Algorytmy równoległe

Rozwiązywanie układów równań liniowych

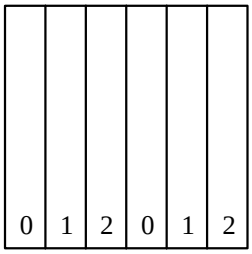
Marcin Rociek
Informatyka, IV rok

21 grudnia 2001

Równoległa implementacja metody eliminacji Gaussa z częściowym pivotingiem w środowisku MPI.

1 Opis algorytmu

Macierz jest podzielona na poszczególne procesy w ten sposób, że kolejne kolumny znajdują się w kolejnych procesach (rys. 1).

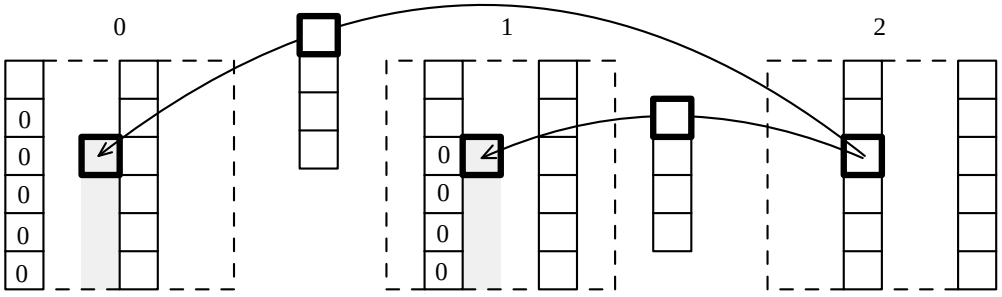


Rysunek 1: Rozkład kolumn macierzy na procesy - każda kolejna kolumna znajduje się w kolejnym procesie.

W każdym kroku inny proces zajmuje się kolumną wiodącą: wyszukuje element wiodący i zamienia odpowiednie wiersze (informuje również pozostałe procesy o numerze zamienianego wiersza), następnie rozsyła odpowiedni fragment kolumny do pozostałych procesów.

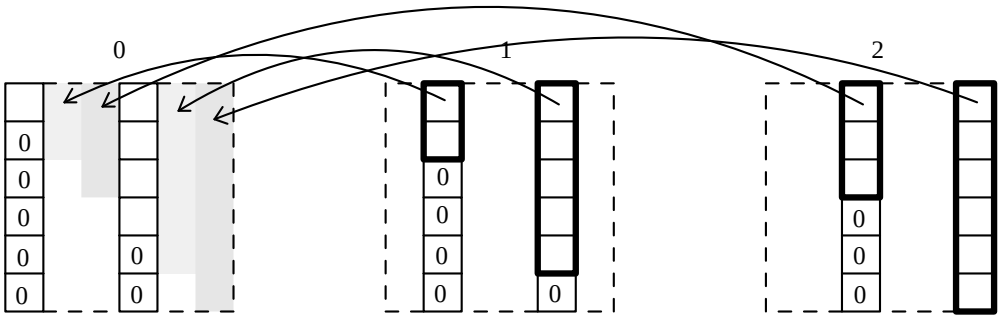
Jeśli w danym kroku proces nie zajmuje się kolumną wiodącą, oczekuje jedynie na otrzymanie (od procesu który się tym aktualnie zajmuje) numeru wiersza do zamiany oraz fragmentu kolumny wiodącej.

Następnie wszystkie procesy dokonują eliminacji pozostałych wierszy (każdy proces dokonuje tego tylko w kolumnach, które posiada).



Rysunek 2: Przykładowy jeden z kroków algorytmu: proces o numerze 2 rozsyła fragment kolumny wiodącej do pozostałych procesów (0 i 1).

Po zakończeniu eliminacji jeden z procesów (główny) zbiera niezerowe fragmenty kolumn (rys. 3), po czym dokonuje podstawienia wstecznego.



Rysunek 3: Zebranie przez proces główny niezerowych fragmentów kolumn.

2 Opis programu

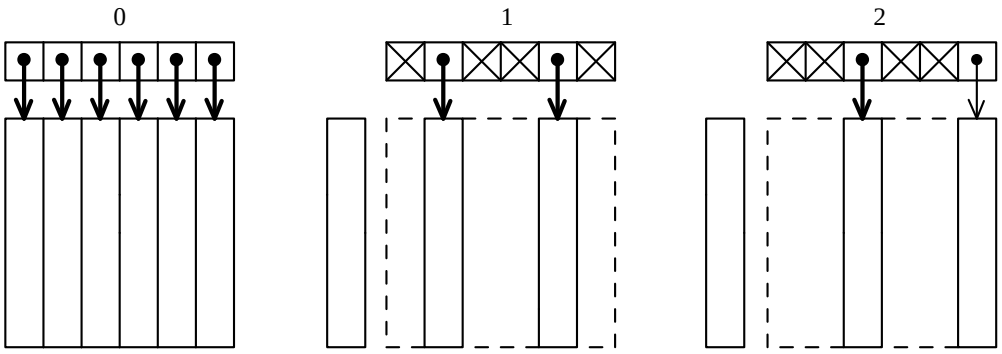
Dla zwiększenia przejrzystości algorytmu poszczególne bloki funkcjonalne umieszczone są w osobnych funkcjach (`Backsubstitution()`, `PivotSearch()`, `DivRow()`, `Elimination()`, `SwapRows()`). Pozwoliło to na wykorzystywanie ich w różnych powtarzających się miejscach, a ponadto umożliwiło w prosty sposób stworzenie funkcji obliczającej zadanie sekwencyjnie (bez MPI).

Funkcja dokonująca eliminacji wykonuje operacje po kolei dla każdej kolumny (a nie wiersza!). Związane jest to oczywiście z tym, że w pamięci obok siebie znajdują się dane z jednej kolumny.

2.1 Organizacja danych w pamięci

Macierz przechowywana jest w pamięci w postaci wektorów, z których każdy reprezentuje jedną kolumnę. Takie rozwiązanie umożliwia proste i szybkie wysyłanie/odbieranie kolumny lub jej fragmentu. Wektor wyrazów wolnych traktowany jest jako jedna z kolumn macierzy (czyli *de facto* macierz ma $N + 1$ kolumn).

Ponieważ tylko proces główny musi znać całą macierz, jedynie on posiada w pamięci zaalokowane wszystkie wektory. Pozostałe procesy mają zaalokowane jedynie te, na których wykonują obliczenia. Dodatkowo muszą posiadać miejsce przeznaczone na przyjmowanie kolumny wiodącej (na rysunku 4 zaznaczone przez dodatkowy wektor po lewej stronie macierzy). W praktyce oznacza to, że dla macierzy 3000×3000 proces główny alokuje ok. 35MB, natomiast każdy z pozostałych procesów po ok. 4,5MB.



Rysunek 4: Przechowywanie macierzy przez poszczególne procesy.

2.2 Komunikacja między procesami

Wysyłanie danych następuje w większości przypadków za pomocą funkcji wysłania komunikatu nieblokującego (`MPI_Isend()`) - tj. proces nie czeka, aż adresat otrzyma przesyłkę. Takie działanie możliwe jest dzięki temu, że na danych które są wysyłane nie przeprowadzane są już potem żadne operacje.

Odbieranie danych za pomocą tradycyjnego odbierania blokującego `MPI_Recv()`.

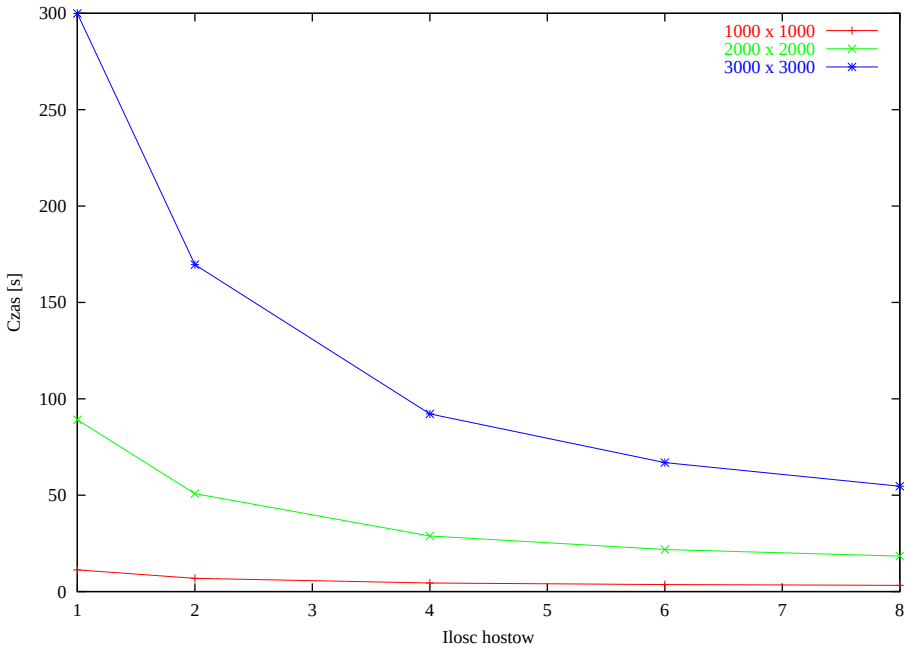
Aby w trakcie działania algorytmu przesłać do pozostałych procesów numer wiersza wiodącego i fragment kolumny wiodącej, dane te są upychane do jednego bufora wysyłanego jako `MPI_FLOAT`. Numer wiersza wiodącego mapowany jest na liczbę typu `float` (!).

3 Przykładowe czasy obliczeń

Testy przeprowadzone na klastrze `anthill` (PIII 500MHz 256MB + 7 × PIII 600 MHz 64MB). Kompilacja `gcc` z opcją `-O3` (maksymalna optymalizacja).

Ilość hostów	Rozmiar macierzy		
	1000×1000	2000×2000	3000×3000
1	11,28	89,166	299,94
2	6,9	50,81	169,63
4	4,43	28,81	92,15
6	3,63	21,85	66,92
8	3,23	18,4	54,64

Tablica 1: Czas obliczeń (w sekundach) w zależności od ilości hostów oraz wielkości macierzy. W przypadku jednego hosta podany jest czas obliczeń zwykłego algorytmu sekwencyjnego.



Rysunek 5: Czas obliczeń w zależności od ilości hostów dla różnych wielkości macierzy.

Widać wyraźnie (zwłaszcza dla macierzy 3000×3000), że wraz z wzrostem ilości hostów maleje przyśpieszenie jakie daje zrównoleglenie. Przykładowo dołożenie dwóch hostów do 6 powoduje skrócenie czasu obliczeń jedynie o 12 sekund.