Міністерство Освіти і Науки України

Національний Університет "Львівська Політехніка"

Кафедра ЕОМ

ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНОЇ ДЕКОМПОЗИЦІЇ ДЛЯ РОЗВ'ЯЗКУ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ

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 2021 року

Львів – 2021

МЕТА РОБОТИ. Вивчити методи декомпозицій задач. Набути навиків розв'язування задач з використанням функціональної декомпозиції.

ТЕОРЕТИЧНІ ВІДОМОСТІ.

Найважливішим та найважчим етапом при створенні програми є розробка алгоритму, особливо, якщо мова йде про паралельний алгоритм. Процес створення паралельного алгоритму можна розбити на чотири кроки.

1. ***Декомпозиція***. На цьому етапі вихідна задача аналізується, оцінюється можливість її розпаралелювання. Іноді виграш від розпаралелення може бути незначним, а трудоємкість розробки паралельної програми велика. В цьому випадку перший крок розробки алгоритму виявляється і останнім. Якщо ж ситуація відмінна від описаної, то задача та пов'язані з нею дані розділяються на дрібніші частини – підзадачі і фрагменти структур даних. Особливості архітектури конкретної обчислювальної системи на цьому етапі можуть не враховуватися.

2. ***Проектування комунікацій***(обміну даними) між задачами. На цьому етапі визначаються зв'язки, необхідні для пересилання вхідних даних, проміжних результатів виконання підзадач, а також комунікації, що необхідні для керування роботою під задач. Обираються методи та алгоритми комунікацій.

3.*Укрупнення*. Підзадачі можуть об'єднуватися у більші блоки, якщо це дозволяє підвищити ефективність алгоритму і знизити трудоємкість розробки. Основними критеріями на даному кроці є ефективність алгоритму (в першу чергу-продуктивність) та трудоємкість його реалізації.

4. ***Планування обчислень***. На цьому кроці виконується розподіл під задач між процесорами. Основний критерій вибору способу розміщення під задач – ефективне використання процесорів з мінімальними затратами часу на обмін даними.

Декомпозиція (сегментування).

На цьому етапі визначається степінь можливого розпаралелення задачі. Іноді декомпозиція поставленої задачі природним чином випливає з самої задачі тому є очевидною, іноді – ні. Чим менший розмір підзадач, отриманих в результаті декомпозиції, чим більша їх кількість, тим гнучкішим виявляється паралельний алгоритм і тим легше забезпечити рівномірне завантаження процесорів обчислювальної системи. Надалі, можливо доведеться укрупнити алгоритм, оскільки слід прийняти до уваги інтенсивність обміну даними та інші фактори. Сегментувати можна як обчислювальний алгоритм, так і дані. Застосовуються різні варіанти декомпозиції.

В методі декомпозиції даних спочатку сегментуються дані, а потім алгоритм їх обробки. Дані розбиваються на сегменти приблизно однакового розміру. З фрагментами даних пов'язуються операції їх обробки, з яких і формуються підзадачі. Визначаються необхідні правила обміну даними. Перекриття частин, на які розбивається задача, повинне бути мінімальним. Це дозволяє уникнути дублювання обчислень. Схема розбиття може в подальшому уточнюватися. Іноді, якщо це необхідно для зменшення кількості обмінів, допускається збільшення степені перекриття фрагментів задачі.

При декомпозиції даних спочатку аналізуються структури даних найбільшого розміру, або ті, до яких найчастіше відбувається звертання. На різних стадіях розрахунків можуть використовуватися різні структури даних, тому можуть бути необхідними динамічні, тобто такі, що міняються в часі схеми декомпозиції цих структур.

В методі функціональної декомпозиції спочатку сегментується обчислювальний алгоритм, а потім під цю схему підбирається схема декомпозиції даних. Успіх у цьому випадку не завжди гарантовано, оскільки схема може вимагати багатьох додаткових пересилань даних. Цей метод декомпозиції може виявитися корисним у випадку, якщо немає структур даних, які б могли бути розпаралелені очевидним чином. Функціональна декомпозиція відіграє важливу роль і як метод структурування програм. Вона може виявитися корисною при моделюванні складних систем, що складаються з множини простих підсистем, зв'язаних між собою набором інтерфейсів.

Розмір блоків, з яких складається паралельна програма, може бути різним. В залежності від розміру блоків, алгоритм може мати різну "зернистість". Її виміром в найпростішому випадку є кількість операцій у блоці. Виділяють три степені зернистості: дрібнозернистий, середньоблоковий та великоблоковий. Зернистість пов'язана з рівнем паралелізму.

Паралелізм на рівні команд найбільш дрібнозернистий. Його масштаб менше ніж 20 команд на блок. Кількість підзадач, що паралельно виконуються – від однієї до кількох тисяч, при чому середній масштаб паралелізму складає біля п'яти команд на блок.

Наступний рівень - паралелізм на рівні циклів. Переважно, цикл містить не більше 500 команд. Якщо ітерації циклу незалежні, вони можуть виконуватися, наприклад за допомогою конвеєра або на векторному процесорі. Це також дрібнозернистий паралелізм.

Паралелізм на рівні підзадач – середньоблоковий. Розмір блоку – до 2000 команд. Виконання такого паралелізму реалізувати важче, оскільки слід враховувати можливі міжпроцедурні залежності. Вимоги до комунікацій менші, ніж у випадку паралелізму на рівні команд.

Паралелізм на рівні програм (задач) – великоблоковий. Він означає виконання незалежних програм на паралельному комп'ютері. Великоблоковий паралелізм повинен підтримуватися операційною системою.

Обмін даними через спільні/розподілені змінні використовується на рівнях дрібнозернистого і середньоблокового паралелізму, а на великоблоковому – засобами передачі повідомлень.

Досягнути ефективності роботи паралельної програми можна, якщо збалансувати зернистість алгоритму і затрати часу на обмін даними.

Частини програми можуть виконуватися паралельно, лише якщо вони незалежні.

Незалежність за даними полягає в тому, що дані, які обробляються однією частиною програми не модифікуються іншою її частиною.

Незалежність за керуванням полягає в тому, що послідовність виконання частин програми може бути визначена лише під час виконання

програми(наявність залежності по виконанню наперед визначає порядок виконання).

Незалежність за ресурсами забезпечується достатньою кількістю комп'ютерних ресурсів(об'єм пам'яті, кількість функціональних вузлів та ін.).

Залежність за виводом виникає, якщо дві підзадачі виконують запис в одну і ту ж змінну. А *залежність за вводом/виводом*, якщо оператори вводу/виводу двох чи декількох під задач звертаються до одного файлу(чи змінної).

Дві підзадачі можуть виконуватися паралельно, якщо вони незалежні за даними, за керуванням і за операціями вводу виводу.

ЗАВДАННЯ.

1.Використовуючи метод функціональної декомпозиції, розробити алгоритм обчислення запропонованого матрично-векторного виразу, який би враховував можливість паралельного виконання і був оптимальним з точки зору часових затрат .

2. На основі створеного алгоритму написати програму використовуючи засоби MPI яка дозволяє обчислити вираз та ілюструє проведену декомпозицію для оптимальної кількості процесорів.

3, Дослідити ефективність паралельного обчислення за критерієм часу.

ПОРЯДОК ВИКОНАННЯ РОБОТИ

1) **Проаналізувати, наведені нижче, правила обрахунку елементів виразу;**
2) **Провести декомпозицію задачі, виходячи з можливості паралельного виконання окремих під задач для заданої кількості процесорів;**
3) **Об'єднати отримані проміжні результати і обрахувати кінцевий вираз;**
4) **Написати послідовну програму обчислення виразу.**
5) **Написати і продемонструвати програму обчислення виразу засобами MPI;**
6) **Визначити паралелізм якого рівня присутній в алгоритмі та зробити висновки щодо залежностей даних, керування, ресурсів, вводу/виводу;**
7) **Дослідити час обчислення виразу в залежності від розмірності вхідних даних**
8) **Скласти звіт про виконану роботу.**

ЗМІСТ ЗВІТУ

1.Тема, мета, аналіз завдання(згідно варіанту).
2. Схема декомпозиції задачі та коментарі до неї.
3. Текст програми та результат її роботи на довільному наборі вхідних даних, для розмірності n>3.
4. Дослідження ефективності паралельної програми у порівнянні з послідовною за часом виконання.
5. Висновки.

Вагомі зауваження.

а) Окрім безпосередніх обчислень, програма повинна мати інтерфейс користувача, який забезпечує:

1) ввід(з клавіатури) розмірності даних (n);
2) можливість вибору–ввід даних(тобто елементів матриці та векторів) з клавіатури чи генерування їх випадковим чином;
3) вивід на екран (або у файл) проміжних результатів за потребою користувача;
4) обов'язковий вивід остаточних результатів на екран і у файл у зрозумілому вигляді.

б) Всі вхідні дані є цілими числами, більшими за нуль.

в) Необхідно знайти такі коефіцієнт нормалізації результатів(тобто пониження чи підвищення їх порядку).

Правила знаходження елементів виразу.

1).Задати* квадратну матрицю A порядку n. Отримати вектор(стовпець) $y_1=A*b$, де b – вектор-стовпець, елементи якого обраховуються за формулою, згідно варіанту.

2).Задати квадратну матрицю A_1 порядку n та вектори-стовпці b_1 та c_1 з n елементами кожен. Отримати вектор y_2 згідно формули, що задається варіантом.

3).Задати квадратні матриці A_2 та B_2 порядку n. Отримати матрицю Y_3, яка залежить від A_2, B_2 та додатково визначеної матриці C_2, елементи якої знаходяться за формулою, вказаною варіантом.

ЛІТЕРАТУРА

С.Немногин О.Стесик "Параллельное программирование для многопроцессорних систем" Петербург "БХВ-Петербург", 2002

Томас Бройнль "Паралельне програмування. Початковий курс"Київ "Вища школа, 1997

ВАРІАНТИ ЗАВДАНЬ.

При чому:

y' означає операцію транспонування; i,j=1…n (n – вхідна розмірність).

1	$x=Y_3^2 y_2+Y_3(y_1+y_2)$ стовпець		
	$b_i=1/(i^2+2)$ для парних i $b_i=1/i$ для непарних i	$A_1(b_1+c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i+2j)$
2	$x=Y_3^2 y_2+Y_3*(y_1+y_2)+y_1y_2'y_2+Y_3^3 y_1$ стовпець		
	$b_i=1/(i^2+2+i)$ для парних i $b_i=1/i$ для непарних i	$A_1(b_1+2c_1)$	$A_2(C_2-B_2)$ $C_{ij}=1/(i+j)$
3	$x=Y_3y_2y_2'+Y_3^3-Y_3+y_2y_1'+Y_3^2y_1y_1'$ матриця		
	$b_i=3/(i^2+3)$ для парних i $b_i=3/i$ для непарних i	$A_1(3b_1+c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i+j)^2$
4	$x=(Y_3y_1+Y_3^2y_2+y_2'y_1Y_3y_1)'*Y_3$ рядок		
	$b_i=4/(i^3+3)$	$A_1(b_1+4c_1)$	$A_2(B_2+C_2)$ $C_{ij}=1/(i+j^2)$
5	$x=(y_1'Y_3*y_1+y_2')*(Y_3*y_2+y_1+y_1y_2'Y_3^2y_2)$ число		
	$b_i=5i^3$	$A_1(5b_1-c_1)$	$A_2(B_2+10C_2)$ $C_{ij}=1/(i^2+j)$
6	$x=y_2'y_1Y_3^2+y_1'y_2Y_3^2+y_2'Y_3y_1Y_3+Y_3$ матриця		
	$b_i=6/i^2$	$A_1(6b_1-c_1)$	$A_2(10B_2+C_2)$ $C_{ij}=1/(i+j)^3$
7	$x=y_2'*(y_1y_2'+Y_3^3+y_1'y_2Y_3)*y_1$ число		
	$b_i=7i$	$A_1(b_1+c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i^3+j^2)$
8	$(y_1'Y_3y_1y_2y_2'+Y_{.3}^2+y_1y_2')*(y_2'Y_3y_2y_1+y_1)$ стовпець		
	$b_i=8/i$	$A_1(2b_1+3c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i+j+2)$
9	$x=y_2'(Y_3+y_1'y_1Y_3^2+y_2y_2')+y_1'(y_1'y_1Y_3+Y_3^3)$ рядок		
	$b_i=9i$	$A_1(b_1-c_1)$	$A_2(B_2+C_2)$ $C_{ij}=1/(i+j)$
10	$x=\left(y_2'(y_1'y_1Y_3)+y_1'Y_3^3+y_2'Y_3\right)*(Y_3y_1+y_2'y_2y_1)$ число		
	$b_i=10/(i^2+1)$	$A_1(b_1+c_1)$	$A_2(C_2+2B_2)$ $C_{ij}=1/(i+2j)$
11	$x=y_2y_1'+y_2'y_2Y_3+y_1'Y_3^2*y_2+Y_3+Y_3y_1y_1'Y_3$ матриця		
	$b_i=11i^2$ для парних i $b_i=11/i$ для непарних i	$A_1(b_1-2c_1)$	$A_2(B_2-2C_2)$ $C_{ij}=1/(i^2+j)$
12	$x=y_2'+\left((y_1'(Y_3^2y_1+y_2)Y_3+y_1y_2')y_2\right)'$ рядок		
	$b_i=i^2/12$ для парних i $b_i=i$ для непарних i	$A_1(12b_1-c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i+j^2)$

ВАРІАНТИ ЗАВДАНЬ.

При чому:

y' означає операцію транспонування; i,j=1…n (n – вхідна розмірність).

13	$x = Y_3 y_1 + Y_3 y_2 + (13 y_1 + y_2) + y_1' y_2 y_1$ стовпець		
	b_i=13/(i+2) для парних i b_i=13/i^2 для непарних i	$A_1 b_1$-c_1	A_2-$B_2 C_2$ C_{ij}=13/(i^2+j^2)
14	$x = Y_3^3 y_1 y_1' + y_2 y_1' + y_2' Y_3\ y_1 Y_3$ матриця		
	b_i=14/(i^3) для парних i b_i=1/(i+14) для непарних i	$A_1(14b_1+14c_1)$	$A_2 C_2$-B_2 C_{ij}=14/(i+j^4)
15	$x = y_2' + y_1' + (Y_3^2 y_1)' + y_2' y_2 y_1' Y_3$ рядок		
	b_i=i для парних i b_i=15/i для непарних i	$15A_1 b_1+c_1$	$A_2 C_2+B_2$ C_{ij}=15/(i^2+j)
16	$x = (Y_3 y_2 + y_2)'(Y_3^2 y_2 + Y_3 y_1) + y_1' y_2$ число		
	b_i=16/(i^3)	$A_1(b_1+16c_1)$	$A_2(B_2+16C_2)$ C_{ij}=16/(i+j)^2
17	$x = Y_3^3 + y_2' y_1 Y_3 + y_1 y_2' + Y_3 y_2 y_1'$ матриця		
	b_i=17/i^2	$A_1(17b_1+c_1)$	$A_2(B_2+C_2)$ C_{ij}=17/(2i+j)
18	$x = y_2 y_1' + y_1 y_2' + Y_3^2 + Y_3 - Y_3 y_2 y_1'$ матриця		
	b_i=18/(i+18)^2	$A_1(b_1-c_1)$	$A_2 B_2$-$A_2 C_2$ C_{ij}=18/(i+2j)
19	$x = y_1' * (y_2 y_1' + Y_3^2 + y_2' y_1 Y_3) * y_2$ число		
	b_i=19/(i^2+1) для парних i b_i=19 для непарних i	$A_1(b_1+19c_1)$	$A_2(B_2+C_2)$ C_{ij}=19/(i+2j)^3
20	$(y_1'(y_2' y_2 Y_3) + y_2' Y_{.3}^3 + y_2' Y_3) * Y_3 y_1$ число		
	b_i=20/(i^3+20)	$A_1(20b_1-c_1)$	$A_2 C_2$- B_2 C_{ij}=20/(i^3- j^3+2)
21	$x = (Y_3^2 y_2 + Y_3 y_1 + y_2' y_2 Y_3 y_1)' * Y_3^2)$ рядок		
	b_i=21/i^4	$A_1(b_1+20c_1)$	$A_2 B_2$-C_2 C_{ij}=21/(i^2+2j)
22	$x = y_2 + y_1 + (y_1'(Y_3 y_1 + y_2) Y_3 + y_2 y_2')y_2$ стовпець		
	b_i=22i для парних i b_i=22 для непарних i	$A_1(b_1-21c_1)$	$A_2(B_2-C_2)$ C_{ij}=22/(i+j)
23	$x = Y_3 y_1 y_2' + y_2 y_1' + Y_3^2 - Y_3 y_1 y_1' Y_3$ матриця		
	b_i=23/i для парних i b_i=23/i^2 для непарних i	$A_1(b_1+c_1)$	$A_2(23B_2+C_2)$ C_{ij}=23/(3i+j)^2

ВАРІАНТИ ЗАВДАНЬ.

При чому:

y' означає операцію транспонування; i,j=1…n (n – вхідна розмірність).

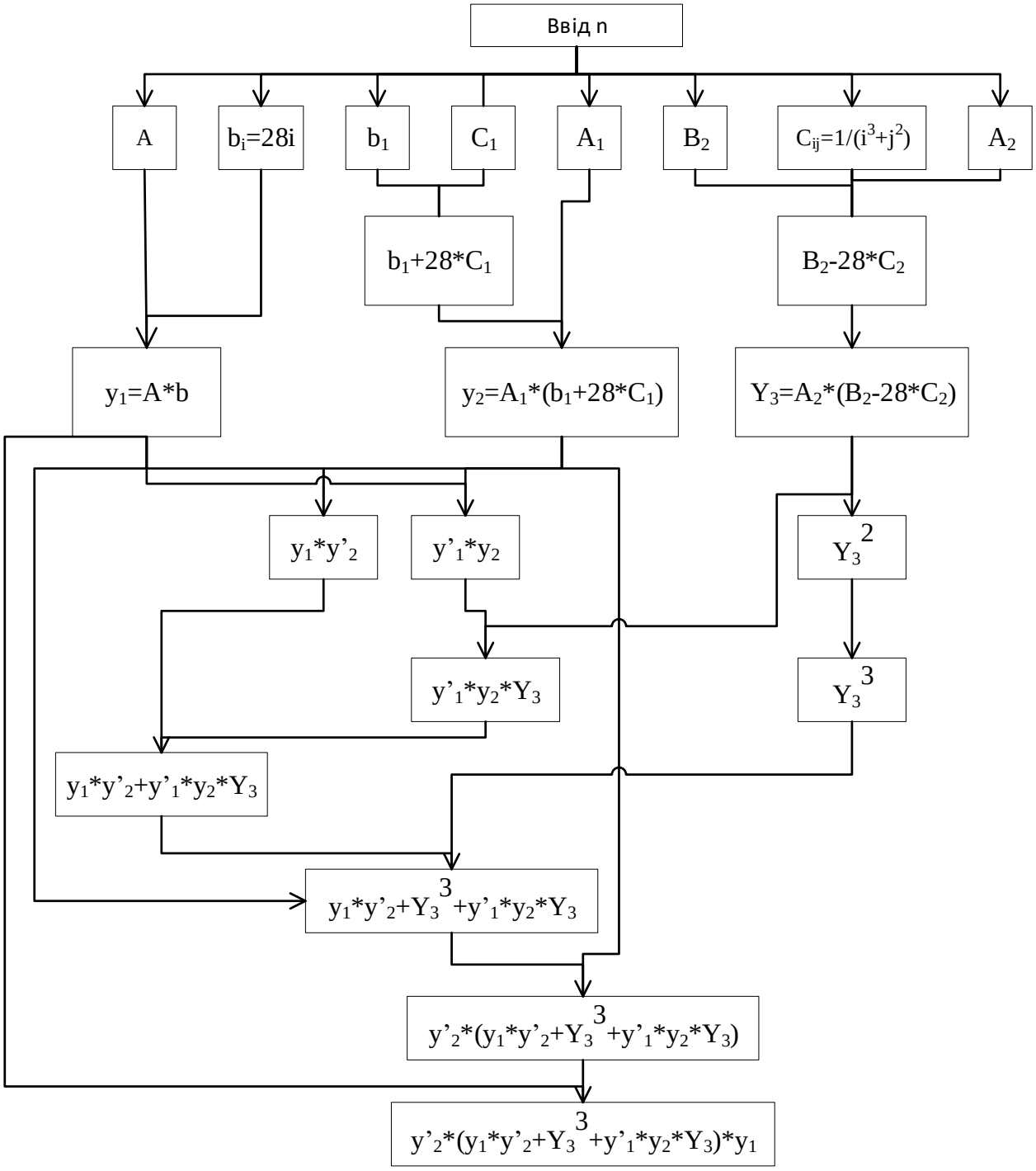
24	$x=(y_2'Y_3^2y_1+y_1')(Y_3y_1+y_2)$ число		
	$b_i=24/(i^2+4)$ для парних i $b_i=24$ для непарних i	$A_1(b_1-24c_1)$	$A_2(B_2+24C_2)$ $C_{ij}=24/(i+3j^2)$
25	$x=(y_1'Y_3\ y_2+y_2')(Y_3^3y_1+y_1+y_1y_2'Y_3y_1)$ число		
	$b_i=25$ для парних i $b_i=25/i^3$ для непарних i	$A_1(b_1+c_1)$	$A_2(B_2+C_2)$ $C_{ij}=25/(i+j)^3$
26	$x=(y_1'Y_3*y_1+y_2')*(Y_3*y_2+y_1+y_1y_2'Y_3^2y_2)$ число		
	$b_i=26i^3$	$A_1(26b_1-c_1)$	$A_2(B_2+26C_2)$ $C_{ij}=1/(i^2+j)$
27	$x=y_2'y_1Y_3^2+y_1'y_2Y_3^2+y_2'Y_3y_1Y_3+Y_3$ матриця		
	$b_i=27/i^2$	$A_1(27b_1-26c_1)$	$A_2(27B_2+C_2)$ $C_{ij}=1/(i+j)^3$
28	$x=y_2'*(y_1y_2'+Y_3^3+y_1'y_2Y_3)*y_1$ число		
	$b_i=28/i$	$A_1(b_1+28c_1)$	$A_2(B_2-28C_2)$ $C_{ij}=1/(i^3+j^2)$
29	$(y_1'Y_3y_1y_2y_2'+Y_{.3}^2+y_1y_2')*(y_2'Y_3y_2y_1+y_1)$ стовпець		
	$b_i=29/i$	$A_1(29b_1+3c_1)$	$A_2(29B_2-C_2)$ $C_{ij}=1/(i+j+2)$
30	$x=y_2'(Y_3+y_1'y_1Y_3^2+y_2y_2')+y_1'(y_1'y_1Y_3+Y_3^3)$ рядок		
	$b_i=30/i$ для парних i $b_i=30/i^2$ для непарних i	$A_1(b_1-c_1)$	$A_2(B_2+C_2)$ $C_{ij}=30/(i+j)$

Приклад виконання роботи.

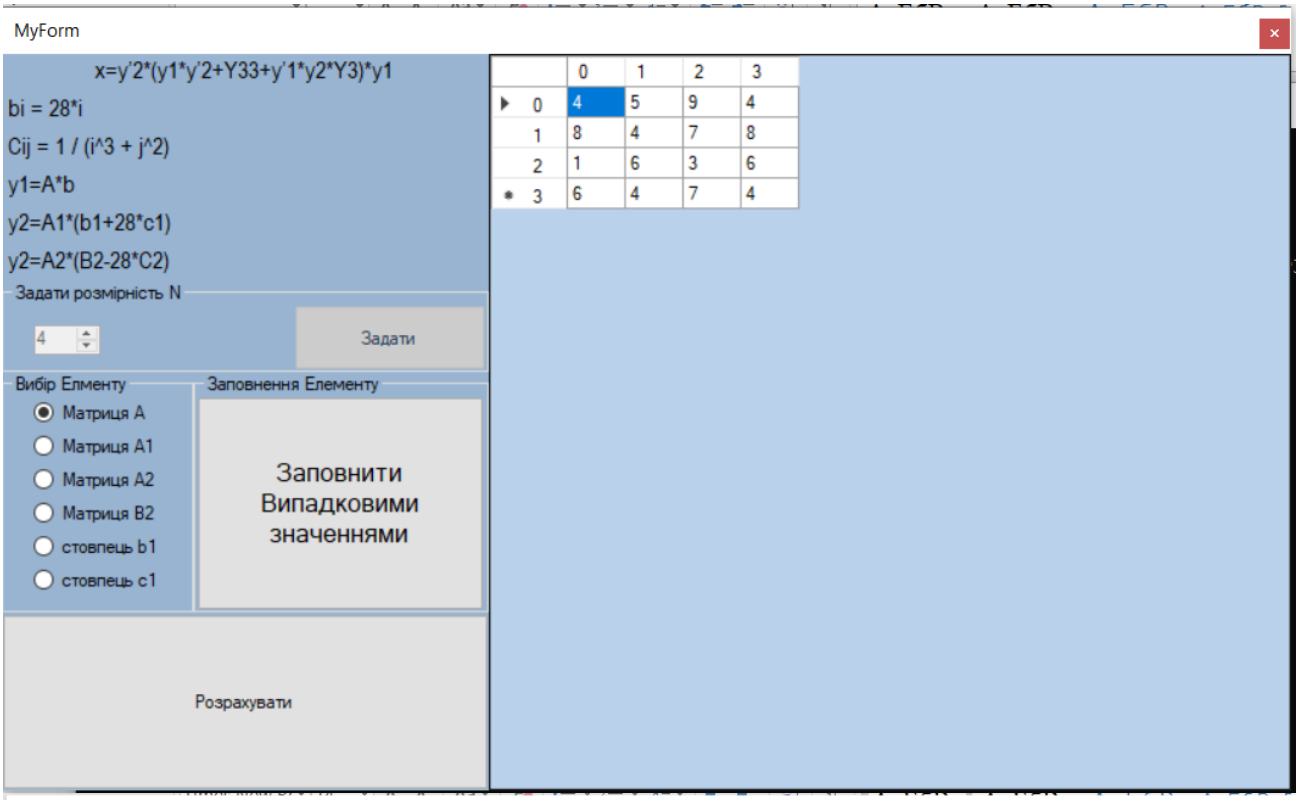
Варіант завдання:

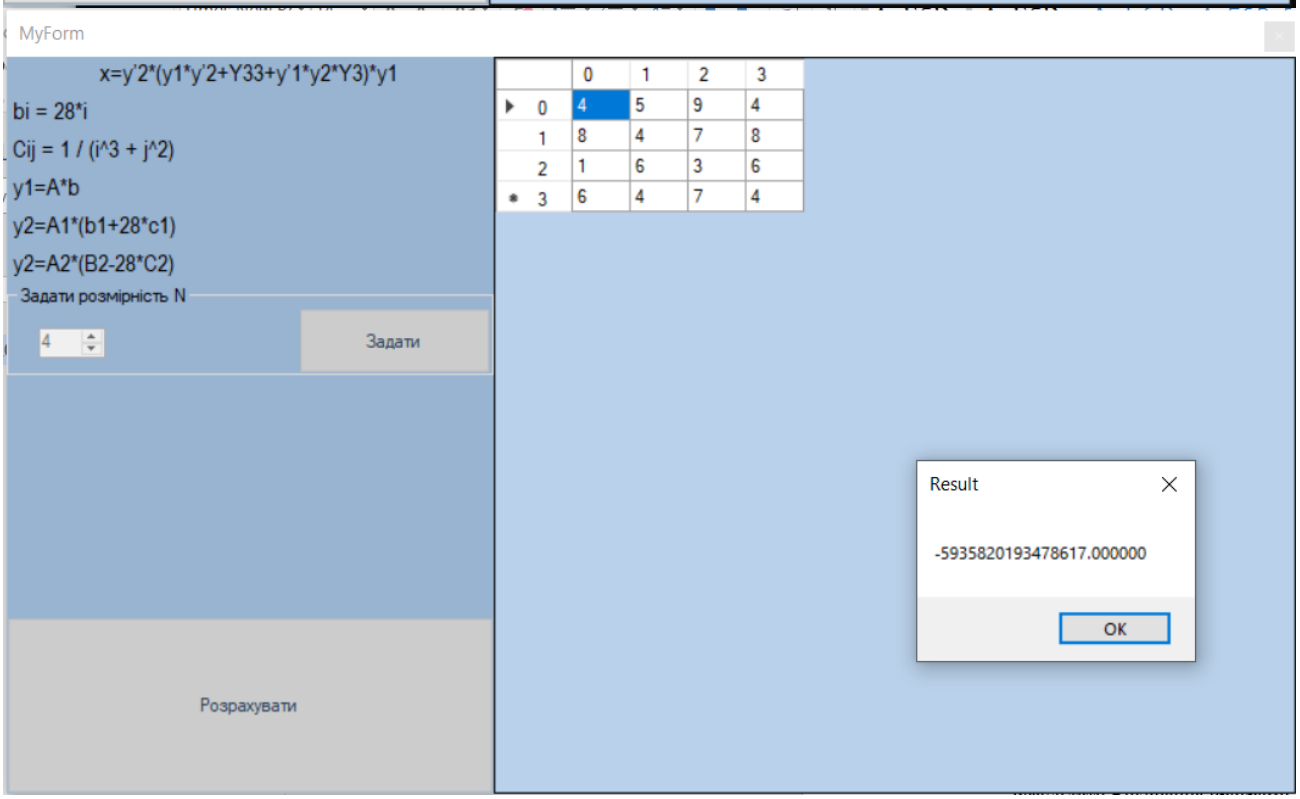
xx	$x=y_2'*(y_1y_2'+Y_3^3+y_1'y_2Y_3)*y_1$ число		
	$b_i=28i$	$A_1(b_1+28c_1)$	$A_2(B_2-28C_2)$ $C_{ij}=1/(i^3+j^2)$

Схема декомпозиція задачі:



Скріни роботи програми:





Результати Виконання програми які зберігаються у файлах:

Processor1.txt

bi:

28
56
84
112

A:

4	5	9	4
8	4	7	8
1	6	3	6
6	4	7	4

y1 = A*bi:

980
1176
840
840

y2:

1049
2003
2953
2704

y2':

1049	2003	2953	2704

y1*y2':

1028020	1962940	2893940	2649920
1233624	2355528	3472728	3179904
881160	1682520	2480520	2271360
881160	1682520	2480520	2271360

Y3^3:

-171695.8982	-1296893.1820	-1321405.8441	-1048809.0603
82085.2859	-193277.2471	-554316.4395	-472230.2138
841832.8051	1230525.5152	-976994.6747	-980935.7742
1550948.4245	2667223.9444	-1204808.1115	-1313893.1701

y1*y2' + Y3^3:

856324.1018	666046.8180	1572534.1559	1601110.9397
1315709.2859	2162250.7529	2918411.5605	2707673.7862
1722992.8051	2913045.5152	1503525.3253	1290424.2258
2432108.4245	4349743.9444	1275711.8885	957466.8299

y1'*y2*Y3:

-210089292.6720	130218914.7392	672751083.0320	630424078.2336
-131045474.2240	-125627279.1760	279578864.4768	237365755.6704
-391672045.6320	-1020521105.0048	311294016.9920	393413027.2240
-826917443.6320	-1208537354.4272	306962715.1248	375137601.7648

y1*y2' + Y3^3 + y1'*y2*Y3:

-209232968.5702	130884961.5572	674323617.1879	632025189.1733
-129729764.9381	-123465028.4231	282497276.0373	240073429.4566
-389949052.8269	-1017608059.4896	312797542.3173	394703451.4498
-824485335.2075	-1204187610.4828	308238427.0133	376095068.5947

y2'(y1*y2' + Y3^3 + y1'*y2*Y3):

-3860262002599.9043	-6371122025676.1035	3030375367439.8184	3326381860255.8799

x:

-5935820193478617

Processor2.txt

b1:

8
7
8
5

c1:

2
1
8
3

A1:

7	8	1	1
1	7	5	6
8	4	8	5
5	8	6	8

b1+28*c1:

64
35
232
89

y2=A1*(b1+28*c1):

1049
2003
2953
2704

y1:

980
1176
840
840

y1':

980	1176	840	840

y1'*y2:

8135428

Y3:

-25.8240	16.0064	82.6940	77.4912
-16.1080	-15.4420	34.3656	29.1768
-48.1440	-125.4416	38.2640	48.3580
-101.6440	-148.5524	37.7316	46.1116

y1'*y2*Y3:

-210089292.6720	130218914.7392	672751083.0320	630424078.2336
-131045474.2240	-125627279.1760	279578864.4768	237365755.6704
-391672045.6320	-1020521105.0048	311294016.9920	393413027.2240
-826917443.6320	-1208537354.4272	306962715.1248	375137601.7648

Processor3.txt

B2:

4	4	9	7
2	3	2	1
2	2	9	3
5	9	4	8

Cij:

0	1	0.2500	0.1111
1	0.5000	0.2000	0.1000
0.1250	0.1111	0.0833	0.0588
0.0370	0.0357	0.0322	0.0277

A2:

1	2	9	9
1	1	4	3
5	3	4	4
5	5	5	4

B2-28*Cij:

4	-24	2	3.8892
-26	-11	-3.6000	-1.8000
-1.5000	-1.1108	6.6676	1.3536
3.9640	8.0004	3.0984	7.2244

Y3=A2*(B2-28*Cij):

-25.8240	16.0064	82.6940	77.4912
-16.1080	-15.4420	34.3656	29.1768
-48.1440	-125.4416	38.2640	48.3580
-101.6440	-148.5524	37.7316	46.1116

Y3^2:

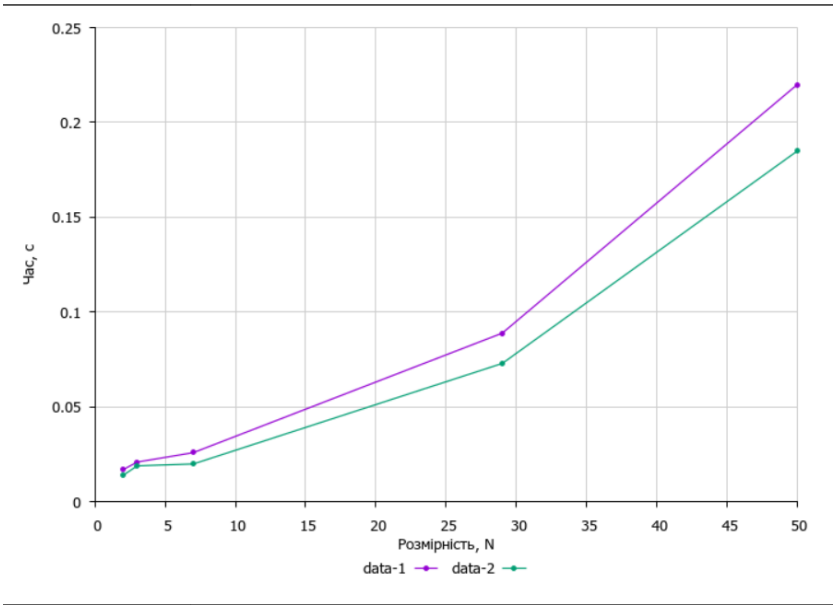
-11448.6876	-22545.2915	4502.6499	6038.0425
-3955.4314	-8664.5352	553.1441	1308.4642
-3493.5986	-10817.1373	-5003.3374	-3310.4855
-1485.7809	-10916.1095	-10326.8149	-8259.8948

Y3^3:

-171695.8982	-1296893.1820	-1321405.8441	-1048809.0603
82085.2859	-193277.2471	-554316.4395	-472230.2138
841832.8051	1230525.5152	-976994.6747	-980935.7742
1550948.4245	2667223.9444	-1204808.1115	-1313893.1701

Дослідження ефективності розпаралелювання (за часом виконання).

	Час виконання обчислень в секундах	
Розмірність	Однопоточна програма	MPI-програма
2	0.017	0.014
3	0.021	0.019
7	0.026	0.020
29	0.089	0.073
50	0.220	0.185



Параметри комп'ютера:
CPU: Intel Core i7-7700HQ 2.8Ghz, 4 Cores
RAM: 8Gb DDR4-2133Mhz
OS: Windows 10

Висновки. Під час виконання лабораторної роботи створено програму, що розраховує результат заданого матричного виразу, з використання технології MPI. Для цього спочатку було розроблено схему декомпозиції. Аналіз схеми показав, що доцільно розбивати обчислення на три процесори, що і було зроблено.

Як видно з графіку, програма яка виконується послідовно, є повільнішою за ту, в якій дані обчислюються паралельно, навіть при невеликих обсягах даних, вона є ефективнішою за послідовну програму. В роботі використано паралелізм на рівні підзадач, оскільки передбачається, що кожен блок зі схеми декомпозиції є реалізований у виді функції. Це є середньо блоковий паралелізм. Обмін даними відбувається через пересилання даних між процесами. MPI-програма не є значно швидшою за послідовну, бо кожен з процесів обчислює різну кількість даних(наприклад в першому процесі множиться вектор-стовпець на вектор-рядок а в третьому процесі матриця підноситься до третього степеню). Інші процеси вимушені чекати, поки кожен процес завершить проміжні обчислення для отримання даних для подальших обчислень.

Додаток А. Код проекту