2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid) | 978-1-7281-9586-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCGrid51090.2021.00021

Adaptive and Hierarchical Large Message All-to-all Communication Algorithms for Large-scale Dense GPU Systems*

Kawthar Shafie Khorassani*, Ching-Hsiang Chu*, Quentin G. Anthony*,
Hari Subramoni† and Dhabaleswar K. Panda†
Department of Computer Science and Engineering
The Ohio State University, Columbus, Ohio
E-mail: *{shafiekhorassani.1, chu.368, anthony.301}@osu.edu, †{subramon, panda}@cse.ohio-state.edu

***Abstract*—In recent years, GPU-enhanced clusters have become more prevalent in High-Performance Computing (HPC), leading to a demand for more efficient multi-GPU communication. This makes it increasingly important to explore performance enhancements that can be attained through the communication middleware such as MPI, in order to fully take advantage of the GPUs available on these systems. In this paper, we propose locality-aware and adaptive schemes for hierarchical All-to-all collective communication on large-scale dense GPU systems. The proposed algorithms utilize the high bandwidth made available through the NVLink interconnect between GPUs in order to overlap communication latency. We focus on personalized and non-personalized all-to-all collective communication. These are components of modern scientific computing applications that utilize matrix transpose and three-dimensional Fast Fourier Transforms (FFT) and becoming more relevant for Deep Learning workloads with model and hybrid parallelisms. The performance evaluation with an application kernel performing three-dimensional FFT indicates that the proposed schemes for personalized all-to-all can lead to up to 15-25% lower execution time on 256 GPUs on the Lassen system. We demonstrate approximately 8% enhancement in training time for distributed K-FAC used in Deep Learning training on up to 128 GPUs. We also demonstrate approximately 22% and 30% improvement in the performance of non-personalized and personalized all-to-all benchmarks, respectively, compared to the state-of-the-art MPI libraries on the Summit and Lassen systems.**

***Index Terms*—Allgather, All-to-all, GPU, MPI, NVLink**

I. Introduction

Graphics Processing Units (GPUs) have become prevalent accelerators in modern high-performance computing (HPC) systems for empowering various applications ranging from traditional scientific applications to artificial intelligence (AI) enabled applications. In recent years, more powerful HPC systems and cloud platforms are being deployed with thousands of GPU nodes along with cutting-edge CPU, and high-speed interconnects [1]. This dense-GPU configuration has become a trend in HPC systems as a result of the rapid development of GPU interconnects such as NVIDIA NVLink/NVSwitch, AMD Infinity Fabric and Intel X^e link. For example, the top-2 Summit system, Figure 1, is powered by 4,608 GPU nodes where each node has 6 NVIDIA V100 GPUs, and each socket is fully connected by high-speed NVLink. The topology of the #3 ranked Sierra system and the #17 ranked Lassen system use a similar hardware configuration as Figure 1 but with 4 GPUs per node. The next generation of exascale HPC systems will also be powered by multiple GPUs, and high-speed interconnects [2], [3].

With the increased deployment of large and dense GPU systems, it is vital to perform efficient data movement between GPUs for running HPC applications at scale. Although plenty of research has been addressing the optimization of point-to-point [4]–[6] and collective communication such as Broadcast and Allreduce [7]–[9] on large-scale GPU systems, GPU-optimized personalized All-to-all (*MPI_Alltoall*) and non-personalized All-to-all (*MPI_Allgather*) communications are rarely discussed in the literature. All-to-all is heavily used to perform matrix transposes or Fast Fourier Transforms (FFT) in many legacy scientific applications when data is distributed into multiple nodes or processes. All-to-all and Allgather operations are also gaining attention and importance recently for performing model parallelism for deep neural network training on GPU clusters [10], [11]. Given these software and hardware developments, it is vital to conduct an in-depth study and to optimize All-to-all and Allgather communication on the current and next-generation dense-GPU systems.

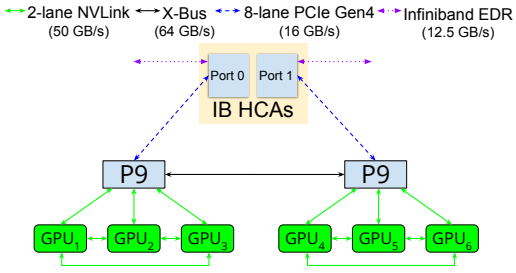


Fig. 1. Hardware configuration of the Summit Cluster. (Courtesy [12])

A. Motivation

All-to-all and Allgather are data-intensive operations as they would inject a large amount of data into the network. The well-known All-to-all algorithms, as summarized in Table I, only consider the communication among nodes through symmetric interconnects, e.g., InfiniBand network. However, there are various interconnect architectures used in modern dense-GPU systems.

*This research is supported in part by NSF grants #1818253, #1854828, #1931537, #2007991, #2018627, and XRAC grant #NCR-130002.

978-1-7281-9586-5/21/$31.00 ©2021 IEEE
DOI 10.1109/CCGrid51090.2021.00021

In such systems with asymmetric interconnects, the traditional All-to-all algorithms can easily yield contention on the slowest links, e.g., all GPUs within a node are performing inter-node communication over IB, while high-speed NVLink and PCIe remain idle. As a result, overall performance and link utilization are low and prohibit us from scaling the All-to-all to a large number of dense-GPU nodes. In this scenario, even though the amount of data transferred over links remains the same, the performance can be significantly varied depending on how the communication is being scheduled. This requires an in-depth analysis of existing personalized and non-personalized All-to-all algorithms and optimized designs for modern dense-GPU systems. This paper addresses the following challenge: **How to design hierarchical, adaptive, and high-utilization personalized and non-personalized All-to-all communication algorithms on dense-GPU systems?**

B. Contribution

In this paper, we present a comprehensive analysis of existing All-to-all and Allgather algorithms to identify their deficiencies in dense-GPU systems with asymmetric interconnects. Based on the extensive evaluation of existing algorithms, we propose advanced algorithms to decompose the hierarchical All-to-all algorithm and make use of non-blocking primitives to enable traffic overlap between NVLink, PCIe, and IB. As a result, the proposed design improves the All-to-all and Allgather performance for large message sizes on dense-GPU systems, including Lassen and Summit. This paper makes the following contributions:

1) Comprehensive evaluation of existing All-to-all and Allgather communication algorithms within the state-of-the-art MPI library (Section IV-A)
2) Propose a new All-to-all algorithm that decomposes the pair-wise communication pattern and utilizes the interconnects such as NVLink available between GPUs to minimize the link idle time (Sections III-A and III-B)
3) Propose an advanced hierarchical algorithm that achieves high overlap between inter- and intra-node communications to improve performance of All-to-all and Allgather communication (Section III-C)
4) Conduct performance evaluation of All-to-all and Allgather collectives with the proposed designs and various CUDA-aware communication Libraries (MVAPICH2-GDR, Spectrum-MPI, NCCL, and OpenMPI) using micro-benchmarks, a 3D FFT application kernel, and KFAC distributed DNN optimizer (Section IV)

II. Background

A. Interconnect Technology

In optimizing an algorithm for dense GPU systems, we utilize the high-bandwidth NVLink interconnects available on systems such as Lassen, and Summit. In doing so, we incorporate the topology of a system, and the interconnects available to ensure that the design takes advantage of the resources available. These interconnects are shown for the Summit cluster in Figure 1. These interconnects also apply to the topology of the Lassen system. The following interconnects are available on the systems we evaluated the designs on:

- **NVLink** is an interconnect provided by NVIDIA that is available between GPU and GPU and also between CPU and GPU on the Summit and Lassen systems.
- The **X-Bus** is an IBM interconnect between POWER9 CPUs on Summit, and Lassen
- The Mellanox **InfiniBand** network connects nodes via dual-rail EDR for the Summit and Lassen clusters. These system topologies are organized as a fat tree. [17]
- **PCIe** is a high-speed universal bus standard that supports many parallel lanes and is used to connect the CPU to the InfiniBand network interface card (NIC) on the Summit and Lassen systems.

B. NVIDIA GPUDirect Technology

NVIDIA GPUDirect [18] allows the CPU, GPU, and network devices to directly read and write device memory, which eliminates the need to copy memory multiple times. Various other features are also supported by GPUDirect, including peer-to-peer transfers and remote direct memory access (RDMA). External devices such as Mellanox Infiniband Host Channel Adapters (HCAs) are able to directly access GPU memory without involving the CPU, which reduces latency further by eliminating the need for copies to CUDA host memory. The CUDA inter-process communication (IPC) interface allows a process to expose its GPU buffer to a remote process. A process may achieve this by creating an IPC handle on its buffer, then sending this handle to the remote process. Then, the remote process can map this buffer into its own address space and directly call operations (like cudaMemcpy) on it.

C. CUDA-aware Communication Libraries

MPI is a parallel programming standard that enables processes to communicate with each other. MPI libraries such as SpectrumMPI [19], OpenMPI [20], and MVAPICH2 [21] provide support for heterogeneous systems with CUDA-aware features at the MPI level. By exploiting efficient GPU-based communication schemes such as staging, CUDA Inter-Process Communication (IPC), and GPUDirect RDMA, MPI libraries are able to provide superior performance across different combinations of GPUs and interconnects [12]. The NVIDIA Collectives Communication Library (NCCL) is a library with optimized collective communication patterns for NVIDIA GPUs [22].

D. Personalized and Non-Personalized all-to-all Collective Communication

There are various forms of communication within the MPI standard [23], ranging from one-sided communication to point-to-point and collective communication. In this paper, we delve deeper into all-to-all and allgather based collective communication.

TABLE I
EXISTING AND PROPOSED ALGORITHMS FOR PERSONALIZED AND NON-PERSONALIZED ALL-TO-ALL COLLECTIVE COMMUNICATION

Algorithm	Used for Personalized or Non-Personalized	Optimal Message Range	Link Utilization	Overlap	Adaptive	Hierarchical
Bruck [13]	Both	Small	✗	✗	✗	✗
Scatter Destination [14]	Personalized	Medium	✗	✗	✗	✗
Pairwise [15]	Personalized	Medium	✗	✗	✗	✗
Inplace	Personalized	Medium	✗	✗	✗	✗
Direct	Non-Personalized	Medium	✗	✗	✗	✗
Recursive Doubling (RD) [16]	Non-Personalized	Medium	✗	✗	✗	✗
Ring [16]	Non-Personalized	Large	✗	✗	✗	✗
Proposed Hierarchical + Overlap	Non-Personalized	Large	✓	✓	✓	✓
Proposed Eager	Personalized	Large	✓	✓	✓	✓
Proposed Hierarchical	Personalized	Large	✓	✓	✓	✓

1) Non-Personalized all-to-all: Non-personalized all-to-all, often referred to as **MPI_Allgather**, is an MPI collective communication pattern that acts as a combination of MPI_Gather followed by MPI_Bcast. All processes gather the data from every other process in rank order. At the end of the operation, every process would have exchanged data with all other processes.

2) Personalized all-to-all: Personalized all-to-all, referred to as **MPI_Alltoall**, is similar to MPIAllgather in that all processes are sharing data with each other. However, every process sends distinct data to other processes making all-to-all a personalized collective communication pattern (i.e., every process will send a different chunk of data to every other process).

E. Distributed K-FAC for Deep Learning Training

The Kronecker-factored Approximate Curvature (K-FAC) is a second-order optimization method that may be used as a preconditioner to the gradient. The gradient can then be updated by a standard optimizer such as stochastic gradient descent (SGD). By using second-order information on the DNN weights, K-FAC can reduce the number of training iterations required for convergence while reducing the frequency of communication [24]. Despite the reduced frequency of communication, however, the communication volume is greatly increased, requiring two calls to MPI_Allreduce and one call to MPI_Allgather at each K-FAC precondition step. These frequent calls to MPI_Allgather make up a noticeable portion of the communication volume and would benefit from improved GPU-Direct designs.

III. PROPOSED DESIGNS

In this section, we thoroughly detail the proposed designs for both personalized and non-personalized all-to-all collectives.

A. Proposed Non-Personalized all-to-all (allgather) Design

The proposed design aims to improve the performance of allgather collective communication algorithms for dense GPU systems by leveraging the full potential of the interconnect technology available on these clusters. In particular, it aims to utilize the high bandwidth of NVLink interconnect between GPUs, yielding faster communication of large data in order to overlap and hide intra-node communication. On large-scale dense GPU systems, where GPUs are connected by NVLink on a node, a different communication pattern can be implemented for intra-node as opposed to inter-node. These communication patterns can be overlapped to occur simultaneously while maintaining program order. It allows for restructuring the communication into a hierarchical setup such that GPUs on a node are communicating directly with the leader process on the node as the inter-node communication is progressing. This enables link utilization by ensuring that NVLink between GPUs is not idle during the communication setup whenever possible.

We propose a collective allgather algorithm that implements a tree-based inter-node communication algorithm in the form of recursive doubling. It simultaneously implements a direct gather (post data through Isend/Irecv to the leader) and broadcast (post data through Isend/Irecv to the GPUs on the node) intra-node communication algorithm from a leader process on the node. This algorithm utilizes a 2-level communicator to distinguish between the intra-node steps and the inter-node steps to establish communication amongst all the leaders for every node involved in the communicator and between the leader and local processes on the node. Figure 2 depicts the proposed algorithm for two nodes. This depicts the first step of the communication where GPU0, the node leader for Node 0, is communicating its data with GPU4, the node leader of Node 1, while simultaneously receiving data from intra-node processes for the next inter-node exchange between the leaders. When the node leader has acquired data from another node, it then sends this data to each of the processes on the node.

An allgather Recursive Doubling algorithm requires the logarithm of the number of processes in steps to complete. In this design, we are utilizing recursive doubling as the inter-node communication mechanism, making the communication bound by the log of the number of nodes instead. In overlapping the intra-node communication with the inter-node communication, we establish the following (**N** is number of nodes, **M** is message size, B_{IB} is the bandwidth of IB, t_s is the added

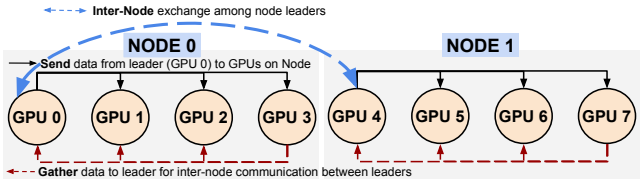


Fig. 2. PROPOSED Allgather on 2 Nodes (8 Processes)

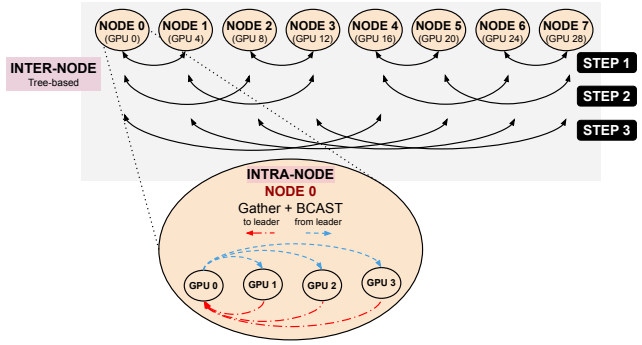


Fig. 3. PROPOSED Allgather on 8 Nodes (32 Processes)

overhead time in seconds, and **GPN** is GPUs per Node):

$$\textbf{Proposed} = log(N) * (\frac{M}{B_{IB}} + t_s) * (GPN - 1) \quad (3)$$

Figure 3 depicts the proposed design on a system with four GPUs per node for an example using 8 nodes. The example depicted in Figure 3 shows the various steps of the algorithm for an eight-node problem with four processes per node. The figure depicts each of the steps of the RD communication between nodes and highlights the intra-node communication that is occurring simultaneously. In this example:

1) Inter-Node Communicator: leader rank of Node 0, **GPU0**, is doing an initial exchange of data with the leader rank of Node 1, **GPU4**. Intra-Node Communicator: Simultaneously, **GPU1** is sending data to **GPU0**
2) Upon receiving the data from GPU4, GPU0 sends that data to each of the processes on the node
3) Upon receiving the data from GPU1, GPU0 will send it across the node to **GPU4** for it to communicate to the local processes on Node 1. This process continues until each of the processes have now gathered all the data from every other process.

Note that in Step 2 above, on Systems such as Summit or Lassen, across socket communication is limited by the bandwidth of the XBus. It would be expected in this scenario to witness more performance gain compared to existing algorithms with no communication overlap due to hiding the overhead of communicating across the socket on these systems. As an example, in Figure 3, when GPU3 attempts to communicate with GPU0, there will be contention over the XBus as a result of the communication happening between GPU2 and GPU0, and vice versa. Algorithm 1 details the implementation associated with this design.

Algorithm 1 PROPOSED: Hierarchical Allgather + Overlap

```
size ← size of communicator
gpn ← GPUs per Node
lrank ← leader rank
n ← Number of Nodes
if lrank then
  for i = 1→gpn do
    MPI_Isend() //Sends from lrank to local GPUs
    MPI_Irecv() //gather data from local GPUs
  end for
  while offset<n do
    dst = (rank/gpn) ∧ offset
    root = dst>>i
    for j = 1→(gpn * offset) do
      if j<gpn then
        MPI_Wait() //wait for intra-node data before sending over inter-node
        MPI_Isend() //send next node data to leader
      end if
      MPI_Irecv() //exchange data between lranks
      MPI_Isend() //send from lrank to local GPUs
    end for
    offset<< = 1
  end while
else
  MPI_Isend() //send data from sendbuf to node leader
  for i = 1→gpn do
    MPI_Irecv() //initial data to lrank from local GPUs
  end for
  while offset<n do
    dst = (rank/gpn) ∧ offset
    root = dst>>i
    for j = 1→(gpn * offset) do
      MPI_Irecv() //recv inter-node data from leader
    end for
    offset<< = 1
  end while
end if
MPI_Waitall()
```

B. Proposed Eager Personalized All-to-all Design

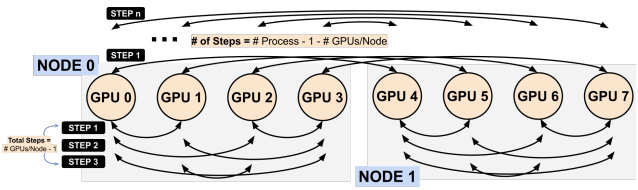


Fig. 4. PROPOSED Eager All-to-all on 2 Nodes (8 Processes)

In Figure 4, we propose an algorithm for all-to-all that utilizes pairwise communication with additional optimizations to hide intra-node communication. As shown in Figure 4, we decompose a pairwise all-to-all algorithm and overlap the

communication within the node with the initial communication across nodes (i.e. in the first step, GPU0 communicates with GPU1 to exchange data while simultaneously exchanging data across the node with GPU4). The benefits of this algorithm are in its ability to hide the latency of intra-node communication by overlapping the intra- and inter-node communication in the first few steps. The disadvantage of this algorithm is that it could lead to contention over the inter-node interconnect, IB as every process is still trying to communicate with every process. For example, while GPU0 is sending to GPU4, GPU1 is waiting on this request to complete in order to communicate with GPU5 and so forth. The details of this implementation can be found in Algorithm 2. This problem is addressed in the next proposed design in Section III-C that utilizes the interconnects available amongst GPUs and a hierarchical implementation in order to minimize idle NVlinks and establish a two-level communication that handles the contention over IB.

Algorithm 2 PROPOSED: Eager Personalized all-to-all

```
1: size ← size of communicator
2: gpn ← GPUs per Node
3: for i = 1→size do
4:    /* Determine src for exchange */
5:    src←rank∧i
6:    MPI_Irecv(src_offset, src)
7: end for
8: for i = 1→(size − gpn) do
9:    /* Determine dest for exchange */
10:   dst1←rank∧i
11:   dst2←rank∧(i + [gpn − 1])
12:   if i<(size − gpn) then
13:      MPI_Isend(dst2_offset, dst2)
14:   end if
15:   if i<gpn then
16:      MPI_Isend(dst1_offset, dst1)
17:   end if
18: end for
19: MPI_Waitall()
```

C. Proposed Hierarchical and Adaptive All-to-all Design

Figures 5-8 depict a step-by-step execution of the proposed design for all-to-all on dense GPU systems. It is a 2-level all-to-all algorithm doing a pairwise communication amongst leaders across nodes and doing a direct gather and scatter of chunks of data sent and received at each step to local processes on the node. The design takes into account the links available on such a system in order to ensure minimal interconnects are idle during the communication. In earlier all-to-all design implementations such as pairwise or scatter-destination, the performance of the algorithm is bound by the IB interconnect as every process communicates with every other process. In this scenario, every process is waiting until the completion of the process acquiring the IB. In the proposed design, we address this bottleneck by modifying the processes that will be communicating with each other and utilize a **hierarchical** approach that entails only the leader would be communicating over IB while intra-processes utilize the NVLink available between GPUs. We evaluate the proposed algorithms in an **adaptive** manner. The comprehensive evaluation in section IV utilizes the adaptive designs such that for every message size, the best algorithm for that message size is selected. Different algorithms are appropriate and optimized for varying message sizes, by adaptively selecting algorithms based on performance in relation to message size, we optimize the performance across a larger message range. The algorithms are beneficial for varying message ranges across different clusters. In their utilization of links between GPUs, they demonstrate improved performance on dense GPU systems connected by NVLink. This is depicted in the evaluations in section IV.

In the first step, every process local to a node sends the relevant data to each of the processes on the node (i.e. GPUs 0-3 communicate the first chunk of data to each other with direct send and receives to each process over NVLink). At this point, GPU 1 posts the chunk of data that is relevant to the node GPU 0 is communicating with to a temporary buffer in GPU 0. Simultaneously, GPU 0 is communicating the chunk of data from its send buffer to GPU 4. When GPU 4 and GPU 0 receive their respective data chunk from each other, they scatter this data to each of the processes on the node with a direct send and receive to each of the local processes. This pattern continues such that every process on a node has sent its data to the leader on the node and the leaders of each node communicate with each other in a pairwise fashion. The implementation details are outlined in Algorithm 3.

To breakdown each of the steps presented in Figures 5-8, we execute the algorithm in the following way (note the communication overlap that is done simultaneously to ensure available links are not idle during execution at each step):

1) GPU0 sends index 1 of buffer to GPU1, index 2 of buffer to GPU2, and so forth. Likewise, every other process communicates with the local processes on the node to establish intra-node all-to-all amongst processes. (**starts in step 1 of execution** (Figure 5))
2) GPU0 sends initial chunk (index 4-7 of buffer) to GPU 4 and receives initial chunk (index 0-3 of GPU4's buffer) over the Infiniband interconnect between the nodes. (**starts in step 1 of execution**)
3) GPU1 sends initial chunk (index 4-7 of buffer) to temporary buffer at GPU0 indexed appropriately. Other processes on the node communicate the same indexed chunk to the temporary buffer at GPU0 over the NVLinks available between the GPUs. On the Summit (Figure1) or Lassen system, cross-socket communication is bound by X-Bus.
4) GPU0 copies index 0 of chunk received from GPU4 to its receive buffer and scatters the data it received from GPU4 amongst the local processes through a direct send and receive (i.e. send index 1 of chunk to GPU1 and so forth).
5) GPU0 waits to receive the first chunk from GPU1 and sends it over Infiniband to GPU4. This happens repeatedly until the appropriate chunk of data from every

local process is communicated over the node. (**starts in step 1 of execution**)

6) As shown in Figure 6, the first simultaneous execution between the first pair of leaders is repeated with the next intra-GPU processes data, and so forth (Figure 7).
7) The inter-node communication utilizes pairwise logic such that Node 0 and Node 1 communicate their data. Then this entire process repeats with different indexing of each of the buffers when Node 0 and Node 2 communicate whilst Node 1 and Node 3 are communicating. Figure 8 depicts the inter-node communication pattern between node leaders on an example with 32 GPUs and 8 nodes.
8) After every receive across the node, the leader node shares the data across the local processes.

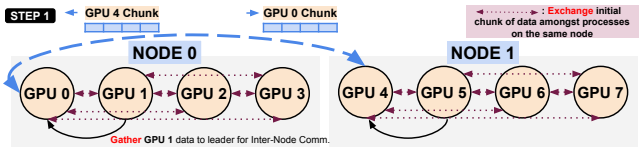


Fig. 5. PROPOSED Hierarchical All-to-all: **Step 1** of Execution

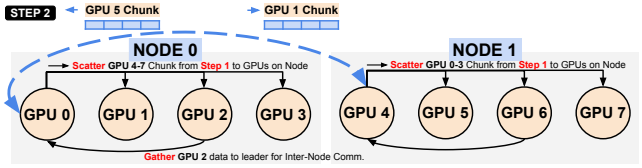


Fig. 6. PROPOSED Hierarchical All-to-all: **Step 2** of Execution

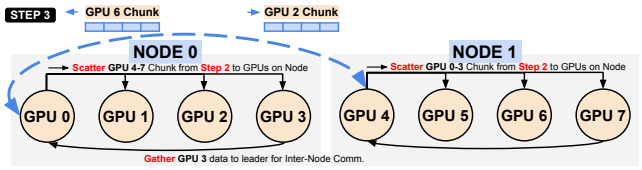


Fig. 7. PROPOSED Hierarchical All-to-all: **Step 3** of Execution

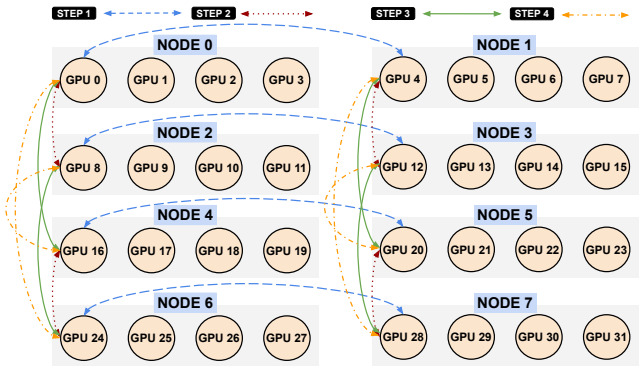


Fig. 8. PROPOSED Hierarchical All-to-all: Inter-Node Communication Pattern Between Node Leaders

Algorithm 3 PROPOSED Hierarchical Personalized all-to-all

```
size ← size of communicator
gpn ← GPUs per Node
lrank ← Rank of node leader
n ← Number of Nodes
tmp_buf
for i = 0→gpn do
    /* recv initial data from local GPUs and send initial data to local GPUs */
    src = dst = i
    MPI_Irecv(src)
    MPI_Isend(dst)
end for
if lrank then
    for i = 1→n do
        for j = 0→gpn do
            define recv_offset← offset in leader recvbuf
            MPI_Irecv() //recv from inter-node leader
            define gather_offset ← offset in leader recvbuf
            MPI_Irecv() //gather data from local GPUs
        end for
        for j = 0→gpn do
            define scat_offset
            MPI_Isend() //send initial chunk from leader
            MPI_Isend() //scatter data to local GPUs
        end for
    end for
else
    for i = 1→n do
        dst←lrank∧i
        MPI_Isend() //data from local GPUs to leader
        for j = 0→gpn do
            offset = dst * gpn
            MPI_Irecv() //scatter inter-node data from lrank
            offset++
        end for
    end for
end if
MPI_Waitall()
```

IV. Evaluation

In this section, we show the benchmark-level and application-level performance of existing algorithms and other CUDA-aware MPI Libraries compared to using the proposed algorithms. We utilize profiling tools including mpiP [25], nvprof [26], and INAM with profiling support for GPU clusters [27] to acquire a deeper insight into the communication pattern used in applications.

A. Existing Designs

Details related to existing designs for non-personalized all-to-all communication can be found in [15]. Many of these designs are also applied for personalized all-to-all communication. We evaluate our proposed designs to the performance

of existing designs for each of these collective communication patterns. Specifically, for non-personalized all-to-all, we evaluate against the following algorithms: Bruck, Recursive Doubling (RD), Ring, and Direct. For personalized all-to-all, we evaluate against RD, Scatter-Destination, Pairwise, and Inplace. These algorithms specifically are bound by InfiniBand for inter-node communication and do not efficiently utilize the links available to achieve optimal performance. We address these drawbacks and challenges in the proposed designs to develop hierarchical and adaptive all-to-all algorithms suitable for dense GPU systems.

B. Benchmark-Level Performance

The algorithms were analyzed at the benchmark level using OSU Micro-benchmarks(OMB) [28] v5.7. We evaluated the performance of the existing all-to-all algorithms within MVAPICH2-GDR 2.3.4 compared to the proposed algorithms and evaluated the performance of all-to-all when using various MPI libraries. OMB supports evaluating point-to-point, multi-pair, and collective CUDA-aware communication using GPU buffers. We used the *osu_alltoall* and *osu_allgather* tests to determine the latency of personalized and non-personalized all-to-all communication, respectively, when the buffers used for communications are located on the device.

The existing algorithms in the MVAPICH2-GDR library mentioned in Section IV-A were evaluated against the proposed designs over 64 Nodes with 4 GPUs per node (256 GPUs) on the Lassen System and over 64 Nodes with 6 GPUs per node (384 GPUs) on the Summit system.

Figure 9 shows the results of running the proposed algorithm compared to existing algorithms in the MVAPICH2-GDR Library on the Lassen system. We scaled the algorithms from 32 GPUs to 256 GPUs and showed benefits for the proposed algorithm beyond 64KB. We typically observed vast performance gain in the range of 20 - 30% at 8KB message size communication and beyond. In Figure 9(d) running the proposed algorithm on 256 GPUs for 1 MB message size, we see approximately a 39% performance gain in the proposed algorithm compared to the best performing existing algorithm for that configuration, scatter-destination. We also evaluated the algorithm on the Summit system. Note that the Summit system has 6 GPUs per node as opposed to 4 GPUs per node on Lassen. Similar to the results presented earlier, when we run the algorithm on 256 GPUs for 2MB we see a 35% enhancement in performance compared to the best performing existing algorithm for this case on Summit, Pairwise. The varying upper limit message ranges presented for each configuration are a result of memory limits when running OMB at scale.

The current implementation of the Bruck algorithm in MVAPICH2-GDR 2.3.4 does not support communication between GPU buffers, excluding this algorithm from the comparisons we make between the existing designs. There were also some limitations to scaling the existing RD algorithm which is demonstrated in the graphs with missing values associated with RD.

Overall, we see more improvement in personalized all-to-all latency on the Summit system (Figure 10) compared to the Lassen system (Figure 9) due to the number of GPUs per node. In overlapping the intra-node communication with the inter-node communication, we hide the contention caused by the X-Bus when processes are communicating across the socket at the initial exchange. Since more processes will be doing so on the Summit system due to the greater number of GPUs per node, we see more benefit in applying this algorithm on such a dense GPU system.

Figure 11 shows the results of existing allgather algorithms compared to the proposed algorithm. The upper bound of the message range shown as we scale up is limited by the memory limitation of running OMB at a large scale with a non-personalized or personalized all-to-all collective algorithm. We see increasing performance benefits ranging from 9 - 18% improvement over the best existing algorithm, ring, for each of the configurations shown. The proposed design overlaps the intra-node communication to hide the latency. The communication overlap hides the intra-node communication and shows performance gain at this message range.

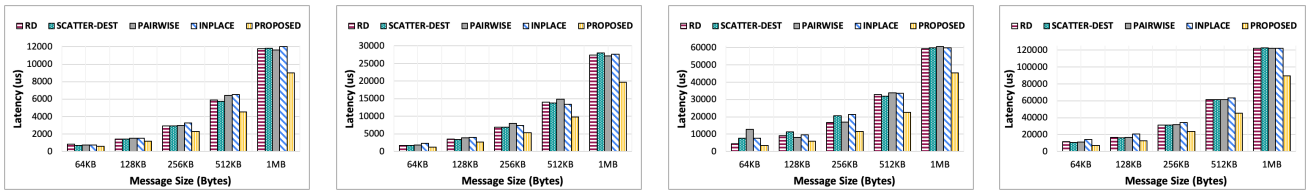
We also evaluated other existing MPI libraries such as Spectrum-MPI which is deployed on the systems that we are running these experiments on (Summit and Lassen). We use Spectrum-MPI 10.3.0.1 to compare with the available and proposed designs. Other CUDA-aware MPI libraries used in the evaluations made include OpenMPI 4.0.4 with UCX 1.8.1, and NCCL 2.7.8.1. We evaluated these libraries using CUDA version 10.1.243 on the Summit and Lassen systems with MLNX-OFED version 4.7. The results are presented in 12. Overall, there is a 30% better latency associated with the proposed designs for large message sizes compared to other state-of-the-art MPI libraries for device performance of all-to-all. Through the recent release of NCCL including support for point-to-point operations, an all-to-all operation can be done using NCCL with a loop of multiple send and receive operations to and from all the peers. Due to limitations in the message range printed with the all-to-all nccl-tests however, these numbers are not shown in this comparison. In Figure 13, we demonstrate 20-60% better latency for allgather with the proposed schemes.

C. 3D-FFT Application Kernel

In this section, we evaluated the performance of each of the existing all-to-all algorithms using an All-to-All FFT Personalized Exchange. This kernel imitates the FFT computation of an application requiring *MPI_Alltoall* communication. Figure 14 depicts the results of using the existing algorithms to run the FFT Suite on the Lassen System for 64, 128, and 256 GPUs. We observe a 15-20% range of improvement in performance across the message ranges shown over existing algorithms.

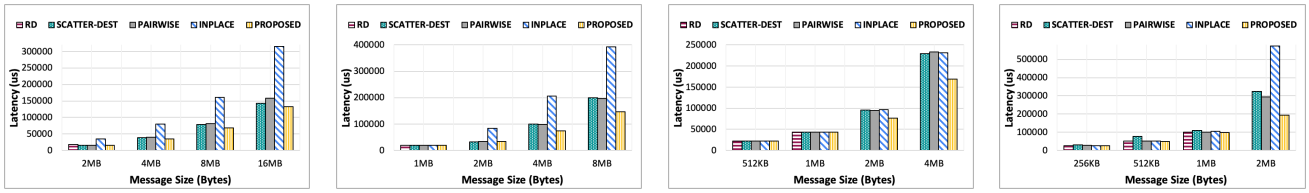
D. Distributed K-FAC

In this section, we observe the benefits in distributed K-FAC performance from the improved allgather algorithm. To compare the communication performances, we recorded and



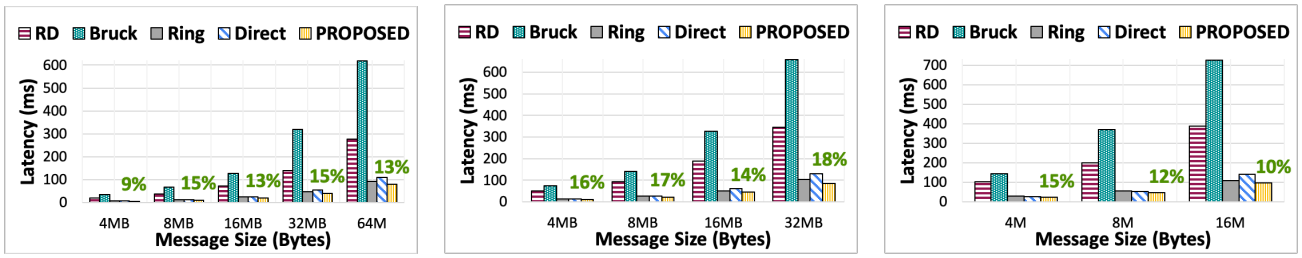
(a) 8 Nodes - 32 GPUs (b) 16 Nodes - 64 GPUs (c) 32 Nodes - 128 GPUs (d) 64 Nodes - 256 GPUs

Fig. 9. Comparison of Different Node/PPN configurations of Personalized all-to-all Algorithms on the Lassen System



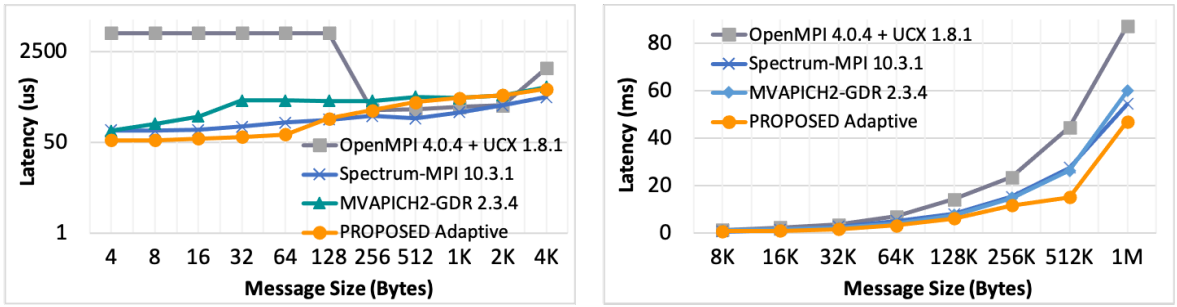
(a) 8 Nodes - 48 GPUs (b) 16 Nodes - 96 GPUs (c) 32 Nodes - 192 GPUs (d) 64 Nodes - 384 GPUs

Fig. 10. Comparison of Different Node/PPN configurations of Personalized all-to-all Algorithms on the Summit System



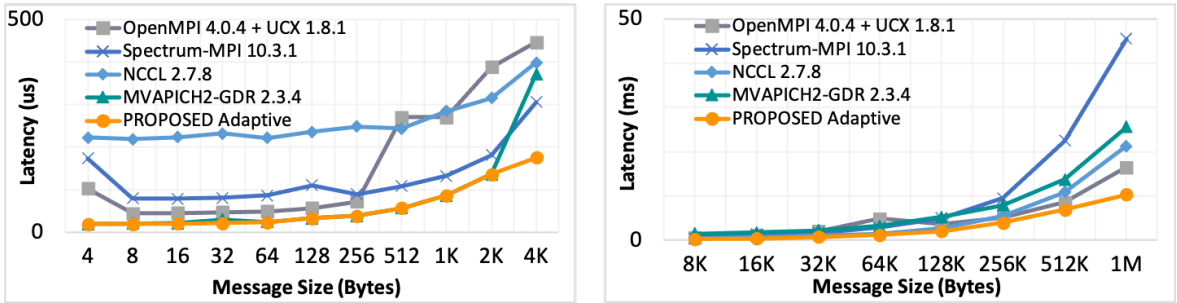
(a) 4 Nodes - 16 GPUs (b) 8 Nodes - 32 GPUs (c) 16 Nodes - 64 GPUs

Fig. 11. Comparison of Different Algorithms for Allgather Algorithms on the Lassen System. The percentages indicate the improvement the proposed scheme has over the best of the state-of-the-art algorithms.



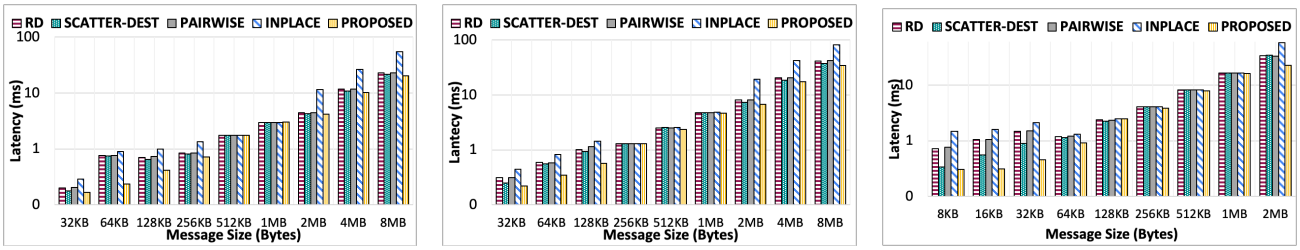
(a) 32 Nodes - 128 GPUs (Medium) (b) 32 Nodes - 128 GPUs (Large)

Fig. 12. **Personalized all-to-all** Comparison using MVAPICH2-GDR 2.3.4, Spectrum-MPI 10.3.1, and OpenMPI 4.0.4 + UCX 1.8 on the Lassen System



(a) 32 Nodes - 128 GPUs (Small Messages) (b) 32 Nodes - 128 GPUs (Large Messages)

Fig. 13. **Allgather** Comparison using MVAPICH2-GDR 2.3.4, Spectrum-MPI 10.3.1, OpenMPI 4.0.4 + UCX 1.8, and NCCL 2.7.8 on the Lassen System



(a) 16 Nodes - 64 GPUs (b) 32 Nodes - 128 GPUs (c) 64 Nodes - 256 GPUs

Fig. 14. Personalized all-to-all Algorithms for 3D FFT application kernel

averaged the total time for each epoch of training on ImageNet [29]. The results are depicted in Figure 15, and show an improvement of up to 8% in comparison to the best performing state-of-the-art library at each GPU count depicted in the figure. KFAC utilizes other MPI operations as well such as MPI_Allreduce. In purely optimizing the MPI_Allgather schemes that are then utilized here, the reduced percentage of the time the application spends in MPI_Allgather leads to overall enhanced training time.

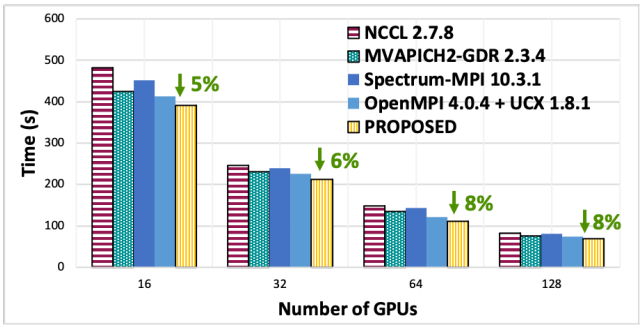


Fig. 15. K-FAC Allgather Evaluation on the Lassen System. The percentages indicate the improvement the proposed scheme has over the best performing state-of-the-art library for each GPU count.

V. Related Work

While there has been effort spent on developing and optimizing allgather algorithms for host buffers [30] [15], few studies exist on allgather algorithms for GPU buffers. While existing all-to-all broadcast collective algorithms for GPUs rely on pipelining [31] [14] or non-blocking methods [32] to hide the communication overhead of GPU collectives, no prior all-to-all collective algorithm has taken advantage of the high-bandwidth NVLink interconnects on modern HPC systems. Leveraging the network topology to design better collective algorithms has demonstrated superior performance on host buffers [33] [34]. In [35], Kielmann et al. account for the hierarchical structure of network topology to propose MagPIe for grid-aware MPI implementations. Despite the shortcomings of previous algorithms on modern GPU clusters, an increasing number of applications rely upon high-bandwidth all-to-all broadcast GPU collectives for data analytics [36] and deep learning [37]. While improvements for topology-aware [38] [39] [40] and memory-aware [41] collective algorithms on GPU buffers have been made, modern network topologies require that new methods be proposed and extended to achieve enhanced performance on dense NVLink GPU systems.

VI. Conclusion

As the trend in deploying dense-GPU hardware configurations for the next-generation HPC systems continues, it is important to revisit and study the GPU-based communication schemes on such systems. In this paper, we conduct a thorough study of personalized and non-personalized all-to-all collective communication. Based on our analysis, we propose adaptive schemes with hierarchical All-to-all and Allgather algorithms that can achieve high overlap between intra- and inter-node communication on large-scale dense GPU systems. As a result, the proposed design demonstrates up to 22% and 30% improvements for personalized and non-personalized alltoall, respectively, on Summit and Lassen systems for large All-to-all data transfer at benchmark-level experiments.

The proposed designs were evaluated against existing designs within the MVAPICH2-GDR library and with state-of-the-art communication libraries such as NCCL, SpectrumMPI, and OpenMPI. At the application level, the performance evaluation with a three-dimensional FFT application kernel indicates that the proposed schemes for personalized all-to-all communication can yield 15-25% lower execution time on the Lassen system on up to 256 GPUs. The non-personalized all-to-all designs demonstrate improvement in performance up to 8% on 128 GPUs on the Lassen system for distributed K-FAC used in Deep Learning training. The proposed designs address these challenges in order to yield performance enhancements at both the benchmark-level and the application-level.

In the future, we plan to evaluate the proposed designs with more applications that heavily rely on GPU-based All-to-all communication [42], [43] to demonstrate performance enhancements that can be gained through hierarchical schemes that utilize link bandwidth on dense GPU systems.

References

[1] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, "TOP 500 Supercomputer Sites," http://www.top500.org, 1993.
[2] ORNL, "Frontier," https://www.olcf.ornl.gov/frontier/.
[3] Intel, "Aurora supercomputer," https://www.intel.com/content/www/us/en/high-performance-computing/supercomputing/exascale-computing.html.
[4] S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, and D. K. Panda, "Efficient Inter-node MPI Communication Using GPUDirect RDMA for InfiniBand Clusters With NVIDIA GPUs," in *Parallel Processing (ICPP), 2013 42nd International Conference on*. IEEE, 2013.

[5] R. Shi, S. Potluri, K. Hamidouche, J. Perkins, M. Li, D. Rossetti, and D. K. Panda, "Designing Efficient Small Message Transfer Mechanism for Inter-node MPI Communication on InfiniBand GPU Clusters," in *2014 21st International Conference on High Performance Computing (HiPC)*, Dec 2014, pp. 1–10.

[6] S. S. Sharkawi and G. A. Chochia, "Communication protocol optimization for enhanced GPU performance," *IBM Journal of Research and Development*, vol. 64, no. 3/4, pp. 9:1–9:9, 2020.

[7] X. Luo, W. Wu, G. Bosilca, T. Patinyasakdikul, L. Wang, and J. Dongarra, "ADAPT: An Event-based Adaptive Collective Communication Framework," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: ACM, 2018, pp. 118–130.

[8] C.-H. Chu, X. Lu, A. A. Awan, H. Subramoni, B. Elton, and D. K. Panda, "Exploiting Hardware Multicast and GPUDirect RDMA for Efficient Broadcast," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 575–588, March 2019.

[9] C.-H. Chu, P. Kousha, A. A. Awan, K. S. Khorassani, H. Subramoni, and D. K. Panda, " NV-Group: Link-Efficient Reduction for Distributed Deep Learning on Modern Dense GPU System," in *The 34th ACM International Conference on Supercomputing (ICS-2020)*, 2020.

[10] T. Ben-Nun and T. Hoefler, "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis," *ACM Comput. Surv.*, vol. 52, no. 4, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3320060

[11] M. Naumov, J. Kim, D. Mudigere, S. Sridharan, X. Wang, W. Zhao, S. Yilmaz, C. Kim, H. Yuen, M. Ozdal, K. Nair, I. Gao, B.-Y. Su, J. Yang, and M. Smelyanskiy, "Deep Learning Training in Facebook Data Centers: Design of Scale-up and Scale-out Systems," *arXiv preprint arXiv:2003.09518*, 2020.

[12] K. S. Khorassani, C.-H. Chu, H. Subramoni, and D. K. Panda, "Performance Evaluation of MPI Libraries on GPU-enabled OpenPOWER Architectures: Early Experiences," in *International Workshop on OpenPOWER for HPC (IWOPH 19) at the 2019 ISC High Performance Conference*, 2019.

[13] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby, "Efficient algorithms for all-to-all communications in multiport message-passing systems," *IEEE Transactions on parallel and distributed systems*, vol. 8, no. 11, pp. 1143–1156, 1997.

[14] A. K. Singh, S. Potluri, H. Wang, K. Kandalla, S. Sur, and D. K. Panda, "Mpi alltoall personalized exchange on gpgpu clusters: Design alternatives and benefit," in *2011 IEEE International Conference on Cluster Computing*, Sep. 2011, pp. 420–427.

[15] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in mpich," *The International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, 2005. [Online]. Available: https://doi.org/10.1177/1094342005051521

[16] R. Thakur and W. D. Gropp, "Improving the performance of collective operations in mpich," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, J. Dongarra, D. Laforenza, and S. Orlando, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[17] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, 1985.

[18] NVIDIA, "NVIDIA GPUDirect," https://developer.nvidia.com/gpudirect.

[19] IBM, "IBM Spectrum MPI: Accelerating high-performance application parallelization," https://www.ibm.com/us-en/marketplace/spectrum-mpi, 2018, Accessed: March 16, 2021.

[20] Open MPI, "Open MPI: Open Source High Performance Computing," https://www.open-mpi.org/, 2004, Accessed: March 16, 2021.

[21] D. K. Panda, H. Subramoni, C.-H. Chu, and M. Bayatpour, "The mvapich project: Transforming research into high-performance mpi library for hpc community," *Journal of Computational Science*, p. 101208, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750320305093

[22] NVIDIA, "NCCL," https://github.com/NVIDIA/nccl, 2016.

[23] Message Passing Interface Forum, http://www.mpi-forum.org/.

[24] J. G. Pauloski, Z. Zhang, L. Huang, W. Xu, and I. T. Foster, "Convolutional neural network training with distributed k-fac," 2020.

[25] LLNL, "LLNL/mpiP: A light-weight MPI profiler," https://github.com/LLNL/mpiP/, Accessed: March 16, 2021.

[26] NVIDIA, "NVIDIA profiling tools," https://docs.nvidia.com/cuda/profiler-users-guide/index.html.

[27] P. Kousha, B. Ramesh, K. Kandadi Suresh, C. Chu, A. Jain, N. Sarkauskas, H. Subramoni, and D. K. Panda, "Designing a profiling and visualization tool for scalable and in-depth analysis of high-performance gpu clusters," in *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, 2019.

[28] D. Bureddy, H. Wang, A. Venkatesh, S. Potluri, and D. K. Panda, "Omb-gpu: A micro-benchmark suite for evaluating mpi libraries on gpu clusters," in *Proceedings of the 19th European Conference on Recent Advances in the Message Passing Interface*, ser. EuroMPI'12, 2012.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[30] U. Wickramasinghe and A. Lumsdaine, "A survey of methods for collective communication optimization and tuning," *CoRR*, vol. abs/1611.06334, 2016. [Online]. Available: http://arxiv.org/abs/1611.06334

[31] A. K. Singh, "Optimizing all-to-all and allgather communications on gpgpu clusters," Ph.D. dissertation, The Ohio State University, 2012.

[32] A. Venkatesh, K. Hamidouche, H. Subramoni, and D. K. Panda, "Offloaded gpu collectives using core-direct and cuda capabilities on infiniband clusters," in *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*, Dec 2015, pp. 234–243.

[33] K. Kandalla, H. Subramoni, A. Vishnu, and D. K. Panda, "Designing topology-aware collective communication algorithms for large scale infiniband clusters: Case studies with scatter and gather," in *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)*, April 2010, pp. 1–8.

[34] T. Ma, T. Herault, G. Bosilca, and J. J. Dongarra, "Process distance-aware adaptive mpi collective communications," in *2011 IEEE International Conference on Cluster Computing*, Sep. 2011, pp. 196–204.

[35] T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang, "Magpie: Mpi's collective communication operations for clustered wide area systems," in *Proceedings of the Seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 131–140. [Online]. Available: https://doi.org/10.1145/301104.301116

[36] T. B. Rolinger, T. A. Simon, and C. D. Krieger, "An empirical evaluation of allgatherv on multi-gpu systems," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2018, pp. 123–132.

[37] M. Naumov, D. Mudigere, H. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C. Wu, A. G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, and M. Smelyanskiy, "Deep learning recommendation model for personalization and recommendation systems," *CoRR*, vol. abs/1906.00091, 2019. [Online]. Available: http://arxiv.org/abs/1906.00091

[38] I. Faraji and A. Afsahi, "Design considerations for gpu-aware collective communications in mpi," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 17, p. e4667, 2018, e4667 cpe.4667. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4667

[39] R. Kobus, D. Jünger, C. Hundt, and B. Schmidt, "Gossip: Efficient communication primitives for multi-gpu systems," in *Proceedings of the 48th International Conference on Parallel Processing*, ser. ICPP 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3337821.3337889

[40] N. T. Karonis, B. R. de Supinski, I. Foster, W. Gropp, E. Lusk, and J. Bresnahan, "Exploiting hierarchy in parallel computer networks to optimize collective operation performance," in *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*, 2000, pp. 377–384.

[41] S. Li, Y. Zhang, and T. Hoefler, "Cache-oblivious mpi all-to-all communications based on morton order," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 542–555, March 2018.

[42] "CGYRO-GACODE: multi-species spectral gyrokinetic solver with sonic rotation capability," https://gafusion.github.io/doc/cgyro.html, 2019, Accessed: March 16, 2021.

[43] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1-2, pp. 19 – 25, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352711015000059