High-Performance Broadcast for Streaming and Deep Learning

Ching-Hsiang Chu

chu.368@osu.edu

Department of Computer Science and Engineering
The Ohio State University
Outline

• Introduction

• Proposed Designs in MVAPICH2-GDR

• Performance Evaluation

• Concluding Remarks
Trends in Modern HPC Architecture

- Multi-core/many-core technologies
- High Performance Interconnects
- Accelerators/Coprocessors are becoming common in high-end systems
- High Performance Storage and Compute devices

Multi-core Processors

High Performance Interconnects – InfiniBand (IB), Omni-Path
< 1 μsec latency, 100 Gbps Bandwidth>

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

SSD, NVMe-SSD, NVRAM

Sunway TaihuLight
K - Computer
Tianhe – 2
Titan

Network Based Computing Laboratory
OSU Booth - SC17
Architectures for Deep Learning (DL)

**Past and Current Trend**
- Multi-core CPUs within a node
  - Multi-core CPUs across nodes
    - IB Networks

**Near-future**
- Multi-core CPUs + Multi-GPU within a node
  - Multi-core CPUs + Multi-GPU across nodes
    - IB Networks
    - E.g., NVIDIA DGX-1 systems

Network Based Computing Laboratory
OSU Booth - SC17
Streaming Applications

- Streaming applications on HPC systems
  1. Communication (MPI)
     - Broadcast-type operations
  2. Computation (CUDA)
     - Multiple GPU nodes as workers

Data Source

Real-time streaming

HPC resources for real-time analytics

Sender

Data streaming-like broadcast operations

Worker
  - CPU
  - GPU

Worker
  - CPU
  - GPU

Worker
  - CPU
  - GPU

Worker
  - CPU
  - GPU

Worker
  - CPU
  - GPU
High-performance Deep Learning

- Computation using GPU
- Communication using MPI
  - Exchanging partial gradients after each minibatch
  - All-to-all (Multi-Source) communications
    - E.g., MPI_Bcast
- Challenges
  - High computation-communication overlap
  - Good scalability for upcoming large-scale GPU clusters
  - No application-level modification
Outline

• Introduction
• Proposed Designs in MVAPICH2-GDR
• Performance Evaluation
• Concluding Remarks
Hardware Multicast-based Broadcast

- For GPU-resident data, using
  - GPUDirect RDMA (GDR)
  - InfiniBand Hardware Multicast (IB-MCAST)

- Overhead
  - IB UD limit
  - GDR limit

Hardware Multicast-based Broadcast (con’t)

• Heterogeneous Broadcast for streaming applications

➢ Free-up PCIe resources

Optimized Broadcast Send

• **Preparing Intermediate buffer** *(im_buf)*
  - Page-locked (pinned) host buffer
    - Fast Device-Host data movement
  - Allocated at initialization phase
    - Low overhead

• **Streaming data through host**
  - Fine-tuned chunked data
  - Asynchronous copy operations
    - Three-stage pipeline

---

Optimized Broadcast Receive

- Zero-copy broadcast receive
  - Pre-posted user buffer \((d_{in})\)
  - Avoids additional data movement
  - Leverages IB Scatter and GDR features
  - Low-latency
  - Free-up PCIe resources for applications

\(\text{MPI\_Bcast}(d_{in},...)\)

Broadcast on Multi-GPU systems

- Proposed Intra-node Topology-Aware Broadcast
  - CUDA InterProcess Communication (IPC)

Multicast steps

cudaMemcpy (Device ↔ Device)

Efficient Reliability Support for IB-MCAST

- 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**

Outline

• Introduction
• Proposed Designs in MVAPICH2-GDR
• Performance Evaluation
• Concluding Remarks
Experimental Environments

- **Ohio State University (OSU) Micro-Benchmark (OMB)**
  
  [http://mvapich.cse.ohio-state.edu/benchmarks/](http://mvapich.cse.ohio-state.edu/benchmarks/)
  
  - osu_bcast - MPI_Bcast Latency Test
  - osu_bcast_streaming – MPI_Bcast streaming Test

- **Deep learning framework: CUDA-Aware Microsoft Cognitive Toolkit (CA-CNTK)**
  
  - AlexNet and VGG models with ImageNet dataset

---

Benchmark Evaluation

- @ RI2 cluster, 16 GPUs, 1 GPU/node

![Graph showing latency vs. message size](image)

- Provide near-constant latency over the system sizes
- Reduces up to 65% of latency for large messages

Lower is better

• **IB-MCAST + GDR + Topology-aware IPC-based schemes**
  
  – Up to **58%** and **79%** reduction for small and large messages

Deep Learning Frameworks

• @ RI2 cluster, 16 GPUs, 1 GPU/node:
  – CUDA-Aware Microsoft Cognitive Toolkit (CA-CNTK) without modification

<table>
<thead>
<tr>
<th>Training Time (s)</th>
<th>Number of GPU nodes</th>
<th>AlexNet model</th>
<th>VGG model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>MV2-GDR-Knomial</td>
<td>MV2-GDR-Knomial</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>MV2-GDR-Ring</td>
<td>MV2-GDR-Ring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MCAST-GDR-Opt</td>
<td>MCAST-GDR-Opt</td>
</tr>
</tbody>
</table>

- *Lower is better*

- Reduces up to 24% and 15% of latency for AlexNet and VGG models
- Higher improvement is expected for larger system sizes
Concluding Remarks

• High-performance broadcast schemes to leverage GDR and IB-MCAST features for streaming and deep learning applications
  – Optimized streaming design for large messages transfers

• High-performance reliability support for IB-MCAST

  ➢ These features are included in MVAPICH2-GDR 2.3a

  ➢ http://mvapich.cse.ohio-state.edu/

  ➢ http://mvapich.cse.ohio-state.edu/userguide/gdr/2.3a/
Thank You!

Ching-Hsiang Chu
chu.368@osu.edu

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

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


Thank You!

- Join us for more tech talks from MVAPICH2 team
  - http://mvapich.cse.ohio-state.edu/conference/677/talks/

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

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

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>Number of processes</td>
<td>N/A</td>
</tr>
<tr>
<td>( m )</td>
<td>Number of broadcast sources</td>
<td>N/A</td>
</tr>
<tr>
<td>( t_s )</td>
<td>Set up time for sending data</td>
<td>sec</td>
</tr>
<tr>
<td>( t_o(n) )</td>
<td>Overhead for issuing an IB-MCAST packet</td>
<td>sec</td>
</tr>
<tr>
<td>( M )</td>
<td>Original message size</td>
<td>bytes</td>
</tr>
<tr>
<td>( C )</td>
<td>Size of a data chunk</td>
<td>bytes</td>
</tr>
<tr>
<td>( U )</td>
<td>Maximum Transmission Unit for IB-MCAST, provided by hardware manufacturer</td>
<td>bytes</td>
</tr>
<tr>
<td>( B_H )</td>
<td>Bandwidth of reading Host memory</td>
<td>bytes/sec</td>
</tr>
<tr>
<td>( B_G )</td>
<td>Bandwidth of reading GPU memory (NVIDIA GPUDirect RDMA)</td>
<td>bytes/sec</td>
</tr>
<tr>
<td>( B_{PCIe} )</td>
<td>PCIe Bandwidth between Host and GPU memory</td>
<td>bytes/sec</td>
</tr>
</tbody>
</table>

### Diagram

- **Message**
  - \( M \)
  - \( C \)
  - \( U \)

- **Bandwidth**
  - \( B_H \gg B_G \)
  - \( B_{PCIe} \)
  - \( B_G \)
Ring-based Broadcast

- **Direct**
  \[(n - 1) \times (t_s + \frac{M}{B_G})\]

- **Pipeline**
  \[\left[\frac{M}{C} + (n - 2)\right] \times (t_s + \frac{C}{B_G})\]

- **Staging**
  \[\frac{M}{B_{PCl_e}} + (n - 1) \times (t_s + \frac{M}{B_H})\]

Poor Scalability

Source
- CPU
- IB HCA
- GPU Data

Destination 1
- CPU
- IB HCA
- GPU Data

Destination 2
- IB HCA
- CPU
- Data
- GPU

Destination 3
- IB HCA
- CPU
- Data
- GPU
K-nomial-based Broadcast

- Direct
  \[ [\log_k n] \times \left( t_s + \frac{M}{B_G} \right) \]

- Pipeline
  \[ \left( \frac{M}{C} \times [\log_k n] \right) \times \left( t_s + \frac{C}{B_G} \right) \]

- Staging
  \[ \frac{M}{B_{ PCIe}} + [\log_k n] \times \left( t_s + \frac{M}{B_H} \right) \]

Non-optimized Scalability
Overlap Opportunities

Timeline

Overlap within a node

Overlap Across Nodes

Broadcast from Node A

Broadcast from Node B

Broadcast from Node C

- : cudaMemcpyAsync
- : IB Hardware Multicast
- : cudaStreamSynchronize
- : GDR Write
MCAST-based Broadcast

• NVIDIA GPUDirect[1]
  – Remote direct memory access (RDMA) transfers between GPUs and other PCIe devices \(\Rightarrow\) GDR
  – and more...

• InfiniBand (IB) hardware multicast (IB MCAST)[2]
  – Enables efficient designs of broadcast operations
    • Host-based[3]
    • GPU-based[4]

Future Work

• Extend the design for other broadcast-based collective algorithms as well as non-blocking operations
  – Allreduce, Allgather, ..., and so on