Міністерство освіти і науки

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи № 1

з дисципліни: “Паралельні та розподілені обчислення”

на тему: “Використання функціональної декомпозиції для розв’язку обчислювальних задач”

Викнова ст.гр :  
Теслюк Н.А.  
Перевірив :  
Козак Н.Б

**Мета лабораторної 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.

**Завдання**

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.

На основі створеного алгоритму написати програму яка дозволяє обчислити вираз та ілюструє проведену декомпозицію.

Правила знаходження елементів виразу.

1).Задати\* квадратну матрицю А порядку n. Отримати вектор(стовпець) y1 = Ab, де b – вектор-стовпець, елементи якого обраховуються за формулою, згідно варіанту.

2).Задати квадратну матрицю А1 порядку n та вектори-стовпці b1 та c1 з n елементами кожен. Отримати вектор y2 згідно формули, що задається варіантом.

3).Задати квадратні матриці А2 та B2 порядку n. Отримати матрицю Y3, яка залежить від А2, B2 та додатково визначеної матриці С2, елементи якої знаходяться за формулою, вказаною варіантом.

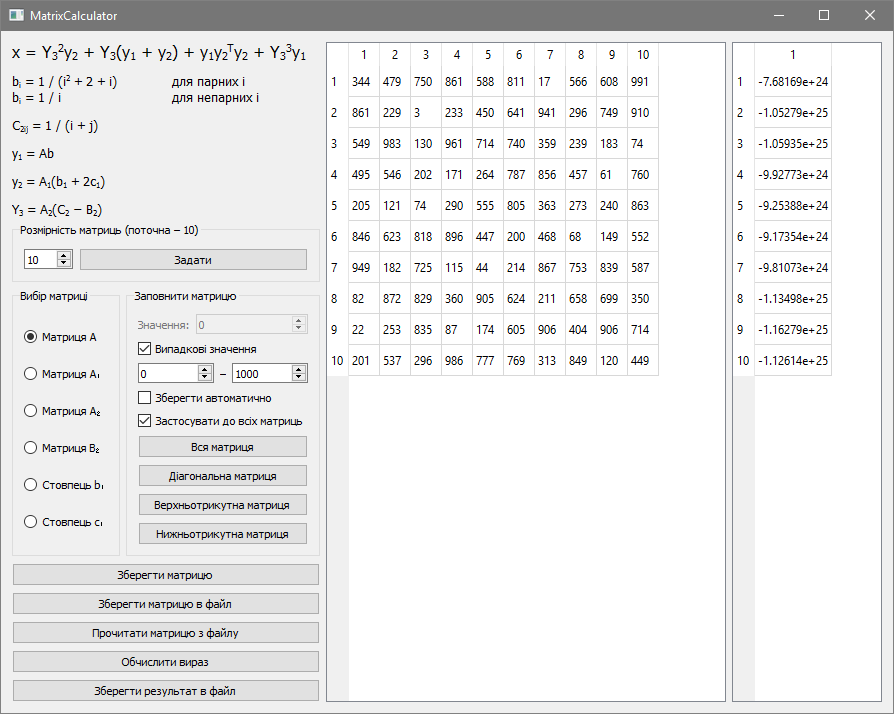
y' означає операцію транспонування; i,j=1…n (n – вхідна розмірність).

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | x = Y32y2 + Y3(y1 + y2) + y1y2'y2 + Y33y1 стовпець | | |
| bi=1/(i2+2+i) для парних і  bi=1/i для непарних і | A1(b1+2c1) | A2(C2-B2)  Cij=1/(i+j) |

**Функціональна декомпозиція**



**Скріншот програми**



**Результати роботи**

Параметри комп’ютера:

CPU: Intel Core i3-4000M 2.40Ghz, 2 Cores, 4 Threads

RAM: 8Gb DDR3-1600Mhz

OS: Windows 10 Enterprise x64

|  |  |  |
| --- | --- | --- |
|  | Час виконання обчислень в секундах | |
| Розмірність | Однопоточна програма | MPI-програма |
| 10 | 0.014 | 0.009 |
| 100 | 0.199 | 0.172 |
| 500 | 8.894 | 8.177 |
| 1000 | 66.324 | 65.356 |
| 1500 | 223.935 | 222.646 |
| 2000 | 507.783 | 503.884 |

**Висновок**

В роботі використано паралелізм на рівні підзадач, оскільки передбачається, що кожен блок зі схеми декомпозиції є реалізований у виді функції. Це є середньоблоковий паралелізм. Обмін даними відбувається через пересилання даних між процесами. MPI-програма не є значно швидшою за однопоточну, бо в одному з процесів матриця множиться на матрицю, а в інших процесах паралельно матриця множиться на стовпець. Інші процеси вимушені чекати, поки кожен процес завершить проміжні обчислення для отримання даних для подальших обчислень.