Grupa 4	Barbara Nowak, Piotr Tomaszewski	06/11/2016

Zadanie 2 - Mnożenie macierzy MPI

Celem naszego zadania było zaimplementowanie programu, którego zadaniem jest obliczenie iloczyn dwóch macierzy kwadratowych wypełnionych liczbami losowymi wg wzorów podanych w specyfikacji zadania oraz zastosowanie standardu MPI.

Program uruchamiany jest z argumentami: „size "- to liczba procesów (pobierany przy kompilacji za pomocą parametru -n) a „rozmiarMac "- to rozmiar macierzy. Pierwszym etapem jest sprawdzenie poprawności podanego rozmiaru macierzy. Podobnie jak w zadaniu 1, deklarujemy potrzebne zmienne oraz macierze, alokujemy pamięć dla macierzy oraz uzupełniamy je wartościami wg wskazanych wzorów. Na tym etapie rozpoczyna się nasza praca ze standardem MPI, który wymaga zainicjowania następujących funkcji, niezbędnych do dalszych obliczeń:

```
1  MPI_Init (&argc, &argv);                                    // inicjuje
       obliczenia MPI (pierwsza funkcja MPI)
2  MPI_Status status;                                          // okreslenie zrodla i
       typu komunikatu
3  MPI_Comm_rank (MPI_COMM_WORLD, &nr_procesu);                // komunikator,
       podanie numeru procesu
4  MPI_Comm_size (MPI_COMM_WORLD, &dostepne_procesy);          // komunikator,
       podanie liczby procesorow
```

Kolejnym krokiem, jest podzielenie macierzy w taki sposób aby każdy proces obliczał wskazany mu fargment, a nie zajmował się niepotrzebnie obliczaniem całej macierzy. Ma to na celu przyśpieszenie obliczeń rozwiązywanego zadania. Zadaniem poniższych funkcji jest wysłanie komunikatów (rozmiarWiersza i rozmiarStart) o długości równej 1 (typu LONG) od procesu docelowego 0 za pomocą komunikatora MPI-COMM-WORLD.

```
1  MPI_Send(&rozmiarWiersza, 1, MPI_LONG, id_zadania, 0, MPI_COMM_WORLD);
2  MPI_Send(&rozmiarStart, 1, MPI_LONG, id_zadania, 0, MPI_COMM_WORLD);
```

Następnie dokonujemy synchronizacji wszystkich procesów w grupie, za pomocą poniższej funkcji MPI-Barrier i rozpoczynamy pomiaru czasu obliczeń z pomocą MPI-Wtime().

```
1  MPI_Barrier(MPI_COMM_WORLD);
```

W następnym etapie używamy funkcji, odpowiedzialnych za odbiór i odczytanie komunikatów, które zawierają dane o tej samej długości i typie jak w funkcji Send, jednak w tym przypadkiem źródłem nadanego komunikatu jest proces o randze 0, który zawiera wszystkie procesy MPI-COMM-WORLD. Znacznik MPI-ANY-TAG informuje, że typ wiadomości nie będzie sprawdzany.

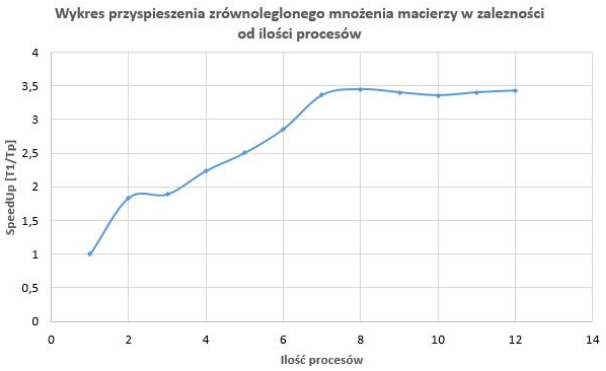
```
1  MPI_Recv(&rozmiarWiersza, 1, MPI_LONG, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &
       status);
2  MPI_Recv(&rozmiarStart, 1, MPI_LONG, 0, MPI_ANY_TAG, MPI_COMM_WORLD, &
       status);
```

Teraz musimy wykonać operację sumowania , gdzie wynik jest zwracany do jednego procesora(roota) za pomocą funkcji MPI-Reduce oraz ponowną synchronizację wszystkich procesów w grupie. Kończymy pomiar czasu obliczeń oraz implementujemy końcową instrukcję, jaką musi wykonać program korzystający z MPI - MPI Finalize() kończącą pracę programu.

Do celów ćwiczeniowych, nasze macierze przyjeliśmy o rozmiarze *1000x1000*. Wykres przyspieszenia zrównoleglonego mnożenia macierzy w zalezności od ilości procesów (Rysunek 1) dla mnożenia macierzy A i B, ukazuje, że wraz ze wzrostem liczby procesów czas obliczeń również się zwiększa. Przy pięciu procesach czas znacznie się zwiększa i drastycznie opada, by znów znacznie się zwiększyć pczy ośmiu procesach po których płynnie opada aż do osiągnięcia stabilnego poziomu.

Rysunek 1: Wykres zależności czasów od liczby procesów

Proces zrównoleglenia przy pomocy MPI również bardzo dobrze obrazuje poniższy wykres czasu zrównoleglonego mnożenia macierzy w zalezności od ilości procesów (Rysunek 2). Widzimy tutaj, że program zwalnia aż do osiągnięcia ośmiu wątków a następnie zrównolegla się.



Rysunek 2: Wykres przyspieszenia

Podsumowanie: MPI (ang. Message Passing Interface) to interfejs przesyłania komunikatów pozwalający na współdzielenie kodu przez wiele proocesów. Każdy z procesów zajmuje się przydzielonymi mu obliczeniami. MPI cechuje uniwersalność, ponieważ może być stosowany na wielu platformach, zrozumiałość gdyż równoległość i komunikacja są jawne. Jest również mobilny, pozwala na przenoszenie programu, a także wygodny, pozwala na zrównoleglenie programu na czym nam głównie zależało. W porównaniu do technologii OpenMP, która działa na wątkach, to MPI na początku jest trudniejsze w zrozumieniu i implementacji. W OpenMP nie musieliśmy zwracać uwagi jak dzielimy dane oraz co i jak przesyłamy między procesami. OpenMP wykonywał wszystko za nas, działając na wątkach.