УДК 004.8

О.В. Плахотник
В.О. Денисюк

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОГО АЛГОРИТМУ СОРТУВАННЯ КУПОЮ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ C++ AMP

Вінницький національний технічний університет

Анотація

У роботі розглянуто розробку паралельного алгоритму сортування купою (Heap Sort) на основі технології C++ AMP, яка забезпечує виконання обчислень на графічному процесорі (GPU). Проаналізовано класичні алгоритми сортування, визначено їхні обмеження в умовах паралельної обробки, обґрунтовано вибір Heap Sort як методу, що природньо побудований на деревоподібних структурах та допускає часткове розпаралелювання.Розроблено структуру програмного модуля, описано механізми побудови двійкової купи та паралельного виконання операції Heapify на GPU. Створено реалізацію алгоритму з використанням конструкцій array_view та parallel_for_each у середовищі C++ AMP. Проведено тестування продуктивності на різних обсягах даних та виконано порівняння з послідовною реалізацією. Показано, що використання GPU дозволяє зменшити час виконання при великих масивах даних та забезпечує коректність кінцевого результату.

Ключові слова: *Heap Sort, C++ AMP, GPU, паралельні обчислення, алгоритми сортування.*

Abstract

The paper explores the development of a parallel heap sort algorithm using C++ AMP technology, enabling computations to be executed on a graphics processing unit (GPU). Classical sorting methods are analyzed, their limitations in parallel environments are discussed, and Heap Sort is justified as a suitable method due to its binary heap structure and the possibility of partial parallelization.A software module architecture was designed, including GPU-based heap construction and the parallel execution of the heapify procedure. The implementation uses array_view and parallel_for_each constructs of C++ AMP. Performance testing was conducted on datasets of different sizes and compared with a sequential CPU implementation. The results show that GPU acceleration improves execution time on large datasets while maintaining sorting correctness.

Keywords: *heap sort, C++ AMP, GPU computing, parallel algorithm, data processing.*

Вступ

Завдання ефективної обробки великих масивів даних залишається актуальним для сучасних інформаційних систем. Сортування є базовою операцією більшості алгоритмів, тому оптимізація процесу сортування має суттєве практичне значення. Алгоритм Heap Sort забезпечує впорядкування за час O(n log n) та ґрунтується на структурі двійкової купи, що робить його стійким та передбачуваним у плані продуктивності [1].

Використання технології C++ AMP дозволяє перенести частину операцій на графічний процесор, який має значно більшу пропускну здатність при роботі з великими масивами даних [2]. Актуальність роботи полягає у дослідженні можливості прискорення алгоритму Heap Sort шляхом перенесення обчислювально складних операцій Heapify на GPU [3].

Постановка задачі дослідження

У роботі поставлено виконати такі задачі: проаналізувати існуючі алгоритми сортування та визначити їх придатність для паралельної реалізації; обґрунтувати вибір Heap Sort як базового алгоритму; дослідити можливості технології C++ AMP для організації паралельних обчислень на GPU; розробити паралельну реалізацію Heap Sort із використанням C++ AMP; провести тестування продуктивності на різних обсягах даних; порівняти роботу паралельної та послідовної реалізацій.

Виклад основного матеріалу

Алгоритми сортування є важливою складовою більшості обчислювальних систем, оскільки забезпечують впорядкування даних у задачах пошуку, аналізу, оптимізації та моделювання. Серед широкого спектра методів особливий інтерес становить Heap Sort, який поєднує стабільну часову складність O(n log n) та використання двійкової купи як основної структури даних. Особливість алгоритму полягає в чіткій ієрархічній організації елементів, що дозволяє виділяти максимум або мінімум за константний час.

У класичній послідовній реалізації Heap Sort найбільші витрати припадають на процедуру Heapify, яка забезпечує відновлення структури купи після кожної операції вилучення кореневого елемента. Ця процедура виконується рекурсивно або ітеративно, що створює потенціал для
паралельної обробки вузлів нижніх рівнів дерева, де оновлення стану не впливає на сусідні гілки. На цьому спостереженні ґрунтується можливість перенесення частини операцій на графічний процесор.

Технологія C++ AMP (Accelerated Massive Parallelism) забезпечує простий спосіб написання коду, що виконується на GPU. Вона базується на концепції масових паралельних обчислень, де велика кількість однотипних операцій може бути розподілена між потоками всередині обчислювальних блоків. У реалізації паралельного Heap Sort використовуються такі ключові механізми AMP: array_view — абстракція для спільного доступу до пам'яті CPU та GPU, яка гарантує синхронізацію даних; parallel_for_each — основний інструмент організації паралельних обчислень; обмеження restrict(amp), яке дозволяє компілятору переносити код у GPU-область.

У процесі розробки було проведено розподіл алгоритму на дві частини: паралельне формування початкової купи, де вузли нижніх рівнів обробляються на GPU; відновлення купи після переміщення кореневого елемента, яке частково виконується на GPU та частково — послідовно на CPU через залежності між елементами верхнього рівня.

Для експериментального оцінювання було сформовано набори даних розміром 10 000, 50 000 та 100 000 елементів. Час виконання вимірювався окремо для послідовної реалізації на CPU та паралельної — на GPU. Аналіз результатів показав, що застосування C++ AMP забезпечує прискорення алгоритму на великих масивах за рахунок розподілу обчислювального навантаження між потоками GPU. Найбільший виграш спостерігається у фазі побудови купи, де операції над незалежними вузлами природно масштабуються в паралельному середовищі.

Висновки

Результати дослідження демонструють, що технологія C++ AMP є ефективним інструментом для оптимізації алгоритмів сортування і дозволяє значно підвищити продуктивність при роботі з великими наборами даних. Паралелізація Heap Sort на GPU забезпечує коректність обчислень та зменшує час роботи порівняно з класичною реалізацією на центральному процесорі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Heap Sort Algorithm — GeeksforGeeks. URL: https://www.geeksforgeeks.org/heap-sort/
2. C++ AMP Documentation – Microsoft.URL: https://learn.microsoft.com/en-us/cpp/parallel/amp
3. GPU Computing – NVIDIA Developer. URL: https://developer.nvidia.com/gpu-computing

Плахотник Олександр Володимирович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: plaxotnuk22@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.

Plakhotnyk Oleksandr Volodymyrovich– student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: plaxotnuk22@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