Raport z implementacji algorytmu "Gra w życie"

Radosław Głombiowski

26 kwietnia 2014

Spis treści

1 Założenia projektu

1.1 Mechanika algorytmu

Na planszy skonstruowanej z komórek zostają wyróżnione te komórki, które są żywe. Każda z komórek posiada ośmiu sąsiadów. Po wprowadzeniu przez użytkownika pozycji żywych komórek następują kolejne etapy życia tych struktur. Pozycje nowych komórek w nowej turze są obliczane według następujących reguł:

1. Jeśli martwa komórka posiada wokół siebie trzech żywych sąsiadów, komórka ożywa.
2. Jeśli żywa komórka posiada wokół siebie dwóch bądź trzech sąsiadów, komórka przeżywa.
3. Jeśli żywa komórka posiada wokół siebie inną liczbę sąsiadów, umiera (czy to z samotności, czy przeludnienia).

1.2 Cel projektu

- Stworzenie programu sekwencyjnego i równoległego działającego na Xeon Phi opartych na algorytmie Gry w życie. Następnie porównać czasy wykonania obu programów i wyciągnąć z tego wnioski.
- Nauczenie się efektywnego programowania równoległego i programowania na co-procesor Xeon Phi.

2 Porównanie wersji sekwencyjnej i równoległej

Testy zostały przeprowadzone na Sigmie oraz Xeon'ie Phi. Pomiary czasu zostały wykonane dla macierzy 1000x1000, 5000x5000, 10 000x10 000 oraz dla ilości kroków równej 1000.
Sekwencyjny:

	1000	5000	10000
Tworzenie danych	0.013487	0.325017	1.279214
Mechanika	28.756995	721.420322	Unicestwiony
Całość	28.770482	721.745339	Unicestwiony

Równoległy (8 rdzeni):

	1000	5000	10000
Tworzenie danych	0.017524	0.208643	0.735874
Mechanika	6.267167	133.991264	Unicestwiony
Całość	6.284691	134.199907	Unicestwiony

Xeon Phi:

	1000	5000	10000
Tworzenie danych	0.285410	0.726887	2.087383
Mechanika	2.670816	40.035989	173.218563
Całość	2.956226	40.762876	175.305946

Porównanie poszczególnych czasów (całość):

	1000	5000	10000
Sekwencyjny	28.770482	721.745339	Unicestwiony
Równoległy (Sigma - 8 rdzeni)	6.284691	134.199907	Unicestwiony
Xeon Phi	2.956226	40.762876	175.305946

Alokacja pamięci:

	1000	5000	10000
Sekwencyjny	0.013487	0.325017	1.279214
Równoległy (Sigma - 8 rdzeni)	0.056674	0.379938	0.841832
Xeon Phi	0.280319	3.816202	9.298919

3 Wnioski

Z porównań czasów jasno wynika, że tylko Xeon Phi poradził sobie z największym zadaniem jakim była macierz 10 000 x 10 000. Można z tego wywnioskować, że im większy problem stoi przed nami, tym bardziej będzie opłacalne zaopatrzenie się w ten sprzęt i jemu pokrewny. Nie można oczywiście zapominać, że samo posiadanie wielu rdzeni znacząco przyspiesza działanie programu. Jak można się przekonać działanie programu na sprzęcie wyposażonym w kilka rdzeni potrafi skrócić czas oczekiwania na wyniki nawet 5-cio krotnie. Z tego powodu każdy początkujący programista powinien uczyć się programowania równoległego już od samego początku. Wiedza ta będzie na pewno mile widziana w oczach pracodawcy.
Jednakże należy uważać w trakcie prac nad wielordzeniową wersją programu. Mimo iż różne elementy wykonywane w trakcie uruchomienia programu intuicyjnie można by wykonywać na wielu rdzeniach, nie jest to wskazane. Takim przypadkiem jest np. alokacja pamięci. Biorąc pod uwagę ludzki tok myślenia jest to jedno z miejsc, które nadają się do wykonywania na wielu wątkach. Jednakże, jak wskazują wyniki, takie działanie jest błędne. Dzieje się to z powodu tego, iż alokacja pamięci i tak musi być wykonana sekwencyjnie, a każdy procesor, który chce

zaalokować pamięć musi wiedzieć gdzie ją zaalokować co powoduje ”zamieszanie” i komunikacyjny problem. Z tego powodu dostajemy spore opóźnienie w stosunku do wersji sekwencyjnej. Należy jednak zwrócić uwagę na pewną anomalię jaką jest szybsza alokacja na komputerze 8-io rdzeniowym. Przyczyny tej anomialii jednak są nieznane.

4 Wiedza na przyszłość

- Znajomość OpenMP i wiedza o krytycznych miejscach uniemożliwiających zrównoleglenie.
- Wiedza jak pisać programy by zmaksymalizować użycie pamięci cache.
- Umiejętność obsługi sprzętu jakim jest Xeon Phi.