



# Введение в CUDA на C++11

Лекторы:
Аспирант МФТИ, Шер Артём Владимирович
Аспирант МФТИ, Зингеренко Михаил Владимирович

22 октября 2024

# Введение в CUDA на C++11

CUDA (Compute Unified Device Architecture) — это параллельная вычислительная платформа, разработанная NVIDIA.

- Она позволяет использовать вычислительные возможности графических процессоров (GPU) для выполнения общего назначения, а не только для графики.
- CUDA поддерживает стандарты C++11, что позволяет интегрировать современные возможности языка C++ с высокопроизводительными вычислениями на GPU.
- Это значительно ускоряет задачи, требующие больших вычислительных ресурсов, такие как обработка изображений, моделирование физических процессов и обучение нейронных сетей.

# Основная функциональность CUDA

Основные концепции CUDA: Нити (Threads), блоки (Blocks) и сетки (Grids)

- Thread — это базовая единица выполнения на GPU.
- Block — это группа нитей, которую выполняет одно и тот же ядро (kernel) и может делиться памятью.
- Grid — это множество блоков, которые запускают одну и ту же задачу.

Ядра CUDA (kernels) — это функции, которые исполняются на GPU. Каждая нить исполняет одно и то же ядро параллельно.

# Иерархия памяти CUDA

- Глобальная память: доступна всем нитям, но она медленная.
- Совместная память (Shared memory): быстрее глобальной и доступна только нитям внутри одного блока.
- Регистры: самая быстрая память, но она доступна только отдельным нитям.

# Основные функции CUDA

- cudaMalloc: Выделение памяти на GPU.
- cudaMemcpy: Копирование данных между хостом (CPU) и устройством (GPU).
- cudaFree: Освобождение выделенной памяти на GPU.

# Пример CUDA — Сложение векторов

Код ядра:

```
__global__ void vectorAdd(const float *A, const float *B, float *C, int N) {
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < N) {
        C[i] = A[i] + B[i];
    }
}
```

# Пример CUDA — Сложение векторов

- __global__: Определяет функцию как ядро, которая будет запускаться на GPU.
- cudaMalloc, cudaMemcpy, cudaFree: Управление памятью для передачи данных между CPU и GPU.

```
int threadsPerBlock = 256;
int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;
vectorAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
```

**До следующей лекции!**