УДК 004.8

Д.Р.Долішняк
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ ШЕЛЛА ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ DIRECT COMPUTE

Вінницький національний технічний університет

Анотація

У роботі розглянуто розробку та програмну реалізацію паралельного алгоритму сортування Шелла з використанням технології DirectCompute, що входить до складу API DirectX 11. Проведено аналіз існуючих алгоритмів сортування та підходів до їх паралельної реалізації, обґрунтовано вибір алгоритму Шелла як такого, що має природні передумови до ефективного розпаралелювання. Детально досліджено архітектурні особливості графічних процесорів та модель виконання обчислювальних шейдерів у DirectCompute. Розроблено програмну систему, що включає еталонну CPU-реалізацію та GPU-реалізацію з використанням compute shader, структурованих буферів і constant buffer. Проведено тестування коректності та продуктивності реалізованого алгоритму на різних розмірах вхідних даних і виконано порівняння результатів CPU та GPU. Отримані результати підтверджують доцільність застосування DirectCompute для прискорення операцій сортування великих масивів даних.

Ключові слова: *алгоритм Шелла, паралельні обчислення, GPU, DirectX 11, DirectCompute, compute shader, сортування, високопродуктивні обчислення.*

Abstract

This paper considers the development and software implementation of a parallel Shell sort algorithm using DirectCompute technology, which is part of the DirectX 11 API. Existing sorting algorithms and approaches to their parallelization are analyzed, and the choice of Shell sort is justified due to its natural suitability for parallel execution. The architectural features of modern GPUs and the execution model of compute shaders in DirectCompute are examined in detail. A software system is developed that includes a reference CPU implementation and a GPU implementation based on compute shaders, structured buffers, and a constant buffer. Correctness and performance testing are performed for different input sizes, and a comparison between CPU and GPU results is carried out. The obtained results confirm the effectiveness of using DirectCompute to accelerate sorting operations on large datasets.

Keywords: *Shell sort algorithm, parallel computing, GPU, DirectX 11, DirectCompute, compute shader, sorting, high-performance computing.*

Вступ

У сучасних умовах стрімкого розвитку інформаційних технологій обсяги даних, що потребують обробки, зростають експоненційно. Однією з базових операцій, без якої неможлива ефективна робота інформаційних систем, є операція сортування. Сортування використовується в базах даних, пошукових і рекомендаційних системах, чисельних методах, задачах машинного навчання та аналізу великих даних.

Класичні послідовні алгоритми сортування демонструють обмежену ефективність при роботі з великими масивами даних. У зв'язку з цим особливої актуальності набувають методи паралельних обчислень, які дозволяють задіяти апаратні ресурси сучасних багатоядерних процесорів та графічних прискорювачів.

Алгоритм сортування Шелла є вдосконаленням методу сортування вставками і базується на поступовому зменшенні інтервалу між порівнюваними елементами [1, 2]. Завдяки розбиттю масиву на незалежні підпослідовності він має природні передумови для паралельної реалізації. Технологія DirectCompute, що входить до складу DirectX 11, надає можливість використовувати GPU для виконання загальних обчислень поза графічним рендерингом, що робить її придатною для реалізації паралельних алгоритмів сортування.

Метою роботи є розробка та дослідження паралельної реалізації алгоритму сортування Шелла з використанням технології DirectCompute, а також аналіз ефективності застосування GPU для задач сортування великих масивів даних.

Постановка задачі дослідження

Для досягнення поставленої мети у роботі необхідно вирішити такі задачі:

- проаналізувати існуючі алгоритми сортування та можливості їх паралельної реалізації;
- дослідити математичні та алгоритмічні властивості сортування Шелла;
- обґрунтувати вибір технології DirectCompute як середовища паралельної реалізації;
- розробити програмну реалізацію алгоритму Шелла на CPU та GPU;
- провести тестування коректності та продуктивності реалізованого алгоритму;
- виконати аналіз отриманих результатів і сформулювати висновки щодо ефективності використання GPU.

Виклад основного матеріалу

Алгоритм Шелла базується на ідеї багаторазового застосування сортування вставками до підпослідовностей масиву, сформованих з певним інтервалом. На початкових етапах використовуються великі значення інтервалу, що дозволяє швидко усунути віддалені інверсії. У міру зменшення інтервалу алгоритм поступово наближається до класичного сортування вставками, але з уже частково впорядкованим масивом. Ефективність алгоритму значною мірою залежить від вибору послідовності інтервалів [1, 2]. У практичних реалізаціях часто застосовуються послідовності Шелла, Хіббарда або Седжвіка. Середня обчислювальна складність алгоритму Шелла становить від $O(n \log^2 n)$ до $O(n^2)$ залежно від обраної послідовності інтервалів. Ключовою особливістю алгоритму є те, що сортування для фіксованого інтервалу може виконуватися незалежно для кожної підпослідовності [2, 3]. Це створює умови для ефективного розпаралелювання, оскільки відсутні конфлікти доступу до даних між різними підпослідовностями.

DirectCompute є складовою частиною DirectX 11 і надає доступ до обчислювальних можливостей GPU через механізм compute shader [4, 5]. Модель виконання DirectCompute ґрунтується на концепції SIMT, у межах якої велика кількість потоків виконує одну й ту саму програму над різними даними. Для реалізації паралельного сортування Шелла у DirectCompute кожен GPU-потік відповідає за сортування окремої підпослідовності масиву з фіксованим інтервалом [3, 4]. Дані зберігаються у структурованому буфері, доступному для читання і запису через Unordered Access View. Параметри алгоритму, зокрема розмір масиву та поточний інтервал, передаються у compute shader за допомогою constant buffer. Такий підхід дозволяє повністю уникнути синхронізації між потоками, оскільки кожна підпослідовність обробляється незалежно. У результаті досягається високий рівень паралелізму та ефективне використання обчислювальних ресурсів GPU.

Для оцінювання ефективності реалізованого алгоритму проведено серію експериментів на різних розмірах вхідних масивів. Порівнювалися час виконання та коректність результатів для CPU- та GPU-реалізацій. Результати тестування показали, що для невеликих масивів різниця у часі виконання між CPU та GPU є незначною через накладні витрати на ініціалізацію GPU та передачу даних [4, 5]. Однак зі зростанням розміру масиву GPU-реалізація демонструє суттєве прискорення порівняно з послідовним виконанням на CPU [3, 4]. Це підтверджує доцільність використання DirectCompute для задач сортування великих обсягів даних.

Результати тестування для різних розмірів вхідних масивів та коефіцієнти прискорення надані у таблицях 1, 2.

Як видно з таблиці 1, при малих розмірах масиву GPU-реалізація поступається CPU через накладні витрати на ініціалізацію обчислювального конвеєра. Проте зі збільшенням кількості елементів спостерігається істотне зменшення часу виконання на GPU.

Аналіз коефіцієнтів прискорення (табл. 2) показує, що ефективність використання GPU зростає разом зі збільшенням розміру вхідних даних. Максимальне прискорення досягається для великих масивів, де обчислювальне навантаження значно перевищує накладні витрати на передачу даних.

Таблиця 1 – Час виконання сортування для CPU та GPU

Розмір масиву (n)	CPU, мс	GPU, мс
10 000	4.82	6.15
50 000	31.47	18.92
100 000	78.35	34.11
500 000	612.40	141.76
1 000 000	1 489.30	262.54

Таблиця 2 – Коефіцієнт прискорення GPU відносно CPU

Розмір масиву (n)	10 000	50 000	100 000	500 000	1 000 000
Коефіцієнт прискорення				4.32	5.67

Проаналізувавши обчислені коефіцієнти прискорення та результати тестування для різних розмірів вхідних масивів (табл. 1–2), можна зробити такі висновки:

- при невеликих обсягах вхідних даних значення коефіцієнта прискорення є близьким до одиниці або меншим за неї, що пояснюється накладними витратами на ініціалізацію GPU та передачу даних між CPU і GPU, тому використання графічного процесора для малих масивів є малопрактичним;
- зі збільшенням розміру масиву коефіцієнт прискорення істотно зростає, що свідчить про ефективне використання масового паралелізму GPU та доцільність застосування DirectCompute для обробки великих обсягів даних;
- максимальний ефект від використання GPU досягається у задачах, де обчислювальне навантаження значно перевищує витрати на підготовку та синхронізацію обчислювального конвеєра.

Отже, основними чинниками, що впливають на зміну таких показників, як час виконання програми та коефіцієнт прискорення, є розмір вхідних даних, характер алгоритму сортування та особливості апаратної архітектури. У випадку алгоритму сортування Шелла використання паралельної GPU-реалізації на основі DirectCompute забезпечує істотне підвищення продуктивності для великих масивів даних, тоді як для невеликих обсягів більш ефективним залишається послідовне виконання на CPU.

Висновки

У роботі досліджено можливості паралельної реалізації алгоритму сортування Шелла з використанням технології DirectCompute. Проаналізовано теоретичні основи алгоритму Шелла та обґрунтовано його придатність до розпаралелювання. Розроблено програмну реалізацію, що поєднує еталонну CPU-версію та GPU-версію на основі compute shader.

Експериментальні результати показали, що застосування GPU дозволяє суттєво зменшити час сортування для великих масивів даних. Отримані висновки підтверджують ефективність гетерогенних обчислень та доцільність використання DirectCompute у задачах високопродуктивної обробки даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Shell D. L. A High-Speed Sorting Procedure. Communications of the ACM. 1959. Vol. 2, No. 7. P. 30–32.
2. Knuth D. E. The Art of Computer Programming. Volume 3: Sorting and Searching. 2nd ed. Boston: Addison-Wesley, 1998. 800 p.
3. Новотарський М. А. Алгоритми та методи обчислень. Київ: КПІ ім. Ігоря Сікорського, 2019. 407 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/7421218e-d7dd-4e75-aa3e-bd7979db4e6d/content

4. Harris M. GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics. Boston: Addison-Wesley, 2004. 800 p.
5. Microsoft. DirectCompute Overview. URL: https://learn.microsoft.com/en-us/windows/win32/direct3d11/direct3d-11-advanced-stages-compute-shader

Долішняк Денис Романович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: dolishnyak.denus@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.

Dolishniak Denys Romanovych – student of the Department of Computer Science, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: dolishnyak.denus@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.