**Лабораторна робота №3**

**Тема:** «Паралельні алгоритми множення матриці на вектор.»

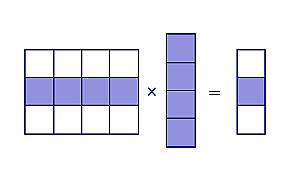
**Мета:** Навчитися реалізовувати задачу множення матриці на вектор за допомогою паралельних алгоритмів стрічкового та блочного поділу.

**Теоретичні відомості**

# Множення матриці на вектор

У результаті множення матриці *А* розмірності *n×m* і вектора *b*, що складається з *n* елементів, виходить вектор *с* розміру *m*, кожен i-ий елемент якого є результат скалярного множення i-го рядка матриці *А* (позначимо цю строчку ) і вектора *b* (рис. 1):

, 



*Рис.1. Алгоритм множення матриці на вектор.*

Тим самим, отримання результуючого вектора *c* припускає повторення m однотипних операцій по множенню рядків матриці *A* і вектора *b*. Кожна така операція включає множення елементів рядка матриці і вектора b і подальше підсумовування отриманих результатів. Псевдокод даної операції виглядає наступним чином:

for (int i = 0; i < m; i++)

{

c[i] = 0;

for (int j = 0; j < n; j++)

{

c[i] += A[i][j]\*b[j];

}

}

# Способи поділу матриць

Для багатьох методів матричних обчислень характерним є повторення одних і тих же обчислювальних дій для різних елементів матриць. Даний момент свідчить про наявність паралелізму за даними при виконанні матричних розрахунків і, як результат, розпаралелювання матричних операцій зводиться в більшості випадків до поділу оброблюваних матриць між процесорами обчислювальної системи. Вибір способу поділу матриць призводить до визначення конкретного методу паралельних обчислень; існування різних схем розподілу даних, що породжує цілий ряд паралельних алгоритмів матричних обчислень.

Найбільш загальні і широко використовувані способи поділу матриць складаються в розбитті даних на смуги (по вертикалі або горизонталі ) або на прямокутні фрагменти (блоки).

1. *Стрічкове розбиття матриці.* При *стрічковому* (block - striped) розбитті кожному процесору виділяється та чи інша підмножина рядків (*rowwise* чи горизонтальне розбиття ) або стовпців (*columnwise* або вертикальне розбиття) матриці ( рис. 2а і 2б ). Поділ рядків і стовпців на смуги в більшості випадків відбувається на безперервній (послідовної) основі. При такому підході для горизонтального розбиття по рядках матриця A представляється у вигляді:



де  - є i - й рядок матриці *A* (передбачається, що кількість рядків m кратно числу процесорів p, тобто m = k ⋅ p ). У всіх алгоритмах матричного множення і множення матриці на вектор, які будуть розглянуті, використовується поділ даних на безперервній основі.

Інший можливий підхід до формування смуг полягає у застосуванні тієї чи іншої схеми *чергування* (циклічності) рядків або стовпців. Як правило, для чергування використовується число процесорів p - в цьому випадку при горизонтальному розбитті матриця *A* приймає вид



Циклічна схема формування смуг може виявитися корисною для кращого балансування обчислювального навантаження процесорів (наприклад , при вирішенні системи лінійних рівнянь з використанням методу).

*2. Блочне розбиття матриці*. При блочному (checkerboard block) поділі матриця ділиться на прямокутні набори елементів - при цьому, як правило, використовується поділ на безперервній основі. Нехай кількість процесорів становить , кількість рядків матриці є кратним s , а кількість стовпців - кратним q , тобто  і . Уявімо вихідну матрицю *A* у вигляді набору прямокутних блоків таким чином (рис. 2в) :

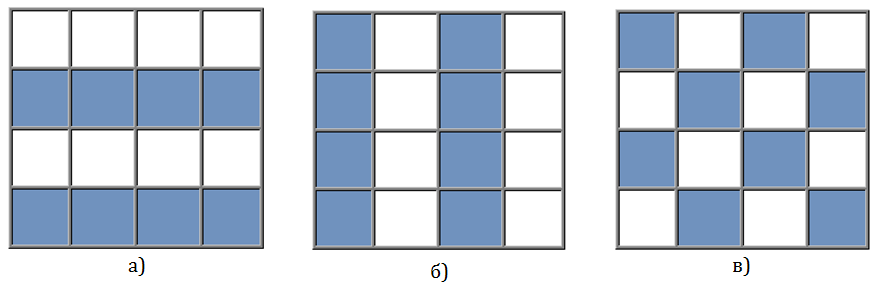


де  - блок матриці, що складається з елементів:





При такому підході доцільно, щоб обчислювальна система мала фізичну або, принаймні, логічну топологію процесорної решітки з s рядків і q стовпців. У цьому випадку при поділі даних на безперервній основі процесори, сусідні в структурі решітки, обробляють суміжні блоки вихідної матриці. Слід зазначити, що і для блокової схеми може бути застосоване циклічне чергування рядків і стовпців.

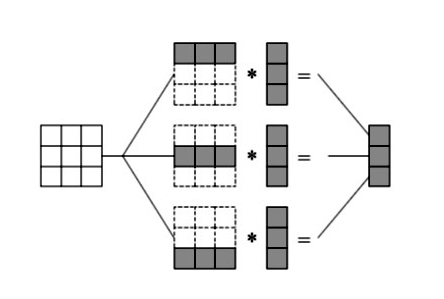


*Рис.2. Способи розподілу елементів матриці між процесорами системи*

# Алгоритм розподілу даних по рядках

При такому способі поділу даних як базова підзадача може бути обрана операція скалярного множення одного рядка матриці на вектор. Після завершення обчислень кожна базова підзадача визначає один з елементів вектора результату *c*.

У загальному випадку схема інформаційної взаємодії підзадач в ході виконуваних обчислень показана на рис. 3.



*Рис.3. Організація обчислень при виконанні паралельного алгоритму множення матриці на вектор, заснованого на поділі матриці по рядках*

У процесі множення щільної матриці на вектор кількість обчислювальних операцій для отримання скалярного добутку однакова для всіх базових підзадач. Тому коли число *p* обчислювальних елементів менше числа *m* базових підзадач, слід об'єднати базові підзадачі таким чином, щоб кожен ОЕ виконував кілька таких задач, відповідних безперервній послідовності рядків матриці А. У цьому випадку після закінчення обчислень кожна базова підзадача визначає набір елементів (блок) результуючого вектора *с.*

Псевдокод алгоритму має наступний вигляд:

void ParallelCalculation(double[] Matrix,

double[] Vector, double[] Result)

{

int i, j;

//паралельний запуск циклу

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

{

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

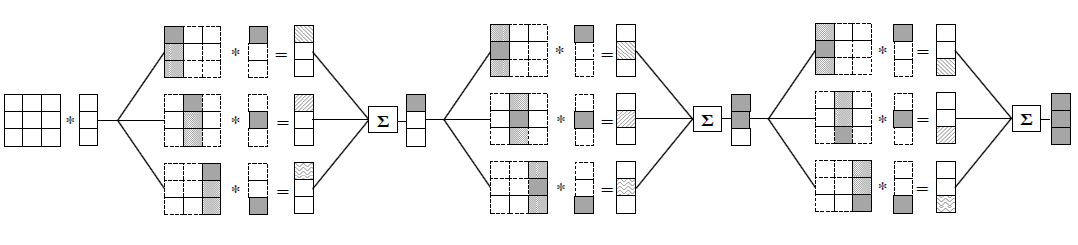
Result[i] += Matrix[i\*Size+j]\*Vector[j];

} }

Дана функція проводить множення рядків матриці на вектор з використанням декількох паралельних потоків. Кожен потік виконанує обчислення над кількома сусідніми рядками матриці і, таким чином, отримує блок результуючого вектора с.

# Алгоритм розподіл даних по стовпцях

При такому способі розділення даних в якості базової підзадачі може бути обрана операція множення стовпця матриці А на один з елементів вектора b . Для організації обчислень в цьому випадку кожна базова підзадача  повинна мати доступ до i - го стовпця матриці А. Кожна базова задача виконує множення свого стовпця матриці А на елемент , у результаті в кожній підзадачі виходить вектор  проміжних результатів. Для отримання елементів результуючого вектора *с* необхідно підсумувати вектори , які були отримані кожної підзадачею.



*Рис.4. Організація обчислень при виконанні паралельного алгоритму множення матриці на вектор з використанням розбиття матриці по стовпцях.*

Для того, щоб реалізувати такий підхід, необхідно розпаралелити внутрішній цикл алгоритму множення матриці на вектор. В результаті даного підходу паралельні області будуть створюватися для обчислення  
кожного окремого елемента результуючого вектора. програма буде представлятися у вигляді набору послідовних (однопотокових) і паралельних (багатопотокових) ділянок . Подібний принцип організації паралелізму отримав назву «fork–join» або пульсуючого паралелізму. При виконанні багатопотокової ділянки кожен потік виконує множення одного елемента свого стовпця вихідної матриці на один елемент вектора. Наступна за цим однопотокова ділянка підсумовує отримані результати для того, щоб обчислити елемент результуючого вектора (рис.4). Таким чином, програма складатиметься з чергування n многопотокових і n однопотокових ділянок .

Псевдокод алгоритму:

void ParallelCalculation(double[] Matrix, double[] Vector, double[] Result)

{

int i, j;

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

{

//паралельний запуск циклу

parallel\_for (j=0; j<Size; j++)

Result[i] += Matrix[i\*Size+j]\*Vector[j];

}

}

Результати експериментів показують, що розроблена вище пседофункція паралельного множення матриці на вектор не завжди дає вірний результат. Причина некоректної роботи криється в неправильному використанні змінних, що являються загальними для декількох потоків. Такими загальними змінними є елементи вектора Result. У ході обчислень кілька різних потоків можуть, наприклад , спробувати одночасно записати нові значення в одну і ту ж загальну змінну, що призведе до отримання помилкових результатів . Для виключення взаємовпливу потоки повинні використовувати загальні дані з дотриманням правил, які забезпечували б користування загальних змінних в кожен момент часу тільки одним потоком (а потоки, що намагаються отримати доступ до «зайнятих » загальних даних, повинні блокуватися).

# Алгоритм блочного розподілу даних

Загальна схема паралельних обчислень для операції множення матриці на вектор при блочному розподілі вихідних даних полягає в наступному. Після перемноження блоків матриці *A* і вектора *b* кожна підзадача *(i, j)* міститиме вектор часткових результатів *c '(i, j),* який визначається у відповідності з виразами:

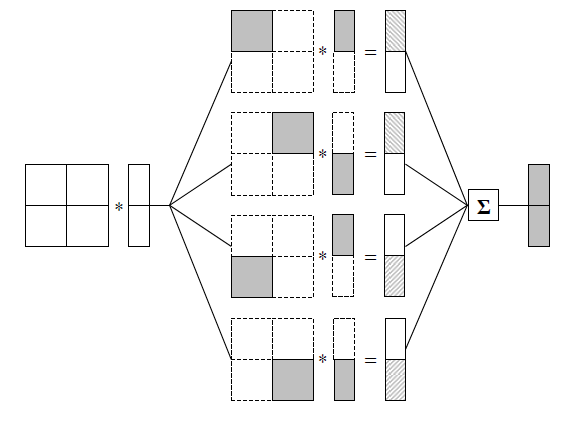
, , , ,



Кожен потік обчислює блок вектора часткових результатів вихідної задачі множення матриці на вектор. Повний результат - вектор c можна визначити сумування елементів блоку вектора часткових результатів з однаковими індексами по всіх підзадачах, що належать до одних і тих самих рядків решітки потоків, тобто:



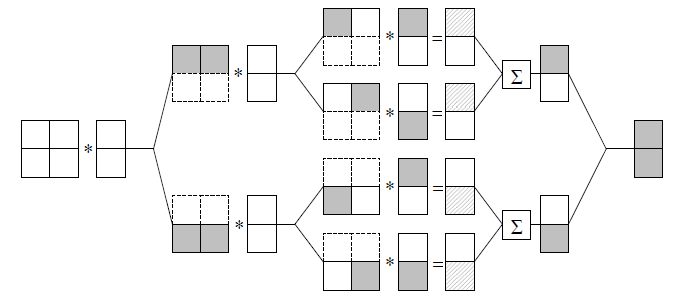
Загальна схема виконуваних обчислень для множення матриці на вектор при блочному поділі даних показана на рис. 5



*Рис.5. Загальна схема паралельного алгоритму множення матриці на вектор при блочному розподілі даних.*

При зроблених припущеннях загальна кількість базових підзадач збігається з числом виділених потоків. Так, якщо визначити число потоків, то розмір блоків матриці A визначається співвідношеннями: k = m/q, l=n/q, де k і l є кількість рядків і стовпців в блоках матриці А. Такий спосіб визначення розміру блоків призводить до того, що обсяг обчислень в кожній підзадачі є рівним і тим самим досягається повна збалансованість обчислювального навантаження між потоками.

Всередині однієї паралельної секції, що реалізує блок, можна оголосити другу паралельну секцію, яка розділить кожен з потоків зовнішньої паралельно секції на кілька вкладених потоків. Продемонструємо описаний підхід у випадку, коли при оголошенні кожної нової паралельної секції потік виконання поділяється на два. При перемножуванні матриці на вектор для зовнішнього циклу оголосимо зовнішню паралельну секцію. Потоки цієї секції поділяють між собою рядки матриці - кожен потік у своїх обчисленнях використовує горизонтальну смугу матриці. Далі, використовуючи механізм вкладеного паралелізму, оголосимо внутрішню паралельну секцію: потоки цієї паралельної секції ділять між собою стовпці рядка матриці. Таким чином, буде реалізовано блоковий поділ даних. Схема виконання даного паралельного алгоритму представлена ​​на рис. 6.



*Рис.6. Організація обчислень при виконанні паралельного алгоритму множення матриці на вектор з використанням розбиття матриці на блоки і вкладеного паралелізму.*

Псевдокод алгоритму:

void ResultCalculation(double[] Matrix,double[] Vector,

double[] Result,int Size)

{

//кількість вкладених потоків

int ThreadsNum = 2;

//встановлення необхідної кількості потоків

setThread(ThreadsNum);

//паралельний запуск циклу

parallel\_for (int i=0; i<Size; i++)

{

double ThreadResult = 0;

//паралельний запуск циклу

parallel\_for (int j=0; j<Size; j++)

{

ThreadResult += Matrix[i\*Size+j]\*Vector[j];

Result[i] = ThreadResult;

}

}

}

Слід відзначити, що в наведеному псевдокоді для завдання кількість потоков, що створюються на кожному рівні вкладеності паралельних областей, використовується змінна ThreadsNum (в даному варіанті програми її значення встановлюється рівним 2 - значення має перевизначатися при зміні необхідного числа потоків).

**Завдання до лабораторної роботи**

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

1. Розробити програму, яка реалізує запропонований паралельний алгоритм множення мантиці А розмірністю *n×m* на вектор В розміністю *n.*

, 

1. Провести 10 експериментів поступово нарощуючи розмірності, спершу *n* (5 експериментів) потім *m*(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*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варіанту |  |  | Тип розподілу даних | Розмірність матриці і вектора |
| 1 |  |  | блочний, стрічковий | *n=m*=10 |
| 2 |  |  | блочний  по стовпцях | *n=5 m*=10 |
| 3 |  |  | блочний,  по стовпцях | *n=7 m*=12 |
| 4 |  |  | блочний, стрічковий | *n= 6 m*=10 |
| 5 |  |  | блочний | *n=m*=12 |
| 6 |  |  | блочний,  по стовпцях | *n=5 m*=15 |
| 7 |  |  | блочний, стрічковий | *n=12 m*=8 |
| 8 |  |  | блочний  по стовпцях | *n=9 m*=20 |
| 9 |  |  | блочний,  по стовпцях | *n=m*=10 |
| 10 |  |  | блочний, стрічковий | *n=6 m*=20 |