УДК 004.8

М.Д. Гусак
В.О. Денисюк

РЕАЛІЗАЦІЯ МАТРИЧНОГО МНОЖЕННЯ ЗА ДОПОМОГОЮ BLOCK MATRIX

Вінницький національний технічний університет

Анотація

У роботі розглянуто задачу підвищення швидкодії матричного множення для даних великої розмірності. Проаналізовано проблему ефективного використання кеш-пам'яті процесора при виконанні класичних алгоритмів. Обґрунтовано вибір блочного методу (Block Matrix Multiplication) для покращення локальності даних та технології OpenMP для реалізації паралельних обчислень. Розроблено програмну реалізацію мовою C++ та проведено експериментальне дослідження, яке показало суттєве скорочення часу виконання обчислень на багатоядерних архітектурах.

Ключові слова: *матричне множення, Block Matrix, OpenMP, кеш-пам'ять, паралельні обчислення, продуктивність.*

Abstract

The paper addresses the optimization of matrix multiplication for large-scale data. The problem of efficient CPU cache utilization during the execution of classical algorithms is analyzed. The choice of the Block Matrix Multiplication method to improve data locality and OpenMP technology for implementing parallel computing is substantiated. A C++ software implementation was created, and experimental testing was conducted, demonstrating a significant reduction in computation time on multi-core architectures.

Keywords: *matrix multiplication, Block Matrix, OpenMP, cache memory, parallel computing, performance.*

Вступ

У сучасному світі високопродуктивних обчислень операції лінійної алгебри, зокрема множення матриць, є фундаментом для вирішення широкого спектра завдань: від комп'ютерної графіки до машинного навчання. Зі зростанням обсягів даних класичні алгоритми множення матриць стають вузьким місцем, обмежуючи загальну продуктивність систем.

Традиційний алгоритм «рядок на стовпець» неефективно використовує ієрархію пам'яті сучасних процесорів, призводячи до частих промахів кеш-пам'яті (cache misses). Актуальність дослідження полягає у необхідності використання блочного підходу (Block Matrix Multiplication), який дозволяє значно підвищити локальність даних, а застосування технологій паралельного програмування, таких як OpenMP, дає змогу задіяти потенціал багатоядерних архітектур [1, 2].

Метою роботи є підвищення швидкодії матричного множення шляхом розробки паралельного блочного алгоритму.

Постановка задачі дослідження

Для досягнення поставленої мети у роботі вирішуються наступні задачі:

- аналіз існуючих алгоритмів множення матриць та виявлення причин їхньої неефективності на великих обсягах даних;
- дослідження впливу архітектури кеш-пам'яті процесора на швидкість обчислень;
- розробка математичної моделі та алгоритму блочного множення матриць;
- програмна реалізація алгоритму мовою C++ із застосуванням стандарту OpenMP;
- проведення експериментального дослідження ефективності розробленого рішення та порівняння часу виконання послідовної і паралельної версій.

Виклад основного матеріалу

Операція множення матриць є однією з найбільш ресурсомістких процедур, що має асимптотичну складність $O(N^3)$. У класичному ітераційному алгоритмі доступ до елементів другої матриці відбувається по стовпцях. Оскільки в мові C++ масиви зберігаються в пам'яті порядково (Row-Major Order), це призводить до порушення принципу просторової локальності. При великих кожен наступний елемент стовпця знаходиться в новій кеш-лінії, що спричиняє масові промахи кешу та простій процесора в очікуванні даних з RAM («стіна пам'яті») [3].

Для вирішення цієї проблеми застосовано блочний алгоритм (Tiling). Глобальна задача розбивається на підзадачі обробки підматриць (блоків) розміром $BS \times BS$. Це трансформує структуру алгоритму з трьох вкладених циклів у шість, але дозволяє багаторазово використовувати дані, завантажені в кеш. Математичне моделювання показує, що умова ефективності без «пробуксовки кешу» (cache thrashing) виконується, коли сумарний розмір трьох робочих блоків не перевищує розмір L1-кешу:

$$3 \cdot BS^2 \cdot sizeof(double) \leq CacheSize$$

Такий підхід знижує обсяг трафіку пам'яті з $O(N^3)$ до $O\left(\frac{N^3}{\sqrt{M}}\right)$, де M – обсяг швидкої пам'яті [1, 4, 5].

Паралелізація алгоритму виконана засобами OpenMP [6, 7]. Оскільки обчислення кожного блоку результуючої матриці є незалежним, застосовано декомпозицію зовнішніх циклів. Використання директиви #pragma omp parallel for collapse(2) дозволило об'єднати ітерації по рядках і стовпцях блоків у єдиний простір задач, що покращує балансування навантаження. Для розподілу ітерацій обрано стратегію schedule(static), оскільки операція множення щільних матриць є регулярною і навантаження на потоки прогнозоване. У кожному потоці використовуються локальні змінні-акумулятори для уникнення хибного розділення кешу (false sharing) при записі результатів.

Експериментальне дослідження проводилося на багатоядерній архітектурі з використанням матриць розмірністю від 500 до 2500 елементів. Результати (табл. 1) демонструють, що для матриці 2500×2500 блочний метод дозволяє скоротити час виконання з 130,4 с до 3,8 с при використанні 12 потоків.

Таблиця 1 – Час виконання та прискорення (при розмірі блоку BS = 32) (с)

Розмір матриці (N)	Послідовний	2 потоки	4 потоки	12 потоків
500	0,1690	0,0618	0,0449	0,0275
1 000	1,6175	0,4067	0,2932	0,1895
1 500	10,8926	2,5929	1,7463	0,9971
2 000	69,0146	5,8488	3,1555	1,9127
2 500	130,4458	9,8756	6,9832	3,8264

Отримане прискорення понад 30 разів пояснюється синергетичним ефектом: блочний доступ мінімізує затримки пам'яті, а багатопотоковість утилізує обчислювальні ядра. Зменшення прискорення на малих матрицях пов'язане з накладними витратами на створення та синхронізацію потоків.

Навіть мінімальне розпаралелювання (2 потоки) дає непропорційно великий приріст продуктивності (рис.1) порівняно з послідовним кодом (табл.2). Це пояснюється тим, що послідовна версія страждає від неефективної роботи з кешем, тоді як блочний алгоритм оптимізує доступ до пам'яті, дозволяючи процесору виконувати обчислення без зайвих очікувань даних з RAM.

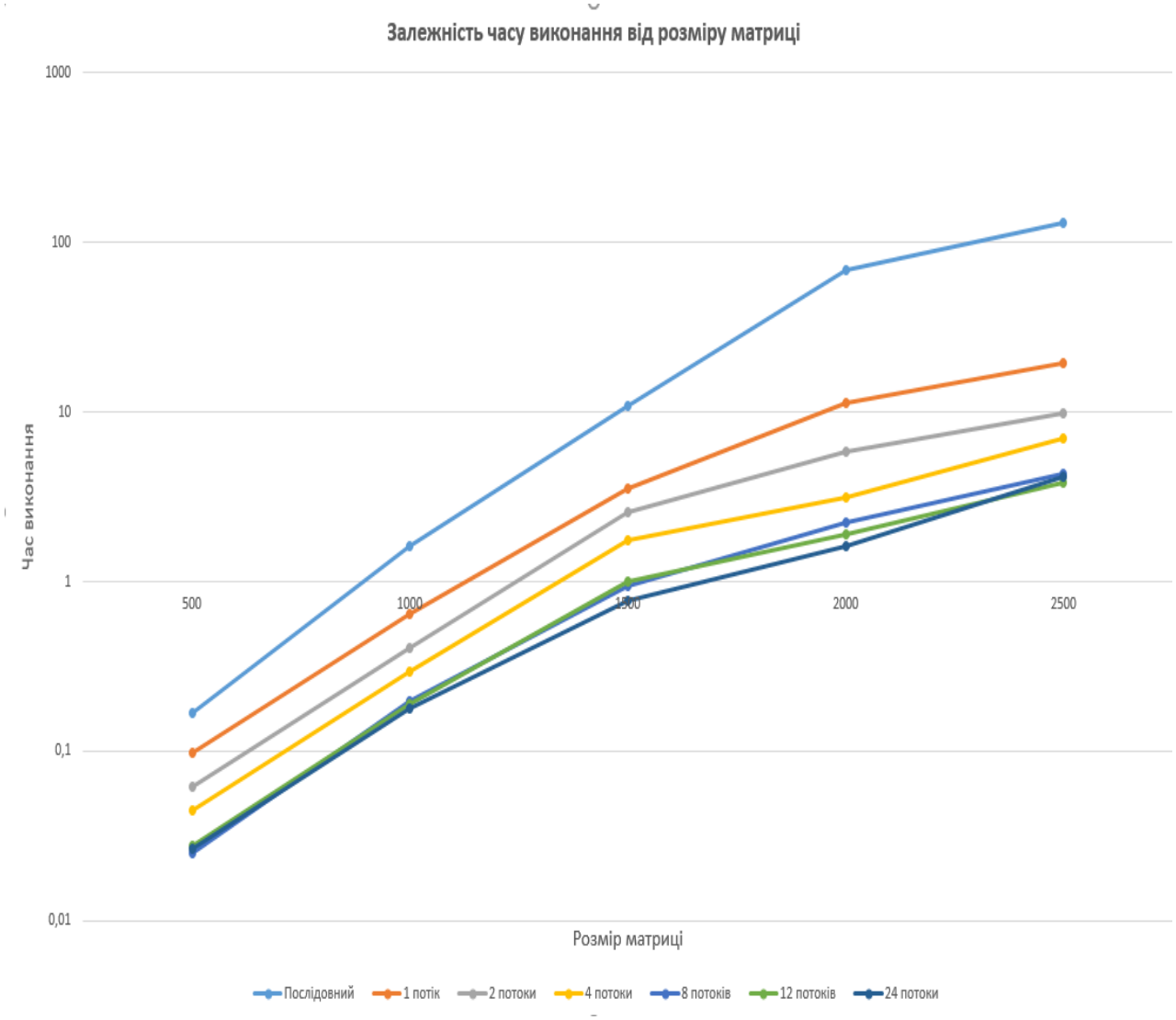

Рисунок 1 – Графік залежності часу виконання від розміру матриці

Таблиця 2 – Ефективність розпаралелення

Кількість потоків	1	2	4	8	12	24
Час виконання, с	19,41	9,88	6,98	4,36	3,83	4,19
Прискорення	1,00x	1,97x	2,78x	4,45x	5,07x	4,63x

Досліджено залежність швидкодії від кількості потоків для матриці 2 500 x 2 500. Максимальну швидкодію зафіксовано на 12 потоках (рис.2). Подальше збільшення кількості потоків призводить до зниження продуктивності через конкуренцію за доступ до спільної пам'яті.

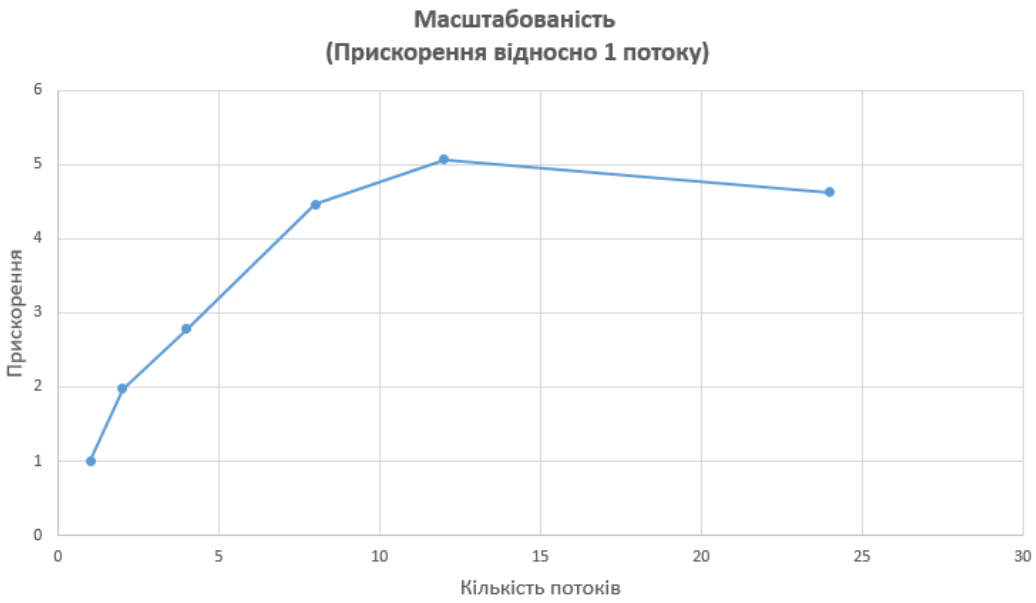

Рисунок 2 – Графік залежності прискорення від кількості потоків

Висновки

У роботі досліджено та реалізовано паралельний алгоритм блочного множення матриць. Встановлено, що класичний метод є неефективним для великих розмірностей через ігнорування ієрархії пам'яті комп'ютера.

Використання блочної декомпозиції дозволило забезпечити виконання умов часової та просторової локальності. Реалізація за допомогою OpenMP продемонструвала високу масштабованість алгоритму. Отримані результати підтвердили, що комбінація алгоритмічної оптимізації (блочний метод) та апаратного паралелізму дозволяє досягти суттєвого підвищення продуктивності обчислювальних систем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Глибовець М. М. Основи паралельних алгоритмів: навч. посіб. Київ: Києво-Могилянська академія, 2018. 240 с.
2. Шпак З. Я. Програмування мовою С++: підручник. Львів: Видавництво Львівської політехніки, 2011. 436 с.
3. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. 6th ed. San Francisco : Morgan Kaufmann, 2019. 936 p.
4. Коцовський В. М. Теорія паралельних обчислень: навч. посіб. Ужгород: ПП «АУТДОР-Шарк», 2021. 188 с.
5. Семеренко В. П. Технології паралельних обчислень: навч. посіб. Вінниця: ВНТУ, 2018. 104 с.
6. Chapman B., Jost G., Pas R. Using OpenMP:Portable Shared Memory Parallel Programming. Cambridge: MIT Press, 2008. 353.
7. The OpenMP API specification for parallel programming. URL: http://www.openmp.org.

Гусак Максим Дмитрович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: goog69169@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.

Husak Maksym Dmytrovych – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: goog69169@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