Advances in MPI and PGAS Programming Models

Talk at  HPC Advisory Council, CHPC (Dec 2014)

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda
Trends for Commodity Computing Clusters in the Top 500 List (http://www.top500.org)
Drivers of Modern HPC Cluster Architectures

- Multi-core processors are ubiquitous
- InfiniBand very popular in HPC clusters
- Accelerators/Coprocessors becoming common in high-end systems
- Pushing the envelope for Exascale computing
Large-scale InfiniBand Installations

- 225 IB Clusters (45%) in the November 2014 Top500 list ([http://www.top500.org](http://www.top500.org))

- Installations in the Top 50 (21 systems):

<table>
<thead>
<tr>
<th>Large-scale InfiniBand Installations</th>
<th>Cores</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>462,462 cores (Stampede) at TACC (7th)</td>
<td>462,462</td>
<td>Stampede at TACC (7th)</td>
</tr>
<tr>
<td>72,800 cores Cray CS-Storm in US (10th)</td>
<td>72,800</td>
<td>Cray CS-Storm in US (10th)</td>
</tr>
<tr>
<td>160,768 cores (Pleiades) at NASA/Ames (11th)</td>
<td>160,768</td>
<td>Pleiades at NASA/Ames (11th)</td>
</tr>
<tr>
<td>72,000 cores (HPC2) in Italy (12th)</td>
<td>72,000</td>
<td>HPC2 in Italy (12th)</td>
</tr>
<tr>
<td>147,456 cores (Super MUC) in Germany (14th)</td>
<td>147,456</td>
<td>Super MUC in Germany (14th)</td>
</tr>
<tr>
<td>76,032 cores (Tsubame 2.5) at Japan/GSIC (15th)</td>
<td>76,032</td>
<td>Tsubame 2.5 at Japan/GSIC (15th)</td>
</tr>
<tr>
<td>194,616 cores (Cascade) at PNNL (18th)</td>
<td>194,616</td>
<td>Cascade at PNNL (18th)</td>
</tr>
<tr>
<td>110,400 cores (Pangea) at France/Total (20th)</td>
<td>110,400</td>
<td>Pangea at France/Total (20th)</td>
</tr>
<tr>
<td>37,120 cores T-Platform A-Class at Russia/MSU (22nd)</td>
<td>37,120</td>
<td>T-Platform A-Class at Russia/MSU (22nd)</td>
</tr>
<tr>
<td>50,544 cores Occigen at France/GENCI-CINES (26th)</td>
<td>50,544</td>
<td>Occigen at France/GENCI-CINES (26th)</td>
</tr>
</tbody>
</table>

- and many more!
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.

**Shared Memory Model**
- SHMEM, DSM

**Distributed Memory Model**
- MPI (Message Passing Interface)

**Partitioned Global Address Space (PGAS)**
- Global Arrays, UPC, Chapel, X10, CAF, ...
MPI Overview and History

• Message Passing Library standardized by MPI Forum
  – C and Fortran

• Goal: portable, efficient and flexible standard for writing parallel applications

• Not IEEE or ISO standard, but widely considered “industry standard” for HPC application

• Evolution of MPI
  – MPI-1: 1994
  – MPI-2: 1996
  – Next plans for MPI 3.1 and 4.0
Major MPI Features

• Point-to-point Two-sided Communication
• Collective Communication
• One-sided Communication
• Job Startup
• Parallel I/O
Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- Plans for MPI 4.0
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS Features and Solutions
Major New Features in MPI-3

• Major Features
  – Non-blocking Collectives
  – Improved One-Sided (RMA) Model
  – MPI Tools Interface

• Specification is available from: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf
Problems with Blocking Collective Operations

- Communication time cannot be used for compute
  - No overlap of computation and communication
  - Inefficient
Concept of Non-blocking Collectives

- Application processes schedule collective operation
- Check periodically if operation is complete
- **Overlap of computation and communication => Better Performance**
- **Catch: Who will progress communication**
Non-blocking Collective (NBC) Operations

- Enables overlap of computation with communication
- Non-blocking calls do not match blocking collective calls
  - MPI may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: Optimized for latency
  - Non-blocking collectives: Optimized for overlap

- A process calling an NBC operation
  - Schedules collective operation and immediately returns
  - Executes application computation code
  - Waits for the end of the collective

- The communication progress by
  - Application code through MPI_Test
  - Network adapter (HCA) with hardware support
  - Dedicated processes / thread in MPI library

- There is a non-blocking equivalent for each blocking operation
  - Has an “I” in the name
    - MPI_Bcast -> MPI_Ibcast; MPI_Reduce -> MPI_Ireduce
MPI-3 Features

- Non-blocking Collectives
- Improved One-Sided (RMA) Model
- MPI Tools Interface
One-sided Communication Model

Global Region Creation

(Buffer Info Exchanged)

Write to P2

Write Data from P1

Write to P3

Post to HCA

Post to HCA

Write data from P2
Improved One-sided (RMA) Model

- New RMA proposal has major improvements
- Easy to express irregular communication pattern
- Better overlap of communication & computation
- MPI-2: public and private windows
  - Synchronization of windows explicit
- MPI-2: works for non-cache coherent systems
- MPI-3: two types of windows
  - Unified and Separate
  - Unified window leverages hardware cache coherence
MPI-3 One-Sided Primitives

- Non-blocking one-sided communication routines
  - Put, Get
  - Accumulate, Get_accumulate
  - Atomics

- Flexible synchronization operations to control initiation and completion
Overlapping Communication with MPI-3-RMA

- Network adapters can provide RDMA feature that doesn’t require software involvement at remote side
- As long as puts/gets are executed as soon as they are issued, overlap can be achieved
- RDMA-based implementations do just that
MPI-3 Features

- Non-blocking Collectives
- Improved One-Sided (RMA) Model
- MPI Tools Interface
MPI Tools Interface

- Introduced to expose internals of MPI tools and applications
- Generalized interface – no defined variables in the standard
- Variables can differ between
  - MPI implementations
  - Compilations of same MPI library (production vs debug)
  - Executions of the same application/MPI library
  - There could be no variables provided
- Two types of variables supported
  - **Control Variables (CVARS)**
    - Typically used to configure and tune MPI internals
    - Environment variables, configuration parameters and toggles
  - **Performance Variables (PVARS)**
    - Insights into performance of an MPI library
    - Highly-implementation specific
    - Memory consumption, timing information, resource-usage, data transmission info.
    - Per-call basis or an entire MPI job
Who should use MPI-T and How?

• Who???
  – Interface intended for tool developers
    • Generally will do *anything* to get the data
    • Are willing to support the many possible variations

• How???
  – Can be called from user code
  – Useful for setting control variables for performance
  – Documenting settings for understanding performance
  – Care must be taken to avoid code that is not portable
  – Several workflows based on role: End Users / Performance Tuners / MPI Implementers
    • Two main workflows
      – Transparently using MPIT-Aware external tools
      – Co-designing applications and MPI-libraries using MPI-T
Co-designing Applications to use MPI-T

Example: Optimizing the eager limit dynamically ->

MPI_T_init_thread()
MPI_T_cvar_get_info(MV2_EAGER_THRESHOLD)
if (msg_size < MV2_EAGER_THRESHOLD + 1KB)
    MPI_T_cvar_write(MV2_EAGER_THRESHOLD, +1024)
MPI_Send(..)
MPI_T_finalize()
Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- Plans for MPI 4.0
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS

Features and Solutions
MPI-3.1 Enhancements (Primarily Errata)

- MPI_IN_PLACE in MPI_Get_accumulate
- MPI_Aint addressing arithmetic
- Thread safety of MPI_INITIALIZED, MPI_FINALIZED, MPI_QUERY_THREAD
- RMA Info key: mpi_shared_buffer
- Add function MPI_AINT_DIFF
Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- Plans for MPI 4.0
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS

Features and Solutions
MPI-4.0 Proposed Features

- Dynamic Endpoints
- User Level Failure Mitigation (ULFM)
- RMA Notifications
- Non-blocking Collective-I/O
- Large Datatype Support
Dynamic Endpoints

• Introduces a new communicator creation function
• Can be used to create additional ranks, or endpoints, at an existing MPI process
• Endpoints behave the same as processes and can be associated with threads, allowing threads to fully participate in MPI operation
User Level Failure Mitigation (ULFM)

- Allows the application react to failures but maintain a minimal code path for failure-free execution
- Processes can invalidate communication objects and prevent waiting indefinitely
- Failures do not alter state of MPI communicators
- Allows point-to-point communication to continue between non-faulty processes
- MPI_Comm_shrink is used to exclude failed processes to resume communication and spawn replacements
User Level Failure Mitigation (ULFM) Example Usage

Coordinated Checkpoint/Restart, Automatic, Compiler Assisted, User-driven Checkpointing, etc.
In-place restart (i.e., without disposing of non-failed processes) accelerates recovery, permits in-memory checkpoint

Naturally Fault Tolerant Applications, Master-Worker, Domain Decomposition, etc.
Application continues a simple communication pattern, ignoring failures

Uncoordinated Checkpoint/Restart, Transactional FT, Migration, Replication, etc.
ULFM makes these approaches portable across MPI implementations

Algorithm Fault Tolerance
ULFM allows for the deployment of ultra-scalable, algorithm specific FT techniques.

RMA Notifications

- In passive target mode, sender needs to send additional message to signal completion after data transfer
- Notification counter associated with the window is incremented at the target after each epoch
- Processes can query number of notifications received
Immediate Non-blocking I/O

- Split-collective I/O routines are limited as it supports at most one active operation at a time
- Proposed non-blocking I/O semantics use MPI_Request
- MPI_Test is used to check for completion

Very large datatype support

- Current datatype interface supports count up to INT_MAX
- Proposed function MPI_Type_contiguous_x allows the creation of very large contiguous datatypes
Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- Plans for MPI 4.0
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS

Features and Solutions
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
  - Libraries
    • OpenSHMEM
    • Global Arrays
    • Chapel
Compiler-based: Unified Parallel C

- UPC: a parallel extension to the C standard
- UPC Specifications and Standards:
  - Introduction to UPC and Language Specification, 1999
  - UPC Language Specifications, v1.0, Feb 2001
  - UPC Language Specifications, v1.1.1, Sep 2004
  - UPC Language Specifications, v1.2, June 2005
- UPC Consortium
  - Academic Institutions: GWU, MTU, UCB, U. Florida, U. Houston, U. Maryland...
  - Government Institutions: ARSC, IDA, LBNL, SNL, US DOE...
  - Commercial Institutions: HP, Cray, Intrepid Technology, IBM, ...
- Supported by several UPC compilers
  - Vendor-based commercial UPC compilers: HP UPC, Cray UPC, SGI UPC
  - Open-source UPC compilers: Berkeley UPC, GCC UPC, Michigan Tech MuPC
- Aims for: high performance, coding efficiency, irregular applications, ...
OpenSHMEM

- SHMEM implementations – Cray SHMEM, SGI SHMEM, Quadrics SHMEM, HP SHMEM, GSHMEM

- Subtle differences in API, across versions – example:

<table>
<thead>
<tr>
<th>SGI SHMEM</th>
<th>Quadrics SHMEM</th>
<th>Cray SHMEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td><code>start_pes(0)</code></td>
<td><code>shmem_init</code></td>
</tr>
<tr>
<td>Process ID</td>
<td><code>_my_pe</code></td>
<td><code>my_pe</code></td>
</tr>
</tbody>
</table>

- Made applications codes non-portable

- OpenSHMEM is an effort to address this:

  “A new, open specification to consolidate the various extant SHMEM versions into a widely accepted standard.” – OpenSHMEM Specification v1.0

  by University of Houston and Oak Ridge National Lab

  SGI SHMEM is the baseline
Hierarchical architectures with multiple address spaces

(MPI + PGAS) Model
- MPI across address spaces
- PGAS within an address space

MPI is good at moving data between address spaces

Within an address space, MPI can interoperate with other shared memory programming models

Can co-exist with OpenMP for offloading computation

Applications can have kernels with different communication patterns

Can benefit from different models

Re-writing complete applications can be a huge effort

Port critical kernels to the desired model instead
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

• Exascale Roadmap*:
  – “Hybrid Programming is a practical way to program exascale systems”

Presentation Outline

- MPI 3.0 Features
- MPI 3.1 Features
- Plans for MPI 4.0
- Upcoming Trends for PGAS and Hybrid MPI+PGAS
- Challenges in Supporting MPI, PGAS and Hybrid MPI+PGAS

Features and Solutions
Designing MPI+X Libraries for Multi-Petaflop and Exaflop Systems: Challenges

Application Kernels/Applications

Middleware

Programming Models
MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenACC, Cilk, etc.

Communication Library or Runtime for Programming Models

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

Networking Technologies
(InfiniBand, 40/100GigE, Aries, and Omni Scale)

Multi/Many-core Architectures

Accelerators (NVIDIA and MIC)
Challenges in Designing (MPI+X) at Exascale

- Scalability for million to billion processors
  - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided RMA)
  - Extremely minimum memory footprint
- Balancing intra-node and inter-node communication for next generation multi-core (128-1024 cores/node)
  - Multiple end-points per node
- Support for efficient multi-threading
- Support for GPGPUs and Accelerators
- Scalable Collective communication
  - Offload
  - Non-blocking
  - Topology-aware
  - Power-aware
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming (MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, ...)
- MPI-Tools Interface
- Virtualization
MVAPICH2/MVAPICH2-X Software

• High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  – MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  – MVAPICH2-X (MPI + PGAS), Available since 2012
  – Support for GPGPUs and MIC
  – **Used by more than 2,250 organizations (HPC Centers, Industry and Universities) in 74 countries**
  – **More than 227,000 downloads from OSU site directly**
  – Empowering many TOP500 clusters
    • 7th ranked 519,640-core cluster (Stampede) at TACC
    • 11th ranked 160,768-core cluster (Pleiades) at NASA
    • 15th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  – Available with software stacks of many IB, HSE, and server vendors including Linux Distros (RedHat and SuSE)
    – [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)
• Partner in the U.S. NSF-TACC Stampede System
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-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 RMA)
  - Extremely minimum memory footprint
- Non-blocking Collectives
- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
- Virtualization
Latency & Bandwidth: MPI over IB with MVAPICH2

Small Message Latency

- Qlogic-DDR
- Qlogic-QDR
- ConnectX-DDR
- ConnectX2-PCIe2-QDR
- ConnectX3-PCIe3-FDR
- Sandy-ConnectIB-DualFDR
- Ivy-ConnectIB--DualFDR

Unidirectional Bandwidth

- DDR, QDR - 2.4 GHz Quad-core (Westmere) Intel PCI Gen2 with IB switch
- FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch
- ConnectIB-Dual FDR - 2.6 GHz Octa-core (SandyBridge) Intel PCI Gen3 with IB switch
- ConnectIB-Dual FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch
MVAPICH2 Two-Sided Intra-Node Performance
(Shared memory and Kernel-based Zero-copy Support (LiMIC and CMA))

Latest MVAPICH2 2.1a
Intel Ivy-bridge
MPI-3 RMA Get/Put with Flush Performance

Latest MVAPICH2 2.1a, Intel Sandy-bridge with Connect-IB (single-port)
Performance of AWP-ODC using MPI-3-RMA

- Experiments on TACC Ranger cluster 64x64x64 data grid per process – 25 iterations – 32KB messages
- On 4K processes
  - 8% with 2sided basic, 11% with 2sided advanced, 12% with RMA
- On 8K processes
  - 2% with 2sided basic, 6% with 2sided advanced, 10% with RMA

Minimizing memory footprint with XRC and Hybrid Mode

- Memory usage for 32K processes with 8-cores per node can be 54 MB/process (for connections)
- NAMD performance improves when there is frequent communication to many peers

Both UD and RC/XRC have benefits
- Hybrid for the best of both
- Available since MVAPICH2 1.7 as integrated interface
- Runtime Parameters: RC - default;
  - UD - MV2_USE_ONLY_UD=1
  - Hybrid - MV2_HYBRID_ENABLE_THRESHOLD=1

Dynamic Connected (DC) Transport in MVAPICH2

- 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

- Initial study done in MVAPICH2
- DCT support available in Mellanox OFED

Memory Footprint for Alltoall

NAMD - Apoa1: Large data set

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-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 RMA)
  – Extremely minimum memory footprint

• Non-blocking Collectives

• Support for GPGPUs
• Support for Intel MICs
• Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
• Virtualization
Adapter-level Support for Collective Offload

- Sender creates a task-list consisting of only send and wait WQEs
  - One send WQE is created for each registered receiver and is appended to the rear of a singly linked task-list
  - A wait WQE is added to make the HCA wait for ACK packet from the receiver
Application benefits with Non-Blocking Collectives based on CX-3 Collective Offload

Modified P3DFFT with Offload-Alltoall does up to 17% better than default version (128 Processes)

Modified Pre-Conjugate Gradient Solver with Offload-Allreduce does up to 21.8% better than default version

Modified HPL with Offload-Bcast does up to 4.5% better than default version (512 Processes)

K. Kandalla, et. al. High-Performance and Scalable Non-Blocking All-to-All with Collective Offload on InfiniBand Clusters: A Study with Parallel 3D FFT, ISC 2011

K. Kandalla, et. al, Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, HotI 2011

K. Kandalla, et. al., Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers, IPDPS ’12

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-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 RMA)
  - Extremely minimum memory footprint
- Non-blocking Collectives
- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
- Virtualization
MVAPICH2-GPU: CUDA-Aware MPI

• Before CUDA 4: Additional copies
  – Low performance and low productivity

• After CUDA 4: Host-based pipeline
  – Unified Virtual Address
  – Pipeline CUDA copies with IB transfers
  – High performance and high productivity

• After CUDA 5.5: GPUDirect-RDMA support
  – GPU to GPU direct transfer
  – Bypass the host memory
  – Hybrid design to avoid PCI bottlenecks
CUDA-Aware MPI: MVAPICH2 1.8-2.1 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
Performance of MVAPICH2 with GPU-Direct-RDMA: Latency

GPU-GPU Internode MPI Latency

Small Message Latency

- MV2-GDR2.0
- MV2-GDR2.0b
- MV2 w/o GDR

- Latency (us)
- Message Size (bytes)

MVAPICH2-GDR-2.0
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA

- 90 %
- 71 %
- 2.18 usec
Performance of MVAPICH2 with GPU-Direct-RDMA: Bandwidth

GPU-GPU Internode MPI Uni-Directional Bandwidth

Small Message Bandwidth

- MV2-GDR2.0
- MV2-GDR2.0b
- MV2 w/o GDR

Bandwidth (MB/s)

Message Size (bytes)

MV2-GDR2.0
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA
Performance of MVAPICH2 with GPU-Direct-RDMA: Bi-Bandwidth

GPU-GPU Internode MPI Bi-directional Bandwidth

Small Message Bi-Bandwidth

- MV2-GDR2.0
- MV2-GDR2.0b
- MV2 w/o GDR

Message Size (bytes)

MVAPICH2-GDR-2.0

Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA
Performance of MVAPICH2 with GPU-Direct-RDMA: MPI-3 RMA

GPU-GPU Internode MPI Put latency (RMA put operation Device to Device)

MPI-3 RMA provides flexible synchronization and completion primitives

![Small Message Latency Graph]

MVAPICH2-GDR-2.0
Intel Ivy Bridge (E5-2680 v2) node with 20 cores
NVIDIA Tesla K40c GPU, Mellanox Connect-IB Dual-FDR HCA
CUDA 6.5, Mellanox OFED 2.1 with GPU-Direct-RDMA
Application-Level Evaluation (HOOMD-blue)

- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- MV2-GDR 2.0 (released on 08/23/14) : try it out !!
  - 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
Using MVAPICH2-GPUDirect Version

- MVAPICH2-2.0 with GDR support can be downloaded from https://mvapich.cse.ohio-state.edu/download/mvapich2gdr/

- System software requirements
  - Mellanox OFED 2.1 or later
  - NVIDIA Driver 331.20 or later
  - NVIDIA CUDA Toolkit 6.0 or later
  - Plugin for GPUDirect RDMA
    - Strongly Recommended: use the new GDRCOPY module from NVIDIA
      - https://github.com/drossetti/gdrcopy
  - Has optimized designs for point-to-point communication using GDR
  - Contact MVAPICH help list with any questions related to the package
    mvapich-help@cse.ohio-state.edu
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-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 RMA)
  - Extremely minimum memory footprint
- Non-blocking Collectives
- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
- Virtualization
MPI Applications on MIC Clusters

• Flexibility in launching MPI jobs on clusters with Xeon Phi

Multi-core Centric

Host-only

Offload (/reverse Offload)

Symmetric

Coprocessor-only

Many-core Centric

Xeon

MPI Program

Xeon Phi

Offloaded Computation

MPI Program

MPI Program
MVAPICH2-MIC Design for Clusters with IB and MIC

- Offload Mode
- Intranode Communication
  - Coprocessor-only Mode
  - Symmetric Mode
- Internode Communication
  - Coprocessors-only
  - Symmetric Mode
- Multi-MIC Node Configurations
MIC-Remote-MIC P2P Communication

Intra-socket P2P

Latency (Large Messages)

<table>
<thead>
<tr>
<th>Message Size (Bytes)</th>
<th>8K</th>
<th>32K</th>
<th>128K</th>
<th>512K</th>
<th>2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (usec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bandwidth

<table>
<thead>
<tr>
<th>Message Size (Bytes)</th>
<th>1</th>
<th>16</th>
<th>256</th>
<th>4K</th>
<th>64K</th>
<th>1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth (MB/sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inter-socket P2P

Latency (Large Messages)

<table>
<thead>
<tr>
<th>Message Size (Bytes)</th>
<th>8K</th>
<th>32K</th>
<th>128K</th>
<th>512K</th>
<th>2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (usec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bandwidth

<table>
<thead>
<tr>
<th>Message Size (Bytes)</th>
<th>1</th>
<th>16</th>
<th>256</th>
<th>4K</th>
<th>64K</th>
<th>1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth (MB/sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Latency (usec)

<table>
<thead>
<tr>
<th>Message Size (Bytes)</th>
<th>8K</th>
<th>32K</th>
<th>128K</th>
<th>512K</th>
<th>2M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (usec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bandwidth

<table>
<thead>
<tr>
<th>Message Size (Bytes)</th>
<th>1</th>
<th>16</th>
<th>256</th>
<th>4K</th>
<th>64K</th>
<th>1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth (MB/sec)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Optimized MPI Collectives for MIC Clusters (Allgather & Alltoall)

32-Node-Allgather (16H + 16 M) Small Message Latency

- MV2-MIC
- MV2-MIC-Opt

32-Node-Allgather (8H + 8 M) Large Message Latency

- MV2-MIC
- MV2-MIC-Opt

32-Node-Alltoall (8H + 8 M) Large Message Latency

- MV2-MIC
- MV2-MIC-Opt

P3DFFT Performance

- Communication
- Computation

Latest Status on MVAPICH2-MIC

• Running on three major systems
  – Stampede : module swap mvapich2 mvapich2-mic/20130911
  – Blueridge (Virginia Tech) : module swap mvapich2 mvapich2-mic/1.9
  – Beacon (UTK) : module unload intel-mpi; module load mvapich2-mic/1.9

• Public version of MVAPICH2-MIC 2.0 is released (12/02/14)!!
Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-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 RMA)
  - Extremely minimum memory footprint
- Non-blocking Collectives
- Support for GPGPUs
- Support for Intel MICs
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
- Virtualization
MVAPICH2-X for Hybrid MPI + PGAS Applications

- Unified communication runtime for MPI, UPC, OpenSHMEM available with MVAPICH2-X 1.9 onwards!
  - [http://mvapich.cse.ohio-state.edu](http://mvapich.cse.ohio-state.edu)

- Feature Highlights
  - Supports MPI(+OpenMP), OpenSHMEM, UPC, MPI(+OpenMP) + OpenSHMEM, MPI(+OpenMP) + UPC
  - MPI-3 compliant, OpenSHMEM v1.0 standard compliant, UPC v1.2 standard compliant
  - Scalable Inter-node and Intra-node communication – point-to-point and collectives
OpenSHMEM Application Evaluation

**Heat Image**

- Improved performance for OMPI-SHMEM and Scalable-SHMEM with FCA
- Execution time for 2DHeat Image at 512 processes (sec):

**Daxpy**

- Execution time for DAXPY at 512 processes (sec):

Hybrid MPI+OpenSHMEM Graph500 Design

- 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

J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC’13), June 2013
J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012
Hybrid MPI+OpenSHMEM Sort Application

- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - Execution Time
    - 4TB Input size at 4,096 cores: MPI – 2408 seconds, Hybrid: 1172 seconds
    - 51% improvement over MPI-based design
  - Strong Scalability (configuration: constant input size of 500GB)
    - At 4,096 cores: MPI – 0.16 TB/min, Hybrid – 0.36 TB/min
    - 55% improvement over MPI based design

Overview of A Few Challenges being Addressed by MVAPICH2/MVAPICH2-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 RMA)
  - Extremely minimum memory footprint
- Non-blocking Collectives
- Support for GPGPUs
- Support for Intel MICs
- Support for MPI-T Interface
- Hybrid MPI+PGAS programming (MPI + OpenSHMEM, MPI + UPC, ...) with Unified Runtime
- Virtualization
Can HPC and Virtualization be Combined?

- Virtualization has many benefits
  - Job migration
  - Compaction
- Not very popular in HPC due to overhead associated with Virtualization
- New SR-IOV (Single Root – IO Virtualization) support available with Mellanox InfiniBand adapters
- Initial designs of MVAPICH2 with SR-IOV support with Openstack
- Will be available publicly soon
EC2 C3.xlarge instance

Compared to SR-IOV-Def, 84% and 158% performance improvement on Lat & BW

Compared to EC2, 219X and 26X performance improvement on Lat & BW

Compared to Native, 3%-7% overhead for Lat, 3%-8% overhead for BW
• EC2 C3.2xlarge instance
• Compared to EC2, up to 77% (FT) improvement for NAS, up to 12X (20,10) improvement for Graph500
• Compared to Native, 2%-8% overhead for NAS, around 6% overhead for Graph500
NSF Chameleon Cloud: A Powerful and Flexible Experimental Instrument

• Large-scale instrument
  – Targeting Big Data, Big Compute, Big Instrument research
  – ~650 nodes (~14,500 cores), 5 PB disk over two sites, 2 sites connected with 100G network

• Reconfigurable instrument
  – Bare metal reconfiguration, operated as single instrument, graduated approach for ease-of-use

• Connected instrument
  – Workload and Trace Archive
  – Partnerships with production clouds: CERN, OSDC, Rackspace, Google, and others
  – Partnerships with users

• Complementary instrument
  – Complementing GENI, Grid’5000, and other testbeds

• Sustainable instrument
  – Industry connections

http://www.chameleoncloud.org/
Concluding Remarks

- Provided an assessment of the MPI standardization effort, including current and emerging features
- Discussed the latest trends in PGAS and Hybrid MPI+PGAS programming models
- Outlined challenges in supporting these programming models on modern clusters
- Demonstrated how solutions to these challenges can be designed with MVAPICH2/ MVAPICH2-X and their performance benefits
- As parallel systems are getting complex, challenging solutions are needed to have MPI and PGAS support with scalability, performance and fault-resilience
Funding Acknowledgments

Funding Support by

[Logos of various sponsors]

Equipment Support by

[Logos of various sponsors]
Personnel Acknowledgments

Current Students
- A. Bhat (M.S.)
- S. Chakraborty (Ph.D.)
- N. Islam (Ph.D.)
- M. Li (Ph.D.)
- R. Rajachandrasekhar (Ph.D.)
- M. Rahman (Ph.D.)
- D. Shankar (Ph.D.)
- A. Venkatesh (Ph.D.)
- J. Zhang (Ph.D.)

Past Students
- P. Balaji (Ph.D.)
- D. Buntinas (Ph.D.)
- S. Bhagvat (M.S.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)

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

Past Research Scientist
- G. Santhanaraman (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- E. Mancini
- S. Marcarelli
- J. Vienne

Past Researchers
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- D. Bureddy
Thank You!

panda@cse.ohio-state.edu

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

The MVAPICH2/MVAPICH2-X Project
http://mvapich.cse.ohio-state.edu/
Additional Challenges and Results will be presented.

- Plenary Talk, Designing Software Libraries and Middleware for Exascale Computing: Opportunities and Challenges
  - Dec 4th, 3:30-4:30pm
Multiple Open Positions in My Group

• Looking for Bright and Enthusiastic Personnel to join as
  – PhD Students
  – Post-Doctoral Researchers
  – MPI Programmer/Software Engineer
  – Hadoop/Big Data Programmer/Software Engineer

• If interested, please contact me at this conference and/or send an e-mail to panda@cse.ohio-state.edu