# Дослідження алгоритмів роботи з розрідженими матрицями в середовищі CUDA

Студент гр. ДА-62
Горбик Олександр
Науковий керівник
Кирюша Богдан Анатолійович

# Задачі, що розв’язувались

- **Дослідження основних технологій обчислень загального призначення на графічних прискорювачах (GPGPU).**
- **Дослідження уніфікованої архітектури CUDA.**
- **Аналіз можливостей реалізації алгоритмів роботи з розрідженими матрицями, використовуючи CUDA.**
- **Аналіз отриманих результатів.**

# Чому розріджені матриці?

- **Розріджені матриці виникають в багатьох прикладних дисциплінах, зокрема, в схемотехніці та моделювання фізичних процесів.**
- **Ефективність роботи алгоритмів з розрідженими матрицями є критичною для багатьох алгоритмів.**
- **Операція множення розрідженої матриці на вектор-стовбець лежить в основі багатьох алгоритмів.**

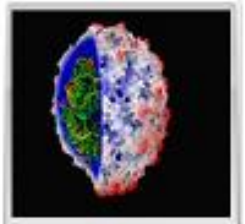
# Чому CUDA?

- **Використання переваг технології GPGPU.**
- **Наявність «продвинутих» архітектур, оптимізованих для наукових обчислень (лінійка Tesla та Fermi).**
- **«Зрілість» засобів розробки, наявність оптимізованих бібліотек та відкритих прикладів роботи з API.**
- **Розвинуте API.**
- **Уніфікована архітектура.**

# Чому CUDA?

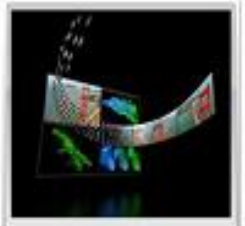
**146X**

Interactive visualization of volumetric white matter connectivity[1]



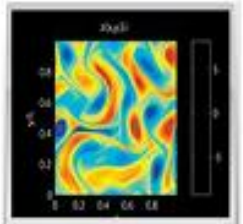
**36X**

Ionic placement for molecular dynamics simulation on GPU[2]



**18X**

Transcoding HD video stream to H.264 for portable video[3]

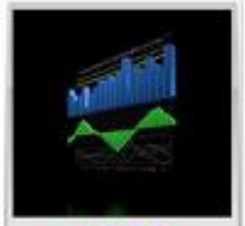


**17X**

Simulation in Matlab using .mex file CUDA function[4]

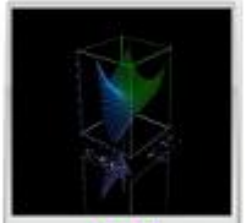
**100X**

Astrophysics N-body simulation[5]



**149X**

Financial simulation of LIBOR model with swaptions[6]



**47X**

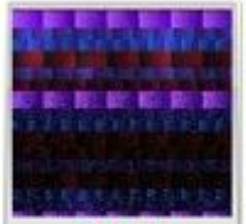
GLAME@lab: M-script API for linear Algebra operations on GPU[7]

**20X**

Ultrasound medical imaging for cancer diagnostics[8]

**24X**

Highly optimized object oriented molecular dynamics[9]



**30X**

Cmatch exact string matching - find similar proteins & gene sequences[10]

# Compute Unified Device Architecture

- **Обчислювальне ядро (“compute engine”) графічних прискорювачів NVIDIA.**
- **Пропрієтарна технологія корпорації.**
- **Програмно-апаратний комплекс, що дозволяє організовувати обчислення на стороні графічних прискорювачів.**
- **Масивно-паралельна архітектура SIMT (Single Instruction Multiple Thread).**

# Ефективна робота з технологією

**Три основні стратегії оптимізації алгоритмів:**

- **збільшення ефективності використання графічного прискорювача;**
- **оптимізація використання пам’яті;**
- **оптимізація використання інструкцій.**

**Оптимізація відрізняється від класичних стратегій для центрального процесора.**

# Розріджені матриці в середовищі CUDA

# Формати матриць

**Формати представлення розріджених матриць:**

- **Diagonal (DIA);**
- **ELLPACK (ELL);**
- **Coordinate (COO);**
- **Compressed Sparse Row (CSR);**
- **ALLTED (ALLTED).**

# Формати матриць

## Текстовий формат

$$\mathtt{A} = \begin{bmatrix} 1 & 7 & 0 & 0 \\ 0 & 2 & 8 & 0 \\ 5 & 0 & 3 & 9 \\ 0 & 6 & 0 & 4 \end{bmatrix}$$

## DIA

$$\mathtt{data} = \begin{bmatrix} * & 1 & 7 \\ * & 2 & 8 \\ 5 & 3 & 9 \\ 6 & 4 & * \end{bmatrix} \quad \mathtt{offsets} = \begin{bmatrix} -2 & 0 & 1 \end{bmatrix}$$

## ELL

$$\mathtt{data} = \begin{bmatrix} 1 & 7 & * \\ 2 & 8 & * \\ 5 & 3 & 9 \\ 6 & 4 & * \end{bmatrix}$$

$$\mathtt{indices} = \begin{bmatrix} 0 & 1 & * \\ 1 & 2 & * \\ 0 & 2 & 3 \\ 1 & 3 & * \end{bmatrix}$$

## COO

$$\begin{aligned} \mathtt{row} &= \begin{bmatrix} 0 & 0 & 1 & 1 & 2 & 2 & 2 & 3 & 3 \end{bmatrix} \\ \mathtt{col} &= \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 2 & 3 & 1 & 3 \end{bmatrix} \\ \mathtt{data} &= \begin{bmatrix} 1 & 7 & 2 & 8 & 5 & 3 & 9 & 6 & 4 \end{bmatrix} \end{aligned}$$

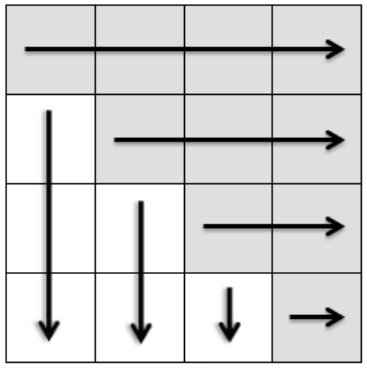
## CSR

$$\begin{aligned} \mathtt{ptr} &= \begin{bmatrix} 0 & 2 & 4 & 7 & 9 \end{bmatrix} \\ \mathtt{indices} &= \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 2 & 3 & 1 & 3 \end{bmatrix} \\ \mathtt{data} &= \begin{bmatrix} 1 & 7 & 2 & 8 & 5 & 3 & 9 & 6 & 4 \end{bmatrix} \end{aligned}$$

# Формати матриць

## ALLTED формат

| | | | |
|---|---|---|---|
| 1 | 7 | 0 | 0 |
| 0 | 2 | 8 | 0 |
| 5 | 0 | 3 | 9 |
| 0 | 6 | 0 | 4 |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Data | 1 | 7 | 2 | 8 | 3 | 9 | 4 | 5 | 6 |

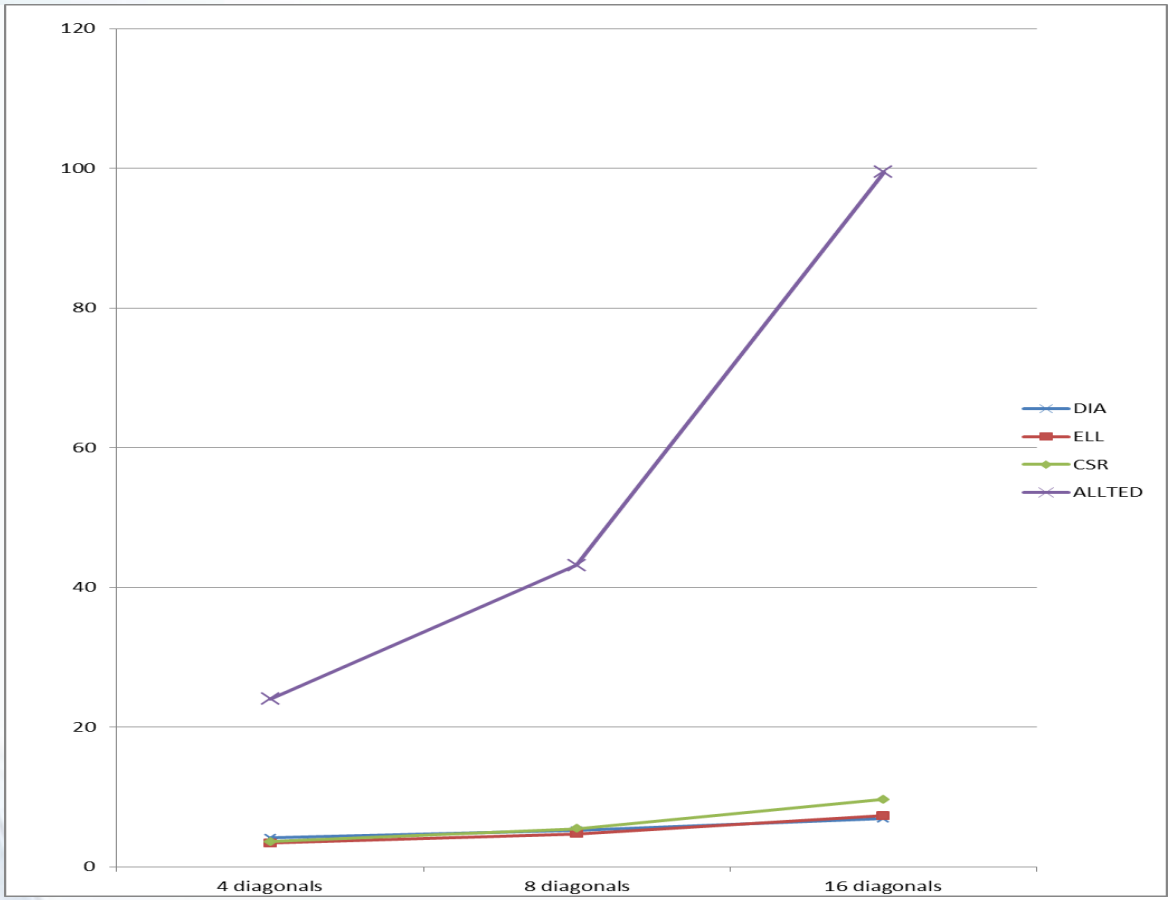
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Row | 1 | 3 | 5 | 7 | 8 | 9 | 10 | 10 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Offset | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 3 | 4 |

# Отримані результати

**5000x5000**
**200 ітерацій**
**GeForce 480M**
**Windows 7**

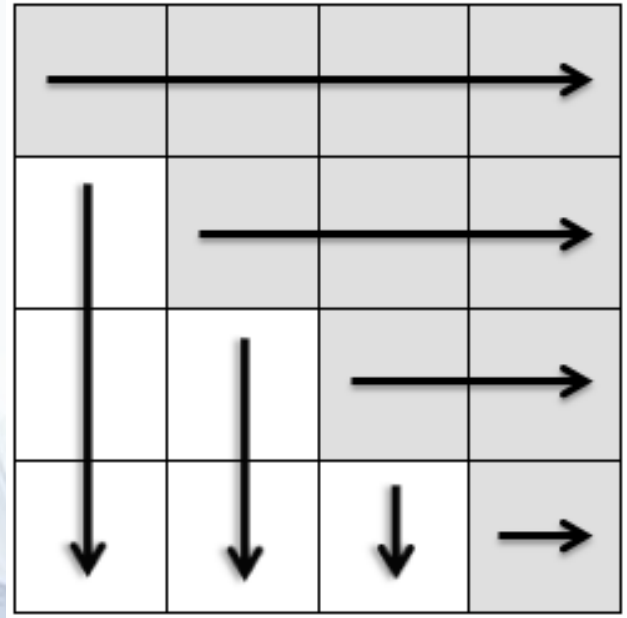
**Діагональний патерн розта-шування нену-левих елементів**



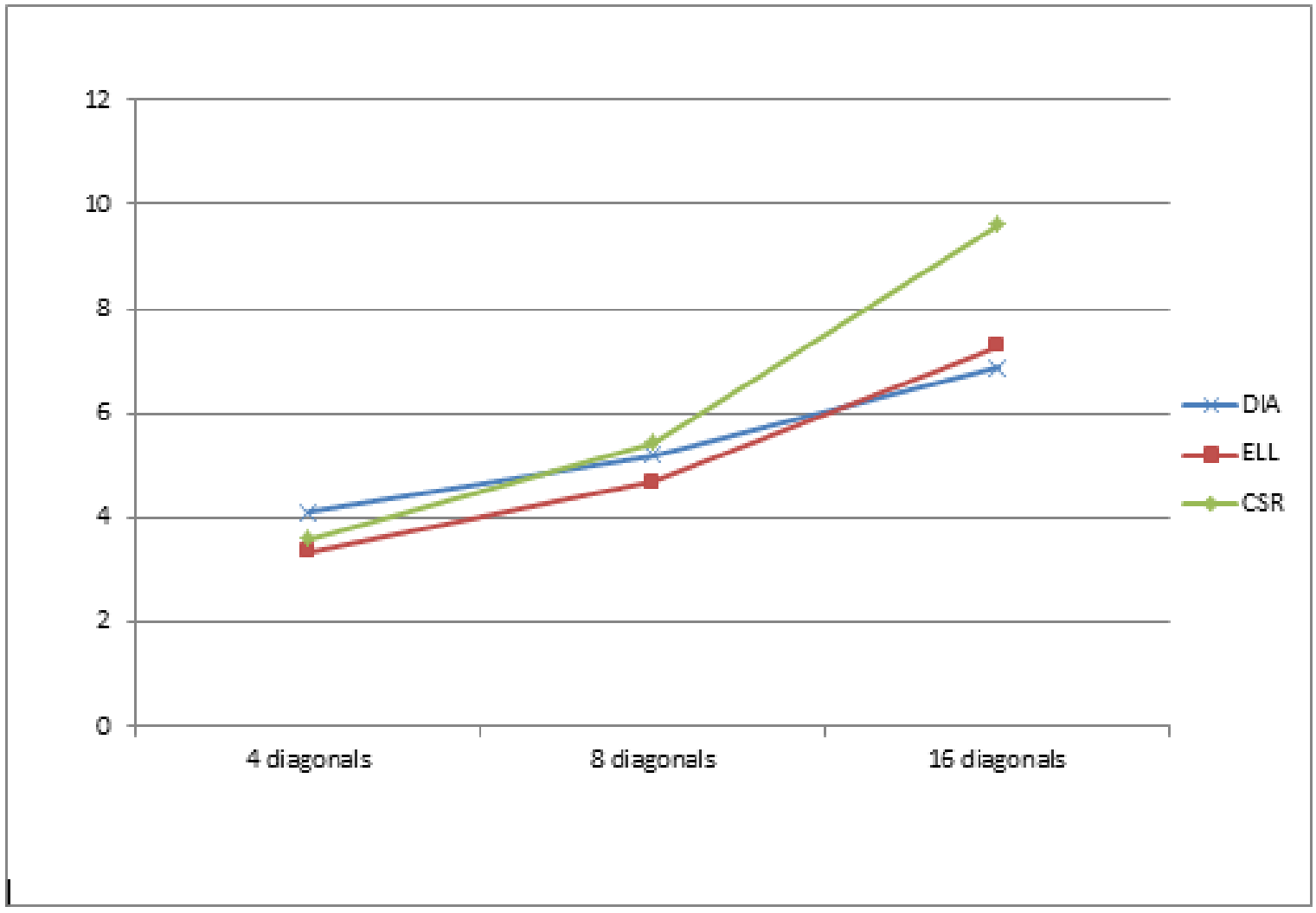

**4 діагоналі**
**{-2, 0, 2, 3}**
**8 діагоналей**
**{-4, -3, -2, 0, 2, 3, 4, 5}**
**16 дігоналей**
**{-8, -7, -6, -5, -4, -3, -2, 0, 2, 3, 4, 5, 6, 7, 8, 9}**

# Формати матриць

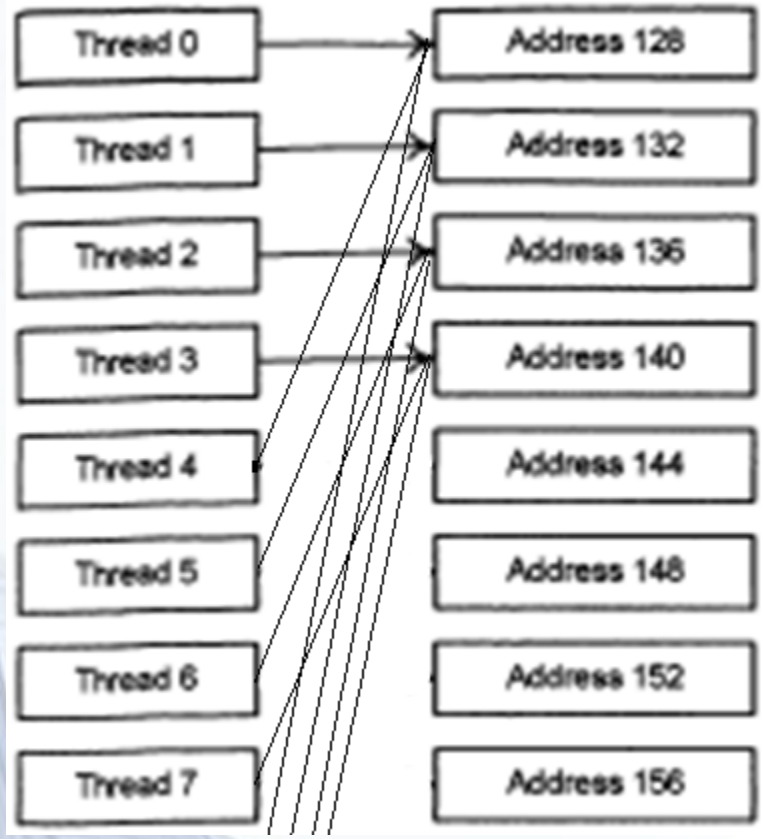
## ALLTED формат



# Отримані результати



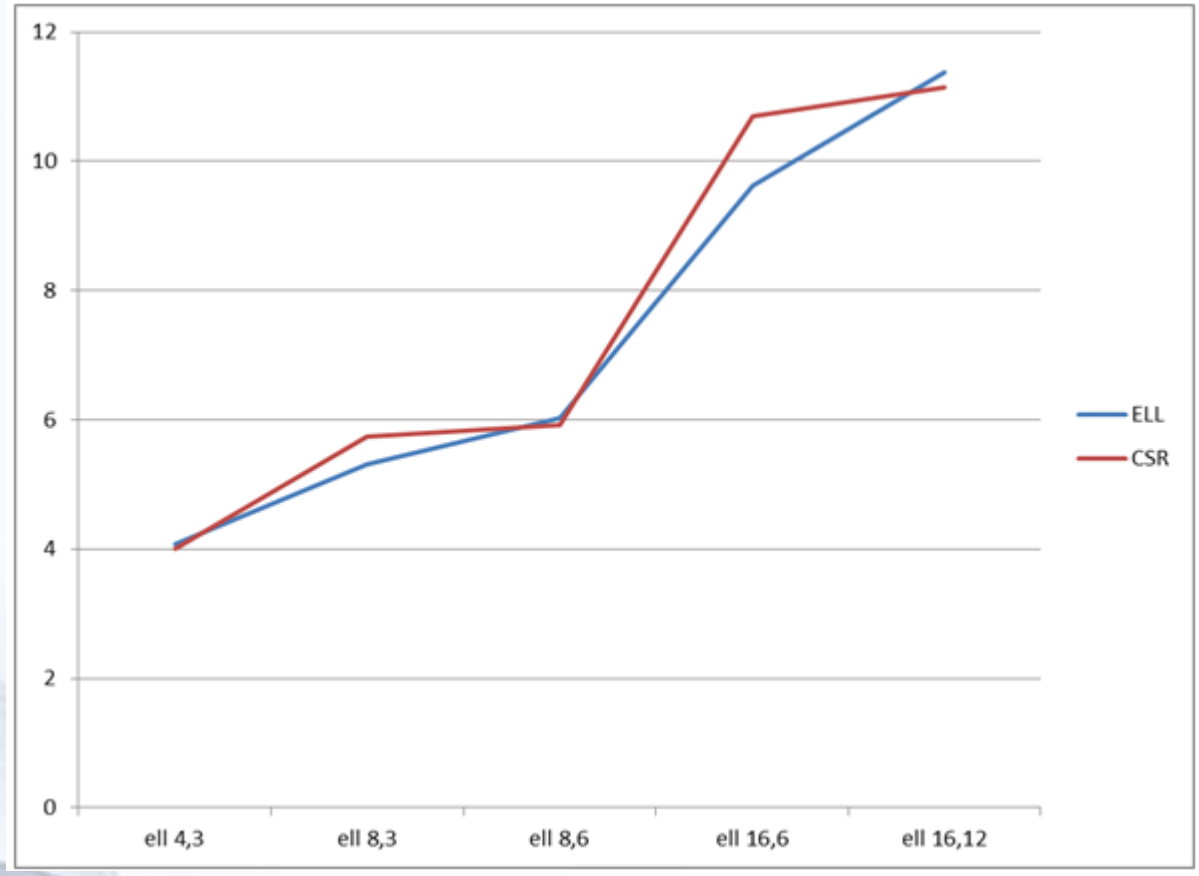
# Патерн звернення до offset масиву формату DIA



# Отримані результати

**Випадковий патерн розподілу ненулевих елементів:**

**1 середня кількість елементів**
**2 відхилення від середньої кількості**



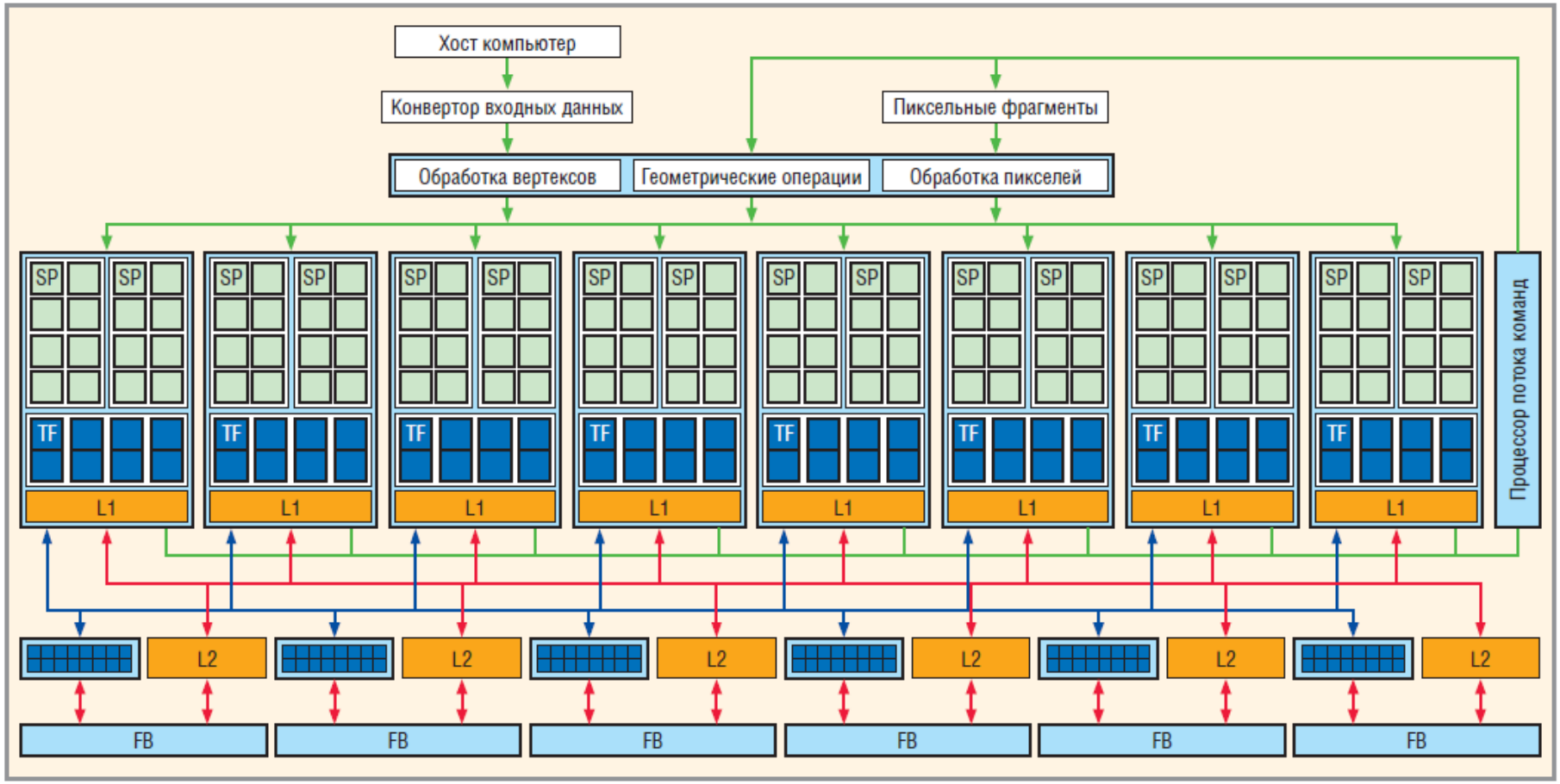
# Висновки

- **Досліджено CUDA (сформульовані стратегії по оптимізації роботи алгоритмів).**
- **Реалізовано функцію ядра для проведення множення матриць в форматі ALLTED.**
- **Реалізовано програму для тестування швидкості виконання операцій множення та рішення СЛАР.**
- **На майбутнє: дослідження можливості використання декількох GPU та оптимізація шляхом використання текстурної пам’яті.**

# Дякую за увагу

**Let’s discuss!**

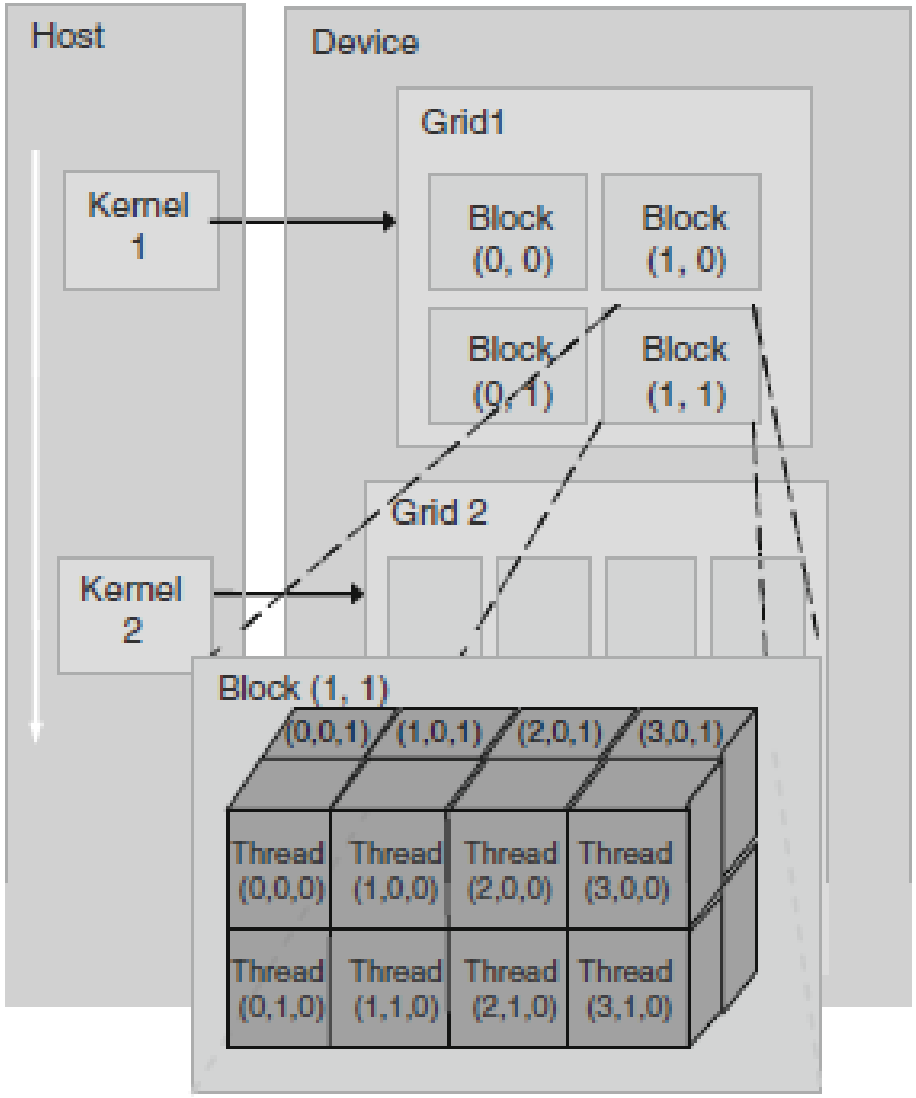
# Архітектура ядра G80



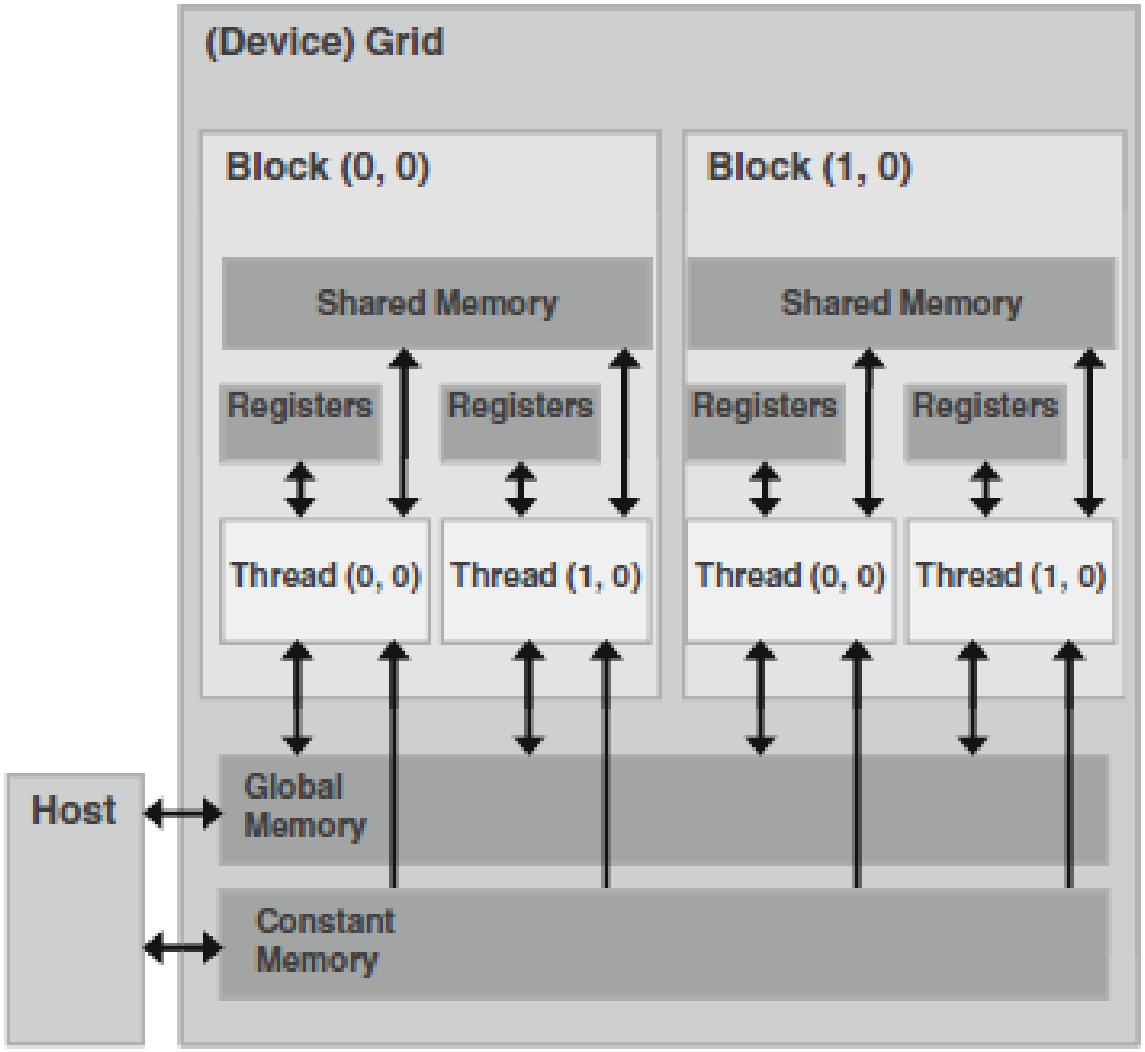
# Порівняння основних технологій

| | CUDA C++ | OpenCL | C++ AMP | OpenGL |
|---|---|---|---|---|
| Доступность технологии | Проприетарная технология NVidia | Открытый стандарт, Khronos group | Открытый стандарт, Microsoft | Открытый стандарт, Khronos group |
| Наличие программных средств для разработки | SDK, NSight – debugger и profiler | Стандарт не подразумевает реализации (AMD предоставляет SDK, debugger) | Интеграция с Microsoft Visual Studio следующего релиза (debugger, profiler) | Стандарт не подразумевает конкретной реализации |
| Использование нескольких устройств | Да | Да. Использование GPU и CPU | | Нет |
| Основные недостатки технологи перед конкурентами | Ориентирована на поставщика – закрытый стандарт | Реализация зависит от поставщика – необходимость дополнительной оптимизации | Новый стандарт, поддержка в тестовом режиме | Технология используется штучно – предназначена для обработки графики |
| Основные преимущества | Наиболее развитое API и множество расширений, специализированная архитектура процессоров | Открытый стандарт, поддержка гетерогенных вычислений (CPU + GPU) | Декларированная поддержка вычислений в облачных системах | Поддержка всеми основными поставщиками |

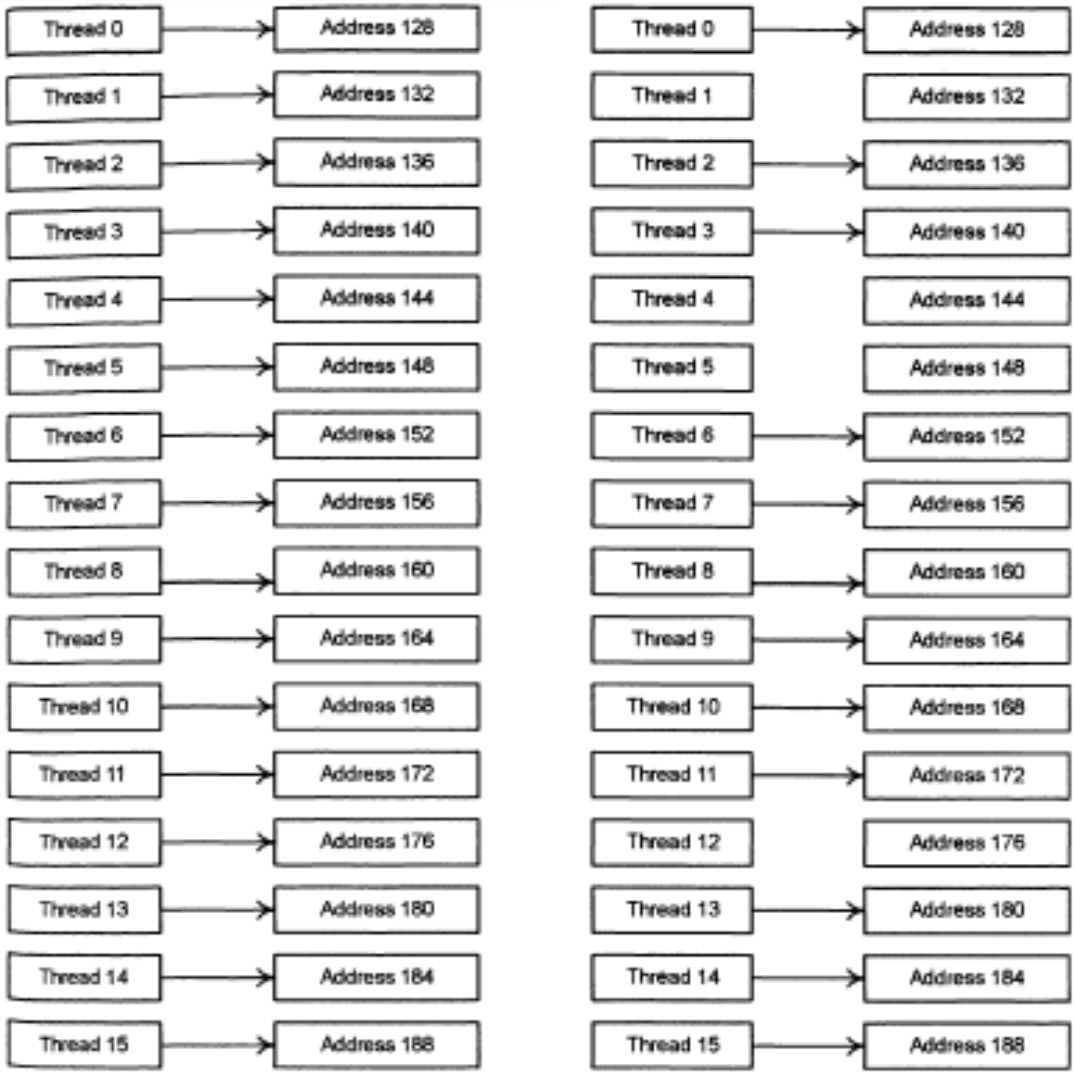
# Ієрархія потоків CUDA



# Ієрархія пам’яті CUDA



# Узгоджене звернення до пам’яті



# Використання графічної карти

**Рівні оптимізації:**

- **застосування (application level);**
- **графічної карти (device level);**
- **мультипроцесора (multiprocessor level ).**

# Оптимізація використання пам’яті

- мінімізувати використання повільної пам’яті.
- мінімізувати передачу даних між хостом та графічним прискорювачем.
- оптимізувати доступ до пам’яті:
  - глобальної;
  - спільної;
  - константної;
  - текстурної.

Пам’ять як обмеження паралелізму.

# Оптимізація використання інструкцій

- **зменшити кількість повільних інструкцій (зменшуючи точність);**
- **використання спеціальних математичних функцій з апаратною підтримкою.**
- **зменшити кількість операторів керування потоком виконання.**

# Модель взаємодії GPU та CPU

$$T = t_{dataalloc} + t_{forwardtransfer} + t_{grid} + t_{backtransfer} + t_{datafree}$$

$$t_{grid} = \left\lceil \frac{X_{gdim} * Y_{gdim} * Z_{gdim}}{N_{MP}} \right\rceil * t_{block}$$

$$t_{block} = N_{warp} * t_{warp} = \left\lceil \frac{X_{bdim}}{32} \right\rceil * Y_{bdim} * Z_{bdim} * t_{warp}$$