SC: G: gZCCL: Compression-Accelerated Collective Communication Framework for GPU Clusters

Jiajun Huang
University of California, Riverside

ABSTRACT

GPU-aware collective communication has become a major bottleneck for modern computing platforms as GPU computing power rapidly rises. A traditional approach is to directly integrate lossy compression into GPU-aware collectives, which can lead to serious performance issues such as underutilized GPU devices and uncontrolled data distortion. In order to address these issues, in this paper, we propose gZCCL, a first-ever general framework that designs and optimizes GPU-aware, compression-enabled collectives with an accuracy-aware design to control error propagation. To validate our framework, we evaluate the gZCCL on up to 512 NVIDIA A100 GPUs with real-world applications and datasets. Experimental results demonstrate that our gZCCL-accelerated collectives, including both collective computation (Allreduce) and collective data movement (Scatter), can outperform Cray MPI as well as NCCL by up to 28.7× and 4.5×, respectively. Furthermore, our accuracy evaluation with an image-stacking application confirms the high quality of the reconstructed data when using our accuracy-aware framework.

1 PROBLEM AND MOTIVATION

In the exascale computing era, efficient large-message collective communications are crucial for achieving high performance on modern GPU-based parallel systems. This is particularly true for scientific and deep learning applications that involve extensive data processing and exchange [1, 2, 4, 5]. For example, the classic LSTM [11] model used in the language modeling task can contain more than 66 million parameters and the communication overhead can be as high as 94% [2]. The need for optimized GPU-aware collective communication increases as message size grows larger [6, 7].

For GPU-aware collective communication, numerous researchers are actively working on mitigating network congestion in large-message collectives. Network saturation is often the main bottleneck because of the limited network bandwidth. For example, even with advanced networks, such as HPE Slingshot 10, the network bandwidth is only about 100 Gbps [21]. A straightforward solution is designing large-message collective communication algorithms that can minimize the transferred data volume instead of latency [3, 19, 23]. Another promising solution is shrinking the message size by error-bound lossy compression techniques [9, 13, 17, 18], as it can significantly reduce the data volume and maintain the data quality.

Previous lossy-compression-integrated approaches can be divided into two categories. The first is compression-enabled point-to-point communication (namely CPRP2P) [25], which directly uses the 1D fixed-rate ZFP [17] to compress the data before sending and decompresses it after receiving. This method may cause significant overheads and unbounded errors in the collective communications as shown in [12, 26]. The other category is to particularly optimize the compression-enabled collectives. Zhou et al. [26] integrated the 1D fixed-rate ZFP [17] into MPI_Alltoall on GPUs; however, this approach is limited to the Alltoall operation and GPU-centric staging algorithm and also results in the issue of unbounded error. Huang et al. [12] designed an optimized general framework for compression-enabled collectives that can realize high performance for all MPI collectives with controlled errors. Nevertheless, this approach suffers from suboptimal performance on modern GPU clusters because of under-utilized GPU devices.

To address the aforementioned limitations, this paper introduces a first-ever generic high-performance framework, namely gZCCL, specifically designed for GPU-aware compression-accelerated collective communications.

2 BACKGROUND AND RELATED WORK

Researchers have long been interested in utilizing compression to enhance MPI communication performance, based on the two communication categories – point-to-point communication and collective communication.

For the first category, a typical latest related work is utilizing 1D fixed-rate ZFP to boost MPI communications on GPU clusters [25]. Their approach, however, focuses on enhancing MPI point-to-point communication performance, yielding suboptimal performance in collective scenarios. Furthermore, their solution could not provide a bounded error due to its fixed-rate design that fixes the compressed data size rather than ensuring accuracy. In contrast, our collective framework integrates error-bounded lossy compression, guaranteeing both high-quality compression and high collective performance. Hence, we regard this work as orthogonal to ours.

As for the second category, several existing studies explored how to optimize the MPI collective performance particularly, while they are limited to either CPU-centric communication (i.e., all the data are transferred through the CPU essentially) and/or have the uncontrolled error propagation. Zhou et al. proposed several optimized MPI collective operations [24, 26] using fixed-rate compression, which leads to inferior compression quality and unbounded error aggregation. On the contrary, our general framework provides a detailed guideline for designing and optimizing compression-accelerated collective algorithms, maximizing the performance of both collective computation and collective data movement while featuring well-controlled data distortion. Hence, we categorize these works as orthogonal works to ours. In addition, Huang et al. proposed an error-controlled compression-enabled framework that is capable of achieving a high performance across all MPI collectives [12]. Their method, however, fails to solve the inefficient GPU utilization, synchronization, and device-host data transfer issues, resulting in suboptimal performance on GPU clusters. In contrast, our GPU-centric framework is capable of fully utilizing the computational power of GPUs, significantly lowering the amounts of
required compression, synchronization, and device-host data transfer, leading to a remarkable performance improvement.

3 UNIQUENESS OF GZCCL

Designing a GPU-aware compression-enabled collective communication system that realizes both high performance and controlled error propagation is non-trivial. There are three key challenges to address.

(1) How can we co-design and implement a compression-enabled collective algorithm that optimizes performance within modern GPU clusters? For Allreduce operations, for example, state-of-the-art GPU-aware collective communication libraries, such as NCCL [8] and MPICH [16], adopt ring-based algorithms to optimize the transmission of large messages. However, it is unclear whether the ring-based model is the best fit when we include lossy compression techniques. In fact, unlike CPU, the GPU-based compression may easily face a low utilization issue, because of the inevitable GPU kernel launching overhead and the limited parallel design in GPU-based compression algorithms, which significantly lowers the performance.

(2) How can we optimize the redesigned algorithms to increase GPU utilization and decrease the required synchronizations and data transfers? This is because unnecessary data transfers and synchronization can considerably increase the overall runtime and eliminate the opportunity for overlapping in the coordination of the host and device.

(3) How can we devise an accuracy-aware co-design that maintains data quality without sacrificing performance? The accuracy of collective operations is at risk due to the data loss from GPU lossy compression. It is important to balance performance with accuracy.

4 EXPERIMENTAL RESULTS

We present and discuss the evaluation results as follows.

4.1 Experimental Setup

We perform the evaluation on a GPU supercomputer that involves 512 NVIDIA A100 80G GPUs (128 nodes each with 4 GPUs, specifically), which features both internode communication and intranode communication. These computational nodes are interconnected via
Evaluation with different message sizes. We evaluate the performance of our gZ-Allreduce algorithm using various data sizes up to 600 MB on a configuration of 64 NVIDIA A100 GPUs across 16 nodes. As observed in Figure 4, our recursive doubling-based gZ-Allreduce (ReDoub) consistently outperforms across all data sizes, achieving up to a speedup of 18.7x compared to Cray MPI and a 3.4x performance improvement over NCCL. Furthermore, with increasing data sizes, the speedup generally rises, demonstrating high scalability with respect to data size. The performance improvement originates from the significantly reduced message size and compression-related overheads in our gZCCL design, which can further mitigate network congestion with enlarging message sizes. However, the ring-based gZ-Allreduce (Ring), despite surpassing Cray MPI for the data size with 50+ MB, fails to outpace NCCL. This is attributed to the inefficient GPU utilization in gZ-Allreduce (Ring), which incurs substantial compression-related costs, outweighing the benefits of reduced message size.

Evaluation with different GPU counts. In this section, we assess the scalability of our gZ-Allreduce algorithm with the complete RTM dataset of 646 MB data size, utilizing up to 512 NVIDIA A100 GPUs across 128 nodes. We start from 8 GPUs, as it is the minimal amount to have both internode and intranode communication with 4 GPUs per node.

As depicted in Figure 5, our recursive doubling-based gZ-Allreduce (ReDoub) consistently performs the best, achieving up to 20.2x and 4.5x speedups compared to Cray MPI and NCCL respectively, across varying GPU counts. This superior performance stems from the substantial reduction in message size with relatively low compression cost achieved by our gZCCL framework. When the GPU count is at 8, Cray MPI appears to suffer from significant performance degradation, as compared to the other three counterparts. Apart from the 8-GPU case, as the number of GPUs increases, both gZ-Allreduce (ReDoub) and NCCL tend to exhibit a greater performance boost compared to Cray MPI, indicating robust scalability with respect to the GPU count. This is because both gZ-Allreduce (ReDoub) and NCCL are optimized for large GPU count scenarios. However, the trend differs for the ring-based gZ-Allreduce (Ring), which outperforms NCCL when the GPU count is 32 or less. As the GPU count increases, its performance deteriorates, ending up with the worst performance compared with other solutions in the case of 512 GPUs. The declining performance is attributed to the reduced input data size for each compression/decompression with an increase of GPU count, leading to lower device utilization and prolonged runtime, thus subpar scalability.

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Simulation Setting 1</th>
<th>Simulation Setting 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>CPR</td>
<td>PSNR</td>
</tr>
<tr>
<td>1E-3</td>
<td>92.28</td>
<td>53.23</td>
</tr>
<tr>
<td>1E-4</td>
<td>73.35</td>
<td>65.67</td>
</tr>
<tr>
<td>1E-5</td>
<td>55.65</td>
<td>78.83</td>
</tr>
</tbody>
</table>

Table 1: Compression ratio (CPR) and quality (PSNR).

4.2 Comparisons of gZCCL with other collective communication libraries

In this section, we compare the performance of our gZCCL framework with other state-of-the-art GPU communication libraries, such as the widely-utilized NCCL and CUDA-aware Cray MPI.

4.2.1 Collective computation. In this section, the performance of our gZCCL collective computation framework is compared with both NCCL and Cray MPI, using the prevalent Allreduce operation.
4.2 Collective data movement. In this section, we assess the performance of our gZCCL collective data movement framework using the widely-used Scatter operation, comparing it with Cray MPI. We exclude NCCL from this comparison as it has no implementation for Scatter.

Evaluation with different message sizes. We evaluate the performance of our gZ-Scatter with data sizes up to 600 MB, using 64 NVIDIA A100 GPUs on 16 nodes. Figure 6 indicates that our gZ-Scatter is able to consistently outperform Cray MPI across all data sizes. The speedup of gZ-Scatter enhances as the data size increases, achieving its maximum (20.2×) at 600 MB. This demonstrates superior scalability with respect to data sizes, which can be attributed to the reduced message sizes and overlapping of compression, kernel launching, and data movement in our gZCCL framework.

Evaluation with different GPU counts. We assess the scalability of our gZ-Scatter with the complete RTM dataset, with a data size of 646 MB, using up to 512 NVIDIA A100 GPUs spread across 128 nodes. From Figure 7, it is evident that our gZ-Scatter outperforms Cray MPI in all cases. As the GPU count increases, the speedup of gZ-Scatter first increases, peaking at 28.7×, and then gradually decreases to 4.75× when the GPU count reaches 512. Unlike the Allreduce scenario, the message size distributed to each non-root GPU in the Scatter communication pattern linearly decreases as the GPU count rises. When the GPU count is less than or equal to 16, the message size on the non-root GPU allows for high GPU utilization, hence the speedup grows with the increasing GPU count. However, when the GPU count exceeds or equals 32, the GPU utilization continues to drop, thereby reducing the collective performance and leading to a decrease in performance enhancement.

4.3 Image Stacking Performance Evaluation with Accuracy Analysis

In this section, we employ the image stacking application to evaluate both the performance and accuracy of our gZCCL. Image stacking, a technique widely used in various scientific fields such as atmospheric science and geology, is employed to generate high-quality images by stacking multiple individual images, which essentially constitutes an Allreduce operation. As demonstrated by Gurhem in 2021 [10], researchers use MPI to merge these individual images into a comprehensive final image. As can be seen from Table 2, our ring-based gZCCL (Ring) outperforms Cray MPI by a factor of 3.99× when using an absolute error bound of 1E-4. Moreover, our recursive doubling-based gZCCL (ReDoub) offers even higher performance with speedups of up to 9.26× and 1.69× compared with Cray MPI and NCCL, respectively. This significant performance enhancement arises from the markedly reduced message sizes and compression-related overheads brought by our gZCCL framework.

The following text presents a performance breakdown analysis. For gZCCL (Ring), 84.08% of the total runtime is consumed by compression, whereas gZCCL (ReDoub) has comparable compression and communication costs at 42.61% and 46.28% respectively. This substantial reduction in compression cost is due to higher GPU utilization and fewer compression operations in our optimized gZ-Allreduce (ReDoub) algorithm compared with gZCCL (Ring).

In addition to performance analysis, we thoroughly evaluate the accuracy using both visualization method and numerical metrics such as the widely-used peak signal-to-noise ratio (PSNR) [20] and normalized root mean squared error (NRMSE) [22]. Our accuracy-aware design allows gZCCL (ReDoub) to achieve excellent reconstructed image quality, even with an error bound of 2E-4, as shown in Figure 8. The reconstructed image of gZCCL (Ring) also exhibits high visual quality, similar to that shown in Figure 8b, hence it is not presented separately here. When the error bound is tightened to 1E-4, as used in our performance analysis, gZCCL (Ring) reaches a great...
This paper presents gZCCL accelerators. Our future work will evaluate inefficient GPU utilization, significant compression-related overheads. Our experiments with up to 512 NVIDIA A100 GPUs illustrate that gZCCL surpasses the Allreduce operation in Cray MPI. Our framework with more collective operations and we plan to extend gZCCL to more hardware such as FPGAs and AI accelerators.

REFERENCES