https://doi.org/10.31891/2307-5732-2023-317-1-67-73
УДК 004.3

ГУРМАН Іван
Хмельницький національний університет
https://orcid.org/0000-0002-2282-3484
e-mail: devastator167384@gmail.com
БОБРОВНІКОВА Кіра
Хмельницький національний університет
https://orcid.org/0000-0002-1046-893X
e-mail: bobrovnikova.kira@gmail.com
БЕДРАТЮК Леонід
Хмельницький національний університет
https://orcid.org/0000-0002-6076-5772
e-mail: leonid.uk@gmail.com
БЕДРАТЮК Ганна
Хмельницький національний університет
https://orcid.org/0000-0003-0224-5549
e-mail: bedratyuk@ukr.net

МЕТОД АНАЛІЗУ ПРОГРАМНОГО КОДУ ДЛЯ ОЦІНКИ ЕНЕРГОСПОЖИВАННЯ ЯДРА CUDA

В роботі запропоновано метод аналізу програмного коду для оцінки енергоспоживання ядра CUDA для підвищення енергоефективності застосунків, які орієнтовані на обчислення на графічних процесорах. Запропонований підхід, заснований на статичному аналізі програми CUDA та методах машинного навчання, дозволяє оцінити енергоспоживання таких застосунків без необхідності виконання на фізичних пристроях.

З метою оцінки ефективності запропонованого підходу було використано три архітектури GPU NVIDIA: PASCAL, TURING та AMPERE. Результати проведених експериментів показали, що для архітектури NVIDIA AMPERE запропонований підхід надає можливість досягти коефіцієнта детермінації на рівні 0.9173. Отримані результати підтверджують ефективність запропонованого методу аналізу програмного коду для оцінки енергоспоживання ядра CUDA.

HURMAN Ivan, BOBROVNIKOVA Kira, BEDRATYUK Leonid, BEDRATYUK Hanna
Khmelnytskyi National University

APPROACH FOR CODE ANALYSIS TO ESTIMATE POWER CONSUMPTION OF CUDA CORE

The graphics processing unit is a popular computing device for achieving exascale performance in high-performance computing programs, which is used not only in graphics tasks, but also in computational tasks such as machine learning, scientific computing, and cryptography. With the help of a graphics processor, you can achieve significant speed and performance compared to the central processing unit. CUDA, Compute Unified Device Architecture, a graphics processing unit software development platform, allows developers to use the high-performance computing capabilities of graphics processing units to solve problems traditionally handled by central processing units.

Even though the graphics processing unit has a relatively high power to performance ratio, it consumes a significant amount of power during computing. The paper proposes an approach for code analysis to estimate power consumption of CUDA core to improve the power efficiency of applications focused on computing on graphics processing units. The proposed approach makes it possible to estimate the power consumption of such applications without the need to run them on physical devices. The proposed approach is based on static analysis of the CUDA program and machine learning methods.

To evaluate the effectiveness of the proposed approach, three graphics processing unit architectures were used: NVIDIA PASCAL, NVIDIA TURING, and NVIDIA AMPERE. The results of the experiments showed that for the NVIDIA AMPERE architecture, the proposed approach using decision trees makes it possible to achieve a determination coefficient of 0.9173. The results obtained confirm the effectiveness of the proposed code analysis method for estimating the power consumption of the CUDA core. This method can be useful for CUDA developers who want to improve the efficiency and power efficiency of their programs.

Keywords: GPU; NVIDIA; CUDA; power consumption; high-performance computing.

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Сьогодні графічні процесори (graphics processing unit, GPU) використовуються в найшвидших сучасних суперкомп'ютерах, є домінуючою платформою для глибокого навчання та забезпечують інтелект розумних пристроїв, починаючи від безпілотних автомобілів і закінчуючи роботами та розумними камерами; для створення фотореалістичних зображень з високою частотою кадрів у реальному часі та як обчислювальні пристрої для вирішення задач у багатьох областях, включаючи наукові дослідження, машинне навчання, фінансову аналітику [1]. GPU продовжують еволюціонувати, додаючи нові властивості та функції для підтримки нових випадків їх використання.

Протягом останніх десятиліть розвитку GPU продуктивність цих пристроїв зросла в кілька сотень разів. Також, з'явилися нові архітектури GPU, такі як Tensor Cores [2, 3], що спеціалізуються на обчисленнях зі штучним інтелектом та машинним навчанням та надають нові можливості для всіх робочих навантажень: від 6-кратного прискорення під час навчання мережі до 3-кратного підвищення продуктивності всіх програм.

Значно підвищити продуктивність обчислень за рахунок використання потужності графічного процесора NVIDIA дозволяє паралельна обчислювальна платформа та модель програмування CUDA

(Compute Unified Device Architecture) від NVIDIA Corporation [4]. CUDA є стандартною функцією всіх графічних процесорів NVIDIA GeForce, Quadro та Tesla, а також рішень NVIDIA GRID [2, 3]. CUDA забезпечує абстракцію, яка дозволяє програмісту визначити, як саме повинна виконуватися програма. Компілятор генерує код PTX, який не залежить від апаратного забезпечення та компілюється для конкретного цільового графічного процесора під час виконання. Сумісність з новими моделями графічних процесорів забезпечується оновленням драйвера щоразу, коли випускається новий графічний процесор. При цьому зміни в кількості регістрів або розмірі спільної пам'яті можуть (але не обов'язково) відкрити можливість для подальшої оптимізації. CUDA переважно дотримується моделі паралельних обчислень, де кожен потік виконує ту саму операцію над різними елементами даних паралельно. Дані розбиваються на сітку блоків 1D, 2D або 3D. Кожен блок також може мати форму 1D, 2D або 3D і може складатися з понад 512 потоків. Потоки в блоці потоків можуть взаємодіяти через спільну пам'ять. Блоки потоків виконуються як менші групи потоків, відомі як «warps». Програми CUDA можуть також розподіляти роботу між кількома GPU, якщо в програмі. передбачено програмування кількох GPU.

Однак, збільшення продуктивності GPU призводить до збільшення енергоспоживання, що може бути проблемою для розробників, які мають обмеження на споживання енергії їх пристроїв. З ростом вимог до продуктивності, особливо в обчислювальних задачах, розробники повинні мати глибоке розуміння ключових особливостей свого коду, які найбільше впливають на споживання енергії. Це необхідно для забезпечення оптимальної продуктивності та енергоефективності пристроїв. Для досягнення цих цілей розробники повинні мати доступ до інструментів аналізу та прогнозування споживання енергії їхнім кодом.

Аналіз останніх досліджень і публікацій

Складність архітектури GPU завжди була перешкодою для моделювання споживання енергії GPU. На відміну від центральних процесорів, графічні процесори мають високопаралельну та гетерогенну архітектуру з великою кількістю процесорів, кеш-пам'яті та з'єднань. Це ускладнює точне моделювання енергоспоживання для різних робочих навантажень і конфігурацій.

На сьогодні відомо багато різних підходів для вимірювання енергоспоживання GPU. З метою поглиблення розуміння мікроархітектури графічних процесорів NVIDIA у [5] проведено дослідження споживання енергії при виконанні 40 різних інструкцій PTX для чотирьох графічних процесорів NVIDIA (Maxwell, Pascal, Volta, and Turing). Крім того, продемонстровано вплив оптимізації компілятора CUDA на споживання енергії кожною інструкцією. З цією метою було використано три програмні методи зчитування давачів потужності графічного процесора, два з яких використовують API NVML (NVIDIA Management Library) [6] від NVIDIA, а третій заснований на використанні компоненту CUDA PAPI (Performance Application Programming Interface) [7].

У статті [8] запропоновано метод прогнозування часу виконання ядра GPU, що дозволяє розробникам виправити неефективний код перед запуском програми, не виконуючи його насправді. Аналітична модель для прогнозування часу виконання ядра графічного процесора створюється на основі аналізу проміжного коду PTX ядра CUDA. Експериментальний аналіз показав, що для 45 додатків оцінка часу виконання має середню абсолютну похибку 26,86% порівняно з фактичним часом виконання.

У роботі [9] досліджується енергоспоживання ядер CUDA з використанням фреймворку SimGrid. Робота базується на моделі виконання CUDA, яка дозволяє розподілити роботу ядра на блоки та розподілити їх між потоковими мультипроцесорами. Модель передбачає енергоспоживання на основі кількості блоків. Результати дослідження були перевірені з використанням шести різних ядер CUDA на двох різних графічних процесорах NVIDIA (Tesla M2075 та Kepler K20Xm). Для оцінки точності було порівняно реальні виміри з результатами моделювання за допомогою середньої відносної помилки. Середні відносні помилки для часу роботи та енергії становили 6,69% та 6,86% для M2075 та 6,54% та 8,33% для K20Xm відповідно.

У дослідженні [10] розглянуто можливість збереження енергії шляхом зміни розміру блоку в конфігурації ядра. Гіпотеза полягає в тому, що оптимальний розмір блоку під час виконання робочого навантаження може досягти більшої економії енергії. Для перевірки гіпотези було обрано два ядра на графічному процесорі NVIDIA Tesla K40, а саме ядра Bitonic Mergesort і Vector Addition. Дослідження включало в себе вимірювання потужності та енергоспоживання GPU з різними розмірами блоків під час виконання робочого навантаження. Отримані результати демонструють вплив зміни розміру блоків на потужність та енергоспоживання GPU. Зокрема, було зроблено висновок, що оптимальний розмір блоку може забезпечити значну економію електроенергії.

Робота [11] присвячена дослідженню проблеми енергетичної пропорційності графічних процесорів. Енергетична пропорційність визначає систему, яка споживає енергію пропорційно кількості роботи, яку вона виконує. Для енергопропорційної системи оптимізація коду з метою підвищення продуктивності також оптимізує споживання енергії. В роботі експериментально досліджені енергопропорційність графічних процесорів NVIDIA K40c та NVIDIA P100 PCIe за допомогою спеціально розробленої програми множення матриці. Було виявлено, що обидва графічні процесори характеризуються слабкою енергопропорційністю, що дає можливість двоцільової оптимізації програми з точки зору динамічної енергії та продуктивності.

Виділення невирішених раніше частин загальної проблеми, котрим присвячується стаття

Незважаючи на значну кількість досліджень в області оцінки та оптимізації енергоспоживання GPU, проблема підвищення енергоефективності програм залишається невирішеною. Тому актуальною задачею є розроблення методу аналізу програмного коду для оцінки енергоспоживання ядра CUDA.

Формулювання цілей статті

Метою роботи є розроблення методу аналізу програмного коду для оцінки енергоспоживання ядра CUDA, який базується на статичному аналізі. Дослідження включає два завдання: (1) створити архітектурно-незалежну модель для передбачення споживання енергії CUDA програм з прийнятною похибкою та точністю; (2) дослідити вплив різних особливостей CUDA програм, що розглядаються у цій роботі, на споживання енергії GPU. Розроблюваний метод покликаний допомогти розробникам програм CUDA в розумінні профілю споживання енергії ядром.

Виклад основного матеріалу

Запропонований метод ґрунтується на аналізі коду з використанням апаратних ознак і коду PTX. З метою пошуку найбільш ефективних ознак, які можуть впливати на енергоспоживання ядра CUDA, було виокремлено 15 ознак (табл. 1). Для зменшення взаємного впливу та мінімізації корелюючих ознак було використано кореляційний аналіз, що дозволило включити в набір ознак лише одну з висококорелюючих ознак. Для цього було використано коефіцієнт кореляції Пірсона. За допомогою отриманої теплової карти коефіцієнта кореляції Пірсона (рис. 1) було виключено ознаки, які сильно корелюють між собою (табл. 1): f_2, f_{13}, f_{15} корелюють з f_7, тому f_2, f_{13}, f_{15} були виключені з набору ознак; f_4, f_6, f_{14} корелюють між собою, тому в наборі ознак залишено лише f_6. Для визначення ознак, які найбільш сильно впливають на енергоспоживання, було використано дерево рішень (з цією метою було застосовано бібліотеку Scikit-learn, яка використовує алгоритм, заснований на CART (Classification and Regression Trees)).

Таблиця 1

Множина ознак, які можуть впливати на енергоспоживання ядра CUDA

Ознака	Опис ознаки	
	Повний набір ознак	**Результуючий набір ознак (джерело)**
f_1	Кількість потоків на блок	Визначається користувачем
f_2	Кількість блоків	Виключено
f_3	Кількість обчислювальних інструкцій в ядрі	На основі аналізу PTX
f_4	Кількість змодельованих обчислювальних інструкцій	Виключено
f_5	Кількість інструкцій глобальної пам'яті в ядрі	На основі аналізу PTX
f_6	Кількість змодельованих інструкцій глобальної пам'яті	Алгоритм аналізу коду
f_7	Цикли випуску інструкцій	Обчислюється на основі даних користувача
f_8	Кількість різних інструкцій у ядрі	На основі аналізу PTX
f_9	Загальна кількість змодельованих різних інструкцій	Алгоритм аналізу коду
f_{10}	Відношення активних груп потоків (warp) в потоковому мультипроцесорі до максимальної кількості активних груп потоків, підтримуваних потоковим мультипроцесором	CUDA Occupancy Calculator
f_{11}	Кількість інструкцій поділюваної пам'яті в ядрі	На основі аналізу PTX
f_{12}	Кількість змодельованих обчислювальних інструкцій поділюваної пам'яті	Виключено
f_{13}	Загальна кількість запущених потоків	Виключено
f_{14}	Загальна кількість змодельованих інструкцій	Виключено
f_{15}	Кількість груп потоків (*wave)* блоків, виконаних на потоковому мультипроцесорі	Виключено

З метою проведення експериментальних досліджень розробленого підходу було зібрано тестовий набір даних за допомогою таких інструментів як CUDA Occupancy Calculator та із застосуванням розробленого алгоритму аналізу коду (рис. 2). Розроблений алгоритм використовує статичний аналіз коду CUDA та аналіз апаратних функцій, наданих виробником GPU. В алгоритмі використано наступні позначення: SM – кількість потокових мультипроцесорів; T_m – максимальна кількість потоків на потоковий мультипроцесор; B – кількість блоків; T – кількість потоків на блок; l – кількість ітерацій циклу; T_s – загальна кількість потоків, запланованих на потоковий мультипроцесор; T_w – кількість потоків в поточній групі потоків. Вхідними даними алгоритму є код PTX разом із параметрами запуску, тобто кількістю блоків, кількістю потоків на блок і кількістю ітерацій циклу. Ці вхідні дані не можна витягнути за

допомогою статичного аналізу, і вони є дуже важливими для вилучення ознак. Параметри запуску (кількість потоків на блок, кількість блоків) надаються користувачем, оскільки їх неможливо отримати з аналізу PTX.

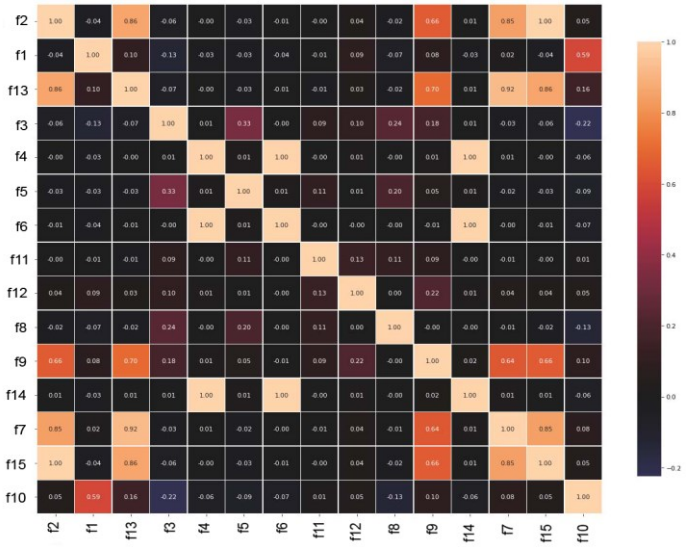

Рис. 1. Використання коефіцієнта кореляції Пірсона для виключення корелюючих ознак

Тестові ядра запускалися з різними параметрами запуску (кількість потоків на блок, кількість блоків). Як відомо, енергоспоживання GPU залежить від робочого навантаження ядра, наприклад, доступ до глобальної пам'яті споживає більше енергії, ніж доступ до поділюваної пам'яті, регістрів на кристалі та виконання арифметичних операцій з плаваючою комою [1]. Значення споживаної потужності були згенеровані за допомогою двох інструментів: Unified Power Profiling Application Programming Interface(UPPAPI) та NVML API. Для мінімізації похибки виміряної потужності в якості результуючого значення було використано середнє значення потужності. Для обчислення теоретичного максимального значення f_{10}, якого можна досягти під час виконання конкретного ядра, використано CUDA Occupancy Calculator та такі вхідні дані: (1) кількість потоків на блок; (2) кількість регістрів, що використовуються на потік; (3) кількість поділюваної пам'яті, яку використовує блок потоків; (4) обчислювальна здатність CUDA пристрою.

Кількість інструкцій кожного типу в ядрі графічного процесора підраховувалась за допомогою статичного аналізу коду PTX для ядра GPU. Кількість змодельованих обчислювальних інструкцій одержувалась за допомогою алгоритму аналізу коду, оскільки для цієї ознаки потрібно брати до уваги специфікацію GPU.

```
procedure PROGRAMANALYSIS(PTX, B, T, l)
T_s = [B/SM* T];
      f15=0;
      while T_s > 0 do
          T_w=T_s >T_m ? T_m: T_s
          for each Block in PTX do
              for each instruction i in G do
                  if i.Type == Computing then f4+=1;
                  end if
                  if i.Type == Global Memory then f6+=1;
                  end if
                  if i.Type == Shared Memory then f12+=1;
                  end if
                  if i.Type == Miscelleneous then f9+=1;
                  end if
              end for
              if Block has a loop then Multiply each feature by l
              end if
          end for
          T_s= T_s - Tm;
          f15= f15+1;
      end while
      return feature values;
end procedure
```

Рис. 2. Алгоритм аналізу коду

Також, енергоспоживання графічного процесора чутливе до доступу до глобальної пам'яті. Глобальні інструкції доступу до пам'яті також підраховувались шляхом аналізу коду PTX. Кількість інструкцій, що виконуються на одному потоковому мультипроцесорі, обчислювалась із використанням алгоритму аналізу коду. Інструкції доступу до поділюваної пам'яті також можуть сприяти енергоспоживанню графічного процесора, особливо якщо вони супроводжуються конфліктами, пов'язаними з банками пам'яті. Кількість інструкцій, що пов'язані з поділюваною пам'яттю, була обчислена із використанням алгоритму аналізу коду.

З метою одержання значень ознак було проведено множину еталонних тестів продуктивності GPU з використанням таких інструментів як CUDA Toolkit Samples, Rodinia Benchmark, Tango GPU [12]. З метою оцінки ефективності запропонованого підходу було використано три архітектури GPU NVIDIA: PASCAL (GeForce GTX 1070), TURING (GeForce RTX 2060) та AMPERE (GeForce RTX 3070). Результати проведених експериментів наведені на рис. 3, 4. В якості показників для оцінки ефективності було використано: коефіцієнт детермінації, R2, кореневе середньоквадратичне відхилення (Root-Mean-Square Deviation, RMSD) та середню абсолютну похибку (Mean Absolute Error, MAE). Результати експериментів наведені в табл. 2.

Таблиця 2

Результати експериментів: оцінка ефективності розробленого підходу

Архітектура GPU	R2	RMSD	MAE
PASCAL	0.8399	10.2232	5.2135
TURING	0.8833	7.8476	3.7081
AMPERE	0.9173	12.3425	5.6370

Наведені результати демонструють високий показник коефіцієнту детермінації R2 для всіх трьох архітектур GPU, що підтверджує ефективність розробленого підходу. Як видно з рис. 3, для всіх контрольних показників прогнозована потужність є досить точною, порівняно з виміряним значенням.

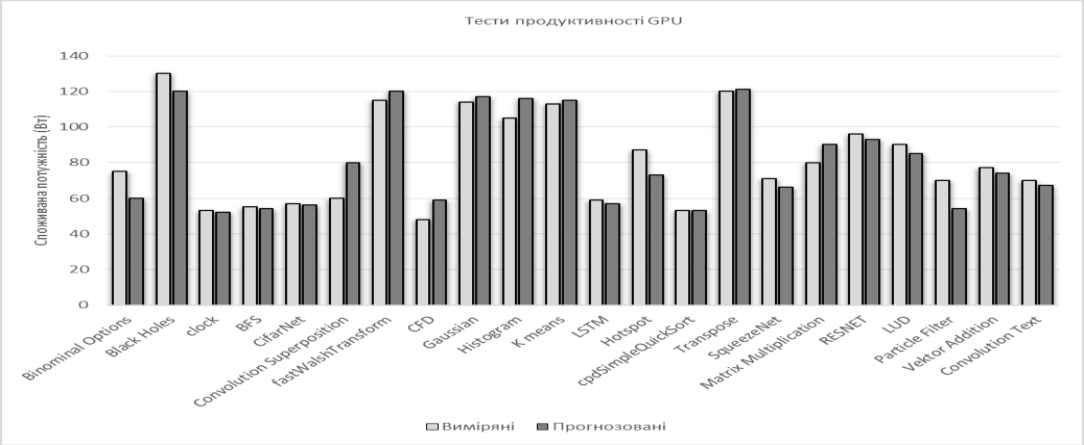

Рис. 3. Значення виміряного та прогнозованого споживання енергії

Як видно з наведених результатів, ефективність підходу не залежить від архітектури, оскільки одержано високу точність для всіх трьох досліджуваних архітектур GPU. Аналіз важливості ознак з використанням дерева рішень для всіх трьох архітектур (рис.4) показав, що ознака f_7 (цикли випуску інструкцій) найбільше впливає на енергоспоживання в трьох архітектурах. Глобальні інструкції доступу до пам'яті мають значний вплив на енергоспоживання, таким чином f_5 та f_6 мають вищі значення для всіх трьох архітектур. Кількість обчислювальних інструкцій f_3 також впливає на прогнозування потужності так само ефективно, як кількість різноманітних інструкцій (f_8) і кількість змодельованих різних інструкцій (f_9).

а) **б)** **в)**

Рис. 4. Відносна важливість ознак для архітектур NVIDIA: а) PASCAL; б) TURING; в) AMPERE

Кількість запущених блоків f_2 може представляти кількість потокових мультипроцесорів, активованих під час виконання коду ядра CUDA, оскільки кожен новий блок співставляється з одним потоковим мультипроцесором. Проте, ця ознака помірно вплинула на енергоспоживання. Відношення активних груп потоків (warp) в потоковому мультипроцесорі до максимальної кількості активних груп потоків, підтримуваних потоковим мультипроцесором, f_{10}, мала бути однією з найважливіших ознак, які сприяють збільшенню енергоспоживання, оскільки вона описує максимальну кількість потоків, що знаходяться в потоковому мультипроцесорі. Однак значення f_{10} є нижчим для всіх трьох архітектур. Це може бути спричинене похибкою, викликаною використанням CUDA Occupancy Calculate, який дає приблизне значення ознаки. Серед усіх типів розглянутих ознак інструкції з поділюваною пам'яттю найменше впливають на енергоспоживання (f_{11}, f_{12}) для всіх трьох архітектур. Це пов'язане з тим, що поділювана пам'ять набагато швидша, порівняно з глобальною.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

З метою підвищення енергоефективності програм, орієнтованих на обчислення на графічних процесорах, в роботі запропоновано метод аналізу програмного коду для оцінки енергоспоживання ядра CUDA. Запропонований підхід дозволяє оцінити енергоспоживання таких програм без необхідності запускати їх на фізичних пристроях. Запропонований метод заснований на статичному аналізі програми CUDA та методах машинного навчання.

Для оцінки ефективності запропонованого підходу було використано три архітектури графічних процесорів: NVIDIA PASCAL, NVIDIA TURING і NVIDIA AMPERE. Результати експериментів показали, що для архітектури NVIDIA AMPERE запропонований підхід дозволяє досягти найвищого коефіцієнта детермінації на рівні 0,9173. Проте ефективність підходу не залежить від архітектури, оскільки одержано високу точність для всіх трьох досліджуваних архітектур GPU. Отримані результати підтверджують ефективність запропонованого методу. Метод може бути корисним для розробників CUDA, які мають на меті покращити ефективність та енергоефективність своїх програм.

Майбутнє дослідження буде спрямоване на вивчення впливу затримки кожного типу інструкцій та використання регістрової пам'яті на енергоспоживання GPU.

Література

1. Bridges R. A., Understanding gpu power: A survey of profiling, modeling, and simulation methods / R. A. Bridges, N. Imam, and T. M. Mintz, // ACM Comput. Surv., vol. 49, no. 3, pp. 41:1-41:27, Sep. 2019.

2. Dally W. J., Evolution of the graphics processing unit (GPU). // Keckler S. W., Kirk D. B. IEEE Micro, 41(6), 2021, P. 42-51.

3. NVIDIA Corporation. (2019) Cuda compiler driver nvcc. [Електронний ресурс]. Режим доступу: https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc – (Дата звернення 4.01.2023). – Назва з екрану.

4. NVIDIA, Cuda samples. [Електронний ресурс]. – Режим доступу: https://docs.nvidia.com/cuda/cuda-samples/index.html – (Дата звернення 02.01.2023). – Назва з екрану.

5. Arafa Y., Verified instruction-level energy consumption measurement for nvidia gpus. ElWazir A., ElKanishy A., Aly Y., Elsayed A., Badawy A. H., Santhi N.// 17th ACM International Conference on Computing Frontiers, 2020, P. 60-70.

6. NVIDIA, Nvml api, reference manual, [Електронний ресурс]. – Режим доступу: https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf – (Дата звернення 28.12.2022). – Назва з екрану.

7. Performance Application Programming Interface (PAPI). (2019) Version5.7. [Електронний ресурс]. Режим доступу: https://icl.utk.edu/papi – (Дата звернення 4.01.2023). – Назва з екрану.

8. Alavani G., Predicting execution time of cuda kernel using static analysis, / K. Varma, and S. Sarkar, // IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications. Los Alamitos, CA, USA: IEEE Computer Society, dec 2018, pp. 948–955.

9. Boughzala D., Predicting the energy consumption of cuda kernels using simgrid. // Lefèvre L., Orgerie A. C., IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2020, P. 191-198.

10. Ikram M. J., Investigating the effect of varying block size on power and energy consumption of GPU kernels. // Saleh M. E., Al-Hashimi M. A., Abulnaja O. A., The Journal of Supercomputing, 78(13), 2022 P. 14919-14939.

11. Manumachu R. R., On Energy Nonproportionality of CPUs and GPUs. Lastovetsky, A., // In 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2022, P. 34-44.

12. Karki A., Tango: A deep neural network benchmark suite for various accelerators, / C. Palangotu Keshava, S. Mysore Shivakumar, J. Skow, G. Madhukeshwar Hegde, and H. Jeon, // IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2019, P. 137-138.

References

1. Bridges R. A., Understanding gpu power: A survey of profiling, modeling, and simulation methods / R. A. Bridges, N. Imam, and T. M. Mintz, // ACM Comput. Surv., vol. 49, no. 3, pp. 41:1-41:27, Sep. 2019.

2. Dally W. J., Evolution of the graphics processing unit (GPU). // Keckler S. W., Kirk D. B. IEEE Micro, 41(6), 2021, P. 42-51.

3. NVIDIA Corporation. (2019) Cuda compiler driver nvcc. [Online]. – Access Mode: https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc – (Date of application 4.01.2023). – name from the screen.

4. NVIDIA, Cuda samples. [Online]. – Access Mode: https://docs.nvidia.com/cuda/cuda-samples/index.html – (Date of application 02.01.2023). – name from the screen.

5. Arafa Y., Verified instruction-level energy consumption measurement for nvidia gpus. ElWazir A., ElKanishy A., Aly Y., Elsayed A., Badawy A. H., Santhi N.// 17th ACM International Conference on Computing Frontiers, 2020, P. 60-70.

6. NVIDIA, Nvml api, reference manual, [Online]. – Access Mode: https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf – (Date of application 28.12.2022). – name from the screen.

7. Performance Application Programming Interface (PAPI). (2019) Version5.7. [Online]. – Access Mode: https://icl.utk.edu/papi – (Date of application 4.01.2023). – name from the screen.

8. Alavani G., Predicting execution time of cuda kernel using static analysis, / K. Varma, and S. Sarkar, // IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications. Los Alamitos, CA, USA: IEEE Computer Society, dec 2018, pp. 948–955.

9. Boughzala D., Predicting the energy consumption of cuda kernels using simgrid. // Lefèvre L., Orgerie A. C., IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2020, P. 191-198.

10. Ikram M. J., Investigating the effect of varying block size on power and energy consumption of GPU kernels. // Saleh M. E., Al-Hashimi M. A., Abulnaja O. A., The Journal of Supercomputing, 78(13), 2022 P. 14919-14939.

11. Manumachu R. R., On Energy Nonproportionality of CPUs and GPUs. Lastovetsky, A., // In 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2022, P. 34-44.

12. Karki A., Tango: A deep neural network benchmark suite for various accelerators, / C. Palangotu Keshava, S. Mysore Shivakumar, J. Skow, G. Madhukeshwar Hegde, and H. Jeon, // IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2019, P. 137-138.