

Algorytmy równoległe

Równoległe rozwiązywanie iteracyjne równania Laplace’a przy pomocy MPI

Oscar Teeninga

Wyniki czasowe

S	P	Czas [s]	P	E
56	1	0,557267	1,00	1,00
56	2	0,282835	1,97	0,99
56	3	0,184304	3,02	1,01
56	4	0,144745	3,85	0,96
56	5	0,115554	4,82	0,96
56	6	0,099753	5,59	0,93
56	7	0,095302	5,85	0,84
56	8	0,085508	6,52	0,81
112	1	2,206515	1,00	1,00
112	2	1,145891	1,93	0,96
112	3	0,806341	2,74	0,91
112	4	0,600359	3,68	0,92
112	5	0,511358	4,32	0,86
112	6	0,442882	4,98	0,83
112	7	0,430616	5,12	0,73
112	8	0,383425	5,75	0,72
168	1	4,91636	1,00	1,00
168	2	2,49422	1,97	0,99
168	3	1,88061	2,61	0,87
168	4	1,46345	3,36	0,84
168	5	1,12505	4,37	0,87
168	6	1,03001	4,77	0,80
168	7	0,91085	5,40	0,77
168	8	0,85330	5,76	0,72
224	1	9,07588	1,00	1,00
224	2	4,45207	2,04	1,02
224	3	2,95591	3,07	1,02
224	4	2,41401	3,76	0,94
224	5	2,06465	4,40	0,88
224	6	1,83650	4,94	0,82
224	7	1,75302	5,18	0,74
224	8	1,70635	5,32	0,66
280	1	13,98542	1,00	1,00
280	2	7,02880	1,99	0,99
280	3	4,97012	2,81	0,94
280	4	4,08135	3,43	0,86
280	5	3,56007	3,93	0,79
280	6	3,29856	4,24	0,71
280	7	3,09588	4,52	0,65
280	8	2,91356	4,80	0,60

Implementacja

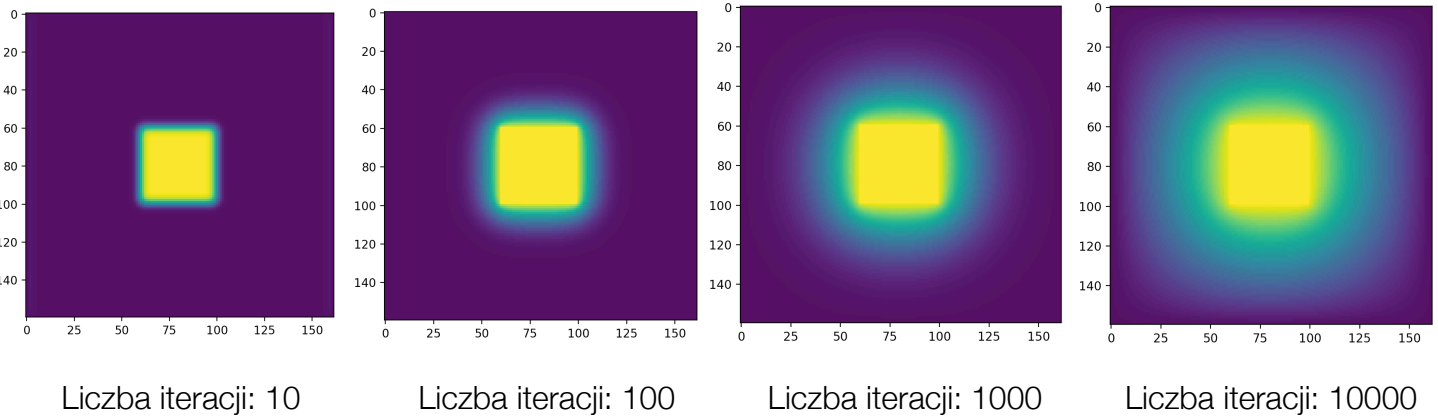
Pomimo wcześniejszej chęci wykonania obliczeń na GPU, zdecydowałem się na wykorzystanie CPU oraz biblioteki MPI. Implementacja w Python 3.9 przy pomocy mpi4py.

Algorytm

1. Podziel zadanie (macierz NxN) na P równych części, gdzie N - bok kwadratu, P - liczba procesorów.
 2. Każdy procesor jest odpowiedzialny za swoją część.
 3. Każdy procesor wysyła swoją część do sąsiadów.
 4. Każdy procesor po skompletowaniu wszystkich potrzebnych danych, wykonuje obliczenie.
 5. Powtarzaj 2-5 dopóki liczba iteracji nie osiągnie I - liczby iteracji.
 6. Połącz części w jedną macierz (na pierwszym procesorze)
- Newralgicznym miejscem jest wysyłka swoich części do sąsiadów. Pierwszy i ostatni procesor wysyła tylko jednemu procesorowi swoją część, natomiast środkowe do obu sąsiadów. Obliczenia są wykonywane na kopii części, a po obliczeniach wartości są przepisywane do bazowej macierzy. Nie ma potrzeby tworzenia bariery pod koniec iteracji, ponieważ jeżeli jakiś procesor wykona obliczenia szybciej - i tak będzie musiał oczekiwać na otrzymanie części od sąsiada.

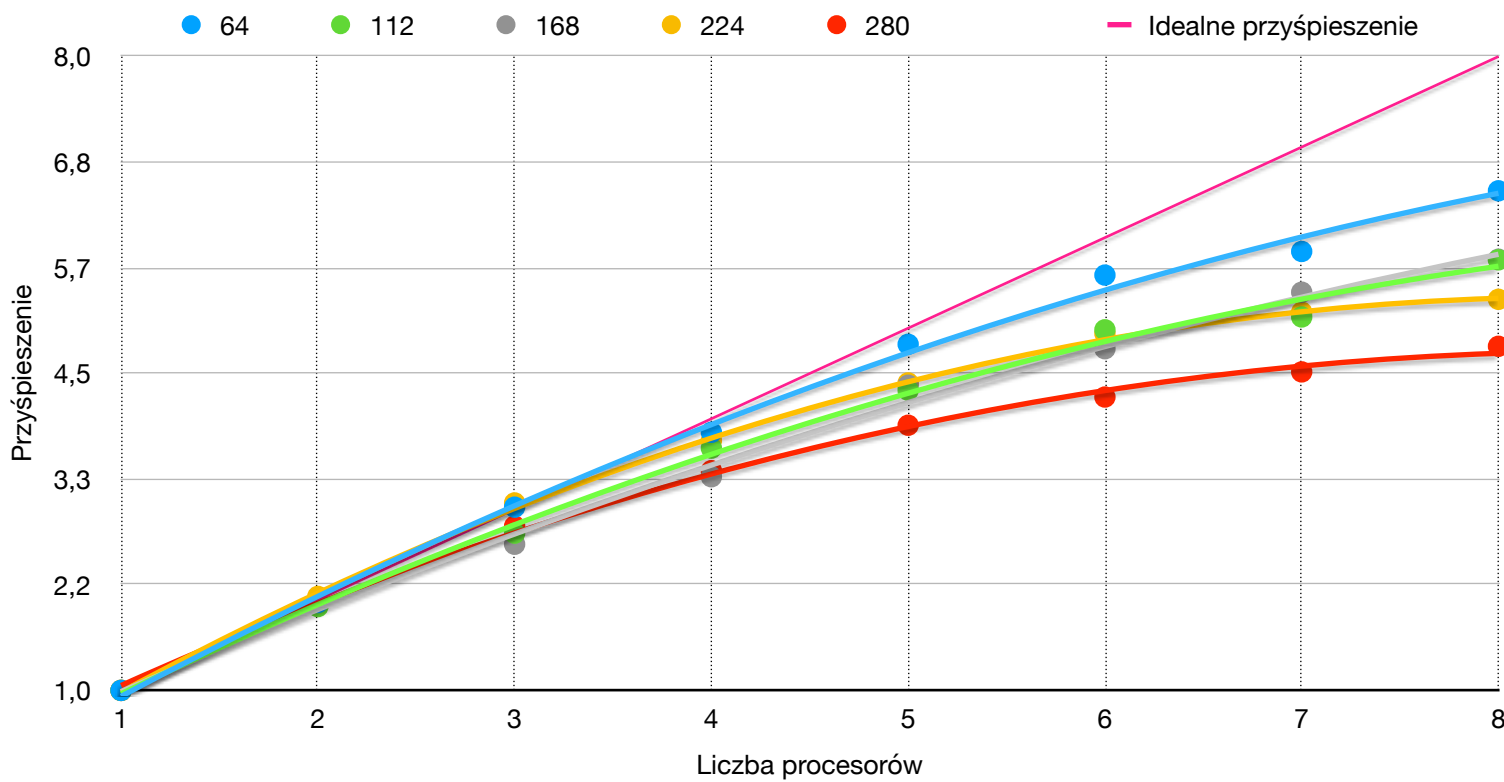
Sprzęt testowy, metodologia, parametry

Do testów posłużył mi prywatny komputer wyposażony w 8 rdzeniowy procesor i9 9880H. Wynik czasowy jest średnią arytmetyczną z 10 pomiarów. Każdy pomiar ma 100 iteracji algorytmu. Przeprowadziłem testowe pomiary dla różnej liczby iteracji. Wynik jest następujący:

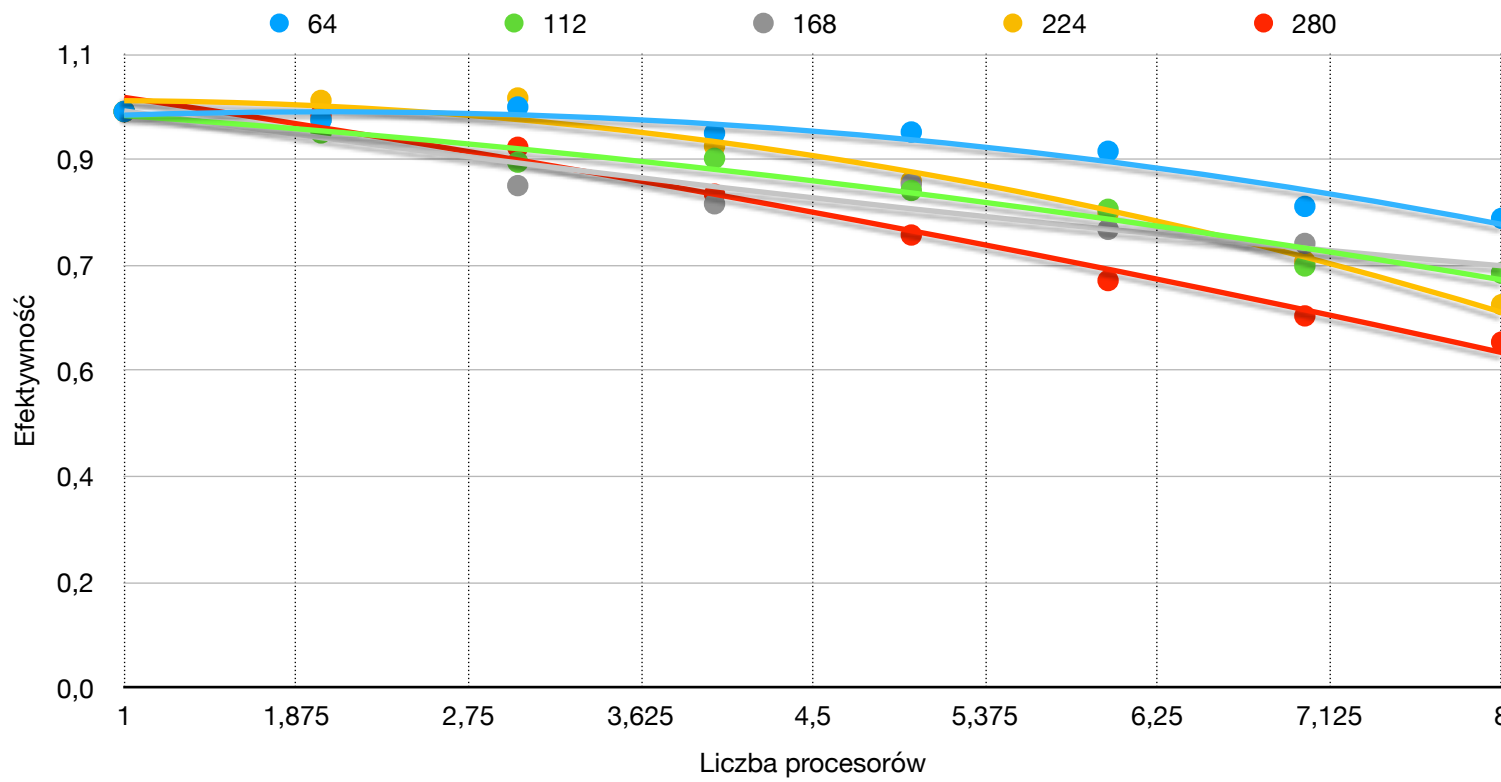


Wykresy przyśpieszenia i efektywności

Na podstawie wyników obok wygenerowałem wykresy przedstawiające przyśpieszenie i efektywność w zależności od liczby procesorów dla różnego rozmiaru problemów.



Wykres przyśpieszenia



Wykres efektywności

Model teoretyczny efektywności

Zauważmy, że algorytm ma złożoność $O(n^2)$, gdzie n to rozmiar boku kwadratu, ponieważ obliczenie każdego punktu jest $O(1)$. Każdy procesor obsługuje $O(n^2/p)$ operacji. Każdy procesor wysyła oraz otrzymuje w przybliżeniu dwie tablice o rozmiarze n^2/p .

t_c - czas obliczeń
 t_s - czas inicjalizacji komunikacji
 t_w - czas przesyłania danych
 $T(p)$ - czas obliczeń dla p procesorów
 $T_{comm}(p)$ - czas komunikacji dla p procesorów

$$T(1) = t_c * n^2$$
$$T_{comm}(p) = p * (t_s + \frac{4t_w n^2}{p}) = t_s p + 4t_w n^2$$
$$T(p) = \frac{T(1) + T_{comm}(p)}{p} = \frac{(t_c + 4t_w)n^2}{p} + t_s$$
$$E(p) = \frac{T_{comp}(1)}{pT(p)} = \frac{t_c n^2}{(t_c + 4t_w)n^2 + t_s p}$$

Wnioski

Zgodnie z oczekiwaniami, otrzymana efektywność jest mniejsza niż idealna. Jest to spowodowane narzutem komunikacji. Widać również, że skalowalność rozwiązania jest poniżej oczekiwań. Wynika to z tego, że czas wysłania wiadomości jest zależne kwadratowo od rozmiaru boku kwadratu, co sprawia, że dla narzutu związany z komunikacją rośnie kwadratowo względem liczby procesorów. Warto wspomnieć o pewnym interesującym zjawisku, mianowicie najlepsze przyśpieszenie osiągnąłem dla najmniejszego problemu. Jest to spowodowane wcześniej wspomnianym narzutem komunikacyjnym. Dodatkowo podejrzewam, że może to być kwestia throttlingu. Pierwszy test jest uruchamiany na chłodnym procesorze, co chwilowo pozwala mu pracować w pełnym zakresie taktowania. Nie bez znaczenia jest fakt, że każda iteracja wymaga synchronizacji między wątkami. Oznacza to, że wraz ze wzrostem problemu, rośnie również prawdopodobieństwo anomalii i np. jeden wątek zgłodzi resztę. Optymalne rozwiązanie powinno bazować na efektywniejszej komunikacji.