Metody programowania równoległego

Sprawozdanie z mierzenia opóźnienia i przepustowości w klastrze

autor: Ilona Tomkowicz

Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji,
Informatyka, II stopień, I semestr

08 marca 2020

Contents

1 Kody źródłowe **1**

1.1 Komunikacja z użyciem Send i Recv 1

1.2 Komunikacja z użyciem Isend i Irecv 2

1.3 Komunikacja z użyciem jednego węzła i pamięci współdzielonej 3

1.4 Komunikacja z użyciem jednego węzła i połączenia sieciowego 3

1.5 Komunikacja z użyciem dwóch węzłów, będących fizycznie na tej samej maszynie i połączenia sieciowego 4

1.6 Komunikacja z użyciem dwóch węzłów, będących fizycznie na różnych maszynach i połączenia sieciowego 4

2 Dane pomiarowe **5**

3 Wykresy dla różnych konfiguracji **9**

4 Wnioski **13**

1. Kody źródłowe

1.1 Komunikacja z użyciem Send i Recv

```
#include <stdio.h>
#include "mpi.h"

void get_device_info(int* rank, int*size)
{
    MPI_Comm_rank (MPI_COMM_WORLD, rank);  /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, size);  /* get number of processes */
    //printf("Current process id %d, number of processes %d.\n", *rank, *size);
}

int main(int argc, char** argv) {
    int rank, size, i;
    int          n = 10000;
    MPI_Status  status;
    double       start, elapsed;

    MPI_Init(&argc, &argv);
    get_device_info(&rank, &size);

    int len = 2<<13; //

    /* opoznienie */
    if (rank == 0)
    {
          int k[len];
          start = MPI_Wtime();
          for (i = 0; i < n; ++i)
          {
              k[0] = i;
              MPI_Send(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD);
              MPI_Recv(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD, &status);
              if (k[0] != i+1) printf("error\n");
          }
        elapsed = MPI_Wtime() - start;
        printf("av message of length %d has time of %g sec\n", len,  elapsed/(2*n));
        fflush(stdout);
```

```
    }

    /* przepustowosc */
    if (rank == 1)
    {
            int k[len];
            start = MPI_Wtime();
            for (i = 0; i < n; ++i)
            {  k[0] = i;
               MPI_Send(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD);
               MPI_Recv(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD, &status);
               if (k[0] != i+1) printf("error\n");
            }
            elapsed = MPI_Wtime() - start;
            long int s = sizeof(k);
            printf("av message of size %lu has %g Mbit/s\n",
                   8*s, 8*s/ (1000000 * elapsed/(2*n)));
            fflush(stdout);
       }
     MPI_Finalize();
     return 0;
}
```

1.2 Komunikacja z użyciem Isend i Irecv

```
#include <stdio.h>
#include "mpi.h"

void get_device_info(int* rank, int*size)
{
    MPI_Comm_rank (MPI_COMM_WORLD, rank);  /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, size);  /* get number of processes */
    //printf("Current process id %d, number of processes %d.\n", *rank, *size);
}

int main(int argc, char** argv) {
    int rank, size, i;
    int          n = 10000;
    MPI_Status  status;
    MPI_Request req1, req2;
    double       start_time, elapsed_time;

    MPI_Init(&argc, &argv);
    get_device_info(&rank, &size);

    int len = 2<<4;

    if (rank == 0)
    {
```

```
            int k[len];
            start_time = MPI_Wtime();
            for (i = 0; i < n; ++i)
            {
                k[0] = i;
                MPI_Isend(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD,&req1);
                MPI_Irecv(k, len, MPI_INT, 1, 123, MPI_COMM_WORLD, &req2);
                MPI_Wait(&req1, &status);
                if (k[0] != i) printf("error\n");
            }
          elapsed_time = MPI_Wtime() - start_time;
          printf("av message of length %d has time of %g sec\n", len,  elapsed/(2*n));
          fflush(stdout);

    }

    if (rank == 1)
    {
            int k[len];
            start_time = MPI_Wtime();
            for (i = 0; i < n; ++i)
            {  k[0] = i;
               MPI_Isend(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD, &req1);
               MPI_Irecv(k, len, MPI_INT, 0, 123, MPI_COMM_WORLD, &req2);
               MPI_Wait(&req2, &status);
               if (k[0] != i) printf("error\n");
            }
            elapsed_time = MPI_Wtime() - start_time;
            printf("av message of size %lu has %g Mbit/s\n",
                   8*s, 8*s/ (1000000 * elapsed/(2*n)));
            fflush(stdout);
       }
     MPI_Wait(&req1, &status);
     MPI_Wait(&req2, &status);
     MPI_Finalize();
     return 0;
}
```

1.3 Komunikacja z użyciem jednego węzła i pamięci współdzielonej

Konfiguracja pliku allnodes zawierała tylko adres tego węzła, a program uruchamiano na tym samym węźle.

```
vnode-01.dydaktyka.icsr.agh.edu.pl:4
```

1.4 Komunikacja z użyciem jednego węzła i połączenia sieciowego

Konfiguracja pliku allnodes zawierała tylko adres tego węzła, a program uruchamiano na innym węźle, w tym wypadku na 02.

1.5 Komunikacja z użyciem dwóch węzłów, będących fizycznie na tej samej maszynie i połączenia sieciowego

Konfiguracja pliku allnodes zawierała adresy używanych węzłów (01, 03), a program uruchamiano na węźle 02.

```
vnode-01.dydaktyka.icsr.agh.edu.pl:4
vnode-03.dydaktyka.icsr.agh.edu.pl:4
```

1.6 Komunikacja z użyciem dwóch węzłów, będących fizycznie na różnych maszynach i połączenia sieciowego

Konfiguracja pliku allnodes zawierała adresy używanych węzłów (05, 06), a program uruchamiano na węźle 02.

```
vnode-05.dydaktyka.icsr.agh.edu.pl
vnode-06.dydaktyka.icsr.agh.edu.pl
```

2. Dane pomiarowe

Pomiary wykonano dla paczek od 64 B do

Table 2.1: Jeden węzeł, pamięć współdzielona

rozmiar [bit]	Send/recv przepustowość [Mbit/s]
64	347.872
256	1328.89
1024	4148.32
4096	12235.5
16384	38741.4
65536	60750.5
262144	61720.4
1048576	51421.7

Table 2.2: Jeden węzeł, pamięć współdzielona

size [bit]	Isend/Irecv przepustowość [Mbit/s]
64	75.871
256	322.474
1024	1240.93
4096	7566.22
16384	27147.9
65536	81429.6
262144	119653
1048576	55264.7

Table 2.3: Jeden węzeł, połączenie sieciowe

size [bit]	Send/recv przepustowość [Mbit/s]
64	339.148
256	1354.45
1024	4119.87
4096	13120.9
16384	36961.9
65536	46702.7
262144	54996.8

Table 2.4: Jeden węzeł, połączenie sieciowe

size [bit]	Isend/Irecv przepustowość [Mbit/s]
64	74.3444
256	275.714
1024	1601.85
4096	4002.86
16384	25237.1
65536	71933.1
262144	75605.2
1048576	43859.2

Table 2.5: Dwa węzły, jeden host

size [bit]	Send/recv przepustowość [Mbit/s]
64	319.243
256	1156.55
1024	4638.69
4096	12530.4
16384	31242.5
65536	59722.3
262144	53988.5

Table 2.6: Dwa węzły, jeden host

size [bit]	Isend/Irecv przepustowość [Mbit/s]
64	74.64
256	337.766
1024	1003.36
4096	4407.19
16384	26874.5
65536	60076.7
262144	120085
1048576	36403.5

Table 2.7: Dwa węzły, różne hosty

size [bit]	Send/recv przepustowość [Mbit/s]
64	3.09821
256	12.1787
1024	49.11
4096	76.0858
16384	120.212
65536	440.153
262144	1206.13
524288	1917.71

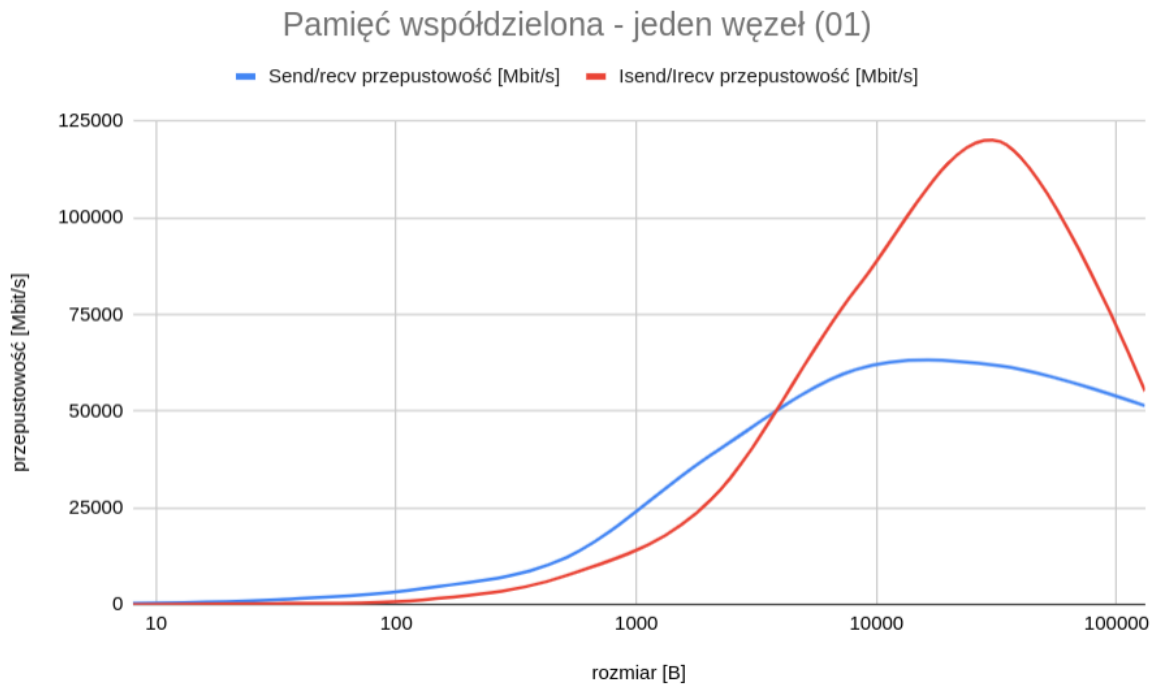
Table 2.8: Dwa węzły, różne hosty

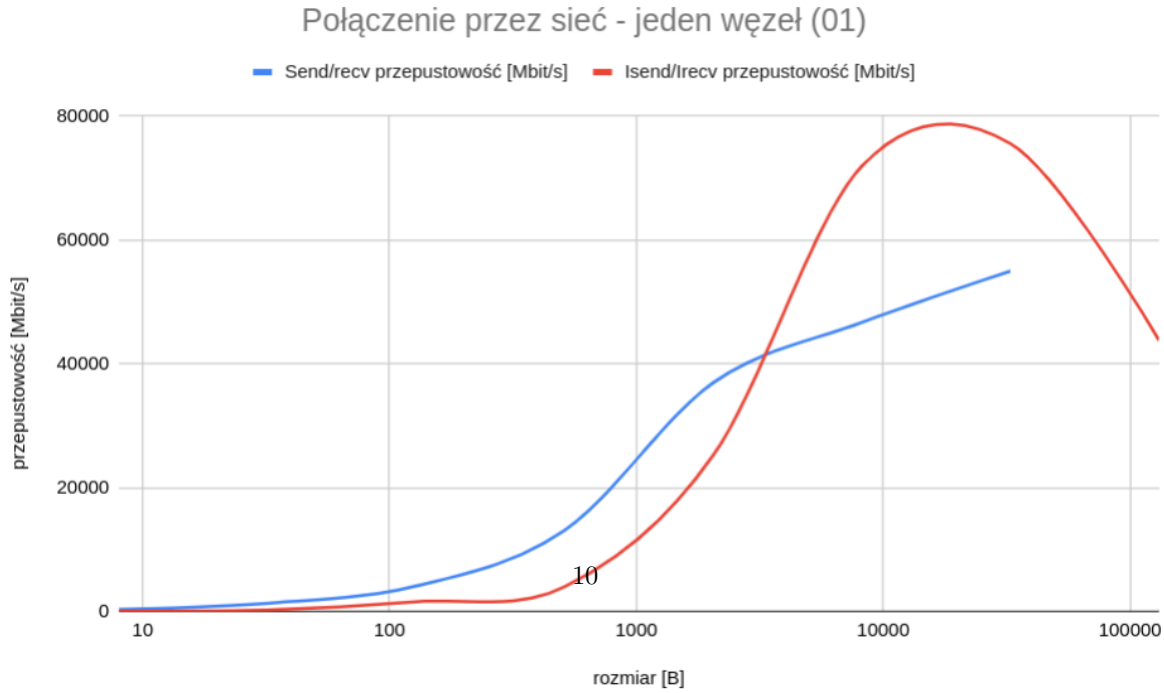
size [bit]	Isend/Irecv przepustowość [Mbit/s]
64	9.89011
256	36.2111
1024	92.1024
4096	360.693
16384	895.855
65536	2388.71
262144	2862.97
1048576	3153.6

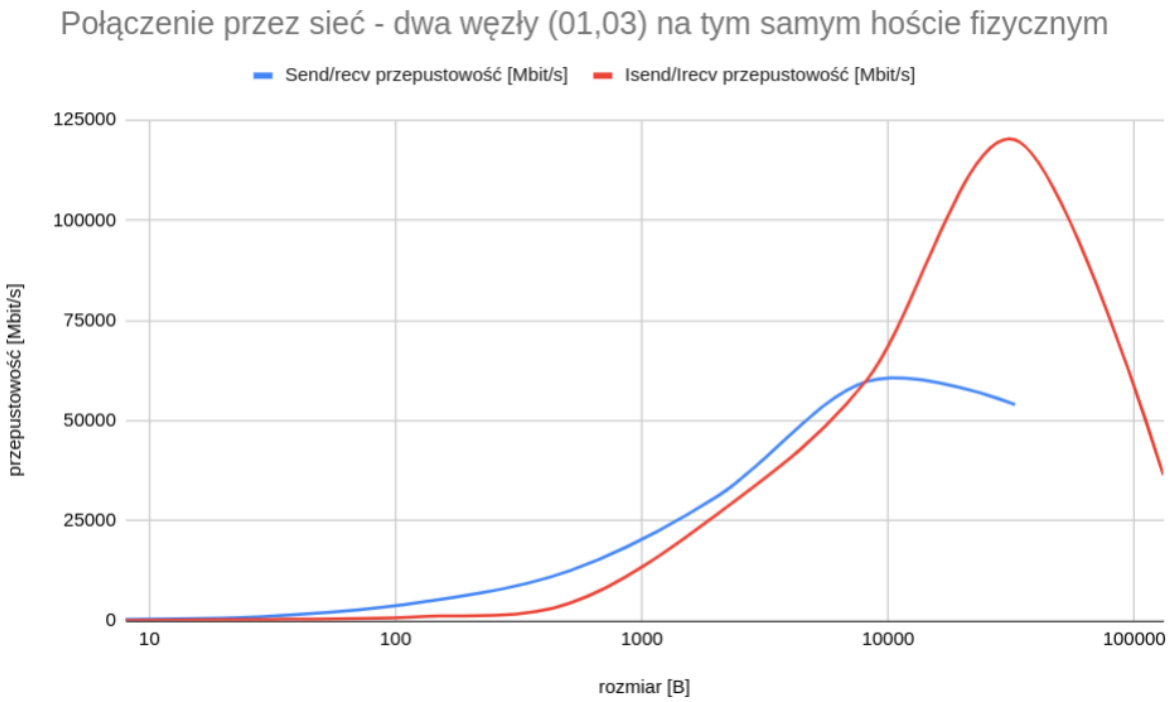
Table 2.9: Porównanie opóźnienia dla małej paczki 64 bitów

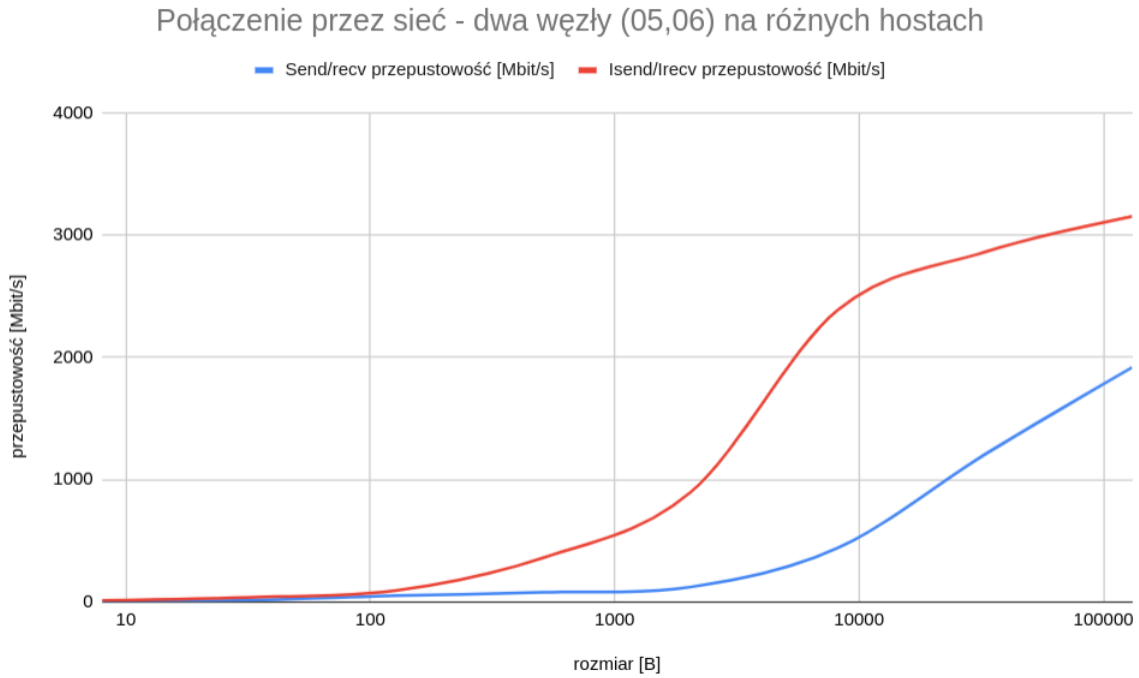
	Send+Recv	Isend+Irecv
zad_1	1.94335e-07 sec	7.73275e-07 sec
zad_2	2.21586e-07 sec	8.59272e-07 sec
zad_3	1.99199e-07 sec	8.57234e-07 sec
zad_4	2.06115e-05 sec	6.07103e-06 sec

3. Wykresy dla różnych konfiguracji






Połączenie przez sieć - dwa węzły (01,03) na tym samym hoście fizycznym
Send/recv przepustowość [Mbit/s]
Isend/Irecv przepustowość [Mbit/s]
125000
100000
75000
50000
25000
0
przepustowość [Mbit/s]
10
100
1000
10000
100000
rozmiar [B]


Połączenie przez sieć - dwa węzły (05,06) na różnych hostach
Send/recv przepustowość [Mbit/s]
Isend/Irecv przepustowość [Mbit/s]
4000
3000
2000
1000
0
przepustowość [Mbit/s]
10
100
1000
10000
100000
rozmiar [B]

4. Wnioski