Designing High-Performance MPI Libraries for Multi-/Many-core Era

Talk at IXPUG-Fall Conference (September ‘18)

J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni and D. K Panda
The Ohio State University
E-mail: {Hashmi.29,chakraborty.52,bayatpour.1,subramoni.1,panda.2}@osu.edu
Increasing Usage of HPC, Big Data and Deep Learning

Convergence of HPC, Big Data, and Deep Learning!

Increasing Need to Run these applications on the Cloud!!
Parallel Programming Models Overview

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

Shared Memory Model
- SHMEM, DSM

Distributed Memory Model
- MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)
- Global Arrays, UPC, Chapel, X10, CAF, ...
Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges

Application Kernels/Applications

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

Middleware

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, 40/100GigE, Aries, and Omni-Path)

Multi-/Many-core Architectures

Accelerators (GPU and FPGA)

Co-Design Opportunities and Challenges across Various Layers
Performance Scalability Resilience
Broad Challenges in Designing Runtimes for (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
Additional Challenges for Designing Exascale Software Libraries

- **Extreme Low Memory Footprint**
  - Memory per core continues to decrease

- **D-L-A Framework**
  - **Discover**
    - Overall network topology (fat-tree, 3D, ...), Network topology for processes for a given job
    - Node architecture, Health of network and node
  - **Learn**
    - Impact on performance and scalability
    - Potential for failure
  - **Adapt**
    - Internal protocols and algorithms
    - Process mapping
    - Fault-tolerance solutions
  - **Low overhead techniques while delivering performance, scalability and fault-tolerance**
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 2,950 organizations in 86 countries
  - More than 494,000 (> 0.49 million) downloads from the OSU site directly
  - Empowering many TOP500 clusters (Jul ’18 ranking)
    - 2nd ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China
    - 12th, 556,104 cores (Oakforest-PACS) in Japan
    - 15th, 367,024 cores (Stampede2) at TACC
    - 24th, 241,108-core (Pleiades) at NASA and many others
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)

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

- Empowering Top500 systems for over a decade
Architecture of MVAPICH2 Software Family

High Performance Parallel Programming Models

- Message Passing Interface (MPI)
- PGAS (UPC, OpenSHMEN, 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: UMR, 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
## MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI with IB, iWARP, Omni-Path, and RoCE</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>Advanced MPI Features/Support, OSU INAM, PGAS and MPI+PGAS with IB, Omni-Path, and RoCE</td>
<td>MVAPICH2-X</td>
</tr>
<tr>
<td>MPI with IB, RoCE &amp; GPU and Support for Deep Learning</td>
<td>MVAPICH2-GDR</td>
</tr>
<tr>
<td>HPC Cloud with MPI &amp; IB</td>
<td>MVAPICH2-Virt</td>
</tr>
<tr>
<td>Energy-aware MPI with IB, iWARP and RoCE</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 A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

• Scalability for million to billion processors
  – Support for highly-efficient inter-node and intra-node communication
  – Scalable Start-up
  – Optimized Collectives using SHArP and Multi-Leaders
  – Optimized CMA-based Collectives
  – Optimized XPMEM-based Collectives
  – SALaR: Scalable and Adaptive Designs for Large Message Reduction Collectives
  – Asynchronous Progress
  – MPI-T support

• Integrated Support for GPGPUs and Deep Learning
• Optimized MVAPICH2 for OpenPower (with/ NVLink) and ARM
• Application Scalability and Best Practices
• High-Performance MPI Library for Cloud
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-DualFDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
- ConnectX-5-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

**Large Message Latency**

- TrueScale-QDR
- ConnectX-3-FDR
- ConnectIB-DualFDR
- ConnectX-5-EDR
- Omni-Path
Bandwidth: MPI over IB with MVAPICH2

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-5-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 IB switch
Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch
• MPI_Init takes 22 seconds on 231,936 processes on 3,624 KNL nodes (Stampede2 – Full scale)
• At 64K processes, MPI_Init and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
• All numbers reported with 64 processes per node, MVAPICH2-2.3a
• Designs integrated with mpirun_rsh, available for srun (SLURM launcher) as well
Benefits of SHARP at Application Level

### Avg DDOT Allreduce time of HPCG

**Latency (seconds)**

<table>
<thead>
<tr>
<th>(Number of Nodes, PPN)</th>
<th>MVAPICH2</th>
<th>12%</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,28)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(8,28)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(16,28)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SHARP support available since MVAPICH2 2.3a**

### Parameter Description

<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/mvapich2-2.3b-userguide.html#x1-990006.26](http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3b-userguide.html#x1-990006.26)
For MPI_Allreduce latency with 32K bytes, MVAPICH2-OPT can reduce the latency by 2.4X.


Available since MVAPICH2-X 2.3b
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
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 in MVAPICH2-X 2.3rc1
Application-Level Benefits of XPMEM-Based Collectives

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

- 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
Impact of SALaR (Scalable Large Message Collectives) Designs on CNTK

- CPU-based training of AlexNet neural network using ImageNet ILSVRC2012 dataset
- SALaR designs show up to 46% improved performance over MVAPICH2 at 896 processes
- The proposed designs show good scalability with increasing system size


Will be available in future MVAPICH2 Release
Benefits of A New Asynchronous Progress Design: SPEC MPI 2008

- Up to 25% performance improvement for SPECMPI applications on 384 processes with KNL + Omni-Path
- Up to 38% performance improvement for SPECMPI applications on 384 processes with Skylake + Omni-Path

*A. Ruhela, H. Subramoni, S. Chakraborty, M. Bayatpour, P. Kousha, and D. K. Panda, Efficient Asynchronous Communication Progress for MPI without Dedicated Resources, EuroMPI ‘18*
Benefits of the New Asynchronous Progress Design: P3DFFT

Up to 44% performance improvement with the P3DFFT application with 448 processes
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

VBUF usage without CVAR based tuning as displayed by ParaProf

<table>
<thead>
<tr>
<th>Name</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Sd.Dev.</th>
<th>NDsamples</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpi2_VBUF_usage (Number of VBUFs used)</td>
<td>3,313,056</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_max (Number of VBUFs used)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_max use (maximum number of VBUFs used)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

VBUF usage with CVAR based tuning as displayed by ParaProf

<table>
<thead>
<tr>
<th>Name</th>
<th>MaxValue</th>
<th>MinValue</th>
<th>MeanValue</th>
<th>Sd.Dev.</th>
<th>NDsamples</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpi2_VBUF_usage (Number of VBUFs used)</td>
<td>3,313,056</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_max (Number of VBUFs used)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_max use (maximum number of VBUFs used)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mpi2_VBUF_free (Number of VBUFs freed)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Integrated Support for GPGPUs and Deep Learning
- Optimized MVAPICH2 for OpenPower (with/ NVLink) and ARM
- Application Scalability and Best Practices
- High-Performance MPI Library for Cloud
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 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
Optimized MVAPICH2-GDR Design

GPU-GPU Inter-node Latency

GPU-GPU Inter-node Bandwidth

Bandwidth (MB/s)

Message Size (Bytes)

Latency (us)

Message Size (Bytes)

- MV2-(NO-GDR)
- MV2-GDR 2.3rc1

GPU-GPU Inter-node Bi-Bandwidth

Bandwidth (MB/s)

Message Size (Bytes)

- MV2-(NO-GDR)
- MV2-GDR-2.3rc1

MVAPICH2-GDR-2.3
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
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)
- Number of Processes: 4, 8, 16, 32
- MV2 vs. MV2+GDR: 2X improvement

**256K Particles**

- Average Time Steps per second (TPS)
- Number of Processes: 4, 8, 16, 32
- MV2 vs. MV2+GDR: 2X improvement
Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland

Wilkes GPU Cluster

CSCS GPU cluster

- 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/
Exploiting CUDA-Aware MPI for TensorFlow (Horovod)

- MVAPICH2-GDR offers excellent performance via advanced designs for MPI_Allreduce.
- Up to **22% better** performance on Wilkes2 cluster (16 GPUs)
MVAPICH2-GDR: Allreduce Comparison with Baidu and OpenMPI

- 16 GPUs (4 nodes) MVAPICH2-GDR vs. Baidu-Allreduce and OpenMPI 3.0

*Available since MVAPICH2-GDR 2.3a
Optimized designs in MVAPICH2-GDR 2.3rc1 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.
OSU-Caffe: Proposed Co-Design Overview

• To address the limitations of Caffe and existing MPI runtimes, we propose the OSU-Caffe (S-Caffe) framework

• At the application (DL framework) level
  – Develop a fine-grain workflow – i.e. layer-wise communication instead of communicating the entire model

• At the runtime (MPI) level
  – Develop support to perform reduction of very-large GPU buffers
  – Perform reduction using GPU kernels

OSU-Caffe is available from the HiDL project page
http://hidl.cse.ohio-state.edu
S-Caffe vs. Inspur-Caffe and Microsoft CNTK

- AlexNet: Notoriously hard to scale-out on multiple nodes due to comm. overhead!
- Large number of parameters ~ 64 Million (comm. buffer size = 256 MB)
- GoogLeNet is a popular DNN
- 13 million parameters (comm. buffer size = ~50 MB)

S-Caffe delivers better or comparable performance with other multi-node capable DL frameworks

Impact of HR

Up to 14% improvement (Scale-up)
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Integrated Support for GPGPUs and Deep Learning
- Optimized MVAPICH2 for OpenPower (with/ NVLink) and ARM
- Application Scalability and Best Practices
- High-Performance MPI Library for Cloud
Intra-node Point-to-Point Performance on OpenPower

Intra-Socket Small Message Latency

- MVAPICH2-2.3
- SpectrumMPI-10.1.0.2
- OpenMPI-3.0.0

Intra-Socket Large Message Latency

- MVAPICH2-2.3
- SpectrumMPI-10.1.0.2
- OpenMPI-3.0.0

Intra-Socket Bandwidth

- MVAPICH2-2.3
- SpectrumMPI-10.1.0.2
- OpenMPI-3.0.0

Intra-Socket Bi-directional Bandwidth

- MVAPICH2-2.3
- SpectrumMPI-10.1.0.2
- OpenMPI-3.0.0

Platform: Two nodes of OpenPOWER (Power8-ppc64le) CPU using Mellanox EDR (MT4115) HCA
Inter-node Point-to-Point Performance on OpenPower

Network Based Computing Laboratory

Platform: Two nodes of OpenPOWER (Power8-ppc64le) CPU using Mellanox EDR (MT4115) HCA
**MVAPICH2-GDR: Performance on OpenPOWER (NVLink + Pascal)**

**Intra-node Latency (Small)**
- Latency (us) vs Message Size (Bytes)
- INTRA-SOCKET (NVLink) vs INTER-SOCKET

**Intra-node Latency (Large)**
- Latency (us) vs Message Size (Bytes)
- INTRA-SOCKET (NVLink) vs INTER-SOCKET

**Inter-node Latency (Small)**
- Latency (us) vs Message Size (Bytes)

**Inter-node Latency (Large)**
- Latency (us) vs Message Size (Bytes)

**Inter-node Bandwidth**
- Bandwidth (GB/sec) vs Message Size (Bytes)
- INTRA-SOCKET (NVLink) vs INTER-SOCKET

**Intra-node Latency: 13.8 us (without GPUDirectRDMA)**

**Inter-node Latency: 23 us (without GPUDirectRDMA)**

Available since MVAPICH2-GDR 2.3a

**Platform:** OpenPOWER (ppc64le) nodes equipped with a dual-socket CPU, 4 Pascal P100-SXM GPUs, and 4X-FDR InfiniBand Inter-connect

**Intra-node Bandwidth: 33.2 GB/sec (NVLink)**

**Inter-node Bandwidth: 6 GB/sec (FDR)**
Scalable Host-based Collectives with CMA on OpenPOWER (Intra-node Reduce & AlltoAll)

**Reduce**

- **MVAPICH2-X-2.3rc1**
- **SpectrumMPI-10.1.0.2**
- **OpenMPI-3.0.0**

**AlltoAll**

- **MVAPICH2-X-2.3rc1**
- **SpectrumMPI-10.1.0.2**
- **OpenMPI-3.0.0**

*Up to 5X and 3x performance improvement by MVAPICH2 for small and large messages respectively*
Optimized All-Reduce with XPMEM on OpenPOWER

- Optimized MPI All-Reduce Design in MVAPICH2
  - Up to 2X performance improvement over Spectrum MPI and 4X over OpenMPI for intra-node

Optimized Runtime Parameters: MV2_CPU_BINDING_POLICY=hybrid MV2_HYBRID_BINDING_POLICY=bunch
Intra-node Point-to-point Performance on ARM Cortex-A72

**Small Message Latency**

- MVAPICH2-2.3

0.27 micro-second
(1 bytes)

**Large Message Latency**

- MVAPICH2-2.3

**Bandwidth**

- MVAPICH2-2.3

**Bi-directional Bandwidth**

- MVAPICH2-2.3

Platform: ARM Cortex A72 (aarch64) MIPS processor with 64 cores dual-socket CPU. Each socket contains 32 cores.
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Integrated Support for GPGPUs and Deep Learning
- Optimized MVAPICH2 for OpenPower (with/ NVLink) and ARM
- Application Scalability and Best Practices
- High-Performance MPI Library for Cloud
SPEC MPI 2007 Benchmarks: Broadwell + InfiniBand

MVAPICH2-X-2.3rc1 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

Intel MPI 18.1.163

MVAPICH2-X-2.3rc1

Execution Time in (s)
SPEC MPI 2007 Benchmarks: KNL + Omni-Path

- MVAPICH2-X outperforms Intel MPI by up to 22%

Configuration: 384 processes on 8 nodes of Intel Xeon Phi 7250 (KNL) with 48 processes per node. KNL contains 68 cores on a single socket and interconnects with 100Gb/sec Intel Omni-Path network.
Application Scalability on Skylake and KNL (Stampede2)

MiniFE (1300x1300x1300 ~ 910 GB)

NEURON (YuEtAl2012)

Cloverleaf (bm64) MPI+OpenMP,
NUM_OMP_THREADS = 2

Runtime parameters:
MV2_SMPI_LENGTH_QUEUE=524288
PSM2_MQ_RNDV_SHM_THRESH=128K
PSM2_MQ_RNDV_HFI_THRESH=128K

Courtesy: Mahidhar Tatineni @SDSC, Dong Ju (DJ) Choi@SDSC, and Samuel Khuvis@OSC ---- Testbed: TACC Stampede2 using MVAPICH2-2.3b
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.
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

• Scalability for million to billion processors
• Integrated Support for GPGPUs and Deep Learning
• Optimized MVAPICH2 for OpenPower (with/ NVLink) and ARM
• Application Scalability and Best Practices
• High-Performance MPI Library for Cloud
Can HPC and Virtualization be Combined?

- Virtualization has many benefits
  - Fault-tolerance
  - Job migration
  - Compaction

- Have not been very popular in HPC due to overhead associated with Virtualization

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

- Enhanced MVAPICH2 support for SR-IOV

- MVAPICH2-Virt 2.2 supports:
  - OpenStack, Docker, and singularity

J. Zhang, X. Lu, J. Jose, R. Shi and D. K. Panda, Can Inter-VM Shmem Benefit MPI Applications on SR-IOV based Virtualized InfiniBand Clusters? EuroPar'14
J. Zhang, X. Lu, J. Jose, M. Li, R. Shi and D.K. Panda, High Performance MPI Library over SR-IOV enabled InfiniBand Clusters, HiPC’14
Application-Level Performance on Chameleon

- 32 VMs, 6 Core/VM
- Compared to Native, 2-5% overhead for Graph500 with 128 Procs
- Compared to Native, 1-9.5% overhead for SPEC MPI2007 with 128 Procs
Application-Level Performance on Singularity with MVAPICH2

- 512 Processes across 32 nodes
- Less than 7% and 6% overhead for NPB and Graph500, respectively

J. Zhang, X. Lu and D. K. Panda, Is Singularity-based Container Technology Ready for Running MPI Applications on HPC Clouds?, UCC ’17, Best Student Paper Award
MVAPICH2-GDR on Container with Negligible Overhead

**GPU-GPU Inter-node Latency**

- **Latency (us)** vs **Message Size (Bytes)**
- **Docker** vs **Native**

**GPU-GPU Inter-node Bandwidth**

- **Bandwidth (MB/s)** vs **Message Size (Bytes)**
- **Docker** vs **Native**

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

- **Bandwidth (MB/s)** vs **Message Size (Bytes)**
- **Docker** vs **Native**

**System Configuration**

- 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

**MVAPICH2-GDR-2.3a**

Works with NVIDIA HPC Container Maker

MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1M-10M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF ...)
  - MPI + Task*
- Enhanced Optimization for GPUs and FPGAs*
- Taking advantage of advanced features of Mellanox InfiniBand
  - Tag Matching*
  - Adapter Memory*
- Enhanced communication schemes for upcoming architectures
  - NVLINK*
  - 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
Funding Acknowledgments

**Funding Support by**

Office of Science  
NSF  
Mellanox Technologies  
Microsoft  
QLogic  
CRAY  
Intel  
Linux Network  
NVIDIA  
NetApp  
Sun Microsystems  
Cisco Systems

**Equipment Support by**

Mellanox Technologies  
Intel  
AMD  
Sun Microsystems  
AFPR  
Advanced Clustering Technologies, Inc.  
Microwave Technology  
NVIDIA  
QLogic
Personnel Acknowledgments

**Current Students (Graduate)**
- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)
- J. Hashmi (Ph.D.)
- V. Gangal (B.S.)
- H. Javed (Ph.D.)
- P. Kousha (Ph.D.)
- D. Shankar (Ph.D.)
- N. Sarkauskas (B.S.)

**Current Students (Undergraduate)**

**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.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (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. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- 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. Potluri (Ph.D.)
- 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. Potluri (Ph.D.)

**Current Research Scientists**
- X. Lu
- H. Subramoni

**Current Research Specialist**
- J. Smith
- M. Arnold

**Current Post-doc**
- A. Ruhela
- K. Manian

**Past Research Scientist**
- K. Hamidouche
- S. Sur

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

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

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

Network Based Computing Laboratory
IXPUG (Sept ’18)
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/

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

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