**Лабораторна робота №4**

**Тема:** «Паралельні алгоритми множення матриць.»

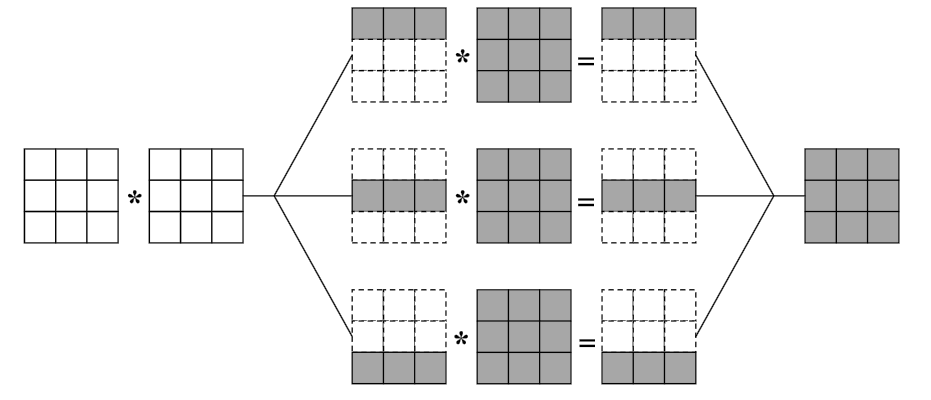
**Мета:** Навчитися реалізовувати задачу множення матриць за допомогою паралельного базового алгоритму та паралельних алгоритмів стрічкового та блочного поділу.

**Теоретичні відомості**

**Базовий паралельний алгоритм множення матриць**

З визначення операції матричного множення випливає, що вирахування всіх елементів матриці С може бути виконано незалежно один від одного. Як результат, можливий підхід для організації паралельних обчислень полягає у використанні в якості базової підзадачі процедури визначення одного елемента результуючої матриці С. Для проведення всіх необхідних обчислень кожна підзадача повинна робити обчислення над елементами одного рядка матриці А і одного стовпця матриці В. Загальна кількість одержуваних підзадач являється рівною (по числу елементів матриці С)..

Для обчислення одного рядка матриці С необхідно, щоб в кажній підзадачі містився рядок матриці А і був забезпечений доступ до всіх стовпців матриці B. Спосіб організації паралельних обчислень представлений на рис. 1.



*Рис.1 Організація обчислень при базовому паралельному алгоритмі множення матриць.*

Коли розмір n матриць виявляється більше, ніж число p обчислювальних елементів (процесорів і/або ядер), базові підзадачі можна укрупнити, об'єднавши в рамках однієї підзадачі кілька сусідніх рядків матриці. У цьому випадку вихідна матриця A і матриця результатів розбиваються на ряд горизонтальних смуг. Розмір смуг при цьому слід вибрати рівним k = n/p (у припущенні, що n кратно p), що дозволить як і раніше забезпечити рівномірність разподілу обчислювального навантаження на елементи.

Псевдокод:

// Функция паралельного множення матриць

void ParallelResultCalculation(double\* pAMatrix, double\* pBMatrix, double\* pCMatrix, int Size)

{

int i, j, k;

//паралельний запуск циклу

Paralell\_for (i=0; i<Size; i++)

for (j=0; j<Size; j++)

for (k=0; k<Size; k++)

pCMatrix[i\*Size+j] += pAMatrix[i\*Size+k]\*

pBMatrix[k\*Size+j];

}

Дана функція проводить множення рядків матриці А на стовпці матриці B з використанням декількох паралельних потоков. Кожен потік виконує обчислення над кількома сусідніми рядками матриці A і, таким чином, отримує кілька сусідніх рядків результуючої матриці С.

**Алгоритм множення матриць, заснований на стрічковому розподілі даних**

В якості базової підзадачі будемо розглядати процедуру визначення одного елемента результуючої матриці С. Загальна кількість одержуваних при такому підході підзадач являється рівним  (по числу елементів матриці С).

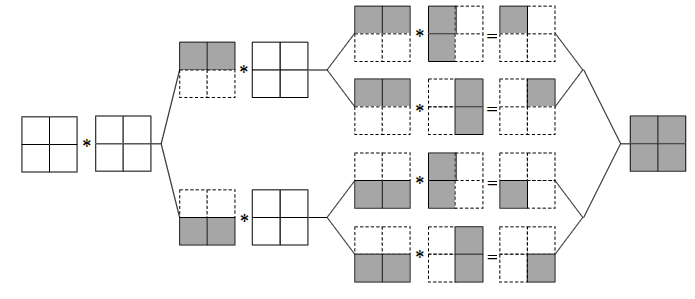
Для зниження складності та спрощення одержуваних співвідношень будемо вважати, що число блоків в матриці С по горизонталі і по вертикалі збігається. Для ефективного виконання паралельного алгоритму множення матриць доцільно виділити число паралельних потоків, що збігається з кількістю блоків матриці С, тобто таку кількість потоків, яка є повним квадратом . Додатково можна відмітити заздалегідь, що для ефективного виконання обчислень кількість потоків має бути, принаймні , кратна числу обчислювальних елементів (процесорів і/або ядер) *p* .

Для обчислення одного елемента  результуючої матриці необхідно виконати скалярне множення i-го рядка матриці A і j-го стовпця матриці B. Отже, для обчислення всіх елетів прямокутного блоку результуючої матриці



необхідно виконати скалярне множення рядків матриці A з індексами  на стовпці матриці B з індексами . Тобто необхідно розділити між потоками паралельної програми як рядки матриці A, так і стовпці матриці В. Для цього варто скористатись механізмом вкладеного паралелізму (попередня робота).

Нехай кожне нове оголошення паралельної секції поділяє потік виконання на q потоків. У цьому випадку поділ ітерацій зовнішнього циклу матричного множення між потоками паралельної програми розділить матрицю A на q горизонтальних смуг. При наступному поділі ітерацій внутрішнього циклу за допомогою механізму вкладеного паралелізму матриця B виявиться розділеною на q вертикальних смуг (рис. 2) .



*Рис.2. Алгоритм стрічкового розподілу даних на 4 потоки.*

Розмір блоків матриці С може бути підібраний таким чином, щоб загальна кількість базових подзадач співпадала з числом виділених потоків π. Якщо, наприклад, визначити розмір блокової решітки матриці С як π = q \*q, то k = m/q, l = n/q.

Псевдокод:

// Функція паралельного матричного множення

void ParallelResultCalculation(double\* pAMatrix, double\*

pBMatrix, double\* pCMatrix, int Size)

{

int i, j, k;

int NestedThreadsNum = 2;

//встановлення кількості потоків

omp\_set\_num\_threads (NestedThreadsNum );

//parallel for private (j, k)

parallel for (i=0; i<Size; i++)

//parallel for private (k)

parallel for (j=0; j<Size; j++)

for (k=0; k<Size; k++)

pCMatrix[i\*Size+j] += pAMatrix[i\*Size+k]\*

pBMatrix[k\*Size+j];

}

Дана функція проводить множення рядків матриці А на стовпці матриці B з використанням декількох паралельних потоків . Кожен потік виконує обчислення над елементами горизонтальної смуги матриці A і елементами вертикальної смуги матриці B, таким чином, отримує значення елементів прямокутного блоку результуючої матриці С.

Відзначимо, що в наведеній програмі для завдання кількість потоків, що створюються на кожному рівні вкладеності паралельних областей, використовується змінна NestedThreadsNum (в даному випадку значення рівне 2 – дане значення повинне переустановлюватися для визначення необхідного числа потоків).

**Блочний алгоритм множення матриць**

Будемо припускати, що всі матриці є квадратними розміру n × n, кількість блоків по горизонталі і вертикалі є однаковим і рівним q (тобто розмір всіх блоків дорівнює k × k, k = n/q). При такому представленні даних операція матричного множення може бути подана в наступному вигляді:



Де кожен елемент визначається за наступним співвідношенням:



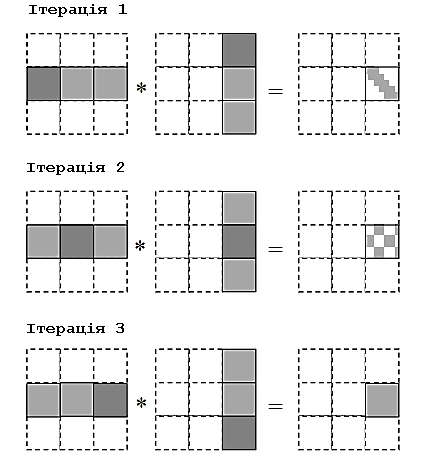
При блочному розбитті даних для визначення базових підзадач взято за основу обчислення, виконуване над матричними блоками. З урахуванням сказаного базова підзадача визначається як процедура обчислення всіх елементів одного з блоків матриці С.

Для виконання всіх необхідних обчислень базовим підзадачам повинні бути доступні відповідні набори рядків матриці A і стовпців матриці B.

Широко відомі паралельні алгоритми множення матриць, засновані на блочному поділі даних, орієнтовані на багатопроцесорні обчислювальні системи з розподіленою пам'яттю. При розробці алгоритмів, орієнтованих на використання паралельних обчислювальних систем з розподіленою пам'яттю, слід враховувати, що розміщення всіх необхідних даних в кожній підзадачі (в даному випадку - розміщення в підзадачах необхідних наборів рядків матриці А і стовпців матриці В) неминуче призведе до дублювання і до значного зростання обсягу пам'яті.

Як результат, обчислення повинні бути організовані таким чином, щоб в кожний поточний момент часу підзадачі містили лише частину необхідних для проведення розрахунків даних, а доступ до основної частини даних забезпечувався б за допомогою передачі повідомлень . До числа алгоритмів, що реалізують описаний підхід, відносяться алгоритм Фокса (Fox) і алгоритм Кеннона (Cannon). Відмінність цих алгоритмів полягає в послідовності передачі матричних блоків між процесорами обчислювальної системи.

За основу паралельних обчислень для матричного множення при блочному поділі даних береться підхід, при якому базові підзадачі відповідають за обчислення окремих блоків матриці C і при цьому підзадачі на кожній ітерації розрахунків обробляють тільки по одному блоку вихідних матриць A і B. Для нумерації підзадач слід використовувати індекси розміщуваних в підзадачах блоків матриці C, тобто підзадача (i,j) відповідає за обчислення блоку  – тим самим, набір підзадач утворює квадратну решітку, що відповідає структурі блочного представлення матриці C.



*Рис.3. Схема організації блочного множення смуг.*

Згідно обчислювальної схемою блочного алгоритму множення матриць, (попередня робота), на кожній ітерації алгоритму кожен потік паралельної програми виконує обчислення над матричними блоками. Номер блоку, який повинен оброблятися потоком в даний момент, обчислюється на підставі положення потоку в «решетці потоків »та номеру поточної ітерації.

Псевдокод:

void ParallelResultCalculation (double\* pAMatrix,

double\* pBMatrix, double\* pCMatrix, int Size)

{

int ThreadNum = 4;

int GridSize = int (sqrt((double)ThreadNum));

int BlockSize = Size/GridSize;

//Запускається паралельно

{ int ThreadID = get\_thread\_num();

int RowIndex = ThreadID/GridSize;

int ColIndex = ThreadID%GridSize;

for (int iter=0; iter<GridSize; iter++)

{

for (int i=RowIndex\*BlockSize;

i<(RowIndex+1)\*BlockSize; i++)

for (int j=ColIndex\*BlockSize;

j<(ColIndex+1)\*BlockSize; j++)

for (int k=iter\*BlockSize;

k<(iter+1)\*BlockSize; k++)

pCMatrix[i\*Size+j] +=

pAMatrix[i\*Size+k] \* pBMatrix[k\*Size+j];

}}}

Для визначення числа потоків, які будуть використовуватися при виконанні операції матричного множення, введено змінну ThreadNum. Встановимо число потоків в значення ThreadNum за допомогою функції omp\_set\_num\_threads . Визначається розмір «решітки потоків » GridSize і розмір матричного блоку BlockSize. Далі слід визначити ідентифікатор потоку ThreadID. Положення потоку в решітці потоків визначається змінним RowIndex і ColIndex. Номер рядка потоків, в якому розташований даний потік, є результат цілочисельного ділення ідентифікатора потоку на розмір решітки потоків. Номер стовпця, в якому розташований потік - є залишок від ділення ідентифікатора потоку на розмір решітки потоків.

Для виконання операції матричного множення кожен потік обчислює елементи блоку результуючої матриці. Для цього кажен потік повинен виконати GridSize ітерацій алгоритму, кожна такая ітерація є множення матричних блоків; ітерації виконуються зовнішнім циклом for по змінній iter .

**Завдання до лабораторної роботи**

Відповідно до варіанту (табл.1.)виконати наступні завдання:

1. Розробити багатопотокову програму, яка реалізує запропонований паралельний алгоритм множення матриці А розмірності  на матрицю В розмірністю 

, 

1. Провести 10 експериментів поступово нарощуючи розмірності, спершу  *m* (5 експериментів) потім *l*(5 експериментів).
2. Для кожного експерименту визначити час виконання алгоритму.
3. Побудувати два графіки, на яких відображається зміна часу виконання алгоритму в залежності від зміни розмірностей *n* та *m.*

*Таблиця 1. Варіанти завдань*

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант № | Елементи матриці А | Елементи матриці В | Алгоритм для реалізації |
| 1 |  |  | *базовий* |
| 2 |  |  | Стрічкове розбиття |
| 3 |  |  | Блочне розбиття |
| 4 |  |  | Базовий |
| 5 |  |  | Стрічкове розбиття |
| 6 |  |  | Блочне розбиття |
| 7 |  |  | Базовий |
| 8 |  |  | Стрічкове розбиття |
| 9 |  |  | Блочне розбиття |
| 10 |  |  | Базовий |