

Міністерство Освіти і Науки України
Національний Університет “Львівська Політехніка”



Кафедра ЕОМ

МНОЖЕННЯ МАТРИЦЬ НА КІЛЬЦЕВІЙ СТРУКТУРІ ЗАСОБАМИ МРІ

Методичні вказівки
до лабораторної роботи з курсу “Паралельні та розподілені обчислення”
для студентів спеціальності 123 - “Комп’ютерна інженерія”

Затверджено
на засіданні кафедри
”Електронні обчислювальні машини”
Протокол № _ від 2022 року

Львів – 2022

МЕТА РОБОТИ. Засвоєння основних методів та алгоритмів моделювання паралельних обчислювальних процесів, принципів побудови відповідних структур, набуття початкових практичних навиків проектування таких засобів та ознайомлення з бібліотекою MPI.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Перемноження матриці на матрицю на кільцевій структурі. Множення матриці на вектор і матриці на матрицю є базовими макроопераціями для багатьох задач лінійної алгебри, наприклад ітераційних методів розв'язання систем лінійних рівнянь і т.п. Тому приведені алгоритми тут можна розглядати як фрагменти в алгоритмах цих методів. Розглянемо три алгоритми множення матриці на матрицю. Розмаїтість варіантів алгоритмів виникає із-за розмаїтості обчислювальних систем і розмаїтості розмірів задач. Розглядаються і *різні варіанти завантаження даних* у систему: завантаження даних через один комп'ютер; і завантаження даних безпосередньо кожним комп'ютером з дискової пам'яті. Якщо завантаження даних здійснюється через один комп'ютер, то дані зчитуються цим комп'ютером з дискової пам'яті, розрізаються на частини, які розсилаються іншим комп'ютерам. Але дані можуть бути підготовлені і заздалегідь, тобто заздалегідь розрізані вроздріб і кожна частина записана на диск у виді окремого файлу зі своїм ім'ям; потім кожен комп'ютер безпосередньо зчитує з диска, призначений для нього файл.

Алгоритм - Перемноження матриці на матрицю на кільцевій структурі.

Задано дві вихідні матриці A і B . Обчислюється добуток $C = A \times B$, де A - матриця $n_1 \times n_2$, і B - матриця $n_2 \times n_3$. Матриця результатів C має розмір $n_1 \times n_3$. Вихідні матриці попередньо розрізані на смуги, смуги записані на дискову пам'ять окремими файлами зі своїми іменами і доступні всім комп'ютерам. Матриця результатів повертається в нульовий процес.

Оскільки за умовою в комп'ютерах знаходиться по одній смузі матриць, то смуги матриці B (або смуги матриці A) необхідно "прокрутити" по кільцю комп'ютерів повз смуги матриці A (матриці B). На кожному з таких кроків обчислюється тільки частина смуги. Процес i обчислює на j -м кроці добуток i -ї горизонтальної смуги матриці A j -ї вертикальної смуги матриці B , добуток отриманий у підматриці (i,j) матриці C .

Обчислення відбувається в такій послідовності.

1. Кожен комп'ютер зчитує з дискової пам'яті відповідну йому смугу матриці A . Нульова смуга повинна зчитуватися нульовим комп'ютером, перша смуга - першим комп'ютером і т.д., остання смуга - зчитується останнім комп'ютером.

2. Кожен комп'ютер зчитує з дискової пам'яті відповідну йому смугу матриці B . У даному випадку нульова смуга повинна зчитуватися нульовим комп'ютером, перша смуга - першим комп'ютером і т.д., остання смуга - зчитується останнім комп'ютером.

3. Обчислювальний крок 1. Кожен процес обчислює одну підматрицю добутку. Вертикальні смуги матриці B зсуваються уздовж кільця комп'ютерів.

4. Обчислювальний крок 2. Кожен процес обчислює одну підматрицю добутку. Вертикальні смуги матриці В зсуваються уздовж кільця комп'ютерів. І т.д.

5. Обчислювальний крок p_1-1 . Кожен процес обчислює одну підматрицю добутку. Вертикальні смуги матриці В зсуваються уздовж кільця комп'ютерів.

6. Обчислювальний крок p_1 . Кожен процес обчислює одну підматрицю добутку. Вертикальні смуги матриці В зсуваються уздовж кільця комп'ютерів.

7. Матриця С збирається в нульовому комп'ютері.

Якщо "прокручувати" вертикальні смуги матриці В, то матриця С буде розподілена горизонтальними смугами, а якщо "прокручувати" горизонтальні смуги матриці А, то матриця С буде розподілена вертикальними смугами.

Алгоритм характерний тим, що після кожного кроку обчислень здійснюється обмін даними. Нехай t_w , t_s , і t_p час операцій, відповідно, множення, додавання і пересилання одного числа в сусідній комп'ютер. Тоді сумарний час операцій множень дорівнює: $U = (n_1 * n_3 * n_2) * t_w$

сумарний час операцій додавань дорівнює: $S = (n_1 * n_3 * (n_2 - 1)) * t_s$

сумарний час операцій пересилань даних по всіх комп'ютерах дорівнює: $P = (n_2 * n_3) * (p_1 - 1) * t_p$

Загальний час обчислень визначимо як: $T = (U + S + P) / p_1$

Відношення часу "обчислень без обмінів" до загального часу обчислень є величина:

$$K = (U + S) / (U + S + P).$$

Якщо час передачі даних великий в порівнянні з часом обчислень, або канали передачі повільні, то ефективність алгоритму буде не висока. Тут не враховується час початкового завантаження і вивантаження даних у пам'ять системи. У смугах матриць може бути різна кількість рядків, а різниця в кількості рядків між смугами - 1. При великих матрицях цим можна знехтувати.

При достатніх ресурсах пам'яті в системі краще використовувати алгоритм, у якому мінімізовані обміни між комп'ютерами в процесі обчислень. Це досягається за рахунок дублювання деяких даних у пам'яті комп'ютерів. Такий підхід використовується, зокрема, при реалізації перемноження матриць на структурі 2D(решітка) або 3D(куб).

Використання технології паралельного програмування Message Passing Interface (MPI) для реалізації паралельного множення матриць. MPI - бібліотека функцій, яка забезпечує взаємодію паралельних процесів за допомогою механізму передачі повідомлень і не має ніяких засобів для розподілення процесів по обчислювальних вузлах і для запуску їх на виконання. MPI не містить механізмів динамічного створення і знищення процесів під час виконання програми.

Для ідентифікації наборів процесів вводиться поняття групи і комунікатора.

Процеси об'єднуються в групи, можуть бути вкладені групи. У середині групи всі процеси понумеровані. З кожною групою асоційований свій комунікатор. Тому

при здійсненні пересилок необхідно вказати ідентифікатор групи, усередині якої проводиться це пересилка.

Процедури MPI:

- ініціалізації та закриття MPI –процесів;
- реалізації комутаційних операцій типу “точка-точка”;
- реалізації колективних операцій;
- для роботи з групами процесів і комунікаторами;
- для роботи з структурами даних;
- формування топології процесів.

До базових функцій MPI відносяться:

- ініціалізація MPI;
- завершення MPI;
- визначення кількості процесів в області зв'язку;
- визначення номеру процесу, який виконується;
- передача повідомлень;
- приймання повідомлень;
- функції відліку часу.

Кожна MPI – функція характеризується способом виконання.

1. Локальна функція – виконується всередині процесу, що її викликав. Її завершення не вимагає комунікацій.

2. Нелокальна функція – для її завершення необхідно виконати MPI – процедуру іншим процесом.

3. Глобальна функція – процедуру повинні виконати всі процеси групи. Невиконання цієї умови може привести до “зависання” задачі.

4. Блокуюча функція – повернення керування з процедури гарантує можливість повторного використання параметрів, які приймали участь у виклику. Ніякої змін в стан процесу, що викликав блокуючий запит до виходу з процедури не може відбуватися.

5. Неблокуюча функція – повернення з процедури відбувається негайно, без очікування завершення операції. Завершення неблокуючих операцій здійснюється спеціальними функціями.

Операції обміну повідомленнями.

Режими обміну:

В загальному випадку є чотири режими обміну: асинхронний (стандартний), синхронний, з буферизацією, по “готовності”.

Обмін типу “точка-точка” – найпростіша форма обміну повідомленнями, в якій приймають участь тільки два процеси: джерело і адресат. Є кілька різновидностей двохточкового обміну:

- синхронний обмін – супроводжується повідомленням про завершення прийому повідомлення;
- асинхронний обмін – таким повідомленням не супроводжується;

- блокуючі прийом/передача – призупиняють виконання процесу на час приймання повідомлення.;

- неблокуючі прийом/передача - виконання процесу продовжується в фоновому режимі, а програма в потрібний момент може запитати підтвердження завершення приймання повідомлення.

Неблокуючий обмін вимагає акуратності при виконанні функцій прийому. Оскільки неблокуючий прийом завершується негайно, для системи неважливо, чи прибуло повідомлення до місця призначення чи ні. Переконалися про це можна за допомогою функції перевірки отримання повідомлення. Звичайно виклик таких функцій розміщується в циклі, який повторюється до тих пір, доки функція перевірки не поверне значення “істина” (перевірка отримання пройшла успішно). Після цього можна викликати функцію прийому повідомлення з буферу повідомлень.

Колективний обмін. В операціях використовуються не два а більше процесів. Різновидностями обміну є:

- широкосмугова передача – передача виконується від одного процесу до всіх;
- обмін з бар’єром – форма синхронізації роботи процесів, коли обмін повідомленнями проходить тільки після того, як до певної процедури звернулася певна кількість процесів;

- операції приведення – вхідними є дані кількох процесів, а результат – одне значення, яке стає доступним всі процесам, які приймали участь в обміні.

Важливою властивістю системи передачі повідомлень є гарантія збереження порядку прийому повідомлень (при відправленні одним процесом іншому кількох повідомлень вони повинні бути прийняті в тій самій послідовності в якій були відправлені). Більшість реалізацій моделі передачі повідомлень забезпечують цю властивість, але не у всіх режимах обміну.

ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Написати програму множення 2 матриць для 1-процесорної послідовної системи.
2. Написати програму множення 2 матриць на кільцевій структурі з 8 процесорів, засобами MPI. При цьому слід дотримуватися таких вимог:

2.1. Розмірності матриць та шлях передачі даних (кільце) визначаються варіантом.

2.2. Результати роботи кожного процесу повинен відображатися у тестових файлах з відповідними назвами. Остаточний результат – окремо.

2.3. Передбачити відображення процесу обміну даними(без відображення самих даних) на екрані.

3. Протестувати створені програми для різних(визначених) наборів даних і визначити час обчислень для паралельної та послідовної системи.

Набір даних 1: матриці A та B містять лише значення «1».

Набір даних 2: матриця А містить лише значення «1», матриця В містить лише значення «3».

Набір даних 3: матриці А та В містять довільні *цілі додатні* числа від 1 до 9.

Набір даних 4: матриці А та В містять довільні *цілі знакові* числа від 10 до 99.

Набір даних 5: матриці А та В містять довільні *дійсні* числа від 100 до 9999.





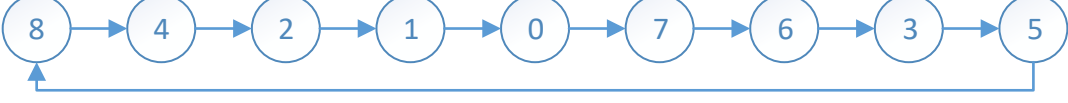
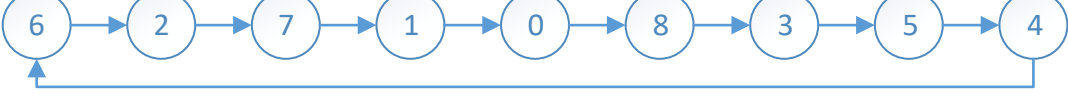





4. Зробити висновки про ефективність розпаралелення.

ЗМІСТ ЗВІТУ

1. Тема, мета, аналіз завдання.
2. Розрахунок розмірів частин матриць, що будуть опрацьовуватися окремими процесорами.
3. Результати обчислень в кожному процесорі і остаточний результат множення(скріншоти)
4. Результати дослідження ефективності паралельної програми у порівнянні з послідовною за часом виконання у табличній і графічній формі.
5. Висновки.

ВАРІАНТИ ЗАВДАНЬ

Варіант №	Граф що відображає конфігурацію зв'язків між процесами		
	N1	N2	N3
1			
	256	541	325
2			
	300	112	214
3			
	113	200	77
4			
5			
	251	75	65
6			
7			
	58	145	217
8			
	69	112	124
9			
10			
	128	89	278

11									
	187			219			120		
12									
	210			211			148		
13									
14									
	149			267			215		
15									
	147			214			265		
16									
	217			149			100		
17									
	101			217			220		
18									
	266			149			187		
19									
	197			211			99		
20									
	87			319			213		
21									
	254			117			101		

22									
	107			247			100		
23									
	187			198			99		
24									
	297			143			88		
25									
	183			140			210		
26									
	221			198			97		
27									
	95			159			137		
28									
	181			150			163		
29									
	176			149			152		
30									
	139			284			73		

ПРИКЛАД ВИКОНАННЯ

N1 = 60, N2 = 248, N3 = 149

$C[60][149] = A[60][248] * B[248][149]$

Визначення розмірів підматриць:

Для матриці A поділ відбувається по рядках. Маємо:

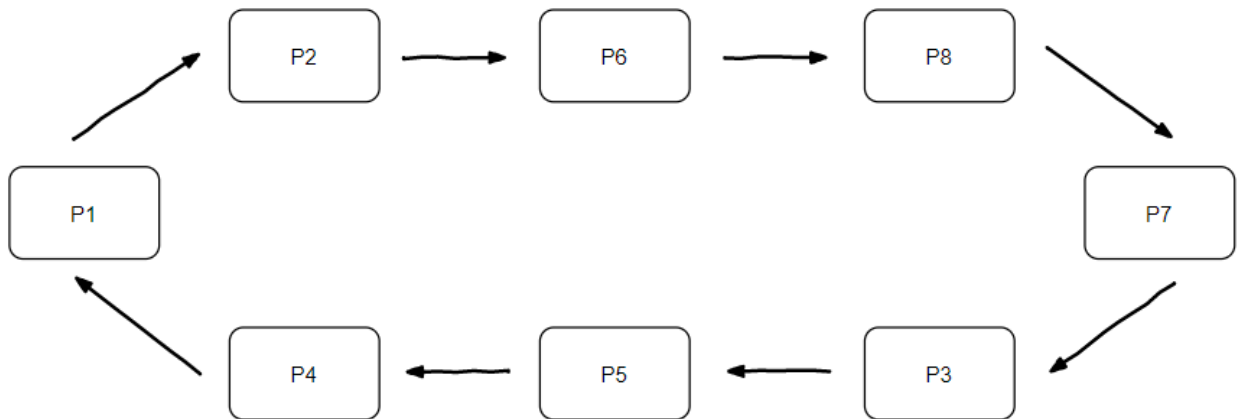
$60/8=7(4\text{-остача})$, тобто $60=4*7+4*8$.

Таким чином 4 підматриці A будуть мати по 7 рядків і 4 підматриці A будуть мати по 8 рядків. Кількість стовпців- 248.

Для матриці B поділ відбувається по стовпцях. Маємо:

$149/8=18(5\text{-остача})$, тобто $149=3*18+5*19$.

Таким чином 3 підматриці B будуть мати по 18 стовпців і 5 підматриць B будуть мати по 19 стовпців. Кількість рядків - 248.



Граф-схема кільцевого зв'язку між процесорами.

Частина програми для розбиття матриць на підматриці та їх пересилання

```
...
CountRowsSubmatrixes = N1 / 8; //кількість рядків в підматрицях
CountColumnSubmatrixes = N3 / 8 - 1; //кількість стовпців в підматрицях
NumOfMoreRows = N1 % 8; //кількість матриць з більшою кількістю рядків
NumOfMoreColumns = N3 % 8; //кількість матриць з більшою кількістю стовпців
cout << "Size of matrixes : A[" << N1 << "]" << N2 << ", B[" << N2 << "]" << N3 << "];" << endl;

cout << "Size of submatrixes from matrix A:\n";
for (int i = 0; i < 8; i++)
{
    if (i < NumOfMoreRows) cout << "A [" << CountRowsSubmatrixes << "]" << "[" << N2
<< "]\n";
    else cout << "A [" << CountRowsSubmatrixes + 1 << "]" << "[" << N2 << "]\n";
}
cout << "Size of submatrixes from matrix B:\n";
```

```

for (int i = 0; i < 8; i++)
{
    if (i > NumOfMoreColumns) cout << "B [" << N2 << "]" << "[" <<
CountColumnSubmatrixes + 1 << "]\n";
    else cout << "B [" << N2 << "]" << "[" << CountColumnSubmatrixes + 2 << "]\n";
...
endColumnsB[0] = 110;
MPI_Send(startRowsA, 1, MPI_INT, 1, SizeOfMatrixes, MPI_COMM_WORLD);
MPI_Send(endRowsA, 1, MPI_INT, 1, SizeOfMatrixes, MPI_COMM_WORLD);
MPI_Send(startColumnsB, 1, MPI_INT, 1, SizeOfMatrixes, MPI_COMM_WORLD);
MPI_Send(endColumnsB, 1, MPI_INT, 1, SizeOfMatrixes, MPI_COMM_WORLD);
for (int j = startRowsA[0]; j <= endRowsA[0]; j++)
    for (int k = 0; k < N2; k++)
        MPI_Send(&(A[j][k]), 1, MPI_DOUBLE, 1, Matrixes,
MPI_COMM_WORLD); //відправити відповідні частини матриці процесам
for (int j = 0; j < 248; j++)
    for (int k = startColumnsB[0]; k <= endColumnsB[0]; k++)
        MPI_Send(&(B[j][k]), 1, MPI_DOUBLE, 1, Matrixes, MPI_COMM_WORLD);

dest = 2;
startRowsA[0] = 14;
endRowsA[0] = 20;
startColumnsB[0] = 36;
endColumnsB[0] = 53;
MPI_Send(startRowsA, 1, MPI_INT, 6, SizeOfMatrixes, MPI_COMM_WORLD);
MPI_Send(endRowsA, 1, MPI_INT, 6, SizeOfMatrixes, MPI_COMM_WORLD);
MPI_Send(startColumnsB, 1, MPI_INT, 6, SizeOfMatrixes, MPI_COMM_WORLD);
MPI_Send(endColumnsB, 1, MPI_INT, 6, SizeOfMatrixes, MPI_COMM_WORLD);
for (int j = startRowsA[0]; j <= endRowsA[0]; j++)
    for (int k = 0; k < N2; k++)
        MPI_Send(&(A[j][k]), 1, MPI_DOUBLE, 6, Matrixes, MPI_COMM_WORLD...

```

Результати обчислень в кожному процесорі і остаточний результат зберігаються окремо.

Загальний вигляд і результат роботи програми:

```
C:\Program Files (x86)\Microsoft SDKs\MPI\MPITEST1\Debug>mpirun -n 8 MPITEST1.exe
Size of matrixes : A[60][248], B[248][149];
send B[248][54] - B[248][72] to 0 processor
send B[248][73] - B[248][91] to 0 processor
send B[248][36] - B[248][53] to 0 processor
send B[248][111] - B[248][129] to 0 processor
send B[248][129] - B[248][148] to 0 processor
send B[248][92] - B[248][110] to 0 processor
send B[248][18] - B[248][35] to 0 processor
send B[248][0] - B[248][17] to 0 processor
time of Multiplication in processor 3 is: 1 ticks
Size of submatrixes from matrix A:
A [7][248]
A [7][248]
A [7][248]
A [7][248]
A [8][248]
A [8][248]
A [8][248]
A [8][248]
Size of submatrixes from matrix B:
B [248][19]
B [248][19]
B [248][19]
B [248][19]
B [248][19]
B [248][19]
B [248][18]
B [248][18]
Send B[248][0] - B[248][17] to 3 processor
Send B[248][54] - B[248][72] to 3 processor
Send B[248][73] - B[248][91] to 3 processor
Send B[248][36] - B[248][53] to 3 processor
Send B[248][111] - B[248][129] to 3 processor
Send B[248][129] - B[248][148] to 3 processor
Send B[248][92] - B[248][110] to 3 processor
Send B[248][18] - B[248][35] to 3 processor
time of Multiplication in processor 0 is: 2 ticks
send B[248][92] - B[248][110] to 7 processor
send B[248][18] - B[248][35] to 7 processor
send B[248][0] - B[248][17] to 7 processor
send B[248][54] - B[248][72] to 7 processor
send B[248][73] - B[248][91] to 7 processor
send B[248][36] - B[248][53] to 7 processor
send B[248][111] - B[248][129] to 7 processor
send B[248][129] - B[248][148] to 7 processor
time of Multiplication in processor 5 is: 3 ticks
```

При тестуванні програми з використанням MPI, було використано послідовну програму для порівняння часу виконання. Для порівняння результатів було використано такі набори даних:

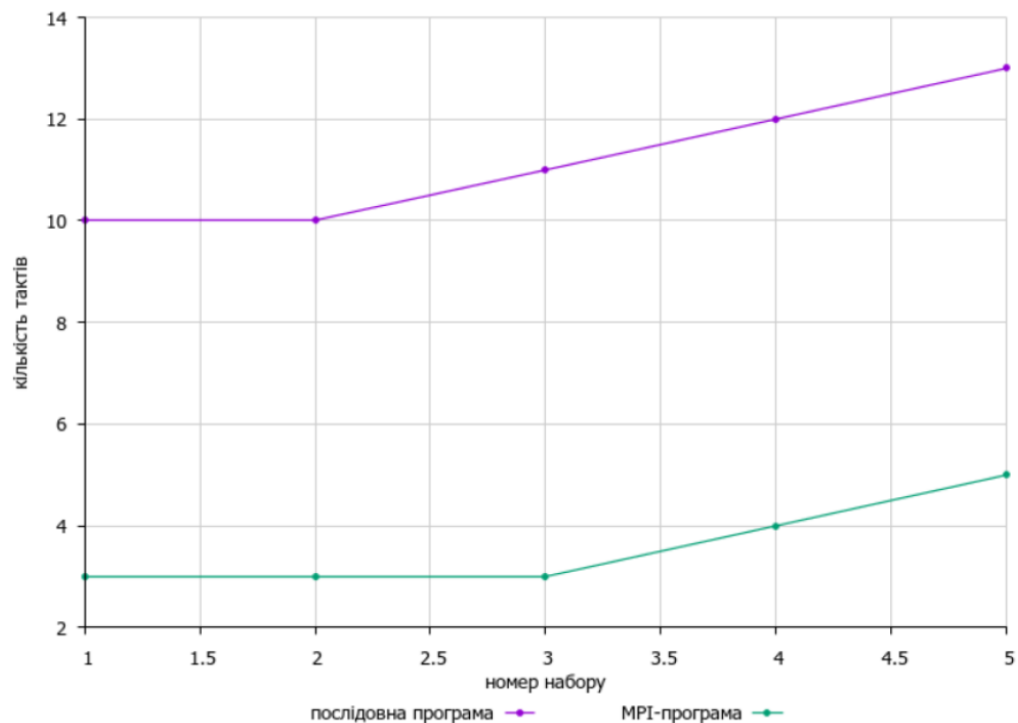
Набір даних 1: матриці А та В містять лише значення «1».

Набір даних 2: матриця А містить лише значення «1», матриця В містить лише значення «3».

Набір даних 3: матриці А та В містять довільні цілі додатні числа від 1 до 9.
 Набір даних 4: матриці А та В містять довільні цілі знакові числа від 10 до 99.
 Набір даних 5: матриці А та В містять довільні дійсні числа від 100 до 9999.

Таблиця. Результати тестувань

№ набору	Кількість тактів	
	Послідовна	MPI
1	10	3
2	10	3
3	11	3
4	12	4
5	13	5



Результати тестувань.

Висновки. Під час виконання лабораторної роботи, розроблено паралельний алгоритм перемноження матриці на матрицю на кільцевій структурі та його програмну реалізацію. Здійснено тестування програми для заданих наборів вхідних даних. Порівняння результатів послідовної та паралельної реалізації множення матриць показала ефективність останньої.