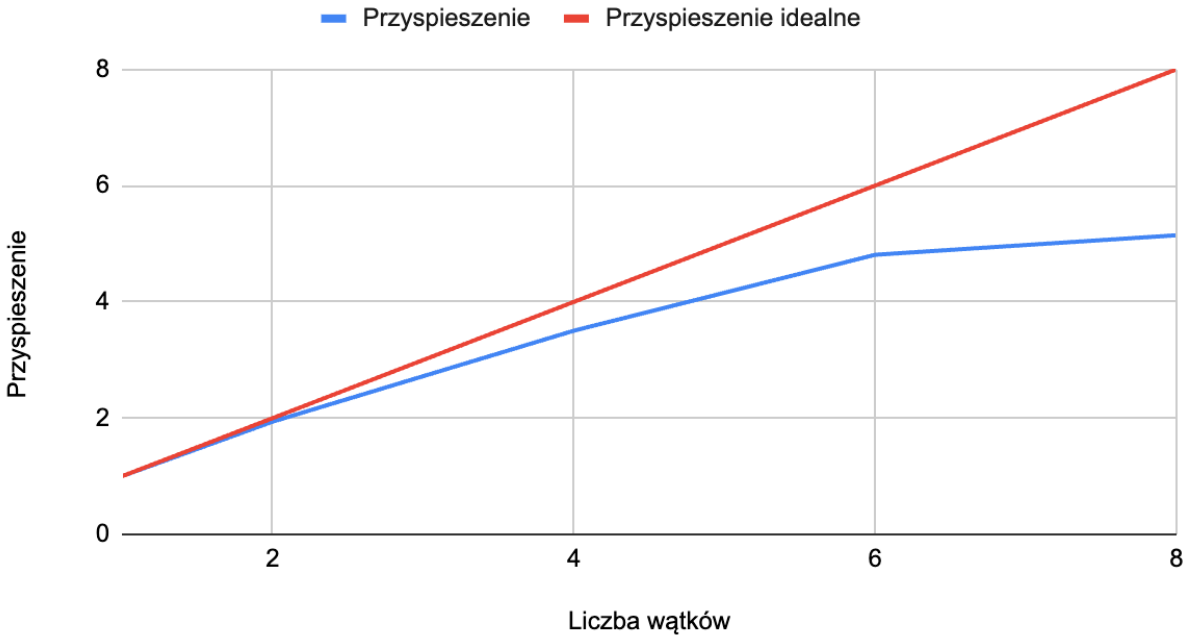
Sprawozdanie

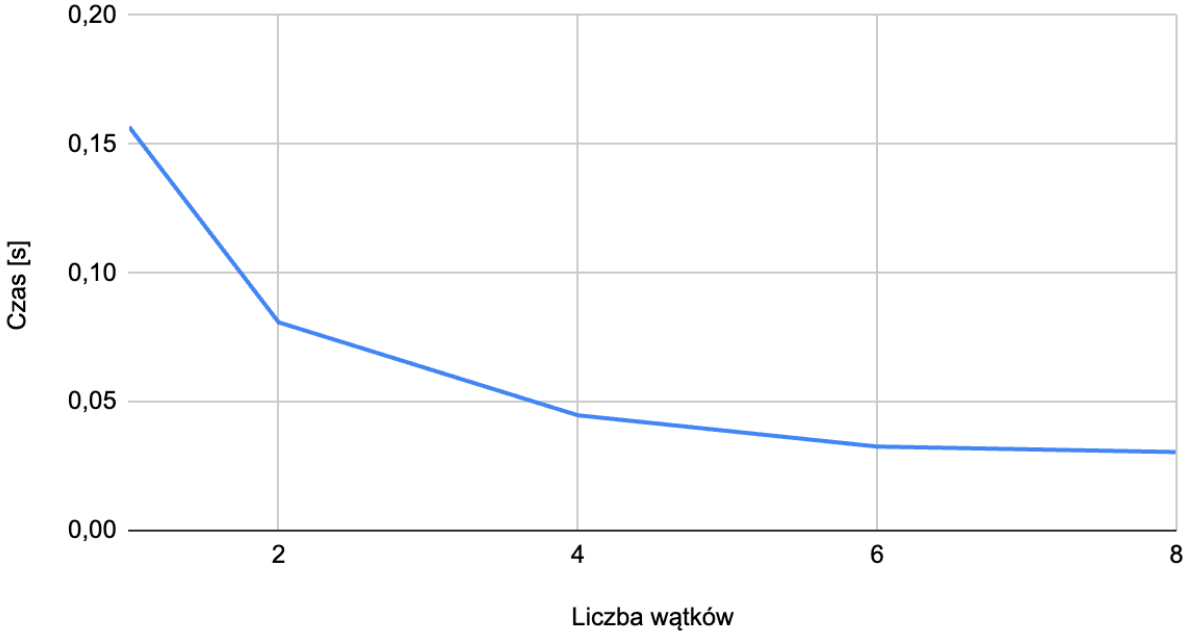
Wyniki dla calka_omp.c

Liczba wątków	Czas 1 [s]	Czas 2 [s]	Czas 3 [s]	Średnia [s]	Przyspieszenie	Przyspieszenie idealne
1	0,154065	0,158409	0,157794	0,156756	1	1
2	0,082208	0,078686	0,081601	0,08083166667	1,93928947	2
4	0,044892	0,045324	0,043987	0,04473433333	3,504154155	4
6	0,031389	0,032548	0,033835	0,03259066667	4,809843309	6
8	0,031922	0,028775	0,030699	0,03046533333	5,145389295	8

Przyspieszenie a przyspieszenie idealne



Czas wykonania w zależności od wątków



1. Ocena skalowalności w sensie silnym

Uzyskana skalowalność jest **dobra, ale nie idealna**.

- **Uzasadnienie**: Wykres "Czas wykonania w zależności od wątków" pokazuje wyraźny spadek czasu wraz ze wzrostem liczby wątków, co jest pożądanym efektem. Jednak wykres "Przyspieszenie a przyspieszenie idealne" pokazuje, że rzeczywista linia przyspieszenia (niebieska) coraz bardziej odbiega od idealnej (czerwonej) wraz ze wzrostem liczby wątków. Dla 8 wątków przyspieszenie wynosi około 5.7, a nie idealne 8, co świadczy o rosnącym wpływie narzutów.

2. Potencjalne źródła narzutu w kodzie

Mimo że problem obliczania całki jest niemal idealnie równoległy, istnieją niewielkie źródła narzutu spowalniające obliczenia:

- **Zarządzanie wątkami**: Sam fakt stworzenia, uruchomienia i zakończenia pracy przez wątki OpenMP generuje pewien koszt, który jest niezależny od właściwych obliczeń.
- **Operacja `reduction`**: Na końcu pętli `for` wszystkie prywatne kopie zmiennej `calka` muszą zostać zsumowane w jedną globalną wartość. Chociaż operacja redukcji jest wysoce zoptymalizowana, stanowi ona punkt synchronizacji i zajmuje pewien czas.
- **Minimalne niezrównoważenie obciążenia**: W teorii, jeśli liczba iteracji pętli (`N`) nie jest idealnie podzielna przez liczbę wątków, niektóre wątki mogą wykonać o jedną iterację więcej. Przy `N = 100000000` ten efekt jest jednak pomijalnie mały.

3. Wyczerpywanie zasobów komputera

Nie, program w testowanych konfiguracjach **nie wyczerpywał zasobów komputera** w sposób, który uniemożliwiłby dalszy wzrost wydajności.

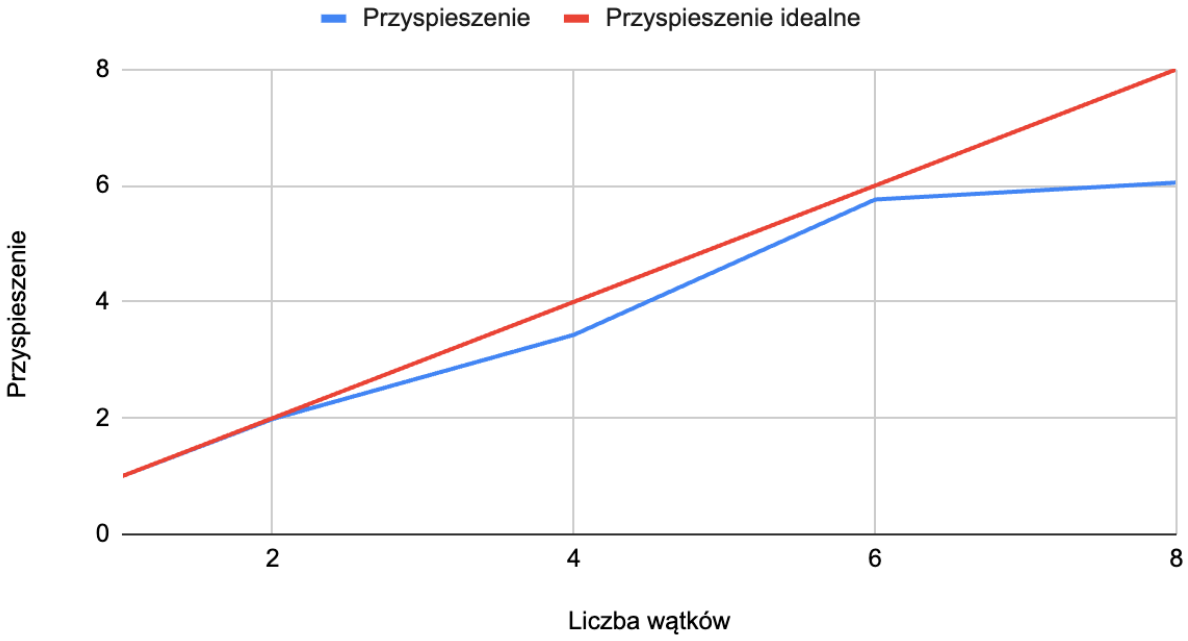
- **Uzasadnienie**: Ten program jest **ograniczony obliczeniowo (compute-bound)**, co oznacza, że jego wydajność zależy głównie od mocy obliczeniowej CPU, a nie od przepustowości pamięci. Krzywa przyspieszenia wciąż rośnie dla 8 wątków, choć wolniej. To sugeruje, że dodanie większej liczby rdzeni wciąż przyniosłoby korzyści. Nie widać tu "ściany", która świadczyłaby o wyczerpaniu np. przepustowości pamięci.

Wyniki dla mat_vec_row_MPI.c

Liczba wątków	Czas 1 [s]	Czas 2 [s]	Czas 3 [s]	Średnia [s]	Przyspieszenie	Przyspieszenie idealne
1	0,628572	0,626842	0,627462	0,6276253333	1	1
2	0,318681	0,316178	0,313882	0,316247	1,984604861	2
4	0,19477	0,157533	0,196239	0,1828473333	3,432510182	4
6	0,111922	0,117031	0,097901	0,1089513333	5,760602593	6
8	0,123365	0,073721	0,114076	0,1037206667	6,05111164	8

Przyspieszenie a liczba wątków

Przyspieszenie a przyspieszenie idealne



1. Ocena skalowalności w sensie silnym

Uzyskana skalowalność jest **umiarkowana i zauważalnie gorsza** niż w pierwszym programie.

- **Uzasadnienie**: Początkowo, dla 2 i 4 procesów, przyspieszenie jest dobre. Jednak przejście z 4 na 8 procesów przynosi już znacznie mniejszy zysk – czas wykonania spada nieznacznie, a krzywa przyspieszenia wyraźnie się spłaszcza. Różnica między przyspieszeniem rzeczywistym a idealnym jest tu znacznie większa niż w przypadku programu `calka_omp`.

2. Potencjalne źródła narzutu w kodzie

Główną przyczyną ograniczonej skalowalności jest **narzut komunikacyjny** między procesami MPI, który w tym algorytmie jest znaczący.

- **Komunikacja `MPI_Bcast` i `MPI_Scatter`**: Przed rozpoczęciem obliczeń, proces główny musi rozesłać fragmenty macierzy `A` do wszystkich procesów (`MPI_Scatter`) oraz rozgłosić cały wektor `x` (`MPI_Bcast`).
- **Komunikacja `MPI_Gather`**: Po zakończeniu obliczeń, wszystkie procesy muszą odesłać swoje wyniki do procesu głównego, co wymaga synchronizacji i transferu danych.
- **Prawo Amdahla w praktyce**: Wraz ze wzrostem liczby procesów, porcja obliczeń przypadająca na jeden proces maleje. W rezultacie stały (a nawet rosnący) koszt komunikacji stanowi coraz większy procent całkowitego czasu wykonania, ograniczając dalsze przyspieszenie.

3. Wyczerpywanie zasobów komputera

Jest bardzo prawdopodobne, że program **zbliżał się do wyczerpania zasobów komunikacyjnych systemu (przepustowości sieci/pamięci)**.

- **Uzasadnienie**: W przeciwieństwie do pierwszego programu, ten jest **ograniczony komunikacyjnie (communication-bound)**. Wyraźne spowolnienie skalowalności między 4 a 8 procesami silnie sugeruje, że to właśnie komunikacja staje się "wąskim gardłem". Dalsze zwiększanie liczby procesów mogłoby przynieść minimalny zysk lub nawet spowolnić program, ponieważ więcej czasu byłoby poświęcane na przesyłanie danych niż na obliczenia.