Gabriel Chęć, Tomasz Gajda

Mnożenie macierzy

Algorytm Cannon'a - Projekt 2 **(UPC)**

1. **Wstęp teoretyczny** (taki sam jak w projekcie z MPI)

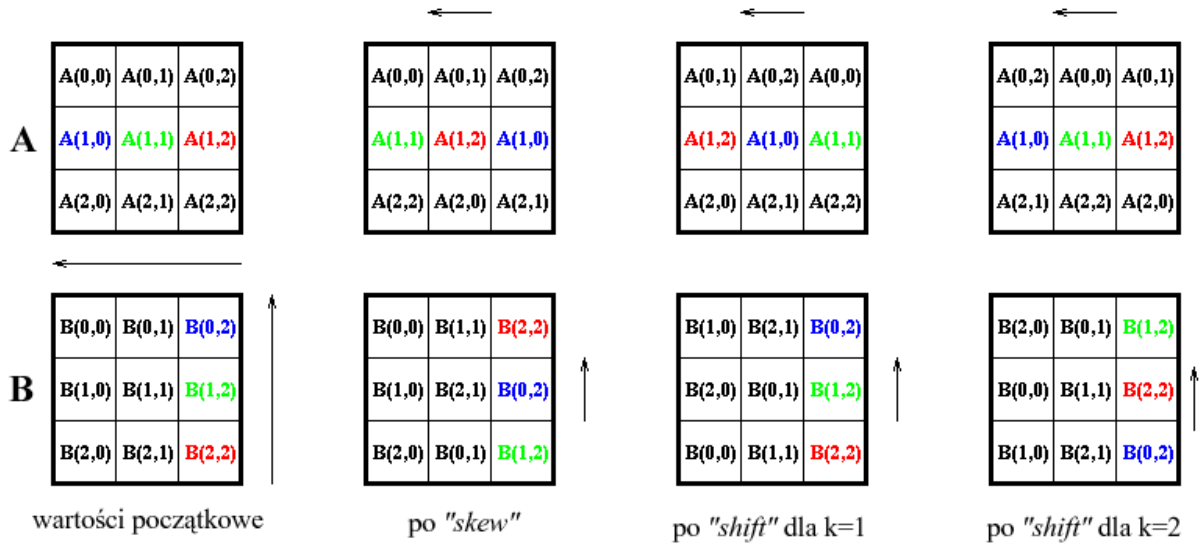
Algorytm Cannon’a jest rozproszonym algorytmem mnożenia macierzy dla siatek dwuwymiarowych, który po raz pierwszy opisany został w 1969 r. przez Lynn’a Elliot’a Cannon’a.

- Algorytm jest efektywny jedynie dla struktury kwadratowej,
- Działających w algorytmie procesorów jest tyle jaki jest rozmiar macierzy

Algorytm Cannon’a:

1. Oznaczamy mnożone macierze jako **A** i **B**, macierz wynikową jako **C**, macierz procesów jako **P.**
2. Proces **P(i, j)** początkowo przechowuje **A(i, j)**, a **B(i, j)** oblicza blok **C(i, j)** macierzy wynikowej.
3. Przekształcamy **A** i **B** w taki sposób, aby każdy proces mógł niezależnie rozpocząć **mnożenie** swoich lokalnych podmacierzy.
   Przesuwamy wszystkie podmacierze **A(i, j)** w lewo o **i** kroków i wszystkich podmacierzy **B(i, j)** w górę o **j** kroków.
4. Wykonujemy mnożenie bloków lokalnych.
5. Każdy blok **A** przesuwamy o jeden krok w lewo, a każdy blok **B** przesuwamy o jeden krok w górę.
6. Wykonujemy mnożenie kolejnych bloków, dodajemy do wyniku częściowego i powtarzamy to, aż **wszystkie bloki zostaną pomnożone**.

Przykład obliczania jednego z elementów macierzy wynikowej



C(1, 2) = A(1, 0) * B(0, 2) + A(1, 1) * B(1, 2) + A(1, 2) * B(2, 2)

2. Implementacja

Program został napisany w języku C z wykorzystaniem **technologii PGAS UPC**. W programie możemy znaleźć 4 funkcje:

- ***main*** - główna funkcja zajmująca się przebiegiem algorytmu, czyli obliczeniami i przekształceniami,
- ***print_matrix*** - funkcja wypisująca macierz do konsoli w sposób sformatowany,
- ***initialize_matrix*** - funkcja inicjalizująca macierz poprzez wczytanie jej z pliku **.csv** i zapisanie w postaci tablicy,
- ***save_matrix*** - funkcja zapisująca macierz w postaci pliku **.csv**,

Program obsługuje flagę -v (verbose). Po uruchomieniu programu z tą flagą możemy zobaczyć wypisane w sposób **macierze mnożone** oraz **macierz wynikową**.

Rozpoczynamy od załączenia używanej przez nas biblioteki za pomocą dyrektywy ***#include "upc.h".*** Potem, alokujemy obiekty globalne które będą współdzielone pomiędzy procesorami. Korzystamy w tym miejscu ze słowa kluczowego **shared**. Trzy obiekty (**A, B i C**) będą zawierały nasze macierze wykorzystane do obliczeń. Za pomocą procesu MASTER i funkcji **initialize_matrix**, inicjalizujemy nasze macierze z plików **csv**. Następnie w pętli **upc_forall** każdy wątek przetwarza obliczenia na zaalokowanych obiektach współdzielonych. Procesy są ułożone w architekturze siatkowej - każdemu z nich odpowiada blok macierzy wejściowych. Rozpoczynamy od wstępnego przesunięcia bloków, procesy przemnażają przypisane bloki, a następnie wykonują kolejne przesunięcie. Możemy określić ile razy wykonuje się ta operacja na podstawie wymiaru naszych macierzy. Uzyskane wyniki sumujemy i zapisujemy w pomocniczej tablicy w polu o indeksie równym ID procesu. Na koniec za pomocą procesu głównego i funkcji **save_matrix** zapisujemy **csv.**

3. Uruchomienie

Do przygotowania i uruchomienia rozwiązania służy plik **makefile**, w którym zdefiniowane są następujące komendy:

- **build -** komenda kompiluje nasz program,
- **nodes -** tworzy plik z węzłami,
- **run** - uruchamia skompilowany program,
- **run__verb -** jw. z parametrem verbose,
- **clean** - usuwa utworzone przy uruchamianiu pliki.

4. Zawartość katalogu

- **main.c -** program obsługujący zadanie,
- **makefile -** plik make pozwalający na łatwe uruchamianie,
- **A.csv -** pierwsza mnożona macierz,
- **B.csv -** druga mnożona macierz,
- **Result.csv -** przykład docelowej wynikowej macierzy,
- **Dokumentacja.pdf** - opis i dokumentacja projektu.

5. Materiały

- **Prezentacja - LINK**
- **GitHub - LINK**