Sprawozdanie

Mikołaj Szymczak 136814

Specyfikacja

Do przeprowadzenia zadania użyto 4 rdzeniowego procesora Intel Core i7-6700HQ, systemu Windows 10 Pro oraz kompilatora Intel C++.

Tabela pomiarów

Każdy czas rzeczywisty został uzyskany poprzez średnią z 3 powtórzeń danego programu. Współczynniki przyspieszenia zostały obliczone w oparciu o wynik programu sekwencyjnego (PI 1), które wyniosło 1.695s.

		Rdzenie		
		8	4	2
PI 2	Czas [s]	0.554	0.604	0.944
	Przyspieszenie	3.063	2.809	1.796
PI 3	Czas [s]	148.945	101.905	66.853
	Przyspieszenie	0.011	0.017	0.025
PI 4	Czas [s]	0.506	0.565	0.913
	Przyspieszenie	3.354	3.003	1.857
PI 5	Czas [s]	0.499	0.555	0.914
	Przyspieszenie	3.396	3.054	1.856
PI 6	Czas [s]	0.937	1.447	1.999
	Przyspieszenie	1.811	1.172	0.848

Uzasadnienia

1. PI 2

Każdy otrzymany wynik obliczeń wyznaczenia liczby pi był błędny. Wszystkie wątki współdzielą zmienną sum oraz x, którą odczytują raz przed rozpoczęciem wykonywania pętli. Ostateczna otrzymana wartość liczby pi jest wartością, którą uzyskał wątek, który jako ostatni zakończył wykonywanie swojej części zadania. Wraz ze zwiększeniem liczby wątków można zauważyć zmniejszenie się wykonywania całego programu, gdyż całkowita liczba iteracji pętli for jest dzielona na większą liczbę wątków, które wykonują swoje iteracje równolegle.

2. PI 3

Tym razem otrzymana liczba pi jest prawidłowa dzięki niepodzielnej operacji ustalania zmiennej sum. Niestety kosztem zastosowania atomowej operacji ustalania zmiennej wynikowej jest zdecydowane zwiększenie się całkowitego czasu wykonywania się programu. Można zauważyć, że wraz ze wzrostem liczby wątków czas całego programu znacznie wzrasta. Dzieję się tak, ponieważ w momencie, gdy pewien wątek będzie wykonywał swoją atomową operację to reszta wątków jest wstrzymywana. Zwiększenie liczby wątków wydłuża ten proces, ponieważ kolejka oczekiwania na przeprowadzenie swojej operacji się wydłuża.

3. PI 4

Ta wersja programu dała prawidłowy wynik liczby pi. Ogólny czas wykonania programu był zdecydowanie lepszy od uzyskanego metodą sekwencyjną. Zastosowanie zmiennych prywatnych wykorzystywanych w pętli for pozwala na to, że każdy wątek oblicza swoją część danej pętli w swoim zakresie i następnie wyniki wszystkich wątków są atomowo sumowane do zmiennej współdzielonej. Wstrzymania wątków zostało zmniejszone do minimum przy sumowaniu ostatecznej sumy co pozwoliło uzyskać bardzo dobry czas.

4. PI 5

Sposób działania tego programu jest identyczny do PI 4 tzn. każdy wątek ustala lokalnie swoja sumę a po zakończeniu pętli lokalne sumy wątków są sumowane. Klauzula reduction tworzy prywatny obiekt, w tym przypadku sum, dla każdego wątku. Na końcu regionu for wartość oryginalnej zmiennej sum jest aktualizowana do sumy lokalnych zmiennych sum z oryginalną wartością współdzielonej zmiennej.

5. PI 6

W tej wersji programu została utworzona współdzielona tablica mieszcząca lokalne sumy dla każdego wątku. Każdy wątek używa funkcji omp_get_thread_num() zwracającej numer danego wątku i nadpisuje tylko wartość przydzieloną dla tego wątku w tablicy, a na koniec wszystkie wartości w tablicy są sumowane dając nam oczekiwany wynik.

6. PI 7

Sposób działania tej wersji programu jest oparty na wersji 6, dokonano kilku zmian, aby przystosować i usystematyzować proces 50 wywołań oraz zastosowałem współdzieloną zmienną x, aby na poniższym wykresie dało się bez problemu odczytać długość linii pp procesora. Eksperyment powiódł się i w oparciu o wykres licząc regularne odstępy pomiędzy spadkami czasów wykonań programu można zdecydowanie stwierdzić, że długość linii pamięci podręcznej procesora jest równa 8.