GPU 在 SIMT 编程本质

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem

Talk Overview

1. **AI 计算体系**
   - 深度学习计算模式
   - 计算体系与矩阵运算
2. **AI 芯片基础**
   - 通用处理器 CPU
   - 通用图形处理器 GPU
   - AI专用处理器 NPU/TPU
3. **GPU详解**
   - 英伟达GPU架构发展
   - Tensor Core和NVLink
4. **国外 AI 芯片**
   - 特斯拉 DOJO 系列
   - 谷歌 TPU 系列
5. **国内 AI 芯片**
   - 壁仞科技芯片架构
   - 寒武纪科技芯片架构
6. **AI芯片的思考**
   - SIMD&SIMT与编程体系
   - AI芯片的架构思路与思考

Talk Overview

I. GPU 在 SIMT 编程本质

- SIMD vs SIMT 回顾
- 编程模型 vs 执行模型
- 并行的编程方式
- 英伟达 GPU 编程模型

4. GPU的编程模型

SIMD vs. SIMT 执行模式

- **SIMD**：单顺序的指令流执行 -> 每条指令多个数据输入同时执行
  - [VLD, VLD, VADD, VST], VLEN
- **SIMT**：标量指令的多个指令流 -> 动态地把线程按wrap分组执行
  - [LD, LD, ADD, ST], NumThreads

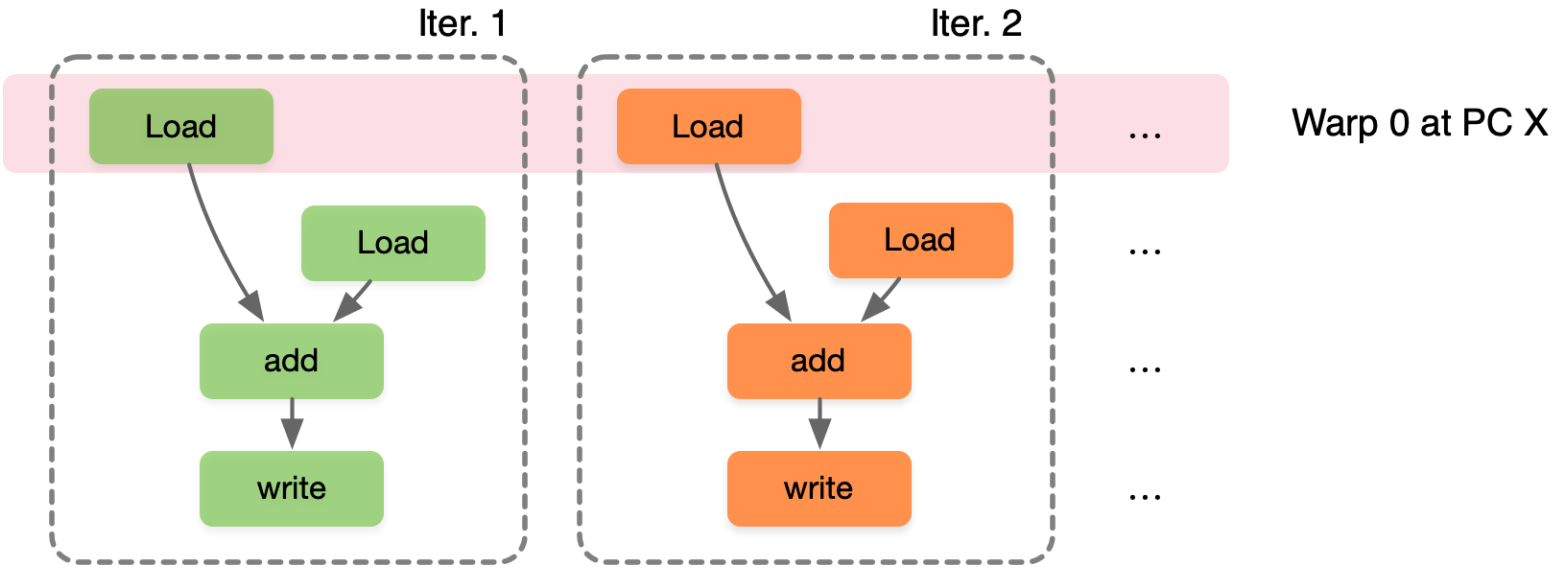
SIMD vs. SIMT 执行模式

- **SIMD**：单顺序的指令流执行 -> 每条指令多个数据输入同时执行
  - [VLD, VLD, VADD, VST], VLEN
- **SIMT**：标量指令的多个指令流 -> 动态地把线程按wrap分组执行
  - [LD, LD, ADD, ST], NumThreads

- **SIMT的优势：**
  - 无需开发者费力把数据凑成合适的矢量长度
  - 从硬件设计上解决大部分 SIMD data path 的流水编排问题
  - 线程可以独立执行，使得每个 thread 相对灵活，允许每个线程有不同的分支
  - 一组执行相同指令的线程由硬件动态组织成 warp，加快了 SIMD 的计算并行度

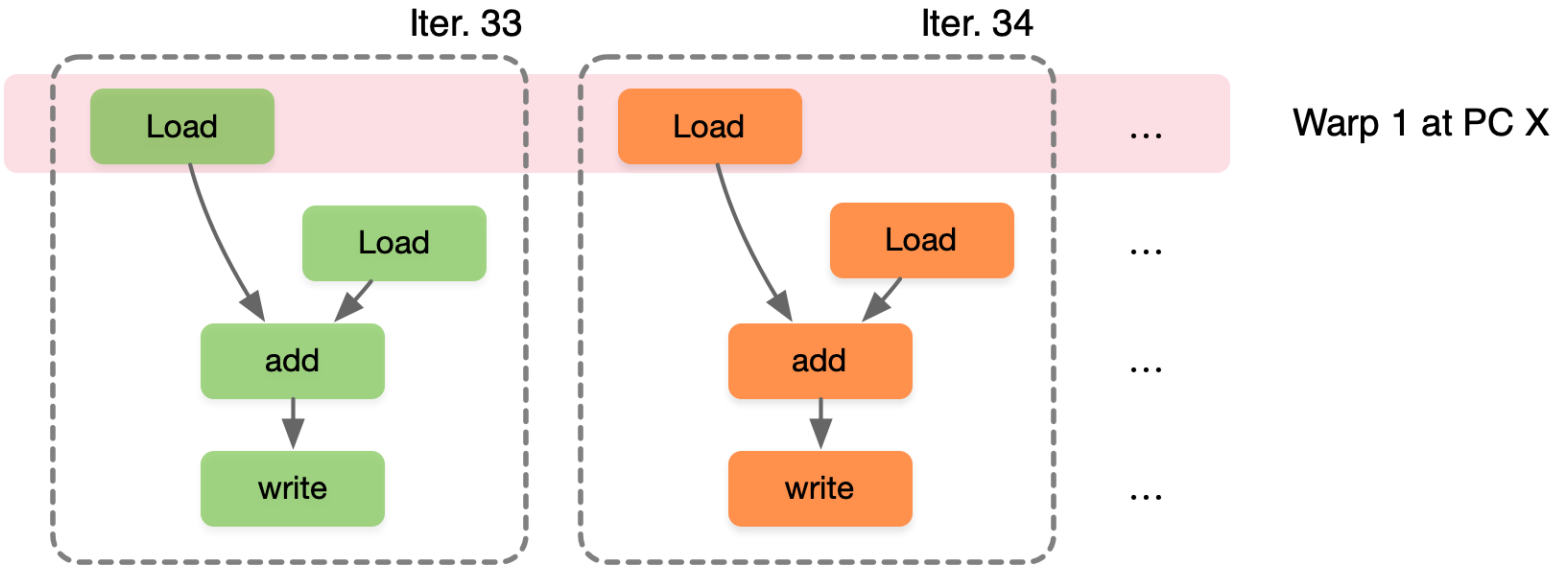
Multithreading of Warps

- GPU中线程调度的基本单位是Warp
- 假设一个 warp 包含32个线程
- 现在要进行 32K 次迭代，每个迭代一个线程 -> 1K wrap



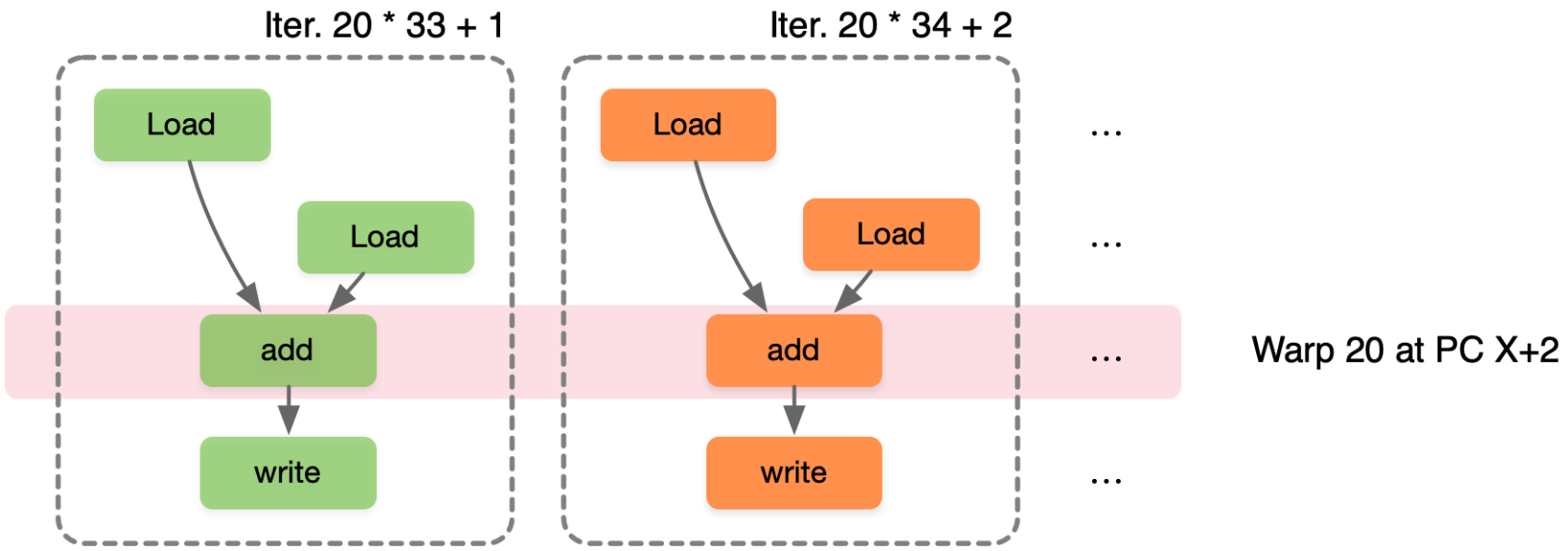
Multithreading of Warps

- GPU中线程调度的基本单位是Warp
- 假设一个 warp 包含32个线程
- 现在要进行 32K 次迭代，每个迭代一个线程 -> 1K wrap



Multithreading of Warps

- GPU中线程调度的基本单位是Warp
- 假设一个 warp 包含32个线程
- 现在要进行 32K 次迭代，每个迭代一个线程 -> 1K wrap



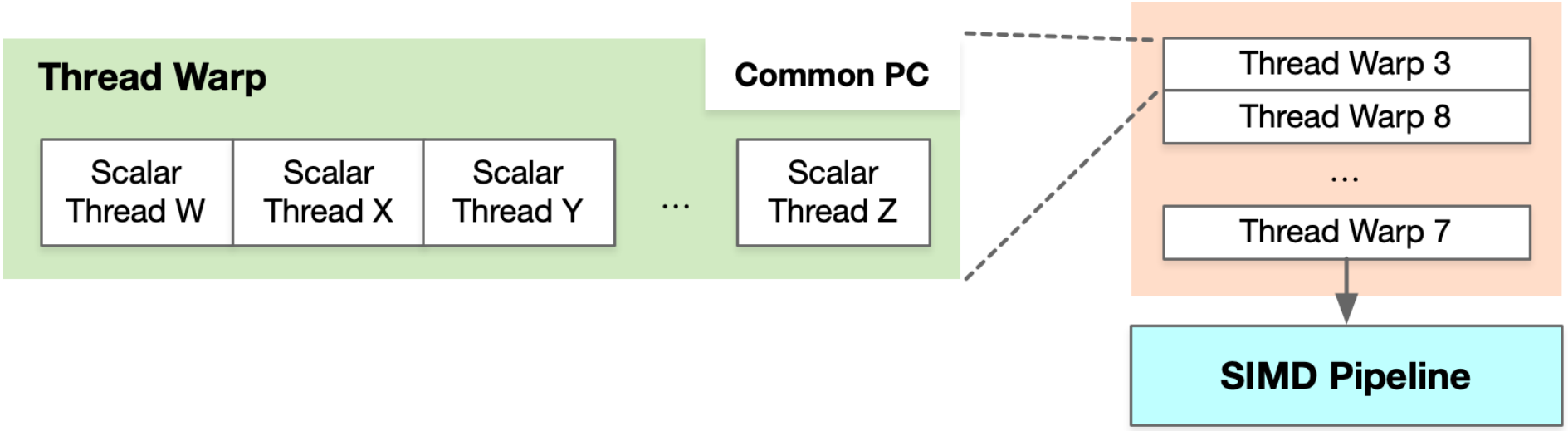
解决并行的瓶颈

程序并行执行最大的瓶颈是访存和控制流：

- SIMD 架构中单线程 CPU 通过大量控制逻辑进行超前执行、缓存、预取等机制来强行缓解瓶颈。
- SIMT 架构通过细粒度的多线程（FGMT）调度来实现访存和计算并行。

Warps 和 Warp-Level FGMT 关系

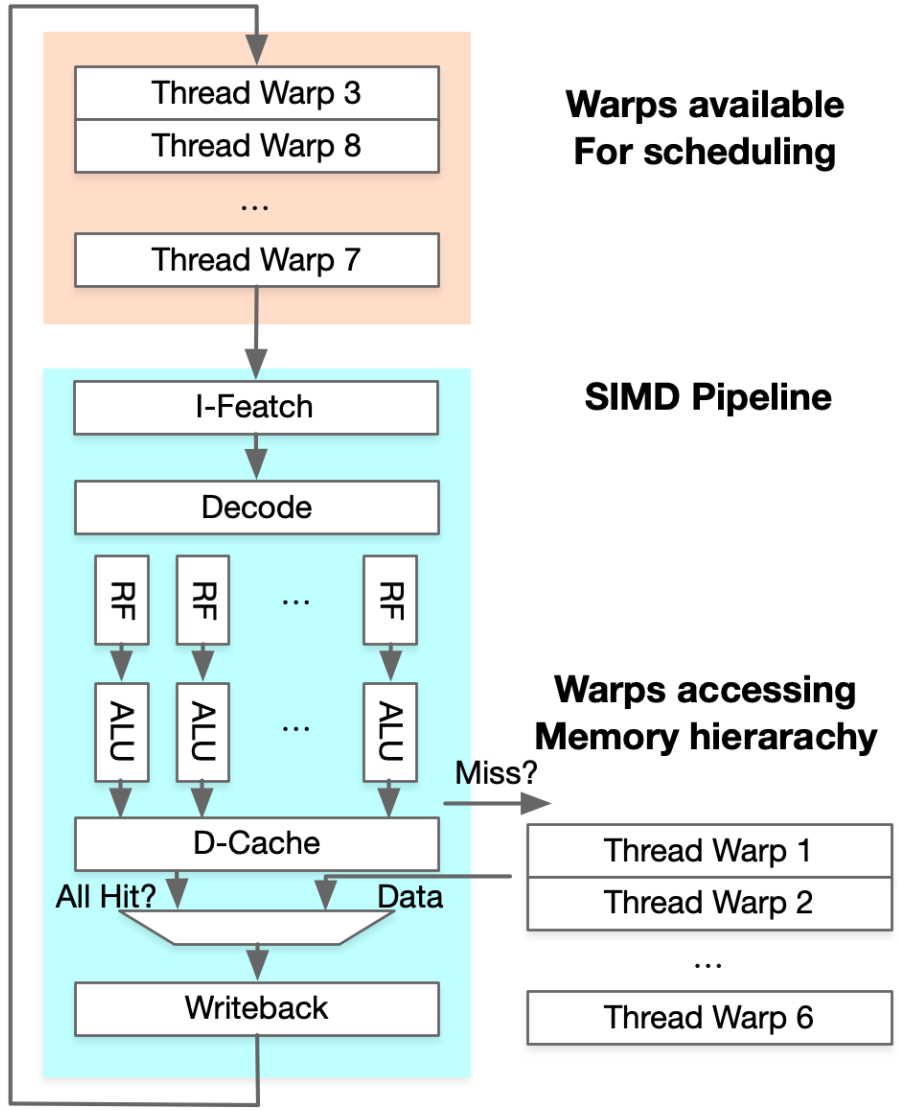
- Warp: 在不同地址数据下，执行相同指令的线程集合
- 所有线程执行相同的代码



通过 Wrap-level FGMT 隐藏延迟

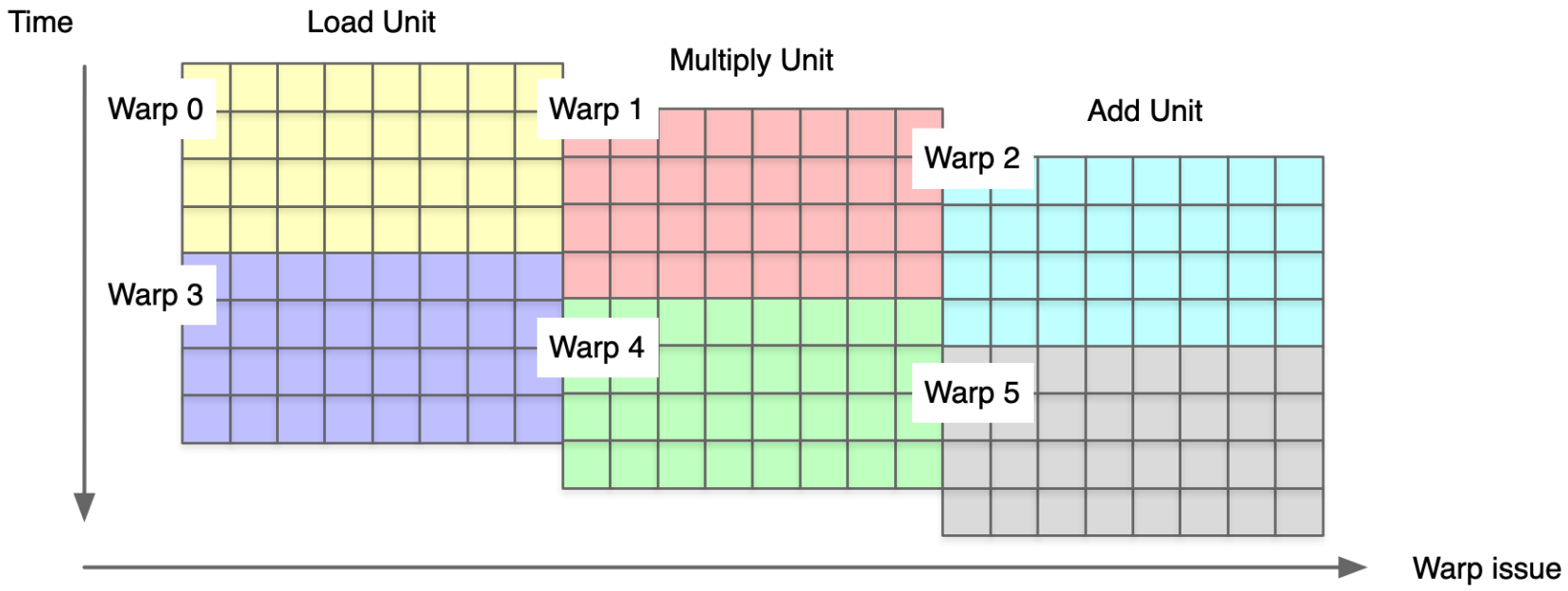
- SIMT 架构通过细粒度多线程（FGMT）实现：
  - Pipeline 中中每个线程一次一条指令
  - Warp 乱序执行以隐藏访存延迟
  - 线程寄存器值都保留在 RF 中
  - FGMT 允许长延迟

- **先访存完毕 Warp 去执行尽可能多的指令，隐藏其它 Warp 的访存时间。**

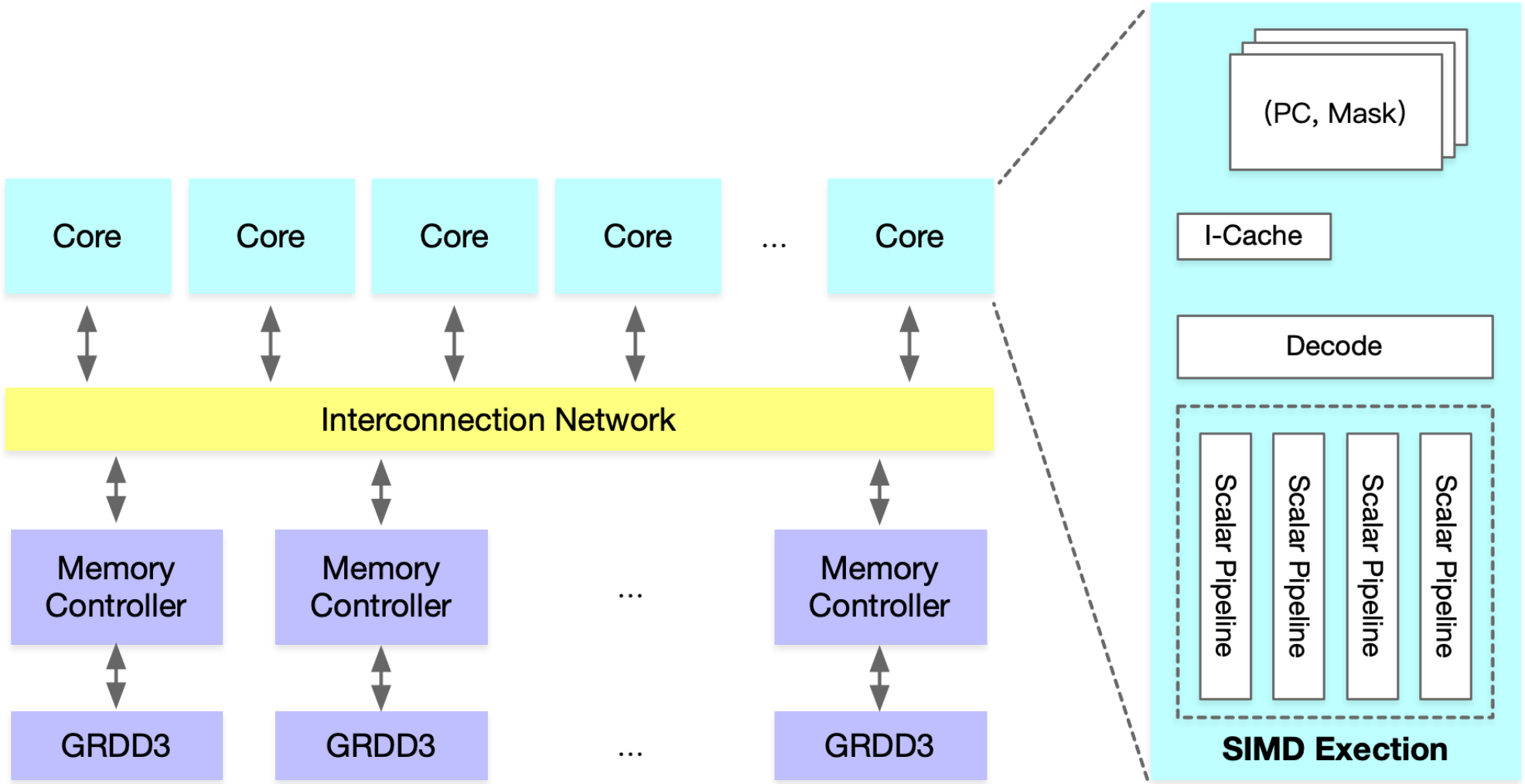


Warp 指令级并行

- SIMT 相比 SIMD 在可编程性上最根本性的优势在于硬件解决了大部分流水编排的问题：
  - 示例中每个 warp 有 32 个线程和 8 条执行通道
  - 完成 24 Operations/cycle，同时发出1 Warp/cycle



High-level view of GPU



5. 总结

执行模型： Traditional SIMD vs Warp-base SIMD(SIMT)

Traditional SIMD

- 包含单条指令执行
- ISA包含矢量/SMD指令信息
- SIMD指令中的锁同步操作，即顺序指令执行
- 编程模型是直接控制指令，没有额外线程控制，软件层面需要知道数据长度

Warp-base SIMD (SIMT)

- 以 SIMD 方式执行的多个标量线程组成
- ISA是标量，SIMD 操作可以动态形成
- 每条线程都可以单独处理，启用多线程和灵活的线程动态分组
- 本质上，是在 SIMD 硬件上实现 SPMD 编程模型

编程模型：SPMD

- **通过单个程序，控制多路数据**
- **针对不同的数据，单个线程执行相同的过程代码**
- **本质上，多个指令流执行同一个程序**
  - 每个程序：1）处理不同数据，2）在运行时可以执行不同的控制流路径
  - 在 SIMD 硬件上以 SPMD 的方式对 GPGPU 进行编程控制 -> CUDA 编程

思考

- AMD 的显卡也是有大量的计算单元和计算核心，为什么没有 SIMT 的编程模式？
- NVIDA 推出 CUDA 并遵循自定义的 SIMT 架构做对了什么？

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem