Extracting Peak Performance for your Applications on Frontera with MVAPICH2 Libraries

A Talk at Frontera User Meeting (Jan’21)

by

Hari Subramoni
The Ohio State University

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

Follow us on
https://twitter.com/mvapich
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)
- 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 3,125 organizations in 89 countries
- More than 1.2 Million downloads from the OSU site directly
- Empowering many TOP500 clusters (Nov ‘20 ranking)
  - 4th, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 9th, 448,448 cores (Frontera) at TACC
  - 14th, 391,680 cores (ABCI) in Japan
  - 21th, 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 9th ranked TACC Frontera system
- Empowering Top500 systems for more than 16 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

Summary of all tests for one commit  
Summary of an individual test  
Details of individual combinations in one test
**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 Release Timeline and Downloads

Number of Downloads

Timeline

MV 0.9.4
MV 2.0.9.0
MV 2.0.9.8
MV 2.1.0
MV 2.1.0.3
MV 2.1.1
MV 2.1.4
MV 2.1.5
MV 2.1.6
MV 2.1.7
MV 2.1.8
MV 2.1.9
MV 2.2
MV 2.2.0
MV 2.2.3
MV 2.2.5
MV 2.3.0
MV 2.3.2
MV 2.3.5
OSU INAM 0.9.6
MV 2.3.2
MV 2.3.5
MV 2-GDR 2.0
MV 2-MIC 2.0
MV 2-Azure 2.0
MV 2-X 2.3
MV 2-Virt 2.2
MV 2-GDR 2.3.5
MV 2-AWS 2.3
MV 2-Virt 2.3
MV 2-GDR 2.3.5
## 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 MVAPICH2 Features

- Job start-up
- Transport Type Selection
- Collectives
- Support for MPI Tools (MPI_T) Interface
- Solutions for NVIDIA/AMD GPU-enabled Systems
- MPI-based Deep Learning for CPUs and GPUs
- Accelerating Data Science Applications
- Application Specific Tuning
• MPI_Init takes 31 seconds on 229,376 processes on 4,096 nodes
• All numbers reported with 56 processes per node

New designs available from MVAPICH2-2.3.4
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

![Graph](image)

- UD
- Hybrid
- RC

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>26%</td>
</tr>
<tr>
<td>256</td>
<td>40%</td>
</tr>
<tr>
<td>512</td>
<td>38%</td>
</tr>
<tr>
<td>1024</td>
<td>30%</td>
</tr>
</tbody>
</table>

### Parameter Significance

<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
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*

---

**Graph:**

- **Neuron with YuEtAl2012**
- **Execution Time (s)**
- **No. of Processes**
- **MVAPICH2**
- **MVAPICH2-X**
- **Overhead of RC protocol for connection establishment and communication**

- 512: 10% & 76%
- 1024: 39%
- 2048: 39%
- 4096: 10%
Collective Communication in MVAPICHH2

- Blocking and Non-Blocking Collective Algorithms in MV2
  - Conventional (Flat)
  - Multi/Many-Core Aware Designs

- Inter-Node Communication
  - Point to Point
  - Hardware Multicast
  - SHARP
  - RDMA

- Intra-Node Communication
  - Point to Point (SHMEM, LiMIC, CMA*, XPMEM*)
  - Direct Shared Memory
  - Direct Kernel Assisted (CMA*, XPMEM*, LiMIC)

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 MVAPICHH2-X

Designed for Performance & Overlap
Hardware Multicast-aware MPI_Bcast on TACC Frontera

- MCAST-based designs improve latency of MPI_Bcast by up to **2X** at 2,048 nodes
- Use MV2_USE_MCAST=1 to enable MCAST-based designs
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
Performance of Collectives with SHARP on TACC Frontera

*Optimized SHARP designs in MVAPICH2-X*

*Up to 9X* performance improvement with SHARP over MVAPICH2-X default for 1ppn MPI_Barrier, *6X* for 1ppn MPI_Reduce and *5X* for 1ppn MPI_Allreduce


*Optimized Runtime Parameters: MV2_ENABLE_SHARP = 1*
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

Available in MVAPICH2

VBUF usage without CVAR based tuning as displayed by ParaProf

VBUF usage with CVAR based tuning as displayed by ParaProf
Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland

- 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

---

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

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6, 8, 8, 8]</td>
<td>[6, 8, 8, 8, 16]</td>
</tr>
</tbody>
</table>

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

---

**Communication Kernel of COSMO Model**

![Improved 3.4X](https://github.com/cosunae/HaloExchangeBenchmarks)

**Platform:** Cray CS-Storm (16 NVIDIA Tesla K80 GPUs per node), CUDA 8.0

**Execution Time (s)**

- Improved 15X
- Improved 3.4X
• Designs GPU-assisted on-the-fly message compression show 37% higher GFLOPs for the AWP-ODC on Frontera-Liquid and Frontera-Longhorn

• Will be available in future MVAPICH2-GDR releases

MVAPICH2-GDR ROCm Support for AMD GPUs

Intra-Node Point-to-Point Latency

![Graph of Intra-Node Point-to-Point Latency](image)

1.77 us

Message Size (Bytes)

Inter-Node Point-to-Point Latency

![Graph of Inter-Node Point-to-Point Latency](image)

3.44 us

Message Size (Bytes)

Allreduce – 64 GPUs (8 nodes, 8 GPUs Per Node)

![Graph of Allreduce](image)

16.54 us

Message Size (Bytes)

Bcast – 64 GPUs (8 nodes, 8 GPUs Per Node)

![Graph of Bcast](image)

7.06 us

Message Size (Bytes)

Corona Cluster - ROCm-3.9.0 (mi50 AMD GPUs)

Available with MVAPICH2-GDR 2.3.5
MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training

ML/DL Applications

- TensorFlow
- PyTorch
- MXNet

Horovod

ML/DL Applications

- PyTorch

MVAPICH2 or MVAPICH2-X for CPU Training

MVAPICH2-GDR for GPU Training

Torch.distributed

DeepSpeed

MVAPICH2 or MVAPICH2-X for CPU Training

MVAPICH2-GDR for GPU Training

More details available from: http://hidl.cse.ohio-state.edu
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
Distributed TensorFlow on TACC Frontera (2,048 CPU nodes with 114,688 cores)

- Scaled TensorFlow to 2048 nodes on Frontera using MVAPICH2
- MVAPICH2 and IntelMPI give similar performance for DNN training
- Report a peak of **260,000 images/sec** on 2,048 nodes
- On 2048 nodes, ResNet-50 can be trained in **7 minutes**!

Distributed TensorFlow on ORNL Summit (1,536 GPUs)

- ResNet-50 Training using TensorFlow benchmark on SUMMIT -- 1536 Volta GPUs!
- 1,281,167 (1.2 mil.) images
- Time/epoch = 3 seconds
- Total Time (90 epochs) = 3 x 90 = 270 seconds = 4.5 minutes!

"ImageNet-1k has 1.2 million images"

"MVAPICH2-GDR reaching ~0.42 million images per second for ImageNet-1k!"

*We observed issues for NCCL2 beyond 384 GPUs

Platform: The Summit Supercomputer (#2 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 10.1
Accelerating cuDF Merge – Longhorn (TACC Frontera GPU Subsystem)

2.91x better on average

![Graph showing performance comparison]

2.90x better on average

![Graph showing performance comparison]


MPI4Dask 0.1 release
(http://hibd.cse.ohio-state.edu)
Accelerating cuML with MVAPICH2-GDR on Longhorn

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
Neuron: Impact of Tuning Transport Protocol

- 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
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
  - 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
Funding Acknowledgments

Funding Support by

Equipment Support by
# Acknowledgments to all the Heroes (Past/Current Students and Staffs)

## Current Students (Graduate)
- Q. Anthony (Ph.D.)
- M. Bayatpour (Ph.D.)
- C.-C. Chun (Ph.D.)
- A. Jain (Ph.D.)
- 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.)
- A. H. Tu (Ph.D.)
- 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.)
- B. Chandrasekharan (M.S.)
- S. Chakraborty (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharpappa (M.S.)
- K. Gopalakrishnan (M.S.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- M. Kedia (M.S.)
- S. Kini (M.S.)
- M. Koo (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.)
- K. Raj (M.S.)
- P. Rajachandrasekar (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- 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.)
- A. Ruhela
- J. Vienne
- H. Wang

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

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

## Past Research Specialist
- M. Arnold

## Current Research Scientists
- A. Shafi
- H. Subramoni
- J. Hashmi

## Current Research Specialist
- J. Smith

## Current Software Engineers
- A. Reifsteck
- N. Shineman
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/