Programowanie równoległe i rozproszone

Laboratoria nr 1

Jakub Niewadzi

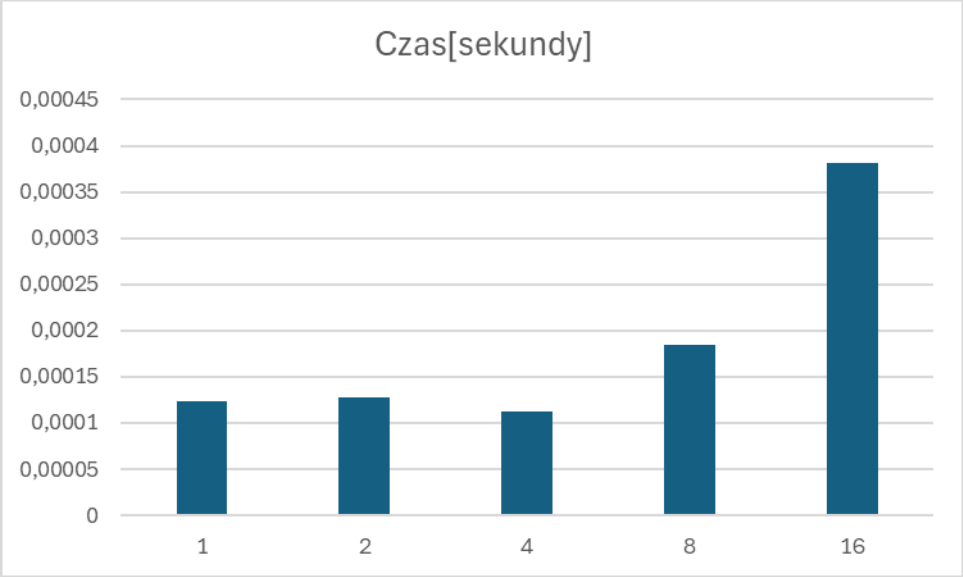
Kacper Sosnowski

1. Liczenie sumy wartości wektora składającego się z osiemdziesięciu elementów.

Wynik na standardowe wyjście

```
~/lab1$ ./a.out
czas startu: 0.000000
czas końca: 0.000000
Czas potrzebny na policzenie wektora przez 1 proceów to: 0.000123 sekund

Suma to: 6400.000000
czas startu: 0.000000
czas końca: 0.000000
Czas potrzebny na policzenie wektora przez 2 proceów to: 0.000128 sekund

Suma to: 6400.000000
czas startu: 0.000000
czas końca: 0.000000
Czas potrzebny na policzenie wektora przez 4 proceów to: 0.000112 sekund

Suma to: 6400.000000
czas startu: 0.000000
czas końca: 0.000000
Czas potrzebny na policzenie wektora przez 8 proceów to: 0.000185 sekund

Suma to: 6400.000000
czas startu: 0.000000
czas końca: 0.000000
Czas potrzebny na policzenie wektora przez 16 proceów to: 0.000381 sekund

Suma to: 6400.000000
```

Czas został policzony dla sumowania, które zostało powtórzone 100000 razy, tak aby sztucznie wydłużyć obliczenia. Najszybciej z obliczeniami poradziła sobie iteracja, która wykorzystała 4 procesy do obliczenia, natomiast najwolniej iteracja wykorzysująca 16 procesów



2. Wnioski

Na podstawie przedstawionego wyżej wykresu, można zauważyć, że czas wykonania zadania sumowania wszystkich elementów wektora przy zwiększaniu liczby procesów, pozostaje mniej więcej stały. Dopiero przy zwiększeniu liczby procesów do 16, widać znaczący wzrost czasu wykonania programu.

Może to wynikać z tego, że przy większej ilości procesów może wystąpić narzut związany z komunikacją między nimi. Gdy liczba procesów przekracza pewną granicę, wzrasta ilość komunikacji i synchronizacji potrzebnych do koordynacji pracy procesów. To może prowadzić do opóźnień i zwiększenia czasu wykonania zadania. Podczas nadmiernego zwiększania liczby procesów mijamy minimum czasu, w którym można wykonać zadanie.