MVAPICH2 MPI Libraries to Exploit Latest Networking and Accelerator Technologies

Talk at NRL booth (SC 2016)

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): ExaFlop & ExaByte

100-200 PFlops in 2016-2018

1 EFlops in 2021-2024?

40K EBytes in 2020?

10K-20K EBytes in 2016-2018

ExaFlop & HPC

ExaByte & BigData

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 and Intel Xeon Phi)

Multi-core Processors

High Performance Interconnects - InfiniBand
<1usec latency, 100Gbps Bandwidth>

Accelerators / Coprocessors
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM
Parallel Programming Models

Shared Memory Model

- SHMEM, DSM
- Global view improves programmer productivity
- Idea is to decouple data movement with process synchronization
- Processes should have asynchronous access to globally distributed data
- Well suited for irregular applications and kernels that require dynamic access to different data
- Different Approaches
  - Library-based (Global Arrays, OpenSHMEM)
  - Compiler-based (Unified Parallel C (UPC), Co-Array Fortran (CAF))
  - HPCS Language-based (X10, Chapel, Fortress)

Distributed Memory Model

- MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)
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 2,690 organizations in 83 countries
  - More than 402,000 (> 0.4 million) downloads from the OSU site directly
  - Empowering many TOP500 clusters (Nov ‘16 ranking)
    - 1st ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China
    - 13th ranked 241,108-core cluster (Pleiades) at NASA
    - 17th ranked 519,640-core cluster (Stampede) at TACC
    - 40th ranked 76,032-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  - 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 at NSC, Wuxi, China (1st in Nov’16, 10,649,640 cores, 93 PFlops)
MVAPICH2 Architecture

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, OmniPath)

- 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 (MIC, KNL*), NVIDIA GPGPU)

- Transport Mechanisms
  - Shared Memory
  - CMA
  - IVSHMEM
- Modern Features
  - MCDRAM*
  - NVLink*
  - CAPI*

* Upcoming
MVAPICH/MVAPICH2 Release Timeline and Downloads

Timeline

MV 0.9.4
MV2 0.9.0
MV2 0.9.8
MV2 1.0
MV2 1.0.3
MV 1.1
MV2 1.4
MV2 1.5
MV2 1.6
MV2 1.7
MV2 1.8
MV2 1.9
MV2 2.1
MV2 2.2rc1
MV2-GDR 2.0b
MV2-MIC 2.0
MV2 2.1
MV2-GDR 2.2rc1
MV2-Virt 2.1rc2
MV2-X

Number of Downloads

Sep-04  Jan-05  May-16
0  50000  100000  150000  200000  250000  300000  350000  400000  450000
Sep-04  Jan-05  May-16
0  50000  100000  150000  200000  250000  300000  350000  400000  450000

Network Based Computing Laboratory
NRL (SC 2016)
### MVAPICH2 Software Family

<table>
<thead>
<tr>
<th>High-Performance Parallel Programming Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MVAPICH2</strong></td>
</tr>
<tr>
<td><strong>MVAPICH2-X</strong></td>
</tr>
<tr>
<td><strong>MVAPICH2-GDR</strong></td>
</tr>
<tr>
<td><strong>MVAPICH2-Virt</strong></td>
</tr>
<tr>
<td><strong>MVAPICH2-EA</strong></td>
</tr>
<tr>
<td><strong>MVAPICH2-MIC</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Microbenchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OMB</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OSU INAM</strong></td>
</tr>
<tr>
<td><strong>OEMT</strong></td>
</tr>
</tbody>
</table>
Outline

• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature
• Streaming Support with IB Multicast and GDR
• High-Performance Deep Learning with MVAPICH2-GDR
• Conclusions
Optimizing MPI Data Movement on GPU Clusters

• Connected as PCIe devices – Flexibility but Complexity

1. Intra-GPU
2. Intra-Socket GPU-GPU
3. Inter-Socket GPU-GPU
4. Inter-Node GPU-GPU
5. Intra-Socket GPU-Host
6. Inter-Socket GPU-Host
7. Inter-Node GPU-Host

8. Inter-Node GPU-GPU with IB adapter on remote socket

• For each path different schemes: Shared_mem, IPC, GPUDirect RDMA, pipeline ...
• Critical for runtimes to optimize data movement while hiding the complexity
MPI + CUDA - Naive

- Data movement in applications with standard MPI and CUDA interfaces

**At Sender:**

```
cudaMemcpy(s_hostbuf, s_devbuf, ...);
MPI_Send(s_hostbuf, size, ...);
```

**At Receiver:**

```
MPI_Recv(r_hostbuf, size, ...);
cudaMemcpy(r_devbuf, r_hostbuf, ...);
```

*High Productivity and Low Performance*
MPI + CUDA - Advanced

- Pipelining at user level with non-blocking MPI and CUDA interfaces

**At Sender:**

```c
for (j = 0; j < pipeline_len; j++)
    cudaMemcpyAsync(s_hostbuf + j * blk, s_devbuf + j * blksz, ...);
for (j = 0; j < pipeline_len; j++) {
    while (result != cudaSuccess) {
        result = cudaStreamQuery(...);
        if(j > 0) MPI_Test(...);
    }
    MPI_Isend(s_hostbuf + j * block_sz, blksz, ...);
}
MPI_Waitall();
```

<<Similar at receiver>>

**Low Productivity and High Performance**
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 1.8-2.2 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
GPU-Direct RDMA (GDR) with CUDA

- OFED with support for GPUDirect RDMA is developed by NVIDIA and Mellanox
- OSU has a design of MVAPICH2 using GPUDirect RDMA
  - Hybrid design using GPU-Direct RDMA
    - GPUDirect RDMA and Host-based pipelining
    - Alleviates P2P bandwidth bottlenecks on SandyBridge and IvyBridge
    - Similar bottlenecks on Haswell
  - Support for communication using multi-rail
  - Support for Mellanox Connect-IB and ConnectX VPI adapters
  - Support for RoCE with Mellanox ConnectX VPI adapters

<table>
<thead>
<tr>
<th></th>
<th>SNB E5-2670</th>
<th>IVB E5-2680V2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intra-socket</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2P read</td>
<td>3.5 GBs</td>
<td>&lt;300 MBs</td>
</tr>
<tr>
<td>P2P write</td>
<td>6.4 GBs</td>
<td>&lt;300 MBs</td>
</tr>
<tr>
<td><strong>Inter-sockets</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2P read</td>
<td>&lt;1.0 GBs</td>
<td>&lt;300 MBs</td>
</tr>
<tr>
<td>P2P write</td>
<td>5.2 GBs</td>
<td>&lt;300 MBs</td>
</tr>
</tbody>
</table>
Performance of MVAPICH2-GPU with GPU-Direct RDMA (GDR)

**GPU-GPU internode latency**
- MV2-GDR2.2
- MV2-GDR2.0b
- MV2 w/o GDR

**GPU-GPU Internode Bandwidth**
- MV2-GDR2.2
- MV2-GDR2.0b
- MV2 w/o GDR

Intel Ivy Bridge (E5-2680 v2) node - 20 cores
NVIDIA Tesla K40c GPU
Mellanox Connect-X4 EDR HCA
CUDA 8.0
Mellanox OFED 3.0 with GPU-Direct-RDMA
Application-Level Evaluation (HOOMD-blue)

- **Platform:** Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- **HoomdBlue Version 1.0.5**
  - GDRCOPY enabled: MV2_USE_CUDA=1 MV2_IBA_HCA=mlx5_0 MV2_IBA_EAGER_THRESHOLD=32768 MV2_VBUF_TOTAL_SIZE=32768 MV2_USE_GPUDIRECT_LOOPBACK_LIMIT=32768 MV2_USE_GPUDIRECT_GDRCOPY=1 MV2_USE_GPUDIRECT_GDRCOPY_LIMIT=16384

### 64K Particles

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Average Time Steps per second (TPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>MV2: 1500, MV2+GDR: 3000</td>
</tr>
<tr>
<td>8</td>
<td>MV2: 1500, MV2+GDR: 3000</td>
</tr>
<tr>
<td>16</td>
<td>MV2: 1500, MV2+GDR: 3000</td>
</tr>
<tr>
<td>32</td>
<td>MV2: 1500, MV2+GDR: 3000</td>
</tr>
</tbody>
</table>

### 256K Particles

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Average Time Steps per second (TPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>MV2: 750, MV2+GDR: 1500</td>
</tr>
<tr>
<td>8</td>
<td>MV2: 750, MV2+GDR: 1500</td>
</tr>
<tr>
<td>16</td>
<td>MV2: 750, MV2+GDR: 1500</td>
</tr>
<tr>
<td>32</td>
<td>MV2: 750, MV2+GDR: 1500</td>
</tr>
</tbody>
</table>

**2X** increase in performance with GDRCOPY enabled.
Full and Efficient MPI-3 RMA Support

Small Message Latency

<table>
<thead>
<tr>
<th>Message Size (bytes)</th>
<th>Latency (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.88 us</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
</tr>
<tr>
<td>2K</td>
<td></td>
</tr>
<tr>
<td>8K</td>
<td></td>
</tr>
</tbody>
</table>

MVAPICH2-GDR-2.2
Intel Ivy Bridge (E5-2680 v2) node - 20 cores, NVIDIA Tesla K40c GPU
Mellanox Connect-IB Dual-FDR HCA, CUDA 7
Mellanox OFED 2.4 with GPU-Direct-RDMA
Outline

• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature
• Streaming Support with IB Multicast and GDR
• High-Performance Deep Learning with MVAPICH2-GDR
• Conclusions
Non-Blocking Collectives (NBC) using Core-Direct Offload

- MPI NBC decouples initiation (Ialltoall) and completion (Wait) phases and provide overlap potential (Ialltoall + compute + Wait) but CPU drives progress largely in Wait (=> 0 overlap)
- CORE-Direct feature provides true overlap capabilities by providing a priori specification of a list of network-tasks which is progressed by the NIC instead of the CPU (hence freeing it)
- We propose a design that combines GPUDirect RDMA and Core-Direct features to provide efficient support of CUDA-Aware NBC collectives on GPU buffers
  - Overlap communication with CPU computation
  - Overlap communication with GPU computation
- Extend OMB with CUDA-Aware NBC benchmarks to evaluate
  - Latency
  - Overlap on both CPU and GPU

CUDA-Aware Non-Blocking Collectives


Platform: Wilkes: Intel Ivy Bridge
NVIDIA Tesla K20c + Mellanox Connect-IB
Available since MVAPICH2-GDR 2.2b
Outline

• MVAPICH2-GPU with GPUDirect-RDMA (GDR)

• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature

• Streaming Support with IB Multicast and GDR

• High-Performance Deep Learning with MVAPICH2-GDR

• Conclusions
Non-contiguous Data Exchange

- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions

- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration
MPI Datatype Processing (Computation Optimization)

- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes

- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels

- Flexible set of parameters for users to tune kernels
  - Vector
    - MV2_CUDA_KERNEL_VECTOR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_VECTOR_YSIZE
  - Subarray
    - MV2_CUDA_KERNEL_SUBARR_TIDBLK_SIZE
    - MV2_CUDA_KERNEL_SUBARR_XDIM
    - MV2_CUDA_KERNEL_SUBARR_YDIM
    - MV2_CUDA_KERNEL_SUBARR_ZDIM
  - Indexed_block
    - MV2_CUDA_KERNEL_IDXBLK_XDIM
MPI Datatype Processing (Communication Optimization)

Common Scenario

MPI_Isend (A,.. Datatype,...)
MPI_Isend (B,.. Datatype,...)
MPI_Isend (C,.. Datatype,...)
MPI_Isend (D,.. Datatype,...)
...

MPI_Waitall (...);

*A, B...contain non-contiguous
MPI Datatype

Waste of computing resources on CPU and GPU

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

Wilkes GPU Cluster

- Default
- Callback-based
- Event-based

Normalized Execution Time vs. Number of GPUs

CSCS GPU cluster

- Default
- Callback-based
- Event-based

Normalized Execution Time vs. Number of GPUs

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

Cosmo model: http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/

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

Outline

- MVAPICH2-GPU with GPUDirect-RDMA (GDR)
- What’s new with MVAPICH2-GDR
  - Efficient MPI-3 Non-Blocking Collective support
  - Maximal overlap in MPI Datatype Processing
  - Efficient Support for Managed Memory
  - Initial support for GPUDirect Async feature
- Streaming Support with IB Multicast and GDR
- High-Performance Deep Learning with MVAPICH2-GDR
- Conclusions
Initial (Basic) Support for GPU Unified Memory

- CUDA 6.0 NVIDIA introduced CUDA Managed (or Unified) memory allowing a common memory allocation for GPU or CPU through `cudaMallocManaged()` call
- Significant productivity benefits due to abstraction of explicit allocation and `cudaMemcpy()`
- Extended MVAPICH2 to perform communications directly from managed buffers (Available in MVAPICH2-GDR 2.2b)
- OSU Micro-benchmarks extended to evaluate the performance of point-to-point and collective communications using managed buffers
  - Available since OMB 5.2

D. S. Banerjee, K Hamidouche, and D. K Panda, Designing High Performance Communication Runtime for GPUManaged Memory: Early Experiences, GPGPU-9 Workshop, to be held in conjunction with PPoPP ’16
Enhanced Support for Intra-node Managed Memory

- CUDA Managed => no memory pin down
  - No IPC support for intra-node communication
  - No GDR support for inter-node communication
- Initial and basic support in MVAPICH2-GDR
  - For both intra- and inter-nodes use “pipeline through” host memory
- Enhance intra-node managed memory to use IPC
  - Double buffering pair-wise IPC-based scheme
  - Brings IPC performance to Managed memory
- High performance and high productivity
- 2.5 X improvement in bandwidth
- Available in MVAPICH2-GDR 2.2
Outline

• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature

• Streaming Support with IB Multicast and GDR
• High-Performance Deep Learning with MVAPICH2-GDR
• Conclusions
Overview of GPUDirect aSync (GDS) Feature: Current MPI+CUDA interaction

CUDA_Kernel_a<<<>>>(A, ..., stream1)
cudaStreamSynchronize(stream1)
MPI_ISend (A, ..., req1)
MPI_Wait (req1)
CUDA_Kernel_b<<<>>>(B, ..., stream1)

100% CPU control
• Limits the throughput of a GPU
• Limits the asynchronous progress
• Wastes CPU cycles
MVAPICH2-GDS: Decouple GPU Control Flow from CPU

CUDA_Kernel_a<<<>>>(A,..., stream1)
MPI_Isend (A,..., req1, stream1)
MPI_Wait (req1, stream1) (non-blocking from CPU)
CUDA_Kernel_b<<<>>>(B,..., stream1)

CPU offloads the compute, communication and synchronization tasks to GPU
- CPU is out of the critical path
- Tight interaction between GPU and HCA
- Hide the overhead of kernel launch
- Requires MPI semantics extensions
  - All operations are asynchronous from CPU
  - Extends MPI semantics with Stream-based semantics
**Latency Oriented:** Able to hide the kernel launch overhead
- 25% improvement at 256 Bytes compared to default behavior

**Throughput Oriented:** Asynchronously to offload queue the Communication and computation tasks
- 14% improvement at 1KB message size
- Requires some tuning and expect better performance for Application with different Kernels

Intel SandyBridge, NVIDIA K20 and Mellanox FDR HCA
Outline

• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature
• Streaming Support with IB Multicast and GDR
• High-Performance Deep Learning with MVAPICH2-GDR
• Conclusions
Streaming Applications

- Examples - surveillance, habitat monitoring, proton computed tomography (pCT), etc..
- Require efficient transport of data from/to distributed sources/sinks
- Sensitive to latency and throughput metrics

- Require HPC resources to efficiently carry out compute-intensive tasks

Src: http://www.symmetrymagazine.org/article/april-2012/proton-beam-on
Motivation

- Streaming applications on HPC systems

  1. Communication (MPI)
     - Broadcast-type operations

  2. Computation (CUDA)
     - Multiple GPU nodes as workers

Data Source

Real-time streaming

Sender

Data streaming-like broadcast operations

HPC resources for real-time analytics

Motivation

Streaming applications on HPC systems

1. Communication (MPI)
   - Broadcast-type operations

2. Computation (CUDA)
   - Multiple GPU nodes as workers
IB Multicast Example

Switch decodes inbound packet header (LRH) DLID to determine target output ports.

Router decodes inbound packet header (GRH) GID multicast address to determine target out-
Problem Statement

- Can we design a GPU broadcast mechanism that can deliver low latency and high throughput for streaming applications?
- Can we combine GDR and MCAST features to
  - Achieve the best performance
  - Free-up the Host-Device PCIe bandwidth for application needs
- Can such design be extended to support heterogeneous configuration (host-to-device)?
- Can we design and efficient MCAST based broadcast for multi-GPU systems?
- Can we design an efficient reliability support on top of the UD-based MCAST broadcast?
- How can we demonstrate such benefits at benchmark and applications level?
Two Major Solutions (So far)

• Handling efficient broadcast on multi-GPU node systems
  

• Providing reliability support
  
  
  • Will be presented on Friday (11/18/16) at 9:15am, Room #355-D
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
SL-based Design for H-D Heterogeneous Support

- Redesigned broadcast benchmark with Root buffer on Host & non-Root on Device
- Inter-node experiments @ Wilkes cluster, 32 GPUs, 1 GPU/node

[Graph showing latency vs message size for different broadcast methods: SL-MCAST, SGL-MCAST, and Host-MCAST.]

Lower is better

56% reduction

39% reduction
Benefits of the Availability of Host-Device PCI Resources

- Mimic the behavior of streaming applications @ CSCS cluster, 88 GPUs, 8 NVIDIA K80 GPUs per node
  - Broadcast operations overlapped with application level Host-Device transfers
    - Main thread performing MCAST (streaming)
    - Helper thread starting CUDA kernels and performing Async H-D copies

![Throughput Graph]

Throughput (GB/s)

Message Size (Bytes)

IPC SL-MCAST

SHMEM SL-MCAST

Higher is better

3.2X
New RMA-based Reliability Design

• Goals of the proposed design
  – Allows the receivers to retrieve lost MCAST packets through the RMA operations without interrupting sender
  ➢ Maintains pipelining of broadcast operations
  ➢ Minimizes consumption of PCIe resources

• Major Benefit of MPI-3 Remote Memory Access (RMA)*
  – Supports one-sided communication ➔ broadcast sender won’t be interrupted

• Major Challenge
  – How and where receivers can retrieve the correct MCAST packets through RMA operations

*”MPI Forum”, http://mpi-forum.org/
Implementing MPI_Bcast: Receiver Side

- When a receiver experiences **timeout** (lost MCAST packet)
  - Performs the RMA Get operation to the sender’s backup buffer to retrieve lost MCAST packets
  - **Sender is not interrupted**
Evaluation: Overhead

- Negligible overhead compared to existing NACK-based design
- RMA-based design outperforms NACK-based scheme for large messages
  - A helper thread in the background performs backups of MCAST packets
Evaluation: Broadcast Rate (Throughput)

- Equal or better than the leading NACK-based design for different message sizes and error rates
- Always yields (up to 56%) a higher broadcast rate than the existing NACK-based design

### Error rates:
- 0.01%
- 0.10%
- 1%

### Message Size (Bytes)
- 1
- 4
- 16
- 64
- 256
- 1K
- 4K
- 16K
- 64K
- 256K
- 1M
- 4M

Normalized to SL-based MCAST with NACK-based retransmission scheme
Outline

• MVAPICH2-GPU with GPUDirect-RDMA (GDR)

• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature

• Streaming Support with IB Multicast and GDR

• High-Performance Deep Learning with MVAPICH2-GDR

• Conclusions
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

- How to address these newer requirements?
  - GPU-specific Communication Libraries (NCCL)
    - NVidia's NCCL library provides inter-GPU communication
  - CUDA-Aware MPI (MVAPICH2-GDR)
    - Provides support for GPU-based communication

- Can we exploit CUDA-Aware MPI and NCCL to support Deep Learning applications?

---

Efficient Large Message Broadcast using NCCL and CUDA-Aware MPI for Deep Learning,
A. Awan, K. Hamidouche, A. Venkatesh, and D. K. Panda,
The 23rd European MPI Users' Group Meeting (EuroMPI 16), Sep 2016 [Best Paper Runner-Up]
Can we optimize MVAPICH2-GDR to efficiently support DL frameworks?

- We need to design large-scale reductions using CUDA-Awareness
- GPU performs reduction using kernels
- Overlap of computation and communication
- Hierarchical Designs

Proposed designs achieve 2.5x speedup over MVAPICH2-GDR and 133x over OpenMPI
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 in MVAPICH2-GDR 2.2GA

![Graphs showing latency vs. message size for different operations and number of GPUs.]
OSU-Caffe: Scalable Deep Learning


- Benefits and Weaknesses
  - Multi-GPU Training within a single node
  - Performance degradation for GPUs across different sockets
  - No Scale-out available

- 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/
Outline

• Overview of the MVAPICH2 Project
• MVAPICH2-GPU with GPUDirect-RDMA (GDR)
• What’s new with MVAPICH2-GDR
  • Efficient MPI-3 Non-Blocking Collective support
  • Maximal overlap in MPI Datatype Processing
  • Efficient Support for Managed Memory
  • Initial support for GPUDirect Async feature
• HiDL Overview: High-Performance Deep Learning with MVAPICH2-GDR
• Conclusions
Conclusions

- MVAPICH2 optimizes MPI communication on InfiniBand clusters with GPUs
- Provides optimized designs for point-to-point two-sided and one-sided communication, datatype processing and collective operations
- Takes advantage of CUDA features like IPC and GPUDirect RDMA families
- New designs help to get good performance for streaming and deep learning applications
Thank You!

panda@cse.ohio-state.edu

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

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/