Designing Hierarchical Multi-HCA Aware Allgather in MPI

Tu Tran, Benjamin Michalowicz, Bharath Ramesh, Hari Subramoni, Aamir Shafi and Dhabaleswar K. Panda

Network Based Computing Laboratory
The Ohio State University
tran.839@osu.edu
Outline

• Motivation and Contributions
• Designing Multi-HCA Aware (MHA) Allgather
• Accelerating Allreduce with MHA Allgather
• Performance Evaluation
  – Microbenchmark-level
  – Application-level
• Summary
Motivation

• **Multi-rail** networks in existing and upcoming exa-scale systems
  – Summit and Sierra [1] – 2 adapters per node
  – ThetaGPU system [2] – 8 adapters per node

• Advantages:
  – Fault-tolerance
  – Performance

• Have we **efficiently utilized** all adapters per node in the context of **MPI**?
  – Yes, at **point-to-point** level
  – **No**, at the **collective** level
    => Collective designs need to be **revisited** and **augmented**
Multi-rail Support at P2P Level

Figure 1: Bandwidth comparison between intra-node and inter-node communication
Collective Performance with Multi-rail Support at P2P Level

Figure 2: Allgather 2 Nodes, 2 PPN communication timeline visualization
Contributions

• **Design** and implementation of Multi-HCA Aware (MHA) **Allgather** to speed up both **intranode** and **internode** performance

• **Performance evaluation** at both **microbenchmark** and **application** levels with state-of-the-art MPI libraries, namely **MVAPICH2-X** and **HPC-X**

• **Accelerating Allreduce** with the proposed Allgather

• **Performance Model** of MHA Allgather
Outline

• Motivation and Contributions
• **Designing Multi-HCA Aware (MHA) Allgather**
• Accelerating Allreduce with MHA Allgather
• Performance Evaluation
  – Microbenchmark-level
  – Application-level
• Summary
Intranode Design

• For **pure intranode** communication
  – Intranode **transfers** performed by CPUs
  – Network **adapters** stay idle

=> **How** can we efficiently **utilize** the idle **adapters** to **accelerate** the communication?

• **Idea:**
  – Each process **offloads** the same amount of **work** to **adapters**

(a) Direct Spread algorithm executes in 3 steps

(b) The proposed MHA-intra executes in 2 steps. Step 1’ is step 2 overlapping with 1st half of step 1. Step 2’ is step 3 overlapping with 2nd half of step 1.
Tuning Algorithm for Intranode Design

• What is the optimal offload size?
  – Processes and adapters need to finish at the same time
  – Otherwise, the one that takes long time will be a source of bottleneck

Figure 5: A chart showing the correlation between the offload size to adapters and latency
**Internode Design**

- **Observations:**
  - Intranode transfer is a source of bottleneck
  - Need to separate intranode and internode communication

- **The proposed hierarchical MHA Allgather**

  ![Diagram of Internode Design](image-url)

  - **Step 1:** Node-level Allgather
  - **Step 2:** Internode Allgather between leaders
  - **Step 3:** Intranode broadcast
### Figure 6: A timeline view of communication events of a node during interleader data exchange and node-level data distribution phases

<table>
<thead>
<tr>
<th>Local proc 0</th>
<th>Local proc 1</th>
<th>...</th>
<th>Local proc L-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i + 1</td>
<td>i</td>
<td></td>
<td>i - 1</td>
</tr>
<tr>
<td>i + 2</td>
<td>i + 1</td>
<td></td>
<td>i + 1</td>
</tr>
<tr>
<td>i + 3</td>
<td>i + 2</td>
<td></td>
<td>i + 2</td>
</tr>
</tbody>
</table>

- Copy chunk $i$ from local buffer to shared memory
- Copy chunk $i$ from shared memory to local buffer
- Internode transfer to get chunk $i$ by HCA

### Figure 7: A comparison of Recursive Doubling and Ring algorithms used in inter-leader data exchange phase
Outline

• Motivation and Contributions
• Designing Multi-HCA Aware (MHA) Allgather
• Accelerating Allreduce with MHA Allgather
• Performance Evaluation
  – Microbenchmark-level
  – Application-level
• Summary
Accelerating Allreduce with MHA Allgather

- **Ring Allreduce** is proven to be **bandwidth-optimal**, which is particularly suitable for **large messages**
- The **algorithm** executes in **two phases**

```
<table>
<thead>
<tr>
<th>Reduce-Scatter</th>
<th>Allgather</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$P_0$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$P_1$</td>
</tr>
<tr>
<td>$P_{N-2}$</td>
<td>$P_{N-2}$</td>
</tr>
<tr>
<td>$P_{N-1}$</td>
<td>$P_{N-1}$</td>
</tr>
</tbody>
</table>
```

...
Outline

• Motivation and Contributions
• Designing Multi-HCA Aware (MHA) Allgather
• Accelerating Allreduce with MHA Allgather

• **Performance Evaluation**
  – Microbenchmark-level
  – Application-level

• Summary
Experimental Setup

• Thor cluster of HPC Advisory Council

• Comparison with
  – MVAPICH2-X v2.3
  – HPC-X v2.10.0

• Application software
  – PyTorch v1.8.0
  – Horovod v0.20.0

<table>
<thead>
<tr>
<th>Specification</th>
<th>Thor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Nodes</td>
<td>32</td>
</tr>
<tr>
<td>Processor Family</td>
<td>Xeon Broadwell</td>
</tr>
<tr>
<td>Processor Model</td>
<td>E5-2697AV4</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>2.6 GHz</td>
</tr>
<tr>
<td>Sockets</td>
<td>2</td>
</tr>
<tr>
<td>Cores per Socket</td>
<td>16</td>
</tr>
<tr>
<td>RAM (DDR4)</td>
<td>256 GB</td>
</tr>
<tr>
<td>GPU Family</td>
<td>Tesla V100</td>
</tr>
<tr>
<td>GPUs</td>
<td>1</td>
</tr>
<tr>
<td>GPU Memory</td>
<td>32 GB</td>
</tr>
<tr>
<td>Interconnect</td>
<td>2x IB-HDR (100Gb/s)</td>
</tr>
</tbody>
</table>
OMB - Intranode Allgather Evaluation

(a) 2 Processes

65% improv.

(b) 4 Processes

70% improv.

(c) 8 Processes

55% improv.

(d) 16 Processes

10-35% improv.
OMB - Internode Allgather Evaluation

Figure 13: Proposed MPI_Allgather against state of the art libraries via OSU Microbenchmarks on 512 processes (16 nodes 32 PPN)

44-53% improv.

Figure 14: Proposed MPI_Allgather against state of the art libraries via OSU Microbenchmarks on 1024 processes (32 nodes 32 PPN)

61% improv.
OMB - Allreduce Evaluation

Figure 15: Evaluation of Proposed Inter-node MPI_Allreduce Design against state of the art libraries via OSU Microbenchmarks at scale (32 PPN)

15-34% improv.  31-39% improv.  44-56% improv.
Matrix-Vector Multiplication Kernel

- 1D row layout partition
- The proposed Allgather **outperforms** both HPC-X and MVAPICH2-X
  - By up to **1.98x** and **1.42x** for strong scaling
  - And **1.84x** and **1.94x** for weak scaling experiments with **1024 processes**

![Graph](image1)

**Figure 16:** Performance Evaluation of MHA against state of the art MPI libraries in a Matrix-Vector Multiplication kernel for Weak and Strong Scaling
Deep Learning Training

- **CPU-based** training
  - with **25.6**, **44.7** and **60.4** millions of parameters, respectively
- Up to **7.83%** better than **MVAPICH2-X**

![Graphs showing performance improvements](image)

*Figure 17: Proposed MHA design against MVAPICH2-X via PyTorch + Horovod DL Performance Evaluation: Images Per Second*
Outline

• Motivation and Contributions
• Designing Multi-HCA Aware (MHA) Allgather
• Accelerating Allreduce with MHA Allgather
• Performance Evaluation
  – Microbenchmark-level
  – Application-level
• Summary
Summary

- This paper proposed a **Multi-HCA Allgather** that
  - Utilizes all the available network adapters within a node
  - Provides high overlap between inter-node and intra-node communication

- At the micro-benchmark level
  - The Improvements are up to **62%** and **61%** better than HPC-X and MVAPICH2-X for 1024 processes
  - The design also boosts the performance of Ring **Allreduce** by **56%** and **44%** compared to HPC-X and MVAPICH2-X

- At the application level
  - The enhanced Allgather shows **1.98x** and **1.42x** improvement in a matrix-vector **multiplication kernel** when compared to HPC-X and MVAPICH2-X
  - Allreduce performs up to **7.83%** better in **deep learning training against** MVAPICH2-X
Thank You!

tran.839@osu.edu

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

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

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

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/
References