Інформаційні технології

УДК 004.424

Куб'юк Євгеній Юрійович
студент
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Кубюк Евгений Юрьевич
студент
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Kubuik Yevhenii
Student of the
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

ОГЛЯД ВИКОРИСТАННЯ ПАРАДИГМИ ПОТОКІВ ДАНИХ З ГРАФІЧНИМИ ПРОЦЕСОРАМИ
ОБЗОР ИСПОЛЬЗОВАНИЯ ПАРАДИГМЫ ПОТОКОВ ДАННЫХ С ГРАФИЧЕСКИМИ ПРОЦЕССОРАМИ
OVERVIEW OF USING THE DATA FLOW PARADIGM WITH GRAPHICS PROCESSING UNITS

Анотація. *У дослідженні здійснено аналіз сучасного програмного забезпечення управління потоками даних на графічних процесорах*

Ключові слова: *парадигма потоків даних, графічні процесори, розподілені обчислення.*

Аннотация. *В исследовании проведен анализ современного программного обеспечения управления потоками данных на графических процессорах*

Ключевые слова: *парадигма потоков данных, графические процессоры, распределенные вычисления.*

Summary. *The research carried out an analysis of modern software for managing data flow on graphics processing units.*

Key words: *data flow, graphics processing units, distributed computing.*

Постановка проблеми. Існуюча парадигма програмування – потік інструкцій (англ. control flow) не дає можливості ефективно використовувати обчислювальну потужність багатопроцесорних середовищ.

Виклад основного матеріалу. Застосування розподілених обчислень з розповсюдженням потужних графічних процесорів стає все більш поширеним. Проте існуюча парадигма програмування – потік інструкцій (англ. control flow) не дає можливості ефективно використовувати обчислювальну потужність багатопроцесорних середовищ. Це пояснюється тим, що, в контексті потоку інструкцій, обчислювальні задачі поступають на обробку до процесора послідовно, на зразок конвеєра, що сильно ускладнює обробку в багатопроцесорному середовищі. Рішенням даної проблеми є парадигма потоків даних [1] (англ. data flow), в рамках якої, дані представляються у вигляді графу, де гранями є операнди, а вершинами – операції над операндами. Такий підхід дозволяє «перекласти» проблему розподілення задач на обчислювальну систему, але накладає певні умови, щодо формату вхідних даних [2].

Проте не слід забувати, що парадигма потоків даних це всього лише спосіб представлення даних. Обробка інструкцій в процесорі все одно буде проходити в режимі конвеєра. Цей факт наштовхує на створення проміжного слою між вхідними даними та, безпосередньо, розподіленою системою, задачею якого, буде розбиття вхідних даних на більш дрібні задачі, а також процес управління цими задачами.

Однією з перших спроб створити подібну архітектуру, яка б дозволила ефективно опрацьовувати дані в форматі обчислювального графу на графічному процесорі була описана в статті про гетерогенну архітектуру потоків даних [3]. Дана архітектура представляє собою декілька рівнів абстракції, які беруть на себе відповідальність за декомпозицію вхідних даних на, так звані, Завдання Орієнтовні Модулі (англ. Task Oriented Modules), а також за менеджмент ресурсів.

Завдання Орієнтовний Модуль представляє собою блок з певною кількістю входів та виходів, а також набір імплементацій певної задачі під різні вимоги. Кожен такий ЗОМ «обгортається» віртуальним обробником (англ. Virtual Processing Element), який в свою чергу відповідальний за управління обчислювальними ресурсами, планування задач та певні процеси валідації. Все це працює в рамках виконавчого двигуна (англ. Execution Engine), повну картину якого можна побачити на Рис.1.

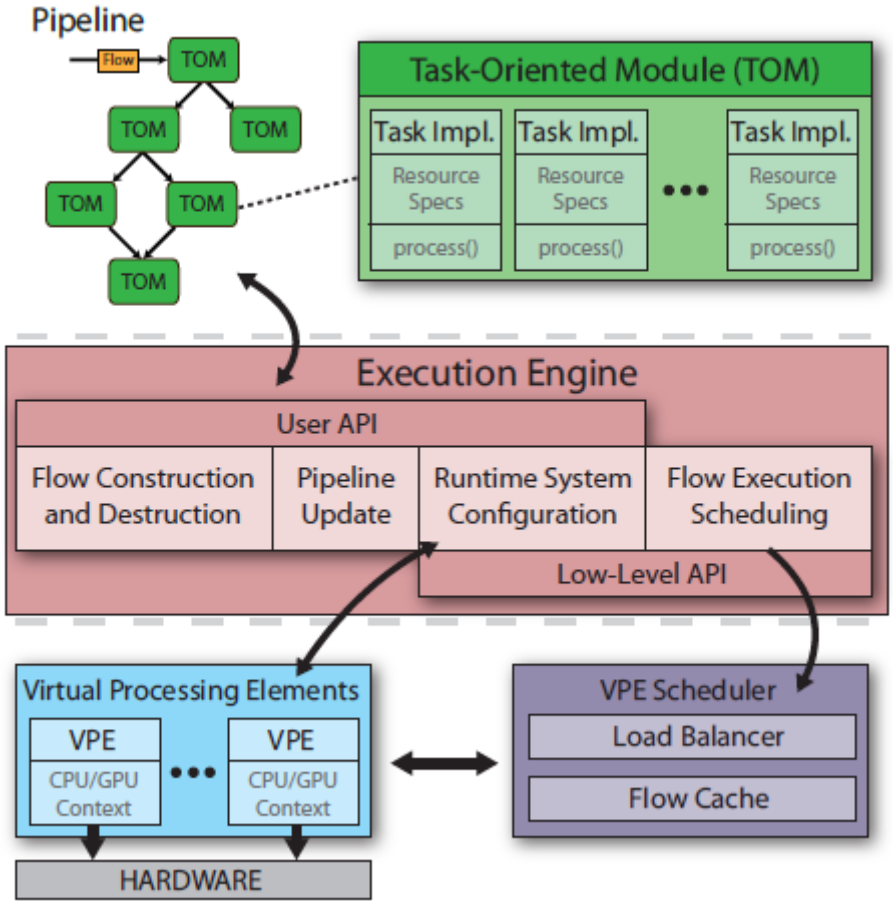

Рис. 1. Архітектура HyperFlow [3]

Головною проблемою даної архітектури є те, що в ній відсутній рівень абстракції для програмних інтерфейсів обчислювального процесора. Тобто одна і та сама задача не може бути виконана і на графічному процесорі з підтримкою CUDA, і на графічному процесорі з підтримкою OpenCL, без безпосередньої її реалізації під вказані процесори. Це забов'язує розробників кросплатформенного програмного забезпечення переписувати одну і ту ж саму програму під кожну платформу, на якій програма має працювати.

Архітектуру, що дозволяє ефективно використовувати парадигму потоків даних з кластерами графічних процесорів описано в статті [4]. Автори також використовують концепцію менеджера ресурсів який має розпоряджатись коли і яку задачу потрібно обчислювати, проте, якщо в HyperFlow задачі, отримані в результаті декомпозиції вхідного графу, ніяк не перетинались в процесі обчислення, то тут завдяки спеціально розробленій бібліотеці паралельних обчислень зі зв'язком, що перекривається (англ. Parallel Computations with Communications Overlap), була налагоджена взаємодія окремих процесів, що безпосередньо дає приріст в швидкості. Архітектура додатку показана на Рис.2.

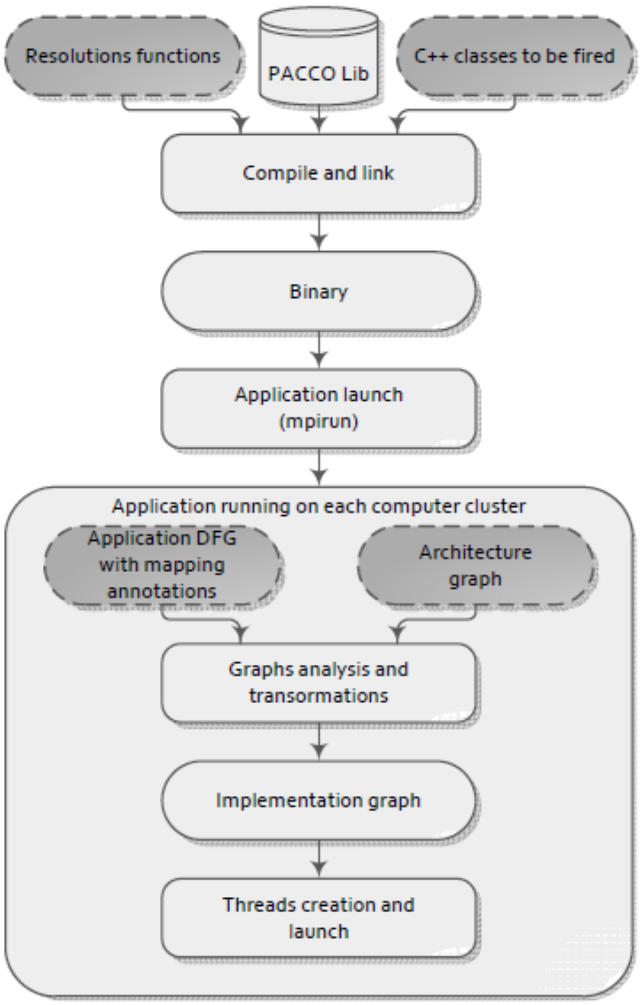

Рис. 2. Архітектура додатку для обчислення графу потоків даних на кластері графічних процесорів [4]

Як видно з Рис.2. для того щоб приступити до обчислення вхідного графу, програма має побудувати на його основі та на основі архітектурного графу (англ. Architecture graph) граф реалізації (англ. Implementation graph), який буде містити в собі інформацію про шлях обчислення вхідного графу, оптимізацію виділення пам'яті, планування задач та їх розподілення між доступними обчислювальними елементами.

Таке рішення має ряд переваг над HyperFlow, проте воно сильно прив'язане до технології з якою воно працює. Мається на увазі те, що дане програмне забезпечення має можливість працювати з графічними процесорами тільки з підтримкою або CUDA або OpenCL. Тобто заміна графічного процесора в кластері може призвести до неможливості виконання

програм, без імплементації під новий графічний процесор. В цьому плані даний додаток уступає перед HyperFlow, адже HyperFlow мав можливість автоматично обирати яку імплементацію задачі використовувати в залежності від процесора, на якому ця задача буде виконуватися.

На сьогоднішній день провідним фреймворком для управління потоками даних на графічних процесорах є TensorFlow [5], розроблений компанією Google. Наряду з Tensorflow, також використовується фреймворк FRED [6], проте він націлений на використання у, так званих, Туманних Обчисленнях (англ. Fog Computing) [7] та Інтернеті Речей (англ. Internet of Things) [8]. Найчастіше TensorFlow застосовується в області машинного навчання (англ. machine learning), це зумовлено тим, що розробники надають широкий інструментарій для розв'язку задач класифікації, регресії, кластеризації та ін. На архітектурному рівні Tensorflow складається з декількох прошарків абстракцій, представлених на Рис.3.

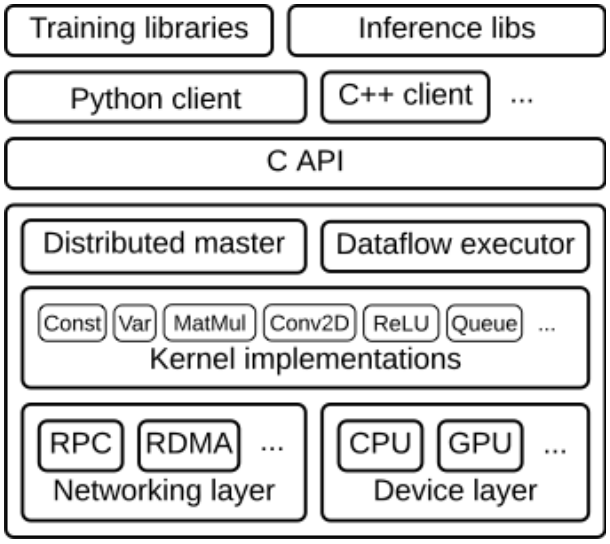

Рис. 3. Загальна архітектура бібліотеки TensorFlow [5]

Як видно з Рис.3, TensorFlow підтримує широкий спектр клієнтів, написаних на Python, C/C++, Java, Go; які використовуються для задання обчислювального графу потоків даних, а також для взаємодії з програмним інтерфейсом фреймворку. Принцип роботи фреймворку схожий з принципом роботи додатку для обчислення графу потоків даних на кластері графічних

процесорів, представленого в цій статті, графічно він виглядає наступним чином Рис.4.

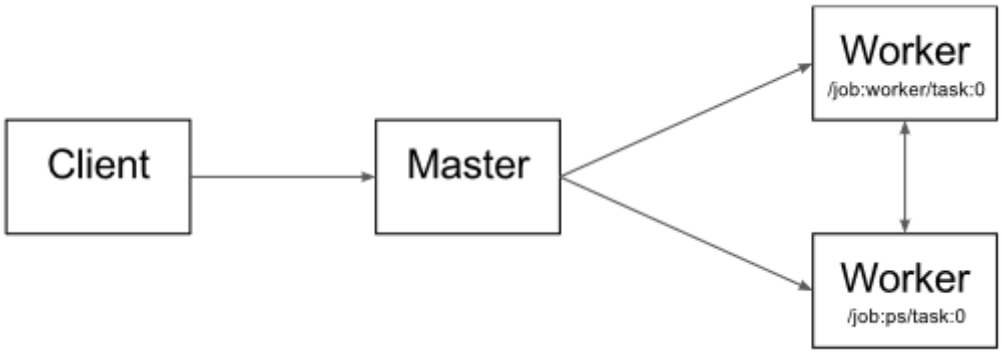

Рис. 4. Взаємодія компонентів TensorFlow [5]

Client задає вхідні дані для обчислення та передає їх в Master, який в свою чергу бере на себе відповідальність за декомпозицію вхідних даних на дрібні задачі, а також за управління цими задачами. Як видно з Рис.4, окремі процеси (Worker) також взаємодіють один з одним, що допомагає зменшити навантаження на Master, та пришвидшити обчислення.

Наразі, TensorFlow підтримує взаємодію з графічними процесорами лише через CUDA, це зумовлено тісною взаємодією ядра TensorFlow з програмним інтерфейсом CUDA.

Висновки з даного дослідження. На сьогоднішній день, обчислювальна потужність процесорів досягла того рівня, коли використання парадигми потоку інструкцій вже не є доцільним, порівняно з парадигмою потоків даних. Це твердження справедливе не тільки для графічних процесорів, адже з розвитком машинного навчання, все більше і більше пристроїв оснащуються спеціальними процесорами для ефективного розв'язку задач машинного навчання, до таких пристроїв належать смартфони, бортові комп'ютери автомобілів та, навіть, пристрої інтернету речей. Саме тому парадигма потоків даних є перспективним і, більше того, актуальним напрямком для розвитку в області розподілених обчислень.

Література

1. Jack B. Dennis, "First version of a data flow procedure language", Symposium on Programming 1974: 362-376
2. Kharchenko K. V., Beznosyk O. U. "The Input File Format for IoT Management Systems Based on a Data Flow Virtual Machine", IEEE International Conference on Dependable Systems, Services and Technologies, 2018
3. Huy T. Vo, Daniel K. Osmari, João Luiz Dihl Comba, Peter Lindstrom, Cláudio T. Silva, "HyperFlow: A Heterogeneous Dataflow Architecture", EGPGV, 2012
4. Vincent Boulos, Sylvain Huet, Vincent Fristot, Luc Salvo, Dominique Houzet, "Efficient implementation of data flow graphs on multi-gpu clusters", Journal of Real-Time Image Processing, March 2014, Volume 9, Issue 1, pp 217–232
5. TensorFlow an open source machine learning framework [Електронний ресурс] — Режим доступу: https://www.tensorflow.org/
6. Flow-based programming for IoT leveraging fog computing [Електронний ресурс] — Режим доступу: https://www2.cs.arizona.edu/~gniady/papers/wetice2017.pdf
7. Bar-Magen Numhauser, Jonathan (2012). Fog Computing introduction to a New Cloud Evolution. Spain: University of Alcala. pp. 111–126. ISBN 978-84-15595-84-7
8. R.S. Raji, "Smart networks for control". June 1994, IEEE Spectrum.