УДК 681.3

АВТОМАТИЗОВАНА СИСТЕМА ВИМІРЮВАННЯ ТА АНАЛІЗУ ОБЧИСЛЮВАЛЬНОЇ ПОТУЖНОСТІ ВІДЕОАДАПТЕРА

С.Д. Погорілий, Ю.В. Бойко, М.И. Трібрат, Д.Б. Грязнов, А.О. Листопад
Київський національний дослідницький університет імені Тараса Шевченка

В роботі запропоновано підхід до автоматизації тестування швидкодії реалізації алгоритмів на відеоадаптерах програмно-апаратної платформи CUDA. Проаналізовано залежність швидкодії алгоритму від використаного типу пам‘яті.

Вступ

Останнім часом загострюється проблема недостатності обчислювальних ресурсів для своєчасного вирішення задач, що постійно ускладнюються і потребують все більшої продуктивності обчислень. Аналіз цієї проблематики розкрито в [1].

Компанія NVIDIA створила програмно–апаратну платформу CUDA (Compute Unified Device Architecture), яка дала змогу використовувати для математичних обчислень відеоадаптери (Graphics Processing Unit (GPU)) та відкрила нові можливості для реалізації високопродуктивних паралельних обчислень на них.

Метою роботи є створення автоматизованої системи вимірювання обчислювальної потужності відеоадаптера, яка позволяє визначити максимальну швидкодію створених за допомогою CUDA застосувань для виконання їх графічним процесором, а також кількість потоків, при яких максимальна швидкодія досягається. Створена автоматизована система дає змогу визначити максимально ефективній підхід для роботи застосувань з різними типами пам’яти, що доступні в відеоадаптері.

Особливості архітектури відеоадаптерів

Для розгляду архітектури будемо використовувати введений фірмою NVIDIA термін *warp*. Warp в CUDA являє собою групу з 32 потоків (thread) і є мінімальним обсягом даних, що оброблюються SIMD-способом у мультипроцесорі [2].

Для підвищення пропускної спроможності вся загальна пам'ять розбита на 16 банків (bank). Кожен банк працює незалежно від інших і можна одночасно виконати до 16 звернень до загальної пам'яті. Половина warp (half-warp) складається з 16 потоків і доступ до пам'яті в програмно-апаратній платформі CUDA відбувається окремо для кожного half-warp. Банк-конфлікт (bank conflict) виникає якщо відбувається кілька звернень до того самого банку, а ці звернення виконуються по черзі (рис. 1.а,б,в).

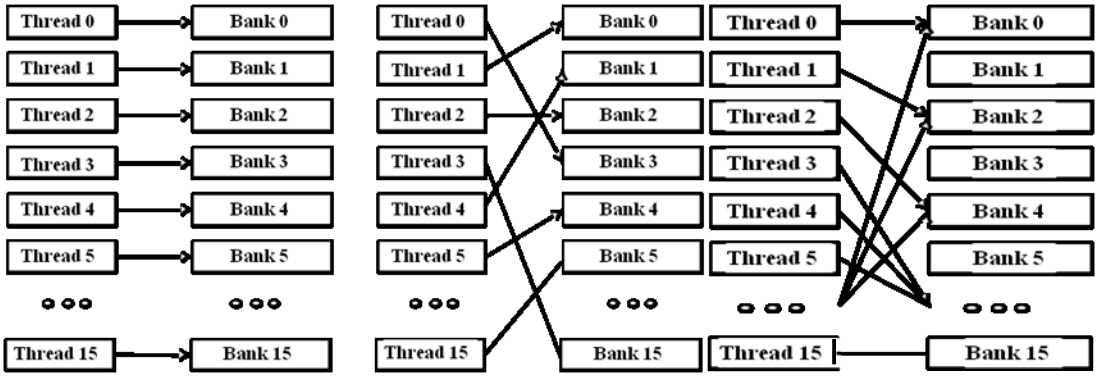

Рис. 1. а- безконфліктне лінійне звернення до загальної пам'яті; б – безконфліктне випадкове звернення; конфлікт другого порядку.

При написанні програм для GPU необхідно оптимізувати і алгоритм і використання ресурсів відеоадаптера програмою. Зміна кількості і конфігурації потоків, чи оптимізація роботи з пам'яттю, критично впливають на швидкодію всієї програми.

Автоматизована система вимірювання

При створення автоматизованої системи вимірювання обчислювальної потужності, використовувався відеоадаптер GeForce 8600 GT.

Стандартні можливості CUDA не дозволяють компілювати програму на виконання з різною кількістю потоків так, щоб в ній можна було викликати певну процедуру з динамічною модифікацією кількості потоків. Необхідно змінювати кількість потоків вручну та знов компілювати програму.

Для автоматизації процесу, значення константи, що задає кількість потоків, виноситься в окремий файл заголовків (header файл) і створена зовнішня програма автоматично збільшує його значення на одиницю при кожній ітерації. Далі відбувається компіляцію проекту з подальшим виконанням з виводом результатів в файл. Цей метод є універсальним оскільки в файл заголовків можна винести будь-яку константу, яка може відповідати за потрібні особливості конкретного алгоритму.

Програма виконує таку послідовність дій:

- створення вхідної послідовності значень параметрів (читання з файлу чи за певним алгоритмом);
- запис і-того параметра в файл заголовків;
- компіляція програми за написаної на CUDA і її з виконання.

Результати тестування

В ході тестування відбувалось багатократне додавання зі збереженням у загальну пам'ять потокового мультипроцесора та у

глобальну пам'ять відеоадаптера. Вимірювання проводилось за двома показниками:

- банк-конфлікти пам'яті потокового мультипроцесора (подвійні та потрійні);
- доступ до глобальної пам'яті відеоадаптера.

Якщо кілька потоків одночасно звертаються до одного банку пам’яті виникає конфлікт і всі звернення виконуються послідовно за декілька тактів, що призводить до втрати швидкодії (рис.2).

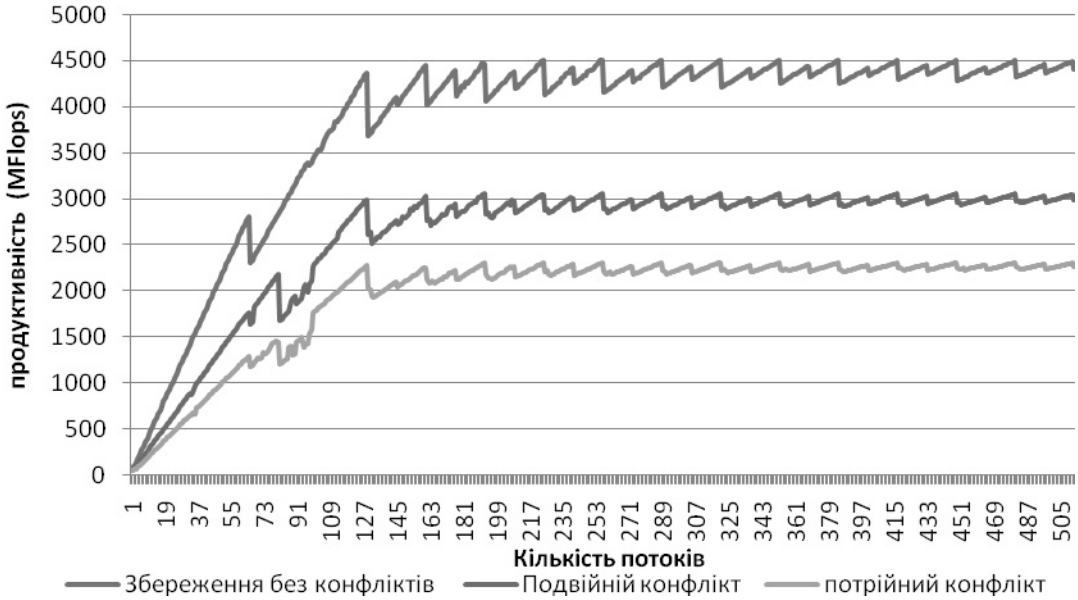

Рис. 2. Результаті тестування зі збереженням до загальної пам’яті.

Коли кількість потоків є кратною 16 (розмір half-warp та кількість банків), то можна отримати максимальну швидкодію, але якщо кількість потоків не кратна 16, то навидь один зайвій потік може зменшити продуктивність більш ніж на 500 MFlops. Втрати швидкодії відбуваються також при зверненні цілого warp до банків пам’яті.

З аналізу графіків випливає, що банк-конфлікти призводять до значної втрати швидкодії і їх розв’язання займає стільки тактів, скільки потоків одночасно звертаються до одного банку.

При звернені до глобальної пам'яті відеоадаптера, потоковий мультипроцесор використовує повільну (відносно графічного процесора) 128-бітну шину даних, що призводить до значних втрат швидкодії. Для оптимізації доступу рекомендовано звертатися до пам’яті одночасно великою кількістю потоків, щоб мінімізувати вплив затримок (рис.3).

Для запису використовувалися три методи: збереження даних кожним потоком, збереження даних кожним парним потоком, збереження даних першою половиною потоків.

Рис. 3. Результаті тестування зі збереженням до глобальної пам'яті відеоадаптера

Перший варіант показує максимальну швидкодію, яка в середньому на 3GFlops менша ніж при використанні загальної пам'яті.

Аналіз другого та третього варіантів показує, що при запису половиною потоків досягається різна швидкодії для різної конфігурації потоків, які здійснюють запис:

- якщо запис здійснюється першою половиною потоків – то досягається максимальна швидкодія після досягнення необхідний кількості потоків для перекриття латентності пам'яті;
- при запису кожним парним потоком досягається лише половина максимальної швидкодії;
- запис першою половиною потоків дає подвійний виграш порівняно з другим після 224 потоків (14 записів в пам'ять).

Тобто швидкодія програми загалом може сильно відрізнятися для різних конфігурацій збереження/зчитування пам'яті.

Графіки залежності швидкодії від кількості потоків – це зростаюча пряма до певного критичного значення, а потім пилкоподібна крива з періодом в 16 потоків (8 для запису кожним парним потоком, або 32 для запису першою половиною потоків). Аналіз цієї структури призводить до висновків, що менеджер потоків задає одну команду для half-warp. Ця команда виконується всіма потоками і тільки потім відбувається обробка наступного half-warp. Коли не всі потоки в ньому виконують одну й ту саму команду, то відбувається втрата швидкодії, кратна відношенню кількості потоків, що виконують цю команду, до максимальної кількості потоків, які можуть її виконувати в half-warp.

При цьому спочатку виконується перша половина потоків, за зростанням їх номеру, потім друга тому період в запису кожним парним потоком 8 оскільки максимальне заповнення першої половини досягається вже при 4 корисних потоках, а порожній цикл другої половини займає мало часу порівняно з записом в глобальну пам‘ять GPU. Тому для кожної програми існує деяке оптимальне значення кількості потоків (кратне 16) після якого не відбувається збільшення швидкодії і на якому досягається її максимум.

Висновки

Запропоновано метод тестування швидкодії GPU в залежності від параметрів алгоритму. Для різної конфігурації кількості потоків, блоків, отримано різну швидкодію. В зв‘язку з цим вельми актуальним є питання підбору максимально ефективної конфігурації для кожного алгоритму розрахунків на GPU.

Для отримання високої швидкодії виконання програм на відеоадаптері за допомогою програмно-апаратної платформи CUDA, необхідно аналізувати та враховувати усі архітектурні особливості GPU: максимальну кількість потоків у блоці, кількість загальної, пам’яті потокових мультипроцесорів, глобальної пам’яті відеоадаптера тощо.

Максимальна швидкодія досягається при кількості потоків кратних 16 (особливості менеджера потоків вбудованого в GPU), з мінімальною кількістю звернень до глобальної пам’яті та використанням загальної пам’яті.

Банк-конфлікти загальної пам’яті призводять до значної втрати швидкодії і їх розв’язання займає стільки тактів, скільки потоків одночасно звертаються до одного банку.

В зв‘язку із цим основою оптимізації є оптимізація роботи з пам'яттю та максимальне використання shared-пам'яті.

Література

1. Погорелый С.Д. Анализ методов повышения производительности компьютеров с использованием графических процессоров и программно-аппаратной платформы CUDA / С.Д. Погорелый, Ю.В. Бойко, М.И. Трибрат, Д.Б. Грязнов// Математичні машини та системи (у друці). – 2010. – №1.
2. NVIDIA CUDA Programming Guide 2.3 [Електронний ресурс]. – Режим доступу: http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf

Отримано 27.05.09