УДК 004.8

Д.О.Трачук
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ ШВИДКОГО СОРТУВАННЯ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ CUDA

Вінницький національний технічний університет

Анотація

Розглянуто розробку та програмну реалізацію паралельного алгоритму швидкого сортування з використанням технології CUDA. Проаналізовано існуючі методи сортування з точки зору досягнення високої обчислювальної продуктивності та обґрунтовано вибір інструментів і підходів для створення програмного модуля. Розроблено структуру програмного забезпечення, зокрема діаграми класів, а також визначено доцільне програмне середовище реалізації. У роботі реалізовано паралельний алгоритм із використанням бібліотеки PyTorch та технології CUDA і проведено експериментальне дослідження його продуктивності на наборах даних різного обсягу. Отримані результати підтверджують можливість суттєвого підвищення швидкодії та ефективності програмних систем, що здійснюють обробку великих масивів даних.

Ключові слова: *швидке сортування, CUDA, GPU, паралельні обчислення, продуктивність.*

Abstract

The development and software implementation of a parallel quick sort algorithm using CUDA technology are considered. Existing sorting methods are analyzed in terms of achieving high computational performance, and the selection of tools and approaches for software module development is justified. The software structure, including class diagrams, is designed, and an appropriate implementation environment is determined. A parallel algorithm is implemented using PyTorch and CUDA, followed by experimental performance evaluation on datasets of different sizes. The obtained results confirm the possibility of significantly improving the execution speed and efficiency of software systems that process large data arrays.

Keywords: *quick sort, CUDA, GPU, parallel computing, performance.*

Вступ

Актуальність розробки паралельного алгоритму швидкого сортування з використанням технології CUDA зумовлена зростаючими вимогами до продуктивності сучасних обчислювальних систем. Однією з ключових особливостей CUDA є можливість значного прискорення обчислень за рахунок використання тисяч ядер графічного процесора, що є критично важливим під час обробки великих масивів даних у таких сферах, як аналіз Big Data, обробка баз даних, комп'ютерна графіка та наукові обчислення.

Застосування паралельних архітектур GPU дозволяє одночасно виконувати сортування різних підмасивів на багатьох обчислювальних ядрах, що суттєво зменшує загальний час виконання алгоритму. Такий підхід забезпечує високу масштабованість і продуктивність при розв'язанні обчислювально складних задач, зокрема у галузях науки про дані, інженерії та фінансової аналітики. Технологія CUDA орієнтована на масивно-паралельні обчислення, у яких кожен потік GPU виконує окрему частину алгоритму сортування, а результати обчислень поєднуються у кінцевий відсортований масив. Це робить CUDA ефективним інструментом для підвищення швидкодії та продуктивності програмних рішень у сучасних обчислювальних системах.

Метою роботи є дослідження та розробка паралельного алгоритму швидкого сортування з використанням технології CUDA з метою підвищення швидкодії та ефективності обробки великих масивів даних у сучасних обчислювальних системах.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- розробка ефективної архітектури паралельного алгоритму для оптимального розподілу обчислень між потоками GPU;
- мінімізація затримок передачі даних між пам'яттю хоста (CPU) та пристрою (GPU);
- розробка програми для виконання швидкого сортування за допомогою CUDA з використанням бібліотеки PyTorch;
- тестування програми на масивах різного розміру та аналіз отриманих результатів продуктивності.

Виклад основного матеріалу

Швидке сортування є одним із найпоширеніших алгоритмів упорядкування даних і широко використовується для обробки великих масивів у базах даних, пошукових системах та інформаційно-пошукових алгоритмах з метою оптимізації доступу до даних і підвищення швидкодії пошуку [1–3]. Класичний послідовний алгоритм quick sort має середню часову складність $O(n \log n)$, однак у найгіршому випадку його складність зростає до $O(n^2)$. Для зменшення ймовірності виникнення такого випадку застосовуються оптимізовані варіанти з вибором опорного елемента за принципом медіани [4]. Подальший розвиток алгоритмів сортування пов’язаний із використанням паралельних обчислень на графічних процесорах, що дозволяє суттєво прискорити обробку масивів з мільйонів елементів [5, 8, 9].

Для обґрунтування доцільності застосування технології CUDA в задачах паралельного сортування проведено аналіз архітектури GPU та принципів паралельного програмування, що дозволило ефективно реалізувати алгоритм швидкого сортування на графічних процесорах [6, 7].

CUDA (Compute Unified Device Architecture) — це паралельна обчислювальна платформа та модель програмування, розроблена компанією NVIDIA для використання обчислювальних можливостей графічних процесорів у задачах загального призначення [12]. GPU містить тисячі простих обчислювальних ядер, згрупованих у потокові мультипроцесори (Streaming Multiprocessors, SM). Потоки в CUDA організовані ієрархічно: окремі потоки об’єднуються у блоки, а блоки формують грід, що забезпечує ефективний розподіл обчислень та досягнення високого рівня паралелізму [5, 6].

CUDA-програми здатні одночасно обробляти мільйони елементів завдяки масивному паралелізму GPU. Kernel-функції виконуються тисячами потоків паралельно, а доступ до глобальної пам’яті оптимізується за рахунок узгоджених (coalesced) звернень. Перед початком обчислень дані копіюються з оперативної пам’яті CPU до пам’яті GPU, після чого виконання відбувається асинхронно. Незалежність потоків дозволяє ефективно використовувати апаратні ресурси навіть у випадку нерівномірного навантаження.

На відміну від традиційного багатопоточного програмування на CPU, де створення потоків супроводжується значними накладними витратами, у CUDA використовуються легкі апаратні потоки, що виконуються безпосередньо на GPU. Це забезпечує кращу масштабованість і значно вищу продуктивність у задачах обробки великих масивів даних.

CUDA-архітектура розвивалась у кількох поколіннях GPU, серед яких Fermi, Kepler, Maxwell, Pascal, Volta/Turing, Ampere та Ada Lovelace/Hopper, кожне з яких забезпечувало підвищення продуктивності, енергоефективності та розширення функціональних можливостей [13, 14].

Основні переваги використання технології CUDA для задач сортування полягають у наступному:

- масивний паралелізм, що дозволяє одночасно виконувати операції порівняння та обміну елементів тисячами потоків;
- висока пропускна здатність пам’яті GPU, яка забезпечує суттєве прискорення алгоритмів сортування порівняно з CPU;
- масштабованість, що дозволяє використовувати один і той самий код на різних поколіннях GPU;

- ефективна ієрархія пам’яті, яка підвищує продуктивність при правильному використанні локальності даних;
- зменшення затримок доступу до даних завдяки використанню shared memory та кеш-пам’яті.

У роботі розглянуто чотири основних способи оптимізації алгоритму швидкого сортування з використанням CUDA. Перший спосіб ґрунтується на паралелізації обчислень шляхом поділу масиву між блоками та потоками GPU, що дозволяє паралельно виконувати операції розбиття та сортування сегментів. Другий спосіб полягає в ефективній роботі з пам’яттю GPU із застосуванням shared memory та узгодженого доступу до глобальної пам’яті. Третій спосіб передбачає використання оптимізованих алгоритмів, зокрема bitonic sort та паралельного quick sort із вибором медіани, адаптованих для виконання на GPU. Четвертий спосіб базується на ефективній синхронізації потоків усередині блоків і між блоками для забезпечення коректності обчислень.

Програмну реалізацію алгоритму виконано з використанням бібліотеки PyTorch та технології CUDA [10, 11]. Реалізована програма забезпечує паралельне сортування великих масивів даних, вимірювання часу виконання та аналіз продуктивності. Для коректного вимірювання часу застосовано синхронізацію GPU за допомогою функції torch.cuda.synchronize().

Аналіз результатів тестування програми виконано для різної кількості елементів масиву (табл.1, 2).

Таблиця 1 – Порівняння часу виконання CPU та GPU сортування

Розмір масиву	CPU (с)	GPU (с)	Прискорення
10 000	0.1076	0.00025	524.90×
100 000	0.5427	0.01110	48.90×
1 000 000	5.6948	0.01310	433.60×
5 000 000	33.6423	0.06460	520.70×

Таблиця 2 – Детальний аналіз ефективності GPU

Розмір масиву	Час GPU (мс)	Прискорення	Пропускна здатність (млн.елем./с)
10 000	0.205	524.90×	48.78
100 000	11.099	48.90×	09.01
1 000 000	13.134	433.60×	76.14
5 000 000	64.608	520.70×	77.39

Проаналізувавши результати тестування (табл. 1, 2), можна зробити такі висновки:

- для всіх розмірів масивів використання GPU забезпечує суттєве прискорення порівняно з CPU;
- найвища ефективність досягається при сортуванні масивів середнього та великого розміру;
- накладні витрати на організацію паралельних обчислень менш суттєві при збільшенні обсягу вхідних даних.

Отже, основними чинниками, що впливають на продуктивність паралельного швидкого сортування, є розмір масиву, архітектура GPU, ефективність роботи з пам’яттю та обраний алгоритм сортування. Використання технології CUDA дозволяє суттєво зменшити час виконання алгоритму та підвищити ефективність обробки великих обсягів даних.

Висновки

У ході виконання роботи досліджено алгоритм швидкого сортування як один із базових алгоритмів упорядкування даних, визначено основні сфери його застосування та проаналізовано особливості реалізації в умовах паралельних обчислень. Розглянуто принципи використання технології CUDA та архітектури графічних процесорів для прискорення обробки великих масивів даних, а також обґрунтовано доцільність застосування GPU для задач сортування.

Програмно реалізовано паралельний алгоритм швидкого сортування з використанням бібліотеки PyTorch та технології CUDA, описано основні компоненти програмної реалізації та принципи організації обчислень на GPU. Проведено тестування розробленої програми для масивів різного розміру та виконано аналіз отриманих результатів, зокрема часу виконання та коефіцієнтів прискорення.

Встановлено, що використання графічного процесора забезпечує значне прискорення сортування порівняно з послідовною реалізацією на CPU, причому ефективність паралелізації зростає зі збільшенням розміру вхідних даних. Для масивів великої розмірності досягнуто стабільно високі значення коефіцієнта прискорення, що підтверджує доцільність застосування GPU для обробки великих обсягів даних. Також підтверджено коректність роботи алгоритму для всіх досліджуваних випадків.

Визначено, що основними чинниками, які впливають на ефективність паралельного сортування, є розмір вхідних даних, архітектура графічного процесора, ефективність обміну даними між CPU та GPU, а також використання оптимізованих бібліотечних функцій. Таким чином, застосування технології CUDA у поєднанні з високорівневими інструментами програмування дозволяє ефективно вирішувати задачі сортування великих масивів даних у сучасних обчислювальних системах.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кнут Д. Е. Мистецтво програмування, том 3: Сортування та пошук. 2-е видання. Вільямс, 2000. 824 с.
2. Новотарський М. А. Алгоритми та методи обчислень. Київ: КПІ ім. Ігоря Сікорського, 2019. 407 с.
3. Седжвік Р. Алгоритми на C++. Аналіз, структури даних, сортування, пошук. Діасофт, 2001. 496 с.
4. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms, Third Edition. MIT Press, 2009. 1312 p.
5. Kirk D. B., Hwu W. W. Programming Massively Parallel Processors: A Hands-on Approach, Third Edition. Morgan Kaufmann, 2016. 560 p.
6. Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional, 2010. 312 p.
7. Мінайленко Р. М. Паралельні та розподілені обчислення: Навчальний посібник. Кропивницький: Видавець Лисенко В. Ф., 2021. 153 с.
8. Satish N., Harris M., Garland M. Designing Efficient Sorting Algorithms for Manycore GPUs // IEEE International Symposium on Parallel & Distributed Processing. 2009. DOI: 10.1109/IPDPS.2009.5161005
9. Cederman D., Tsigas P. GPU-Quicksort: A practical Quicksort algorithm for graphics processors // Journal of Experimental Algorithmics. 2010. Vol. 14. Article 1.4.
10. Paszke A., Gross S., Massa F., et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library // Advances in Neural Information Processing Systems. 2019. Vol. 32.
11. PyTorch documentation — PyTorch 2.9 documentation [Електронний ресурс] Режим доступу: https://pytorch.org/docs/stable/index.html
12. NVIDIA CUDA C Programming Guide [Електронний ресурс] Режим доступу: https://docs.nvidia.com/cuda/cuda-c-programming-guide/
13. CUDA Toolkit Documentation 12.8 [Електронний ресурс] Режим доступу: https://docs.nvidia.com/cuda/archive/12.8.0/
14. PyTorch CUDA Semantics [Електронний ресурс] Режим доступу: https://pytorch.org/docs/stable/notes/cuda.html

Трачук Дмитро Олександрович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: tdofficial05@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.

Trachuk Dmytro Olexandrovich – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: tdofficial05@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