УДК 004.8

В.Ю. Шевчук
В.О. Денисюк

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ ЗЛИТТЯМ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ CUDA

Вінницький національний технічний університет

Анотація

Розглянуто розробку паралельного алгоритму сортування злиттям за допомогою архітектури CUDA з використанням графічних процесорів (GPU). Розглянуто питання аналізу існуючих методів сортування для досягнення високої продуктивності, обґрунтовано вибір засобів розробки програмного модуля, розроблено діаграми класів програмного модуля, обґрунтовано вибір програмного середовища реалізації. У роботі створено програмну реалізацію паралельного алгоритму з використанням CUDA та проведено тестування його продуктивності на різних наборах даних. Використання результатів дозволить покращити швидкодію і продуктивність програм та алгоритмів, які потребують обробки та впорядкування великих об'ємів даних.

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

Abstract

The development of a parallel merge sort algorithm using CUDA architecture and GPUs is considered. The issue of analyzing existing sorting methods to achieve high performance is considered, the choice of software module development tools is justified, class diagrams of the software module are developed, and the choice of software implementation environment is justified. In the work, a software implementation of a parallel algorithm using CUDA is created and its performance is tested on various data sets. Using the results will allow improving the speed and performance of programs and algorithms that require processing and ordering large amounts of data.

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

Вступ

Актуальність реалізації сортування злиттям за допомогою технології CUDA полягає у тому, що особливістю паралельних обчислень на GPU є значне прискорення операцій над масивами, а це є критично важливим для сучасних задач обробки великих обсягів даних. Таких як – робота з базами даних, обробка сигналів, системне програмування та аналітика Big Data. Використання масивно-паралельних архітектур дозволяє одночасно виконувати операції порівняння та злиття елементів на тисячах ядер графічного процесора, що суттєво скорочує час виконання операцій. Це забезпечує масштабованість і високу продуктивність для розв'язання складних алгоритмічних задач у таких сферах, як наука, інженерія та IT-індустрія.

Технологія CUDA може використовуватися для розподіленого обчислення всередині відеокарти, де кожен потік виконує частину порівнянь, і результати об'єднуються в кінцевий відсортований масив, що є особливо корисним в сучасних системах, де паралелізм може покращити швидкодію та продуктивність.

Метою роботи є розробка паралельного алгоритму, що реалізує сортування злиттям за допомогою CUDA для покращення швидкодії та продуктивності обчислювальних систем.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань: розробка ефективної архітектури взаємодії потоків CUDA для розподілу даних; мінімізація затримок передачі даних між пам'яттю хоста (CPU) та пристрою (GPU); розробка програми для виконання сортування злиттям за допомогою CUDA; тестування програми та аналіз отриманих результатів.

Виклад основного матеріалу

Сортування злиттям використовується для впорядкування великих масивів даних, організації пошуку та оптимізації роботи алгоритмів у багатьох сферах комп'ютерних наук [1-3]. Методи розв'язання включають: стандартний послідовний алгоритм зі складністю $O(n\log n)$; алгоритми швидкого сортування; а також більш ефективні методи в розподілених та паралельних обчисленнях для швидшого виконання операцій над великими масивами [5].

Для того, щоб правильно визначити ідею використання CUDA в паралельних обчисленнях, виконано дослідження у рамках архітектури графічних процесорів, що допоможе використати паралелізм для сортування злиттям [6, 7].

CUDA (Compute Unified Device Architecture) – це архітектура паралельних обчислень від NVIDIA, що дозволяє використовувати графічний процесор для обчислень загального призначення. Процесор у системі виступає в ролі керуючого вузла (host), а GPU – як обчислювальний пристрій (device). Кожен потік у CUDA виконує ідентичний код (кернел) над різними даними. Поєднання тисяч потоків дозволяє інформації оброблятися паралельно без значних затримок. CUDA-мережі також називають «масивно-паралельними», оскільки вони дозволяють масштабувати обчислення відповідно до кількості ядер GPU [6, 7].

На відміну від традиційного послідовного виконання на CPU, архітектура CUDA дозволяє створювати ієрархію потоків (threads), блоків (blocks) та сіток (grids). Оскільки сортування злиттям має рекурсивну природу, воно ідеально підходить для ітеративної паралельної реалізації на GPU. CUDA допомагає досягти стабільного прискорення при роботі з мільйонами елементів.

Основні переваги та ідеї застосування CUDA включають такі критерії: локальний обмін даними, тобто використання роздільної пам'яті (shared memory) всередині блоків, що сприяє швидкому обміну даними та зменшенню навантаження на глобальну пам'ять; завдяки тисячам ядер, можливе паралельне виконання порівнянь на різних ділянках масиву, що дозволяє розподіляти завдання та прискорювати обчислення; структура CUDA легко масштабується, додавання нових обчислювальних одиниць у сучасних GPU дозволяє підвищувати продуктивність без зміни коду; використання пам'яті, локальний характер обміну даними в межах одного SM (Streaming Multiprocessor) робить алгоритм менш чутливим до затримок шини PCI-E.

Розглянуто та використано чотири основні способи оптимізації алгоритму сортування злиттям за допомогою CUDA.

1. За рахунок паралелізації обчислень. Розподіл даних — масив ділиться на сегменти, які сортуються окремо в різних блоках потоків. Використання кернелів — обчислюються етапи злиття паралельно, що дозволяє кожному блоку формувати впорядковані підмасиви.
2. За рахунок архітектури пам'яті. Використання Shared Memory дозволяє потокам всередині одного блоку швидко обмінюватися елементами під час фази злиття. Це мінімізує звернення до повільної глобальної пам'яті відеокарти.
3. За рахунок використання специфічних алгоритмів. Використання ітеративного підходу до злиття (bottom-up merge sort), який дозволяє уникнути рекурсії, що є неефективною для архітектури GPU. Більш складні схеми злиття адаптовані для паралельних обчислень у CUDA.
4. За рахунок взаємодії між потоками. Синхронізація потоків (_syncthreads()) після кожного етапу злиття забезпечує правильність обчислень і уникнення конфліктів при доступі до даних у пам'яті.

Програмно реалізовано задану задачу та описано всі компоненти коду [8].

Аналіз результатів тестування програми виконано для різної кількості елементів масиву (табл.1-3). Проаналізувавши обчислені коефіцієнти прискорення та коефіцієнти ефективності при різних вхідних даних і різних кількостях потоків (табл. 1-3), можливо зробити висновки:

- при невеликих вхідних даних (малі масиви) витрати на передачу даних між CPU та GPU нівелюють виграш у швидкості, тому прискорення є мінімальним;
- збільшення кількості елементів масиву призводить до зростання коефіцієнта прискорення, оскільки паралельна архітектура GPU краще розкривається на великих обсягах даних;
- надмірне збільшення кількості потоків при сталих даних може призвести до зниження ефективності через накладні витрати на синхронізацію.

Отже, основними чинниками, які впливають на зміну розглянутих показників, є розмір вхідного масиву, кількість блоків/потоків у сітці CUDA та ефективність використання спільної пам'яті.

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

Кількість потоків (паралелізм)	Кількість елементів		
	100 000	1 000 000	10 000 000
256	1.8543	15.4210	165.723
512	1.4231	10.1853	112.933
1 024	1.5602	9.5974	98.197

Таблиця 2 – Коефіцієнти прискорення

Кількість елементів	100 000	1 000 000	10 000 000
Коефіцієнт прискорення	2.14	8.45	24.12

Таблиця 3 – Коефіцієнти ефективності

Кількість потоків	256	512	1 024
Коефіцієнт ефективності	0.4215	0.2105	0.1182

Висновки

Досліджено сортування злиттям як одну із ключових операцій в алгоритміці. Розглянуто методи реалізації операції сортування, описано принципи роботи CUDA та її застосування для паралельних обчислень. На основі літературних джерел досліджено архітектуру GPU як складову частину сучасних обчислювальних систем. Програмно реалізовано паралельний алгоритм сортування злиттям та описано всі компоненти коду. Проведено тестування, проаналізовано результати, досліджено коефіцієнти прискорення та ефективності. Визначено, що для досягнення максимальної продуктивності на GPU необхідно оперувати великими обсягами даних, щоб компенсувати час на пересилання даних через шину пам'яті.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кнут Д. Е. Мистецтво програмування. Том 3. Сортування та пошук. 2007.
2. Сандерс Дж., Кендрот Е. CUDA за прикладами: Вступ до програмування GPU. 2011.
3. Кормен Т., Лейзерсон Ч. Вступ до алгоритмів. Київ: К.І.С., 2019.
4. Паралельні алгоритми сортування URL: https://studfile.net/preview/5740087/page:19/
5. Минайленко Р.М. Паралельні та розподілені обчислення: Навчальний посібник. 2021.
6. NVIDIA CUDA C++ Programming Guide. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/
7. Introduction to Parallel Computing with GPU. 2022.
8. Paul Norvig: Merge Sort Implementation. URL: https://scispace.com/papers/design-patterns-for-sorting-algorithms-pqy106x5h9

Шевчук Вероніка Юріївна– студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: veronikashevchuk2525@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.

Shevchuk Veronika Yuriyivna – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: oleksandrkusnir25@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