

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

<b>1</b>	<b>Assignments</b>	<b>2</b>
1.1	Assignment 8. MPI. Bandwidth measurement. . . . .	2
1.1.1	Formulation of the problem . . . . .	2
1.1.2	Bandwidth measurement technique . . . . .	2
1.1.3	Latency measurement technique . . . . .	2
1.1.4	Example of launch parameters and output. Detailed description of solution .	3
1.2	Appendix . . . . .	3

# 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 of 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`

```
(base) aptmess@improfeo:~/ITM0/parallel_algorithms/HT/hw_mpi$ mpic++ -o ./cpf/8_fix.o Assignment8.c
(base) aptmess@improfeo:~/ITM0/parallel_algorithms/HT/hw_mpi$ mpirun --oversubscribe -np 2 ./cpf/8_fix.o
rank=0 len=1 L(bytes)=4 T(us)=83.2 R[bandwidth](MB/s)=0.916995 s[latency](us)=4.16
rank=1 len=1 L(bytes)=4 T(us)=49.7 R[bandwidth](MB/s)=1.53509 s[latency](us)=2.485
rank=0 len=10 L(bytes)=40 T(us)=11.7 R[bandwidth](MB/s)=65.2085 s[latency](us)=0.585
rank=1 len=10 L(bytes)=40 T(us)=13.6 R[bandwidth](MB/s)=56.0985 s[latency](us)=0.68
rank=0 len=100 L(bytes)=400 T(us)=10.2 R[bandwidth](MB/s)=747.98 s[latency](us)=0.51
rank=1 len=100 L(bytes)=400 T(us)=43 R[bandwidth](MB/s)=177.428 s[latency](us)=2.15
rank=0 len=1000 L(bytes)=4000 T(us)=28.3 R[bandwidth](MB/s)=2695.9 s[latency](us)=1.415
rank=1 len=1000 L(bytes)=4000 T(us)=29.3 R[bandwidth](MB/s)=2603.89 s[latency](us)=1.465
rank=0 len=10000 L(bytes)=40000 T(us)=86 R[bandwidth](MB/s)=8871.39 s[latency](us)=4.3
rank=1 len=10000 L(bytes)=40000 T(us)=291.8 R[bandwidth](MB/s)=2614.6 s[latency](us)=14.59
rank=0 len=100000 L(bytes)=400000 T(us)=1191.2 R[bandwidth](MB/s)=6404.8 s[latency](us)=59.56
rank=1 len=100000 L(bytes)=400000 T(us)=720.7 R[bandwidth](MB/s)=10586.1 s[latency](us)=36.035
rank=0 len=1000000 L(bytes)=4000000 T(us)=10040.7 R[bandwidth](MB/s)=7598.47 s[latency](us)=502.035
rank=1 len=1000000 L(bytes)=4000000 T(us)=13889.6 R[bandwidth](MB/s)=5492.88 s[latency](us)=694.48
(base) aptmess@improfeo:~/ITM0/parallel_algorithms/HT/hw_mpi$
```

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

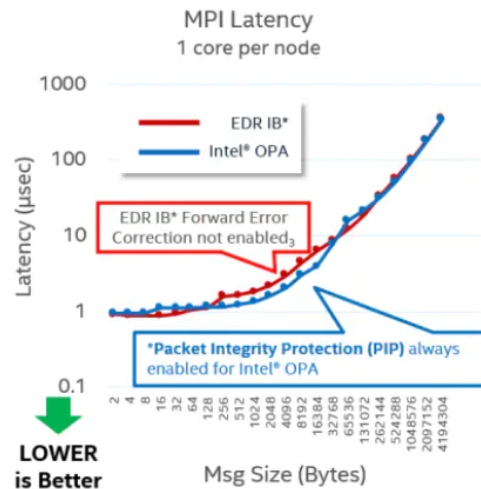
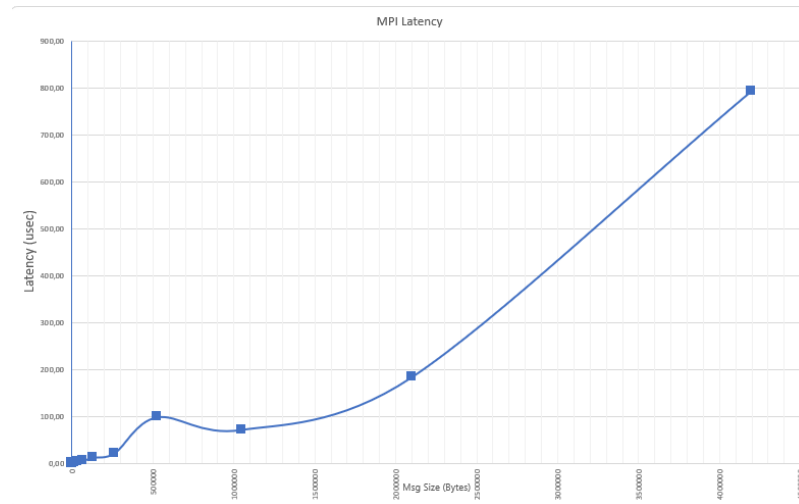
```
1  #include <mpi.h>
2  #include <iostream>
3  #include <stdexcept>
4
5  using namespace std;
6
7  static int const root = 0;
8  static int const N = 10;
9
10 void FactoryMessage(
11     string variant,
12     int to_home_sent,
13     int len,
14     int *msg,
15     MPI_Status status
16 )
17 {
18     double start_time, end_time, T, R, s;
19     int L = len * sizeof(int);
20
21     start_time = MPI_Wtime();
22     for (int i = 0; i < N; i++)
23     {
24         if (variant == "send")
25         {
26             MPI_Send(msg, len, MPI_INT, to_home_sent, root, MPI_COMM_WORLD);
27         }
28         else if (variant == "recv")
29         {
30             MPI_Recv(msg, len, MPI_INT, to_home_sent, root, MPI_COMM_WORLD, &status);
31         }
32     }
33     end_time = MPI_Wtime();
34
35     T = (end_time - start_time) * 1000000;
36     R = 2 * L * N / T / 1024 / 1024;
37     s = T / 2 / N;
38
39     if (variant == "send")
40     {
41         cout << "rank=" << (to_home_sent ^ 1);
42         cout << " len=" << len;
43         cout << " L(bytes)=" << L;
44         cout << " T(us)=" << T;
45         cout << " R[bandwidth](MB/s)=" << R;
46         cout << " s[latency](us)=" << s;
47         cout << '\n' << endl;
48     }
49 }
50
51 int main(int argc, char **argv)
52 {
53     int rank, n;
54     MPI_Init(&argc, &argv);
55     MPI_Comm_size(MPI_COMM_WORLD, &n);
56     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
57     MPI_Status status;
58     int *msg;
59
60     if (n != 2)
61     {
62         throw invalid_argument( "Only 2 processes is acceptable" );
63     }
64
65     for (int len = 1; len <= 1000000; len *= 10)
66     {
67         msg = new int[len]();
68         if (rank == 0)
69         {
70             for (int i = 0; i < len; i++)
71             {
72                 msg[i] = 1;
73             }
74
75             FactoryMessage("send", 1, len, msg, status);
76             FactoryMessage("recv", 1, len, msg, status);
77         }
78         else if (rank == 1)
79         {
80             FactoryMessage("recv", 0, len, msg, status);
81             FactoryMessage("send", 0, len, msg, status);
82         }
83         delete[] msg;
84     }
85
86     MPI_Finalize();
87     return 0;
88 }
```

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 receives 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 experiment!



Experiment 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 source code which is placed on my [github](#).