Michał Szczygieł	Aleksander Śmierciak	24/03/2014

Liczba PI metodą Monte Carlo w OMP

Zadanie to polegało na zaimplementowaniu metody Monte Carlo, obliczającej przybliżoną wartośc liczby PI. Do zrównoleglenia obliczeń liczb pseudo losowych wykorzystany został interfejs OpenMP. Metoda Monte Carlo użyta w tym programie opiera się na generowaniu losowych liczb z zakresu [0;1)

```
1  unsigned int generatePoints(unsigned int threadCount, int pointCount)
2  {
3      unsigned int wheelHits = 0;
4      unsigned short seed[3];
5      double x;
6      double y;
7
8     #pragma omp parallel num_threads(threadCount)
9      {
10         wheelHits = 0;
11         initializeSeed(seed, threadCount);
12        #pragma omp for firstprivate(seed) private(x,y) reduction(+:wheelHits
               )
13         for (int i = 0; i < pointCount; i++)
14         {
15            x = erand48(seed);
16            y = erand48(seed);
17            if (std::pow(x, 2) + std::pow(y, 2) <= 1)
18            {
19               ++wheelHits;
20            }
21         }
22     }
23
24     return wheelHits;
25 }
```

Czy w programie mozna zastosowac funkcje rand do otrzymywania liczb losowych? W jaki sposob zmodyfikowac dyrektywe OpenMP, aby była wydajniejsza niż domyślna?

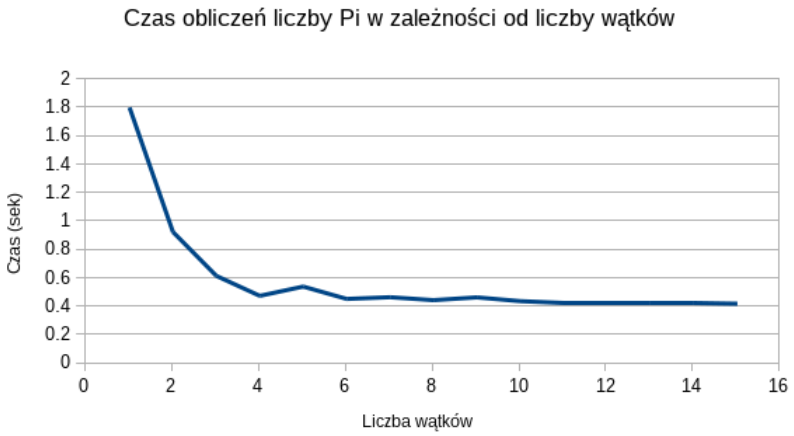
Nie powinno się stosować funkcji rand() przy użyciu OpenMP, ponieważ funkcja ta nie jest zrównoleglana. Instrukcja rand() modyfikuje wewnętrzny stan generatora liczb pseudolosowych, możemy mieć do czynienia ze zjawiskiem wyścigu przy jej stosowaniu. Co więcej, modyfikacja globalnych obiektów wiąże się z zaburzeniami przetwarzania równoległego, co odbija się na zwiększającym się wraz ze zwiekszaniem liczby wątków czasie wykonania. Rozwiązaniami są alternatywne (dostarczane przez konkretne biblioteki) instrukcje, które bezpieczeństwo w architekturze wielowątkowej gwarantują, takie jak erand48().
Dyrektywa OpenMP parallel for może zostać zastąpiona dyrektywą reduction.

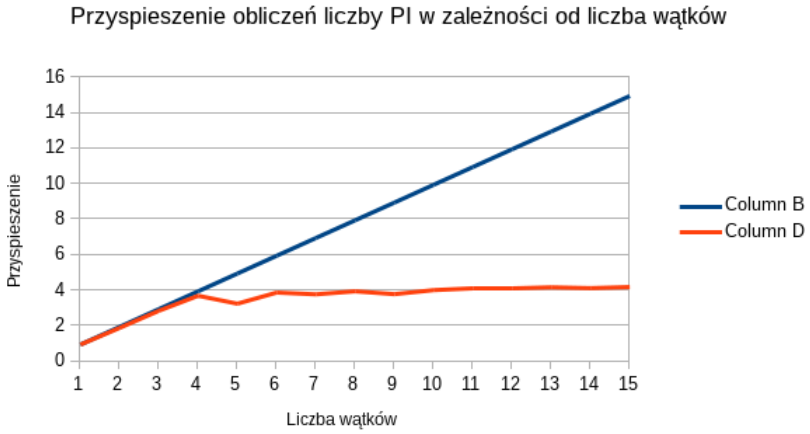
Przebieg

Obliczenia zostały wykonane na serwerze CUDA. Do zadania generowaliśmy 99999999 punktów. W każdym z przypadków uzyskano bardzo dobre przybliżenie liczby PI (z

dokładnością do 6 miejsc po przecinku).

Poniżej wykresy przedstawiający rezultat przeprowadzonych operacji na wątkach.





Wnioski

Przy użyciu OpenMP możemy zrównoleglić każdy fragment kodu uzyskując przy tym wysoką wydajność na maszynach wielordzeniowych. Pomiary wykonywane na serwerze CUDA ilustują nam że przyspieszenie osiągane jest do 4 wątków. Gdyż wraz ze wzrostem liczby wątków obserwujemy spadek czasu potrzebnego na wykonanie programu. Ten spadek nie jest jednak liniowy - dla pewnej liczby wątków funkcja przyspieszenia stabilizuje się. Jest to związane z osiągnięciem stanu, w którym tworzenie kolejnych wątków, ich synchronizacja i zarządzanie są operacjami o kosztach przewyższających zyski ze zrównoleglania wykonywanych obliczeń.