Data-Level Parallelism

Flynn's Taxonomy

- SISD
- SIMD
- MISD
- MIMD

DLP 即对不同数据执行同样操作（区别于 data flow，并行执行不同操作；区别于 TLP，并行执行不同线程）

SIMD：同样的指令在操作不同的数据，可分为两种

- Array Processor：指令在同一时间不同空间对多个数据操作
- Vector Processor：指令在同一空间连续时间对多个数据操作

Vector Processor

Vector Processor

vector 是一个 1 维的数据数组

vector processor 即对向量而非标量处理的机器

需要实现以下基本操作

- 向量的存取：vector register
- 需要对不同长度的向量操作：VLEN (Vector Length Register)
- 需要控制元素间的步长：VSTR (Vector Stride Register)

一个向量指令连续处理各个元素：vector function unit 是流水线，每个 stage 处理一个数据

流水线可以做得很深，且设计简单

- 没有内部依赖，不需要 bypassing
- 没有控制流
- 知道步长可以预取

还有一个 VMASK 用于控制仅对 vector 中某几个元素操作的情况

Tradeoff

优点

- vector 内部无依赖，可以设计很好的流水线
- 每个指令可以完成许多工作，减少取指的带宽需求
- 内存访问模式固定，可以在不同 bank 存数据，最大程度利用带宽
- 不需要显式的循环，branch 更少

缺点

- 只适用于规则的并行
- 如果并行不规则，效率很低
- 内存带宽会成为瓶颈

Load/Store Vectors

如何实现一个 cycle 一个元素，如果访存的时间超过一个周期？Memory Banking

将内存分为 bank，每个 bank 可以独立访问，共享地址线和数据线，然后将数据间隔存在 bank 中

vector bypassing：当后一个指令需要前一个指令的结果时，在计算出第一个分量后后面的指令即可开始执行

vector strip-mining：如果数据数大于 vector register 的大小，将循环拆分，一个循环处理一个 vector 的数据，如果有余数需要单独处理

Array vs. Vector

现代很多 SIMD 都是 array 和 vector 的组合，如向量相加操作，可以将向量划分为多个 lane 同时处理（array），处理每个子向量时按照 vector processor 的方式

如一个 32 个元素，8 lane 的机器，做 $C = kA + b$，则可以先读取前 8 个元素，然后对这 8 个元素做乘法时再读 8 个，达到每个 cycle 只 issue 一条指令却可以执行 24 次操作的效果：8 次加载，8 次乘法，8 次加法

需要编译器将代码向量化

Conditional Execution

如果循环中添加了对元素的判断，则需要 vector mask 来控制对哪些元素操作

简单的实现方式是全部执行，对没有 mask 的元素不写回

效率高的方法是只执行 mask 了的部分

Vector Reductions

因为有 reduction variable（e. g. 对向量分量求和），产生了依赖

解决方法：重新组织代码，按照二叉树方法求和（划分-部分和-求和部分和）

Vector Scatter/Gather

如果向量操作有间接访问 `A[i] = B[i] + C[D[i]]`

解决方法：先将数据全部读取再进行计算

Summary

vector machine 适合规则的 DLP

性能的提升受限于代码的可向量化程度，如果全是标量操作，效果会很差

Amdahl's Law

SIMD

一个指令**同时**对多个数据操作，类似 array processing 但是没有 VLR，且步长固定为 1

最初设计是为了多媒体操作

SIMD 使用现有的 64-bit 寄存器，将其划分为 2 个 32-bit 或是 4 个 16-bit 或者 8 个 8-bit 分别执行操作

GPU

Graphics Processing Unit

- 最早是使用高效浮点运算单元来实现 3D 图像的生成，用户可以配置流水线，但是不能真正的编程
- 后来增加了编程能力，可以为每个 vertex 或是 pixel 编写程序，并行度高但是编程模型有限
- GPGPU (General-Purpose GPU)：使用 GPU 实现通用计算，只需要将输入输出映射为图像

GPGPU

Nvidia 发布的新语言 cuda (compute unified device architecture)，以及 OpenCL

基本思想是利用 GPU 的高性能和高带宽加速通用计算，由 CPU 控制，GPU 计算

SIMD engines underneath

SPMD (Single Program Multiple Data)：指令流水线类似 SIMD 流水线，但是使用多线程编程而不是 SIMD 指令

区分编程模型和执行模型

- 编程模型是程序员表达程序的形式，如串行（冯诺依曼），数据并行（SIMD），数据流，多线程等
- 执行模型是底层硬件执行的形式，如乱序执行，向量机，多处理器等

SPMD 是编程模型，每个 PE (Processing Element) 执行一样的操作，但是数据不同，可以使用 barrier 来进行同步

即同时有多个执行流在执行同一个程序，操作不同数据，可以有不同的控制流

程序可以在多处理器上以 MIMD 形式实现或是在 GPU 上以 SIMD 形式实现

GPU Execution Model

GPU 是 SIMD 的机器，但是不是用 SIMD 指令编程，而是使用 SPMD 模型，每个线程执行相同的代码，操作不同的数据，每个线程有各自的上下文，可以独立地操作

GPU 的执行模型是 SIMT (Single Instruction Multiple Threads)

一系列执行相同指令的线程被划分为 warp，一个 warp 本质上是硬件的 SIMD 操作

一个 warp 中有多个标量线程，但其执行进度一致（warp 内共享一个 pc）

> SIMD vs. SIMT
>
> SIMD 是一个指令流，由 SIMD 指令组成（VLD, VST, VADD...），每个指令接受多个输入
>
> SIMT 是多个指令流，由标量指令组成（LD, ST, ADD...），指令流（即线程）组织为 warp
>
> SIMT 主要优点是
>
> - 可以分别处理每个线程，执行可以在任意标量的流水线
> - 可以灵活地将线程组织为 warp

如 `c[i] = a[i] + b[i]`，开始时 warp 在读 `a[i]`，然后是读 `b[i]`，然后是相加和存储

GPU Hardware Structure

GPU 由多个并行的核组成，每个核有一个 SIMD 的处理器，多条 lane

CPU 将 grid 发送到 GPU，将线程分发到各个核

不同 warp 可以按照细粒度多线程的方式组织在同一个流水线，即每个流水线 stage 运行不同的 warp，通过这种方式掩盖指令的延迟（warp 内部进度一致，但是不同 warp 有各自的 PC，可以运行不同指令，产生不同控制流）

所有线程的寄存器数据都在寄存器堆中，需要大量寄存器，动态分配

不同线程的同一指令访存时使用线程 ID 索引

Warp-based SIMD vs. Traditional SIMD

传统的 SIMD

- 单线程，下一条向量指令必须等待前一条完成
- 按照 SIMD 编程，软件需要知道向量长度等硬件信息
- ISA 需要有 SIMD 指令

Warp-based

- 多个标量线程，按照 SIMD 形式执行（多个线程共享 PC，执行同一条指令）
- 每个线程可以被单独处理（如分配到另一个 warp），编程模型不是 SIMD，而是普通的多线程编程
- ISA 有标量指令即可
- 本质是 SIMD 机器上实现了 SPMD

Branch Divergence

在 warp-based SIMD 中，线程可以执行不同的路径，而 GPU 是 SIMD 流水线

当 warp 中线程出现了不同的执行路径，则发生 branch divergence

处理可以是动态地条件执行，使用一个栈（参见第 9 周 ppt）

当执行完不同路径后再 merge，如果有大量 warp 都分歧，则可以组织路径相同的等待中的 thread 为一个新 warp

Memory Access

memory divergence：在一个 warp 内的访存可以有非常复杂的形式（由于重新组织了 warp）

现代的 GPU 都增加了 cache，用于减少与主存的通信

- 然而同一 warp 内可能有线程 hit 了有线程 miss 了
- 一个线程 miss 了会卡住 warp 内所有线程

需要有策略同时处理 branch 和 memory 的 divergence

Acceleration for ML

机器学习（尤其是深度学习）带来了新的要求

- 典型的网络：MLP/CNN/RNN
- 典型的计算任务：矩阵乘法，非线性的激活函数

加速方法有多种

- GPU：实现通用计算
- FPGA：灵活，可编程适应不同任务，高并发
- ASIC：嵌入式解决方案，如 TPU, DianNao 等