<u>Тема №3:</u> Паралельни алгоритми для задач лінійної алгебри

Питання:

1. Схеми алгоритмів задач лінійної алгебри

2. Алгоритми перемноження матриці на матрицю і їх реалізація на структурах типу: кільцева, 2D (решітка), 3D (куб)

Вправи і завдання до теми №3

1. Схеми алгоритмів задач лінійної алгебри

Паралельні алгоритми використовуються для розв'язання таких задач: множення матриці на матрицю, задача Діріхле, розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) методом Гауса і методом простої ітерації та інші. В простому варіанті сіткової задачі (задача Діріхле) крок сітки в просторі обчислень однаковий і не змінюється в процесі обчислень. При динамічній зміні кроку сітки треба було б розв'язувати задачу паралельного програмування, як перебалансування обчислювального простору між комп'ютерами, для вирівнювання обчислювального навантаження комп'ютерів.

<u>Особливості алгоритмів:</u>

1. Задачі відносяться до задач, що *розпаралелюються грубозернистими методами*.

2. Для представлення алгоритмів використовується SPMD-модель обчислень (*розпаралелення за даними).*

3. *Однорідне розподілення даних* по комп'ютерах - основа для гарного балансу часу обчислення і часу, затрачуваного на взаємодії віток паралельної програми. Такий розподіл використовується з метою забезпечення *рівності обсягів частин даних*, що розподіляються, і *відповідності нумерації частин даних*, що розподіляються, *нумерації комп'ютерів у системі*.

4. Вихідними даними розглянутих алгоритмів є *матриці, вектори і 2D (двовимірний) простір обчислень.*

5 У розглянутих алгоритмах застосовуються такі способи однорідного розподілу даних: *горизонтальними смугами, вертикальними смугами і циклічними горизонтальними смугами.*

При розподілі горизонтальними смугами матриця, вектор або *2D* простір "розрізається" на смуги по рядках. Нехай *M* - кількість рядків матриці, кількість елементів вектора або кількість рядків вузлів *2D* простору, *P* - кількість віртуальних комп'ютерів у системі, *C1 = M /P* - ціла частина від ділення, *C2 = M /P* - дробова частина від ділення. Дані розрізаються на *P* смуг. Перші *(P-C2)* смуг мають по *C1* рядків, а інші *C2* смуги мають по *C1+1* рядків.

Смуги даних розподіляються по комп'ютерах таким чином. Перша смуга поміщається в комп'ютер з номером 0, друга смуга - у комп'ютер 1, і т.д. Такий розподіл смуг по комп'ютерах враховується в паралельному алгоритмі. Розподіл вертикальними смугами аналогічний попередньому, тільки в розподілі беруть участь стовпці матриці або стовпці вузлів *2*D простору. При розподілі циклічними горизонтальними смугами дані розрізаються на кількість смуг значно більшу, від кількості комп'ютерів. І найчастіше смуга складається з одного рядка. Перша смуга завантажується в комп'ютер 0, друга - у комп'ютер 1, і т.д., потім, *P-1-a* смуга знову в комп'ютер 0, *P-a* смуга в комп'ютер 1, і т.д.

Проте, тільки однорідність розподілу даних ще недостатня для ефективного виконання алгоритму. Ефективність алгоритмів залежить і від способу розподілу даних. Різний спосіб представлення даних веде, відповідно, і до різної організації алгоритмів, що обробляють ці дані.

Точні значення ефективності конкретного паралельного алгоритму можуть бути визначені на конкретній обчислювальній системі на деякому наборі даних. Тобто, ефективність паралельних алгоритмів залежить, по-перше, від *обчислювальної системи*, на якій виконується задача, а, по-друге, від *структури самих алгоритмів*. Вона визначається як відношення часу реалізації паралельного алгоритму задачі до часу реалізації послідовного алгоритму цієї ж задачі. Ефективність можна вимірювати і співвідношенням між часом, витраченим на обмін даними між процесами, і загальним часом обчислень. Зауважимо, що ефективність алгоритмів, що використовують глобальний обмін даними, знижується з збільшенням кількості паралельних віток. Тобто з збільшенням кількості комп'ютерів у системі, швидкість виконання

глобальної операції обміну буде падати. До таких задач можна віднести, наприклад, задачу розв'язання СЛАР ітераційними методами. Ефективність алгоритмів, у яких обмін даними здійснюється тільки локально, буде незмінною з збільшенням кількості паралельних віток.

2. Алгоритми перемноження матриці на матрицю і їх реалізація на структурах типу: кільцева, 2D (решітка), 3D (куб)

Множення матриці на вектор і матриці на матрицю є базовими макрооперaціями 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 1- Перемноження матриці на матрицю на кільцевій структурі

Задано дві вихідні матриці A і B. Обчислюється добуток **C** = **A** x **B**, де **A** - матриця n_1 x n_2, і **B** - матриця n_2 x n_3. Матриця результатів **C** має розмір n_1 x 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.

Реалізація алгоритму виконується на кільці з p_1 комп'ютерів. Матриці розрізані як показане на рис. 7.1: матриця **A** розрізана на p_1 горизонтальних смуг, матриця **B** розрізана на p_1 вертикальних смуг, і матриця результату **C** розрізана на p_1 смуги. Тут передбачається, що в пам'ять кожного комп'ютера завантажується і може знаходитися тільки одна смуга матриці **A** і одна смуга матриці **B**.

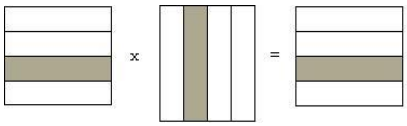


Рис. 3.1 *Розрізування даних для паралельного алгоритму добутку двох матриць при обчисленні на кільці комп'ютерів. Виділені смуги розташовані в одному комп'ютері.*

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**). Кожний зсув смуг уздовж кільця і відповідна операція множення наведена на рис.3.2 у виді окремого кроку. На кожному з таких кроків обчислюється тільки частина смуги. Процес i обчислює на j-м кроці добуток i-й 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. На рис. 3.2 смуги матриці **A** і **B** пронумеровані.

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 **B** зсуваються уздовж кільця комп'ютерів. І т.д.

5. Обчислювальний крок p1-1. Кожен процес обчислює одну підматрицю добутку. Вертикальні смуги матриці **B** зсуваються уздовж кільця комп'ютерів.

6. Обчислювальний крок p1. Кожен процес обчислює одну підматрицю добутку. Вертикальні смуги матриці **B** зсуваються уздовж кільця комп'ютерів.

7. Матриця **C** збирається в нульовому комп'ютері.

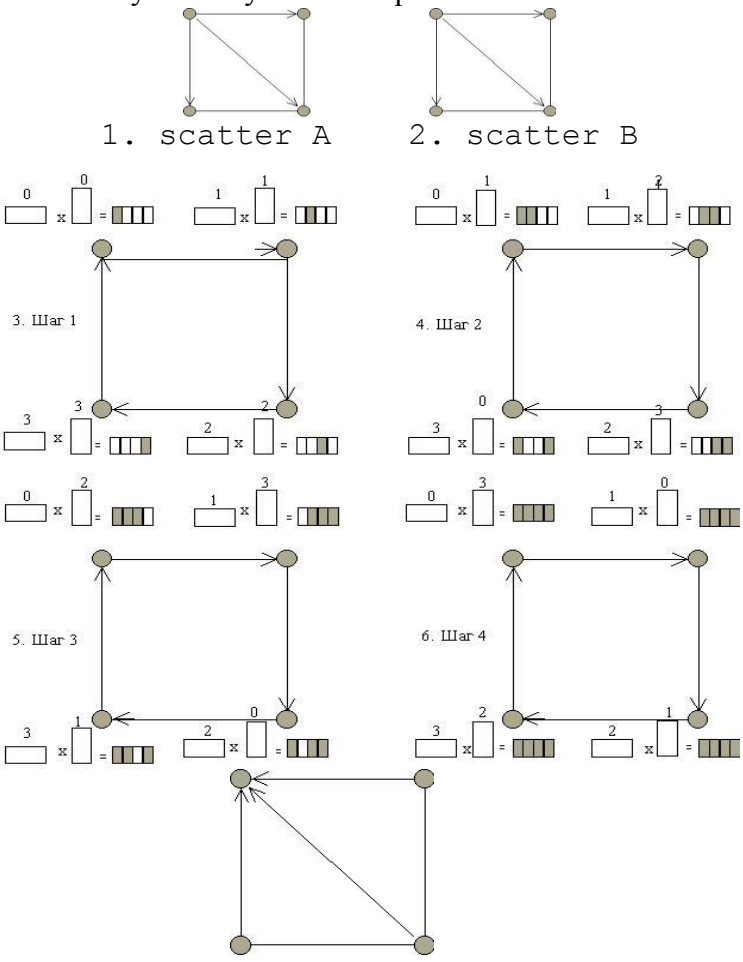


Рис. 3.2 *Стадії обчислень добутку матриць у кільці комп'ютерів.*

Якщо "прокручувати" вертикальні смуги матриці **B**, то матриця **C** буде розподілена горизонтальними смугами, а якщо "прокручувати" горизонтальні смуги матриці **A**, то матриця **C** буде розподілена вертикальними смугами.

Алгоритм характерний тим, що після кожного кроку обчислень здійснюється обмін даними. Нехай t_u, t_s, *і* t_p час операцій, відповідно, множення, додавання і пересилання одного числа в сусідній комп'ютер. Тоді сумарний час операцій множень дорівнює:

$$U = (n_1*n_2)*(n_3*n_2)*t_u,$$

сумарний час операцій додавань дорівнює:

$$S = (n_1*n_2)*(n_3*(n_2-1))*t_s,$$

сумарний час операцій пересилань даних по всіх комп'ютерах дорівнює:

$$P = (n_3*n_2)*(p_1-1)*t_p.$$

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. У наступних двох алгоритмах використовується цей підхід.

Алгоритм 2 - Перемноження матриці на матрицю на2D решітці

Обчислюється добуток ***C*** = ***A*** *x* ***B,*** де **A** - матриця n_1 x n_2, і **B** - матриця n_2 x n_3. Матриця результатів **C** має розмір n_1 x n_3. Вихідні матриці спочатку доступні на нульовому процесі, і матриця результатів повертається в нульовий процес.

Паралельне виконання алгоритму здійснюється на двовимірній (2D) решітці комп'ютерів розміром p_1 x p_2. Матриці розрізані, як показано на рис. 3.3: матриця **A** розрізана на p_1 горизонтальних смуг, матриця **B** розрізана на p_2 вертикальних смуг, і матриця результату **C** розрізана на p_1 x p_2 підматриці (або субматриці).

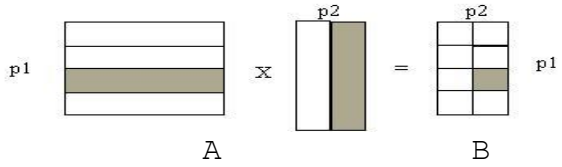


Рис. 3.3 *Розрізування даних для паралельного алгоритму добутку двох матриць при обчисленні на 2D решітці комп'ютерів. Виділені дані розташовані в одному комп'ютері.*

Кожен комп'ютер *(i,j)* обчислює добуток *i*-й горизонтальної смуги матриці **A** і *j*-й вертикальної смуги матриці **B**, добуток отримується у підматриці (*i*,*j*) матриці **C**.

Послідовність стадій обчислення наведена на рис.3.4:

*1. Матриця **A** розподіляється по горизонтальних смугах уздовж координати (x,0).*

*2. Матриця **B** розподіляється по вертикальних смугах уздовж координати (0,y).*

*3. Смуги **A** поширюються у вимірі y.*

*4. Смуги **B** поширюються у вимірі x.*

5. Кожен процес обчислює одну підматрицю добутку.

*7. Матриця **C** збирається з (x,y) площини.*

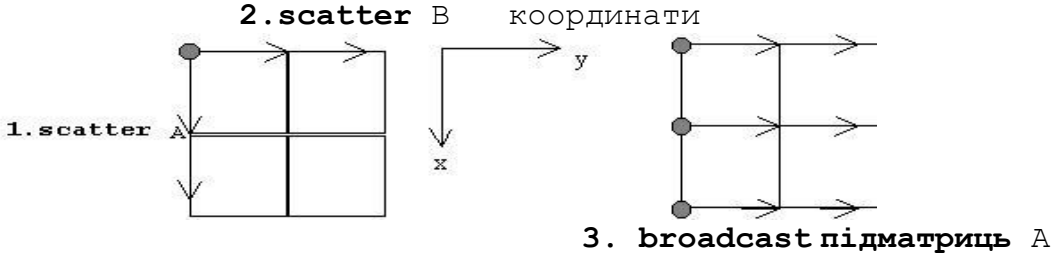
Здійснювати пересилання між комп'ютерами під час обчислень не потрібно, тому що всі смуги матриці **A** перетинаються з усіма смугами матриці B у пам'яті комп'ютерів системи.

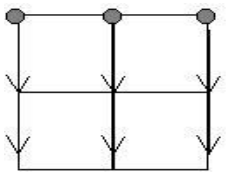
Цей алгоритм ефективніший від попереднього, тому що непродуктивний час пересилань даних здійснюється тільки при завантаженні даних у пам'ять комп'ютерів і при їхньому вивантаженні, і обміни даними в процесі обчислень відсутні. Оскільки час обмінів рівний нулеві, а час завантаження і вивантаження тут не враховується, то загальний час обчислень дорівнює:

$$T = (U+S)/(p_1*p_2)$$

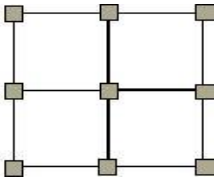
А відношення часу "обчислень без обмінів" до загального часу обчислень є величина:

$$K = (U+S)/(U+S) = 1.$$

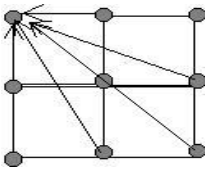




4. broadcast підматриць B



5. обчислення добутків (підматриць в C**)**



6. *збір результатів в* C

Рис. 3.4 *Стадії обчислення добутку матриць у 2D паралельному алгоритмі.*

Алгоритм 3 - *Перемноження матриці на просторовій сітці комп'ютерів*

Для великих матриць, час обчислення добутків може бути зменшений шляхом застосуванням алгоритму, що здійснює обчислення на 3-мірній (просторовій) сітці комп'ютерів.

У приведеному нижче алгоритмі відображаються основні дані обсягом $n_1 x n_2 + n_2 x n_3 + n_1 x n_3$ на об'ємну сітку комп'ютерів розміром $p_1 x p_2 x p_3$. Матриці розрізані, як показано на рис. 3.5: Матриця **A** розрізана на $p_1 x p_2$ субматриці, матриця ***B*** розрізана на $p_2 x p_3$ субматриці, і матриця **C** розрізана на $p_1 x p_3$ субматриці. Комп'ютер (*i,j,k*) обчислює добуток субматриці (*i,j*) матриці **A** і субматриці (*j,k*) матриці **B**. Субматриця (*i,k*) матриці **C** виходить підсумовуванням проміжних результатів, обчислених у комп'ютерах *(i,j,k), j = 0,...,p2-1*.

Послідовність стадій обчислення наведена на рис. 3.6.

*1. Субматриці **A** розподіляються в (x,y,0) площині.*
*2. Субматриці **B** розподіляються в (0,y,z) площині.*
*3. Субматриці **A** поширюються у вимірі z.*
*4. Субматриці **B** поширюються у вимірі x.*
5. Кожен процес обчислює одну субматрицю.
6. Проміжні результати редукується у вимірі y.
*7. Матриця **C** збирається з (x,0,z) площини.*

Алгоритм подібний до попереднього, але додатково розрізаються ще смуги матриць, і ці розрізані смуги розподіляються в третьому вимірі *y*. У даному випадку в кожному комп'ютері будуть перемножуватися тільки частини векторів рядків матриці **A** і частини стовпців матриці ***B***. В результаті буде тільки часткова сума для кожного елемента результуючої матриці **C**. Операція підсумовування уздовж координати *y* цих отриманих часткових сум для результуючих елементів і завершує обчислення матриці ***C***.

Загальний час обчислень у цьому алгоритмі дорівнює:

$$T = (U+S)/(p_1*p_2*p_3)$$

А відношення часу "обчислень без обмінів" до загального часу обчислень є величина:

$$K = (U+S)/(U+S)=1.$$

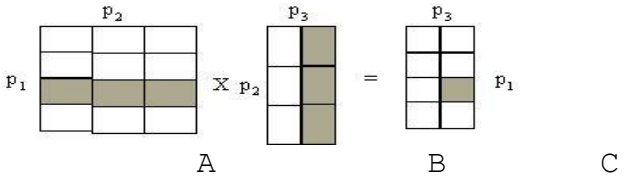
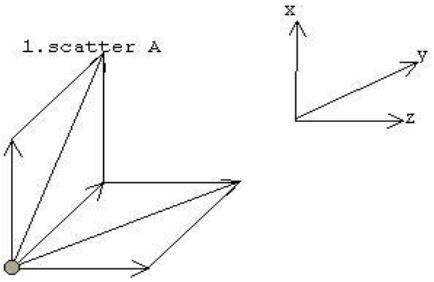
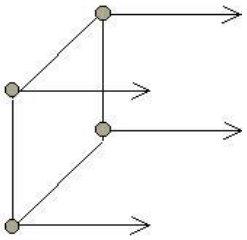


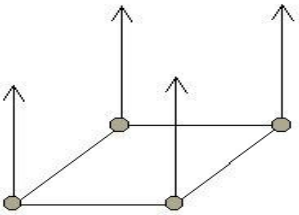
Рис. 3.5 *Розрізування даних для паралельного алгоритму добутку двох матриць при обчисленні на просторовій сітці комп'ютерів.*



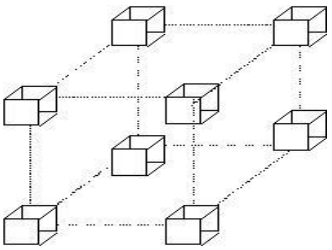
2. scatter B



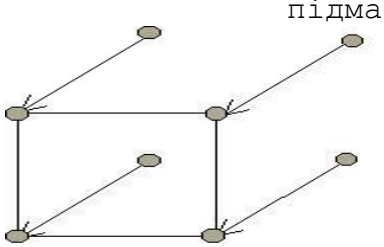
3. broadcast підматриць з A



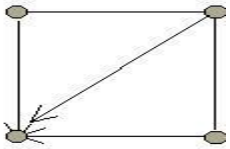
4. broadcast підматриць з У



5. Обчислення добутків підматриць у С)



6. reduce (підсумовування)добутків



7. gather C (збір результатів)

Рис. 7.6. *Стадії обчислень у 3D паралельному алгоритмі добутку матриць.*

Вправи і завдання до теми №3

1. Розробіть паралельний алгоритм обчислення величини $C = \sum_{i=1}^{N} A_k * B_k$, де A і B – одновимірні масиви.
2. Дані матриці **A** і **B**. Розробіть алгоритм обчислення матриці **C** = **A*****B** – **B*****C**.
3. Дана матриця **A** і вектори *a* і *b*. Розробіть алгоритм обчислення матриці **C** = *a* - **A*** *b*.
4. Чи можлива різна кількість смуг, на які діляться матриці **A** і **B** при обчисленні добутку двох матриць на кільцевій структурі?
5. Наведіть переваги і недоліки організації перемноження двох матриць на структурах, які описані в даній темі?

Наведіть переваги і недоліки організації початкового завантаження при перемноженні двох матриць на структурах, які описані в даній темі?