
<th>Threshold=17K</th>
<th>Threshold=64K</th>
<th>Threshold=128K</th>
<th>Dynamic Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relative Memory Consumption of Amber**

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Default</th>
<th>Threshold=17K</th>
<th>Threshold=64K</th>
<th>Threshold=128K</th>
<th>Dynamic Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 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), 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>
MVAPICH2-X for Hybrid MPI + PGAS Applications

<table>
<thead>
<tr>
<th>High Performance Parallel Programming Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPI</strong></td>
</tr>
<tr>
<td>Message Passing Interface</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>High Performance and Scalable Unified Communication Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Diverse APIs and Mechanisms</strong></td>
</tr>
<tr>
<td>Optimized Point-to-point Primitives</td>
</tr>
<tr>
<td>Support for Efficient Intra-node Communication (POSIX SHMEM, CMA, LiMIC, XPMEM...)</td>
</tr>
</tbody>
</table>

- **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 2.3rc2

- Released on 03/01/2019
- Major Features and Enhancements
  - MPI Features
  - Based on MVAPICH2 2.3.1
    - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
  - MPI (Advanced) Features
    - Improved performance of large message communication
    - Support for advanced co-operative (COOP) rendezvous protocols in SMP channel
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Support for RGET, RPUT, and COOP protocols for CMA and XPMEM
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Support for load balanced and dynamic rendezvous protocol selection
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Support for XPMEM-based MPI collective operations (Broadcast, Gather, Scatter, Allgather)
      - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
    - Extend support for XPMEM-based MPI collective operations (Reduce and All-Reduce for PSM-CH3 and PSM2-CH3 interfaces
  - Improved connection establishment for DC transport
    - OFA-IB-CH3 interface
  - Add improved Alltoallv algorithm for small messages
    - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
  - OpenSHMEM Features
    - Support for XPMEM-based collective operations (Broadcast, Collect, Reduce_all, Reduce, Scatter, Gather)
  - UPC Features
    - Support for XPMEM-based collective operations (Broadcast, Collect, Scatter, Gather)
  - UPC++ Features
    - Support for XPMEM-based collective operations (Broadcast, Collect, Scatter, Gather)
  - Unified Runtime Features
    - Based on MVAPICH2 2.3.1 (OFA-IB-CH3 interface). All the runtime features enabled by default in OFA-IB-CH3 and OFA-IB-RoCE interface of MVAPICH2 2.3.1 are available in MVAPICH2-X 2.3rc2
## 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
Overview of Some of the MVAPICH2-X Features

• Direct Connect (DC) Transport
• Co-operative Rendezvous Protocol
• Advanced All-reduce with SHARP
• CMA-based Collectives
• Asynchronous Progress
• XPMEM-based Reduction Collectives
• XPMEM-based Non-reduction Collectives
• Optimized Collective Communication and Advanced Transport Protocols
• PGAS and Hybrid MPI+PGAS Support
Minimizing Memory Footprint by Direct Connect (DC) Transport

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

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
Cooperative Rendezvous Protocols

- Use both sender and receiver CPUs to progress communication concurrently
- Dynamically select rendezvous protocol based on communication primitives and sender/receiver availability (load balancing)
- Up to 2x improvement in large message latency and bandwidth
- Up to 19% improvement for Graph500 at 1536 processes

Cooperative Rendezvous Protocols for Improved Performance and Overlap
S. Chakraborty, M. Bayatpour, J Hashmi, H. Subramoni, and DK Panda,
SC ‘18 (Best Student Paper Award Finalist)

Platform: 2x14 core Broadwell 2680 (2.4 GHz)
Mellanox EDR ConnectX-5 (100 GBps)
Baseline: MVAPICH2X-2.3rc1, Open MPI v3.1.0
Available in MVAPICH2-X 2.3rc2
Advanced Allreduce Collective Designs Using SHArP and Multi-Leaders

- Socket-based design can reduce the communication latency by 23% and 40% on Broadwell + IB-EDR nodes

- Support is available since MVAPICH2-X 2.3b

Performance of MPI_Allreduce On Stampede2 (10,240 Processes)

- MPI_Allreduce latency with 32K bytes reduced by 2.4X
Optimized CMA-based Collectives for Large Messages

Performance of MPI_Gather on KNL nodes (64PPN)

- 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
Overview of Some of the MVAPICH2-X Features

- Direct Connect (DC) Transport
- Co-operative Rendezvous Protocol
- Advanced All-reduce with SHARP
- CMA-based Collectives
- Asynchronous Progress
- XPMEM-based Reduction Collectives
- XPMEM-based Non-reduction Collectives
- Optimized Collective Communication and Advanced Transport Protocols
- PGAS and Hybrid MPI+PGAS Support
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
Reduction Collectives on IBM OpenPOWER

- Two POWER8 dual-socket nodes each with 20 ppn
- Up to 2X improvement for Allreduce and 3X improvement for Reduce at 4MB message
- Used osu_reduce and osu_allreduce from OSU Microbenchmarks v5.5
Application Level Benefits of XPMEM-based Designs

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

- 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

MiniAMR (dual-socket, ppn=16)
Impact of XPMEM-based Designs on MiniAMR

- Two POWER8 dual-socket nodes each with 20 ppn
- MiniAMR application execution time comparing MVAPICH2-2.3rc1 and optimized All-Reduce design
  - MiniAMR application for weak-scaling workload on up to three POWER8 nodes.
  - Up to 45% improvement over MVAPICH2-2.3rc1 in mesh-refinement time
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

![Graph showing latency vs message size for broadcast and gather operations]

**Broadcast**
- Intel MPI 2018
- OpenMPI 3.0.1
- MV2X-2.3rc1 (CMA Coll)
- MV2X-2.3rc2 (XPMEM Coll)

**Gather**
- Intel MPI 2018
- OpenMPI 3.0.1
- MV2X-2.3rc1 (CMA Coll)
- MV2X-2.3rc2 (XPMEM Coll)

Latency (us) vs Message Size (Bytes)
Impact of Optimized Small Message MPI_Alltoallv Algorithm

- Optimized designs in MVAPICH2-X offer significantly improved performance for small message MPI_Alltoallv

![Graph showing latency vs. message size for MVAPICH2-X and HPE-MPI. The graph indicates that MVAPICH2-X is ~5X better than HPE-MPI.](image)

- Up to 5X benefits over HPE-MPI using optimized using optimized Alltoallv algorithm and Direct Connected transport protocol
- Numbers taken on bbpv2.epfl.ch
  - 96 KNL nodes with 64 ppn (6,144 processes)
  - osu_alltoallv from OSU Micro Benchmarks
- 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*

*Courtesy: Pramod Shivaji Kumbhar@EPFL*
Application Level Performance with Graph500 and Sort

- 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

- 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

J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012
MVAPICH2-X-AWS 2.3

- Released on 08/12/2019
- Major Features and Enhancements
  - Based on MVAPICH2-X 2.3
  - New design based on Amazon EFA adapter's Scalable Reliable Datagram (SRD) transport protocol
  - Support for XPMEM based intra-node communication for point-to-point and collectives
  - Enhanced tuning for point-to-point and collective operations
  - Targeted for AWS instances with Amazon Linux 2 AMI and EFA support
  - Tested with c5n.18xlarge instance
Point-to-Point Performance

- Both UD and SRD shows similar latency for small messages.
- SRD shows higher message rate due to lack of software reliability overhead.
- SRD is faster for large messages due to larger MTU size.
Collective Performance: MPI Gatherv

- Up to 33% improvement with SRD compared to UD
- Root does not need to send explicit acks to non-root processes
- Non-roots can exit as soon as the message is sent (no need to wait for acks)
Collective Performance: MPI Allreduce

- Up to 18% improvement with SRD compared to UD
- Bidirectional communication pattern allows piggybacking of acks
- Modest improvement compared to asymmetric communication patterns
Application Performance

- Up to 10% performance improvement for MiniGhost on 8 nodes
- Up to 27% better performance with CloverLeaf on 8 nodes

S. Chakraborty, S. Xu, H. Subramoni and D. K. Panda, Designing Scalable and High-Performance MPI Libraries on Amazon Elastic Adapter, Hot Interconnect, 2019
MVAPICH2-X Upcoming Features

- XPMEM-based MPI Derived Datatype Designs
- Exploiting Hardware Tag Matching
Efficient Zero-copy MPI Datatypes for Emerging Architectures

- New designs for efficient zero-copy based MPI derived datatype processing
- Efficient schemes mitigate datatype translation, packing, and exchange overheads
- Demonstrated benefits over prevalent MPI libraries for various application kernels
- To be available in the upcoming MVAPICH2-X release

![Graph showing performance improvements](image)

- **3D-Stencil** Datatype Kernel on Broadwell (2x14 core)
- **MILC** Datatype Kernel on KNL 7250 in Flat-Quadrant Mode (64-core)
- **NAS-MG** Datatype Kernel on OpenPOWER (20-core)
Hardware Tag Matching Support

• Offloads the processing of point-to-point MPI messages from the host processor to HCA

• Enables zero copy of MPI message transfers
  – Messages are written directly to the user's buffer without extra buffering and copies

• Provides rendezvous progress offload to HCA
  – Increases the overlap of communication and computation
Impact of Zero Copy MPI Message Passing using HW Tag Matching

Removal of intermediate buffering/copies can lead up to 35% performance improvement in latency of medium messages
Impact of Rendezvous Offload using HW Tag Matching

The increased overlap can lead to 1.8X performance improvement in total latency of osu_iscatterv
# 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>
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: MVAPICH2-GDR 1.8-2.3.2 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
• Unified memory
MVAPICH2-GDR 2.3.2

- Released on 08/08/2019
- Major Features and Enhancements
  - Based on MVAPICH2 2.3.1
  - Support for CUDA 10.1
  - Support for PGI 19.x
  - Enhanced intra-node and inter-node point-to-point performance
  - Enhanced MPI_Allreduce performance for DGX-2 system
  - 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
  - Enhanced intra-node and inter-node point-to-point performance for DGX-2 and IBM POWER8 and IBM POWER9 systems
  - Enhanced Allreduce performance for DGX-2 and IBM POWER8/POWER9 systems
  - Enhanced small message performance for CUDA-Aware MPI_Put and MPI_Get
  - Flexible support for running TensorFlow (Horovod) jobs
Optimized MVAPICH2-GDR Design

**GPU-GPU Inter-node Latency**

- MV2-(NO-GDR)
- MV2-GDR 2.3

1.85us, 10x

**GPU-GPU Inter-node Bandwidth**

- MV2-(NO-GDR)
- MV2-GDR-2.3

9x

**GPU-GPU Inter-node Bi-Bandwidth**

11x

Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
NVIDIA Volta V100 GPU
Mellanox Connect-X4 EDR HCA
CUDA 9.0
Mellanox OFED 4.0 with GPU-Direct-RDMA
Device-to-Device Performance on OpenPOWER (NVLink2 + Volta)

**Intra-node Latency (Small)**: 5.36 us (without GDRCopy)

**Intra-node Latency (Large)**: 5.66 us (without GDRCopy)

**Inter-node Latency (Small)**: 

**Inter-node Latency (Large)**: 

**Available since MVAPICH2-GDR 2.3a**

**Intra-node Bandwidth**: 70.4 GB/sec for 128MB (via NVLINK2)

**Inter-node Bandwidth**: 23.7 GB/sec (2 port EDR)

Platform: OpenPOWER (POWER9-ppc64le) nodes equipped with a dual-socket CPU, 4 Volta V100 GPUs, and 2port EDR InfiniBand Interconnect
Application-Level Evaluation (HOOMD-blue)

- **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

---

**64K Particles**

Average Time Steps per second (TPS)

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>MV2</th>
<th>MV2+GDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**256K Particles**

Average Time Steps per second (TPS)

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>MV2</th>
<th>MV2+GDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland

Wilkes GPU Cluster

- Default
- Callback-based
- Event-based

Normalized Execution Time

Number of GPUs

CSCS GPU cluster

- Default
- Callback-based
- Event-based

Normalized Execution Time

Number of GPUs

- 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/
Deep Learning: New Challenges for MPI Runtimes

- Deep Learning frameworks are a different game altogether
  - Unusually large message sizes (order of megabytes)
  - Most communication based on GPU buffers

- Existing State-of-the-art
  - cuDNN, cuBLAS, NCCL --> **scale-up** performance
  - NCCL2, CUDA-Aware MPI --> **scale-out** performance
    - For small and medium message sizes only!

- Proposed: Can we **co-design** the MPI runtime (**MVAPICH2-GDR**) and the DL framework (**Caffe**) to achieve both?
  - Efficient **Overlap** of Computation and Communication
  - Efficient **Large-Message** Communication (Reductions)
  - What **application co-designs** are needed to exploit communication-runtime co-designs?

---

Scalable TensorFlow using Horovod, MPI, and NCCL

- Efficient Allreduce is crucial for Horovod’s overall training performance
  - Both MPI and NCCL designs are available
- We have evaluated Horovod extensively and compared across a wide range of designs using gRPC and gRPC extensions
- MVAPICH2-GDR achieved up to 90% scaling efficiency for ResNet-50 Training on 64 Pascal GPUs

https://arxiv.org/abs/1810.11112
MVAPICH2-GDR vs. NCCL2 – Allreduce Operation

- Optimized designs in MVAPICH2-GDR 2.3 offer better/comparable performance for most cases
- MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 16 GPUs

Platform: Intel Xeon (Broadwell) nodes equipped with a dual-socket CPU, 1 K-80 GPUs, and EDR InfiniBand Inter-connect
Optimized designs in upcoming 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 9.2
MVAPICH2-GDR: Enhanced MPI_Allreduce at Scale

- Optimized designs in upcoming 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)**

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

Platform: The Summit Supercomputer (#1 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 9.2
MVAPICH2-GDR Upcoming Features for HPC and DL

- Scalable Host-based Collectives
- Enhanced Derived Datatype
- Integrated Collective Support with SHArP from GPU Buffers
- Optimization for PyTorch and MXNET
Scalable Host-based Collectives on OpenPOWER (Intra-node Reduce & AlltoAll)

**Reduce**

- MVAPICH2-GDR
- SpectrumMPI-10.1.0.2
- OpenMPI-3.0.0

*Up to 5X and 3x performance improvement by MVAPICH2 for small and large messages respectively*

**Alltoall**

- MVAPICH2-GDR
- SpectrumMPI-10.1.0.2
- OpenMPI-3.0.0

*Up to 3.6X and 3.3X performance improvement by MVAPICH2 for small and large messages respectively*
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

**GPU-based DDTBench mimics MILC communication kernel**

Speedup

<table>
<thead>
<tr>
<th>MILC Problem size</th>
<th>OpenMPI 4.0.0</th>
<th>MVAPICH2-GDR 2.3.1</th>
<th>MVAPICH2-GDR-Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6, 8,8,8]</td>
<td>0</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>[6, 8,8,16]</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>[6, 8,8,16,16]</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>[6, 16,16,16,16]</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

Platform: Nvidia DGX-2 system
*(NVIDIA Volta GPUs connected with NVSwitch), CUDA 9.2*

**Communication Kernel of COSMO Model**
(https://github.com/cosunae/HaloExchangeBenchmarks)

Improved 3.4X

<table>
<thead>
<tr>
<th>Number of GPUs</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Improved 15X</td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>

Platform: Cray CS-Storm
*(16 NVIDIA Tesla K80 GPUs per node), CUDA 8.0*
## 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>
Overview of OSU INAM

• A network monitoring and analysis tool that is capable of analyzing traffic on the InfiniBand network with inputs from the MPI runtime
• Monitors IB clusters in real time by querying various subnet management entities and gathering input from the MPI runtimes
• Capability to analyze and profile node-level, job-level and process-level activities for MPI communication
  – Point-to-Point, Collectives and RMA
• Ability to filter data based on type of counters using “drop down” list
• Remotely monitor various metrics of MPI processes at user specified granularity
• "Job Page" to display jobs in ascending/descending order of various performance metrics in conjunction with MVAPICH2-X
• Visualize the data transfer happening in a “live” or “historical” fashion for entire network, job or set of nodes
• OSU INAM 0.9.4 released on 11/10/2018
  – Enhanced performance for fabric discovery using optimized OpenMP-based multi-threaded designs
  – Ability to gather InfiniBand performance counters at sub-second granularity for very large (>2000 nodes) clusters
  – Redesign database layout to reduce database size
  – Enhanced fault tolerance for database operations
    • Thanks to Trey Dockendorf @ OSC for the feedback
  – OpenMP-based multi-threaded designs to handle database purge, read, and insert operations simultaneously
  – Improved database purging time by using bulk deletes
  – Tune database timeouts to handle very long database operations
  – Improved debugging support by introducing several debugging levels
OSU INAM Features

- 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

Comet@SDSC --- Clustered View

(1,879 nodes, 212 switches, 4,377 network links)
OSU INAM Features (Cont.)

- **Job level view**
  - Show different network metrics (load, error, etc.) for any live job
  - Play back historical data for completed jobs to identify bottlenecks

- **Node level view - details per process or per node**
  - CPU utilization for each rank/node
  - Bytes sent/received for MPI operations (pt-to-pt, collective, RMA)
  - Network metrics (e.g. XmitDiscard, RcvError) per rank/node

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

More Details in Tutorial/Demo

Session Tomorrow
OSU Microbenchmarks

• Available since 2004

• Suite of microbenchmarks to study communication performance of various programming models

• Benchmarks available for the following programming models
  – Message Passing Interface (MPI)
  – Partitioned Global Address Space (PGAS)
    • Unified Parallel C (UPC)
    • Unified Parallel C++ (UPC++)
    • OpenSHMEM

• Benchmarks available for multiple accelerator based architectures
  – Compute Unified Device Architecture (CUDA)
  – OpenACC Application Program Interface

• Part of various national resource procurement suites like NERSC-8 / Trinity Benchmarks

• Continuing to add support for newer primitives and features

• Please visit the following link for more information
  – http://mvapich.cse.ohio-state.edu/benchmarks/
Applications-Level Tuning: Compilation of Best Practices

- MPI runtime has many parameters
- Tuning a set of parameters can help you to extract higher performance
- Compiled a list of such contributions through the MVAPICH Website
  - [http://mvapich.cse.ohio-state.edu/best_practices/](http://mvapich.cse.ohio-state.edu/best_practices/)
- Initial list of applications
  - Amber
  - HoomDBlue
  - HPCG
  - Lulesh
  - MILC
  - Neuron
  - SMG2000
  - Cloverleaf
  - SPEC (LAMMPS, POP2, TERA_TF, WRF2)
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.
MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1-10M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF ...)
  - MPI + Task*
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features of Mellanox InfiniBand
  - Tag Matching*
  - Adapter Memory*
- Enhanced communication schemes for upcoming architectures
  - 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
Commercial Support for MVAPICH2, HiBD, and HiDL Libraries

- Supported through X-ScaleSolutions (http://x-scalesolutions.com)
- Benefits:
  - Help and guidance with installation of the library
  - Platform-specific optimizations and tuning
  - Timely support for operational issues encountered with the library
  - Web portal interface to submit issues and tracking their progress
  - Advanced debugging techniques
  - Application-specific optimizations and tuning
  - Obtaining guidelines on best practices
  - Periodic information on major fixes and updates
  - Information on major releases
  - Help with upgrading to the latest release
  - Flexible Service Level Agreements
- Support provided to Lawrence Livermore National Laboratory (LLNL) for the last two years
Silver ISV Member for the OpenPOWER Consortium + Products

• Has joined the OpenPOWER Consortium as a silver ISV member
• Provides flexibility:
  – To have MVAPICH2, HiDL and HiBD libraries getting integrated into the OpenPOWER software stack
  – A part of the OpenPOWER ecosystem
  – Can participate with different vendors for bidding, installation and deployment process
• Introduced two new integrated products with support for OpenPOWER systems (Presented yesterday at the OpenPOWER North America Summit)
  – X-ScaleHPC
  – X-ScaleAI
  – Send an e-mail to contactus@x-scalesolutions.com for free trial!!
Funding Acknowledgments

Funding Support by

Equipment Support by
Personnel Acknowledgments

Current Students (Graduate)
- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- C.-H. Chu (Ph.D.)
- J. Hashmi (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Kandadi (M.S.)
- Awan Raj (M.S.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- A. Quentin (Ph.D.)
- B. Ramesh (M.S.)
- S. Xu (M.S.)

Current Research Scientist
- Q. Zhou (Ph.D.)

Past Students
- 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.)
- B. Chandrasekharan (M.S.)
- S. Chakraborty (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhananjay (M.S.)
- T. Gangadharappa (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.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandall (Ph.D.)
- M. Li (Ph.D.)

Past Research Scientists
- 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.)
- A. G. Vaidyanathan (Ph.D.)
- E. Mancini
- H. Subramoni

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

Current Students (Undergraduate)
- V. Gangal (B.S.)
- N. Sarkauskas (B.S.)

Current Research Specialist
- J. Smith

Past Research Scientists
- M. S. Ghazimeersaeed
- A. Ruhela
- K. Manian

Past Programmers
- D. Bureddy
- J. Perkins

Past Research Specialist
- M. Arnold
Thank You!

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

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/