Lecture 14 从客户端到云的并行处理器

1. 引言

概念

- **多处理器 multiprocessor**
  至少含有两个处理器的计算机系统
- **单处理器 uniprocessor**
  只含有一个处理器的计算机系统
- **集群 cluster**
  通过局域网连接的一组计算机，其作用等同于一台大型多处理器计算机
- **任务级并行 task-level parallelism / 进程级并行 process-level parallelism**
  通过同时运行独立程序的方法来利用多处理器，这些程序是独立的单线程应用程序
- **并行处理程序 parallel processing program**
  同时运行在多个处理器上的单一程序
- **多核微处理器 multicore microprocessors**
  在单一集成电路上包含多个处理器（“核”）的微处理器，目前基本上所有台式机和服务器都是多核微处理器
- **共享内存处理器 shared memory processor（SMP）**
  共享同一地址空间的并行处理器

硬件及软件的并行

硬件

- 串行 Serial
  单核处理器
    - Intel Pentium4
- 并行 Parallel
  多核处理器
    - Intel Core i7

软件

- 顺序 Sequential

        - 矩阵乘法
- 并发 Concurrent
        - 操作系统

软件和硬件的串行和并行可以两两组合，构成四种情况

		软件	
		顺序	并发
硬件	串行	在 Intel Pentium4 上运行的使用 Matlab编写的矩阵乘法	在 Intel Pentium4 上运行的 Windows Vista 操作系统
	并行	在 Intel Core i7 上运行的使用 Matlab 编写的矩阵乘法	在 Intel Core i7 上运行的 Windows Vista 操作系统

2. 创建并行处理程序的困难

困难

并行程序的困难有

- 调度：将任务分割成可并行的部分
- 协调：负载均衡，同步时间
- 通信：程序之间的相互交流

Amdahl 定律

例题1 加速比的困难

如果希望在一个 100 个处理器上获得加速比 90，原始计算中能有多少是 sequential 的呢？

在 1 个处理器的情况下 $T_{old} = F_{sequential}+F_{paralizable}$

在 100 个处理器的情况下 $T_{new} = F_{sequential}+\frac{F_{paralizable}}{100}$

如果希望加速比 90，即

可以得到 $\frac{F_{paralizable}}{F_{sequential}}=890$，即 $F_{paralizable}$ 的占比高于 0.999

例题2 更大规模的问题

一个程序执行两个加法，一个是对 10 个标量求和，另一个是对一个 10 × 10 的二维矩阵求和，假设只有矩阵求和可以并行化

- 使用 10 个和 40 个处理器达到的加速比是多少？
- 如果矩阵维数是 20 × 20 呢？

使用 10 个和 40 个处理器的情况

- 单处理器下 $T = (10 + 100) × t_{add} =110t_{add}$
- 10 个处理器下 $T = (10 + \frac{100}{10}) × t_{add}=20t_{add}$
- 40 个处理器下 $T = (10 + \frac{100}{40}) × t_{add}=12.5t_{add}$

加速比为

- $\frac{110}{20}=5.5$
    - 潜在加速比为 $\frac{5.5}{10}=55\%$
- $\frac{110}{12.5}=8.8$
    - 潜在加速比为 $\frac{8.8}{40}=22\%$

如果矩阵维数是 20 × 20，则需要进行 400 次加法运算

- 单处理器下 $T = (10 + 400) × t_{add} =410t_{add}$
- 10 个处理器下 $T = (10 + \frac{400}{10}) × t_{add}=50t_{add}$
- 40 个处理器下 $T = (10 + \frac{400}{40}) × t_{add}=20t_{add}$

加速比为

- $\frac{410}{50}=8.2$
    - 潜在加速比为 $\frac{8.2}{10}=82\%$
- $\frac{410}{20}=20.5$
    - 潜在加速比为 $\frac{20.5}{40}=51\%$

可以看到，提高问题规模的时候，潜在加速比变得更大了

	10 × 10	**20 × 20**
10 core	5.5 (55% of potential)	8.2 (82% of potential)
40 core	8.8 (22% of potential)	20.5 (51% of potential)

我们引入两个术语

- **强比例缩放 strong scaling**
  在多处理器上不需要增加问题规模即可以获得加速比
- **弱比例缩放 weak scaling**
  在多处理器上增加处理器数量同时按比例增加问题规模所能获得的加速比

例题3 负载均衡的问题

继续例题2 假设 20 × 20 的矩阵情况下，负载不均衡，存在一个处理器计算矩阵乘法中的 20 个 add，而其它的 39 个处理器计算剩下的 380 个 add，计算此时的加速比

则此时的加速比为 $\frac{410}{30}=14<20.5$

- 潜在加速比为 $\frac{14}{40}=35\%$

3. SISD、MIMD、SIMD、SPMD 和向量机

概念

		数据流	
		单	多
指令流	单	SISD：Intel Pentium 4	SIMD：x86 的 SSE 指令
	多	MISD：至今没有实例	MIMD：Intel Core i7

- **SISD Single Instruction, Single Data**
  单指令流单数据流处理器
  - Intel Pentium4
- **SIMD Single Instruction, Multiple Data**
  单指令流多数据流处理器
  - x86 的 SSE 指令
- **MIMD Multiple instruction, Multiple Data**
  多指令流多数据流处理器
  - Intel Core i7
- **MISD Multiple instruction, Single Data**
  多指令流单数据处理器
  - 目前还不存在

SPMD

在 MIMD 计算机上可编写独立的程序并运行在不同的处理器上，而且这些程序可以协同完成一个共同的大型目标，但是编程人员通常只编写单一程序，将其运行在 MIMD 计算机的所有处理器上，然后使用条件控制语句使不同的处理器执行不同的代码段，它被称为：**SPMD Simple Program Multiple Data 单程序多数据**，它是 MIMD 计算机编程的常见方式

SIMD

SIMD 的优点是对所有的并行执行单元同步，所有的处理器同时执行相同的程序，它们对都来自同一程序计数器 PC 的同一指令做出相应

它的优点为

- 简化同步
- 简化指令控制硬件
    - 降低指令宽度和空间，只需要执行一个代码副本
- 适用于高度**数据并行 data-level parallelism** 的应用程序
    - 使用 for 循环处理数组等具有大量相同结构的数据

向量机

向量体系结构的基本理念是

1. 从存储器中收集数据单元
2. 将它们按顺序放到**一大组寄存器**中
3. 在寄存器中使用**流水化**的执行单元对它们以此操作
4. 最后将结果写回寄存器

向量体系结构的关键特征是拥有一组向量寄存器

在向量处理器中，流水线阻塞在每次向量操作中**只会发生一次，不是每次对向量数据元进行操作时都会发生一次**，由于向量元素是相互独立的，它们可以**并行执行**

在 MIPS 中，有

```
lv,sv # 加载/存储向量
addv.d # 双精度向量加和
addvs.d # 把标量加到双精度向量的每个元素上
```

4. 硬件多线程

线程和进程

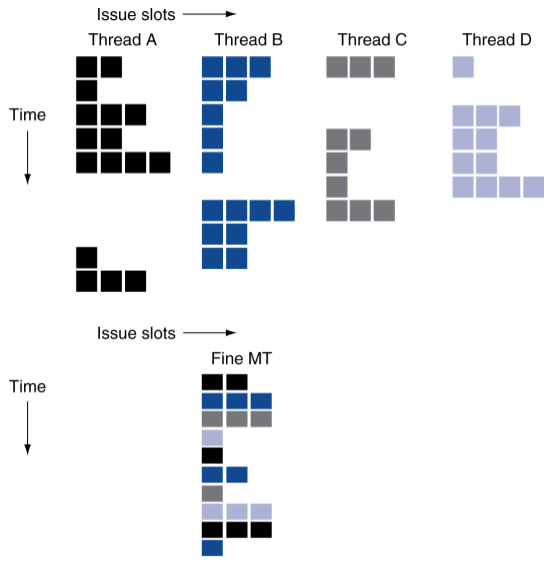
硬件多线程 hardware multithreading 就是一个和 MIMD 相关的概念，在线程阻塞的时候处理器可以切换到另一个线程实现

- 进程 **process**
    - 一个进程包含一个或多个线程、地址空间和操作系统
    - 一次进程切换通常需要操作系统的介入
- 线程 **thread**
    - 一个线程包含程序计数器、寄存器状态和内存栈
    - 它是一个轻量级的进程
    - 多个线程通常共享一个地址空间，而进程则不是
    - 线程的切换不需要操作系统的接入

MIMD 依靠多个进程 process 或线程 thread 来努力使多个处理器处于忙碌状态，而硬件多线程允许多个线程以重叠的方式共享一个处理器的功能单元

硬件多线程的实现

细粒度多线程 Fine-grain multithreading



特点

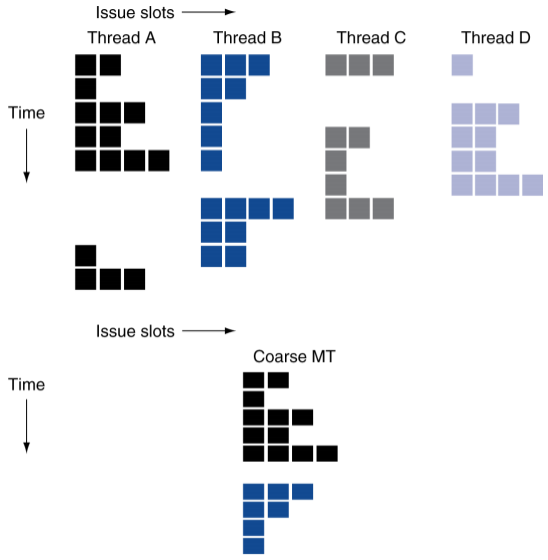
- 在每个周期后切换线程

- 交错不同的指令执行
- 如果一个线程阻塞，则执行其他线程

优缺点

- 可以同时隐藏由短阻塞和长阻塞引起的吞吐量损失
- 但是降低了单个线程的执行速度

粗粒度多线程 Coarse-grain multithreading



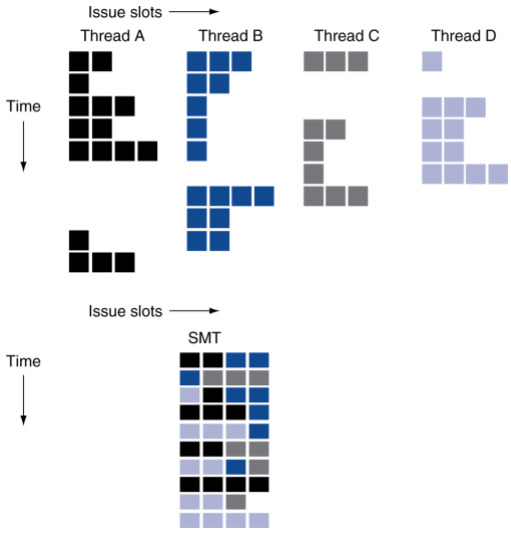
特点

- 仅在高开销阻塞的时候才进行切换
- 简化硬件，但不隐藏短阻塞
- 一般仅在一些重要事件（如 L-2 cache miss）之后进行进程切换

优缺点

- 在隐藏吞吐量损失能力方面受限

同时多线程 Simultaneous Multithreading（SMT）



它使用多发射动态调度流水线的资源挖掘线程级并行，同时保证指令级并行

- 从多个线程调度指令
- 当函数单元可用时，来自独立线程的指令执行
- 在线程中，通过调度和寄存器重命名来处理依赖关系

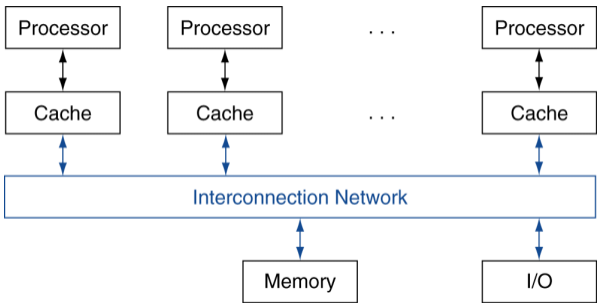
例如 Intel Pentium-4 HT 使用 SMT

5. 多核和其它共享内存的多处理器

共享内存多处理器 SMP Shared Memory Multiprocessor

- 为程序员提供跨越所有处理器的单一物理地址空间的多处理器
- 这些系统共享同一物理地址空间，但是它们可以在自己的虚拟地址空间中单独地运行程序

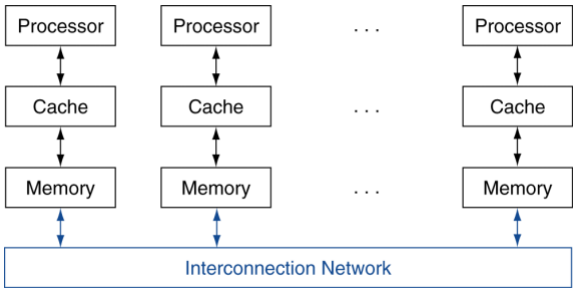
下图为 SMP 的架构



消息传递多处理器 Message Parsing Multiprocessors

- 每个处理器有自己的物理地址空间
- 这些多处理器通过显式的**消息传递 message passing** 进行通信

下面是 MMP 的架构



集群

通过局域网连接的一系列计算机

- 每个都有自己的存储器和操作系统
- 通过标准网络开关将 I / O 系统相互连接
    - 以太网 / 交换机、因特网等
- 适用于具有独立任务的应用程序
    - Web、数据库
- 高可用性、可扩展、价格合理

云计算和数据中心

它们可以归类为大型的集群，但是它们拥有更加复杂的体系结构和操作

这类计算机集群叫**仓储级计算机 Warehouse-Scale Computer WSC**

特点

- 大量简单的并行
- 软件即服务 Software as a Service（SaaS）：交互式 Internet 服务引用
- 经营成本：WSC 拥有更长的寿命

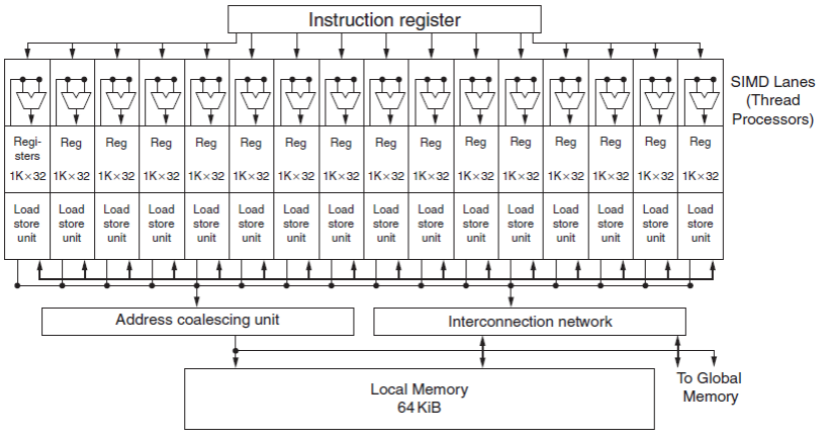
云计算 cloud computing

- 云计算公司可以将自己的数据中心的部分服务器出租给用户

6. GPU

GPU 和 CPU

- GPU 是补充 CPU 的加速器，它们专注于图形方向的资源
- GPU 解决问题的**规模**通常是 100MB - 10GB 范围，而不是 100GB 到 TB 范围的
- GPU 的特点是**不需要多级缓存**，在存储器请求和数据到达之间，它会执行**数百千计的与该请求无关的线程**
- GPU 是**面向带宽而不是面向延迟的**
    - 图形存储器很大，且具有高带宽
    - GPU内存比CPU小，4-6GB，而不是 256GB 或以上
- GPU 通过**多线程并行来获取高存储器带宽**，除了多线程，它还有很多 SIMD 处理器，相比 CPU 有更多的线程和处理器，它是一个由**多个多线程 SIMD 处理器组成的大型 MIMD 处理器**



主流趋势为

- CPU / GPU 异构系统
- CPU 用于顺序代码，GPU 用于并行代码

GPU 的编程语言和 API

- DirectX、OpenGL
- 面向图形编程的 C、高级选然语言（HLHL）
- Compute Unified Device Architecture CUDA

xPU

未来有更多的 xPU 被开发

- CPU：便于控制，顺序编程
- GPU：适合图形处理，并行编程
- TPU：Tenser 处理单元，专门用于加速 tenserflow，适用于机器学习的训练和测试
- DPU：深度学习处理单元，由DeePhi Tech提出，基于 FPGA 的处理单元
- NPU：神经网络处理单元，如 IBM 的 TrueNorth
- BPU：大脑处理单元