Hign Performance Computing 2023/2024

Assignment 1:

Performance Evaluation of MPI Collective Operations on the Orfeo HPC Cluster: A Comparative Analysis of Broadcast and Scatter Algorithms

Malasi Denis

IN2000213

1.Introduction

In the realm of High-Performance Computing (HPC), efficient communication between processes is critical for maximizing performance and scalability. In MPI-based applications the collective operations, such as broadcast and scatter, play a pivotal role in this communication.

Broadcast involves sending a single message from one process, known as the root, to all other processes in a communicator. This operation is commonly used for distributing common data or instructions to all processes participating in a parallel computation.

Scatter, on the other hand, involves partitioning data from one process (the root) and sending chunks of it to all other processes in the communicator. It is particularly useful when different parts of a dataset need to be processed independently by different processes.

These operations enable the efficient distribution and aggregation of data across multiple processes, which is essential for parallel computing tasks that HPC systems are designed to handle. By optimizing these collective operations, we can significantly enhance the overall performance of HPC applications, leading to faster computation times and more efficient use of computational resources.

The analysis was conducted using the OSU MPI Benchmark suite. This suite provides a comprehensive set of benchmarks designed to measure the performance of various MPI operations, including collective operations like broadcast and scatter. The benchmarks were executed on two fully utilized THIN nodes of the ORFEO cluster.

For the broadcast operation, we examined the chain algorithm, which performs a sequential data distribution, and the scatter-allgather ring algorithm, known for its efficient pipelined data transfer. For the scatter operation, we evaluated the non-blocking linear algorithm, which allows processes to continue computation while the scatter operation is still in progress, and the basic linear algorithm, which sends data sequentially from the root to all other processes. These algorithms were chosen to highlight differences in performance and to identify the most efficient methods for our specific HPC environment.

2.Methodology for Data Collection

Software Stack

To conduct our performance analysis, we used the OSU MPI Benchmark suite (version 7.3), which provides various benchmarks for measuring the performance of MPI operations.

Setting Up the Environment

Before running the benchmarks, it was necessary to load the appropriate MPI module to ensure compatibility with the system and the benchmarks. This was done using the following command:

$$module\ load\ openMPI/4.1.6/gnu/12.2.1$$

This command loads the OpenMPI version 4.1.5 with the GNU compiler version 12.2.1. Loading this module ensures that the system uses the correct MPI library.

Allocation of Resources

The experiments were performed on the ORFEO cluster, utilizing two THIN nodes. Each node was fully utilized to ensure maximum efficiency and accuracy in measuring the performance of the collective operations. The resource allocation was handled using the SLURM workload manager with the following command:

$$salloc - p\ - N2 - -tasks - per - node = 24 - pTHIN - -time = 0{:}\,30{:}\,00$$

This command allocated two nodes with 24 tasks per node from the THIN partition for a duration of 30 minutes.

3.Execution of Benchmarks

After allocating the necessary resources, the benchmarks were executed using *mpirun* with specific options to measure the performance of the selected algorithms for broadcast and scatter operations. The following commands were used to run the broadcast benchmarks:

$$mpirun\ - -mca\ coll\_tuned\_use\_dynamic\_rules\ true$$
$$- - mca\ coll\_tuned\_bcast\_algorithm < number > osu\_bcast\ - x\ 100\ - i\ 1000$$

Scatter benchmarks:

$$mpirun\ - -mca\ coll\_tuned\_use\_dynamic\_rules\ true$$
$$- - mca\ coll\_tuned\_scatter\_algorithm\ < number >\ osu\_scatter\ - x\ 100\ - i\ 1000$$

The '*<number>*' specifies the broadcast/scatter algorithm to use (2 for chain, 9 for scatter-allgather ring, 1 for basic linear, 3 for non-blocking linear).

'-x 100' sets the number of warm-up iterations to 100, *'-i 1000'* sets the number of total iterations to 1000.

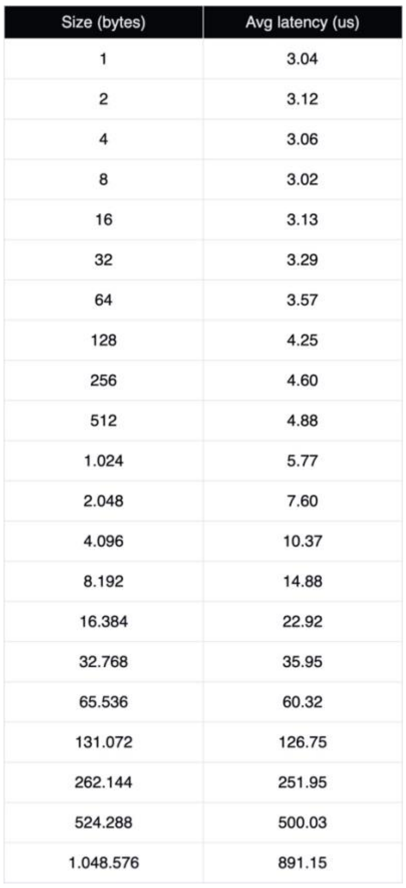
The benchmark suite was configured to automatically test a predetermined range of message sizes, spanning from 1 byte to 1,048,576 bytes.

4.Results

The performance evaluation of broadcast and scatter operations provided valuable insights into the efficiency of different algorithms. By measuring the time taken for these operations with varying message sizes, key trends and differences in performance were identified. Below, the results for both broadcast and scatter operations are presented and analyzed.

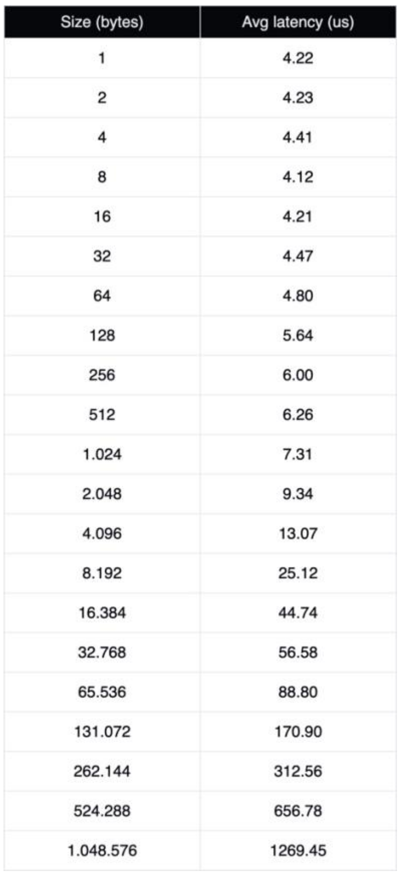
Broadcast

The performance of the chain algorithm and the scatter-allgather ring algorithm for broadcast operations was compared against the baseline broadcast algorithm. The baseline algorithm, serving as a reference point, provided a standard measure of performance for distributing data from the root process to all other processes.



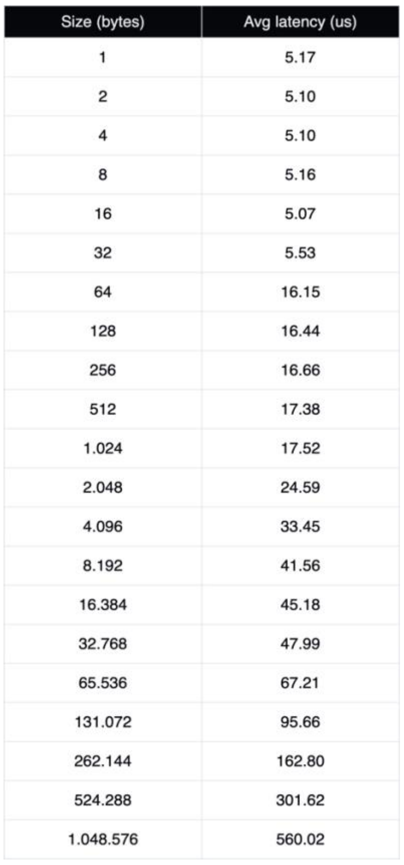
Size (bytes)	Avg latency (us)
1	3.04
2	3.12
4	3.06
8	3.02
16	3.13
32	3.29
64	3.57
128	4.25
256	4.60
512	4.88
1.024	5.77
2.048	7.60
4.096	10.37
8.192	14.88
16.384	22.92
32.768	35.95
65.536	60.32
131.072	126.75
262.144	251.95
524.288	500.03
1.048.576	891.15

Table 1: Broadcast baseline latency



Size (bytes)	Avg latency (us)
1	4.22
2	4.23
4	4.41
8	4.12
16	4.21
32	4.47
64	4.80
128	5.64
256	6.00
512	6.26
1.024	7.31
2.048	9.34
4.096	13.07
8.192	25.12
16.384	44.74
32.768	56.58
65.536	88.80
131.072	170.90
262.144	312.56
524.288	656.78
1.048.576	1269.45

Table 2: Broadcast chain latency



Size (bytes)	Avg latency (us)
1	5.17
2	5.10
4	5.10
8	5.16
16	5.07
32	5.53
64	16.15
128	16.44
256	16.66
512	17.38
1.024	17.52
2.048	24.59
4.096	33.45
8.192	41.56
16.384	45.18
32.768	47.99
65.536	67.21
131.072	95.66
262.144	162.80
524.288	301.62
1.048.576	560.02

Table 3: Broadcast scatter-allgather ring latency

The tables above illustrate the average latencies observed for the broadcast operation across various message sizes using three different algorithms: the baseline broadcast, the chain algorithm, and the scatter-allgather ring algorithm.

To provide a clearer comparison of these results, the following graph consolidates the data from all three tables, enabling a visual comparison of the algorithms' performance across the range of message sizes. This visualization aids in identifying the relative efficiencies and scaling behaviors of each algorithm.

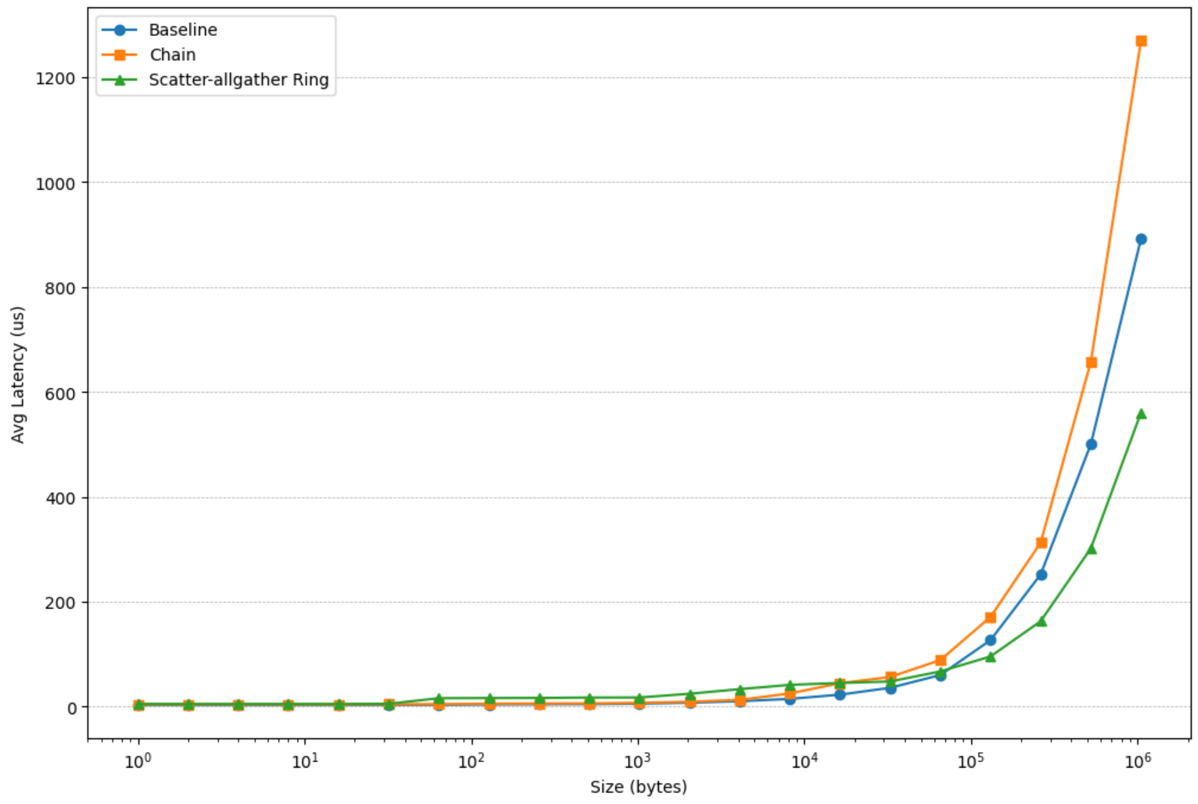
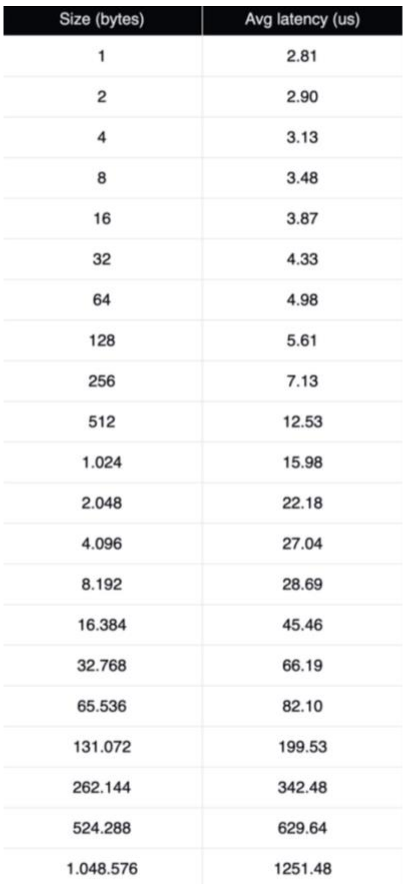


Figure 1: Comparison of Average Latencies for Broadcast Operations

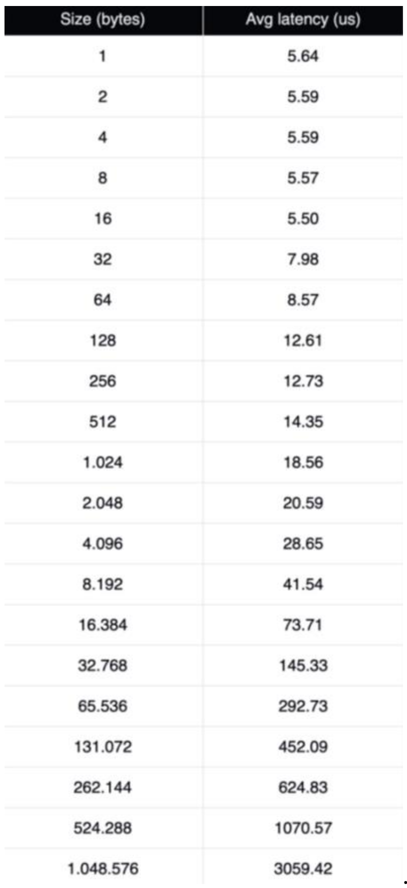
Scatter

The performance of the non-blocking linear algorithm and the basic linear algorithm for scatter operations was evaluated and compared to the baseline scatter algorithm. The baseline algorithm served as a benchmark, providing a standard measure of performance for partitioning and distributing data from the root process to all other processes.



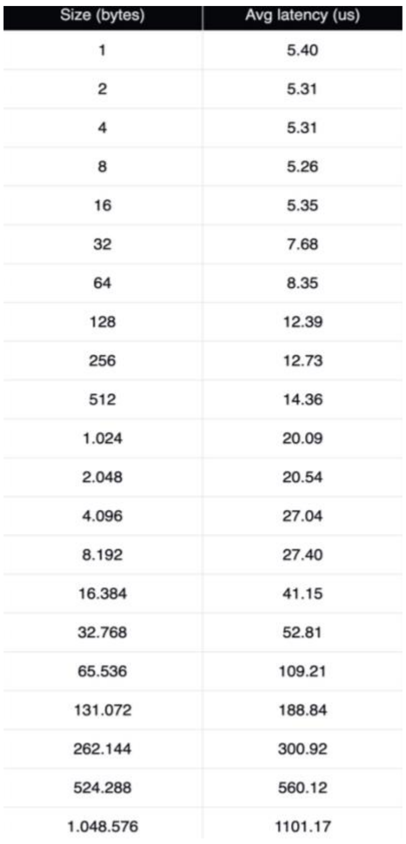
Size (bytes)	Avg latency (us)
1	2.81
2	2.90
4	3.13
8	3.48
16	3.87
32	4.33
64	4.98
128	5.61
256	7.13
512	12.53
1.024	15.98
2.048	22.18
4.096	27.04
8.192	28.69
16.384	45.46
32.768	66.19
65.536	82.10
131.072	199.53
262.144	342.48
524.288	629.64
1.048.576	1251.48

Table 4: Scatter baseline latency



Size (bytes)	Avg latency (us)
1	5.64
2	5.59
4	5.59
8	5.57
16	5.50
32	7.98
64	8.57
128	12.61
256	12.73
512	14.35
1.024	18.56
2.048	20.59
4.096	28.65
8.192	41.54
16.384	73.71
32.768	145.33
65.536	292.73
131.072	452.09
262.144	624.83
524.288	1070.57
1.048.576	3059.42

Table 5: Scatter basic linear latency



Size (bytes)	Avg latency (us)
1	5.40
2	5.31
4	5.31
8	5.26
16	5.35
32	7.68
64	8.35
128	12.39
256	12.73
512	14.36
1.024	20.09
2.048	20.54
4.096	27.04
8.192	27.40
16.384	41.15
32.768	52.81
65.536	109.21
131.072	188.84
262.144	300.92
524.288	560.12
1.048.576	1101.17

Table 6: Scatter non-blocking linear latency

The tables display the average latency results for the scatter operations using the three algorithms: non-blocking linear, basic linear, and the baseline scatter algorithm. These data points provide a detailed view of each algorithm's performance across various message sizes.

To enhance the clarity of the performance differences among these algorithms, the following graph visually represents the latency results. This graph helps to clearly illustrate trends and differences in average latency for different message sizes, facilitating a direct comparison between the three approaches.

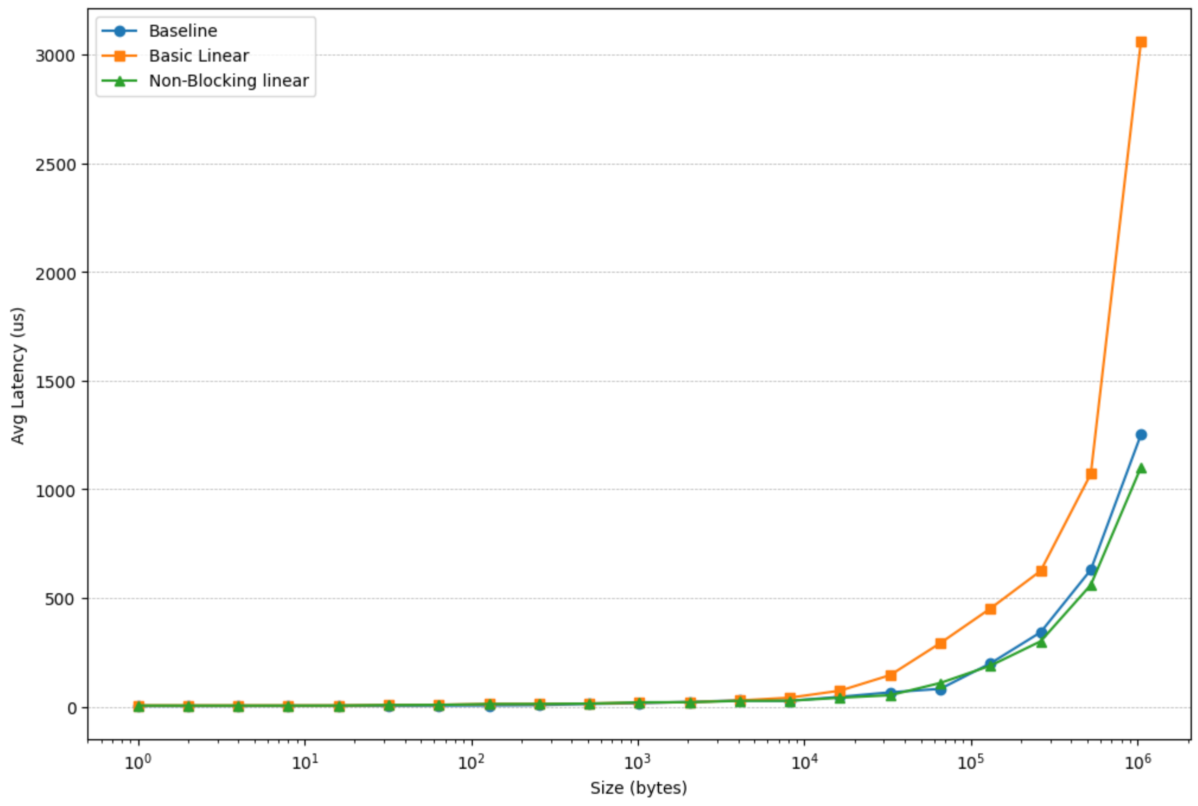


Figure 2: Comparison of Average Latencies for Scatter Operations

5.Discussion of the results

The performance evaluation of broadcast and scatter operations on the ORFEO cluster provided valuable insights into the efficiency and scalability of different algorithms. By analyzing the average latency across various message sizes, key performance trends were identified. These trends are critical for understanding how different algorithms can impact the overall efficiency of HPC applications. Below, we discuss the results in detail, focusing on the specific performance characteristics of each algorithm in the context of Orfeo's architecture.

Broadcast

Chain Algorithm

The Chain algorithm, compared to the baseline, shows a mixed performance. For smaller message sizes, its performance is comparable to the baseline, but it exhibits higher latencies as the message size increases. Within Orfeo's architecture, the Chain algorithm benefits from the low-latency InfiniBand network but still faces intrinsic scalability limitations for larger messages. This is because it does not take full advantage of the full-duplex capabilities of Orfeo's network due to its sequential nature of data transmission.

Scatter-Allgather Ring Algorithm

The Scatter-Allgather Ring algorithm performs significantly better than the baseline, particularly for larger message sizes. This algorithm excels in Orfeo's architecture due to its efficient handling of large data transfers via the high-speed, low-latency InfiniBand network. The parallel nature of scatter and allgather operations significantly reduces overall broadcast latency, making it highly suitable for larger messages. Orfeo's network architecture, with its full-duplex capability, allows simultaneous send and receive operations. This feature is fully exploited by the Scatter-Allgather Ring algorithm, enhancing throughput and reducing communication time by allowing data to be transmitted in both directions at once. This parallelism ensures that the communication load is evenly distributed across the network, minimizing bottlenecks and improving overall efficiency. Additionally, the robust bandwidth and low-latency characteristics of the InfiniBand network further amplify the performance benefits of this algorithm, particularly under heavy data loads.

Scatter

Basic Linear Algorithm

The Basic Linear algorithm, when compared to the baseline scatter algorithm, performs worse for larger message sizes. This degradation in performance is attributed to its sequential nature, which does not leverage the parallel communication capabilities of Orfeo's full-duplex network. Instead, it operates primarily in a send-wait mode, resulting in increased latency as each data segment must be sent and acknowledged before the next segment can be transmitted. Consequently, the lack of parallelism in data transmission and reception leads to suboptimal utilization of Orfeo's high-speed, low-latency InfiniBand network, particularly under conditions of heavy data load.

Non-Blocking Linear Algorithm

The Non-Blocking Linear algorithm demonstrates a significant improvement over the baseline, particularly for larger message sizes. By enabling multiple simultaneous communications, this algorithm effectively leverages Orfeo's full-duplex and high-speed InfiniBand network to minimize latency. The Non-Blocking Linear algorithm's ability to handle concurrent send and receive operations is particularly advantageous. This capability fully exploits the full-duplex nature of the network, leading to a substantial reduction in communication time. Consequently, the algorithm achieves higher efficiency and throughput, making it well-suited for handling larger message sizes by optimizing the use of available network resources.

6.Conclusion

In conclusion, the choice of algorithm has a significant impact on the performance of broadcast and scatter operations in a high-performance computing cluster like Orfeo. Algorithms that can leverage the full-duplex and high-speed capabilities of the network, like Scatter-Allgather Ring and Non-Blocking Linear, demonstrate superior performance, particularly for larger message sizes. This analysis underscores the importance of aligning algorithm design with the underlying network architecture to achieve optimal efficiency and scalability.