Przemysław Szymoniak

Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 4.
Celem laboratorium było nabycie umiejętności tworzenia i implementacji programów równoległych w środowisku Pthreads.

W ramach zajęć zrealizowałem następujące kroki:

- Pobranie i rozpakowanie pliku „**pthreads_suma.tgz**” i skopiowanie ich do wcześniej utworzonego folderu
- Tabela przedstawiająca sprawdzenie działania **„pthreads_suma”**:

2 WĄTKI			
Lp.	**Suma**	**Mutex**	**No Mutex**
1.	50000000.500000	0.049921	0.050531
2.	50000000.500000	0.049814	0.049855
3.	50000000.500000	0.051278	0.050588
4.	50000000.500000	0.050548	0.050114
5.	50000000.500000	0.051153	0.050787

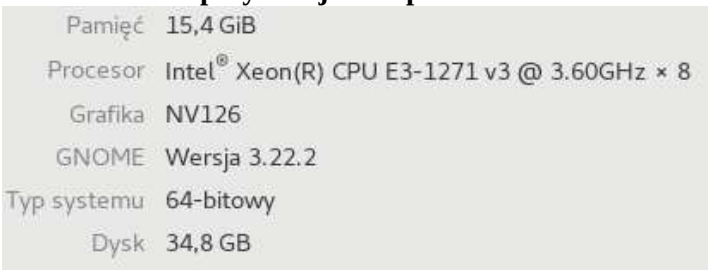
3 WĄTKI			
Lp	**Suma**	**Mutex**	**No Mutex**
1.	50000000.500000	0.049879	0.050256
2.	50000000.500000	0.049832	0.049763
3.	50000000.500000	0.049875	0.050380
4.	50000000.500000	0.051143	0.050307
5.	50000000.500000	0.050135	0.049577

4 WĄTKI			
Lp.	**Suma**	**Mutex**	**No Mutex**
1.	50000000.500000	0.049533	0.049376
2.	50000000.500000	0.049622	0.049364
3.	50000000.500000	0.049753	0.049434
4.	50000000.500000	0.049648	0.050035
5.	50000000.500000	0.049627	0.049595

5 WĄTKÓW			
Lp.	**Suma**	**Mutex**	**No Mutex**
1.	50000000.500000	0.049981	0.049935
2.	50000000.500000	0.050025	0.049922
3.	50000000.500000	0.050012	0.049831
4.	50000000.500000	0.050006	0.049898
5.	50000000.500000	0.050047	0.049882

6 WĄTKI			
Lp.	**Suma**	**Mutex**	**No Mutex**
1.	49999997.500000	0.050139	0.049986
2.	49999997.500000	0.050925	0.050582
3.	49999997.500000	0.050824	0.051080
4.	49999997.500000	0.050824	0.051080
5.	49999997.500000	0.051263	0.050929

Specyfikacja komputera:

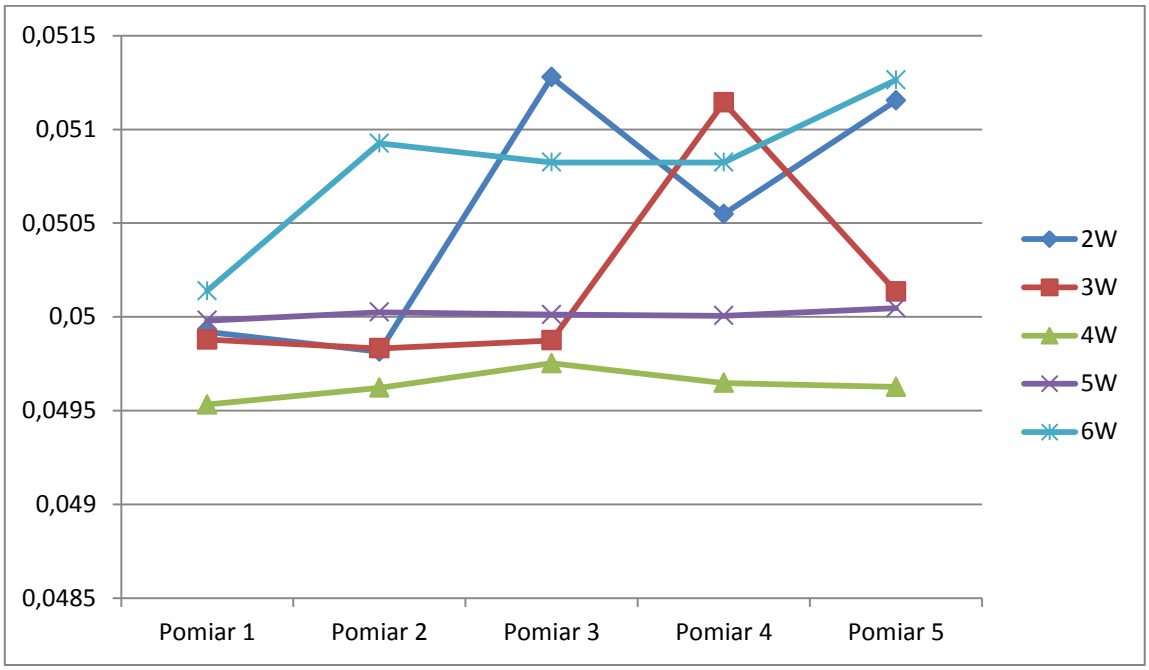


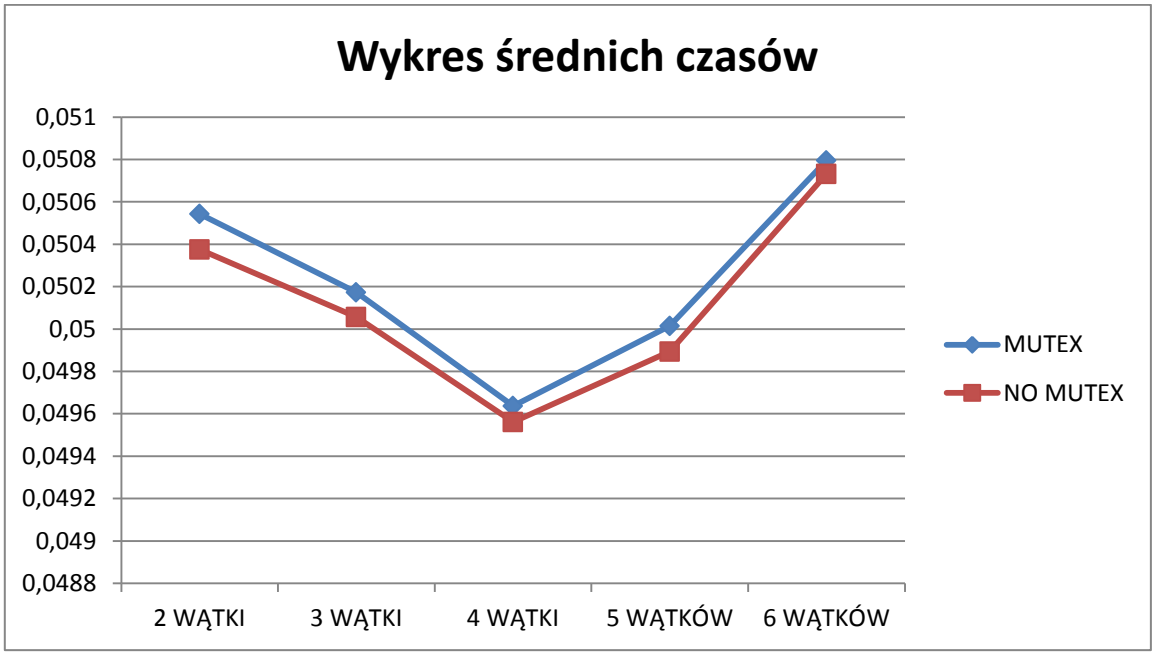
- Wykorzystanie przedstawionego algorytmu obliczania całki z sin(x) w przedziale od 0 do PI
- Tabela przedstawiająca dane dla różnych dokładności całkowania:

Lp.	N	Dx	Wynik
1.	10	0.31415926535	1.983524
2.	100	0.03141592654	1.999836
3.	1000	0.00314159265	1.999998
4.	10000	0.00031415927	1.999999
5.	100000	0.00003141593	2.000000

- Funkcja obliczająca całkę równolegle:

```
void *calka( void *arg_wsk){
        int i, j, moj_id, rekompensata=0;
        double dx = (3.14159265359-0)/N;
        moj_id = *( (int *) arg_wsk );
        double x2=3.14159265359, x1 = 0, c=0.0;
         j=N/LICZBA_W;
        for( i=j*moj_id+1; i<=j*(moj_id+1); i++){
                x2=x1+dx;
                c+=0.5*(f(x1)+f(x2))*dx;
                x1=x2;
        }
        pthread_mutex_lock( &muteks );
        suma += c;
        pthread_mutex_unlock( &muteks );
        pthread_exit( (void *)0);
}
```





Wnioski:

- Dla działania sumowania poprawy wynik dodawania otrzymujemy do momentu użycia 6ściu wątków współbieżnych. Przy ilości 6ściu wątków algorytm zaniża wynik o 3. To może być spowodowane błędami w zaokrągleniach niedokładności typu double
- Dla działania sumowania niższy średni czas dla każdego przypadku uzyskujemy bez użycia Mutexu jest to spowodowane tym, że Mutex blokuje inne wątki, które muszą czekać na zdjęcie blokady wydłużając przy tym całkowity czas trwania operacji.
- Najmniejszy średni czas dla operacji sumowania otrzymujemy przy użyciu czterech wątków bez zamków typu Mutex.
- Balansując zmienną N oraz Dx w trakcie obliczania całki z sin(x) jesteśmy w stanie uzyskać różną dokładność. Wraz ze wzrostem tym zmiennych uzyskujemy wzrost dokładności kosztem zwiększenia czasu wykonywania algorytmu, jest to naturalne ponieważ istotnie zwiększamy ilość przedziałów a zatem ilość przejść pętli.
- Przedstawiony przeze mnie kod równoległego obliczania całki działa prawidłowo działa tylko dla parzystych ilości wątków. Algorytm nie ma uwzględnionej rekompensaty związanej z niepodzielnością zmiennej N przez liczbę wątków.
- Porównując uzyskane wyniki do specyfikacji CPU(4R/8W) z łatwością zauważymy, że najlepszy wynik dla czterech wątków nie jest bez związku ponieważ przy takim układzie każdy rdzeń procesora wykonuje tylko po jednym wątku. Jeśli użyjemy większej liczby wątków mimo wykorzystywania HT to co najmniej jeden rdzeń musi wykonywać równocześnie 2 wątki za pomocą metody przełączania kontekstów co z oczywistego punktu widzenia zmniejsza efektywność w stosunku do użycia jednego wątku na rdzeń.