TPR - SPRAWOZDANIE 1

MPI - Komunikacja PP

autor: Wojciech Buś

1. Cel ćwiczenia:

Celem ćwiczenia było przeprowadzenie analizy nad przepustowością oraz pingiem dwóch sposobów komunikacji w MPI. Przepustowość oraz ping pozwolą nam stwierdzić, jak wydajne (szybkie) są te metody.
Metody które wybrałem to:

- Send() - zwykłe wysyłanie
- Ssend() - synchronizowane wysyłanie

Wstępnie przypuszczam, iż Ssend() będzie wolniejszy dla każdej z konfiguracji opisanych w poleceniu. Synchroniczna komunikacja jest bezpieczniejsza od zwykłej, lecz niestety płaci za to szybkością. Coś za coś, zasada równorzędnej wymiany.

2. Kody programów:

Mierzenie przepustowości poprzez Send():

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

int main(int argc, char** argv) {
   MPI_Init(NULL, NULL);
   int world_rank;
   MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
   int world_size;
   MPI_Comm_size(MPI_COMM_WORLD, &world_size);
   int partner_rank = (world_rank + 1) % 2;

   if (world_size != 2) {
       printf("World size must be equal 2!\n");
       MPI_Abort(MPI_COMM_WORLD, 1);
   }

   int buffer_sizes[9] = {
           100,
           100,
           1000,
```

```
            10000,
            100000,
            1000000,
            10000000,
            100000000,
            1000000000
    };

    int no_sendings = 50;
    double Mb_in_byte = 125000.0;

    if (world_rank == 0) {
        MPI_Barrier(MPI_COMM_WORLD);
        for (int i = 0; i < 9; i++) {
            char* buffer = malloc(buffer_sizes[i]);
            double total_time = 0;

            for (int j = 0; j < no_sendings; j++) {
                double start = MPI_Wtime();
                MPI_Send(buffer, buffer_sizes[i], MPI_CHAR, partner_rank,
0, MPI_COMM_WORLD);
                double end = MPI_Wtime();
                total_time += end - start;
            }

            free(buffer);
            printf("Buffer nr: %d. Bandwidth is %lf Mb/s.\n", i,
(buffer_sizes[i] / Mb_in_byte) / total_time * no_sendings);
        }
    } else {
        MPI_Barrier(MPI_COMM_WORLD);
        for (int i = 0; i < 9; i++) {
            char* buffer = malloc(buffer_sizes[i]);

            for (int j = 0; j < no_sendings; j++) {
                MPI_Recv(buffer, buffer_sizes[i], MPI_CHAR, partner_rank,
0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            }

            free(buffer);
        }
    }
    MPI_Finalize();

    return 0;
```

```
}
```

Mierzenie pingu poprzez Send():

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

int main(int argc, char** argv) {
   MPI_Init(NULL, NULL);
   int world_rank;
   MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
   int world_size;
   MPI_Comm_size(MPI_COMM_WORLD, &world_size);
   int partner_rank = (world_rank + 1) % 2;

   if (world_size != 2) {
       printf("World size must be equal 2!\n");
       MPI_Abort(MPI_COMM_WORLD, 1);
   }

   int no_sendings = 10000;

   if (world_rank == 0) {
       MPI_Barrier(MPI_COMM_WORLD);
       char* buffer = malloc(1);
       double start = MPI_Wtime();

       for (int i = 0; i < no_sendings; i++) {
           MPI_Send(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD);
           MPI_Recv(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
       }

       double end = MPI_Wtime();
       free(buffer);
       printf("It tooks %lf.\n", (end - start) / 2);
   } else {
       MPI_Barrier(MPI_COMM_WORLD);
       char* buffer = malloc(1);

       for (int i = 0; i < no_sendings; i++) {
           MPI_Recv(buffer, 1, MPI_CHAR, partner_rank, 0,
```

```
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            MPI_Send(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD);
        }

        free(buffer);
    }
    MPI_Finalize();

    return 0;
}
```

Mierzenie przepustowości poprzez Ssend():

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

int main(int argc, char** argv) {
    MPI_Init(NULL, NULL);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    int partner_rank = (world_rank + 1) % 2;

    if (world_size != 2) {
        printf("World size must be equal 2!\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    int buffer_sizes[9] = {
            100,
            100,
            1000,
            10000,
            100000,
            1000000,
            10000000,
            100000000,
            1000000000
    };

    int no_sendings = 50;
    double Mb_in_byte = 125000.0;
```

```
   if (world_rank == 0) {
       MPI_Barrier(MPI_COMM_WORLD);
       for (int i = 0; i < 9; i++) {
           char* buffer = malloc(buffer_sizes[i]);
           double total_time = 0;


           for (int j = 0; j < no_sendings; j++) {
               double start = MPI_Wtime();
               MPI_Ssend(buffer, buffer_sizes[i], MPI_CHAR,
partner_rank, 0, MPI_COMM_WORLD);
               double end = MPI_Wtime();
               total_time += end - start;
           }

           free(buffer);
           printf("Buffer nr: %d. Bandwidth is %lf Mb/s.\n", i,
(buffer_sizes[i] / Mb_in_byte) / total_time * no_sendings);
       }
   } else {
       MPI_Barrier(MPI_COMM_WORLD);
       for (int i = 0; i < 9; i++) {
           char* buffer = malloc(buffer_sizes[i]);

           for (int j = 0; j < no_sendings; j++) {
               MPI_Recv(buffer, buffer_sizes[i], MPI_CHAR, partner_rank,
0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
           }

           free(buffer);
       }
   }
   MPI_Finalize();

   return 0;
}
```

Mierzenie pingu poprzez Ssend():

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

int main(int argc, char** argv) {
```

```
    MPI_Init(NULL, NULL);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    int partner_rank = (world_rank + 1) % 2;

    if (world_size != 2) {
        printf("World size must be equal 2!\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    int no_sendings = 10000;

    if (world_rank == 0) {
        MPI_Barrier(MPI_COMM_WORLD);
        char* buffer = malloc(1);
        double start = MPI_Wtime();

        for (int i = 0; i < no_sendings; i++) {
            MPI_Ssend(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD);
            MPI_Recv(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        }

        double end = MPI_Wtime();
        free(buffer);
        printf("It tooks %lf.\n", (end - start) / 2);
    } else {
        MPI_Barrier(MPI_COMM_WORLD);
        char* buffer = malloc(1);

        for (int i = 0; i < no_sendings; i++) {
            MPI_Recv(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            MPI_Ssend(buffer, 1, MPI_CHAR, partner_rank, 0,
MPI_COMM_WORLD);
        }

        free(buffer);
    }
    MPI_Finalize();

    return 0;
```

```
}
```

3. Pliki konfiguracyjne - machinefiles:

Przygotowane zostały pliki:

```
nodes1:
    vnode-01.dydaktyka.icsr.agh.edu.pl
nodes2:
    vnode-01.dydaktyka.icsr.agh.edu.pl:2
nodes3:
    vnode-02.dydaktyka.icsr.agh.edu.pl
    vnode-03.dydaktyka.icsr.agh.edu.pl
nodes4:
    vnode-06.dydaktyka.icsr.agh.edu.pl
    vnode-09.dydaktyka.icsr.agh.edu.pl
```

4. Pomiary i wyniki - ping:

Dla Send() otrzymaliśmy dla kolejnych plików konfiguracyjnych:

```
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes1 -np 2 ./send_ping.exe
It tooks 0.002929.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes2 -np 2 ./send_ping.exe
It tooks 0.002799.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes3 -np 2 ./send_ping.exe
It tooks 0.841481.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes4 -np 2 ./send_ping.exe
It tooks 0.484525.
```

Dla Ssend() otrzymaliśmy dla kolejnych plików konfiguracyjnych:

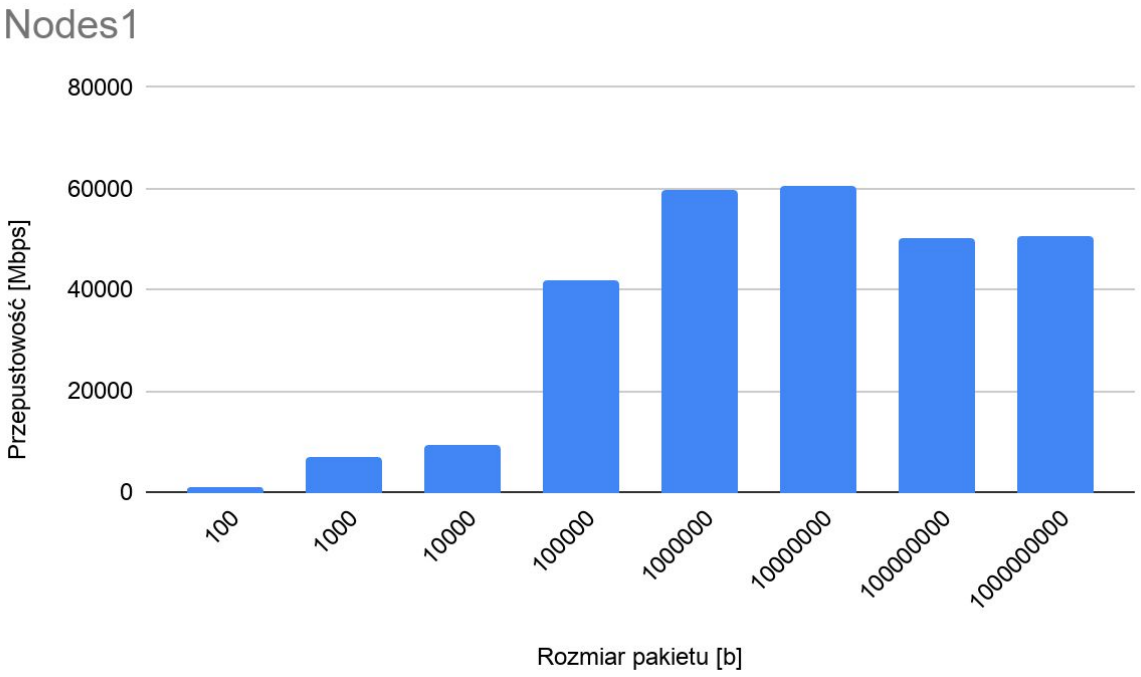
```
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes1 -np 2 ./synchronous_send_ping.exe
It tooks 0.006146.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes2 -np 2 ./synchronous_send_ping.exe
It tooks 0.006136.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes3 -np 2 ./synchronous_send_ping.exe
It tooks 2.945683.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes4 -np 2 ./synchronous_send_ping.exe
It tooks 1.700725.
```

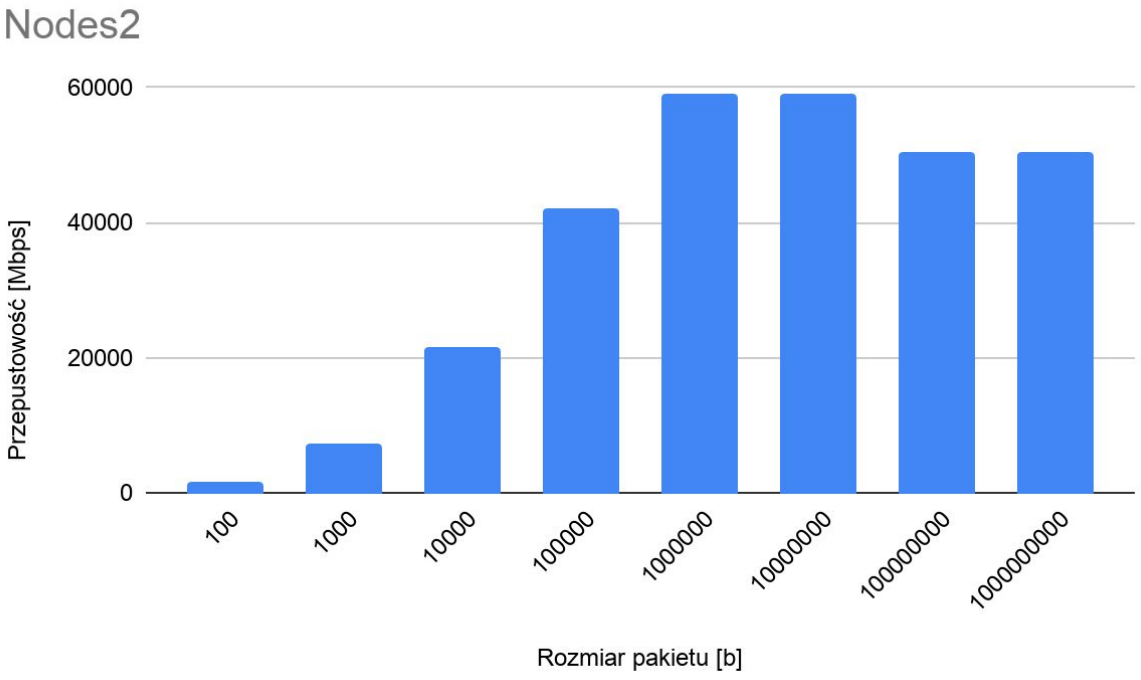
5. Pomiary i wyniki - przepustowość:

Dla Send() otrzymaliśmy dla kolejnych plików konfiguracyjnych:

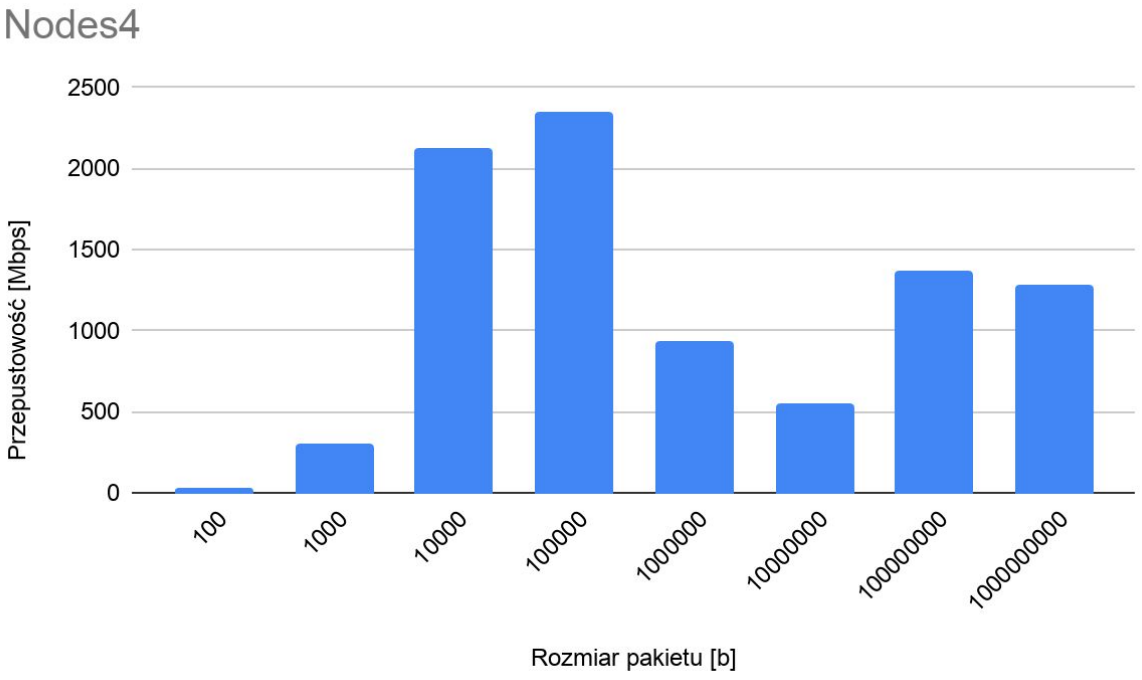
```
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes1 -np 2 ./send_bandwidth.exe
Buffer nr: 0. Bandwidth is 2071.261235 Mb/s.
Buffer nr: 1. Bandwidth is 1189.873475 Mb/s.
Buffer nr: 2. Bandwidth is 7078.994093 Mb/s.
Buffer nr: 3. Bandwidth is 9425.402247 Mb/s.
Buffer nr: 4. Bandwidth is 42037.624655 Mb/s.
Buffer nr: 5. Bandwidth is 59618.407306 Mb/s.
Buffer nr: 6. Bandwidth is 60583.532844 Mb/s.
Buffer nr: 7. Bandwidth is 50335.488598 Mb/s.
Buffer nr: 8. Bandwidth is 50412.377373 Mb/s.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes2 -np 2 ./send_bandwidth.exe
Buffer nr: 0. Bandwidth is 2123.698228 Mb/s.
Buffer nr: 1. Bandwidth is 1677.721600 Mb/s.
Buffer nr: 2. Bandwidth is 7326.295197 Mb/s.
Buffer nr: 3. Bandwidth is 21704.031048 Mb/s.
Buffer nr: 4. Bandwidth is 42206.832704 Mb/s.
Buffer nr: 5. Bandwidth is 58964.664534 Mb/s.
Buffer nr: 6. Bandwidth is 59197.685332 Mb/s.
Buffer nr: 7. Bandwidth is 50551.917872 Mb/s.
Buffer nr: 8. Bandwidth is 50431.399796 Mb/s.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes3 -np 2 ./send_bandwidth.exe
Buffer nr: 0. Bandwidth is 173.497580 Mb/s.
Buffer nr: 1. Bandwidth is 181.375308 Mb/s.
Buffer nr: 2. Bandwidth is 1229.100073 Mb/s.
Buffer nr: 3. Bandwidth is 12201.611636 Mb/s.
Buffer nr: 4. Bandwidth is 6408.899076 Mb/s.
Buffer nr: 5. Bandwidth is 8967.744970 Mb/s.
Buffer nr: 6. Bandwidth is 9679.505583 Mb/s.
Buffer nr: 7. Bandwidth is 3300.340452 Mb/s.
Buffer nr: 8. Bandwidth is 3695.927611 Mb/s.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes4 -np 2 ./send_bandwidth.exe
Buffer nr: 0. Bandwidth is 13.392844 Mb/s.
Buffer nr: 1. Bandwidth is 27.598645 Mb/s.
Buffer nr: 2. Bandwidth is 301.748489 Mb/s.
Buffer nr: 3. Bandwidth is 2128.548084 Mb/s.
Buffer nr: 4. Bandwidth is 2350.474376 Mb/s.
Buffer nr: 5. Bandwidth is 937.804313 Mb/s.
Buffer nr: 6. Bandwidth is 551.485090 Mb/s.
Buffer nr: 7. Bandwidth is 1371.547051 Mb/s.
Buffer nr: 8. Bandwidth is 1290.630809 Mb/s.
```

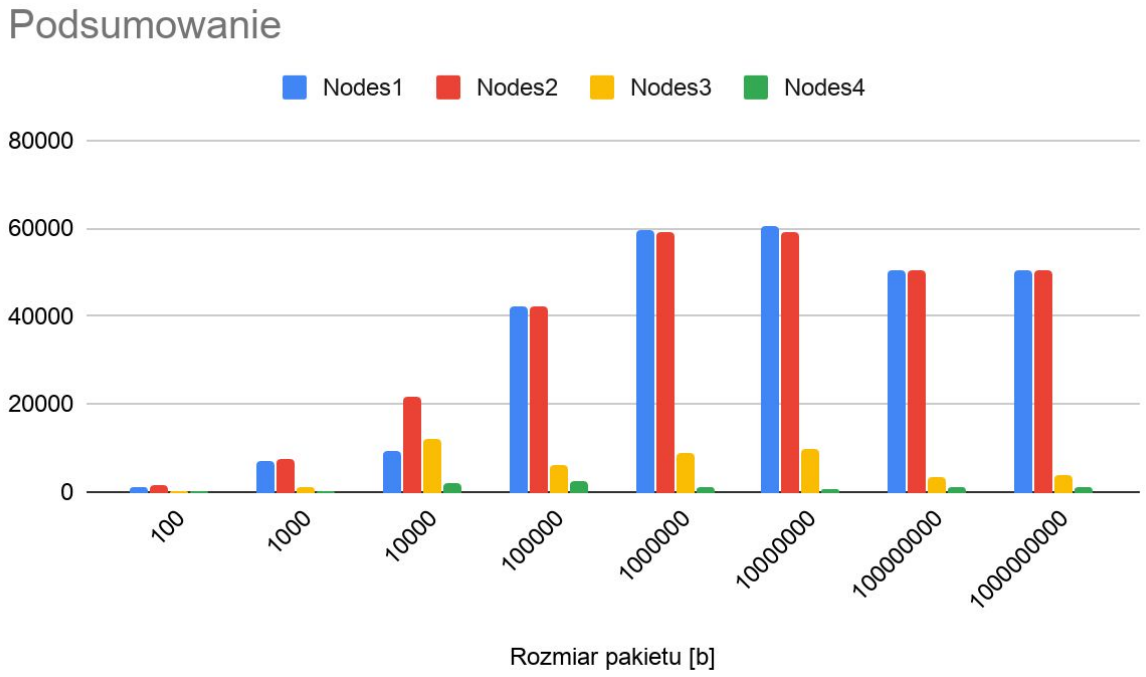
A na wykresach:


Nodes1
80000
60000
40000
20000
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]


Nodes2
60000
40000
20000
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]


Nodes3
12500
10000
7500
5000
2500
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]

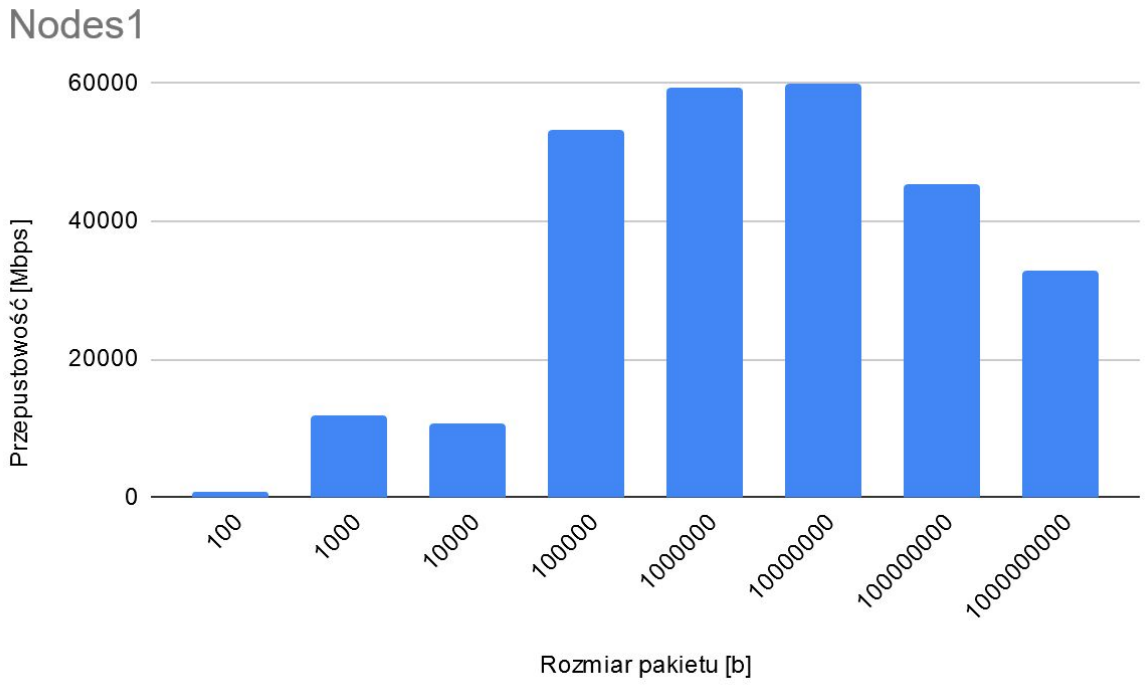

Nodes4
2500
2000
1500
1000
500
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]

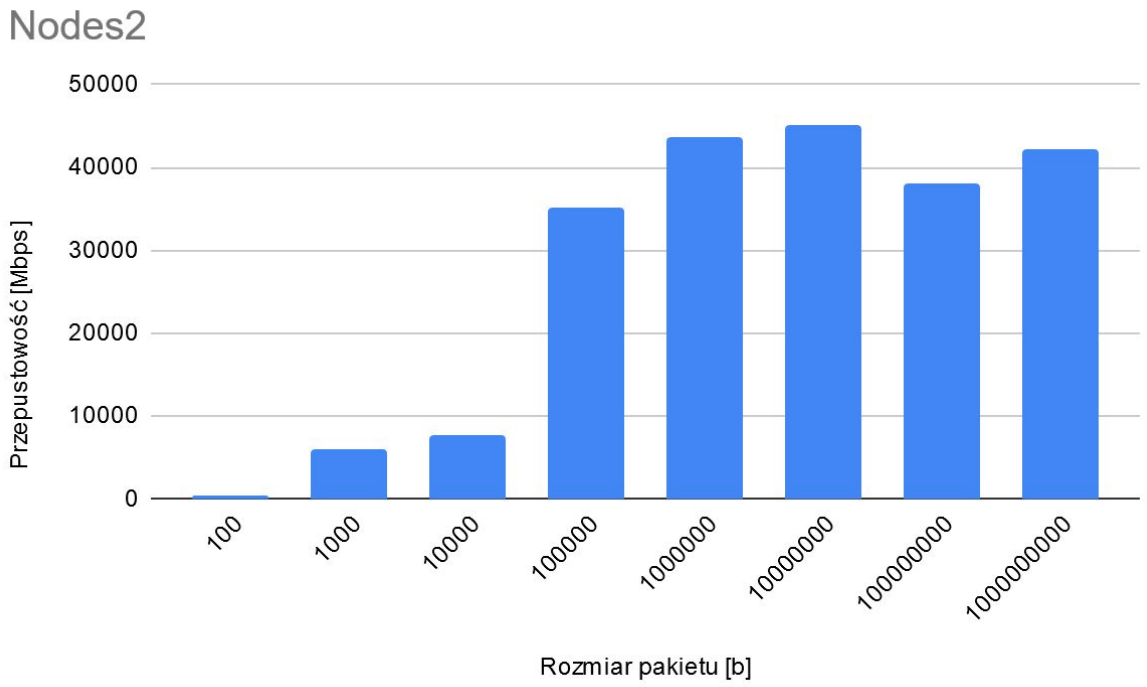


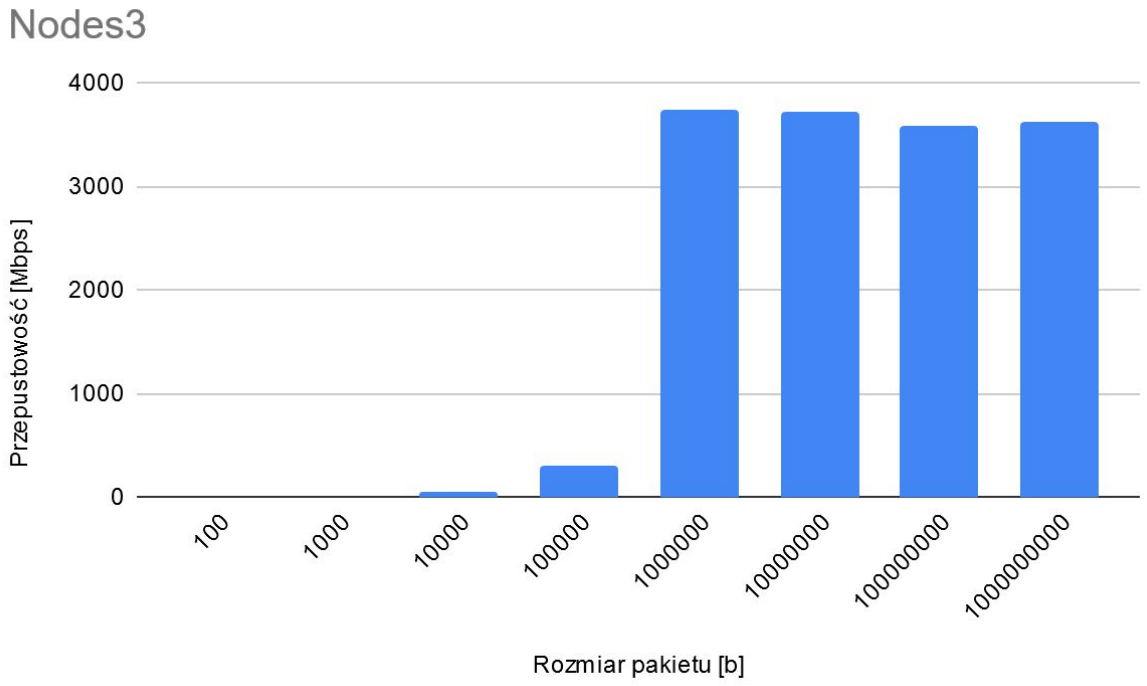
Dla Ssend() otrzymaliśmy dla kolejnych plików konfiguracyjnych:

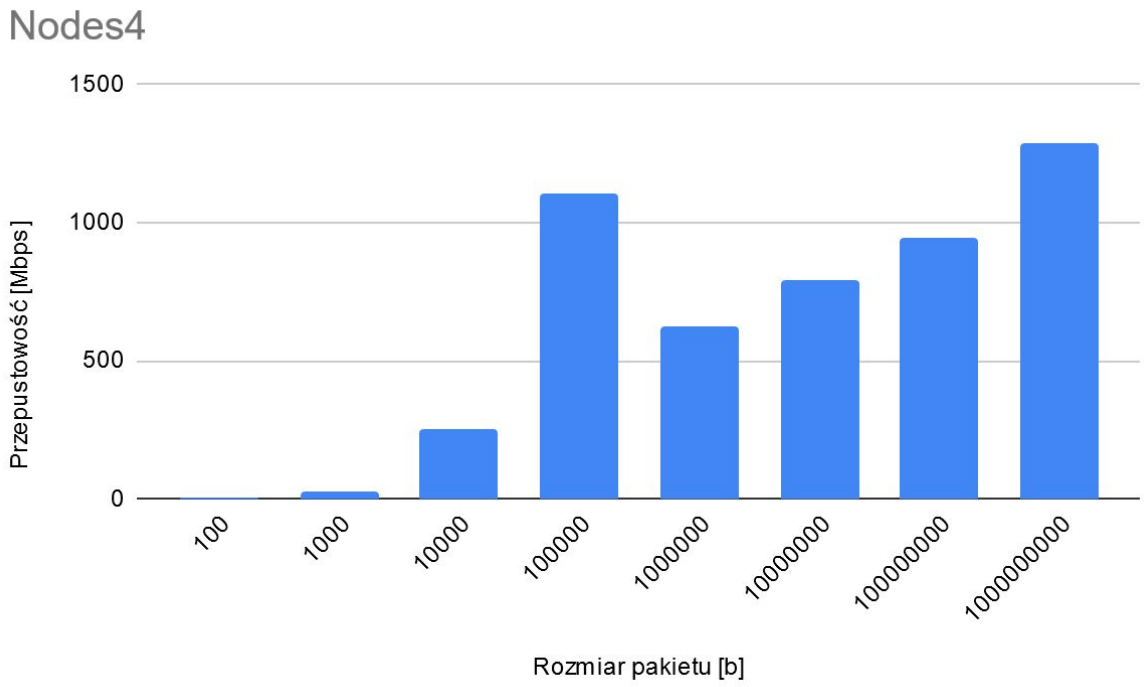
```
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes1 -np 2 ./synchronous_send_bandwidth.exe
Buffer nr: 0. Bandwidth is 366.314760 Mb/s.
Buffer nr: 1. Bandwidth is 682.000650 Mb/s.
Buffer nr: 2. Bandwidth is 11732.318881 Mb/s.
Buffer nr: 3. Bandwidth is 10834.495318 Mb/s.
Buffer nr: 4. Bandwidth is 53337.199173 Mb/s.
Buffer nr: 5. Bandwidth is 59314.887750 Mb/s.
Buffer nr: 6. Bandwidth is 59840.942205 Mb/s.
Buffer nr: 7. Bandwidth is 45320.297219 Mb/s.
Buffer nr: 8. Bandwidth is 32840.228941 Mb/s.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes2 -np 2 ./synchronous_send_bandwidth.exe
Buffer nr: 0. Bandwidth is 263.378587 Mb/s.
Buffer nr: 1. Bandwidth is 451.000430 Mb/s.
Buffer nr: 2. Bandwidth is 5970.539502 Mb/s.
Buffer nr: 3. Bandwidth is 7796.104089 Mb/s.
Buffer nr: 4. Bandwidth is 35146.571698 Mb/s.
Buffer nr: 5. Bandwidth is 43710.586856 Mb/s.
Buffer nr: 6. Bandwidth is 45120.662668 Mb/s.
Buffer nr: 7. Bandwidth is 37976.688223 Mb/s.
Buffer nr: 8. Bandwidth is 42131.231257 Mb/s.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes3 -np 2 ./synchronous_send_bandwidth.exe
Buffer nr: 0. Bandwidth is 0.473129 Mb/s.
Buffer nr: 1. Bandwidth is 0.363724 Mb/s.
Buffer nr: 2. Bandwidth is 2.867640 Mb/s.
Buffer nr: 3. Bandwidth is 49.071396 Mb/s.
Buffer nr: 4. Bandwidth is 314.987008 Mb/s.
Buffer nr: 5. Bandwidth is 3743.418589 Mb/s.
Buffer nr: 6. Bandwidth is 3729.978552 Mb/s.
Buffer nr: 7. Bandwidth is 3599.998477 Mb/s.
Buffer nr: 8. Bandwidth is 3623.346286 Mb/s.
[wbus@vnode-01 tpr_lab]$ mpiexec -machinefile ./nodes4 -np 2 ./synchronous_send_bandwidth.exe
Buffer nr: 0. Bandwidth is 8.218284 Mb/s.
Buffer nr: 1. Bandwidth is 4.538246 Mb/s.
Buffer nr: 2. Bandwidth is 27.247905 Mb/s.
Buffer nr: 3. Bandwidth is 254.717396 Mb/s.
Buffer nr: 4. Bandwidth is 1107.699459 Mb/s.
Buffer nr: 5. Bandwidth is 624.972890 Mb/s.
Buffer nr: 6. Bandwidth is 789.452352 Mb/s.
Buffer nr: 7. Bandwidth is 944.086570 Mb/s.
Buffer nr: 8. Bandwidth is 1104.407191 Mb/s.
```

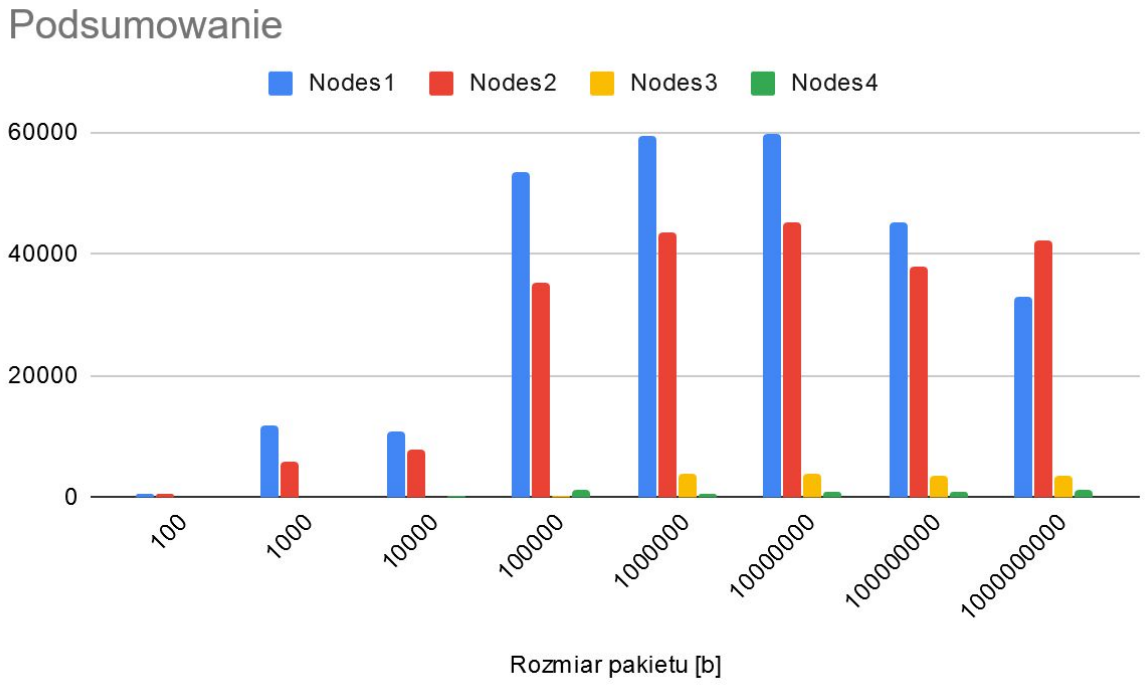
A na wykresach:


Nodes1
60000
40000
20000
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]


Nodes2
50000
40000
30000
20000
10000
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]


Nodes3
4000
3000
2000
1000
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]


Nodes4
1500
1000
500
0
Przepustowość [Mbps]
100
1000
10000
100000
1000000
10000000
100000000
1000000000
Rozmiar pakietu [b]



6. Podsumowanie

Ping:

W pingu dobrze widać, iż przy małych pakietach (1 bajt) narzut Ssenda jest znaczący. Wynika to z tego, iż przed wysłaniem pakietu, nadawca musi się wstępnie przywitać z odbiorcą (żeby ustanowić zsynchronizowane połączenie.
Z ogólniejszych spostrzeżeń, bardzo ciekawe jest to, iż ping między 2 nodami na tym samym hoście fizycznym jest wolniejszy od tego, gdzie nodey znajdują się na różnych hostach.

Bandwidth:

Send() - Tutaj widzimy, że przy zwykłym sendzie, nie ma różnicy czy komunikujemy się przez sieć, czy nie na jednym nodzie. Różnice pokazują się dopiero przy komunikacji między dwoma node'ami. Nie dość, że sama komunikacja jest wolniejsza pomiędzy dwoma node'ami to jeszcze, gdy znajdują się one na różnych fizycznych hostach to jest jeszcze wolniejsza.
Ssend() - Jeśli chodzi o komunikację na jednym nodzie, to komunikacja przez sieć wypadła gorzej od tej ze wspólną pamięcią (najprawdopodobniej ma to związek z tym narzutem o którym pisałem przy pingu). Co do komunikacji między dwoma node'ami - wnioski podobne jak przy Send().

Komunikacja przez Send() jest zdecydowanie szybsza niż przez Ssend().
Największą przepustowość uzyskano co ciekawe dla pakietów o wielkości 1Mb - 10Mb. Nie jest więc prawdą stwierdzenie, iż większe pakiety dadzą większą przepustowość przez mniejszy narzut związany z “samym nawiązaniem kontaktu nadawcy z odbiorcą”.