并行编程笔记-v 0.1

2019 年 1 月 14 日

1 并行计算简介

什么是并行计算 传统软件是串行执行的：

- 一个大问题被分散成多个分散的指令。
- 每个指令依次顺序执行。
- 所有指令在单处理器中执行。
- 一次只有一个执行在处理器中执行。

并行计算：

- 一个大问题被分散成多个可以同时执行的任务。
- 每一任务可以被分散成多个指令。
- 每一任务中的指令可以在不同的处理器中同时执行。
- 需要全局的调度机制。

概念和术语

- 冯·诺依曼结构（von Neumann Architecture）
  略
- 弗林分类（Flynn‘s Taxonomy）
  - SISD：Single Instruction stream Single Data stream
  - SIMD：Single Instruction stream Multiple Data stream
  - MISD：Multiple Instruction stream Single Data stream
  - MIMD: Multiple Instruction stream Multiple Data stream
- 一些通用术语
  - 超算（Supercomputing）
  - 高性能计算（High Performance Computing(HPC)）
  - 结点（Node)：在超算领域用来指代一台计算机，超级计算机由多个通过网络连接在一起的计算机构成。[1]

[1] 包含I/O,处理器等部件

- CPU/Socket/Processor/Core
- 任务；逻辑上的划分
- 管道：流水线
- 共享内存
- 分布式内存：每个处理器均配有相应的内存
- 对称多处理器：每个处理器所拥有的资源，和到每个资源的花费相等。
- 通讯
- 同步
- 粒度：并行工作划分的细致程度
- 并行总成：包括读取，处理，输出操作之后总体结果
- 阿姆达尔定律(Amdahl's law）$speedup = \frac{1}{1-P}$[2]

[2]可加速潜能，与可并行的代码数量有关

2 并行计算内存架构

2.1 共享内存

内存由多个处理器共享，分为UMA(Uniform Memory Access)和NUMA(Non-Uniform Memory Access)。

编程简单。

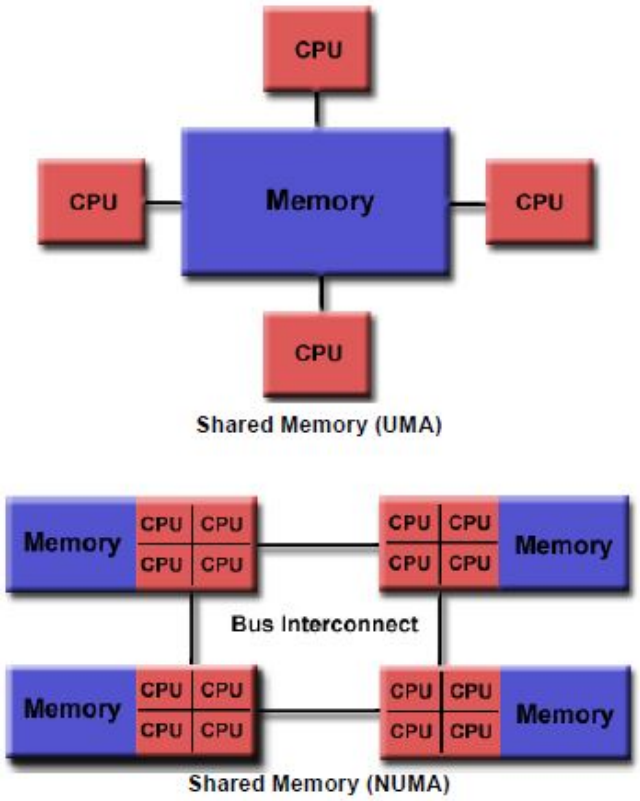

图 1: 共享内存

2.2 分布式内存

每个处理器都拥有自己的本地内存。

每个内存都有对应的处理器，访问速度快。

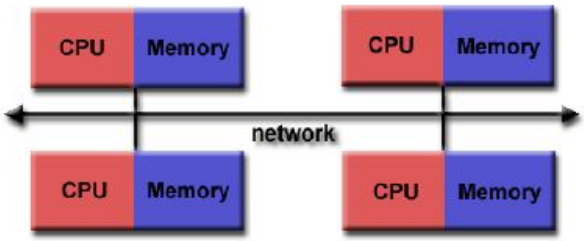

图 2: 分布内存

2.3 混合分布-共享内存

上述两种模式混合。

3 并行编程的设计

3.1 先搞明白问题再编程

开发并行软件的第一步就是搞明白你需要用并行程序解决的问题.

在开始开发一个并行的程序之前,先搞明白这个问题是否可以并行解决.

一个例子: 计算一个分子的，数千个相互独立构象的势能,之后,计算能量构象的最小值.

这个问题可以并行解决.每个分子构象是独立的，是个并行问题.

另一个例子: 利用公式$F(n) = F(n-1) + F(n-2)$,计算斐波那契数列.

计算F(n)的值,首先需要计算F(n-1)和F(n-2)的值.

确定程序的'热点' :

1. 知道主要的工作其实是在哪完成的.其实大部分科学或技术程序中的大部分问题都是在一小块程序中解决的.
2. 分析工具能帮上忙.
3. 关注平行的热点问题,忽略占用cpu很少的部分.

确定程序的'瓶颈' :

1. 是否有不成比例的慢或者导致平行工作终止或等待的区域?例如,I/O通常会拖慢程序.
2. 也许重构程序或者使用不同的算法可以去掉没有必要的慢.

确定并行的限制.通常数据间的相关性就是其中一类 可能的话,可以开发其他算法.这也许是开发一个并行应用最主要的需要考虑的问题. 利用优化过的第三方并行软件和高度优化的数学库(IBM's ESSL, Intel's MKL, AMD's AMCL, etc.)

3.2 分割

设计并行程序的第一步就是将一个问题分成多个分散的"块",这叫分割.有两种分割的基本方法:区域分割,功能分割.

3.2.1 区域分割

这种分割,跟问题有关的数据会被分割,每个并行的任务会只作用于一部分数据. 有不同的方法分割数据.

3.2.2 功能分割

对于这种方法,主要关注计算是如何被实现的,而非数据是如何被操作的.问题会被分成必须要被完成的任务,一个任务会实现一整个问题的一部分.

功能分割适合可以将问题分成不同任务的问题.

结合两分割方法是比较普遍的.

3.2.3 举例：生态系统模型

每段程序计算一组生物的总数量,该数量取决于邻近组的数量.随着时间发展,每个进程计算它的当前状态,然后和邻近组交换信息.随后再计算它的状态.

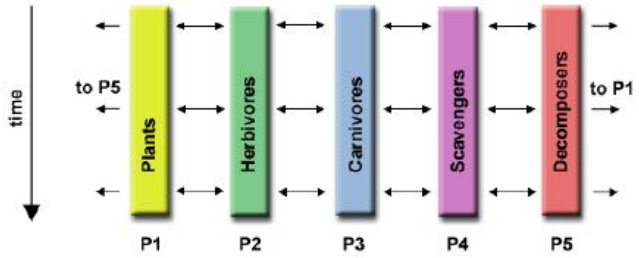

图 3: 生态系统模型

3.2.4 举例：信号处理

一个音频信号数据集通过四个滤波器.每个滤波器都是独立的,第一段数据必须在通过第二个滤波器之前,先通过第一个滤波器.与此同时,第二段数据正在通过第一个滤波器.使得四个进程都同时工作.

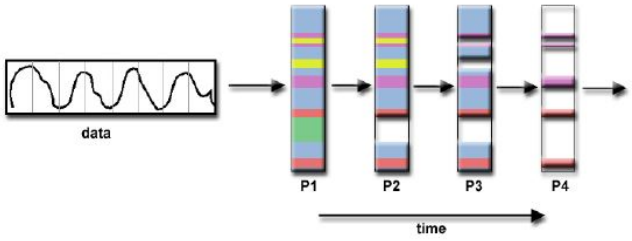

图 4: 音频处理模型

3.2.5 举例：气候模型

不同的任务将通过不同的模型的部分.箭头将指示不同部分间的数据交换.大气模型产生风速数据,然后用于海洋模型,海洋模型将产生海表面温度用于大气模型.

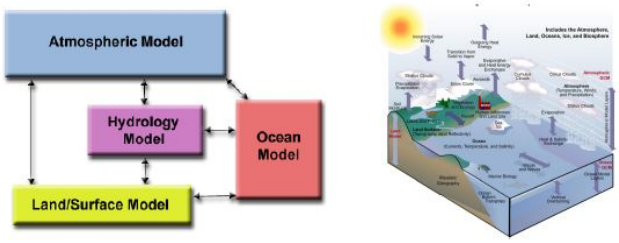

图 5: 气候模型

3.3 通信

3.3.1 谁需要通信?

通信的需求取决于具体问题.

3.3.2 不需要通信

有些问题可以分解成其实并不需要通信的任务.这些问题叫做"embarrassingly parallel", 不需要通信或者只需要很少的通信.

比如,想象一个图像处理问题,一张图片需要将每个黑像素或白像素翻转.图像数据可以很容易的被分为相互独立的任务.

3.3.3 需要通信

大多数应用都不会这么简单,需要不同任务间的数据分享.

比如,2-D热量扩散问题.一点的温度需要使用邻近的温度才能计算出来,改变邻近区域的温度将直接影响本点的温度数据.