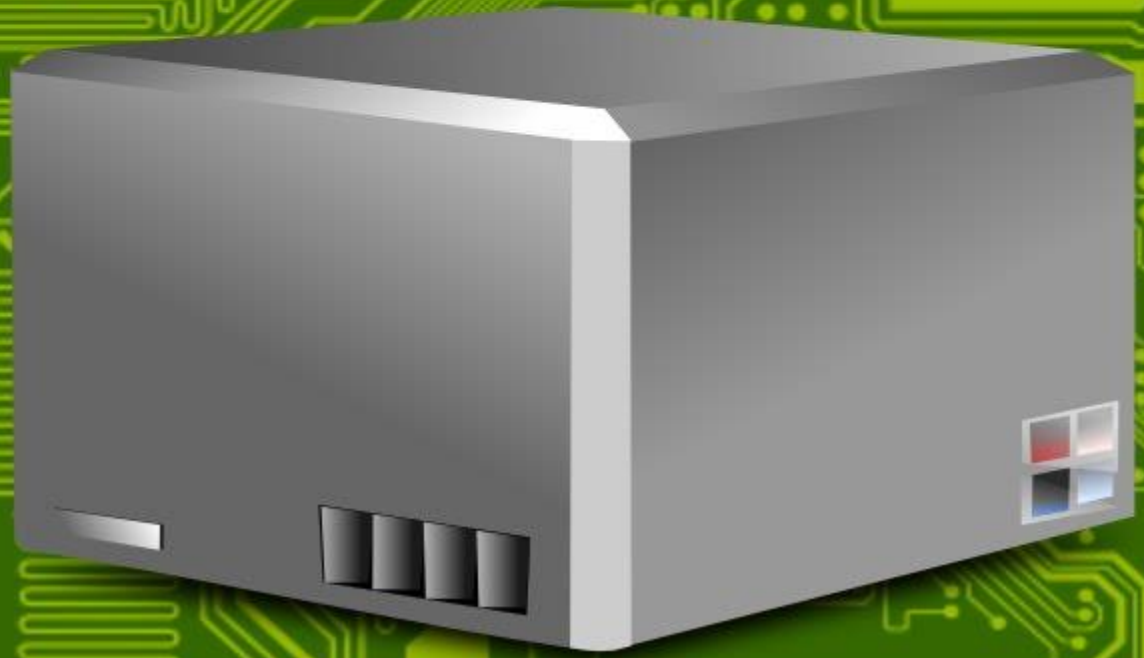
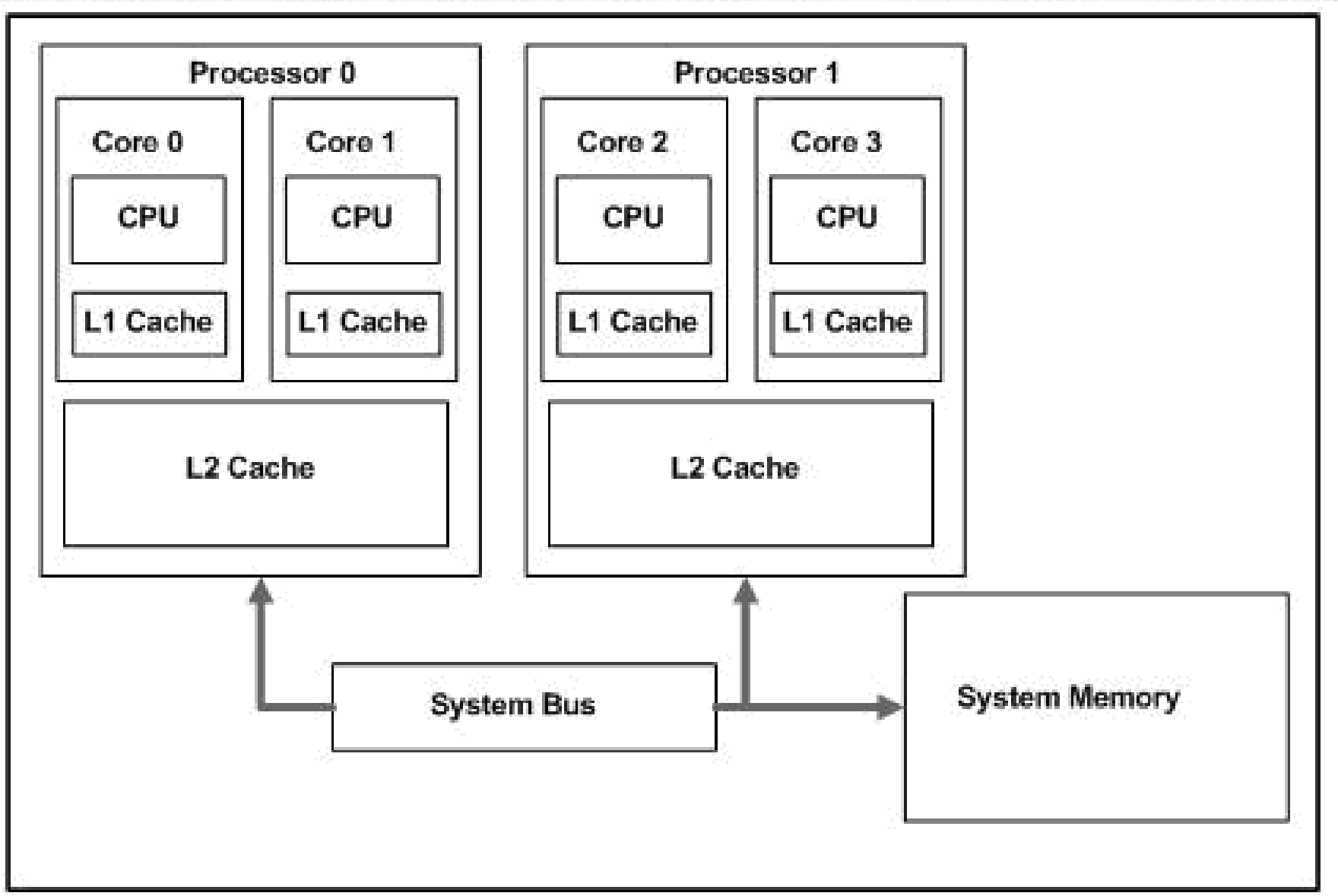
GPU Computing



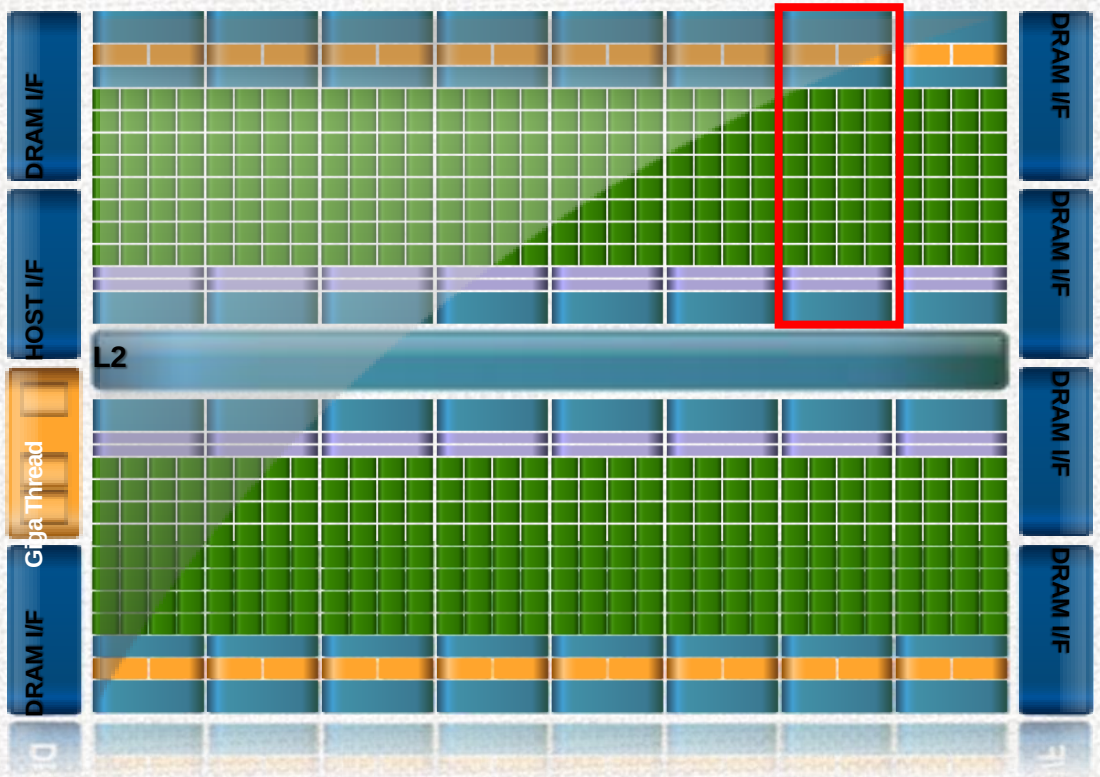
什么是 GPU 计算

Hui Liu
Email: hui.sc.liu@gmail.com

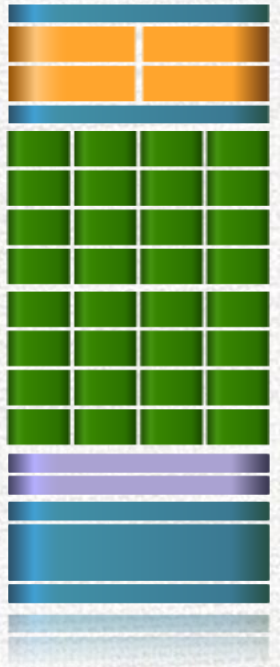
CPU 架构


Processor 0
Core 0
CPU
L1 Cache
Core 1
CPU
L1 Cache
L2 Cache
Processor 1
Core 2
CPU
L1 Cache
Core 3
CPU
L1 Cache
L2 Cache
System Bus
System Memory

GPU 架构 (C2050)

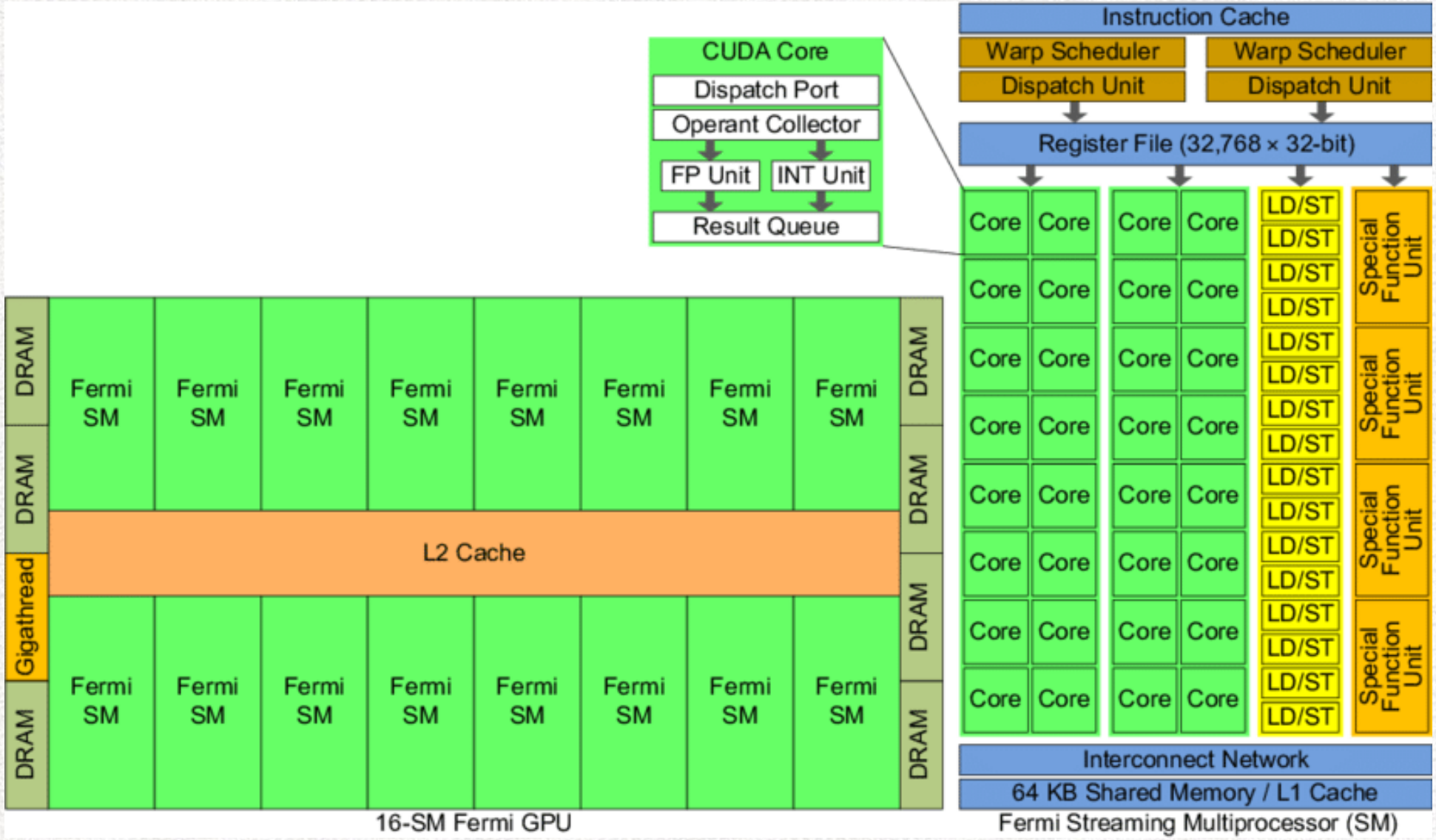


GPU



SM

GPU 架构 (Fermi)


CUDA Core
Dispatch Port
Operant Collector
FP Unit
INT Unit
Result Queue
Instruction Cache
Warp Scheduler
Warp Scheduler
Dispatch Unit
Dispatch Unit
Register File (32,768 × 32-bit)
Core
LD/ST
Special Function Unit
Interconnect Network
64 KB Shared Memory / L1 Cache
Fermi Streaming Multiprocessor (SM)
DRAM
Fermi SM
L2 Cache
Gigathread
16-SM Fermi GPU

GPU 架构


PCI Express 3.0 Host Interface
GigaThread Engine
Memory Controller
Memory Controller
Memory Controller
Memory Controller
Memory Controller
Memory Controller
GPC
Raster Engine
SMM
L2 Cache

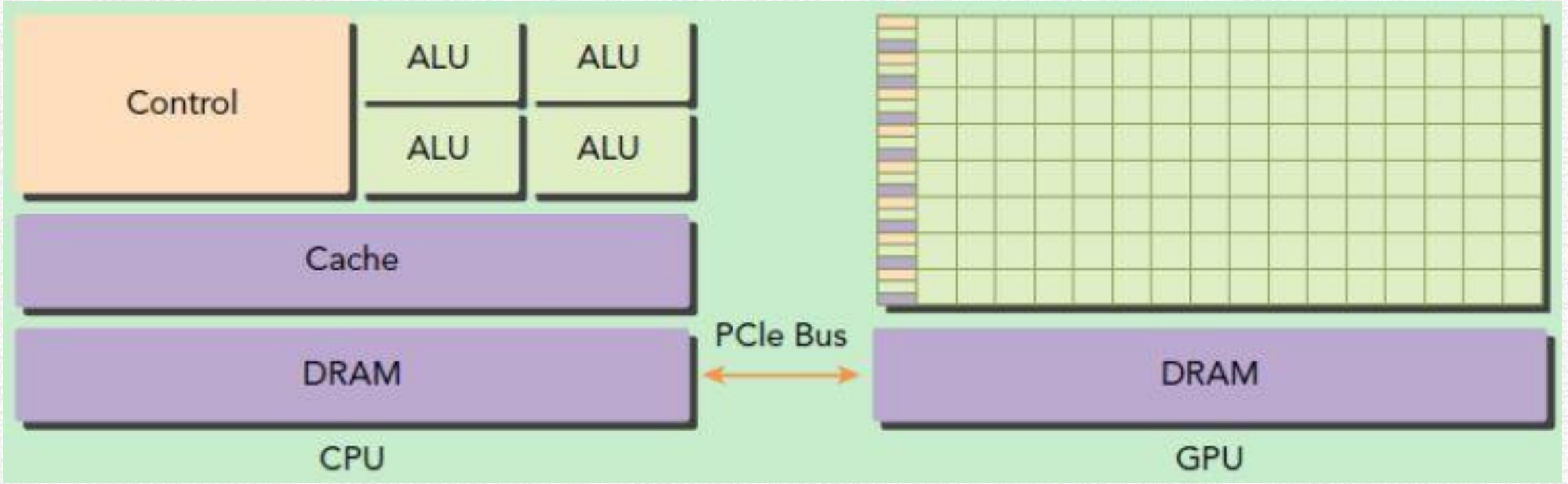
什么是GPU 计算

- NVIDIA公司发布了CUDA，C它是建立在NVIDIA的CPUs上的一个通用并行计算平台和编程模型，基于CUDA编程可以利用GPUs的并行计算引擎来更加高效地解决比较复杂的计算难题。
- GPU并不是一个独立运行的计算平台，而需要与CPU协同工作，可以看成是CPU的协处理器，因此当我们在说GPU并行计算时，其实是指的基于CPU+GPU的异构计算架构。
- 在异构计算架构中，GPU与CPU通过PCIe总线连接在一起来协同工作
- CPU所在位置称为为主机端 (host), 而GPU所在位置称为设备端 (device).

为什么使用 GPU 计算

- GPUs的并行计算引擎强大, 可以大幅度加快计算速度, 例如15倍左右
- 超级计算机使用加速器, 例如天河, Summit
- 机器学习以及人工智能需要训练模型, 需要大量的计算, 特别是稠密矩阵向量计算, GPU 可以快十倍以上
- GPU最成功的一个应用就是深度学习领域，基于GPU的并行计算已经成为训练深度学习模型的标配。

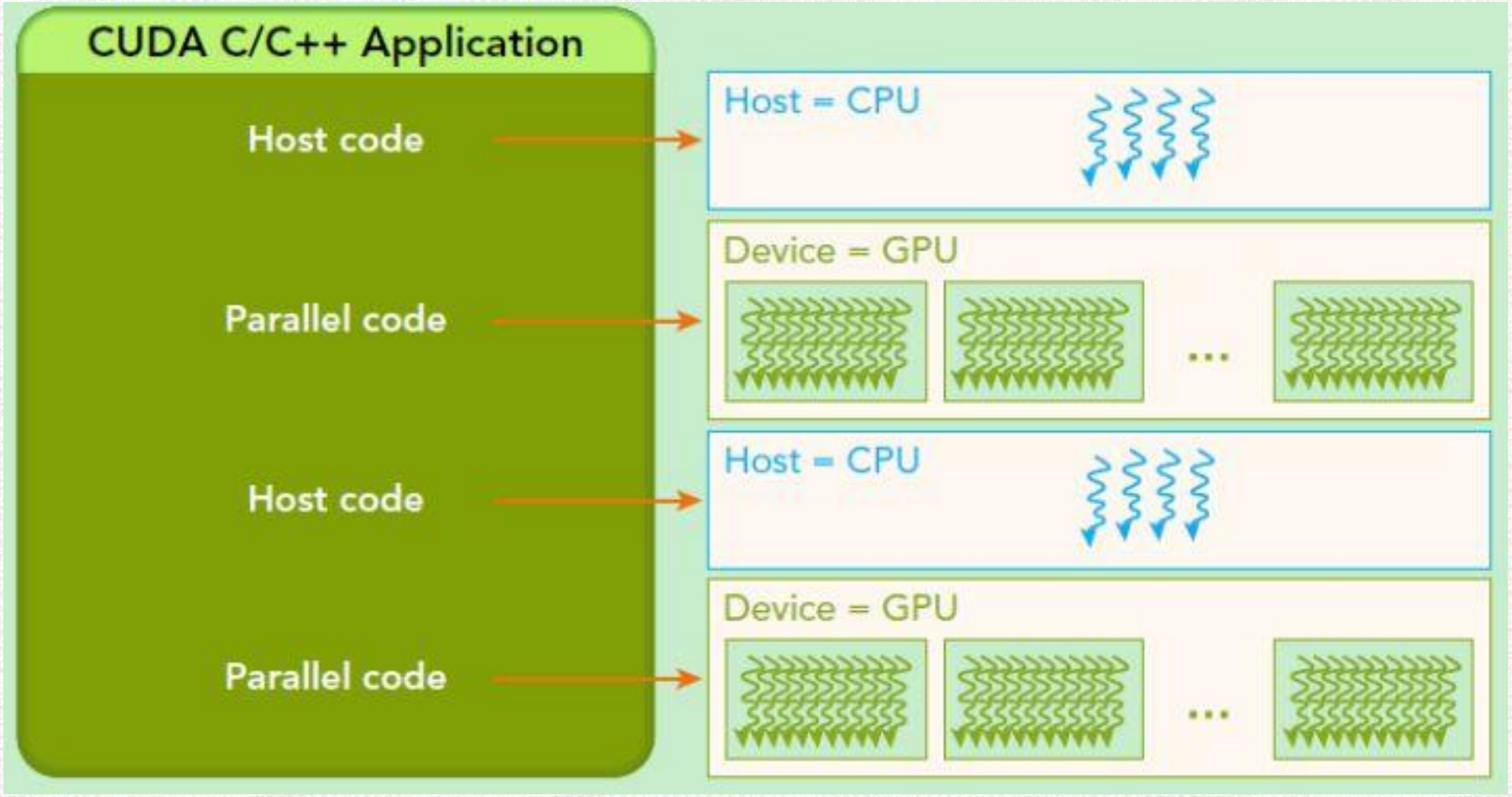
GPU 计算架构



CPU 与 GPU 分工与协作

- GPU包括更多的运算核心，其特别适合数据并行的计算密集型任务，如大型矩阵运算，
- CPU的运算核心较少，但是其可以实现复杂的逻辑运算，因此其适合控制密集型任务。
- CPU上的线程是重量级的，上下文切换开销大
- GPU由于存在很多核心，其线程是轻量级的
- 基于CPU+GPU的异构计算平台可以优势互补，CPU负责处理逻辑复杂的串行程序，而GPU重点处理数据密集型的并行计算程序，从而发挥最大功效。

程序架构


CUDA C/C++ Application
Host code
Parallel code
Host code
Parallel code
Host = CPU
Device = GPU
Host = CPU
Device = GPU

语言选取

- CUDA是NVIDIA公司所开发的GPU编程模型，它提供了GPU编程的简易接口，基于CUDA编程可以构建基于GPU计算的应用程序。

- CUDA提供了对其它编程语言的支持，如C/C++，Python，Fortran等语言，这里我们选择CUDA C/C++接口对CUDA编程进行讲解。

编译器

- CUDA: NVIDIA, latest CUDA v10, nvcc
- Windows, Mac OSX, Linux
- Linux: Fedora, Ubuntu, RHEL, CentOS
- 推荐 Linux :1) 容易写编译脚本, Makefile; 2) 很多命令行可以尝试; 3) 轻量级操作环境; 4) 免费

CUDA 工具

- 编译器: nvcc (C/C++)
- 调试器: nvcc-gdb
- 性能分析: nsight, nvprof
- 函数库: cublas, nvblas, cusolver, cufftw, cusparse, nvgraph

示例程序

- https://github.com/huiscliu/GPU-Computing-Intro
- Env: Linux, Mac OSX
- Windows: 时间函数需要删除或者替换

THANK YOU