* Michał Dorosz
* Grzegorz Denert

**Sprawozdanie**

1. **Treść zadania**

Program typu ping-pong. Aplikacja MPI składa się z dwóch procesów: P0 oraz P1. Obydwa procesy wymieniają się komunikatami stosując wzorzec wymiany jak w grze w ping-ponga. P0 wysyła komunikat do P1 następnie P1 odsyła komunikat do P0. Ta sekwencja wymiany jest powtarzana odpowiednią (bardzo dużą) liczbę iteracji, tak aby program wykonywał się przez kilka sekund. Czas pracy programu jest obliczany przy pomocy funkcji MPI\_Wtime. Dzieląc ten czas przez (liczbę iteracji pomnożoną przez dwa) (ponieważ komunikat podróżuje w obie strony) otrzymujemy czas transmisji komunikatu w jedną stronę. Program powinien wykonywać szereg takich pomiarów dla rosnącej długości komunikatów, tak abyś mógł w stanie sporządzić wykres czas\_komunikatu=f(długość).

Z kolei dzieląc długość komunikatu przez czas jego transmisji otrzymujemy przepustowość w bajtach/s dla danej długości komunikatu. Należy sporządzić również wykres przepustowość=f(długość). Program powinien implementować test ping-pong w dwóch wersjach: przy pomocy blokujących operacji MPI\_Send/MPI\_Recv oraz przy pomocy nieblokujących MPI\_Isend/MPI\_Irecv wraz z MPI\_Wait. **Wykonaj pomiar na klastrze.**i sporządź raport z wykonania zadania zawierający wydruk programu + obydwa wykresy. Porównaj otrzymane wyniki z wynikiem polecenia  ping oraz z parametrami sieci Gigabit Ethernet. Porównaj wynik na czas transmisji komunikatu z modelem przedstawionym na wykładzie. Czy przy pomocy MPI możliwe jest osiągnięcie maksymalnej teoretycznej przepustowości sieci ? Jeżeli nie to wymień przyczyny, które Twoim zdaniem to uniemożliwiają.

**Ważne:**

Aby zmierzyć przepustowość sieci Gigabit Ethernet, należy zadbać, aby obydwa procesy aplikacji wykonywały się na róznych komputerach połączonych tą siecią. W klastrze w sali A226 możemy to osiągnąć poprzez następujące polecenia systemu SLURM:

#SBATCH -n 2  
#SBATCH -c 6

1. **Realizacja**

***Kod źródłowy 1.*** *Test ping-ponga z użyciem operacji blokujących*

|  |
| --- |
| #include <mpi.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  int main(int argc, char \*argv[])  {      const int PING\_PONG\_LIMIT = 10;      int npes;      int myrank;      MPI\_Init(&argc, &argv);      MPI\_Comm\_size(MPI\_COMM\_WORLD, &npes);      MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myrank);       if (argc < 2) {          if (myrank == 0) {              fprintf(stderr, "Użycie: %s <rozmiar\_komunikatu>\n", argv[0]);          }          MPI\_Finalize();          return 1;      }      int dlugoscKomunikatu = atoi(argv[1]);      if (dlugoscKomunikatu <= 0) {          if (myrank == 0) {              fprintf(stderr, "Podano niepoprawny rozmiar komunikatu: %d\n", dlugoscKomunikatu);          }          MPI\_Finalize();          return 1;      }      char \*komunikatDoWyslania = (char \*)malloc(dlugoscKomunikatu \* sizeof(char));      char \*komunikatPobrany = (char \*)malloc(dlugoscKomunikatu \* sizeof(char));      memset(komunikatDoWyslania, 'x', dlugoscKomunikatu - 1);      komunikatDoWyslania[dlugoscKomunikatu - 1] = '\0';      double t1 = MPI\_Wtime();      int ping\_pong\_count = 0;      while (ping\_pong\_count < PING\_PONG\_LIMIT)      {          if (myrank == 0)          {              MPI\_Status status;              MPI\_Send(komunikatDoWyslania, dlugoscKomunikatu, MPI\_CHAR, 1, 13, MPI\_COMM\_WORLD);              MPI\_Recv(komunikatPobrany, dlugoscKomunikatu, MPI\_CHAR, 1, 14, MPI\_COMM\_WORLD, &status);              ping\_pong\_count ++;              printf("%d",ping\_pong\_count);          }          else if (myrank == 1)          {              MPI\_Status status;              MPI\_Recv(komunikatPobrany, dlugoscKomunikatu, MPI\_CHAR, 0, 13, MPI\_COMM\_WORLD, &status);              MPI\_Send(komunikatDoWyslania, dlugoscKomunikatu, MPI\_CHAR, 0, 14, MPI\_COMM\_WORLD);              ping\_pong\_count ++;              printf("%d",ping\_pong\_count);          }      }      double t2 = MPI\_Wtime();      printf("Czas obliczen: %f sekund\n", t2 - t1);      MPI\_Finalize();      return 0;  } |

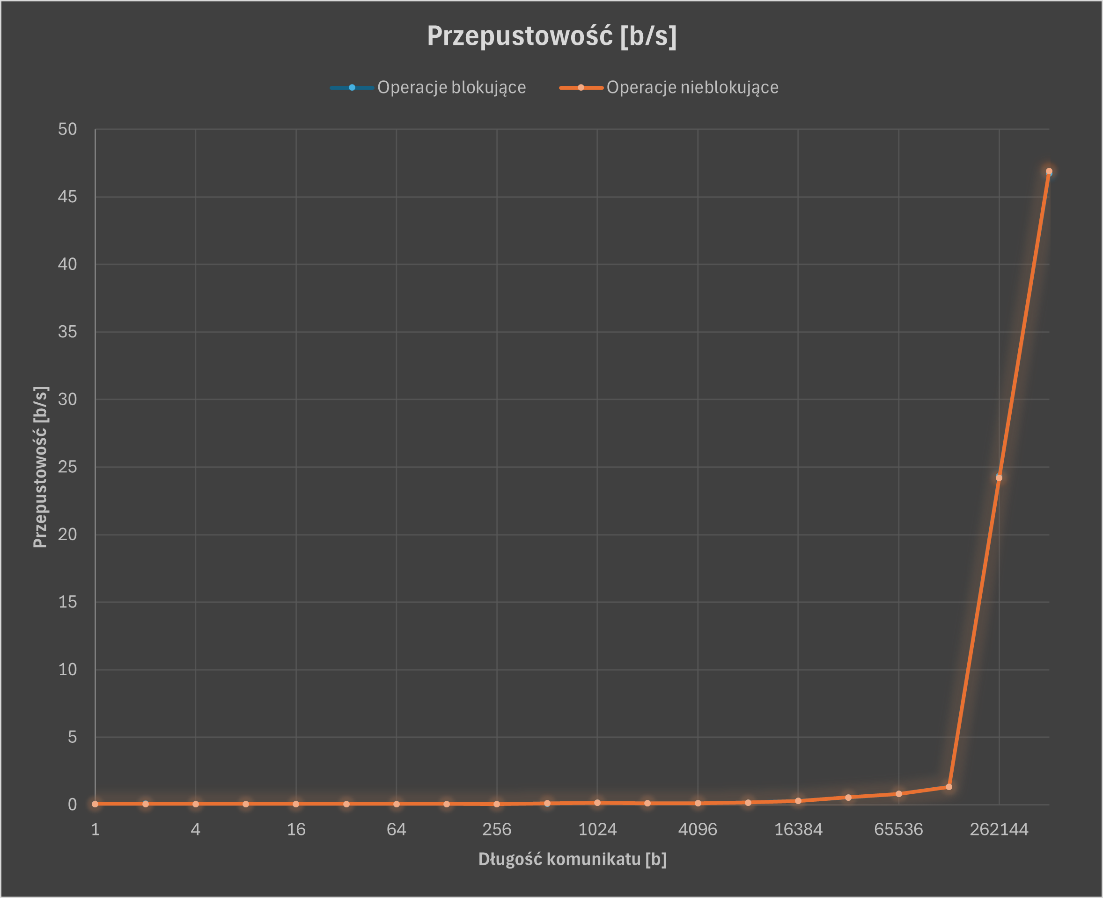
***Kod źródłowy 2.*** *Test ping-ponga z użyciem operacji nieblokujących*

|  |
| --- |
| #include <mpi.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  int main(int argc, char \*argv[])      {      const int PING\_PONG\_LIMIT = 10000;      int npes;      int myrank;      MPI\_Init(&argc, &argv);      MPI\_Comm\_size(MPI\_COMM\_WORLD, &npes);      MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myrank);      if (argc < 2) {          if (myrank == 0) {          fprintf(stderr, "Użycie: %s \n", argv[0]);          }          MPI\_Finalize();          return 1;      }      int dlugoscKomunikatu = atoi(argv[1]);      if (dlugoscKomunikatu <= 0) {          if (myrank == 0) {              fprintf(stderr, "Podano niepoprawny rozmiar komunikatu: %d\n", dlugoscKomunikatu);          }          MPI\_Finalize();          return 1;      }      char \*komunikatDoWyslania = (char \*)malloc(dlugoscKomunikatu \* sizeof(char));      char \*komunikatPobrany = (char \*)malloc(dlugoscKomunikatu \* sizeof(char));      memset(komunikatDoWyslania, 'x', dlugoscKomunikatu - 1);      komunikatDoWyslania[dlugoscKomunikatu - 1] = '\0';      double t1 = MPI\_Wtime();      int ping\_pong\_count = 0;      while (ping\_pong\_count < PING\_PONG\_LIMIT) {          MPI\_Request reqtab[2];          MPI\_Status stattab[2];          if (myrank == 0) {          // Proces 0 wysyła i odbiera          MPI\_Isend(komunikatDoWyslania, dlugoscKomunikatu, MPI\_CHAR, 1, 13, MPI\_COMM\_WORLD, &reqtab[0]);          MPI\_Wait(&reqtab[0], &stattab[0]);          MPI\_Irecv(komunikatPobrany, dlugoscKomunikatu, MPI\_CHAR, 1, 14, MPI\_COMM\_WORLD, &reqtab[1]);          MPI\_Wait(&reqtab[1], &stattab[1]);          ping\_pong\_count++;          } else if (myrank == 1) {          // Proces 1 odbiera i wysyła          MPI\_Irecv(komunikatPobrany, dlugoscKomunikatu, MPI\_CHAR, 0, 13, MPI\_COMM\_WORLD, &reqtab[0]);          MPI\_Wait(&reqtab[0], &stattab[0]);          MPI\_Isend(komunikatDoWyslania, dlugoscKomunikatu, MPI\_CHAR, 0, 14, MPI\_COMM\_WORLD, &reqtab[1]);          MPI\_Wait(&reqtab[1], &stattab[1]);          ping\_pong\_count++;          }      }      double t2 = MPI\_Wtime();      printf("Proces %d zakończył. Czas obliczeń: %f sekund\n", myrank, t2 - t1);      // Zwolnij pamięć      free(komunikatDoWyslania);      free(komunikatPobrany);      MPI\_Finalize();      return 0;  } |

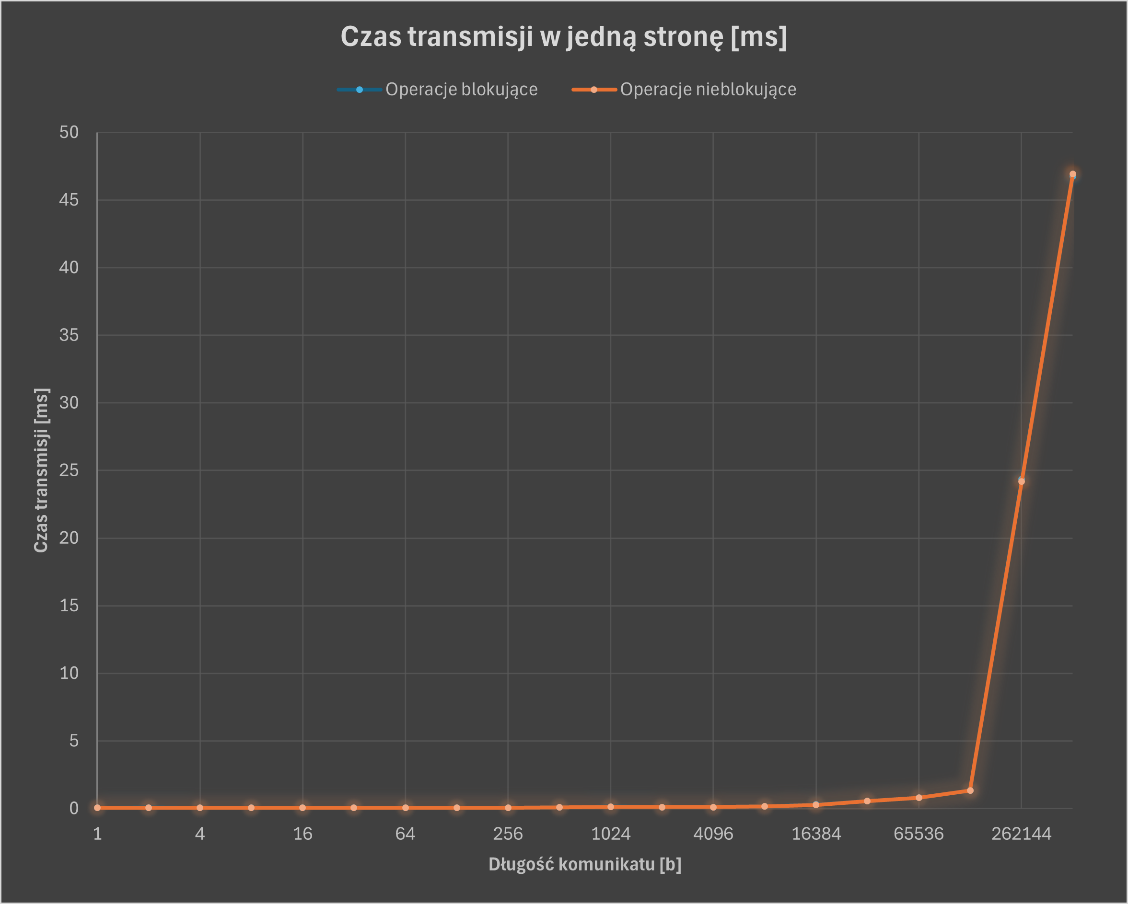
**Otrzymane pomiary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Operacje blokujące** | | **Operacje nieblokujące** | | |
| **Długość komunikatu [b]** | **Czas transmisji w jedną stronę [ms]** | **Przepustowość [b/s]** | **Czas transmisji w jedną stronę [ms]** | | **Przepustowość [b/s]** |
| 1 | 0,06435 | 0,01554 | 0,06475 | 0,01544 | |
| 2 | 0,07497 | 0,02668 | 0,06456 | 0,03098 | |
| 4 | 0,06459 | 0,06193 | 0,06466 | 0,06186 | |
| 8 | 0,06464 | 0,12377 | 0,06468 | 0,12368 | |
| 16 | 0,06463 | 0,24756 | 0,06281 | 0,25474 | |
| 32 | 0,06490 | 0,49305 | 0,06279 | 0,50963 | |
| 64 | 0,06297 | 1,01631 | 0,06278 | 1,01940 | |
| 128 | 0,06499 | 1,96947 | 0,06516 | 1,96440 | |
| 256 | 0,06149 | 4,16358 | 0,06113 | 4,18763 | |
| 512 | 0,09729 | 5,26237 | 0,10833 | 4,72623 | |
| 1024 | 0,14460 | 7,08172 | 0,14392 | 7,11520 | |
| 2048 | 0,11368 | 18,01544 | 0,11218 | 18,25648 | |
| 4096 | 0,11402 | 35,92422 | 0,11122 | 36,82920 | |
| 8192 | 0,17086 | 47,94598 | 0,16575 | 49,42482 | |
| 16384 | 0,30583 | 53,57307 | 0,27801 | 58,93251 | |
| 32768 | 0,54094 | 60,57638 | 0,54937 | 59,64704 | |
| 65536 | 0,79870 | 82,05352 | 0,79975 | 81,94519 | |
| 131072 | 1,34687 | 97,31589 | 1,33953 | 97,84918 | |
| 262144 | 24,34687 | 107,67052 | 24,18823 | 108,37667 | |
| 524288 | 46,73739 | 112,17742 | 46,92036 | 111,73999 | |

***Tabela 1.*** *Długość transmisji oraz przepustowość względem długości komunikatu.*

****

***Wykres 1.*** *Przedstawia przepustowość przy operacjach blokujących i nieblokujących, względem długości komunikatu*

****

***Wykres 2.*** *Przedstawia czas transmisji danych w jedną stronę przy operacjach blokujących i nieblokujących, względem długości komunikatu*

**Wynik działania polecenia ping wywołanego na jednym z komputerów pracowni**

|  |
| --- |
| **ping** [**192.168.0.112**](http://192.168.0.112/)  **PING** [**192.168.0.112**](http://192.168.0.112/) **(**[**192.168.0.112**](http://192.168.0.112/)**) 56(84) bytes of data.**  **64 bytes from** [**192.168.0.112**](http://192.168.0.112/)**: icmp\_seq=1 ttl=64 time=0.861 ms**  **64 bytes from** [**192.168.0.112**](http://192.168.0.112/)**: icmp\_seq=2 ttl=64 time=0.353 ms**  **64 bytes from** [**192.168.0.112**](http://192.168.0.112/)**: icmp\_seq=3 ttl=64 time=0.458 ms** |

1. **Wnioski**

* Przepustowość oraz czas transmisji danych są bardzo zbliżone przy operacjach blokujących i nieblokujących.
* Wraz ze wzrostem długości komunikatu rośnie przepustowość sieci
* Przesyłanie komunikatu o długości 84 bitów poleceniem ping trwa zwykle ok. 0,5ms, wynik ten zbliżony jest do uzyskanych przez nas pomiarów.