Parallel Computing Workshop

Ms. Kranti Ingale (CS22D003)
Prime Minister's Research Fellows (**PMRF**)
PACE Lab - CSE, IIT Madras

Roadmap

- Introduction to GPU
- CUDA Program Flow and CPU-GPU Communication
- Thread organization (Grids, Blocks, Threads, 1D/2D)
- CUDA Memory Model
- CUDA Functions
- CUDA Thrust

Roadmap

- Introduction to GPU
- CUDA Program Flow and CPU-GPU Communication
- Thread organization (Grids, Blocks, Threads, 1D/2D)
- CUDA Memory Model
- CUDA Functions
- CUDA Thrust

Introduction

- A Graphics Processing Unit (GPU) is a microprocessor designed specifically for the processing of 3D graphics.
- GPU forms the heart of modern graphics card, relieving the CPU (Central Processing Unit) of much of the graphics processing load.
- GPU has become integral part of today’s computer systems.
- Over the last few years, there has been remarkable increase in performance and capabilities of GPU

Why GPU?

- To provide separate graphics processing resources
- To relieve some of the burden of the main system resources

GPU VS CPU

GPU

- Tailored for highly parallel operations
- GPUs have many parallel execution units
- Many-core

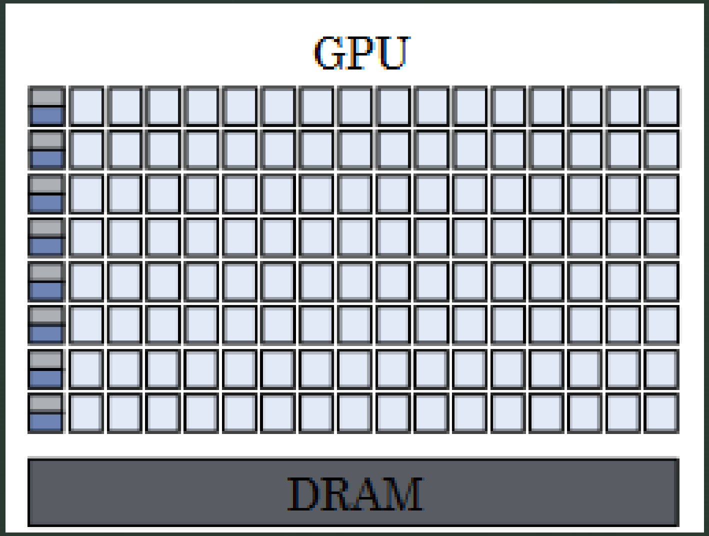
CPU

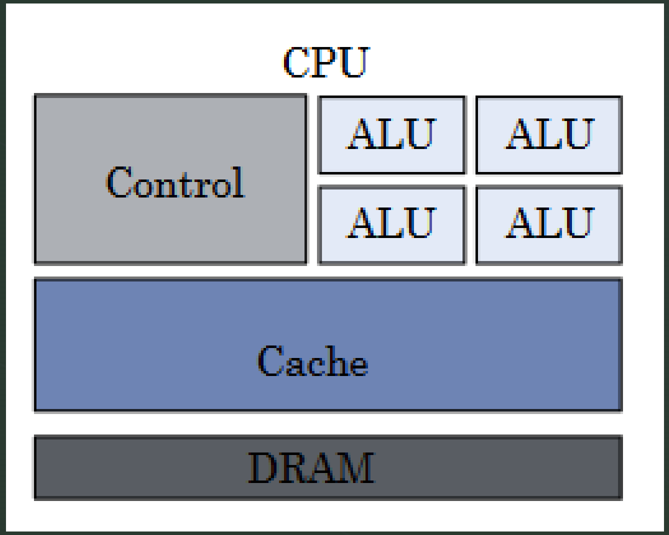
- CPU executes programs sequentially
- CPUs have few execution unit
- Multi-core

GPU VS CPU

GPU

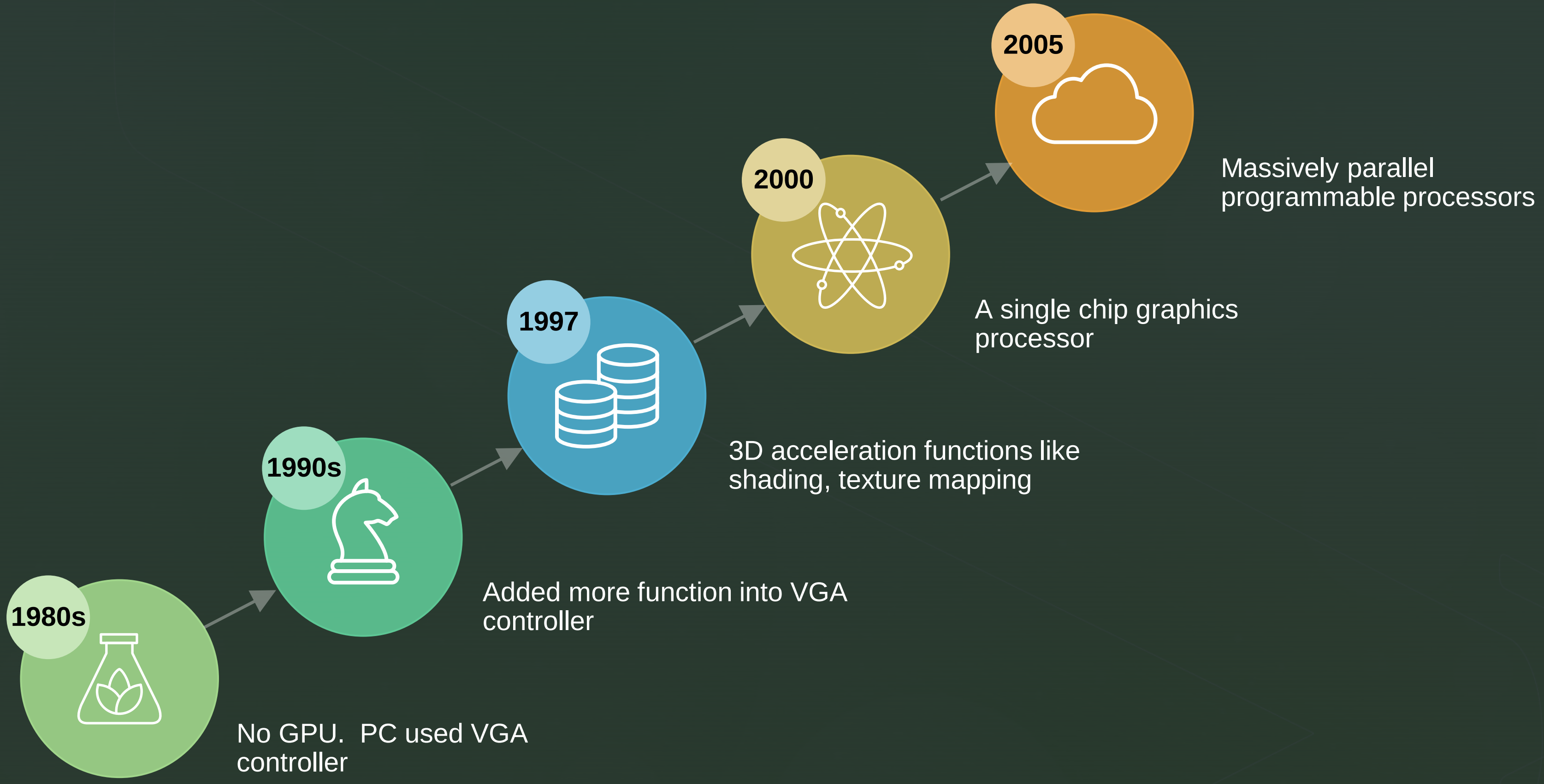
CPU



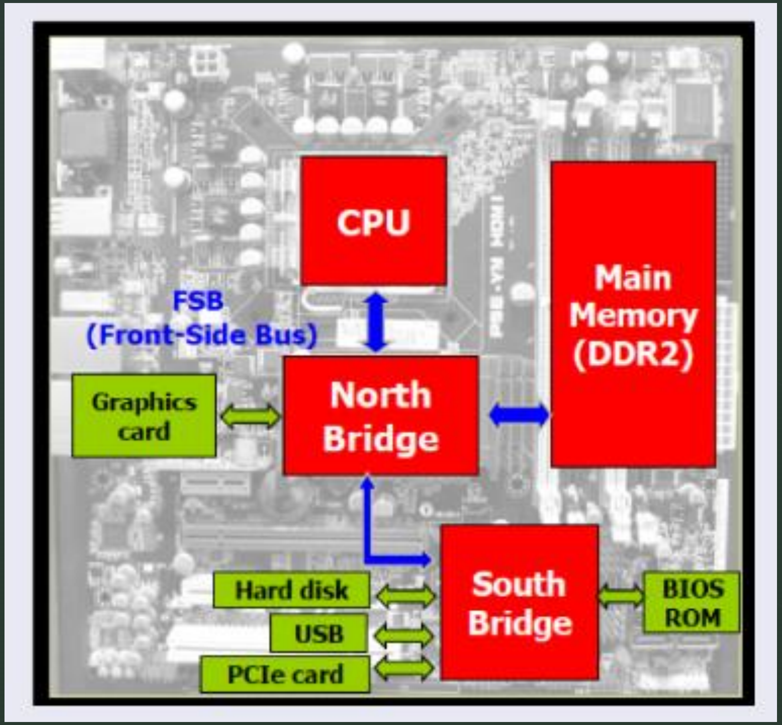


GPU Evolution

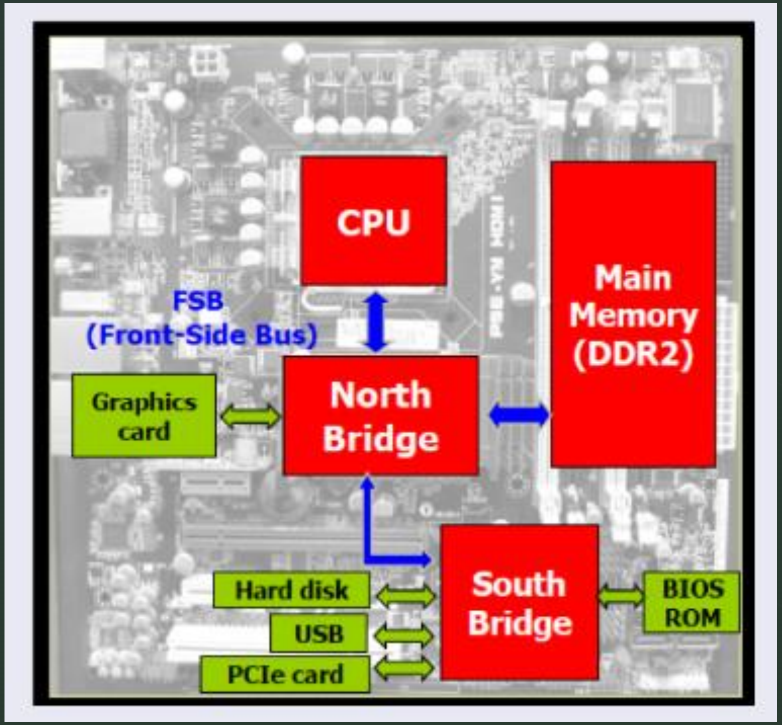
Highly parallel, highly multithreaded multiprocessor optimized for graphic computing and other applications

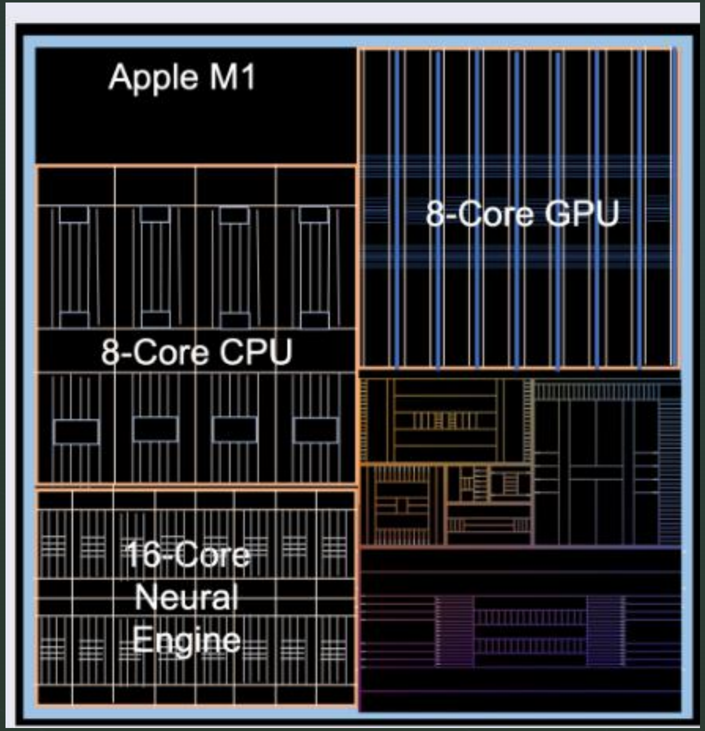

1980s
No GPU. PC used VGA controller
1990s
Added more function into VGA controller
1997
3D acceleration functions like shading, texture mapping
2000
A single chip graphics processor
2005
Massively parallel programmable processors

GPUs: Discrete to Integrated



GPUs: Discrete to Integrated





GPU Vendors

- NVIDIA
- AMD
- Intel
- Qualcomm
- ARM
- …

GPU Programming

- GPU Programming empowers developers to harness the immense computational power of GPUs
- General Purpose computation (GP GPU): using GPU in applications other than 3D graphics
- Applications
  - Game effects (FX) physics, image processing
  - Physical modeling, computational engineering, matrix algebra, convolution, correlation, sorting

GPU Languages

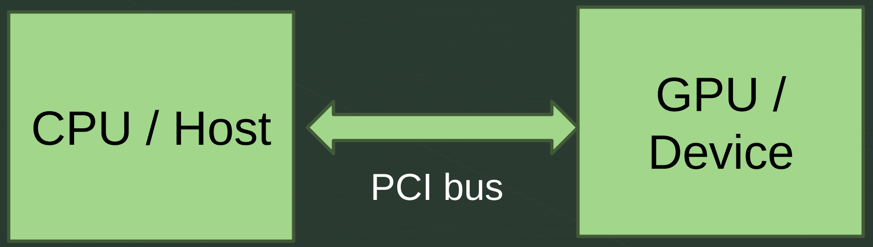
- CUDA (compute unified device language)
- OpenCL (open computing language)
- OpenACC (open accelerator)
- Sycl

GPU Languages

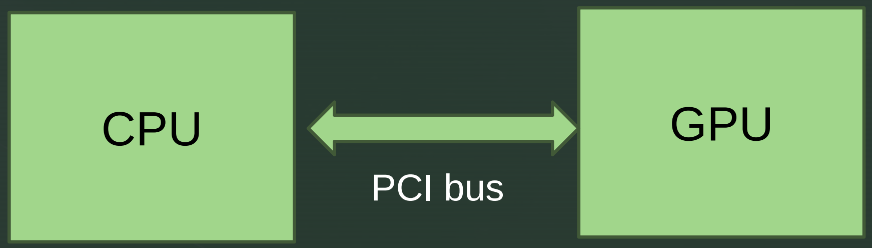
- CUDA (compute unified device language)
- OpenCL (open computing language)
- OpenACC (open accelerator)
- Sycl

Terminology

- **Host** : The CPU and its memory (host memory)
- **Device** : The GPU and its memory (device memory)

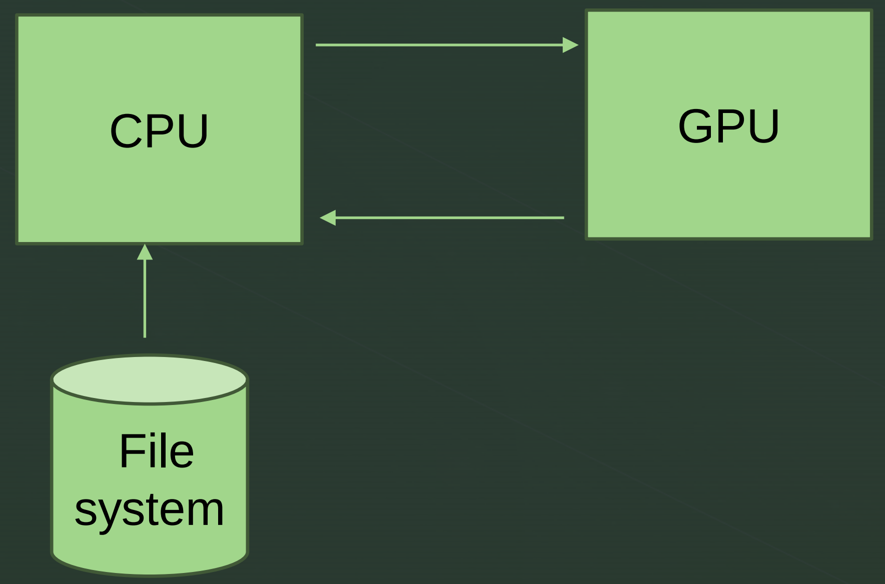


Separate Memories



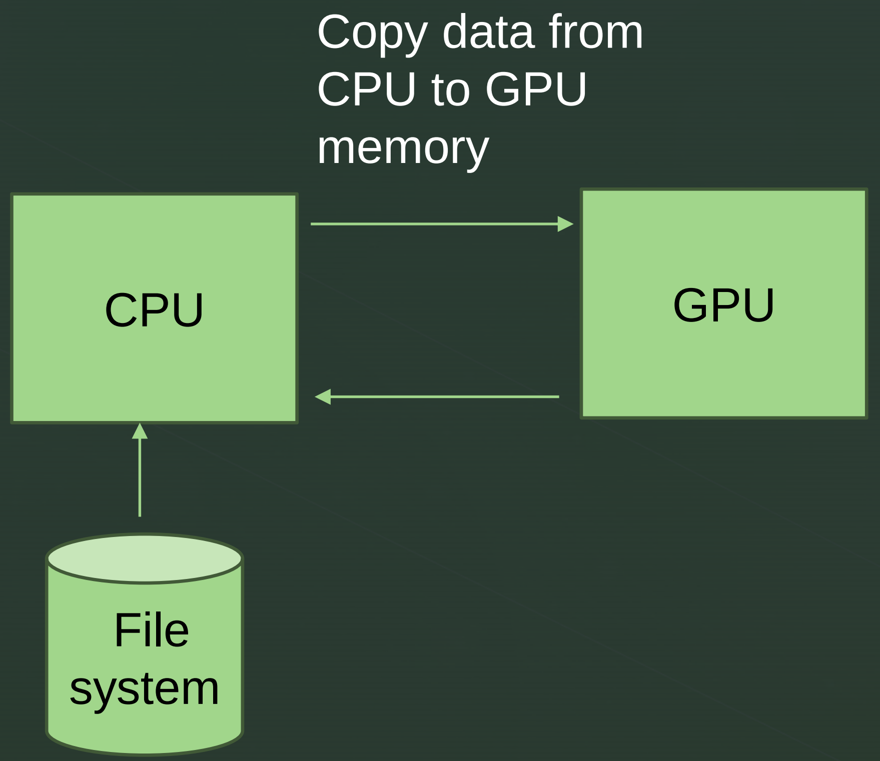
- The CPU and its corresponding discrete GPUs possess distinct physical memory (RAM).
- Direct access to a variable in CPU memory is not possible from a GPU .
- A programmer is responsible for maintaining copies of variables, and it is their duty to ensure that these copies remain in sync.

Programming using CUDA

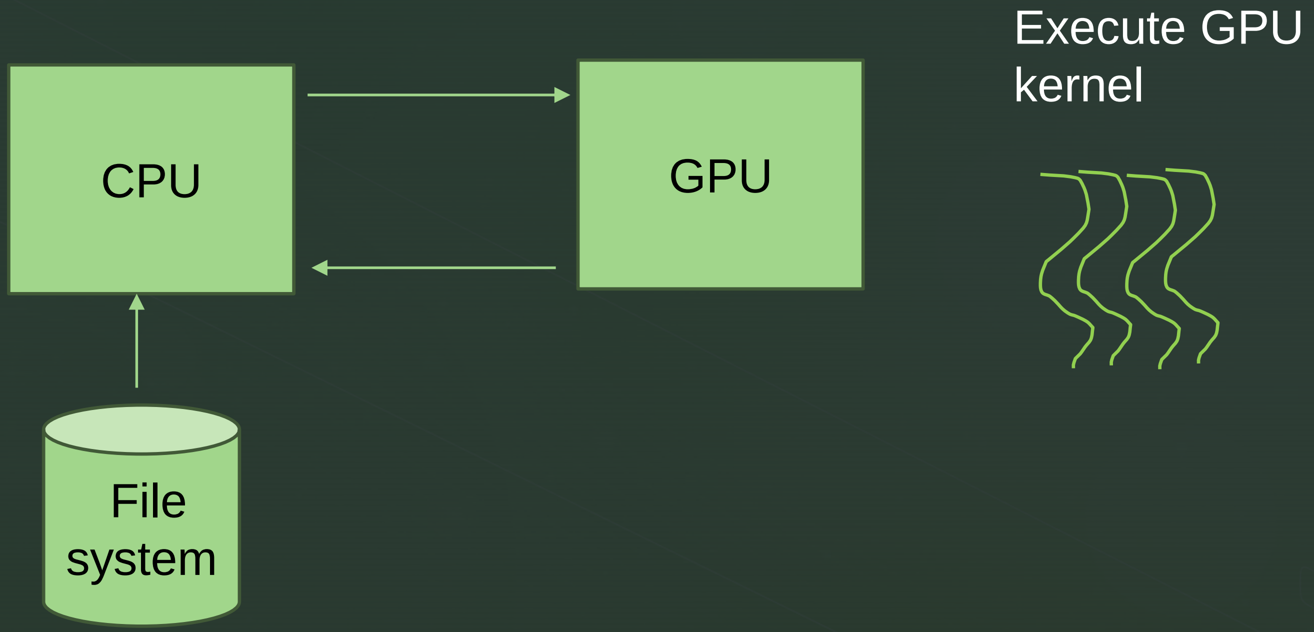


Load data into CPU memory

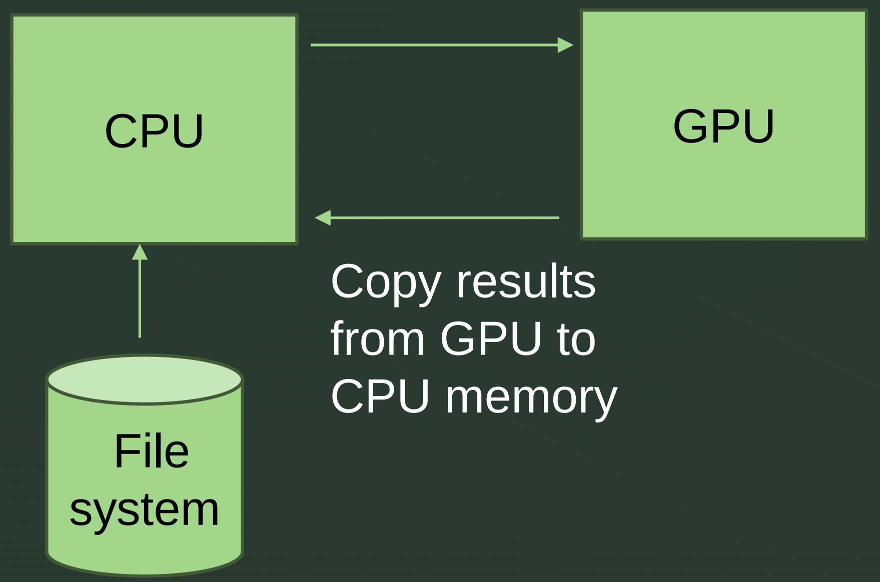
Programming using CUDA



Programming using CUDA



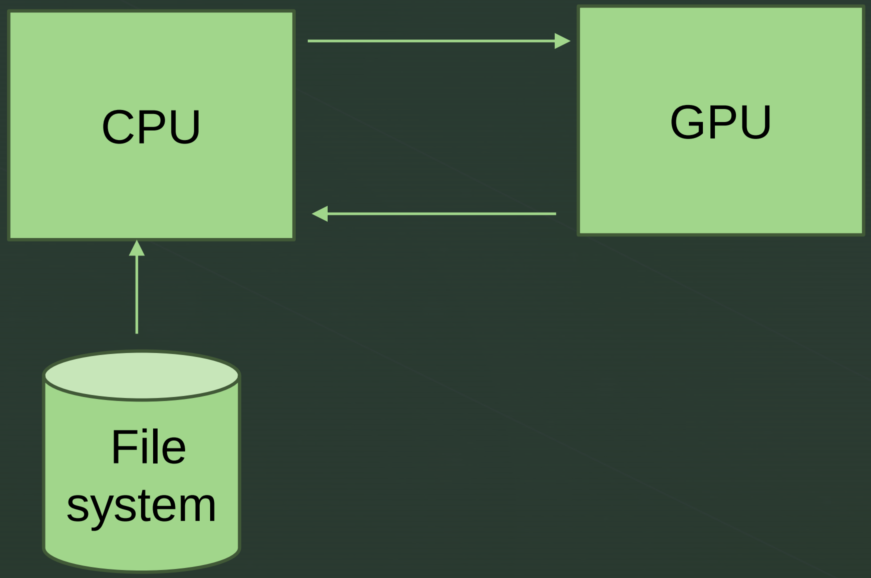
Programming using CUDA



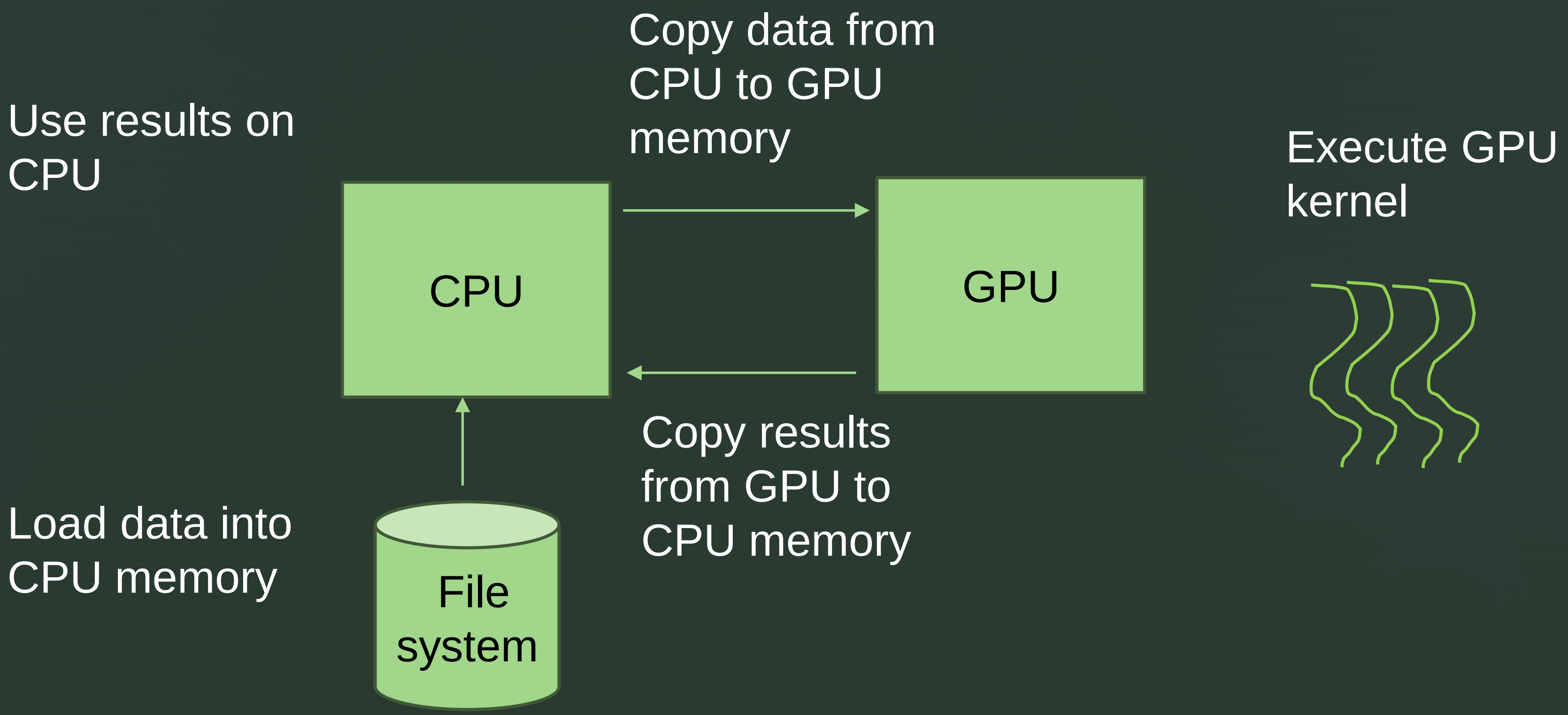
Copy results from GPU to CPU memory

Programming using CUDA

Use results on CPU



Programming using CUDA

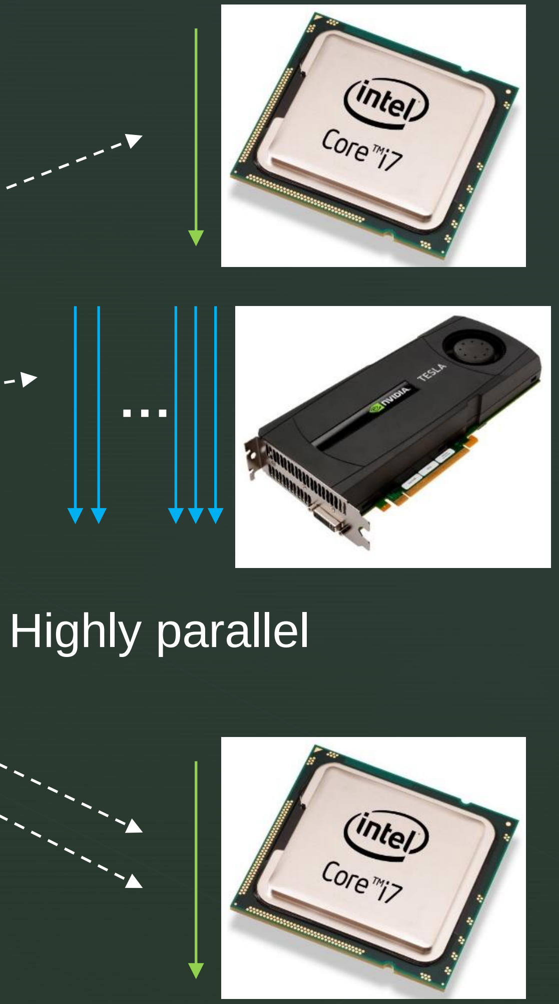

Use results on CPU
Copy data from CPU to GPU memory
Execute GPU kernel
CPU
GPU
Copy results from GPU to CPU memory
Load data into CPU memory
File system

Programming using CUDA

- Load data into CPU memory
- Copy data from CPU to GPU memory
- Call GPU kernel
- Copy results from GPU to CPU memory
- Use results on CPU

Programming using CUDA

- do_something_on_host();
- kernel<<<nBlk, nTid>>>(args);
- cudaDeviceSynchronize();
- do_something_else_on_host();



Thank You