УДК 004.3

Методика вимірювання обчислювальної потужності відеоадаптера (платформа CUDA)

С.Д. Погорілий, М.І. Трібрат, Ю.В. Бойко, Д.Б. Грязнов
Київський національний дослідницький університет імені Тараса Шевченка
sdp@rpd.univ.kiev.ua, mike3b@univ.kiev.ua, boyko@univ.kiev.ua, dima@univ.kiev.ua

Abstract

Pogorilyy S.D., Tribrat M.I., Boyko U.V., Gryaznov D.B. A method of Measuring Computation Power of Video Adapter (Platform CUDA) *The method of testing and measuring productivity of video adapter is offered for applications created by platform CUDA. The performance of applications in various configurations with different numbers of flows and blocks with the use of different types of memory is analyzed. Optimum parameters for the start of the programs on video adapters are defined.*

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 надає 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 методики вимірювання обчислювальної потужності відеоадаптера для застосувань, створених за допомогою CUDA. Розроблений підхід дає змогу визначити можливу максимальну швидкодію при різних конфігураційних параметрах застосувань а також проаналізувати налаштування застосувань для максимально прийнятного та швидкого виконання їх на відеоадаптері.

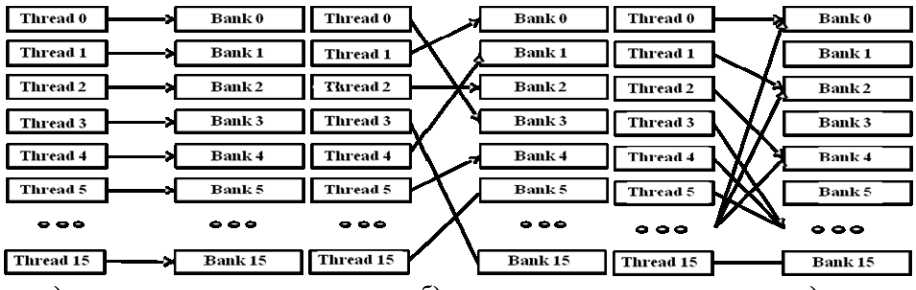
Створена автоматизована система дає змогу визначити максимально ефективній підхід для роботи застосувань з різною кількістю потоків, блоків та різними типами пам’яти, що доступні в відеоадаптері.

Особливості архітектури відеоадаптерів

Для розгляду 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 зручності роботи з потоками використовують блоки (blocks), які являють собою об’єднання максимум до 512 потоків (обмеження апаратної платформи) та полегшують роботу з різними типами пам‘яті відеоадаптера.

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).



а) б) в)

Рисунок 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 потоків та блоків, чи оптимізація роботи з пам'яттю, критично впливають на швидкодію всієї програми.

Методика системи вимірювання

При вимірюванні та аналізі обчислювальної потужності, використовувався відеоадаптер GeForce 8600 GT (табл.1.).

Таблиця 1. Основні характеристики відеоадаптера GeForce 8600 GT

Кодове ім'я	G84
Кількість потокових мультипроцесорів	4
Кількість потокових процесорів	32
Частота потокових процесорів(МГц)	1190
Частота ядра (МГц)	540
Частота пам'яті (МГц)	1400
Розрядність шини пам'яті (біт)	128
Об'єм пам'яті (GDDR3, Мбайт)	256
Пропускна спроможність (Гбайт/с)	22,4
Продуктивність (GFlops)	76

Потоковий мультипроцессор являє собою SIMD-процесор розрядністю 32 біта. Графічний процесор відеоадаптера GeForce 8600 GT містить 32 потокових процесора. Виходячи за того, що потокові процесори працюють на частоті 1,190 ГГц (табл. 1), можна розрахувати пікову продуктивність даного відеоадаптера. Кількість потокових процесорів 32 необхідно помножити на частоту 1,190 Ггц і помножити на кількість операцій за такт (операції типу MAD (multiplication and addition)) – 2 [2], які виконує графічний процесор. Розрахована продуктивність становить 32 х 1,190 Ггц х 2 = 76,2 GFlops (Floating point Operations Per Second (Flops) [3]).

При можливості установки в один системний блок комп'ютера чотирьох відеоадаптерів, отримуємо пікову продуктивність 304 GFlops на одному комп'ютері.

Ринкова вартість відеоадаптера складає 65 доларів США, тобто кожен наступний GFlops коштує менш ніж 1 US $.

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 та блоків. Необхідно змінювати кількість потоків вручну та знов компілювати програму.

Для автоматизації процесу була запропонована методика відповідно до якої, значення константи, що задає кількість потоків (або блоків), виноситься в окремий файл заголовків (header файл) і створена зовнішня програма автоматично збільшує його значення на одиницю при кожній ітерації (до 512 потоків). Далі відбувається компіляція 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.

Створена зовнішня програма, яка відповідно до зазначеного вище підходу виконує послідовність дій:

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

Результати тестування максимальної швидкодії

В ході тестування виконувалась велика кількість (1млрд) однотипних операцій додавання та помноження зі збереженням у загальну пам'ять потокового мультипроцесора

Максимальна розрахована продуктивність відеоадаптера, який використовувався для тестування, складає 76 GFlops. В ході тестування було отримано 73.5 GFlops або 97% від максимальної швидкодії. Даний показник суттєво залежить від параметрів алгоритму, який програмується за допомогою CUDA. Вплив параметрів розглядається нижче.

Таку продуктивність можна отримати при використанні не менш ніж по 160 потоків (5 *warp* по 32 потоки) у 4-х блоках в алгоритмі програми, що аналізується. Кількість блоків у даному випадку відповідає кількості потокових мультипроцесорів (табл. 1), тобто визначається апаратною реалізацією. При використанні кількості блоків меншої за наявні потокові мультипроцесорі, програма не повністю використовує графічний процесор. При конфігурації з одним блоком (block1) максимальна швидкодія досягається не менш ніж при 160 потоках, але ця швидкодія в 4 рази менша ніж при конфігурації з 4-мя блоками, при використанні двох блоків (block2) – у 2 рази, та при трьох блоках (block3) – у 3/4 разів (рис. 2.) [4].

Кількість потоків, необхідних для отримання максимальної швидкодії програми що аналізується, можна отримати використовуючи 32 потоки, але при цьому необхідно збільшити кількість блоків на величину, кратну кількості потокових мультипроцесорів відеоадаптера (block64 рис. 2.).

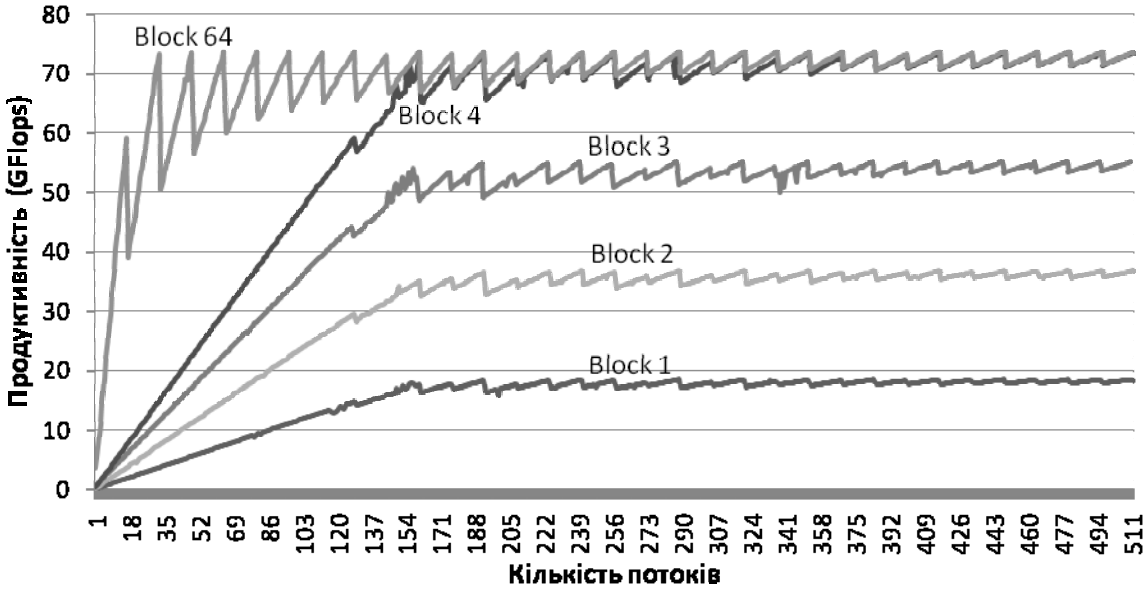


Рисунок 2 – Продуктивність відеоадаптера в залежності від конфігурації кількості блоків та потоків.

Робота з пам'яттю відеоадаптера

В ході тестування обраховувались однотипні операцій додавання зі збереженням в залежності від тесту у загальну пам'ять потокового мультипроцесора або у глобальну пам'ять відеоадаптера.

Зміна алгоритму тестування значно позначилась на отриманій швидкодії. Вона зменшилась відносно максимально можливої більш ніж у 10 разів.

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.3)[4].

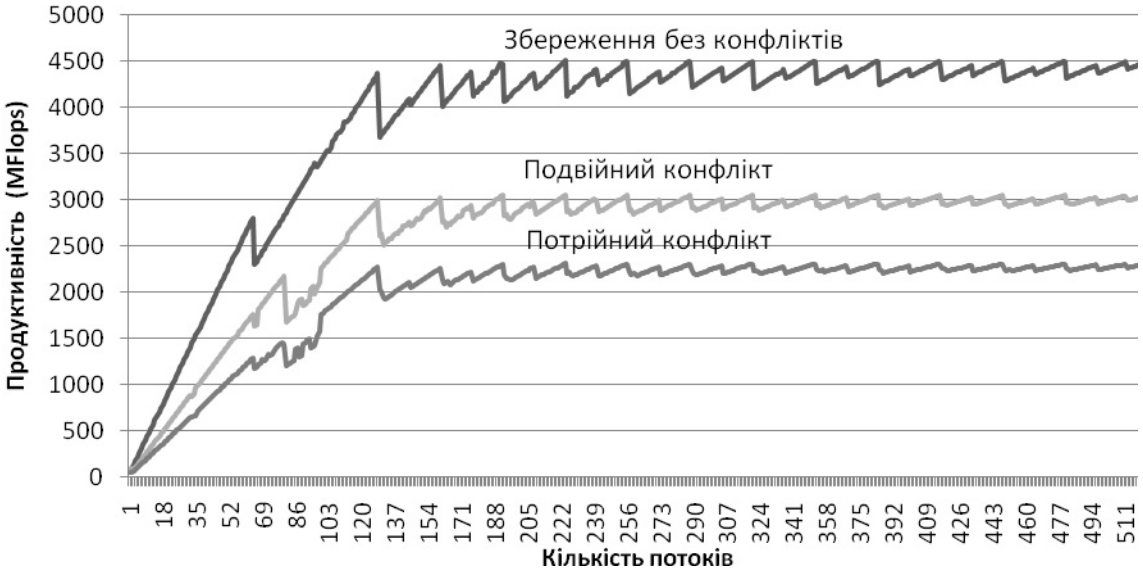


Рисунок 3 – Результаті тестування зі збереженням до загальної пам'яті.

Коли кількість потоків є кратною 16 (розмір *half-warp* та кількість банків), то можна отримати максимальну швидкодію тестової програми, 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 загальної пам'яті.

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.4).

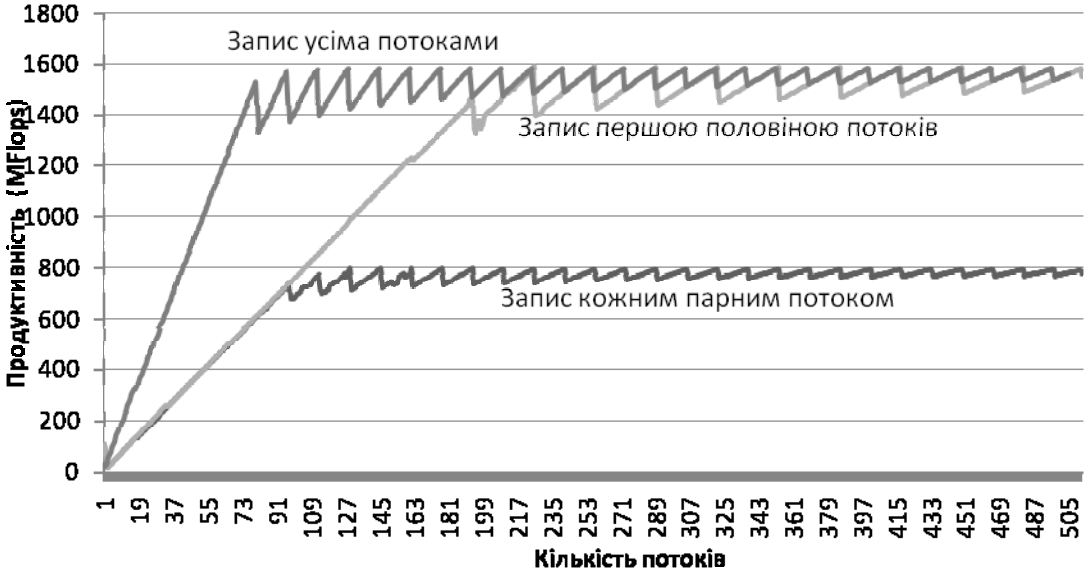


Рисунок 4 – Результати тестування зі збереженням до глобальної пам'яті відеоадаптера

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

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.

Важливе значення має алгоритм, що програмується за допомогою CUDA: за тими чи іншим обставинам, які розглядаються у статті, можна отримати швидкодію реалізації для GPU значно нижчу, ніж реалізації для CPU того ж самого алгоритму.

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 програм 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.

Реалізація алгоритму за допомогою програмно-апаратної платформи CUDA всупереч перерахованим вище факторам та параметрам конфігурації, робить використання відеоадаптера для математичних обрахунків марним.

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. – 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
3. Сапронов А. Обзор некоторых пакетов измерения производительности кластерных систем [Електронний ресурс] / А Сапронов // Журнал IXBT. - 2004 – Режим доступу: http://www.ixbt.com/cpu/cluster-benchtheory.shtml
4. Погорілий С.Д. Автоматизована система вимірювання та аналізу обчислювальної потужності видеоадаптера //С.Д. Погорілий, Ю.В. Бойко, М.И. Трібрат, Д.Б. Грязнов, А.О. Листопад // Моделирование и компьютерная графика: третья междунар. науч.-техн. конф., 7-9 окт 2009г.:тезисы докл. – Донецк.

Надійшла до редакції 11.01.2010