Grupa 4	Barbara Nowak, Piotr Tomaszewski	26/10/2016

Zadanie 1 - Macierze OMP

Celem naszego zadania było zaimplementowanie programu, którego zadaniem jest obliczenie iloczyn dwóch macierzy kwadratowych wypełnionych liczbami losowymi wg wzorów podanych w specyfikacji zadania. Do zrównoleglenia obliczeń wykorzystaliśmy bibliotekę OpenMP.

Program uruchamiany jest z dwoma argumentami: „ilWatkow"- to liczba wątków a „rozmiarMac "- to rozmiar macierzy. Pierwszym i bardzo ważnym etapem w naszym programie jest sprawdzenie poprawności wprowadzanych danych przed przystąpieniem do wykonywania obliczeń.

```
1  void liczenieMacierzy(double **AAA, double **BBB, double **CCC, int
       rozmiarMac){
2      int i,j,k;
3
4     #pragma omp parallel for default(none) shared(AAA, BBB, CCC)
          firstprivate(rozmiarMac) private(i, j, k)
5      for (i = 0; i < rozmiarMac; i++) {
6          for (j = 0; j < rozmiarMac; j++) {
7              for (k = 0; k < rozmiarMac; k++) {
8                  CCC[i][j] += AAA[i][k] * BBB[k][j]; //obliczanie iloczynu -
                       algorytm naiwny
9              }
10         }
11     }
12 }
```

Zadaniem powyższej funkcji jest pomnożenie elementów macierzy A i B a następnie zapisanie wyniku do macierzy C. Do przyspieszenia obliczeń użyto dyrektywy OpenMP w celu zrównoleglenia pętli. Dyrektywa ta składa się z parametrów: **parallel** - wskazanie kompilatorowi obszaru kodu, który będzie zrównoleglany; **omp** - słowo kluczowe odnoszące się do OpenMP; **for** - zrównoleglana będzie pętla for; **default(none)** - ustawienie domyślnego zasięgu zmiennych wewnątrz równolegle przetwarzanego obszaru, gdzie parametr ńoneńarzuca deklarowanie każdej zmiennej; **shared** - określenie wspólnych zmiennych (zmienne, do których kazdy wątek ma dostęp); **private** - określenie zmiennych prywatnych (zmienne, do których tylko jeden wątek ma dostęp); **firstprivate** - określenie zmiennych prywatnych już zainicjowanych.

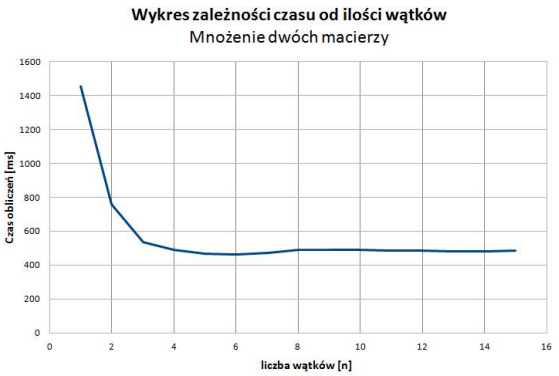
```
1 #pragma omp parallel for default(shared)
```

Powyższa dyrektywa różni się, od zastosowanej w kodzie programu, tym, że zmienne są współdzielone. Polega to na tym, że w procesie zrównoleglania zmienne będą musiały być ciągle synchronizowane pomiędzy wątkami. Oczywiście uzyskamy przyspieszenie w obliczeniach ale nie tak efektywne. Dyrektywa zastosowana w programie określa jasno, które zmienne ma traktować jako prywatne a które są współdzielone, przez co program nie musi tracić czas na synchronizację.

```
1 omp_set_num_threads(ilWatkow);
2
3 startTime = omp_get_wtime();
4 liczenieMacierzy(AAA, BBB, CCC, rozmiarMac);
5 stop = omp_get_wtime();
```

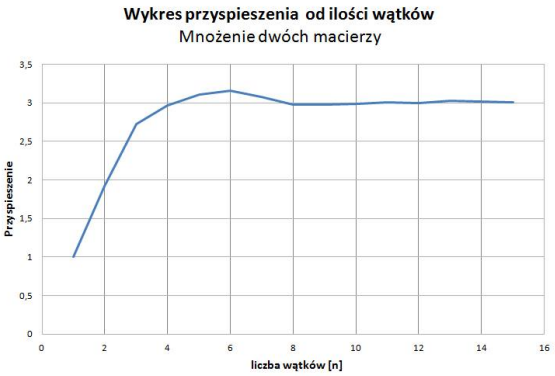
Powyższy fragment kodu jest odpowiedzialny za ustawienie liczby wątków dla zrównoleglnych obszarów, ustawienie momentu uruchomienia pomiaru czasu wykonywanego zadania, uruchomienie funkcji mnożącej dwie macierze oraz ustalenie zakończenia pomiaru czasu.

Do celów ćwiczeniowych, nasze macierze przyjeliśmy o rozmiarz *500x500*. Wykres zależności czasu od liczby wątków (Rysunek 1) dla mnożenia macierzy A i B, ukazuje, że wraz ze wzrostem liczby wątków czas obliczeń znacznie spada. Przy czterech wątkach czas łagodnie i płynnie opada aż do osiągnięcia stabilnego poziomu.



Rysunek 1: Wykres zależności czasu od liczby wątków

Proces zrównoleglenia przy pomocy biblioteki OpenMP również bardzo dobrze obrazuje poniższy wykres przyspieszenia (Rysunek 2). Widzimy tutaj, że program przyspiesza aż do osiągnięcia ośmiu wątków a następnie zrównolegla się. Serwer uruchomieniowy jest wyposarzony w 4-rdzeniowy procesor z technologią Hyper-Threading dlatego pozwalająca mu pracować na 8 wątkach.



Rysunek 2: Wykres przyspieszenia

Podsumowanie: OpenMP to rozszerzenie dla języków C/C++ pozwalające mieć wpływ na programowanie wielowątkowe. Każdy z wątków może posiadać własny tymczasowy widok na wspólny obszar pamięci. Zmienne zawarte w bloku parallel można podzielić na private - zmienne do których tylko jeden wątek ma dostęp i shared - dostęp wielu wątków do zmiennych. Używanie większej ilości wątków niż jest to sprzętowo możliwe to nie najlepsze rozwiązanie. Można pomyśleć, że przy nieskończonej ilości wątków czas będzie dążył do zera - to nie jest prawda. Po przekroczeniu możliwości procesora, czas wykonywania programu wzrośnie. Przy wykonywaniu obliczeń występowały zawahania czasowe - spowodowane było tym że kilka osób jednocześnie wykonywało pracę. Rozwiązaniem tego problemu było ponowne uruchomienie obliczeń.