УДК 004.8

Ю.О. Шкляренко
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ ВИБОРОМ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ OPENCL

Вінницький національний технічний університет

Анотація

Розглянуто розробку алгоритму сортування Selection Sort та досліджено особливості його послідовної й паралельної реалізації з використанням технології OpenCL. У роботі проведено аналіз існуючих підходів до реалізації алгоритмів сортування з метою підвищення продуктивності обчислень, обґрунтовано вибір засобів розробки програмного модуля мовою C++, а також бібліотеки OpenCL для реалізації паралельних обчислень. Розроблено діаграми класів програмного модуля та обґрунтовано вибір програмного середовища реалізації. Створено програмну реалізацію алгоритму Selection Sort у послідовному та паралельному варіантах і проведено порівняльний аналіз часу їх виконання. Здійснено тестування програмного модуля на різних типах масивів даних. Отримані результати дозволяють оцінити переваги паралельного сортування, підвищити наочність алгоритму та підтверджують доцільність використання розробленого програмного модуля в навчальних і практичних задачах.

Ключові слова: *Selection Sort, паралельне сортування, C++, порівняння часу виконання.*

Abstract

The development of the Selection Sort algorithm is considered, and the features of its sequential and parallel implementations using OpenCL technology are studied. The issue of analyzing existing approaches to sorting algorithm implementation in order to improve computational performance is addressed, the choice of tools for developing a software module in C++ and the OpenCL library for parallel computing is justified, class diagrams of the software module are developed, and the choice of the software implementation environment is substantiated. In the work, software implementations of the Selection Sort algorithm in sequential and parallel forms are created, and a comparative analysis of their execution time is performed. The software module is tested on various types of data arrays. The obtained results allow evaluating the advantages of parallel sorting and confirm the feasibility of using the developed software module for educational and practical tasks.

Keywords: *Selection Sort, parallel sorting, C++, performance comparison.*

Вступ

Актуальність реалізації алгоритму сортування вибором полягає в тому, що ефективність сортування є критично важливою для сучасних задач обробки великих обсягів даних. У таких сферах, як аналіз даних, інформаційні системи, наукові обчислення та програмні комплекси реального часу, використання традиційних послідовних алгоритмів сортування може призводити до значних витрат часу та зниження загальної продуктивності програмного забезпечення. Застосування паралельних підходів до сортування дозволяє суттєво прискорити обробку великих масивів даних і підвищити ефективність обчислень.

Алгоритм Selection Sort належить до класу алгоритмів вибору та ґрунтується на пошуку максимального елемента масиву з подальшим його переміщенням у відсортовану частину. Застосування паралельних обчислювальних архітектур дозволяє здійснювати пошук максимальних елементів одночасно в різних сегментах масиву, скорочуючи час виконання алгоритму без зміни його базової логіки.

Метою роботи є розробка та дослідження паралельного алгоритму сортування Selection Sort з використанням сучасних засобів паралельного програмування для підвищення швидкодії та продуктивності обчислювальних систем, а також оцінка переваг його застосування у навчальних і практичних задачах.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань: розробка ефективної паралельної реалізації Selection Sort для оптимального розподілу обчислень; мінімізація затримок передачі даних між вузлами; розробка програми для обчислення масивів даних за допомогою паралельного Selection Sort; тестування програми та аналіз отриманих результатів.

Виклад основного матеріалу

Алгоритми сортування широко використовуються для впорядкування даних у базах даних, інформаційних системах, програмних комплексах аналізу даних, а також у багатьох прикладних задачах обробки великих масивів інформації. Сортування є базовою операцією, від ефективності якої залежить швидкодія подальших обчислень [1-3]. Серед класичних методів сортування виділяють алгоритм Selection Sort, який характеризується простою реалізацією та часовою складністю $O(n^2)$, що обмежує його продуктивність при роботі з масивами великого розміру [4]. У зв'язку з цим актуальним є застосування паралельних та розподілених обчислень для підвищення ефективності виконання операцій сортування [5].

Для того, щоб правильно визначити ідею використання паралельних підходів у реалізації алгоритму Selection Sort проведено аналіз сучасних архітектур обчислювальних систем та методів паралельної обробки даних. Паралельна реалізація передбачає розбиття масиву на окремі сегменти та одночасний пошук екстремальних елементів у цих сегментах із подальшим об'єднанням результатів. Такий підхід дозволяє зменшити кількість послідовних операцій і скоротити загальний час виконання алгоритму без зміни його базової логіки [6].

Паралельні обчислювальні системи, що базуються на багатоядерних процесорах та графічних прискорювачах, забезпечують ефективне виконання однотипних операцій над великими обсягами даних. Використання таких систем дає змогу рівномірно розподіляти обчислювальне навантаження між потоками або обчислювальними елементами, що підвищує продуктивність сортування та стабільність роботи програмного забезпечення.

На відміну від послідовної реалізації, де всі операції виконуються по черзі, паралельна реалізація Selection Sort дозволяє виконувати частину обчислень одночасно, використовуючи можливості сучасних апаратних платформ. Це сприяє скороченню часу обробки великих масивів даних і робить алгоритм більш придатним для використання в сучасних обчислювальних системах та навчальних програмних продуктах[7].

Алгоритми сортування вибором можуть бути шести різних типів:

- послідовний Selection Sort;
- Selection Sort з пошуком мінімального елемента;
- Selection Sort з пошуком максимального елемента;
- двосторонній Selection Sort;
- паралельний Selection Sort;
- Selection Sort для розподілених систем;

Основні переваги та ідеї застосування паралельної реалізації Selection Sort включають:

- розподіл масиву даних на незалежні сегменти, що дозволяє виконувати пошук мінімальних або максимальних елементів одночасно в різних частинах масиву та зменшує час виконання алгоритму;
- можливість паралельного виконання операцій порівняння, що забезпечує ефективне використання багатоядерних процесорів та паралельних обчислювальних пристроїв;
- масштабованість алгоритму за рахунок збільшення кількості потоків або обчислювальних елементів без зміни його базової логіки;

- зменшення навантаження на центральний процесор шляхом розподілу обчислювальних операцій між кількома обчислювальними ресурсами, що підвищує загальну продуктивність системи;
- скорочення загального часу сортування великих масивів даних за рахунок паралельної обробки та зменшення кількості послідовних операцій.

Розглянуто та використано чотири основних способи оптимізації алгоритму сортування вибором шляхом його паралельної реалізації.

1. За рахунок паралелізації обчислень. Виконано розподіл масиву даних на окремі сегменти, кожен з яких обробляється незалежно. Пошук мінімальних або максимальних елементів здійснюється паралельно в кожному сегменті, що дозволяє кожному обчислювальному потоку визначати локальні екстремальні значення та зменшити загальний час виконання алгоритму.
2. За рахунок архітектури паралельних обчислень. Обчислювальні елементи повинні мати можливість обмінюватися проміжними результатами для визначення глобального мінімального або максимального елемента. Використання ефективних механізмів синхронізації та швидкого обміну даними між потоками забезпечує коректність і високу швидкодію паралельної реалізації.
3. За рахунок використання оптимізованих алгоритмічних підходів. Застосовано модифіковані варіанти Selection Sort, зокрема двосторонній пошук екстремальних елементів та скорочення кількості порівнянь. Також використано блокову обробку масиву, що дозволяє підвищити ефективність роботи з пам’яттю та зменшити накладні витрати.
4. За рахунок взаємодії між обчислювальними потоками. Після завершення локальних обчислень виконується обмін результатами для визначення глобального екстремального елемента та його переміщення у відповідну позицію відсортованої частини масиву. Синхронізація потоків забезпечує правильність виконання алгоритму та запобігає конфліктам при доступі до спільних даних.

Програмно реалізовано задану задачу та описано всі компоненти коду.

Аналіз результатів тестування програми виконано для різної кількості елементів масиву (табл.1-3).

Таблиця 1 – Час виконання програми на різних вхідних даних

Тип обчислень	Загальна розмірність масиву		
	50	5 000	10 000
Паралельний	0.769	2.721	3.396
Послідовний	0.010	86.842	345.796

Таблиця 2 – Коефіцієнти прискорення

Загальна розмірність масиву	500	1000	5000
Коефіцієнт прискорення	0	31.9154	101.816

Таблиця 3 – Коефіцієнти ефективності

Кількість процесів	2	4	8
Коефіцієнт ефективності	0.3411	0.26305	0.1215

Проаналізувавши обчислені коефіцієнти прискорення та коефіцієнти ефективності при різних вхідних даних і різних кількостях процесів (табл.1-3) можливо зробити висновки:

- при невеликих вхідних даних значення коефіцієнта прискорення та коефіцієнта ефективності є близькими до нуля, тому кількість процесів при виконанні програми з відносно невеликими вхідними даними фактично не має значення;
- збільшення кількості процесів при виконанні задачі з великими вхідними даними призводить до відносно незначного збільшення коефіцієнта прискорення, проте значного зменшення коефіцієнта ефективності.

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 невідсортованого масиву даних, кількість процесів у програмній реалізації і також метод виконання сортування, якщо замінити розглянутий паралельний метод на будь-який інший, якість досліджуваних показників може відповідно погіршитись.

Висновки

Досліджено алгоритм сортування Selection Sort як один із базових методів впорядкування даних та визначено основні сфери його застосування. Розглянуто послідовну та паралельну реалізації алгоритму, описано принципи їх роботи, особливості виконання та обмеження. На основі аналізу літературних джерел досліджено паралельні обчислювальні архітектури та методи паралелізації, що дозволило сформулювати підхід до паралельної реалізації алгоритму Selection Sort і оцінити доцільність його використання в сучасних обчислювальних системах.

Програмно реалізовано версію алгоритму сортування Selection Sort та детально описано її основні компоненти. Також розроблено UML-діаграми програмного модуля. Проведено тестування розробленого програмного забезпечення, проаналізовано отримані результати та досліджено показники прискорення й ефективності паралельної реалізації.

Визначено, що збільшення кількості паралельних обчислювальних елементів призводить до обмеженого зростання коефіцієнта прискорення та зниження коефіцієнта ефективності, що зумовлено алгоритмічними особливостями Selection Sort і наявністю послідовних етапів у його структурі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Parallel Sorting – University of Porto, Faculty of Computer Science. URL: https://www.dcc.fc.up.pt/~ricroc/aulas/1516/cp/apontamentos/slides_sorting.pdf.
2. Selection Sort Algorithm in C++ . URL: https://www.geeksforgeeks.org/dsa/selection-sort-algorithm- 2/.
3. Singh R., Sharma P. Parallel Sorting Algorithms: Performance Evaluation and Comparative Analysis. Electronic edition. IEEE Xplore, 2022.
4. Дорошенко А. Ю., Андон П. І., Яценко О. А., Жереб К. А. Алгеброалгоритмічні моделі та методи паралельного програмування. Київ: Видавничий дім «Академперіодика», 2018. 192 с.
5. Минайленко Р.М. Паралельні та розподілені обчислення: Навчальний посібник.Кропивницький: Видавець Лисенко В. Ф., 2021. URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/396e02d2-725b-47b5-a1c0-ae07a9bec326/content .
6. Selim G. Akl. Parallel Sorting Algorithms. San Diego: Academic Press, 1993. 300 с.
7. Rajasekaran S., Reif J. (Eds.). Handbook of Parallel Computing: Models, Algorithms and Applications. Boca Raton: CRC Press, 2010. 800 с.

Шкляренко Юрій Олександрович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: ashklayrenkoo@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.

Shkliarenko Yurii Olexandrovich – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: ashklayrenkoo@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