КОМП'ЮТЕРНІ НАУКИ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, СИСТЕМНИЙ АНАЛІЗ ТА КІБЕРБЕЗПЕКА

DOI 10.31891/2307-5732-2021-297-3-19-24
УДК 004.75

В. В. РУСІНОВ, О. В. ЧЕРЕВАТЕНКО, Л. М. ПУСТОВІТ, О. М. ПУСТОВІТ
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"

СПОСІБ РОЗРОБКИ ІЗОЕФЕКТИВНОЇ ГЕТЕРОГЕННОЇ СИСТЕМИ НА ОСНОВІ МАШИННОГО НАВЧАННЯ ДЛЯ ЗАДАЧІ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР'Є

У роботі розглянуто питання ізоефективності систем MPP та гетерогенних систем CPU-GPU на задачі дискретного перетворення Фур'є. Розробка паралельних додатків у якості своєї цілі може мати не лише скорочення часу виконання, але і забезпечення можливостей вирішення проблем більшої розмірності. Особливість паралелізації алгоритму включає ефективне використання апаратних засобів при збільшенні розмірності задачі є важливою характеристикою паралельних обчислень.

Ключові слова: ізоефективність, гетерогенні обчислення, перетворення Фур'є.

V. V. RUSINOV, O. V. CHEREVATENKO, L. M. PUSTOVIT, O. M. PUSTOVIT
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

METHOD OF DEVELOPMENT OF ISOEFFICIENT HETEROGENEOUS SYSTEM USING MACHINE LEARNING FOR THE PROBLEM OF DISCRETE TRANSFORMATION OF FOURIER

Abstract — In this paper, the isoefficiency of MPP systems and heterogeneous CPU-GPU systems on the problem of discrete Fourier transform is considered. The development of parallel applications as its goal can not only reduce execution time, but also provide opportunities to solve problems of a larger dimension. The peculiarity of algorithm parallelization includes the efficient use of hardware while increasing the dimension of the problem is an important characteristic of parallel computing. However, currently heterogeneous systems have not been researched extensively to determine isoefficiency characteristics and build application-specific systems around said method, although there are articles that show potential using isoefficiency to design the system and using heterogeneous approach to accelerate performance of different tasks. Discrete Fourier Transform algorithm lets build systems that discretize analogue and digital signals and it can serve as a benchmark to test different systems. Algorithms suited for MPP systems can use analytical approach to find out issoefficiency function and to determine how scaling the system or changing the size of the task will change its performance metrics. One of the most popular approaches to linking up processing units in MPP systems is using hypercube topology. MPP system that is connected using this topology will be analyzed. CPU-GPU heterogeneous system will be analyzed using an approach based on polynomial regression. Due to the nature of heterogeneous systems, analytic approach used in MPP system is impossible. Predictive model based on polynomial regression will use modelling results from using CPU and GPU separately to estimate how much time it will take for heterogeneous system to finish the task. To ensure accuracy of the experiment, several systems will be used to model the task. Using this approach, resulting issoefficient heterogeneous system will be analyzed using performance metrics such as time. From the resulting graphs, we can see the isoefficient properties across all of the heterogeneous systems.

Keywords: isoefficiency, heterogeneous calculations, Fourier transformation

Постановка задачі

Створення ізоефективних систем дозволяє розгортати систему для деякої задачі з урахуванням її ефективності. Ефективність паралельних обчислень залежить від алгоритму реалізації відображення задачі на систему. Метою даної статті є розглянути процес створення ізоефективної системи для розв'язання практичних задач на прикладі перетворення Фур'є, покласти алгоритм в MPP систему та використати емпіричний підхід на основі машинного навчання для апроксимації функції ізоефективності для систем з різними архітектурними рішеннями.

Задача яка буде виконуватись на паралельній та гетерогенній системі – алгоритм дискретного перетворення Фур'є. Дискретне перетворення Фур'є використовується для розв'язання задач спектрального аналізу – для дослідження сигналу через розділ його на сукупність сигналів. Цікавим з точки зору паралельної обробки є нелінійна складність цього алгоритму.

На основі результатів моделювання задачі на розглянутих системах буде описано функцію ізоефективності. Для паралельної MPP системи, результат може бути отриманий аналітично. Для гетерогенної системи CPU-GPU на сьогоднішній день неможливо аналітично отримати функцію ізофективності, натомість буде застосований підхід з використанням алгоритмів машинного навчання.

Аналіз останніх досліджень та публікацій

Існує низка наукових публікацій, які вивчають тематику ізоефективних систем на основі архітектури MPP [1, 2]. Розглядаються методи масштабування паралельних алгоритмів для вирішування тих чи інших прикладних задач для їх відображення на MPP системи із заданою топологічною організацією. Підхід створення ізоефективних систем дозволяє зробити аналіз алгоритму на його якість та можливості до паралелізації на заданій топології і, як результат, можливості до масштабування системи із передбачуваною ефективністю виконання задачі.

Більшість сучасних суперкомп'ютерів і великих центрів обробки даних використовують системи, що мають як центральні процесори (CPU), так і графічні процесори (GPU) на вузлах. Загальні обчислення на графічному процесорі (GPGPU) відкрили користувачам ПК дорогу для експериментів та роботи над проектами, які залучають графічні процесори для передачі трудомістких задач. Гетерогенні системи на основі CPU та GPU достатньо широко досліджені та є досить перспективним напрямком розвитку паралельних обчислень. Незважаючи на розповсюдження систем на основі CPU-GPU, на сьогоднішній день ізоефективність не досліджена по відношенню до гетерогенних систем.

Виклад основного матеріалу

Ізоефективні системи – системи із заданою ефективністю вирішення задач, яка, будучи описаною заздалегідь, постійно підтримується. Розглянемо теоретичну можливість створення таких систем.

Формула ефективності паралельної обробки на паралельній системі виглядає наступним чином, визначаючи можливість досягнення необхідної ефективності паралельних обчислювальних систем:

$$E = \frac{S}{N}. \tag{1}$$

За рахунок варіювання параметрами *n*, тобто розмірність задачі, та *N*, кількість процесорів, можна досягти лінійного нарощування продуктивності при збільшенні кількості процесорів. Це означає, що при виконанні обчислювальних процесів можна заздалегідь визначати необхідну ефективність їх реалізації [3].

Як було зазначено раніше, використання формули ефективності та виведення формули ізоефективності неможливе для гетерогенних систем через те що гетерогенні системи використовують різні архітектури. Для встановлення ефективності гетерогенних систем із виведенням емпіричної формули ізоефективності можна використовувати підхід машинного навчання, а саме модель лінійної регресії – моделі залежності змінної від однієї або декількох інших змінних або факторів з лінійною функцією залежності. Лінійна регресія дозволяє розподілити розмірність задач між графічним процесором і центральним процесором.

Для створення моделі, яка підійде до нелінійної часової залежності від розмірності задачі, можна використовувати поліноміальну регресію. У поліноміальної регресії ступінь деяких незалежних змінних перевищує 1. У формульному вигляді це можна представити як:

$$Y = a_1 X_1 + (a_2)^2 X_2 + (a_3)^4 X_3 + (a_n)^{2^{n-1}} X_n + b \tag{2}$$

У деяких змінних є ступінь, у інших – ні. Також можна вибрати певний ступінь для кожної змінної, але для цього необхідні певні знання про те, як вхідні дані пов'язані з вихідними [4].

Поліноміальна регресія моделює нелінійно розділені дані (чого не може лінійна регресія). Вона більш гнучка і може моделювати складні взаємозв'язки. Крім того, вона дає повний контроль над моделюванням змінних об'єкта (вибір ступеня), однак для вибору найбільш підходящого ступеня необхідно мати деякі знання про дані, оскільки при неправильному виборі ступеня дана модель може бути перенасичена.

Час виконання завдання прогнозується за допомогою «прецедентальної» інформації – за який час виконувались завдання того ж типу. Створюється планувальник задач, який формує початковий результат, що змодельований із аналізу вже виконаних задач, потім застосовується модель для прогнозування часу виконання нових задач. Планувальник аналізує розміри даних, з якими працюють задачі, та час їх виконання, а також враховує час попередніх задач цього ж типу.

Використовуючи подібну детерміновану модель кореляції передбачається час виконання задачі того ж типу, де вхідним параметром є розмір її даних.

Задача, яка буде змодельована на системі MPP та гетерогенній системі CPU-GPU – це дискретне перетворення Фур'є. Реалізація дискретного перетворювача Фур'є (ДПФ), що лежить в основі спектрального аналізу, представляє собою неформальне подання сигналів, тобто досліджувані сигнали представляються послідовністю відліків $x(k)$.

$$F_x(p) = \sum_{k=0}^{N-1} x(k) \cdot e^{-jk\Delta tp\Delta\omega} \tag{3}$$

$$\omega \to \omega_p \to p\Delta\omega \to p \qquad \Delta\omega = \frac{2\pi}{T} \tag{4}$$

З формули (2) видно, що інтервали подання сигналів є однаковими 2π, що є періодом низьких частот. Щоб підвищити точність, необхідно збільшити інтервал Т.

$$t \to t_k \to k\Delta t \to k; \qquad \Delta t = \frac{T}{N} = \frac{1}{k_{зап.}} \cdot f'гр. \tag{5}$$

ДПФ – проста обчислювальна процедура «звірочного» типу, оцінка її складності: N2 + N. Для її реалізації потрібно обрахувати поворотні коефіцієнти ДПФ:

$$W_N^{pk} = e^{-jk\Delta t\Delta\omega p} \tag{6}$$

Ці поворотні коефіцієнти записуються в ПЗУ, тобто є константами.

$$W_N^{pk} = e^{-jk\frac{T}{N}p\frac{2\pi}{T}} = e^{-j\frac{2\pi}{N}pk} \tag{7}$$

У формулі (5) W_N^{pk} (поворотні коефіцієнти) не залежать від T, а лише від розмірності перетворення N, тому подаються не в експоненційній формі, а в тригонометричній.

$$W_N^{pk} = \cos\left(\frac{2\pi}{N}pk\right) - j\sin\left(\frac{2\pi}{N}pk\right) \tag{8}$$

Поворотні коефіцієнти повторюються, з цієї причини зміни величин описані саме до зазначених значень: p – до (N-1), k – до (N-1), з періодом N(2π). При винесенні знаку коефіцієнта можна зберігати лише половину коефіцієнтів. У ПЗУ окремо зберігаються дійсні та уявні частини коефіцієнтів [5].

У загальній формі ДПФ можна представити наступним чином:

$$F_x(p) = \sum_{k=0}^{N-1} x(k) \cdot W_N^{pk} \tag{9}$$

З такого формульного визначення ДПФ доречно представити у вигляді графа.

CUDA (Compute Unified Device Architecture) – платформа паралельного обчислення та API розроблена компанією Nvidia. Вона дозволяє розробникам використовувати відеокарти Nvidia для загальних обчислень. Платформа CUDA надає розробнику прямий доступ до системи команд відеокарти та елементів паралельного обчислення, для виконання обчислювальних ядер (compute kernel).

Обчислювальне ядро – основна робоча одиниця, за допомогою якої, розробник описує алгоритм. Такий термін використовується не лише для GPU, також він використовується для FPGA, TPU, DSP. Для CUDA, парадигма програмування дуже близько інтегрована з векторними обчисленнями, за основу взято припущення, що виклик ядра виконується одночасно в низці незалежних елементах, що дозволяє паралелізм на рівні даних. Проте, також існують атомарні операції, які можуть бути використані для синхронізації між елементами. Кожен виклик отримує індекси, для 1 або більше розмірностей, за допомогою яких здійснюється адресація даних, або буферизація [6].

Архітектура GPU відноситься до класу SIMD, тобто на кожне вище згадане ядро надсилаються дані над якими за один такт виконується одна операція. В якості прикладу пропонується оглянути TU102, який лежить в основі mainstream GPU RTX 2080Ti та в професіональному GPU Quadro RTX 6000. Він складається з 6 кластерів графічного процесінгу, 36 кластерів процесінгу текстур та 72 Streaming Multiprocessors (SM). SM складається з 64 CUDA ядер, 8 тенсорних ядер, 256 кілобайт файлу регістру, 4 Texture Units, 96 кілобайт спільної пам'яті. До цього, кожне ядро має доступ до 6144 кілобайт L2 кешу.

Результати

Для дослідження ефективності обчислень, необхідно використовувати метрики часу виконання алгоритму послідовно (на одному процесорі) та паралельно (на декількох процесорах). На основі отриманих значень часу можна дослідити ефективність паралельного алгоритму.

Перша розглянута система MPP – гіперкуб 1 ступеня MPP (розшифровується як massive parallel processing) є масивно-паралельною архітектурою комп'ютерних систем. В такому типі архітектури пам'ять фізично розділяється. В системі знаходяться окремі блоки (модулі), всередині яких знаходяться процесор, комунікаційні процесори (роутери), локальний банк оперативної пам'яті, мережеві адаптери, жорсткі диски, пристрої вводу/виводу.

Доступ до оперативної пам'яті окремого вузла мають лише процесори з цього ж модуля. Між собою блоки з'єднуються комунікаційними каналами. Для користувачів існує можливість отримати номер процесора та процесорів, до яких він під'єднаний, після чого можна ініціювати обмін даними між ними.

Основними перевагами систем з MPP-архітектурою: добра здатність до масштабування, відсутність необхідності тактової синхронізації процесорів за рахунок того, що у кожному блоці доступ до локального банку оперативної пам'яті мають лише «власні» процесори, висока продуктивність та результативність, практично доведена на MPP-машинах з великою кількістю процесорів (кілька тисяч) [7].

Гіперкуб є однією з найбільш розповсюджених топологій, зокрема для MPP систем і має добре описані характеристики та відомості щодо застосування його для різних задач. Дана топологія представляє собою окремий випадок структури решітки, коли по кожній розмірності сітки є тільки два процесори (тобто гіперкуб містить 2*N* процесорів при розмірності *N*) [8].

Топологія гіперкуба досить широко поширена на практиці при об'єднанні паралельних процесорів. Лінія, що з'єднує два вузли визначає одновимірний гіперкуб. Квадрат, утворений чотирма вузлами – двовимірний гіперкуб, а куб з восьми вузлів – тривимірний гіперкуб і т.д. Оскільки система складається з декількох процесорних елементів з локальною пам'яттю, під час виконання задачі необхідно враховувати час, витрачений на передачу даних. На наступній схемі зображено алгоритм передачі даних між вузлами під час виконання ДПФ для 4 сигналів.

На основі алгоритму (рис. 1) встановимо функції часу послідовної обробки та паралельної обробки.

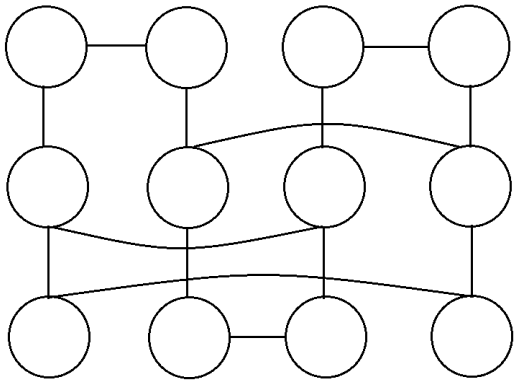

Рис. 1. Діаграма взаємодії процесорів для MPP системи з параметрами N та n 4

Застосуємо формули прискорення та ефективності для отримання аналітичної формули ізоефективності.

$$T_{1n} = n + k_n n \tag{10}$$

$$T_{pn} = n + k_n \frac{n}{N} \tag{11}$$

$$E_n = \frac{T_1}{NT_p} = \frac{1 + k_n}{N + k_n}, \tag{12}$$

де N – кількість процесорів, n – розмірність задачі, kn – коефіцієнт розмірності. Коефіцієнт розмірності можна обрахувати за наступною формулою:

$$k_n = 2k_{\frac{n}{2}} + 2 \tag{13}$$

$$k_2 = 1 \tag{14}$$

Для прикладу наведеному на рисунку 1, де N=4 та n=4, значення часу паралельної обробки становитиме:

$$T_{pn} = n + k \frac{n}{N} \tag{15}$$

Для отримання результатів гетерогенних систем було застосовано декілька різних систем, наведених в таблиці 1. В представлених системах використовуються різні графічні та центральні процесори, що представляє додаткову складність для аналітичного підходу до встановлення ізоефективності.

Таблиця 1

№	Процесор	Графічний процесор
1	AMD Ryzen 9 3900X	RTX 2060
2	AMD Ryzen 5 2400G	GTX 1060-3GB
5	Intel Core i5-7200U	Geforce 940MX
6	Intel Core i5-9600KF	GTX 1080

Спершу, необхідно встановити час виконання задач із однаковою розмірністю на процесорі та графічному прискорювачі (рис. 2). На основі отриманих даних буде розроблена модель, яка дозволятиме розподіляти навантаження між процесорами, щоб максимально швидко виконати задачу.

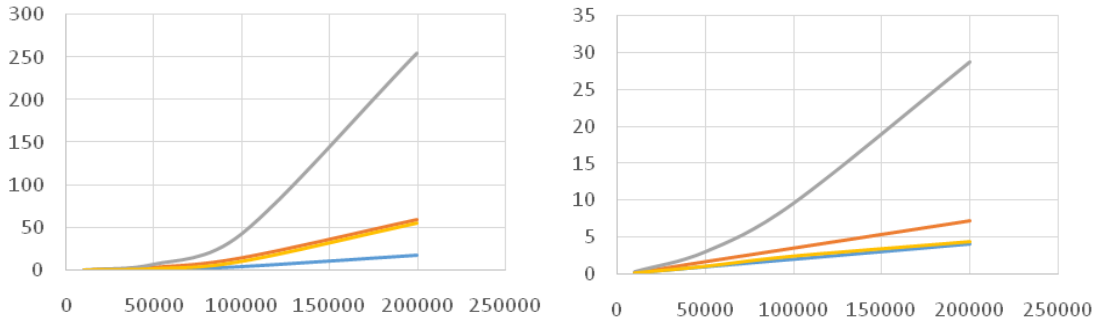

Рис. 2. Час виконання задачі на CPU (зліва) та GPU (справа)

Перше, що можна встановити з графіків, – це нелінійна складність алгоритму вирішення задачі. Це зумовлює необхідність використання алгоритмів машинного навчання для апроксимації нелінійної функції. Також можна побачити різницю між тим як із зміною розмірності задачі змінюється приріст часу. Перед тим як застосувати машинне навчання для розподілу задач між процесором та відеокартою, необхідно встановити час передачі даних між загальною пам'яттю системи та пам'яттю на GPU.

Використовуючи підхід на основі поліноміальної регресії, встановимо розподіл розмірності задачі на CPU та GPU. На основі часових метрик виконання задачі на системах залучаючи обидва процесори, можна емпірично встановити ефективність системи. Розглянемо час відправлення даних з пам'яті GPU на спільну (рис. 3).

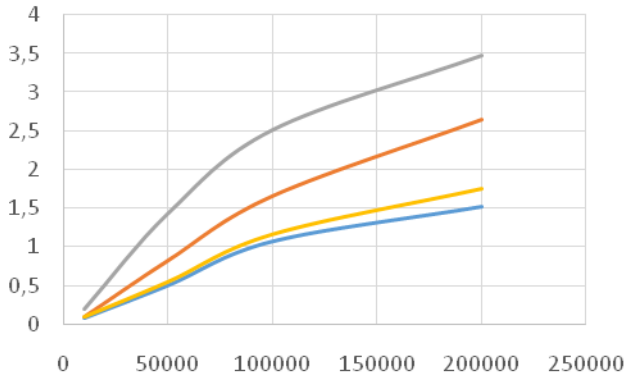

Рис. 3. Часові затрати на відправку даних з GPU на спільну пам'ять

На основі діаграми отриманих часових даних (рис. 4), можна встановити ефективність гетерогенних систем та встановити необхідну розмірність задачі для отримання такої самої ефективності на іншій системі.

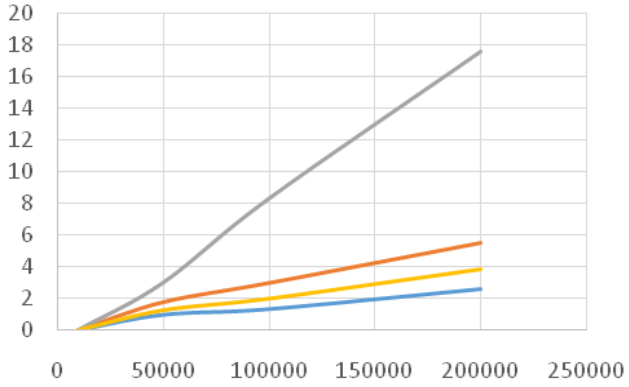

Рис. 4. Час виконання задачі на гетерогенній системі

Ефективність системи можна встановити на основі часу виконання задачі одночасно на процесорі та на відеоприскорювачі (табл. 2). Запропонований підхід нижче використовує розподіл розмірності задачі на процесори для встановлення загальної ефективності гетерогенної системи з точки зору прискорення відносно найбільш швидкого процесору з двох представлених.

Таблиця 2

№	Система	Графічний процесор	$n = 50000$	$n = 100000$	$n = 200000$
1	AMD Ryzen 9 3900X	RTX 2060	1.004133	1.419191	1.446006
2	AMD Ryzen 5 2400G	GTX 1060-3GB	1.011028	1.360679	1.402923
3	Intel Core i5-7200U	Geforce 940MX	0.931325	1.228246	1.63615
4	Intel Core i5-9600KF	GTX 1080	1.014851	1.245034	1.465462

Висновки

В ході виконання дослідження основним результатом можна вважати, що гетерогенні системи CPU-GPU можна використовувати для ізоефективних обчислень. Основною перевагою такого підходу є передбачуваність при побудові систем орієнтованих під конкретну задачу, в рамках даної роботи такою задачею є дискретне перетворення Фур'є. На основі запропонованого підходу можна проводити масштабування системи за рахунок прискорювачів на основі іншої архітектури та розробляти ізоефективні алгоритми.

Результати моделювання показують природу виконання обчислення задачі нелінійної складності. Використання методів машинного навчання, а саме поліноміальної регресії, дозволяє створювати алгоритм розподілу задачі між CPU та GPU зі збереженням коефіцієнту прискорення. З результатів, також можна зробити висновок про те, що система з найбільш потужним процесором показує найкращі результати для n = 100000, при цьому результати для n = 50000, 200000 більш збалансовані.

Література

1. Hwang K. Scalability and programmability of massively parallel processor. Parallel Processing: CONPAR 94-VAPP VI. Springer, Berlin, Heidelberg, 1994. P. 1–4.

2. Grama A. Y., Gupta A., Kumar V. Isoefficiency: Measuring the scalability of parallel algorithms and architectures. IEEE Parallel & Distributed Technology: Systems & Applications. 1993. Vol. 1. Issue 3. P. 12–21.

3. Drozdowski M., Singh G., Marszalkowski J. M. Isoefficiency Maps for Divisible Computations in Hierarchical Memory Systems. PPAM (1). 2019. P. 224–234.

4. Ostertagová E. Modelling using polynomial regression. Procedia Engineering. 2012. Vol. 48. P. 500–506.

5. Bracewell R. N., Bracewell R. N. The Fourier transform and its applications. New York: McGraw-Hill, 1986. Vol. 31999. P. 267–272.

6. Harish P., Narayanan P. J. Accelerating large graph algorithms on the GPU using CUDA. International conference on high-performance computing. Springer, Berlin, Heidelberg, 2007. P. 197–208.

7. Hey T., Scott C., Surridge M. Simulation and modelling applications on mpp systems. Massively Parallel Processing Applications and Development. Elsevier, 1994. P. 15–21.

8. Yongchang J. et al. A scalability metric for algorithm-machine on NOW and MPP. Proceedings Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region. IEEE, 2000. Vol. 1. P. 405–407.

В. В. РУСІНОВ, ORCID ID: 0000-0002-4362-0248 volodymyr.r.v@ukr.net
О. В. ЧЕРЕВАТЕНКО, ORCID ID: 0000-0001-9686-0555 chereva@ukr.net
Л. М. ПУСТОВІТ, Pusamy1998@gmail.com
О. М. ПУСТОВІТ lrerednawl@gmail.com

Рецензія/Peer review : 13.05.2021 р. Надрукована/Printed :30.06.2021 р.