Łukasz Oprych Gr. Lab. 5 Informatyka Techniczna	MPI – komunikacja grupowa	13.01.2024

Cel ćwiczenia:

Opanowanie programowania z przesyłaniem komunikatów MPI.

Przebieg ćwiczenia:

Po przygotowaniu struktury katalogowej i pobraniu z plików zgodnie z poleceniem prowadzącego, instalacji MPI, przystąpiono do napisania kodu obliczającego liczbę π na bazie pliku oblicz_PI.c uzupełniając o elementy niezbędne na równoległej wersji MPI.

Wczytanie liczby wyrazów do zmiennej max_liczba_wyrazow, której wartość jest przesyłana do procesów przy użyciu funkcji MPI_Bcast(), wyrazy będą dzielone pomiędzy procesami blokowo, podział zdefiniowano zmiennymi my_start, my_stride, my_end.

```
int main( int argc, char** argv ){
  int rank, size;
  MPI_Init( &argc, &argv );
  MPI_Comm_rank( MPI_COMM_WORLD, &rank );
  MPI_Comm_size( MPI_COMM_WORLD, &size );

  int max_liczba_wyrazow = 0;

  if (rank ==0) {
  printf("Podaj maksymalną liczbę wyrazów do obliczenia przybliżenia PI\n");
  scanf("%d", &max_liczba_wyrazow);
  }

  MPI_Bcast(&max_liczba_wyrazow, 1, MPI_INT, 0, MPI_COMM_WORLD);

  int my_start = rank * ceil(1.0 * max_liczba_wyrazow / size) ;
  int my_stride = 1;
  int my_end = (rank + 1) * ceil(1.0 * max_liczba_wyrazow / size) ;
  if (my_end > max_liczba_wyrazow) { my_end = max_liczba_wyrazow;}
```

Następnie dokonujemy obliczeń częściowych wartości π w każdym procesie.

```
SCALAR suma_plus=0.0;
SCALAR suma_minus=0.0;

for(int i = my_start; i < my_end; i+= my_stride){
  int j = 1 + 4*i;
  suma_plus += 1.0/j;
  suma_minus += 1.0/(j+2.0);
}
```

Następnie zliczamy cząstkowe wyniki, gdzie każdy proces wysyła do całościowej sumy pi_approx przy użyciu funkcji MPI_Allreduce bądź MPI_Reduce. W MPI_Reduce wynik zapisujemy w procesie wyznaczonym (w tym przypadku rank==0), w MPI_Allreduce każdy proces ma dostęp do wyniku.

```
SCALAR my_result = suma_plus - suma_minus;

SCALAR pi_approx;

//MPI_Reduce(&my_result, &pi_approx, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Allreduce(&my_result, &pi_approx, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
```

Na podstawie procesu 3 przedstawiono działanie programu.

```
if(rank == 3) {
  printf("Proces o randze: %d\n", rank);
  pi_approx *= 4;
  printf("PI obliczone: \t\t\t%20.15lf\n", pi_approx);
  printf("PI z biblioteki matematycznej: \t%20.15lf\n", M_PI);
  printf("Czas obliczeń: %lf\n", t);
}

MPI_Finalize();
```

Wynik kolejno dla 10, 100, 1000, 10000 wyrazów.

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_12/MPI_pi/MPI_PI$ make run
mpiexec -oversubscribe -np 6 ./MPI_PI
Podaj maksymalną liczbę wyrazów do obliczenia przybliżenia PI
10
Proces o randze: 3
PI obliczone:                   3.091623806667839
PI z biblioteki matematycznej:  3.141592653589793
Czas obliczeń: 0.000002
```

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_12/MPI_pi/MPI_PI$ make run
mpiexec -oversubscribe -np 6 ./MPI_PI
Podaj maksymalną liczbę wyrazów do obliczenia przybliżenia PI
100
Proces o randze: 3
PI obliczone:                   3.136592684838817
PI z biblioteki matematycznej:  3.141592653589793
Czas obliczeń: 0.000002
```

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_12/MPI_pi/MPI_PI$ make run
mpiexec -oversubscribe -np 6 ./MPI_PI
Podaj maksymalną liczbę wyrazów do obliczenia przybliżenia PI
1000
Proces o randze: 3
PI obliczone:                   3.141092653621039
PI z biblioteki matematycznej:  3.141592653589793
Czas obliczeń: 0.000003
```

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_12/MPI_pi/MPI_PI$ make run
mpiexec -oversubscribe -np 6 ./MPI_PI
Podaj maksymalną liczbę wyrazów do obliczenia przybliżenia PI
10000
Proces o randze: 3
PI obliczone:                   3.141542653589802
PI z biblioteki matematycznej:  3.141592653589793
Czas obliczeń: 0.000006
```

Zwiększenie ilości wyrazów zwiększa dokładność obliczeń, przy czym nie ma znaczących różnic w czasie wykonywania obliczeń.

Wnioski:

MPI_Reduce to jedna z funkcji komunikacji grupowej, pozwala na agregowanie wartości odbieranych na komunikatorze. Możliwe jest ustawienie operacji agregującej oraz procesu, który je wykonuje. W naszym przypadku została wykorzystana do zliczania sumy całki.

MPI_Allreduce różni się od MPI_Reduce w tym, że wyniki są dostępne we wszystkich procesach, a nie tylko w jednym (root). MPI_Allreduce wykonuje redukcję podobnie jak MPI_Reduce, ale wynik jest rozsyłany do wszystkich procesów w komunikatorze. W MPI_Reduce możliwe jest ustawienie dowolnej operacji redukującej, w MPI_Allreduce wykorzystywana jest operacja redukująca, ale wynikiem jest wartość przekształcona przez tę operację dostępną we wszystkich procesach.

MPI_Bcast ułatwia podział zadań pomiędzy procesami.

Proces to wykonujący się program wraz z dynamicznie przydzielanymi mu przez system zasobami (np. pamięcią operacyjną, zasobami plikowymi) oraz ewentualnie, innymi kontekstami wykonania programu (np. obiektami tworzonymi przez program).

Wątek to sekwencja działań, która może wykonywać się równolegle z innymi sekwencjami działań w kontekście danego procesu (programu), w jednym procesie może istnieć wiele wątków.