УДК 004.421

К.т.н., доцент Замятін Д.С., студент Чередніченко А.К.

Національний технічний університет України
«Київський політехнічний інститут»

ОЦІНКА ШВИДКОДІЇ БАЗОВИХ АЛГОРИТМІЧНИХ ЗАДАЧ В ГЕТЕРОГЕННИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

Abstract

Denis Zamyatin, assoc. prof., PhD; Antonina Cherednichenko, student
Performance evolution of base algorithmic problems for heterogeneous computation
This paper concerns the task of effective heterogeneous systems using.
The experimental study and comparison of the performance for four different computing algorithms types on CPU and GPU is investigated. Practical importance of this research is identification of the most appropriate algorithms for heterogeneous computation. The ways for further research are proposed as well.

Вступ

Швидкодія персонального комп'ютера напряму пов'язана з тактовою частотою центрального процесора (CPU). Останнім часом динаміка зростання тактової частоти CPU помітно знизилась. Натомість з'явилась нова тенденція – виникнення гетерогенних систем. Програмування для таких систем здійснюється на базі відкритого стандарту OpenCL [1, 2].

Для досягнення максимальної продуктивності слід визначити вимоги до алгоритмів програм, які виконуються на окремих обчислювальних модулях гетерогенних систем. Одними з найпоширеніших модулів таких систем є графічні процесори (GPU). Сучасні GPU фактично є багатопроцесорними системами SIMD-архітектури з високою піковою продуктивністю, що стимулює інтерес до їх використання не тільки для обробки графічної інформації, а й для вирішення обчислювальних задач (*GPGPU*) [3]. Саме тому визначення критеріїв ефективності виконання програмного коду на GPU є актуальним питанням сьогодення.

Постановка задачі

Мета роботи полягає в експериментальному дослідженні швидкодії

базових програмних алгоритмів для графічних процесорів (GPU) у порівнянні з швидкодією центрального процесора (CPU). Як базові для порівняння були обрані наступні види алгоритмів: лінійні матрично-векторні алгоритми (знаходження суми векторів, добутку матриць), паралельна редукція суми елементів, двійковий пошук, порозрядне сортування.

Термінологія

Гетерогенна система – це комп'ютерна система, яка складається з різних обчислювальних модулів. Обчислювальним модулем може бути процесор загального призначення CPU *(central processing unit)*, спеціалізований процесор SPP (special-purpose processor): цифровий сигнальний процесор DSP (digital signal processor), графічний процесор GPU *(graphics processing unit)* , спеціалізовані інтегральні схеми ASIC (application-specific integrated circuit) тощо. В даній роботі під гетерогенною системою розуміється система з GPU.

GPGPU (General-Purpose computing on Graphics Processing Units) – використання графічних процесорів для вирішення неграфічних задач.

Основний матеріал

GPGPU програмування передбачає виконання коду на двох різних платформах – центральному процесорі та графічному адаптері. При цьому звичайний послідовний код виконується на CPU, а для масивно-паралельних обчислень застосовується GPU з використанням набору потоків, які виконуються одночасно. При цьому максимальне прискорення (S), яке можна отримати при такому підході визначається за законом Амдала (Amdahl Law) [4]:

$$S = \frac{1}{1 - P + \frac{P}{N}} .$$

У цій формулі P – частина часу виконання програми, яка може бути розпаралелена на N процесорних елементів. При збільшенні кількості процесорів N максимальний виграш прямує до $1/(1-P)$.

Саме тому програми, які добре розпаралелюються, на графічних процесорах з великою кількістю процесорних елементів N виконуються набагато швидше, ніж на центральному процесорі. Але при цьому необхідно пам'ятати, що крім безпосередньо часу виконання фрагменту

коду на GPU ($T_{GPU_{вик}}$) слід враховувати додатковий час, який витрачається на створення контексту виконання програми на GPU, виділення ресурсів графічного адаптеру, копіювання даних з основної пам'яті у пам'ять відео карти, копіювання результату та вивільнення ресурсів GPU ($T_{GPU_{додат}}$). Отже, доцільно виконувати код на GPU тільки тоді, коли

$$T_{GPU_{заг}} \leq T_{CPU_{вик}}, \text{ де } T_{GPU_{заг}} = T_{GPU_{вик}} + T_{GPU_{додат}},$$

де $T_{CPU_{вик}}$ – час виконання програми на центральному процесорі.

Розглянемо ряд базових алгоритмів та перевіримо доцільність їх виконання на графічних процесорах для даних різної розмірності. Нехай коефіцієнт K – відношення кількості операцій, які виконуються на GPU до кількості пересилок даних між CPU та GPU.

1. Лінійні матрично-векторні алгоритми.

1.1 Знаходження суми векторів.

Нехай необхідно знайти поелементну суму двох векторів розмірності *n*. Маємо *n* операцій (додавань) та *3n* переміщень між CPU та GPU. Коефіцієнт K – 1:3 або $\theta(1)$. Отже, виграш від виконання такого коду на GPU буде дуже незначним або взагалі відсутнім.

1.2 Знаходження добутку матриць.

Для знаходження добутку двох матриць розмірності *n*n* слід виконати n^3 операцій (множень та додавань) та $3n^2$ переміщень. Коефіцієнт K – $\theta(n)$. Таким чином, виконання такого коду на GPU буде набагато швидше ніж на CPU і виграш стрімко зростатиме із збільшенням розмірності.

2. Паралельна редукція суми.

Редукцією масиву $a_0, a_1, a_2, \ldots, a_{n-1}$ відносно заданої операції (суми) буде наступна величина A = $(((a_0 + a_1) + a_2) + \ldots + a_{n-1})$. Коефіцієнт K – 1:2 або $\theta(1)$, а отже виграшу у швидкодії не буде.

3. Двійковий пошук.

Для знаходження елементу масиву розмірності *n* за допомогою двійкового пошуку необхідно здійснити $\log_2 n$ операцій та *2n* переміщень. У такому випадку виконання коду на GPU є нераціональним.

4. Порозрядне сортування (radix sort).

Часова складність порозрядного сортування для масиву розмірності *n* становить $\theta(pn)$, де *p* – додатковий об'єм пам'яті необхідний для сортування ($p < n$). Кількість переміщень між CPU та GPU – *2n*. Отже,

виграш у швидкодії буде помітним, але не таким значним як для випадку 1.2.

Практичні результати дослідів представлені на рис. 1. Використане апаратне забезпечення – комп'ютер з процесором Intel Core 2 Duo T5850 (2,16 ГГц) та графічним адаптером NVIDIA GeForce 8600M GS.

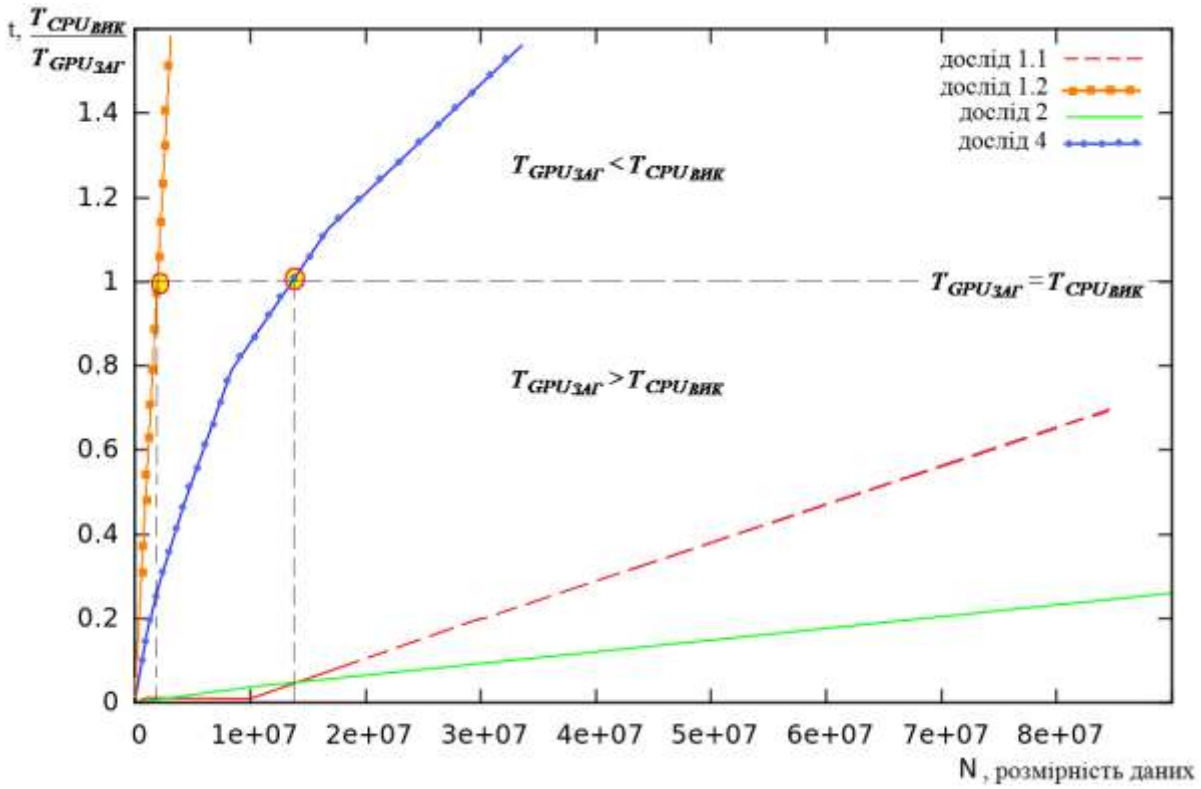

Рис. 1. Експериментальні дані швидкодії програмних алгоритмів

Отримані графіки відображають відношення загального часу виконання програми на GPU до часу її виконання на CPU для даних різної розмірності. Очевидно, що на графічному процесорі серед розглянутих операцій найдоцільніше виконувати перемноження матриць (дослід 1.2) та порозрядне сортування (дослід 4).

Висновки

Отже, у роботі експериментальним способом було доведено, що на графічних процесорах раціонально виконувати програмний код у таких випадках:

- Для масивно-паралельних обчислень над великими обсягами даних, для яких складність виконуваних на GPU операцій нівелює

вартість пересилок даних між CPU та GPU. Теоретична перевірка цього твердження зводиться до розрахунку відношення кількості операцій до кількості переміщень. Якщо порядок цього відношення залежить від розмірності даних, то швидкодія виконання програмного коду на графічному процесорі у порівнянні із звичайним процесором стрімко зростатиме при збільшенні розмірності даних.

- Оскільки кількість пересилок між центральним та графічним процесорами повинна бути мінімальною, то дані слід зберігати в пам'яті GPU якнайдовше та послідовно виконувати над ними декілька операцій. В подальшому було б корисно дослідити цей підхід з метою досягнення максимальної ефективності функціонування гетерогенної системи.

Література

1. *Benedict R. Gaster*. The OpenCL C++ Wrapper API, Version 1.1, Khronos OpenCL Working Group, June 14, 2010.
2. NVIDIA, OpenCL Best Practices Guide, May 27, 2010.– С. 10-53.
3. General-Purpose Computation Using Graphics Hardware: [http://gpgpu.org].
4. *Kirk D., Hwu W.* Programming Massively Parallel Processors : A Hands-on Approach, Morgan Kaufmann Publishers, 2010. – С. 10-34, 95-120.