УДК 004.75, 004.8

М.І. Патик
В.О. Денисюк

РЕАЛІЗАЦІЯ РОЗПОДІЛЕНОГО МНОЖЕННЯ МАТРИЦЬ НА ОСНОВІ АРХІТЕКТУРИ MASTER-SLAVE ІЗ ВИКОРИСТАННЯМ СОКЕТІВ TCP/IP

Вінницький національний технічний університет

Анотація

Розглянуто розробку програмного комплексу для розподіленого множення матриць на основі архітектури Master-Slave із використанням сокетів TCP/IP. Проаналізовано ефективність розпаралелювання обчислювально складних задач лінійної алгебри. Наведено результати тестування продуктивності та показники прискорення для матриць великої розмірності.

Ключові слова: *розподілені обчислення, множення матриць, сокети, Master-Slave, продуктивність, Python.*

Abstract

The development of a software complex for distributed matrix multiplication based on the Master-Slave architecture using TCP/IP sockets is considered. The efficiency of parallelization of computationally complex linear algebra problems is analyzed. Performance testing results and speedup metrics for large-scale matrices are presented.

Keywords: *distributed computing, matrix multiplication, sockets, Master-Slave, performance, Python.*

Вступ

Актуальність реалізації розподіленого матричного множення обумовлена стрімким зростанням обсягів даних у задачах машинного навчання, наукового моделювання та обробки зображень. Класичний послідовний алгоритм має часову складність $O(N^3)$, що робить його неефективним для матриць великої розмірності на одному вузлі.

Використання розподілених систем дозволяє поділити задачу на підзадачі та виконувати їх паралельно, що суттєво скорочує час отримання результату.

Метою роботи є розробка та дослідження ефективності програмної реалізації алгоритму множення матриць у клієнт-серверній архітектурі.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- розробка ефективної клієнт-серверної архітектури (Master-Slave) для оптимального розподілу матричних обчислень між вузлами мережі;
- мінімізація накладних витрат на передачу даних шляхом використання ефективних механізмів серіалізації та протоколу TCP/IP [1];
- створення програмної реалізації алгоритму, що дозволяє динамічно підключати клієнтські модулі та масштабувати обчислення;
- проведення порівняльного аналізу продуктивності послідовного та розподіленого методів для визначення коефіцієнтів прискорення та ефективності.

Виклад основного матеріалу

Матричне множення є фундаментальною операцією, яка використовується у багатьох сферах: від комп'ютерної графіки та фізичних симуляцій до статистичних методів і нейронних мереж. Стандартний послідовний алгоритм має високу кубічну складність, що робить його ресурсомістким. Для підвищення швидкодії застосовуються паралельні обчислення - підхід, що передбачає одночасне виконання частин задачі на різних апаратних ресурсах [2, 3].

Для реалізації взаємодії між вузлами у роботі використано технологію **сокетів** (Sockets). Сокети - це програмний інтерфейс для обміну даними між процесами, який дозволяє будувати гнучкі системи на основі протоколу TCP/IP [1]. На відміну від систем зі спільною пам'яттю, сокети реалізують модель передачі повідомлень (Message Passing), де кожен вузол має власну пам'ять, а дані передаються мережею у вигляді серіалізованих пакетів. Це забезпечує незалежність вузлів та можливість масштабування системи.

У роботі реалізовано архітектуру типу **Master-Slave** («Начальник–Підлеглі»). Ця модель є найбільш ефективною для матричних операцій у мережевому середовищі. Вона працює за наступними принципами:

- **вузол Master (Сервер) -** генерує вхідні матриці, виконує декомпозицію (розбиття) першої матриці на горизонтальні смуги або блоки рядків, сервер ініціює з'єднання, очікує підключення клієнтів і розподіляє завдання;
- **вузли Slaves (Клієнти) -** підключаються до сервера, кожен клієнт отримує свою унікальну частину першої матриці та повну копію другої матриці, клієнт виконує операцію множення над отриманими даними, використовуючи оптимізовані математичні бібліотеки (NumPy), і повертає обчислений блок результату назад серверу;
- **збір результатів - с**ервер отримує часткові результати від усіх клієнтів, синхронізує потоки та об'єднує блоки у єдину результуючу матрицю.

Оптимізація алгоритму досягається за рахунок «грубозернистого паралелізму». Замість передачі окремих чисел, система передає великі блоки даних за один раз. Це дозволяє мінімізувати накладні витрати на мережеву комунікацію, оскільки час передачі даних часто є вузьким місцем у розподілених системах. Також використано багатопотоковість на сервері для одночасного обслуговування декількох клієнтів без блокування роботи.

Програмна реалізація виконана мовою Python [4]. Вона включає модуль сервера, який керує процесом та візуалізує результати, та модуль клієнта, який виконує математичні обчислення. Для передачі складних структур даних (масивів) використано механізм серіалізації *pickle*.

Аналіз результатів тестування програми Тестування проводилося шляхом порівняння часу виконання послідовного алгоритму (на одному процесорі) та розподіленого алгоритму (з використанням двох клієнтів) на матрицях різної розмірності (табл.1) (рис.1).

Таблиця 1 – Час виконання та показники ефективності (2 клієнти)

Розмір матриці (N×N)	Час послідовний, с	Час розподілений, с	Прискорення	Ефективність
100 × 100	0.0013	0.1364	0.01	0.00
500 × 500	0.0996	0.2249	0.44	0.22
1 000 ×1 000	0.9924	0.7840	1.27	0.63
2 000 × 2 000	11.1059	8.4320	1.32	0.66
5 000 × 5 000	212.2020	161.0911	1.32	0.66

Аналіз даних (табл. 1) (рис.1) дозволяє зробити наступні висновки:

- при малих розмірах матриць (до 500 елементів) розподілена система працює повільніше за послідовну, це пояснюється тим, що час на встановлення з'єднання та передачу даних перевищує час самих обчислень;
- при збільшенні розмірності матриць (від 1000 і більше) спостерігається стабільне прискорення, частка корисних обчислень зростає, а вплив мережевих затримок стає менш значущим;
- для великих матриць (5 000 x 5 000) досягнуто коефіцієнт прискорення 1.32, це підтверджує, що запропонований підхід є ефективним для обробки великих масивів даних.

Отже, основними чинниками, які впливають на зміну розглянутих показників, таких як коефіцієнт прискорення, коефіцієнт ефективності та час виконання програми, є розмір вхідних даних, в даному випадку розмірність досліджуваної матриці, та накладні витрати на мережеву комунікацію. Експериментально встановлено, що розподілений метод стає виправданим лише

при досягненні певного порогу обчислювальної складності (у даному дослідженні N≥1000), коли час корисних обчислень починає переважати над часом передачі даних та синхронізації процесів у мережі.

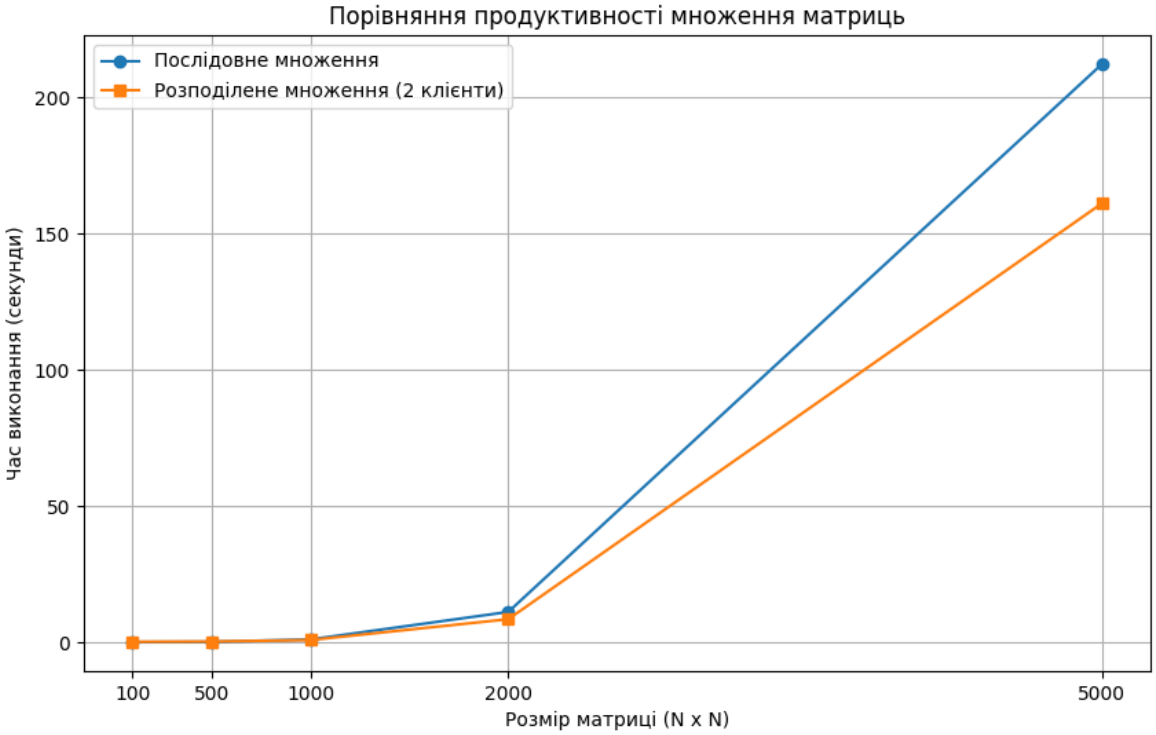

Рисунок 1 – Графік порівняння продуктивності послідовного та розподіленого множення матриць для двох клієнтів.

Висновки

Досліджено матричне множення як одну з фундаментальних та найбільш обчислювально містких операцій у задачах лінійної алгебри, визначено основні проблеми його виконання на одному вузлі при роботі з великими масивами даних.

Розглянуто методи розпаралелювання обчислювальних процесів та обґрунтовано доцільність використання архітектури з розподіленою пам’яттю, де взаємодія реалізується за моделлю Master-Slave («Начальник–Підлеглі»).

На основі літературних джерел досліджено технологію сокетів (TCP/IP) як інструмент для побудови гнучких розподілених систем, сформульовано принципи організації обміну повідомленнями (Message Passing) між незалежними обчислювальними вузлами.

Програмно реалізовано алгоритм розподіленого множення матриць мовою Python із використанням бібліотеки NumPy та описано всі компоненти коду. Також наведено UML-діаграми компонентів та діяльності алгоритму.

Проведено тестування розробленої програми, проаналізовано її результати на матрицях різної розмірності, досліджено коефіцієнти прискорення та ефективності використання ресурсів.

Визначено, що для задач великого обсягу (від 1000 елементів) розподілений підхід забезпечує стабільне прискорення обчислень (до 1.32 рази), тоді як на малих обсягах даних переваги нівелюються накладними витратами на мережеву комунікацію.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. TCP/IP Modbus (Master/slave communication). URL: https://stackoverflow.com/questions/61025518/tcp-ip-modbus-master-slave-communication
2. Parallel Scaling Guide // Colorado school of mines. URL: https://rc-docs.mines.edu/pages/user_guides/Parallel_Scaling_Guide.html?utm_source=chatgpt.com.

3. Foster I. Designing and Building Parallel Programs: Concepts and Experience / I. Foster. Reading, MA: Addison-Wesley, 1995. 417 p.
4. The Python Standard Library: socket — Low-level networking interface. // Python Documentation. URL: https://docs.python.org/3/library/socket.html.

Патик Максим Іванович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: mpatik2006@gmail.com;

Денисюк Валерій Олександрович – канд. техн. наук, доцент, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, м.Вінниця, e-mail: vad64@i.ua.

Patyk Maksym Ivanovych – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: mpatik2006@gmail.com;

Denysiuk Valerii Olexandrovich – Ph.D., Assistant Professor, Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: vad64@i.ua