Projekt OpenMP – podstawy

Marek Subocz 136806

29.03.2020r.

1 Opis wykorzystywanego systemu równoległego

- **procesor** 2.6 GHz Intel Core i5 (2-rdzeniowy), z hyper-threadingiem na 4 procesory logiczne
- **system operacyjny** macOS Mojave (wersja 10.14.6)
- **kompilator** G++ 9.1.0

2 Rezultaty

Czas podawany jest w sekundach. Kolumna HALF była wykonywana na jednym wątku, PHYSICAL na dwóch, LOGIC na czterech.

wersja	LOGIC	PHYSICAL	HALF	przyspieszenie	uzasadnienie
P1	9.003374	9.003374	9.003374	1	wersja bez zrównoleglenia
P2	16.443293	22.365031	15.040152	0.9147	Używanie wszystkich zmiennych jako współdzielonych spowalnia obliczenia, w dodatku powodując przy tym błędy.
P3	107.719747	68.058967	20.612911	0.1914	Dyrektywa atomic jest bardzo powolna, a wywoływana jest za każdym dodaniem do sumy
P4	4.647594	5.664190	11.330755	2.4380	Wyraźne przyspieszenie spowodowane zmniejszeniem liczby użyć atomic
P5	4.652023	6.223397	11.725883	2.5206	Uzyskane wyniki są podobne do wersji P4, bez dodawania sum częściowych na koniec, ale jest ich znikoma ilość
P6	6.175133	9.992137	11.340937	1.8365	False sharing spowalnia obliczenia wykonywane wielowątkowo, ale nie zmienia szybości obliczeń w kolumnie HALF (patrz P4)

3 Odpowiedzi na pytania

3.1 [P2] Jakie mogą być przyczyny niepoprawnego wyniku?

Błędy w liczeniu pi wynikają z tego, że wszystkie zmienne są współdzielone, co powoduje m.in. odczyt innej wartości zmiennej sum niż tej, która zostaje nadpisywana.

3.2 [P3] W jaki sposób można na poziomie wątku zapewnić atomowość uaktualnienia zmiennej współdzielonej w systemie?

Aby zapewnić niepodzielność na poziomie wątku możemy użyć dyrektywy omp critical.

4 Eksperyment

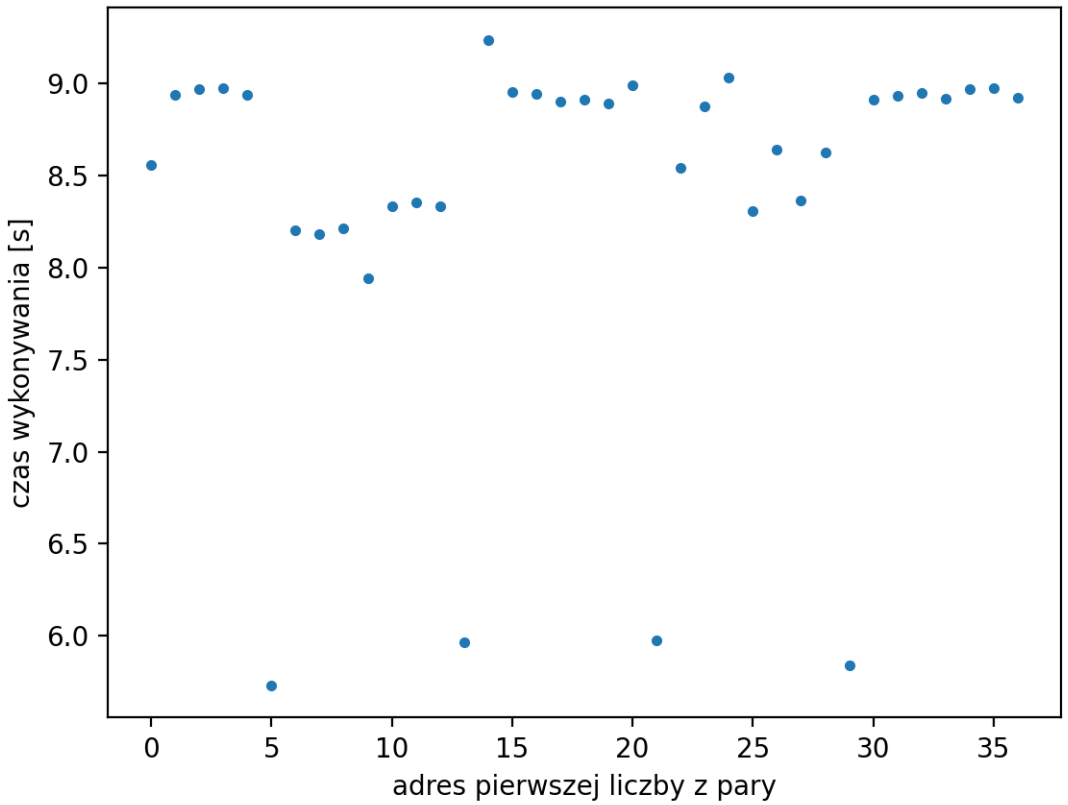
4.1 Opis eksperymentu

Eksperyment polegał na doświadczalnym wyznaczeniu długości linii pamięci swojego komputera za pomocą zjawiska **false sharing**. Zjawisko to polega na tym, że rdzenie procesora wykorzystujące tę samą linię pamięci spowalniają się nawzajem, ponieważ linia pamięci jest najmniejszą możliwą częścią pamięci do której wątek może zablokować dostęp.

W przeprowadzonym eksperymencie użyto tablicy zmiennych typu double (8-mio bitowych), co pozwoliło na zajęcie ciągłej sekcji pamięci. Dzięki temu, przesuwając parę zmiennych używanych do obliczeń przez dwa rdzenie komputera po 8 bitów, można było wyznaczyć interwał, z jakim zmienne występują w osobnych liniach pamięci, a w konsekwencji wyznaczyć jej długość.

4.2 Wynik

Wyniki eksperymentu widać na poniższym wykresie:



Jak można łatwo zauważyć, co 8 prób powtarza się wyjątkowo krótki czas wykonywania zadania. Trend ten powtarzał się dalej, ale dla jasności na wykresie umieściłem tylko pierwsze 37 wyliczeń liczby pi.

Oznacza to, że linia pamięci mojego komputera jest długości 8 * 8 bajtów, to jest 64 bajty. Jest to zgodne z architekturą mojego komputera.

4.3 Trudności napotkane w zadaniu

Największą trudnością okazał się fakt, że prędkości wykonania obliczeń zaczęły różnić się dopiero wtedy, kiedy dodana została dyrektywa schedule(guided). Przy domyślnej wartości schedule te różnice nie wystepowały w ogóle.