**SPRAWOZDANIE**

Przedmiot: Programowanie komputerów wielordzeniowych

Laboratorium: 4

Student: Artur Ziemba

Grupa: C

Celem sprawozdania było sprawdzenie czasu oraz sposobu działania programu generującego historie przebiegu gry w życie w postacji pliku .gif przy użyciu klauzur single oraz sections oraz przy różnych parametrach gry.

Wyniki zostały uzyskane na komputerze z systemem Windows 7 (64bit), 16GB RAM oraz procesorze Intel(R) Core(TM) i7-7700K *CPU* 4.20 GHz, 4 korowym, (8 procesorów logicznych). Do kompilacji programu wykorzystano kompilator g++. Każdy test został powtórzony 10 razy, a uzyskane wyniki czasowe prezentują wartość średnią.

1. Zbadanie czasu dla róznych rozmiarów mapy gry, ilości iteracji oraz wątków

**Przykładowo wygenerowany przebieg gry:**

|  |  |
| --- | --- |
| 1 iteracja | Ostatnia iteracja |
|  |  |

**Czasy obliczeń (sec):**

|  |  |  |  |
| --- | --- | --- | --- |
| Długość gry (iteracji):  Rozmiar mapy gry: | 10 | 200 | 1000 |
| 10 | 1 wątek: 0.00830002  4 wątki: 0.00379999  6 wątki: 0.00459998  8 wątków: 0.00510001 | 1 wątek: 0.0313  4 wątki: 0.0636999  6 wątki: 0.067  8 wątków: 0.0728 | 1 wątek: 0.1078  4 wątki: 0.2183  6 wątki: 0.295  8 wątków: 0.3582 |
| 100 | 1 wątek: 0.0167  4 wątki: 0.0135  6 wątki: 0.0128  8 wątków: 0.0129 | 1 wątek: 0.1842  4 wątki: 0.1876  6 wątki: 0.1417  8 wątków: 0.2333 | 1 wątek: 0.6654  4 wątki: 0.5174  6 wątki: 0.5214  8 wątków: 0.5127 |
| 500 | 1 wątek: 0.3515  4 wątki: 0.1345  6 wątki: 0.1299  8 wątków: 0.1271 | 1 wątek: 2.5427  4 wątki: 0.8468  6 wątki: 1.0983  8 wątków: 0.85 | 1 wątek: 8.2127  4 wątki: 3.2797  6 wątki: 3.7878  8 wątków: 3.3305 |

**Wnioski**:

Można zauważyć że poprzez zwiększenie ilości iteracji gry zmiejsza się przewaga rozwiązań wielwątkowych. Musimy poświęcić bowiem część czasu na tworzenie wątków podczas każdej iteracji. Jednoznacznie zwiększając rozmiar mapy możemy zauważyć, odpowiednio duże przyśpieszenie rozwiązań równoległych.

1. Single

Klauzura #pragma omp single musi zostać użyta w środku bloku współbieżnego. Kod wewnątrz bloku single zostanie wykonany tylko przez 1 wątek wykonujący współbieżny kod. Reszta wątków bedzie czekać na zakończenie obliczeń z bloku single lub jeśli została użyta klauzura nowait zacznie wykonywać kod bloku współbieżnego.

1. Sections

Derektywa Sections wskazuje, że dzielimy kod współbieżny na sekcie bloków, które będą równolegle wykonywane. Dyrektywa Section określa blok, który będzię przydzielony jednemu z wątków. Możemy również użyć dyrektyw typu shared/private w celu określenia dostępu do zmiennych.

Gra została zmodyfikowana, aby dwa wątki realizowały gre według innych zasad na jednej mapie. Wątki mogły zmieniać wartość komórki tylko jeśli komórka była martwa lub jeśli drugi wątek nie zmienił jeszcze wartości komórki. Jeden z wątków sprawdzał komórki od początku, drugi natomiast od końca. Aby zapewnić sekcje krytyczną między wątkami na każdą komórkę gry stworzyłem zamek biblioteki openmp (omp\_lock\_t) dla każdej komórki gry. Jeden z wątków korzysta z klasycznych reguł, zaś drugi podtrzymuje gre przy życiu – kiedy dookoła są tylko 2 żywe komórki przywraca komórke do życia, oraz ogranicza przed nadmiernym zaludnieniem – kiedy jest więcej niż 5 żywych sąsiadów komórka ginie.

Wyniki dla 200 iteracji, mapy o rozmiarze 100x100

Czas - 6 wątków: 7.2686 sec

Czas - 2 wątków: 9.9256 sec

|  |  |
| --- | --- |
| 1 iteracja | Ostatnia iteracja |
|  |  |

Wyniki dla 50 iteracji, mapy o rozmiarze 10x10

Czas - 6 wątków: 1.11051 sec

Czas - 2 wątków: 0.4132 sec

|  |  |
| --- | --- |
| 1 iteracja | Ostatnia iteracja |
|  |  |

**Wnioski**:

Jak można zauważyć synchronizacja wątków potrzebuje określonego czasu.W przypadku małych wartości użycie większej ilości wątków może nie przynieść oczekiwanego przyśpieszenia, a wręcz przeciwnie.