Лабораторна робота №1

**Тема**: CUDA: Інсталяція та основи технології

**Мета**: Одержати практичні навички інсталяції та використання технології CUDA.

1. Теоретичні відомості

Складність обчислювальних завдань вимагає різкого збільшення ресурсів і швидкодії комп’ютерів. Найбільш перспективним напрямком збільшення швидкості розв’язку прикладних завдань є широке впровадження ідей паралелізму в роботу обчислювальних систем. Сьогодні спроектовано і випробувано сотні різних комп'ютерів, що використовують у своїй архітектурі той чи інший вид паралельної обробки даних. Основна складність при проектуванні паралельних програм – забезпечення правильної послідовності взаємодій між різними обчислювальними процесами, а також координація ресурсів, що розділяються між ними.

CUDA розшифровується як Compute Unified Device Architecture і є розширенням мови програмування C, створеним Nvidia. Використання CUDA дозволяє програмісту скористатися потужністю паралельних обчислень відеокарти Nvidia для виконання обчислень загального призначення.

При використанні цієї технології слід знати такі основні поняття:

* **пристрій (device)** – сама відеокарта, графічний процесор (GPU) – виконує команди центрального процесора;
* **хост (host)** – центральний процесор (CPU) – запускає різні завдання на пристрої, виділяє пам'ять тощо;
* **ядро (kernel)** – функція (завдання), що буде виконуватися на GPU.

Запуск та виконання програми на графічному процесорі відбувається у декілька кроків:

1. Хост виділяє необхідну кількість пам’яті на пристрої.
2. Хост копіює дані із своєї пам’яті в пам’ять пристрою.
3. Хост запускає ядро на пристрої.
4. Пристрій виконує це ядро.
5. Хост копіює результати із пам’яті пристрою в свою пам’ять.

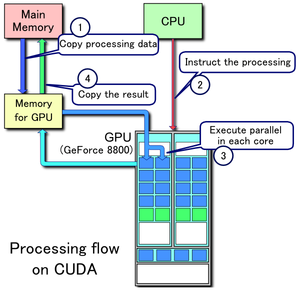


Рисунок 1. Етапи запуску та виконання програми

Хост взаємодіє з графічним процесором через CUDA Runtime API, CUDA Driver API та CUDA Libraries. Runtime та Driver API відрізняються між собою рівнем абстракції. Тобто, перший варіант є більш високого рівня у плані програмування, більш абстрактний, а другий – навпаки більш низького (на рівні драйвера). Загалом, Runtime API є абстрактною обгорткою над Driver API. Під час програмування, ви можете використовувати будь-який варіант.

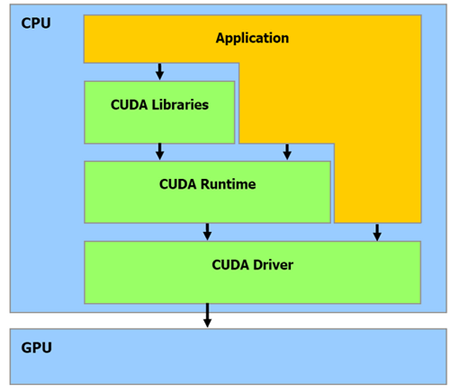


Рисунок 2. Взаємодія CPU та GPU

Такі процесори, як Intel Core 2 Duo та AMD Opteron, добре справляються з одним або двома завданнями одночасно, і виконують ці завдання дуже швидко. З іншого боку, графічні карти добре справляються з великою кількістю завдань одночасно, і виконують ці завдання відносно швидко, бо архітектура GPU побудована трохи інакше, ніж CPU. Оскільки графічні процесори спочатку використовувалися тільки для графічних розрахунків, що припускають незалежну паралельну обробку даних, то GPU призначений саме для паралельних обчислень. Він спроектований так, щоб виконувати велику кількість потоків (елементарних паралельних процесів). GPU містить дуже багато простих арифметико-логічних пристроїв ([АЛП](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%BE-%D0%BB%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%B8%D1%81%D1%82%D1%80%D1%96%D0%B9)), які об'єднані в декілька груп і мають спільну пам'ять (рис. 3). Це допомагає підвищити продуктивність в обчислювальних завданнях, але трохи ускладнює програмування. **Для досягнення найкращого прискорення необхідно продумувати стратегії доступу до пам'яті й враховувати апаратні особливості.**

GPU орієнтований на виконання програм з великим об'ємом даних та розрахунків і являє собою масив потокових процесорів (Streaming Processor Array), що складається з кластерів текстурних процесорів (Texture Processor Clusters, TPC) (рис. 4). TPC складається з набору мультипроцесорів (SM – Streaming Multi-processor), кожен з яких містить кілька потокових процесорів (SP – Streaming Processors) або ядер (у сучасних графічних процесорах к-сть ядер перевищує 1024). Набір ядер кожного мультипроцесора працює за принципом [SIMD](https://uk.wikipedia.org/wiki/SIMD) (проте, з деякою відмінністю) – реалізація, що дозволяє групі процесорів, що працюють паралельно, працювати з різними даними, але при цьому всі вони в будь-який момент часу повинні виконувати однакову команду. Якщо по-простому, то декілька потоків виконують одне і те ж саме завдання.

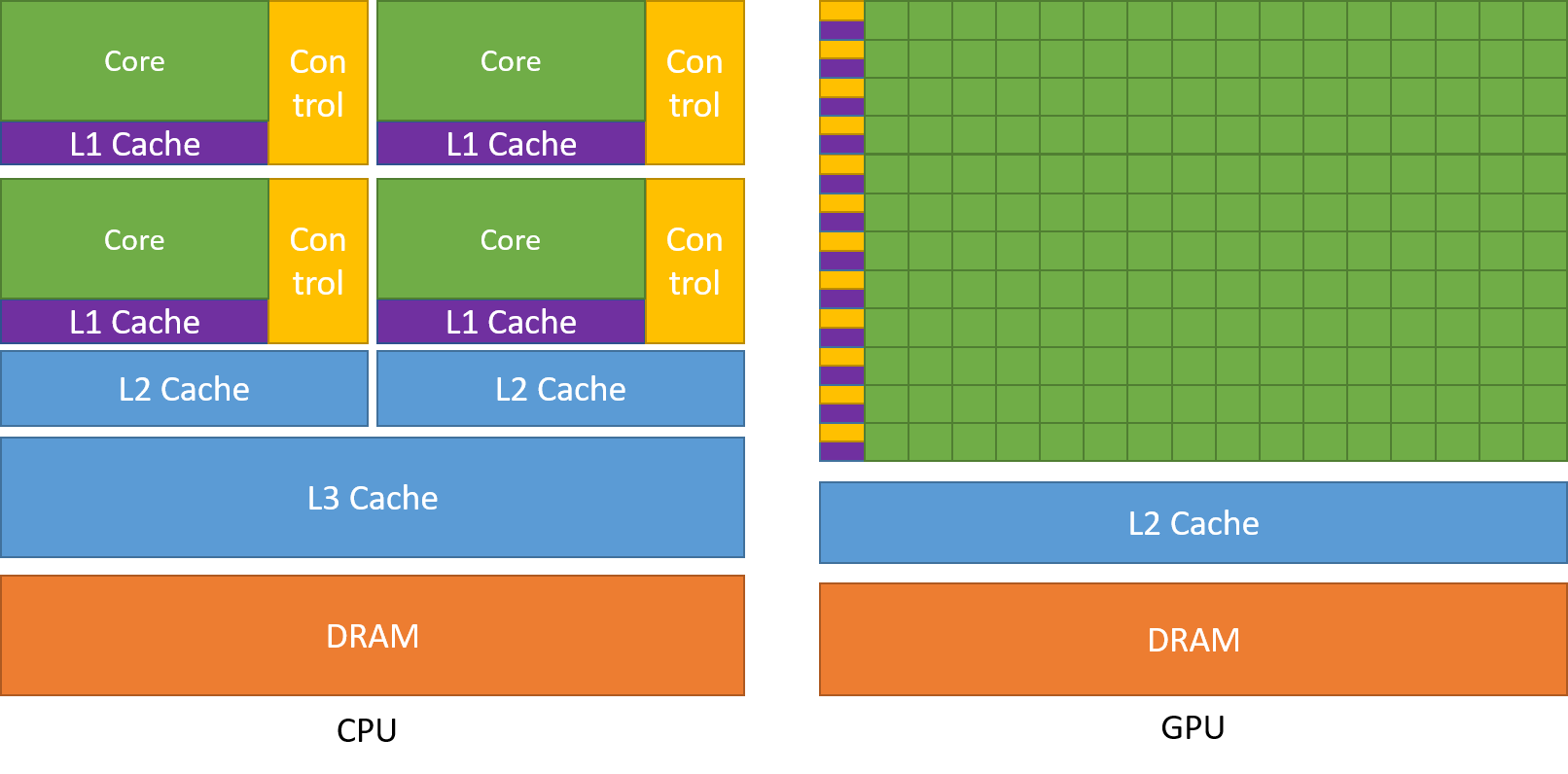


Рисунок 3. Відмінності архітектури CPU від GPU

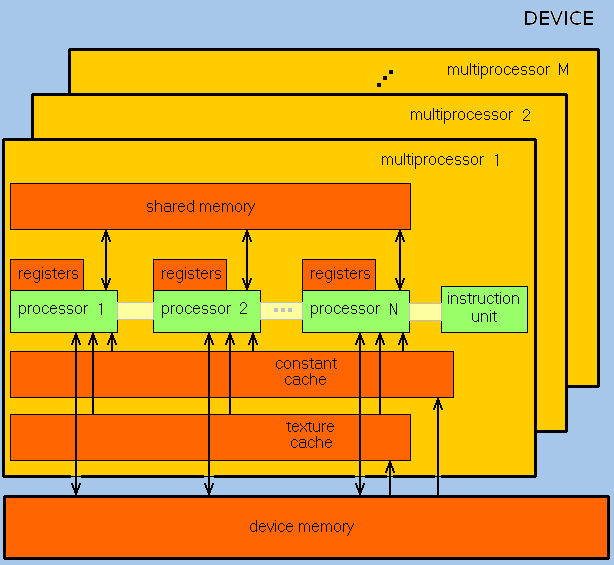


Рисунок 4. Мультипроцесори GPU

CUDA використовує велику кількість окремих потоків для розрахунків. Всі вони групуються в ієрархію – grid / block / thread. Верхній рівень – grid – відповідає ядру й об'єднує всі потоки, що виконують дане ядро. Grid – одновимірний або двомірний масив блоків (block). Кожен блок (block) являє собою 1 / 2 / 3 -мірний масив потоків (threads). При цьому кожен блок являє собою повністю незалежний набір скоординованих між собою потоків. Потоки з різних блоків не можуть між собою взаємодіяти.

Є ще таке поняття, як warp – група з 32 потоків. Так ось, тільки потоки в межах однієї групи (warp) можуть фізично виконуватися одночасно. Потоки різних варпів можуть знаходитися на різних стадіях виконання програми. Такий метод обробки даних позначається терміном SIMT (Single Instruction – Multiple Theads). Управління роботою варпів виконується на апаратному рівні (рис. 5).

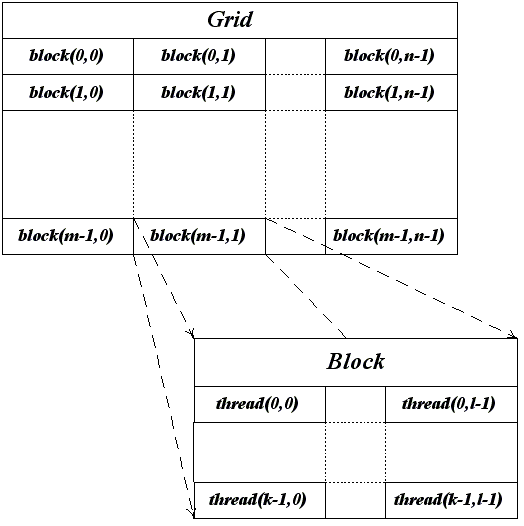


Рисунок 5. Ієрархія потоків.

Одна важлива річ, про яку слід пам’ятати, полягає в тому, що всю вашу програму не потрібно писати на CUDA. Якщо ви пишете велику програму з інтерфейсом користувача та багатьма іншими функціями, то більша частина вашого коду буде написана мовою C++ або іншою мовою, яку ви виберете. Тоді, коли потрібно щось надзвичайно інтенсивне з точки зору обчислень, ваша програма може просто викликати функцію ядра CUDA, яку ви написали.

Отже, основна ідея полягає в тому, що CUDA слід використовувати лише для найбільш інтенсивних обчислювальних частин вашої програми. Найважливішим моментом є те, що CUDA добре підходить лише для обчислень, які можуть бути розбиті та виконані тисячами потоків паралельно.

2. Хід роботи.

**Інсталяція CUDA на Windows ПК із графічним процесором Nvidia**

1. Перевірити чи графічний процесор, що використовується, підтримує технологію CUDA. Зробити це можна на сайті <https://developer.nvidia.com/cuda-gpus>.
2. Перейти на сайт <https://developer.nvidia.com/cuda-downloads> та завантажити інсталяційний файл відповідно до операційної системи.
3. Запустити інсталятор та дотримуватися усіх кроків (оновити драйвери графічного процесора, якщо потрібно).
4. Після успішної інсталяції виконати в терміналі команду nvcc -V. На екрані повинна відобразитися інформація про версію CUDA.

**Інсталяція CUDA у віртуальному середовищі Google Colab**

1. Відкрити файл <https://colab.research.google.com/drive/1uLpXx-qlgUN-AntDZtMI7yKfF8pVOoxa?usp=sharing> та виконати інструкції.

3. Структура звіту лабораторної роботи.

* Титульна сторінка.
* Тема та мета роботи.
* Копія екрану з результатом виконання команди nvcc -V.
* Висновки.

4. Контрольні запитання

1. Що таке CUDA?

2. Основні системні вимоги для інсталяції СUDA?

3. Переваги GPU над CPU?

Список літератури:

1. Вступ до технології CUDA - <https://codeguida.com/post/1150>
2. CUDA Toolkit - <https://developer.nvidia.com/cuda-toolkit>