Designing HPC, Deep Learning, and Cloud Middleware for Exascale Systems

Keynote Talk at HPCAC Stanford Conference (Feb ‘18)

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): Towards Exascale

Expected to have an ExaFlop system in 2019-2020!
Increasing Usage of HPC, Big Data and Deep Learning

Convergence of HPC, Big Data, and Deep Learning!

HPC
(MPI, RDMA, Lustre, etc.)

Big Data
(Hadoop, Spark, HBase, Memcached, etc.)

Deep Learning
(Caffe, TensorFlow, BigDL, etc.)

Increasing Need to Run these applications on the Cloud!!
Focus of Today’s Talk: HPC, Deep Learning, and Cloud

- Traditional HPC
  - Message Passing Interface (MPI), including MPI + OpenMP
  - Support for PGAS and MPI + PGAS (OpenSHMEM, UPC)
  - Exploiting Accelerators

- Deep Learning
  - Caffe and CNTK

- Cloud for HPC
  - Virtualization with SR-IOV and Containers

Focus of Tomorrow’s Talk: HPC + Big Data, Deep Learning, and Cloud
Focus of Tomorrow’s Talk: HPC, BigData, Deep Learning and Cloud

• Big Data/Enterprise/Commercial Computing
  – Spark and Hadoop (HDFS, HBase, MapReduce)
  – Memcached for Web 2.0
  – Kafka for Stream Processing
  – Swift

• Deep Learning
  – TensorFlow with gRPC

• Cloud for Big Data Stacks
  – Virtualization with SR-IOV and Containers
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, ...
Partitioned Global Address Space (PGAS) Models

- Key features
  - Simple shared memory abstractions
  - Light weight one-sided communication
  - Easier to express irregular communication

- Different approaches to PGAS
  - Languages
    - Unified Parallel C (UPC)
    - Co-Array Fortran (CAF)
    - X10
    - Chapel
  - Libraries
    - OpenSHMEM
    - UPC++
    - Global Arrays
Hybrid (MPI+PGAS) Programming

• Application sub-kernels can be re-written in MPI/PGAS based on communication characteristics

• Benefits:
  – Best of Distributed Computing Model
  – Best of Shared Memory Computing Model
Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges

Application Kernels/Applications

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

Multi-/Many-core Architectures

Accelerators (GPU and FPGA)
Broad Challenges in Designing Communication Middleware 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
- 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 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, 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-3.0), 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 2,875 organizations in 85 countries
  - More than 445,000 (> 0.4 million) downloads from the OSU site directly
  - Empowering many TOP500 clusters (Nov ‘17 ranking)
    - 1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China
    - 12th, 368,928-core (Stampede2) at TACC
    - 17th, 241,108-core (Pleiades) at NASA
    - 48th, 76,032-core (Tsubame 2.5) at Tokyo Institute of Technology
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
    - http://mvapich.cse.ohio-state.edu
- Empowering Top500 systems for over a decade
  - System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops) ->
  - Sunway TaihuLight (1st in Jun’17, 10M cores, 100 PFlops)
MVAPICH2 Release Timeline and Downloads

Network Based Computing Laboratory

HPCAC-Stanford (Feb ’18)

MVAPICH2 Release Timeline and Downloads

Network Based Computing Laboratory

HPCAC-Stanford (Feb ’18)
## Architecture of MVAPICH2 Software Family

### 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**
  - 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 2.3rc1

- Released on 02/19/2018
- Major Features and Enhancements
  - Enhanced performance for Allreduce, Reduce_scatter_block, Allgather, Allgatherv through new algorithms
  - Enhance support for MPI_T PVARs and CVARs
  - Improved job startup time for OFA-IB-CH3, PSM-CH3, and PSM2-CH3
  - Support to automatically detect IP address of IB/RoCE interfaces when RDMA_CM is enabled without relying on mv2.conf file
  - Enhance HCA detection to handle cases where node has both IB and RoCE HCAs
  - Automatically detect and use maximum supported MTU by the HCA
  - Added logic to detect heterogeneous CPU/HFI configurations in PSM-CH3 and PSM2-CH3 channels
  - Enhanced intra-node and inter-node tuning for PSM-CH3 and PSM2-CH3 channels
  - Enhanced HFI selection logic for systems with multiple Omni-Path HFIs
  - Enhanced tuning and architecture detection for OpenPOWER, Intel Skylake and Cavium ARM (ThunderX) systems
  - Added ‘SPREAD’, ‘BUNCH’, and ‘SCATTER’ binding options for hybrid CPU binding policy
  - Rename MV2_THREADS_BINDING_POLICY to MV2_HYBRID_BINDING_POLICY
  - Added support for MV2_SHOW_CPU_BINDING to display number of OMP threads
  - Update to hwloc version 1.11.9
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
  - Support for highly-efficient communication
  - Scalable Start-up
  - Optimized Collectives using SHArP and Multi-Leaders
  - Optimized CMA-based Collectives
  - Optimized XPMEM-based Collectives
  - Dynamic and Adaptive Communication and Tag Matching
  - Performance Engineering with MPI-T Support
  - Integrated Network Analysis and Monitoring

- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, ...)
- Integrated Support for GPGPUs
- Optimized MVAPICH2 for OpenPower and ARM
- Application Scalability and Best Practices
One-way Latency: MPI over IB with MVAPICH2

**Small Message Latency**

- Latency (us) vs Message Size (bytes)

- Key Points:
  - 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**

- Latency (us) vs Message Size (bytes)

- Key Points:
  - 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
Bandwidth: MPI over IB with MVAPICH2

**Unidirectional 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-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 IB switch
- **Omni-Path** - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch

**Message Size (bytes)**

- 4
- 16
- 64
- 256
- 1K
- 4K
- 64K
- 256K
- 1M

- **Bandwidth (MBytes/sec)**
  - 0
  - 2000
  - 4000
  - 6000
  - 8000
  - 10000
  - 12000
  - 14000

**Bidirectional Bandwidth**

- **TrueScale-QDR**
- **ConnectX-3-FDR**
- **ConnectIB-DualFDR**
- **ConnectX-5-EDR**
- **Omni-Path**

**Message Size (bytes)**

- 4
- 16
- 64
- 256
- 1K
- 4K
- 64K
- 256K
- 1M

- **Bandwidth (MBytes/sec)**
  - 0
  - 5000
  - 10000
  - 15000
  - 20000
  - 25000
Startup 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, 16, and 17
Advanced Allreduce Collective Designs Using SHArP

**Avg DDOT Allreduce time of HPCG**

- MVAPICH2
- MVAPICH2-SHArP

**Mesh Refinement Time of MiniAMR**

- MVAPICH2
- MVAPICH2-SHArP

SHArP Support is available since MVAPICH2 2.3a

Performance of MPI_Allreduce On Stampede2 (10,240 Processes)

- For MPI_Allreduce latency with 32K bytes, MVAPICH2-OPT can reduce the latency by 2.4X


Available in 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 in 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

**OSU_Allreduce (Broadwell 256 procs)**

- MVAPICH2-2.3b
- IMPI-2017v1.132
- MVAPICH2-Opt

**OSU_Reduce (Broadwell 256 procs)**

- MVAPICH2-2.3b
- IMPI-2017v1.132
- MVAPICH2-Opt

**Message Size**
- 16K
- 32K
- 64K
- 128K
- 256K
- 512K
- 1M
- 2M
- 4M

**Latency (us)**
- 16K: 73.2
- 32K: 36.1
- 64K: 16.8
- 128K: 37.9
- 256K: 1.8X
- 512K: 4X

*J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. Panda, Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores, International Parallel & Distributed Processing Symposium (IPDPS '18), May 2018.*

Will be available in future
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

MiniAMR (Broadwell, ppn=16)

- Up to 27% benefits over IMPI
- Up to 15% improvement over MVAPICH2
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>Wall Clock Time (seconds)</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Threshold=128K</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Threshold=64K</td>
<td>100</td>
<td>100</td>
<td>50</td>
<td>25</td>
</tr>
<tr>
<td>Dynamic Threshold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Relative Memory Consumption of Amber

<table>
<thead>
<tr>
<th>Relative Memory Consumption</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Threshold=128K</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Threshold=64K</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Dynamic Threshold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


Network Based Computing Laboratory

HPCAC-Stanford (Feb ‘18)
Dynamic and Adaptive Tag Matching

Challenge
Tag matching is a significant overhead for receivers
Existing Solutions are
- Static and do not adapt dynamically to communication pattern
- Do not consider memory overhead

Solution
A new tag matching design
- Dynamically adapt to communication patterns
- Use different strategies for different ranks
- Decisions are based on the number of request object that must be traversed before hitting on the required one

Results
Better performance than other state-of-the-art tag-matching schemes
Minimum memory consumption
Will be available in future MVAPICH2 releases

 normalized total tag matching time at 512 processes normalized to default (lower is better)
 normalized memory overhead per process at 512 processes compared to default (lower is better)

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

VBUF usage without CVAR based tuning as displayed by ParaProf

VBUF usage with CVAR based tuning as displayed by ParaProf

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, Euro MPI, 2017 [Best Paper Award]
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
  - [http://mvapich.cse.ohio-state.edu/tools/osu-inam/](http://mvapich.cse.ohio-state.edu/tools/osu-inam/)
- Monitors IB clusters in real time by querying various subnet management entities and gathering input from the MPI runtimes
- OSU INAM v0.9.2 released on 10/31/2017
- Significant enhancements to user interface to enable scaling to clusters with thousands of nodes
- Improve database insert times by using 'bulk inserts'
- Capability to look up list of nodes communicating through a network link
- Capability to classify data flowing over a network link at job level and process level granularity in conjunction with MVAPICH2-X 2.3b
- "Best practices " guidelines for deploying OSU INAM on different clusters
- 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 Features

Comet@SDSC --- Clustered View

Finding Routes Between Nodes

(1,879 nodes, 212 switches, 4,377 network links)

- 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
OSU INAM Features (Cont.)

Visualizing a Job (5 Nodes)

- 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

- Estimated Link Utilization view
  - Classify data flowing over a network link at different granularity in conjunction with MVAPICH2-X 2.2rc1
    - Job level and
    - Process level
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, ...)
- Integrated Support for GPGPUs
- Optimized MVAPICH2 for OpenPower and ARM
- Application Scalability and Best Practices
## MVAPICH2-X for Hybrid MPI + PGAS Applications

### High Performance Parallel Programming Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td>Message Passing Interface</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 Unified Communication Runtime

<table>
<thead>
<tr>
<th>Diverse APIs and Mechanisms</th>
<th>Support for Modern Networking Technologies</th>
<th>Support for Modern Multi-/Many-core Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized Point-to-point Primitives</td>
<td>(InfiniBand, IWARP, RoCE, Omni-Path...)</td>
<td>(Intel-Xeon, OpenPower...)</td>
</tr>
<tr>
<td>Remote Memory Access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active Messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collectives Algorithms (Blocking and Non-Blocking)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalable Job Startup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introspection &amp; Analysis with OSU INAM</td>
<td></td>
<td></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)
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

Optimized OpenSHMEM with AVX and MCDRAM: Application Kernels Evaluation

Heat-2D Kernel using Jacobi method

- On heat diffusion based kernels AVX-512 vectorization showed better performance
- MCDRAM showed significant benefits on Heat-Image kernel for all process counts. Combined with AVX-512 vectorization, it showed up to 4X improved performance
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, ...)
- Integrated Support for GPGPUs
  - CUDA-aware MPI
  - Optimized GPUDirect RDMA (GDR) Support
  - Support for Streaming Applications
- Optimized MVAPICH2 for OpenPower and ARM
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:**

```c
MPI_Send(s_devbuf, size, ...);
```

**At Receiver:**

```c
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**

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

**GPU-GPU Inter-node Bandwidth**

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

---

**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
Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland

Wilkes GPU Cluster

- Default
- Callback-based
- Event-based

CSCS GPU cluster

- Default
- Callback-based
- Event-based

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

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


Cosmo model: [http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/](http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/)
High-Performance Heterogeneous Broadcast for Streaming Applications

• Streaming applications on GPU clusters
  – Using a pipeline of broadcast operations to move host-resident data from a single source—typically live—to multiple GPU-based computing sites
  – Existing schemes require explicitly data movements between Host and GPU memories
    ➔ Poor performance and breaking the pipeline
• IB hardware multicast + Scatter-List
  – Efficient heterogeneous-buffer broadcast operation
• CUDA Inter-Process Communication (IPC)
  – Efficient intra-node topology-aware broadcast operations for multi-GPU systems
• Available MVAPICH2-GDR 2.3a!

Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, …)
- Integrated Support for GPGPUs
- Optimized MVAPICH2 for OpenPower and ARM
- Application Scalability and Best Practices
Inter-node Point-to-Point Performance on OpenPower

**Small Message Latency**

- MVAPICH2

**Large Message Latency**

- MVAPICH2

**Bandwidth**

- MVAPICH2

**Bi-directional Bandwidth**

- MVAPICH2

*Platform: Two nodes of OpenPOWER (Power8-ppc64le) CPU using Mellanox EDR (MT4115) HCA.*
Intra-node Point-to-point Performance on ARMv8

**Small Message Latency**

- MVAPICH2

**Large Message Latency**

- MVAPICH2

**Bandwidth**

- MVAPICH2

**Bi-directional Bandwidth**

- MVAPICH2

**Platform:** ARMv8 (aarch64) MIPS processor with 96 cores dual-socket CPU. Each socket contains 48 cores.
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: 14.6 us (without GPUDirectRDMA)**

**INTRA-NODE LATENCY (LARGE)**

- **Latency (us)** vs **Message Size (Bytes)**
- **INTRA-SOCKET** (NVLINK) vs **INTER-SOCKET**

**Intra-node Latency: 14.6 us (without GPUDirectRDMA)**

**INTER-NODE LATENCY (SMALL)**

- **Latency (us)** vs **Message Size (Bytes)**
- **INTRA-SOCKET** (NVLINK) vs **INTER-SOCKET**

**Inter-node Latency: 23.8 us (without GPUDirectRDMA)**

**INTER-NODE LATENCY (LARGE)**

- **Latency (us)** vs **Message Size (Bytes)**
- **INTRA-SOCKET** (NVLINK) vs **INTER-SOCKET**

**Inter-node Latency: 23.8 us (without GPUDirectRDMA)**

**INTRA-NODE BANDWIDTH**

- **Bandwidth (GB/sec)** vs **Message Size (Bytes)**
- **INTRA-SOCKET** (NVLINK) vs **INTER-SOCKET**

**Intra-node Bandwidth: 33.9 GB/sec (NVLINK)**

**INTER-NODE BANDWIDTH**

- **Bandwidth (GB/sec)** vs **Message Size (Bytes)**
- **INTRA-SOCKET** (NVLINK) vs **INTER-SOCKET**

**Inter-node Bandwidth: 11.9 GB/sec (EDR)**

---

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

Available in MVAPICH2-GDR 2.3a
Overview of A Few Challenges being Addressed by the MVAPICH2 Project for Exascale

- Scalability for million to billion processors
- Unified Runtime for Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, CAF, UPC++, ...)
- Integrated Support for GPGPUs
- Optimized MVAPICH2 for OpenPower and ARM
- Application Scalability and Best Practices
Application Scalability on Skylake and KNL

**MiniFE** (1300x1300x1300 ~ 910 GB)

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL: 64ppn</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>2048</td>
<td>0</td>
</tr>
<tr>
<td>4096</td>
<td>50</td>
</tr>
<tr>
<td>8192</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake: 48ppn</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>2048</td>
<td>0</td>
</tr>
<tr>
<td>4096</td>
<td>50</td>
</tr>
<tr>
<td>8192</td>
<td>100</td>
</tr>
</tbody>
</table>

**NEURON** (YuEtAl2012)

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL: 64ppn</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td>50</td>
</tr>
<tr>
<td>256</td>
<td>100</td>
</tr>
<tr>
<td>512</td>
<td>150</td>
</tr>
<tr>
<td>1024</td>
<td>200</td>
</tr>
<tr>
<td>2048</td>
<td>250</td>
</tr>
<tr>
<td>4096</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake: 48ppn</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>192</td>
<td>100</td>
</tr>
<tr>
<td>384</td>
<td>150</td>
</tr>
<tr>
<td>768</td>
<td>200</td>
</tr>
</tbody>
</table>

**Cloverleaf** (bm64) MPI+OpenMP, NUM_OMP_THREADS = 2

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNL: 68ppn</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td>50</td>
</tr>
<tr>
<td>256</td>
<td>100</td>
</tr>
<tr>
<td>512</td>
<td>150</td>
</tr>
<tr>
<td>1024</td>
<td>200</td>
</tr>
<tr>
<td>2048</td>
<td>250</td>
</tr>
<tr>
<td>4096</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake: 48ppn</td>
<td>MVAPICH2</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>192</td>
<td>100</td>
</tr>
<tr>
<td>384</td>
<td>150</td>
</tr>
<tr>
<td>768</td>
<td>200</td>
</tr>
<tr>
<td>1536</td>
<td>250</td>
</tr>
<tr>
<td>3072</td>
<td>300</td>
</tr>
</tbody>
</table>

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

**Testbed:** TACC Stampede2 using MVAPICH2-2.3b

**Courtesy:** Mahidhar Tatineni @SDSC, Dong Ju (DJ) Choi@SDSC, and Samuel Khuvis@OSC

Network Based Computing Laboratory

HPCAC-Stanford (Feb '18)
Performance of SPEC MPI 2007 Benchmarks (KNL + Omni-Path)

Mvapich2 outperforms Intel MPI by up to 10%

Intel MPI 18.0.0

MVAPICH2 2.3 rc1

448 processes on 7 KNL nodes of TACC Stampede2 (64 ppn)
Performance of SPEC MPI 2007 Benchmarks (Skylake + Omni-Path)

MVAPICH2 outperforms Intel MPI by up to 38%
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
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.
HPC, Big Data, Deep Learning, and Cloud

• Traditional HPC
  – Message Passing Interface (MPI), including MPI + OpenMP
  – Support for PGAS and MPI + PGAS (OpenSHMEM, UPC)
  – Exploiting Accelerators

• Deep Learning
  – Caffe, CNTK, TensorFlow, and many more

• Cloud for HPC
  – Virtualization with SR-IOV and Containers
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
  - 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?

Efficient Broadcast: MVAPICH2-GDR and NCCL

1. NCCL 1.x had some limitations
   - Only worked for a single node; no scale-out on multiple nodes
   - Degradation across IOH (socket) for scale-up (within a node)
2. We propose optimized MPI_Bcast design that exploits NCCL [1]
   - Communication of very large GPU buffers
   - Scale-out on large number of dense multi-GPU nodes
3. Hierarchical Communication that efficiently exploits:
   - CUDA-Aware MPI_Bcast in MV2-GDR
   - NCCL Broadcast for intra-node transfers
4. Can pure MPI-level designs be done that achieve similar or better performance than NCCL-based approach? [2]

Large Message Optimized Collectives for Deep Learning

- MV2-GDR provides optimized collectives for large message sizes
- Optimized Reduce, Allreduce, and Bcast
- Good scaling with large number of GPUs
- Available since MVAPICH2-GDR 2.2GA
MVAPICH2-GDR vs. Baidu-allreduce

- Initial Evaluation shows promising performance gains for MVAPICH2-GDR 2.3a*
- 8 GPUs (4 nodes) MPI_Allreduce (MVAPICH2-GDR) vs. Baidu-Allreduce

*Available with MVAPICH2-GDR 2.3a
OSU-Caffe: Scalable Deep Learning

• Caffe: A flexible and layered Deep Learning framework.

• Benefits and Weaknesses
  – Multi-GPU Training within a single node
  – Performance degradation for GPUs across different sockets
  – Limited Scale-out

• OSU-Caffe: MPI-based Parallel Training
  – Enable Scale-up (within a node) and Scale-out (across multi-GPU nodes)
  – Scale-out on 64 GPUs for training CIFAR-10 network on CIFAR-10 dataset
  – Scale-out on 128 GPUs for training GoogLeNet network on ImageNet dataset

OSU-Caffe publicly available from
http://hidl.cse.ohio-state.edu/
HPC, Big Data, Deep Learning, and Cloud

• Traditional HPC
  – Message Passing Interface (MPI), including MPI + OpenMP
  – Support for PGAS and MPI + PGAS (OpenSHMEM, UPC)
  – Exploiting Accelerators

• Deep Learning
  – Caffe, CNTK, TensorFlow, and many more

• Cloud for HPC
  – Virtualization with SR-IOV and Containers
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 Libray 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
• 64 Containers across 16 nodes, pining 4 Cores per Container
• Compared to Container-Def, up to 11% and 73% of execution time reduction for NAS and Graph 500
• Compared to Native, less than 9% and 5% overhead for NAS and Graph 500
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
High Performance SR-IOV enabled VM Migration Framework

- Consist of SR-IOV enabled IB Cluster and External Migration Controller
- **Detachment/Re-attachment of virtualized devices:** Multiple parallel libraries to coordinate VM during migration (detach/reattach SR-IOV/IVShmem, migrate VMs, migration status)
- **IB Connection:** MPI runtime handles IB connection suspending and reactivating
- Propose Progress Engine (**PE**) and Migration Thread based (**MT**) design to optimize VM migration and MPI application performance

Performance Evaluation of VM Migration Framework

Pt2Pt Latency

- Migrate a VM from one machine to another while benchmark is running inside
- Proposed MT-based designs perform slightly worse than PE-based designs because of lock/unlock
- No benefit from MT because of NO computation involved

Bcast (4VMs * 2Procs/VM)

Will be available in upcoming MVAPICH2-Virt
Concluding Remarks

- Next generation HPC systems need to be designed with a holistic view of Deep Learning and Cloud
- Presented some of the approaches and results along these directions
- Enable Deep Learning and Cloud community to take advantage of modern HPC technologies
- Many other open issues need to be solved
Funding Acknowledgments

Funding Support by

[Logos of various sponsors]

Equipment Support by

[Logos of various sponsors]
# Personnel Acknowledgments

## Current Students (Graduate)
- A. Awan (Ph.D.)
- R. Biswas (M.S.)
- M. Bayatpour (Ph.D.)
- S. Chakraborthy (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)
- J. Hashmi (Ph.D.)
- H. Javed (Ph.D.)
- P. Kousha (Ph.D.)
- D. Shankar (Ph.D.)
- H. Shi (Ph.D.)
- J. Zhang (Ph.D.)

## Current Students (Undergraduate)
- N. Sarkauskas (B.S.)

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

## Current Post-doc
- A. Ruhela

## Past Students
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- 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. Gangadhara (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- 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.)
- W. 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.)

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

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

## Past Post-Docs
- D. Banerjee
- X. Besseron
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne
- H. Wang
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- 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.)
Thank You!

panda@cse.ohio-state.edu

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

MVAPICH
The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

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

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