Wojciech Ługowski, Informatyka techniczna grupa 5

Sprawozdanie 5
Programowanie równoległe

1. Cel laboratorium

Celem laboratorium była nauka tworzenia i implementacji programów równoległych za pomocą biblioteki Pthreads przy użyciu mechanizmów dekompozycji.

2. Przebieg laboratorium

a. Program pthreads_suma.c

Program ten otrzymywał wynik sumowania na 3 sposoby – wersja sekwencyjna bez dzielenia zadania na wątki oraz dwie wersje równoległe jedna wykorzystująca mechanizm muteksów, a druga nie. Program zwracał wynik czasu obliczeń każdej z wersji – zostały zebrane wyniki dla rozmiarów tablicy 1000, 1000000, 100000000 oraz dla 2 i 4 wątków. Wyniki zostały zebrane na komputerze dostępnym w laboratorium z procesorem 2 rdzeniowym, 4 wątkowym.

Do obliczeń użyto dekompozycji blokowej dzielącą pętle na kilka mniejszych, które obliczane są za pomocą wątków. Pętla jest dzielona na mniejsze bloki zależne od rozmiaru wykonywanego zadania jak i liczby wątków programu. Początek wykonywania się pętli w zadaniu wątku zależny jest od jego id.

```
j=ceil( (float)ROZMIAR/LICZBA_W );
for( i = j*moj_id; i < j*(moj_id+1); i++){
  moja_suma += tab[i];
}
```

Przykład działania programu dla rozmiaru tablicy 10^8 oraz 2 wątków.

```
[wojtek@pc-59 zad1]$ ./pthreads_suma
Obliczenia sekwencyjne
suma = 50000000.500000
Czas obliczen sekwencyjnych = 0.102744
Poczatek tworzenia watkow
suma = 50000000.500000
Czas obliczen 2 wątków = 0.059105
Poczatek tworzenia watkow
suma = 50000000.500000
Czas obliczen 2 wątków (no mutex) = 0.053039
```

b. Program pthreads_calka.c

Program ten służy do obliczania całki z sin(x) na przedziale [0, π] za pomocą obliczeń sekwencyjnych oraz metodą zrównoleglenia pętli. Podczas obliczeń równoległych sprawdzane były wersje z dekompozycją cykliczną lub blokową. Do programu przekazywane były wysokość pojedynczego trapezu oraz ilość wątków na której mają być wykonywane obliczenia równoległe. Liczba trapezów zależna była od wysokości trójkąta. Ilość trapezów, długość kroku jak i przedziały były dostępne jako zmienne globalne. W programie tworzone były wątki, które jako argument dostawały swoje unikalne id, za pomocą którego wykonywany był odpowiedni

fragment pętli. Aby obliczyć początek, koniec i krok dla pętli w każdym wątku zostały użyte dwa rodzaje dekompozycji:

- Cykliczna – pętla wykonywana jest od elementu, którym jest id wątku i iterujemy ją o liczbę wątków. Pętla wykonuje się do ilości trapezów.

```
int my_start = my_id;
int my_end = N;
int my_stride = l_w_global;
```

- Blokowa – pętla dzielona jest na bloki, wielkość bloku obliczana jest jako ilość trapezów przez liczbę wątków, początkiem pętli jest id wątku razy wielkość bloku i wykonywana jest tyle razy jaka jest wielkość bloku. Długość kroku wynosi jeden.

```
int el_na_watek =  ceil( (float) N_global / l_w_global );
int my_start =  el_na_watek*my_id;
int my_end = el_na_watek*(my_id+1);
if(my_end>N) my_end = N;
int my_stride = 1;
```

Następnie wykonywane są obliczenia w pętli for w zadaniu wątku według wyznaczonych wcześniej wartości początkowych i końcowych pętli jak i długości kroku. Finalnie za pomocą muteksa otrzymany wynik zadania wątku dodawany jest do globalnej zmiennej wynikowej.

```
int i;
double calka = 0.0;
for(i=my_start; i<my_end; i+=my_stride){

  double x1 = a + i*dx;
  calka += 0.5*dx*(funkcja(x1)+funkcja(x1+dx));
}

pthread_mutex_lock( &muteks );
calka_global += calka;
pthread_mutex_unlock( &muteks );
pthread_exit( (void *)0);
```

Przykład działania programu dla wysokości trapezu 0,01 oraz dwóch wątków.

```
[wojtek@pc-59 zad2]$ ./pthreads_calka

Program obliczania całki z funkcji (sinus) metodą trapezów.

Podaj wysokość pojedynczego trapezu:  0.01

Podaj liczbę wątków:  2

Poczatek obliczeń sekwencyjnych
Obliczona liczba trapezów: N = 315

Koniec obliczen sekwencyjnych
        Czas wykonania 0.000053.        Obliczona całka = 1.999983422153753

Poczatek obliczeń równoległych (zrównoleglenie pętli)
Obliczona liczba trapezów: N = 315

Wątek 0
my_start 0, my_end 315, my_stride 2

Wątek 1
my_start 1, my_end 315, my_stride 2

Koniec obliczen  równoległych (zrównoleglenie pętli)
        Czas wykonania 0.000344.        Obliczona całka = 1.999983422153753
```

3. Wyniki

a. Program pthreads_suma.c

Czas wykonywania programu dla rozmiarów tablic 10^3, 10^6, 10^8 oraz dla 2 i 4 wątków. Wyniki sumy obliczeń równoległych zgadzały się z obliczeniami sekwencyjnymi.

Rozmiar tab/Wątki	1000	1000000	100000000
Sekwencyjne			
0		0.000900	0.812233
Z muteksem			
2	0.000165	0.000711	0.560668
4	0.000202	0.000490	0.538895
Bez muteksa			
2	0.000030	0.000519	0.540113
4	0.000043	0.000311	0.525235

b. Program pthreads_calka.c

Czas wykonywania programu dla wysokości trapezu 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-6} dla obliczeń sekwencyjnych, dekompozycji cyklicznej oraz dekompozycji blokowej dla 2 i 4 wątków. Wyniki całki obliczeń równoległych zgadzały się z obliczeniami sekwencyjnymi.

Wysokość/Wątki	0.1	0.01	0.001	0.0001	0.000001
Sekwencyjne					
	0.000033	0.000036	0.000122	0.001053	0.104642
Zrównoleglenie cykliczne					
2	0.000325	0.000334	0.000597	0.000854	0.058794
4	0.000372	0.000412	0.000636	0.001000	0.061628
Zrównoleglenie blokowe					
2	0.000380	0.000369	0.000417	0.000860	0.055717
4	0.000395	0.000414	0.000477	0.001123	0.060397

4. Wnioski

Dla programu pthreads_suma.c obliczenia sekwencyjne miały przewagę nad obliczeniami równoległymi tylko dla najmniejszego rozmiaru tablicy, co spowodowane mogło być czasem tworzenia wątku. Dla większych rozmiarów tablic obliczenia równoległe były szybsze od obliczeń sekwencyjnych.

Wyniki czasów dla większej ilości wątków powinny być mniejsze niż dla mniejszej ilości wątków jednak obliczenia dla 4 wątków wykonywały się dłużej niż obliczenia dla 2 wątków w każdym przypadku, co powtórzyło się następnie dla obliczeń programu pthreads_calka.c ,czyli w żadnym przypadku nie poprawiła się wydajność mimo korzystania z maksymalnej liczby wątków procesora – mogło to być spowodowane szybkością tworzenia większej liczby wątków bądź program nie mógł wykorzystać 100 procent wydajności komputera jak i jego zasobów.

Obliczenia dla wersji bez muteksa były znacznie szybsze od obliczeń z muteksem, przez brak konieczności czekania na zwolnienie muteksa przez inny wątek. Im większy był rozmiar tablicy tym mniejsza była różnica na korzyść programu bez muteksa.

Dla programu pthreads_calka.c obliczenia równoległe zyskały przewagę czasową dla mniejszej wysokości trapezu, czyli dla większej ilości trapezów – obliczenia sekwencyjne okazały się wolniejsze dla wysokości trapezu 10^{-4}. Obliczenia dla 4 wątków ponownie okazały się wolniejsze niż obliczenia dla 2 wątków. Wyniki dla dekompozycji cyklicznej jak i blokowej są do siebie bardzo zbliżone i można powiedzieć że są one w granicy błędu.