FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

ITMO UNIVERSITY

# Report

MPI. Assignments 8

Parallel algorithms for the analysis and synthesis of data

Performed by
Aleksandr Shirokov
J4133c
Accepted by
Petr Andriushchenko
Deadline: 20.12.21

St. Petersburg
2021

# Contents

# 1 Assignments

## 1.1 Assignment 8. MPI. Bandwidth measurement.

### 1.1.1 Formulation of the problem

1. Write an MPI program in which two processes exchange messages
2. Measure the time per exchange iteration
3. Determine the dependence of the exchange time on the message length.
4. Determine the latency and maximum achievable bandwidth of the communication network.
5. Print the message length in bytes and the throughput in MB/s to the console.
6. Change the length of the message in a loop starting from 1 element and increase to $1,000,000$ elements, increasing by 10 times at each iteration.

### 1.1.2 Bandwidth measurement technique

The following technique is used to measure point-to-point bandwidth. Process 0 sends a message of length $L$ bytes to process 1.

$$L = \text{number of elements} \cdot \text{length in bytes of data type}$$

$$L = 100 \cdot \text{sizeof}(int)$$

Process 1, having received a message from process 0, sends it a reply message ot the same length. Blocking MPI calls are used (MPI_Send, MPI_Recv). These actions are repeated $N = 10$ times in order to minimize the error due to averaging. Process 0 measures the time $T$ taken for all these exchanges.

The bandwidth $R$ is determined by formula:

$$R = \frac{2NL}{T}$$

### 1.1.3 Latency measurement technique

Latency is measured as the time it takes to transmit a signal or message of zero length. At the same time, to reduce the influence of the error and low resolution of the system timer, it is important to repeat the operation of sending a signal and receiving a response a large number of times.

Thus, if the time for $N$ iterations of sending zero-length messages back and forth was $T$ sec., Then the latency is measured as:

$$s = \frac{T}{2N}$$

### 1.1.4 Example of launch parameters and output. Detailed description of solution

Code for **assignment 6** is here.
Compilation example: mpic++ -o ./cpf/8.o Assignment8.c
Launch example: mpirun –oversubscribe -np 2 ./cpf/8.o

```
aptmess@improfeo: ~/ITMO/parallel_algorithms/HT/hw_mpi
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$ mpirun --oversubscribe -np 2 ./cpf/8.o
rank=1 len=1 L(bytes)=4 T(us)=32.7 R[bandwidth](MB/s)=2.33315e-06 s[latency](us)=1.635
rank=0 len=1 L(bytes)=4 T(us)=44 R[bandwidth](MB/s)=1.73395e-06 s[latency](us)=2.2
rank=1 len=10 L(bytes)=40 T(us)=9.4 R[bandwidth](MB/s)=8.11638e-05 s[latency](us)=0.47
rank=0 len=10 L(bytes)=40 T(us)=10.6 R[bandwidth](MB/s)=7.19754e-05 s[latency](us)=0.53
rank=0 len=100 L(bytes)=400 T(us)=15.1 R[bandwidth](MB/s)=0.000505258 s[latency](us)=0.755
rank=1 len=100 L(bytes)=400 T(us)=56 R[bandwidth](MB/s)=0.000136239 s[latency](us)=2.8
rank=0 len=1000 L(bytes)=4000 T(us)=48 R[bandwidth](MB/s)=0.00158946 s[latency](us)=2.4
rank=1 len=1000 L(bytes)=4000 T(us)=74.2 R[bandwidth](MB/s)=0.00102822 s[latency](us)=3.71
rank=0 len=10000 L(bytes)=40000 T(us)=131.9 R[bandwidth](MB/s)=0.00578423 s[latency](us)=6.595
rank=1 len=10000 L(bytes)=40000 T(us)=126.9 R[bandwidth](MB/s)=0.00601213 s[latency](us)=6.345
rank=0 len=100000 L(bytes)=400000 T(us)=446.2 R[bandwidth](MB/s)=0.0170986 s[latency](us)=22.31
rank=1 len=100000 L(bytes)=400000 T(us)=529.3 R[bandwidth](MB/s)=0.0144141 s[latency](us)=26.465
rank=0 len=1000000 L(bytes)=4000000 T(us)=9034.5 R[bandwidth](MB/s)=0.00844473 s[latency](us)=451.725
rank=1 len=1000000 L(bytes)=4000000 T(us)=9834.9 R[bandwidth](MB/s)=0.00775747 s[latency](us)=491.745
(base) aptmess@improfeo:~/ITMO/parallel_algorithms/HT/hw_mpi$
```

Let's move to the the code and explain how it works.

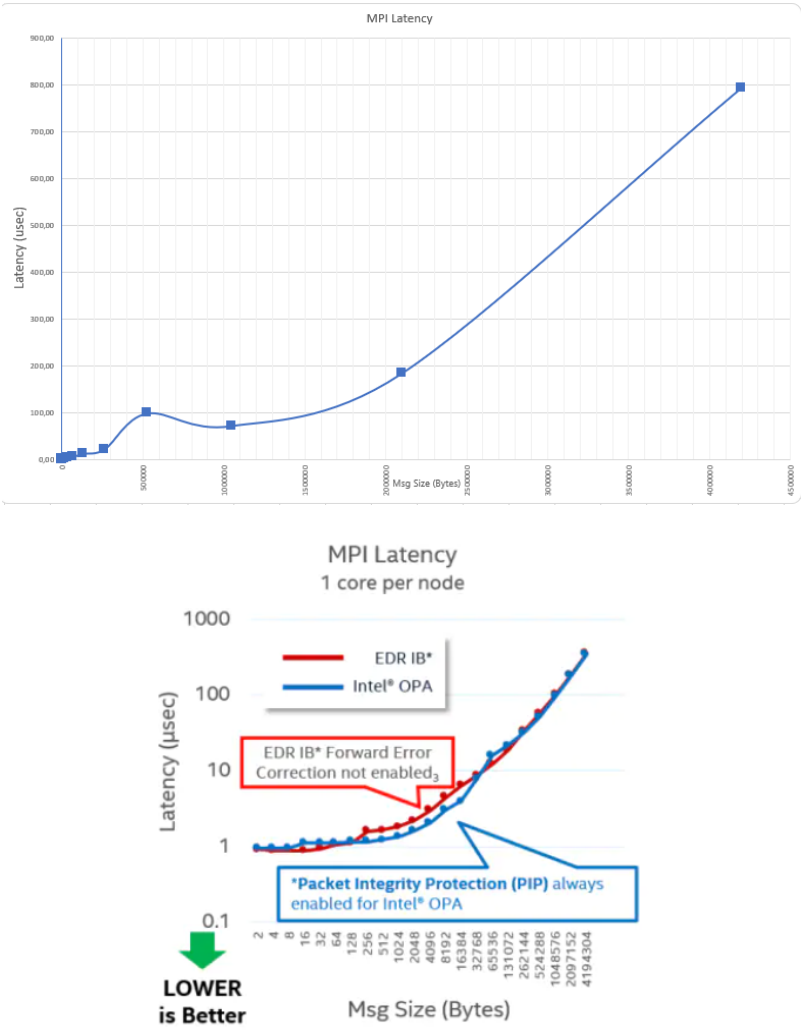
```cpp
#include <mpi.h>
#include <iostream>
#include <stdexcept>

using namespace std;

static int const root = 0;
static int const N = 10;

void FactoryMessage(
    string variant,
    int to_home_sent,
    int len,
    int *msg,
    MPI_Status status
)
{
    double start_time, end_time, T, R, s;
    int L = len * sizeof(int);

    start_time = MPI_Wtime();
    for (int i = 0; i < N; i++)
        if (variant == "send")
        {
            MPI_Send(msg, len, MPI_INT, to_home_sent, root, MPI_COMM_WORLD);
        }
        else if (variant == "recv")
        {
            MPI_Recv(msg, len, MPI_INT, to_home_sent, root, MPI_COMM_WORLD, &status);
        }
    end_time = MPI_Wtime();

    T = (end_time - start_time) * 1000000;
    R = 2 * L * N / T / 1024 / 1024;
    s = T / 2 / N;

    if (variant == "send")
    {
        cout << "rank=" << (to_home_sent ^ 1);
        cout << " len=" << len;
        cout << " L(bytes)=" << L;
        cout << " T(us)=" << T;
        cout << " R[bandwidth](MB/s)=" << R;
        cout << " s[latency](us)=" << s;
        cout << '\n' << endl;
    }
}
```

```cpp
int main(int argc, char **argv)
{
    int rank, n;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &n);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Status status;
    int *msg;

    if (n != 2)
    {
        throw invalid_argument( "Only 2 processes is acceptable" );
    }

    for (int len = 1; len <= 1000000; len *= 10)
    {
        msg = new int[len]();
        if (rank == 0)
        {
            for (int i = 0; i < len; i++)
            {
                msg[i] = 1;
            }

            FactoryMessage("send", 1, len, msg, status);
            FactoryMessage("recv", 1, len, msg, status);
        }
        else if (rank == 1)
        {
            FactoryMessage("recv", 0, len, msg, status);
            FactoryMessage("send", 0, len, msg, status);
        }
        delete[] msg;
    }

    MPI_Finalize();
    return 0;
}
```

Assignment8 code

In the code there are on function FactoryMessage, that start MPI_Send or MPI_Recv function depends on input parameter *variant*, after that there are a computation of metrics such as average time in seconds, latency and bandwidth by formulas in previous subsections and printing the result. In main function process 0 send array of different length on each iteration of for cycle to process 1, process 1 then recieves it and send to process 0 and it is continue for cycle len = [1..1000000] by increasing it 10 times in each iteration.

We can also check if results are more or less correct - I tried to compare latency from intel comparison and it is more or less the same as in our experiement!




Experiement by Intel of MPI latency and our - looks similar

We have explained the code in assignment 8 and compare the results.

## 1.2 Appendix

The link to the sourse code which is placed on my github.