УДК 004.8

О.А.Кушнір
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ COCKTAIL SORT ЗА ДОПОМОГОЮ OPENMP

Вінницький національний технічний університет

Анотація

Розглянуто розробку паралельного алгоритму сортування Cocktail Sort із використанням багатопотокових обчислень. Проаналізовано особливості двонапрямного сортування масивів та існуючі підходи до підвищення продуктивності алгоритмів сортування шляхом паралелізації. Обґрунтовано вибір мови програмування C++ та технології OpenMP для реалізації програмного модуля, а також вибір середовища розробки. У роботі розроблено програмну реалізацію послідовної та паралельної версій алгоритму Cocktail Sort і проведено експериментальне тестування ефективності для різних розмірів масивів та типів вхідних даних. Отримані результати підтверджують доцільність використання паралельних обчислень для зменшення часу сортування та можуть бути застосовані при обробці великих масивів даних.

Ключові слова*: паралельне сортування, Cocktail Sort, багатопотокові обчислення, OpenMP, продуктивність.*

Abstract

The paper considers the development of a parallel Cocktail Sort algorithm using multithreaded computing. The features of bidirectional array sorting and existing approaches to improving the performance of sorting algorithms through parallelization are analyzed. The choice of the C++ programming language and the OpenMP technology for implementing the software module, as well as the development environment, is substantiated. A software implementation of both sequential and parallel versions of the Cocktail Sort algorithm is developed, and experimental performance testing is conducted for different array sizes and input data types. The obtained results confirm the effectiveness of parallel computing in reducing sorting time and can be applied to the processing of large data arrays.

Keywords*: parallel sorting, Cocktail Sort, multithreaded computing, OpenMP, performance.*

Вступ

Актуальність реалізації паралельного алгоритму сортування Cocktail Sort полягає в зростаючій потребі швидкого опрацювання великих обсягів даних у сучасних інформаційних системах. Операції сортування є базовими для багатьох прикладних задач, зокрема аналізу даних, обробки сигналів, фінансових обчислень та інформаційного пошуку. У таких умовах використання послідовних алгоритмів часто не забезпечує необхідного рівня продуктивності, що зумовлює доцільність застосування паралельних методів обчислень.

Алгоритм Cocktail Sort є вдосконаленням бульбашкового сортування та відрізняється двонаправленим проходом масиву, що дозволяє ефективніше усувати локальні невпорядкованості. Його структура робить алгоритм зручним для реалізації у паралельному середовищі, оскільки порівняння сусідніх елементів у межах одного проходу можуть виконуватися незалежно. Використання багатопотокових обчислень дає змогу розподілити обробку даних між кількома ядрами процесора та зменшити загальний час виконання сортування [1-3].

Метою роботи є розробка та дослідження паралельного алгоритму сортування Cocktail Sort з використанням багатопотокових обчислень для підвищення швидкодії та ефективності обробки масивів даних.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- аналіз структури та принципів роботи алгоритму сортування Cocktail Sort;

- дослідження можливостей його розпаралелювання на багатоядерних обчислювальних системах;
- розробка програмної реалізації послідовної та паралельної версій алгоритму сортування Cocktail Sort;
- оптимізація алгоритму з метою зменшення часу виконання за рахунок багатопотокової обробки;
- проведення тестування реалізованого алгоритму на масивах різного розміру та аналіз отриманих результатів.

Виклад основного матеріалу

Алгоритми сортування відіграють важливу роль у комп'ютерних науках та застосовуються в системах обробки даних, базах даних, інформаційно-пошукових системах, алгоритмах машинного навчання та чисельних методах. Впорядкування даних дозволяє зменшити складність подальших операцій, таких як пошук, аналіз або агрегація інформації. Залежно від підходу до обробки елементів, алгоритми сортування можуть мати різну часову та просторову складність, що визначає доцільність їх використання у конкретних задачах. Серед відомих алгоритмів сортування виділяють прості методи з квадратичною складністю, такі як Bubble Sort, Insertion Sort та Selection Sort, а також більш ефективні алгоритми, зокрема Quick Sort, Merge Sort та Heap Sort, які мають асимптотичну складність порядку O(n log n). Попри нижчу ефективність простих алгоритмів у послідовному виконанні, їхня структура часто є зручною для дослідження методів оптимізації та паралелізації [4-5].

Cocktail Sort являє собою двонапрямлене бульбашкове сортування. Він є модифікацією класичного алгоритму Bubble Sort. Його особливістю є виконання двох послідовних проходів масиву: прямого (зліва направо) та зворотного (справа наліво). Під час прямого проходу більші елементи поступово переміщуються до кінця масиву, а під час зворотного — менші елементи зміщуються на початок. Такий підхід дозволяє зменшити кількість ітерацій у порівнянні з класичним бульбашковим сортуванням, особливо для частково впорядкованих даних. З точки зору обчислювальної складності алгоритм Cocktail Sort у найкращому, середньому та найгіршому випадках має квадратичну оцінку $O(n^2)$. Однак його двофазна структура (forward pass та backward pass) створює передумови для ефективного розпаралелювання, оскільки порівняння та обміни виконуються між сусідніми елементами у незалежних ділянках масиву.

Паралельна реалізація алгоритму Cocktail Sort базується на розподілі операцій порівняння між кількома потоками [3]. У межах одного проходу (прямого або зворотного) кожен потік обробляє власну підмножину непересічних пар елементів, що дозволяє уникнути конфліктів доступу до пам'яті. Синхронізація потоків виконується після завершення кожної фази, що забезпечує коректність подальших обчислень. Використання багатопотокових технологій, зокрема OpenMP, дозволяє автоматизувати процес створення та керування потоками, а також мінімізувати зміни у вихідному коді. OpenMP забезпечує ефективний розподіл ітерацій циклів між потоками, підтримує механізми синхронізації та дає змогу реалізувати редукцію логічних змінних для визначення факту виконання обмінів під час фаз сортування [5].

Оптимізація паралельного алгоритму Cocktail Sort досягається за рахунок кількох підходів. По-перше, використовується механізм раннього завершення, який дозволяє припинити роботу алгоритму у разі відсутності обмінів у прямій та зворотній фазах. По-друге, рівномірний розподіл ітерацій між потоками зменшує простої та підвищує ефективність використання обчислювальних ресурсів. По-третє, локальний характер доступу до пам'яті сприяє кращій кеш-ефективності та зниженню накладних витрат на синхронізацію.

Програмна реалізація паралельного алгоритму Cocktail Sort дозволяє на практиці дослідити вплив паралелізації на алгоритми з квадратичною складністю. Отримані результати підтверджують, що хоча асимптотична складність алгоритму не змінюється, використання кількох потоків суттєво зменшує фактичний час виконання, особливо для великих розмірів вхідних масивів.

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

Таблиця 1 – Час виконання програми на різних вхідних даних

Розмір масиву	Сортування з багатопоточністю, с	Сортування одним потоком, с
1 000	0,0027	0,0066
10 000	0,2346	0,6553
20 000	0,9166	2,5909
50 000	5,6462	16,5664

Проаналізувавши результати тестування програмної реалізації паралельного алгоритму сортування Cocktail Sort для різних розмірів вхідного масиву (табл. 1), можна зробити такі висновки.

1. Для невеликих масивів час виконання послідовної та паралельної версій алгоритму є майже однаковим. У таких випадках переваги паралельної обробки практично не проявляються, оскільки витрати на створення потоків та синхронізацію між фазами алгоритму компенсують виграш від одночасного виконання операцій.
2. Зі збільшенням кількості елементів у масиві різниця між часом виконання послідовної та паралельної реалізацій стає більш помітною. Паралельна версія алгоритму демонструє менший час сортування порівняно з послідовною, що свідчить про ефективність використання багатопотокової обробки для великих обсягів даних. Це пояснюється можливістю одночасного виконання порівнянь та обмінів незалежних пар елементів у межах кожної фази алгоритму.
3. Незважаючи на зменшення фактичного часу виконання, паралелізація не змінює асимптотичної часової складності алгоритму, яка залишається квадратичною $O(n^2)$. Таким чином, застосування паралельної реалізації Cocktail Sort є доцільним переважно для середніх і великих масивів, де виграш у часі перевищує накладні витрати на організацію паралельних обчислень.

Висновки

У роботі досліджено алгоритм сортування Cocktail Sort як один із простих методів упорядкування масивів, що ґрунтується на двонаправленому проході та послідовному порівнянні сусідніх елементів. Визначено особливості структури алгоритму, його місце серед базових методів сортування та можливості застосування паралельних обчислень для підвищення продуктивності. Розглянуто принципи організації паралельної реалізації алгоритму з урахуванням чергування прямого та зворотного проходів по масиву.

Розроблено програмну реалізацію послідовної та паралельної версій алгоритму Cocktail Sort мовою програмування C++ із використанням технології OpenMP [5, 6]. Описано основні компоненти програмного коду та побудовано UML-діаграми, які відображають структурну організацію та логіку виконання алгоритму. Проведено тестування розробленої програми для різних розмірів вхідних масивів та виконано порівняльний аналіз часу виконання послідовної і паралельної реалізацій.

Результати експериментів показали, що для невеликих масивів переваги паралельної обробки є незначними, тоді як для більших обсягів даних паралельна реалізація дозволяє зменшити фактичний час сортування. Водночас встановлено, що паралелізація не змінює асимптотичної часової складності алгоритму, яка залишається квадратичною $O(n^2)$. Отримані результати підтверджують доцільність використання паралельної версії Cocktail Sort для обробки масивів значного розміру та можуть бути використані з навчальною і дослідницькою метою у галузі паралельних обчислень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Алгоритми сортування в теорії та практиці. URL: https://javarush.com/ua/groups/posts/uk.1997. algoritmi-sortuvannja-v-teor-ta-na-praktic
2. A systematic analysis on performance and computational complexity of sorting algorithms. URL: https://www.researchgate.net/publication/397210223_A_systematic_analysis_on_performance_and_computationa l_complexity _of_sorting_algorithms
3. Cocktail Sort. How does it work? URL: https://www.baeldung.com/cs/cocktail- sort
4. Basics of parallel sorting algorithms. URL: https://www.meegle.com/en_us/topics/algorithm/ parallel-sorting-algorithms
5. Концепція OpenMP. URL: https://ridnij.sotka.cx.ua/shho-take-koncepciya- vidkritogo-mp/
6. Shaker Sort in C++ Sanfoundry. URL: https://www.sanfoundry.com/cpp-program-perform-shaker-sort/

Борецький Владислав Віталійович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: vladboretskyi.work@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.

Boretskyi Vladyslav Vitaliyovich – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: vladboretskyi.work@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