Designing High-Performance and Scalable Middleware for HPC, AI and Data Science based on MPI: The MVAPICH2-based Approach

Keynote Talk at ExaMPI ‘21 Workshop (Nov. ‘21)

by

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

Follow us on
https://twitter.com/mvapich
High-End Computing (HEC): PetaFlop to ExaFlop

100 PetaFlops in 2017

442 PetaFlops in 2020 (Fugaku in Japan with 7.63M cores)

1 ExaFlops

Expected to have an ExaFlop system in 2022!
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)

Percentage of Clusters vs. Number of Clusters over time. The graph shows an increasing trend from Nov-96 to Nov-19, with a percentage of 98.4% by Nov-19.
Drivers of Modern HPC Cluster Architectures

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.
Increasing Usage of HPC, Deep/Machine Learning, and Data Science

Convergence of HPC, Deep/Machine Learning, and Data Science!

Increasing Need to Run these applications on the Cloud!!

Can MPI-Driven Middleware be designed and used for all three domains?
Designing Communication Libraries for Multi-Petaflop and Exaflop Systems: Challenges

**Application Kernels/Applications (HPC, DL, Data Science)**

**Middleware**

**Programming Models**
- MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenMP, OpenACC, Hadoop, Spark (RDD, DAG), TensorFlow, PyTorch, 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, Ethernet, RoCE, Omni-Path, and Slingshot

**Multi-/Many-core Architectures**

**Accelerators**
- (GPU and FPGA)

**Co-Design Opportunities and Challenges across Various Layers**
- Performance
- Scalability
- Resilience

**Performance**
- Scalability
- Resilience
Presentation Overview

- **MVAPICH Project**
  - MPI and PGAS (MVAPICH) Library with CUDA-Awareness
  - Accelerating applications with DPU

- **HiDL Project**
  - High-Performance Deep Learning
  - High-Performance Machine Learning

- **HiBD Project**
  - Accelerating Data Science Applications with Dask

- **Optimizations and Deployments in Public Cloud**
  - AWS, Azure, and Oracle

- **Commercial Support and Value-Added Products**

- **Conclusions**
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
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 (since 2014) and AMD (since 2020) GPUs
  – 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 3,200 organizations in 89 countries

• More than 1.5 Million downloads from the OSU site directly

• Empowering many TOP500 clusters (June ‘21 ranking)
  – 4th, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  – 10th, 448,448 cores (Frontera) at TACC
  – 20th, 288,288 cores (Lassen) at LLNL
  – 31st, 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 10th ranked TACC Frontera system

• Empowering Top500 systems for more than 15 years
MVAPICH2 Release Timeline and Downloads

Number of Downloads

Timeline

- MV 0.9.4
- MV2 0.9.0
- MV2 0.9.8
- MV1 1.0
- MV2 1.0.3
- MV2 1.1
- MV2 1.4
- MV2 1.5
- MV2 1.6
- MV2 1.7
- MV2 1.8
- MV2 1.9
- MV2-GDR 2.0b
- MV2-Virt 2.2
- MV2-MIC 2.0
- MV2-Azure 2.3.2
- MV2-GDR 2.3.6
- OSU INAM 0.9.6
- MV2-AWS 2.3
- MV2 Virt 2.2
- MV2-GDR 2.3.6
- MV2 Virt 2.6
- MV2-GDR 2.3.6

Network Based Computing Laboratory
Exa-MPI '21 (Nov '21)
# Architecture of MVAPICH2 Software Family for HPC, DL/ML, and Data Science

## High Performance Parallel Programming Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Passing Interface (MPI)</td>
<td></td>
</tr>
<tr>
<td>PGAS</td>
<td>(UPC, OpenSHMEM, CAF, UPC++)</td>
</tr>
<tr>
<td>Hybrid --- MPI + X</td>
<td>(MPI + PGAS + OpenMP/Cilk)</td>
</tr>
</tbody>
</table>

## 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, Elastic Fabric Adapter)*

- **Transport Protocols**
  - RC
  - SRD
  - 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 Protocols**
  - SHM
  - CMA
  - IVSHMEM
  - XPMEM

- **Modern Features**
  - Optane*
  - 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/Machine 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>
Highlights of some of the MVAPICH2 Designs

- Cooperative Rendezvous Protocol for intra-node communication
- Direct Connect (DC) Protocol for Scalable inter-node communication with Reduced Memory Footprint
- Scalable Collective Communication Support with SHARP
- Hardware Tag-Matching Support
- Neighborhood Collectives
- Optimized Derived Datatype Support
- QoS-aware Design
- Non-blocking Collective Support with DPUs
- GPU-Direct RDMA (GDR) Design
- On-the-Fly Compression for GPU-GPU Communication
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 since MVAPICH2-X 2.3rc2
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

**Memory Footprint for Alltoall**

- NAMD - Apoa1: Large data set

**Normalized Execution Time**

- Connection Memory (KB)
- Number of Processes

Impact of DC Transport Protocol on Neuron

Neuron with YuEtAl2012

- 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

More details in talk

“Building Brain Circuits: Experiences with shuffling terabytes of data over MPI”, by Matthias Wolf at MUG’20

https://www.youtube.com/watch?v=TFi8O3-Hznw
AMD Milan + HDR 200

**Intra-Node CPU Point-to-Point**

**Latency**

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

**Bandwidth**

![Intra-Node CPU Point-to-Point Bandwidth Graph](image)

**Inter-Node CPU Point-to-Point**

**Latency**

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

**Bandwidth**

![Inter-Node CPU Point-to-Point Bandwidth Graph](image)

AMD EPYC 7V13 64-Core Processor, Mellanox ConnectX-6 HDR HCA
Support for ARM A64FX with InfiniBand (Ookami)

Inter-Node Latency

![Graph showing inter-node latency for various message sizes with MVAPICH2-X and OpenMPI comparisons.]

Inter-Node Bidirectional Bandwidth

![Graph showing inter-node bidirectional bandwidth for various message sizes with MVAPICH2-X and OpenMPI comparisons.]

Intra-Node Latency

![Graph showing intra-node latency for various message sizes with MVAPICH2-X and OpenMPI comparisons.]

MPI_Bcast (8 nodes 48 PPN)

![Graph showing MPI_Bcast latency for various message sizes with MVAPICH2-X and OpenMPI comparisons.]

Network Based Computing Laboratory
Exa-MPI ‘21 (Nov ‘21)
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 of MPI_Ialltoall using HW Tag Matching

- Up to 1.8x Performance Improvement, Sustained benefits as system size increases
Execution time of the proposed scheme is up to 35% better than Intel-MPI, 7.8% better than OpenMPI, and 9% better than MVAPICH2 at a scale of 128 nodes (7K cores).

K. Suresh, C. Chen, B. Ramesh, SM Ghazimirsaeed, M. Bayatpour, A. Shafi, H. Subramoni and DK Panda, "Layout-aware Hardware-assisted Designs for Derived Data Types in MPI", HiPC ’21 (To be presented)
Optimized Designs for Neighborhood Collectives

- **SpMM**
  - up to 34x speedup

- **NAS DT**
  - up to 15% improvement

Will be available in upcoming MVAPICH2-X Release

Inter-node point-to-point Latency and Bandwidth (Rockport Networks)

**Small message Latency**

- MVAPICH2

**Medium/Large message Latency**

- MVAPICH2

- MVAPICH2 delivers around 3.0 microsec latency for small messages
- Using multiple QPs gives up to 3.9x reduction in latency

**Uni-directional Bandwidth**

- MVAPICH2
- MVAPICH2-4-QP

**Bi-Directional Bandwidth**

- MVAPICH2

- MVAPICH2-4-QP delivers
  - 10229 MB/sec peak unidirectional bandwidth
  - 20165 MB/Sec peak bi-directional bandwidth

- 4x
- 4.5x

- 3.9x
Performance of Neuroscience Mini-Application with MVAPICH2-X

Comparison of Execution Time (s) on 9,920 cores

<table>
<thead>
<tr>
<th>Neighbors Selection</th>
<th>HPE-MPI</th>
<th>MVAPICH2-X</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>188.07</td>
<td>120.22</td>
<td>36.07</td>
</tr>
<tr>
<td>Consecutive</td>
<td>124.46</td>
<td>116.34</td>
<td>6.52</td>
</tr>
</tbody>
</table>

• EPFL Mini-Application is a benchmark where each process sends data to a selected set of neighbors.
• Up to 36% benefits over HPE-MPI on Neuroscience Mini-Application at 496 nodes and 20 PPN

Available in upcoming version of MVAPICH2-X
Highlights of some of the MVAPICH2 Designs

- Cooperative Rendezvous Protocol for intra-node communication
- Direct Connect (DC) Protocol for Scalable inter-node communication with Reduced Memory Footprint
- Scalable Collective Communication Support with SHARP
- Hardware Tag-Matching Support
- QoS-aware Design
  - Non-blocking Collective Support with DPUs
  - GPU-Direct RDMA (GDR) Design
  - On-the-Fly Compression for GPU-GPU Communication
Overview of BlueField-2 DPU

- ConnectX-6 network adapter with 200Gbps InfiniBand
- System-on-chip containing eight 64-bit ARMv8 A72 cores with 2.75 GHz each
- 16 GB of memory for the ARM cores

How can one re-design an MPI library to take advantage of DPUs and accelerate scientific applications?
Can MPI Functions be Offloaded to Bluefield-DPU?

- State-of-the-art BlueField DPUs bring more compute power into the network
- Can we exploit additional compute capabilities of modern BlueField DPUs into existing MPI middleware to extract
  - Peak pure communication performance
  - Overlap of communication and computation

For dense non-blocking collective communications?

- What will be the benefits at the applications level?
Overlap of Communication and Computation with osu_Ialltoall (32 nodes)

Delivers peak overlap

32 Nodes, 16 PPN

32 Nodes, 32 PPN
Total Execution Time with osu_ialltoall (32 nodes)

Benefits in Total execution time (Compute + Communication)
P3DFFT Application Execution Time (32 nodes)

Benefits in application-level execution time

32 Nodes, 16 PPN

32 Nodes, 32 PPN
Total Execution Time with osu_iallgather (16 nodes)

Benefits in Overall iallgather
(Computation and Communication)

16 Nodes, 1 PPN

16 Nodes, 16 PPN
Total Execution Time with osu_ibcast (16 nodes)

Benefits in Overall Ibcast
(Computation and Communication)
Total Execution Time with osu_Iscatter (16 nodes)

Benefits in Overall Iscatter (Computation and Communication)

Available in upcoming MVAPICH2-DPU Release
MVAPICH2-DPU Library 2021.08 Release

- Based on MVAPICH2 2.3.6
- Released on 08/22/21
- Supports all features available with the MVAPICH2 2.3.6 release
  (http://mvapich.cse.ohio-state.edu)
- Novel framework to offload non-blocking collectives to DPU
  - MPI_Ialltoall to DPU
  - MPI_Iallgather to DPU
  - MPI_Ibcast to DPU

Available from X-ScaleSolutions, please send a note to contactus@x-scalesolutions.com to get a trial license.
Highlights of some of the MVAPICH2 Designs

- Cooperative Rendezvous Protocol for intra-node communication
- Direct Connect (DC) Protocol for Scalable inter-node communication with Reduced Memory Footprint
- Scalable Collective Communication Support with SHARP
- Hardware Tag-Matching Support
- QoS-aware Design
- Non-blocking Collective Support with DPUs
- **GPU-Direct RDMA (GDR) Design**
- **On-the-Fly Compression for GPU-GPU Communication**
Optimized MVAPICH2-GDR with CUDA-Aware MPI Support

**GPU-GPU Inter-node Latency**

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

**GPU-GPU Inter-node Bandwidth**

- MV2-(NO-GDR)
- MV2-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
**ROCM-aware MVAPICH2-GDR** – Support for AMD GPUs

Corona Cluster @ LLNL - ROCm-4.3.0 (mi50 AMD GPUs)

**Intra-Node GPU Point-to-Point**

**Latency**

- MVAPICH2-GDR
- OpenMPI 4.1.1 + UCX 1.11.1

**Bandwidth**

- MVAPICH2-GDR
- OpenMPI 4.1.1 + UCX 1.11.1

**Inter-Node GPU Point-to-Point**

**Latency**

- MVAPICH2-GDR
- OpenMPI 4.1.1 + UCX 1.11.1

**Bandwidth**

- MVAPICH2-GDR
- OpenMPI 4.1.1 + UCX 1.11.1

---

ROCm-aware MVAPICH2-GDR - RCCL Integration

Broadcast:

Allreduce:

Corona Cluster @ LLNL - ROCm-4.3.0 (mi50 AMD GPUs)

RCCL v4.5.0

Will be available in MVAPICH2-GDR-Next Release

“On-the-fly” Compression Support in MVAPICH2-GDR

- Weak-Scaling of HPC application **AWP-ODC** on Lassen cluster (V100 nodes) [1]
- MPC-OPT achieves up to **+18%** GPU computing flops, **-15%** runtime per timestep
- ZFP-OPT achieves up to **+35%** GPU computing flops, **-26%** runtime per timestep

---

**MVAPICH Accelerates Parallel 3-D FFT at Oak Ridge**

Comparison of achievable bandwidth for two-node exchange via MPI_Send

MVAPICH is around 10-20% faster than SpectrumMPI 10.3 for heFFTe Library

Accelerating the communication cost on parallel 3-D FFTs, Stan Tomov and Alan Ayala, The University of Tennessee, Knoxville

(http://mug.mvapich.cse.ohio-state.edu/static/media/mug/presentations/21/Ayala.pdf)
Collectives Performance on DGX-A100

**Allgater (4 nodes, 8 ppn, 32 GPUs)**

- MVAPICH2-GDR 2.3.6
- NCCL 2.10.3

**Bcast (4 nodes, 8 ppn, 32 GPUs)**

- MVAPICH2-GDR 2.3.6
- NCCL 2.10.3

**Reduce (4 nodes, 8 ppn, 32 GPUs)**

- MVAPICH2-GDR 2.3.6
- NCCL 2.10.3

**Allreduce (4 nodes, 8 ppn, 32 GPUs)**

- MVAPICH2-GDR 2.3.6
- NCCL 2.10.3
Collectives Performance on DGX-A100 - Alltoall

Alltoall (16 nodes, 8 ppn, 128 GPUs) - small size

- MVAPICH2-GDR
- NCCL 2.11.4
- OpenMPI 4.1.1 + UCX 1.11.1

Latency (us) vs Size (bytes)

Alltoall (16 nodes, 8 ppn, 128 GPUs) - large size

- MVAPICH2-GDR
- NCCL 2.11.4
- OpenMPI 4.1.1 + UCX 1.11.1

Latency (us) vs Size (bytes)

heffte w/ alltoall (4 nodes, 8 ppn, 32 GPUs)

- MVAPICH2-GDR 2.3.6
- OpenMPI 4.1.1 + UCX 1.11.1

Performance (GFlops/s) vs Size (cube)

heffte w/ alltoall (16 nodes, 8 ppn, 128 GPUs)

- MVAPICH2-GDR 2.3.6
- OpenMPI 4.1.1 + UCX 1.11.1

Performance (GFlops/s) vs Size (cube)

heffte: https://github.com/af-ayala/heffte
MVAPICH Drives Nuclear Energy Research at Idaho National Lab

The MOOSE Multiphysics Computational Framework for Nuclear Power Applications: A Special Issue of Nuclear Technology

MVAPICH Integration for PBS Pro, HPC Team, Idaho National Laboratory
(http://mug.mvapich.cse.ohio-state.edu/static/media/mug/presentations/21/inl.pdf)
Presentation Overview

- **MVAPICH Project**
  - MPI and PGAS (MVAPICH) Library with CUDA-Awareness
  - Accelerating applications with DPU

- **HiDL Project**
  - High-Performance Deep Learning
  - High-Performance Machine Learning

- **HiBD Project**
  - Accelerating Data Science Applications with Dask

- **Optimizations and Deployments in Public Cloud**
  - AWS, Azure, and Oracle

- **Commercial Support and Value-Added Products**

- **Conclusions**
MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training

ML/DL Applications

- TensorFlow
- PyTorch
- MXNet

Horovod

MVAPICH2 or MVAPICH2-X for CPU Training

MVAPICH2-GDR for GPU Training

ML/DL Applications

- PyTorch

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
Multiple Approaches taken up by OSU

- MPI-driven Deep Learning with Data Parallelism
- Out-of-core DNN training
- Exploiting Hybrid (Data and Model) Parallelism
- Use-Case: AI-Driven Digital Pathology
- Accelerating CuML Applications
Distributed TensorFlow on TACC Frontera (2,048 CPU nodes with 114,688 cores)

- Scaled TensorFlow to 2048 nodes on Frontera using MVAPICH2
- 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!

*We observed issues for NCCL2 beyond 384 GPUs

*ImageNet-1k has 1.2 million images*

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

Platform: The Summit Supercomputer (#2 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 10.1
PyTorch at Scale: Training ResNet-50 on 256 V100 GPUs

- Training performance for 256 V100 GPUs on LLNL Lassen
  - ~10,000 Images/sec faster than NCCL training!

<table>
<thead>
<tr>
<th>Distributed Framework</th>
<th>Torch.distributed</th>
<th>Horovod</th>
<th>DeepSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images/sec on 256 GPUs</td>
<td>61,794</td>
<td>72,120</td>
<td>84,659</td>
</tr>
<tr>
<td>Communication Backend</td>
<td>NCCL 2.7</td>
<td>MVAPICH2-GDR</td>
<td>NCCL 2.7</td>
</tr>
</tbody>
</table>
Multiple Approaches taken up by OSU

- MPI-driven Deep Learning with Data Parallelism
- Out-of-core DNN training
- Exploiting Hybrid (Data and Model) Parallelism
- Use-Case: AI-Driven Digital Pathology
- Accelerating CuML Applications
Out-of-Core Training and **Hybrid Parallelism (HyPar-Flow)**

- **Why Hybrid parallelism?**
  - Data Parallel training has limits! →
- **We propose HyPar-Flow**
  - An easy-to-use Hybrid parallel training framework
    - Hybrid = Data + Model
  - Supports Keras models and exploits TF 2.0 Eager Execution
  - Exploits MPI for Point-to-point and Collectives

**Benchmarking large-models lead to better insights and ability to develop new approaches!**

HyPar-Flow at Scale (512 nodes on TACC Frontera)

- ResNet-1001 with variable batch size
- Approach:
  - 48 model-partitions for 56 cores
  - 512 model-replicas for 512 nodes
  - Total cores: 56 x 512 = 28,672
- Speedup
  - 253X on 256 nodes
  - 481X on 512 nodes
- Scaling Efficiency
  - 98% up to 256 nodes
  - 93.9% for 512 nodes

Two Approaches:
- Memory Aware Synchronized Training (MAST)
- Memory Aware Synchronized Training with Enhanced Replications (MASTER)

Setup
- ResNet-1k on 512 x 512 images
- 128 Replications on 1024 GPUs

Scaling Efficiency
- 97.32% on 1024 nodes

97.32% scaling efficiency on 1024 V100 GPUs (LLNL Lassen)

SUPER: **SU**b-Graph Parallelism for Transform**ER**s

Sub-Graph Parallelism

- Exploits inherent parallelism in modern DNN architectures
- Improves the Performance of multi-branch DNN architectures →
- Can be used to accelerate the training of state-of-the-art Transformer models
- Provides better performance than Data-Parallelism for in-core models

Simple example of a multi-branch DNN architecture

4-way **Sub-Graph Parallelism** combined with Data-Parallelism (D&SP)

We propose sub-graph parallelism integrated with data parallelism to accelerate the training of Transformers.

**Approach**
- Data and Sub-Graph Parallelism (D&SP)
  - #-way D&SP (#: number of sub-graphs)

**Setup**
- T5-Large-Mod on WMT Dataset
- 1,024 NVIDIA V100 GPUs

**Speedup**
- Up to **3.05X** over Data Parallelism (DP)

A. Jain, T. Moon, T. Benson, H. Subramoni, S. Jacobs, D. Panda, B. Essen, “SUPER: SUb-Graph Parallelism for TransformERs”, IPDPS ’21
Multiple Approaches taken up by OSU

- MPI-driven Deep Learning with Data Parallelism
- Out-of-core DNN training
- Exploiting Hybrid (Data and Model) Parallelism
- Use-Case: AI-Driven Digital Pathology
- Accelerating CuML Applications
Exploiting Model Parallelism in AI-Driven Digital Pathology

- Pathology whole slide image (WSI)
  - Each WSI = 100,000 x 100,000 pixels
  - Can not fit in a single GPU memory
  - Tiles are extracted to make training possible

- Two main problems with tiles
  - Restricted tile size because of GPU memory limitation
  - Smaller tiles loose structural information

- Reduced training time significantly
  - GEMS-Basic: 7.25 hours (1 node, 4 GPUs)
  - GEMS-MAST: 6.28 hours (1 node, 4 GPUs)
  - GEMS-MASTER: 4.21 hours (1 node, 4 GPUs)
  - GEMS-Hybrid: 0.46 hours (32 nodes, 128 GPUs)
  - Overall 15x reduction in training time!!!!

A. Jain, A. Awan, A. Aljuhani, J. Hashmi, Q. Anthony, H. Subramoni, D. K. Panda, R. Machiraju, and A. Parwani,


Scaling ResNet110 v2 on 1024×1024 image tiles using histopathology data
Multiple Approaches taken up by OSU

- MPI-driven Deep Learning with Data Parallelism
- Out-of-core DNN training
- Exploiting Hybrid (Data and Model) Parallelism
- Use-Case: AI-Driven Digital Pathology
- Accelerating CuML Applications
Accelerating cuML with MVAPICH2-GDR

- Utilize MVAPICH2-GDR (with mpi4py) as communication backend during the training phase (the fit() function) for Multi-node Multi-GPU (MNMG) setting over cluster of GPUs
- Communication primitives:
  - Allreduce
  - Reduce
  - Broadcast
- Exploit optimized collectives
MPI4cuML 0.1 Release

- MPI4cuML 0.1 was released in Feb ‘21 adding support for high-performance MPI communication to cuML:
  - Can be downloaded from: [http://hidl.cse.ohio-state.edu](http://hidl.cse.ohio-state.edu)

- Features:
  - Built with Python 3.7, CUDA 10.1, 10.2 or 11.0
  - Optimized support at MPI-level for machine learning workloads
    - Efficient large-message and small-message collectives (e.g. Allreduce and Bcast) on GPUs
    - GPU-Direct Algorithms for all collective operations (e.g. Allgather and Alltoall)
    - Support for fork safety
  - Exploits efficient large-message and small-message collectives in MVAPICH2-GDR
  - Tested with
    - Mellanox InfiniBand adapters (FDR and HDR)
    - NVIDIA GPU P100 and V100
    - Various x86-based multi-core platforms (AMD and Intel)
Presentation Overview

• MVAPICH Project
  – MPI and PGAS (MVAPICH) Library with CUDA-Awareness
  – Accelerating applications with DPU

• HiDL Project
  – High-Performance Deep Learning
  – High-Performance Machine Learning

• HiBD Project
  – Accelerating Data Science Applications with Dask

• Optimizations and Deployments in Public Cloud
  – AWS, Azure, and Oracle

• Commercial Support and Value Added Products

• Conclusions
Dask Architecture

Dask
- Dask Bag
- Dask Array
- Dask DataFrame
- Delayed
- Future

Task Graph

Dask-MPI
Dask-CUDA
Dask-Jobqueue

Distributed
- Scheduler
- Worker
- Client

Comm Layer
- tcp.py
- ucx.py
- MPI4Dask

UCX-Py
(Cython wrappers)
- mpi4py
- MVAPICH2-GDR

Comm Layer
- TCP
- UCX

High Performance Computing Hardware

Laptops/Desktops
MPI4Dask Release

• MPI4Dask 0.2 was released in Mar ‘21 adding support for high-performance MPI communication to Dask:
  – Can be downloaded from: http://hibd.cse.ohio-state.edu

• Features:
  – Based on Dask Distributed 2021.01.0
  – Compliant with user-level Dask APIs and packages
  – Support for MPI-based communication in Dask for cluster of GPUs
  – Implements point-to-point communication co-routines
  – Efficient chunking mechanism implemented for large messages
  – (NEW) Built on top of mpi4py over the MVAPICH2, MVAPICH2-X, and MVAPICH2-GDR libraries
  – (NEW) Support for MPI-based communication for CPU-based Dask applications
  – Supports starting execution of Dask programs using Dask-MPI
  – Tested with
    • (NEW) CPU-based Dask applications using numPy and Pandas data frames
    • (NEW) GPU-based Dask applications using cuPy and cuDF
    • Mellanox InfiniBand adapters (FDR and EDR)
    • Various multi-core platforms
    • NVIDIA V100 and Quadro RTX 5000 GPUs
Benchmark #1: Sum of cuPy Array and its Transpose (RI2)

- **3.47x better on average**
- **6.92x better on average**

**Total Execution Time (s)**

<table>
<thead>
<tr>
<th>Number of Dask Workers</th>
<th>IPoIB</th>
<th>UCX</th>
<th>MPI4Dask</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Communication Time (s)**

<table>
<thead>
<tr>
<th>Number of Dask Workers</th>
<th>IPoIB</th>
<th>UCX</th>
<th>MPI4Dask</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.47</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2.67</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>4</td>
<td>1.91</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1.2</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.8</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>


MPI4Dask 0.2 release
(http://hibd.cse.ohio-state.edu)
Benchmark #2: cuDF Merge (TACC Frontera GPU Subsystem)

2.91x better on average

2.90x better on average


MPI4Dask 0.2 release
(http://hibd.cse.ohio-state.edu)
On-the-fly GPU Compression (Benefits with MPI4Dask)

- Data science framework Dask on RI2 cluster (V100 nodes)
- Dask benchmark creates cuPy array and distributes its chunks across Dask workers
- ZFP-OPT achieves up to 1.56x throughput, -37% runtime (rate=8, compression ratio=4)

Presentation Overview

- MVAPICH Project
  - MPI and PGAS (MVAPICH) Library with CUDA-Awareness
  - Accelerating applications with DPU

- HiDL Project
  - High-Performance Deep Learning
  - High-Performance Machine Learning

- HiBD Project
  - Accelerating Data Science Applications with Dask

- **Optimizations and Deployments in Public Cloud**
  - AWS, Azure, and Oracle

- Commercial Support and Value-Added Products

- Conclusions
MVAPICH2-Azure Deployment

- Integrated Azure CentOS HPC Images
  
  https://github.com/Azure/azhpc-images/releases/tag/centos-hpc-20210525

- MVAPICH2 2.3.6 is included
AMD ND A100 v4-series (NDv4) + (8 x HDR 200) - Azure

Intra-Node GPU Point-to-Point

Latency

Bandwidth

Inter-Node GPU Point-to-Point

Latency

Bandwidth

ND A100 v4-series: https://docs.microsoft.com/en-us/azure/virtual-machines/nda100-v4-series

AMD EPYC 7V12 64-Core Processor
CUDA 11.3, NVIDIA A100 GPUs
Mellanox ConnectX-6 HDR HCA
MVAPICH2-X on AWS EFA Arm HPC Instances

- Collective Performance on 32 AWS c6gn.18xlarge instances
WRF Application Results on AWS (MVAPICH2-X 2.3 with native EFA Design)

- Performance of WRF with Open MPI 4.0.3 vs Intel MPI 2019.7.217 vs MVAPICH2-X-AWS v2.3
- Run on c5n.18xlarge instances with libfabric 1.10

![WRF Execution Time Graph]

- Time (s) vs Number of processes
- Time percentages for different numbers of processes:
  - 36 processes: 17% for OpenMPI, 24% for Intel MPI, 41% for MVAPICH2-X-AWS
  - 72 processes: 27% for OpenMPI, 47% for Intel MPI, 58% for MVAPICH2-X-AWS
MVAPICH2-X Performance on OCI HPC System

- Collective performance evaluation on 8 BM.HPC2 instances

**Bcast – 36-ppn**

- MVAPICH2-X
- HPCx
- IntelMPI

**Reduce – 36-ppn**

- MVAPICH2-X
- HPCx
- IntelMPI

- 2.7x better
- 21x better
Presentation Overview

• MVAPICH Project
  – MPI and PGAS (MVAPICH) Library with CUDA-Awareness
  – Accelerating applications with DPU

• HiDL Project
  – High-Performance Deep Learning
  – High-Performance Machine Learning

• HiBD Project
  – Accelerating Data Science Applications with Dask

• Optimizations and Deployments in Public Cloud
  – AWS, Azure, and Oracle

• Commercial Support and Value-Added Products

• Conclusions
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 being provided to National Laboratories and International Supercomputing Centers
Value-Added Products with Support

- Multiple value-added products with support
  - X-ScaleHPC
  - X-ScaleAI
  - MVAPICH2-DPU
X-ScaleHPC Package

- Scalable solutions of communication middleware based on OSU MVAPICH2 libraries
- "out-of-the-box" fine-tuned and optimal performance on various HPC systems including x86, OpenPOWER, and ARM platforms and GPUs (NVIDIA and AMD)
- Contact us for more details and a free trial!!
  - contactus@x-scalesolutions.com
**X-ScaleAI Product**

- High-performance solution for distributed training for your complex AI problems
- Features:
  - Integrated package with TensorFlow, PyTorch, MXNet, Horovod, and MVAPICH2 MPI libraries
  - Targeted for both CPU-based and GPU-based Deep Learning Training
  - Integrated profiling and introspection support for Deep Learning Applications across the stacks (DeepIntrospect)
  - Support for OpenPOWER and x86 platforms
  - Support for InfiniBand, RoCE and NVLink Interconnects
  - Out-of-the-box optimal performance
  - One-click deployment and execution
- Send an e-mail to contactus@x-scalesolutions.com for free trial!!
X-ScaleAI Product with DeepIntrospect (DI) Capability

More capabilities and features are coming ...
Concluding Remarks

• Upcoming Exascale systems and Cloud need to be designed with a holistic view of HPC, Deep/Machine Learning, and Data Science

• Presented an overview of opportunities and challenges in designing scalable and high-performance middleware for such systems

• Presented a set of solutions which enable these communities to take advantage of current and next-generation systems with latest technologies

• Next-generation Exascale and Zetascale systems will need continuous innovations in designing converged software architectures .....
Funding Acknowledgments

**Funding Support by**

[Logos of various companies]

**Equipment Support by**

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

**Current Students (Graduate)**
- N. Alnaasan (Ph.D.)
- Q. Anthony (Ph.D.)
- C.-C. Chun (Ph.D.)
- N. Contini (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Khorassani (Ph.D.)
- A. H. Tu (Ph.D.)

**Current Students (Undergrads)**
- M. Lieber
- L. Xu

**Current Software Engineers**
- B. Seeds
- N. Shineman

**Current Research Scientists**
- A. Shafi
- H. Subramoni

**Current Research Scientists**
- A. H. Tu (Ph.D.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)
- K. Al Attar (M.S.)
- N. Sarkauskas (M.S.)

**Past Students**
- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- M. Bayatpour (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- A. Bhat (M.S.)
- T. Gangadharappa (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. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (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.)
- R. Rajachandrasekar (Ph.D.)

**Past Students**
- D. Shankar (Ph.D.)
- G. Santhanaranaman (Ph.D.)
- N. Sarkauskas (B.S.)
- N. Senthil Kumar (M.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Srivastava (M.S.)
- S. Subramoni (M.S.)
- H. Subramoni (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)
- J. Zhang (Ph.D.)

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

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

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

**Past Research Scientists**
- A. Reifsteck
- D. Bureddy
- J. Perkins

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

**Past Research Specialist**
- M. Arnold
- J. Smith
Thank You!

panda@cse.ohio-state.edu

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

Follow us on
https://twitter.com/mvapich

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/