Ex06_p04

The computations for this exercise were performed on an apple m2 processor using shared memory.
The program *p04.cpp* uses pingpong communication to send messages of sizes 10000...10000000000, 1000 times per message.

p04.cpp
```
#include <mpi.h>
#include <iostream>
#include <vector>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (size != 2) {
        if (rank == 0) {
            std::cout << "This program requires exactly 2
processes." << std::endl;
        }
        MPI_Finalize();
        return 0;
    }

    int num_iterations = 1000;
    int max_message_size = 10000000000;
    std::vector<char> buffer(max_message_size);

    for (int message_size = 10000; message_size <=
max_message_size; message_size *= 10) {
        MPI_Barrier(MPI_COMM_WORLD);
        double start_time = MPI_Wtime();

        for (int i = 0; i < num_iterations; ++i) {
            if (rank == 0) {
                MPI_Send(buffer.data(), message_size,
MPI_CHAR, 1, 0, MPI_COMM_WORLD);
                MPI_Recv(buffer.data(), message_size,
MPI_CHAR, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            } else if (rank == 1) {
                MPI_Recv(buffer.data(), message_size,
MPI_CHAR, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
                MPI_Send(buffer.data(), message_size,
```

```
MPI_CHAR, 0, 0, MPI_COMM_WORLD);
            }
        }

        MPI_Barrier(MPI_COMM_WORLD);
        double end_time = MPI_Wtime();

        if (rank == 0) {
            double time_taken = end_time - start_time;
            double latency = time_taken / (2 *
num_iterations);
            double bandwidth = (message_size *
num_iterations) / (time_taken * 1e6);
            std::cout << "Message size: " << message_size <<
", Latency: " << latency << ", Bandwidth: " << bandwidth <<
" MB/s" << std::endl;
        }
    }

    MPI_Finalize();
    return 0;
}
```

As one can see from the output, as the message size increases the latency also increases. The bandwidth also increases with the message size as the larger messages are more efficiently communicated between processes, until it reaches a cap at around 11000 Mb/s, after which it decreases.

Output from p04.cpp

```
Message size: 100, Latency: 5.895e-07, Bandwidth: 84.8176
MB/s
Message size: 1000, Latency: 5.515e-07, Bandwidth: 906.618
MB/s
Message size: 10000, Latency: 9.02e-07, Bandwidth: 5543.24
MB/s
Message size: 100000, Latency: 9.858e-06, Bandwidth: 5072.02
MB/s
Message size: 1000000, Latency: 4.48155e-05, Bandwidth:
11156.9 MB/s
Message size: 10000000, Latency: 0.000590015, Bandwidth:
1194.94 MB/s
Message size: 100000000, Latency: 0.00656064, Bandwidth:
92.655 MB/s
```