УДК 004.8

Ю. А. Маціпура
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ 3-WAY MERGE SORT

Вінницький національний технічний університет

Анотація

У роботі розглянуто розробку та дослідження ефективності паралельного алгоритму сортування масивів даних на основі тришляхового алгоритму злиття 3-way Merge Sort. Проаналізовано особливості послідовних та паралельних методів сортування, проведено огляд відомих підходів до розпаралелювання рекурсивних алгоритмів, а також обґрунтовано вибір засобів програмної реалізації. У роботі розроблено програмний модуль на мові програмування C#, створено та описано алгоритмічну структуру сортування, реалізовано механізм адаптивного переходу між послідовним та паралельним виконанням на основі порогового значення. Проведено тестування швидкодії, побудовано графіки порівняння часу виконання паралельного і послідовного сортування. Реалізація програмного модуля дозволяє підвищити продуктивність опрацювання великих обсягів даних за рахунок ефективного використання багатопотоковості. Одержані результати можуть бути використані в системах обробки даних, високопродуктивних програмних комплексах та навчальних проектах, що вимагають оптимізації алгоритмів сортування.

Ключові слова: *паралельні обчислення, сортування, 3-way Merge Sort, багатопотоковість, продуктивність алгоритму.*

Abstract

The paper examines the development and performance evaluation of a parallel sorting algorithm for data arrays based on the three-way merge sorting technique (3-way Merge Sort). The features of sequential and parallel sorting methods are analyzed, an overview of known approaches to parallelizing recursive algorithms is provided, and the choice of software implementation tools is justified. A software module was developed in the C# programming language, the algorithmic structure of sorting was created and described, and a mechanism for adaptive switching between sequential and parallel execution based on a threshold value was implemented. Performance testing was conducted, and charts comparing the execution time of parallel and sequential sorting were constructed. The implementation of the software module increases the efficiency of processing large volumes of data through effective use of multithreading. The obtained results can be applied in data processing systems, high-performance software solutions, and educational projects that require optimization of sorting algorithms.

Keywords: *parallel computing, sorting, 3-way Merge Sort, multithreading, algorithm performance.*

Вступ

Актуальність реалізації паралельного сортування масивів даних на основі 3-way Merge Sort полягає у значному прискоренні обчислень, що є критично важливим для сучасних задач обробки великих обсягів інформації, таких як бази даних, машинне навчання, комп'ютерна графіка та наукове моделювання. У міру збільшення обсягів даних виникає потреба не лише у вдосконаленні алгоритмів сортування, а й у їх паралельному виконанні для максимально ефективного використання багатоядерних систем. Алгоритм 3-way Merge Sort дозволяє більш рівномірно розподілити навантаження між потоками та зменшити глибину рекурсії, що сприяє підвищенню продуктивності та зниженню часу обробки великих масивів. Використання паралельних обчислювальних архітектур дає змогу одночасно обробляти різні частини масиву на різних потоках, що суттєво скорочує час виконання операцій. Це забезпечує масштабованість і високу ефективність алгоритму для різних сфер науки, інженерії та бізнес-аналітики. Розробка та дослідження паралельних методів сортування дозволяє закріпити теоретичні знання, а також отримати практичні навички роботи з багатопотоковістю, рекурсивними структурами та оптимізацією програмного коду. Важливим аспектом є також тестування продуктивності розробленого алгоритму та порівняння його з послідовними методами сортування.

Метою роботи є розробка та експериментальне дослідження паралельного алгоритму сортування на основі тришляхового злиття з використанням мови програмування C#.

Об’єктом дослідження є процес сортування даних у багатопотокових обчислювальних системах, а предметом – методи проектування та програмної реалізації паралельного алгоритму сортування.

Завдання роботи включають розробку ефективного програмного модуля сортування, побудову його алгоритмічної структури, реалізацію механізму паралельного виконання та проведення дослідження продуктивності розробленого алгоритму. Реалізація такого алгоритму дозволяє забезпечити прискорення обробки великих наборів даних і підвищити загальну ефективність роботи сучасних обчислювальних систем.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- розробка ефективної архітектури паралельного алгоритму сортування на основі 3-way Merge Sort;
- оптимізація розподілу навантаження між потоками для прискорення сортування;
- розробка програми для паралельного сортування масивів даних із застосуванням багатопотоковості в C#;
- тестування програми та аналіз продуктивності отриманого алгоритму.

Виклад основного матеріалу

Сортування даних є однією з ключових операцій у сучасних обчислювальних системах, оскільки воно широко використовується в базах даних, комп’ютерній графіці, машинному навчанні, обробці сигналів та моделюванні складних процесів [1–3]. Існує велика кількість алгоритмів сортування, серед яких традиційні методи мають складність O(n log n) та демонструють високу ефективність у послідовних обчисленнях. Однак зі зростанням обсягів даних та переходом до багатоядерних систем виникає потреба у розробці більш масштабованих та паралельних методів обробки інформації [4]. Одним із таких перспективних підходів є алгоритм тришляхового злиття — 3-way Merge Sort, який дозволяє зменшити глибину рекурсії та рівномірніше розподілити дані між потоками [5].

Для правильного розуміння принципів роботи 3-way Merge Sort було проведено аналіз традиційних алгоритмів злиття та їхньої поведінки у багатопотокових середовищах. Класичний Merge Sort виконує поділ масиву на дві частини, тоді як тришляхова версія передбачає поділ на три підмасиви, що дає змогу зменшити кількість рекурсивних рівнів та збільшити ступінь паралельності. Такий підхід особливо ефективний у системах, де доступно багато апаратних потоків, наприклад у сучасних процесорах або хмарних середовищах [5, 6].

Алгоритм 3-way Merge Sort складається з двох основних етапів.

1. Рекурсивний поділ масиву на три частини. Масив розбивається на три приблизно рівні частини, після чого кожна з них сортується незалежно. У паралельній реалізації ці частини можуть передаватися різним потокам, що значно прискорює обчислення.
2. Злиття трьох відсортованих підмасивів. На етапі злиття використовуються три покажчики для послідовного порівняння елементів. Це забезпечує лінійну складність операції злиття та дозволяє ефективно поєднувати результати паралельної роботи потоків.

Основні переваги 3-way Merge Sort у паралельних обчисленнях включають:

- покращене балансування навантаження, оскільки три підзадачі розподіляються між потоками більш рівномірно;
- зменшення рекурсивної глибини, що знижує накладні витрати на виклики функцій;
- високу масштабованість, оскільки додавання потоків може забезпечити додатковий приріст швидкодії;
- ефективну організацію злиття, яка зменшує блокування в спільній пам’яті;
- покращену роботу з великими масивами, що є критично важливим для промислових застосувань.

У роботі також проаналізовано оптимізаційні прийоми, які дозволяють підвищити продуктивність паралельної реалізації алгоритму.

1. Паралелізація обчислень. Використано механізми Task Parallel Library (TPL), які дозволяють запускати сортування підмасивів у незалежних потоках. Це значно зменшує час виконання, особливо на масивах великого розміру.

2. Оптимізація структури злиття. Для злиття трьох масивів застосовано алгоритм із трьома індексами, що зменшує кількість порівнянь та мінімізує доступи до пам’яті.
3. Обмеження рівня паралельності. Щоб уникнути надмірного створення потоків на нижчих рівнях рекурсії, використано умову, яка дозволяє переходити до послідовного сортування на малих підмасивах.
4. Зменшення копіювань масивів. Для оптимізації використання пам’яті частково уникається створення додаткових буферів при злитті.

У рамках роботи розроблено програмну реалізацію паралельного алгоритму 3-way Merge Sort на мові програмування C#. Описано структуру програми, логіку взаємодії потоків, а також механізми організації рекурсії та злиття масивів [7, 8]. Виконано тестування алгоритму на наборах даних різних розмірів та проведено аналіз продуктивності, що дозволило оцінити ефективність запропонованих оптимізацій та підтвердити переваги паралельної реалізації алгоритму у порівнянні з послідовною версією.

Аналіз результатів тестування програми виконано для різної кількості елементів масиву (табл.1).

Таблиця 1 – Час виконання програми на різних вхідних даних (с)

Розмір масиву n	Послідовне сортування	Паралельне сортування
10 000	5	31
50 000	28	64
100 000	47	65
1 000 000	381	320
10 000 000	4 241	2 246

Проаналізувавши отримані значення коефіцієнта прискорення та коефіцієнта ефективності для різних обсягів вхідних даних і різної кількості потоків (табл. 1), можна зробити такі висновки.

- При малих розмірах масивів показники прискорення та ефективності залишаються низькими, оскільки накладні витрати на створення та синхронізацію потоків перевищують виграш від паралельної обробки. У цьому випадку збільшення кількості потоків практично не впливає на результат і навіть може уповільнювати виконання.
- Зі збільшенням розміру масиву спостерігається помітне зростання коефіцієнта прискорення, оскільки обсяг роботи між потоками розподіляється ефективніше. Водночас коефіцієнт ефективності зменшується через нерівномірність навантаження, взаємодію потоків та додаткові витрати на злиття трьох підмасивів.
- Подальше збільшення кількості потоків для великих обсягів даних призводить до поступового насичення продуктивності: приріст прискорення стає мінімальним, а ефективність продовжує знижуватися. Це вказує на те, що оптимальна кількість потоків обмежена апаратними можливостями та структурою алгоритму.

Отже, основними чинниками, які впливають на зміну часу виконання програми, коефіцієнта прискорення та коефіцієнта ефективності, є розмір вхідних даних, кількість використаних потоків, а також особливості самого алгоритму 3-way Merge Sort. Якщо замінити даний алгоритм на альтернативний метод сортування або змінити механізм паралелізації (наприклад, застосувати іншу модель розподілу підзадач), значення досліджуваних показників можуть суттєво покращитися або, навпаки, погіршитися.

Висновки

У роботі досліджено алгоритм сортування 3-way Merge Sort як один із ефективних методів обробки великих масивів даних та визначено основні сфери його застосування в сучасних обчислювальних системах. Розглянуто принципи роботи алгоритму тришляхового злиття, проаналізовано його переваги порівняно з класичним двошляховим злиттям, а також особливості його використання у паралельних обчисленнях. На основі літературних джерел проаналізовано підходи до розпаралелювання рекурсивних алгоритмів сортування та особливості організації потоків у багатоядерних системах.

У програмній частині реалізовано паралельний алгоритм 3-way Merge Sort засобами мови C#, детально описано структуру коду, логіку розподілу завдань між потоками та механізм злиття трьох підмасивів. Побудовано UML-діаграми, що відображають архітектуру розробленого програмного модуля. Проведено тестування роботи алгоритму на наборах даних різних розмірів, виконано аналіз отриманих результатів, зокрема розраховано коефіцієнти прискорення та ефективності.

Дослідження показало, що збільшення кількості потоків призводить до зростання коефіцієнта прискорення, однак супроводжується зменшенням коефіцієнта ефективності через зростання накладних витрат на синхронізацію та нерівномірність навантаження. Таким чином, паралельна реалізація 3-way Merge Sort є перспективною для роботи з великими масивами даних, проте потребує раціонального вибору кількості потоків для досягнення максимальної продуктивності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Алгоритми: побудова і аналіз: Пер. з англ. Київ: ВСВ «Мегатайп», 2017. 1296 с.
2. Седжвік Р., Вейн К. Алгоритми. Повний курс. 4-те вид. Київ: Видавнича група КМ-БУКС, 2020. 1056 с.
3. Таненбаум Е., Остін Т. Архітектура комп'ютера. Київ: Вілліамс, 2014. 752 с.
4. Дейтел П., Дейтел Х. C# для програмістів. Київ: Вільямс, 2015. 1200 с.
5. Sutter H. фундаментальний перехід до паралелізму в програмному забезпеченні [пер. назви укр.] // Dr. Dobb's Journal, 2005.
6. McCool M., Reinders J., Robison A. Структуроване паралельне програмування: шаблони для ефективних обчислень [пер. назви укр.]. Morgan Kaufmann, 2012. — 320 p.
7. Gonzalez T. Довідник з апроксимаційних алгоритмів і метаевристик [пер. назви укр.]. Chapman & Hall/CRC, 2007. 1432 p.
8. Microsoft. Task Parallel Library (TPL) Documentation. Microsoft Docs. URL: https://learn.microsoft.com/dotnet/standard/parallel-programming/

Маціпура Юлія Андріївна – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: yliamatsipyra@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.

Matsipura Yulia Andriivna – student of the Department of Computer Science, Faculty of Intellectual Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: yliamatsipyra@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