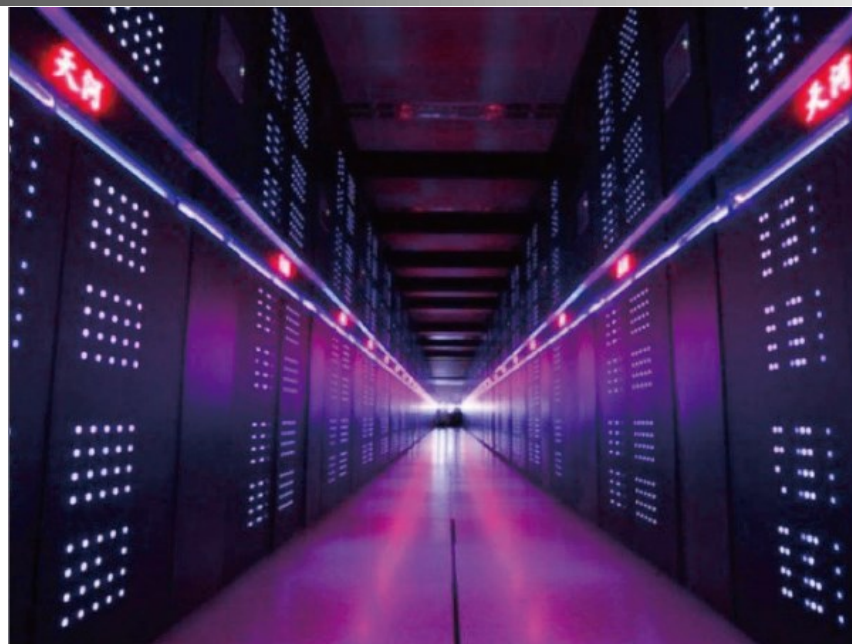


并行程序设计



佛山科学技术学院 数学与大数据学院 许红龙

longer597@163.com

第1章 为什么要并行计算

- 1.1 为什么需要不断提升的性能
- 1.2 为什么需要构建并行系统
- 1.3 为什么需要编写并行程序
- 1.4 怎样编写并行程序
- 1.5 我们将做什么
- 1.6 并发、并行、分布式

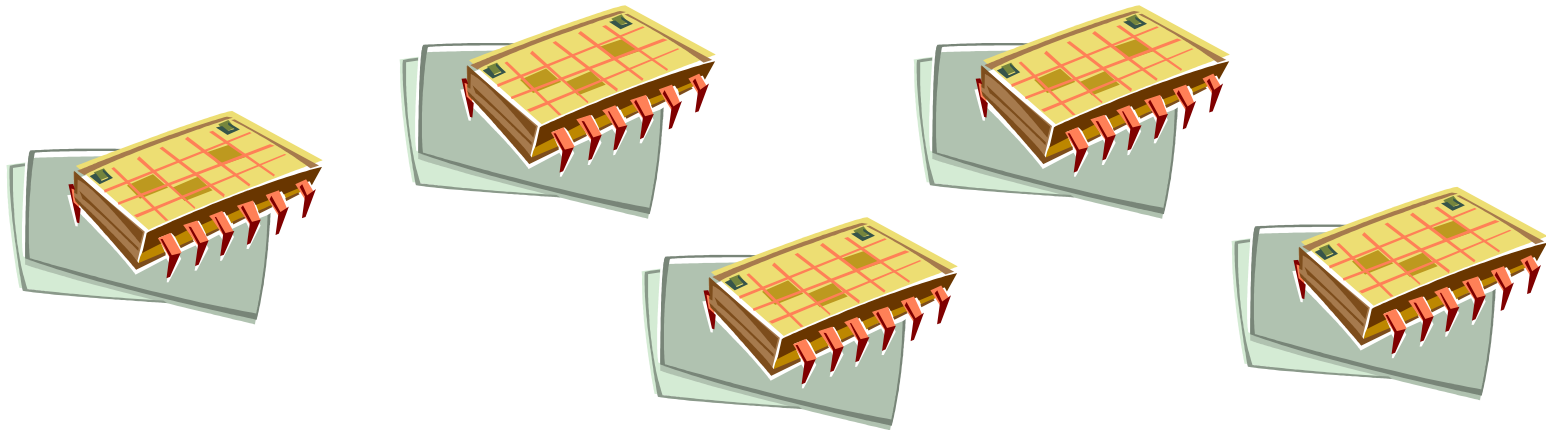
加快速度

- 1986年到2002年, 微处理器的性能平均每年提高50% 。
- 2002年起, 每年的增幅下降到20%左右。



解决方案

- 不是设计制造更快的微处理器, 而是把多个单处理器放在一个集成电路芯片上。



全球超级计算**TOP500**
<https://www.top500.org/>

现在就看程序员了

- 如果程序员不能利用更多处理器，那么增加这些处理器等于浪费。



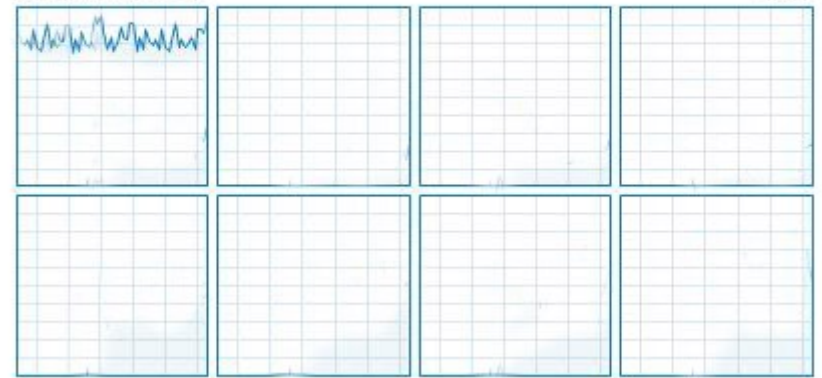
- 大多数情况下，串行程序不会加速。



CPU Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz

60 秒内的利用率 %

100%



为什么我们需要不断提高的性能

- **计算能力**在增加, 但**计算问题**和**计算需求**也在增加。
- 我们做梦也没想到: 很多问题已因计算能力提升而得到解决, 例如解码人类基因组。
- 更复杂的问题仍有待解决。

气候建模



表 1.2 欧洲预报中心 IFS 模式发展规划

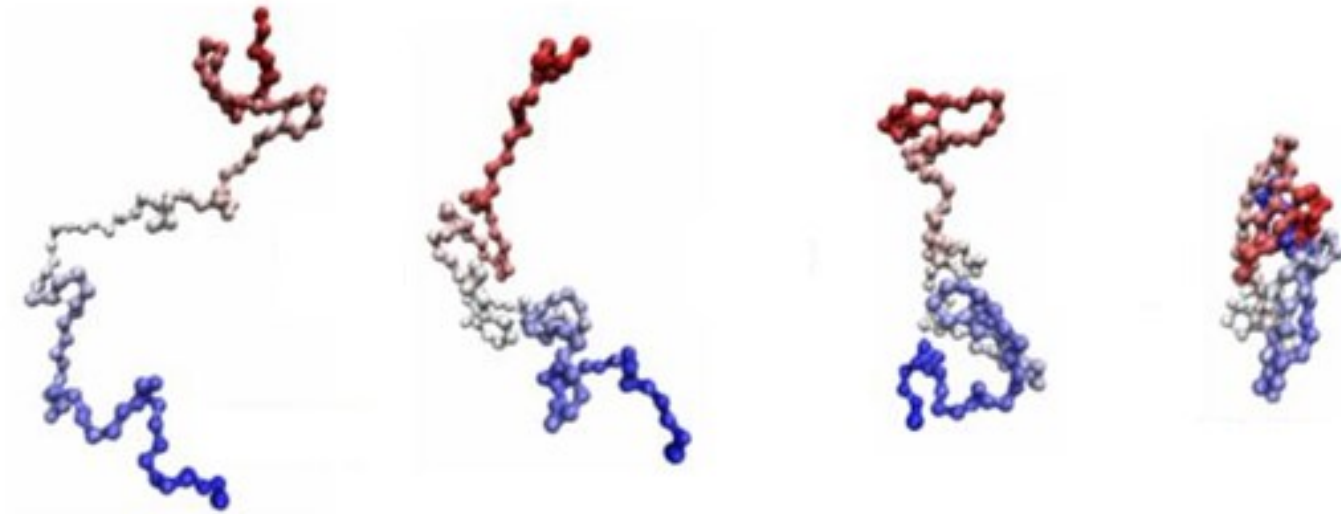
IFS 模式	预计实现时间	水平格点 分辨率/km	时间步长/s	预计需要 核数*
T1279 H	2011 (L91)	16	600	1100
	2012 (L137)			1600
T2047 H	2014-2015	10	450	6000
T3999 NH	2020-2021	5	240	80000
T7999 NH	2025-2026	2.5	30-120	$1 \times 10^6 - 4 \times 10^6$

* 核数估计是按照 10 天模式预报在 1 小时内运行完，所需 IBM Power7 处理器的核数

表 1.3 世界主要业务气象中心全球数值天气预报业务模式并行方式

预报中心 (国家/组织)	全球模式	主要并行方式	是否支持 openMP
欧洲预报中心	IFS	MPI	是
英国气象局	MetUM	MPI	是
加拿大气象局	GEM(Global Environmental Multiscale Model)	MPI	是
美国环境预报中 心	GFS(Global Forecast System)	MPI	是
中国气象局	GRAPES_Global	MPI	否

蛋白质折叠



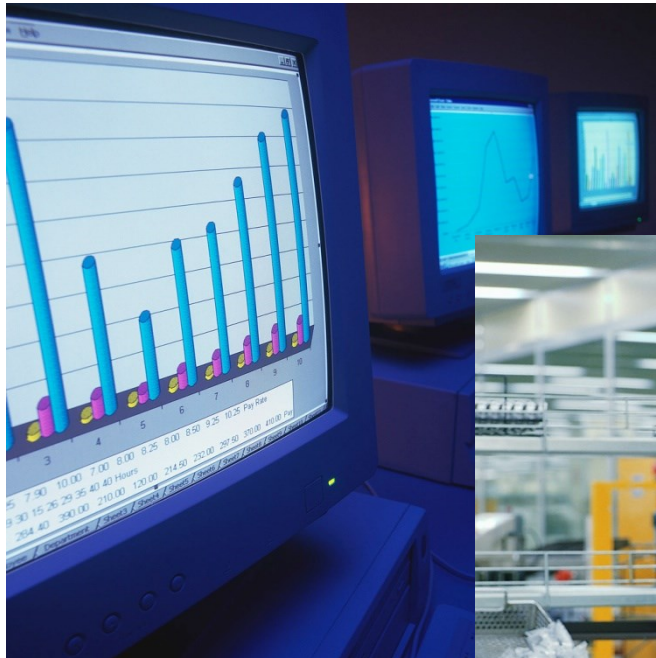
药物发现



能源研究



数据分析

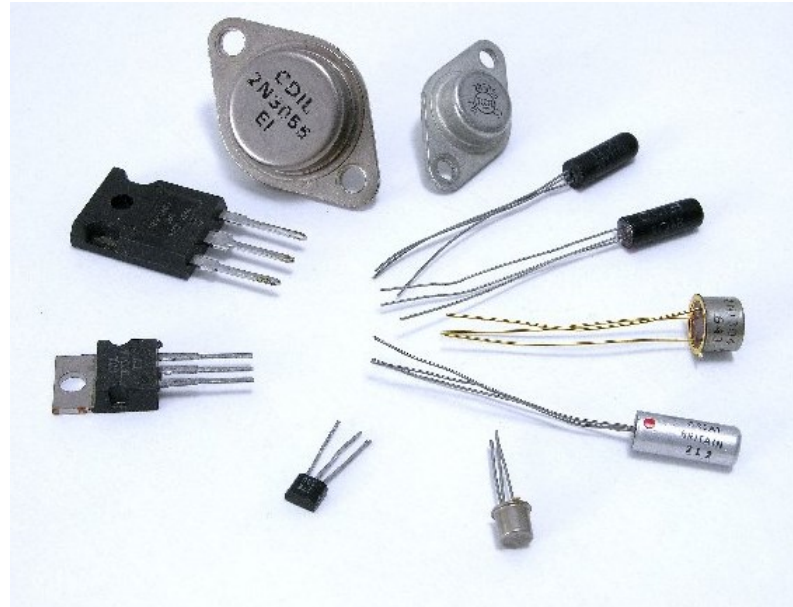


A close-up of a digital display showing a list of positive numerical values, likely representing data analysis results. The values are as follows:

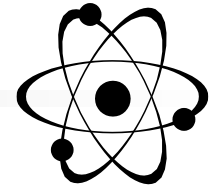
+2.688
+5.000
+1.500
+1.125
+1.062

为什么我们要构建并行系统

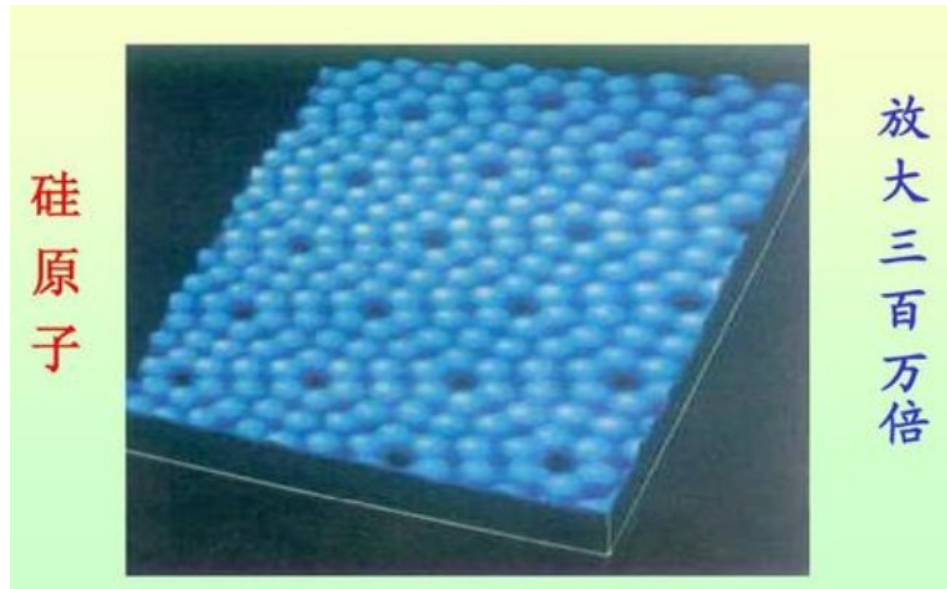
- 到目前为止, 性能的提高是由于晶体管密度的增加。



物理上的一节课

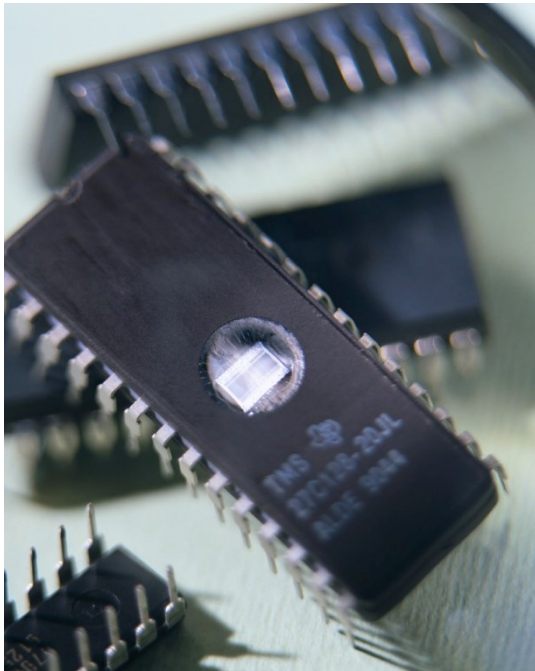


- 更小的晶体管 = 更快的处理器
- 更快的处理器 = 更高的功耗（主频的3次方）
- 更高的功耗 = 增加的热量
- 增加的热量 = 不可靠的处理器



解决方案

- 从单核系统转移到多核处理器。
- "核" = 中央处理器 (CPU)



- 引入并行性!!!

为什么我们需要编写并行程序

- 运行串行程序的多个实例通常没有意义。
- 例如，运行你最喜欢的游戏的多个实例？
- 你真正想要的是它**运行得更快**。



处理串行问题的方法

- 重写串行程序, 使其并行。
- 编写自动将串行程序转换为并行程序的翻译程序? ——鲜有突破!
- 最好的并行解决方案是——设计一个全新的算法


例子

- 计算 n 个值并将它们相加。
- 串行解决方案:

```
sum = 0;
for (i = 0; i < n; i++) {
    x = Compute_next_value(. . .);
    sum += x;
}
```

示例 (续)

- 我们有 p 内核, p 比 n 小很多。
- 每个核心执行大约 np 值的部分和。



```
my_sum = 0;
my_first_i = . . . ;
my_last_i = . . . ;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
    my_x = Compute_next_value( . . . );
    my_sum += my_x;
}
```

每个核心都使用自己的私有变量
并执行这个代码块
独立于其他内核。

示例 (续)

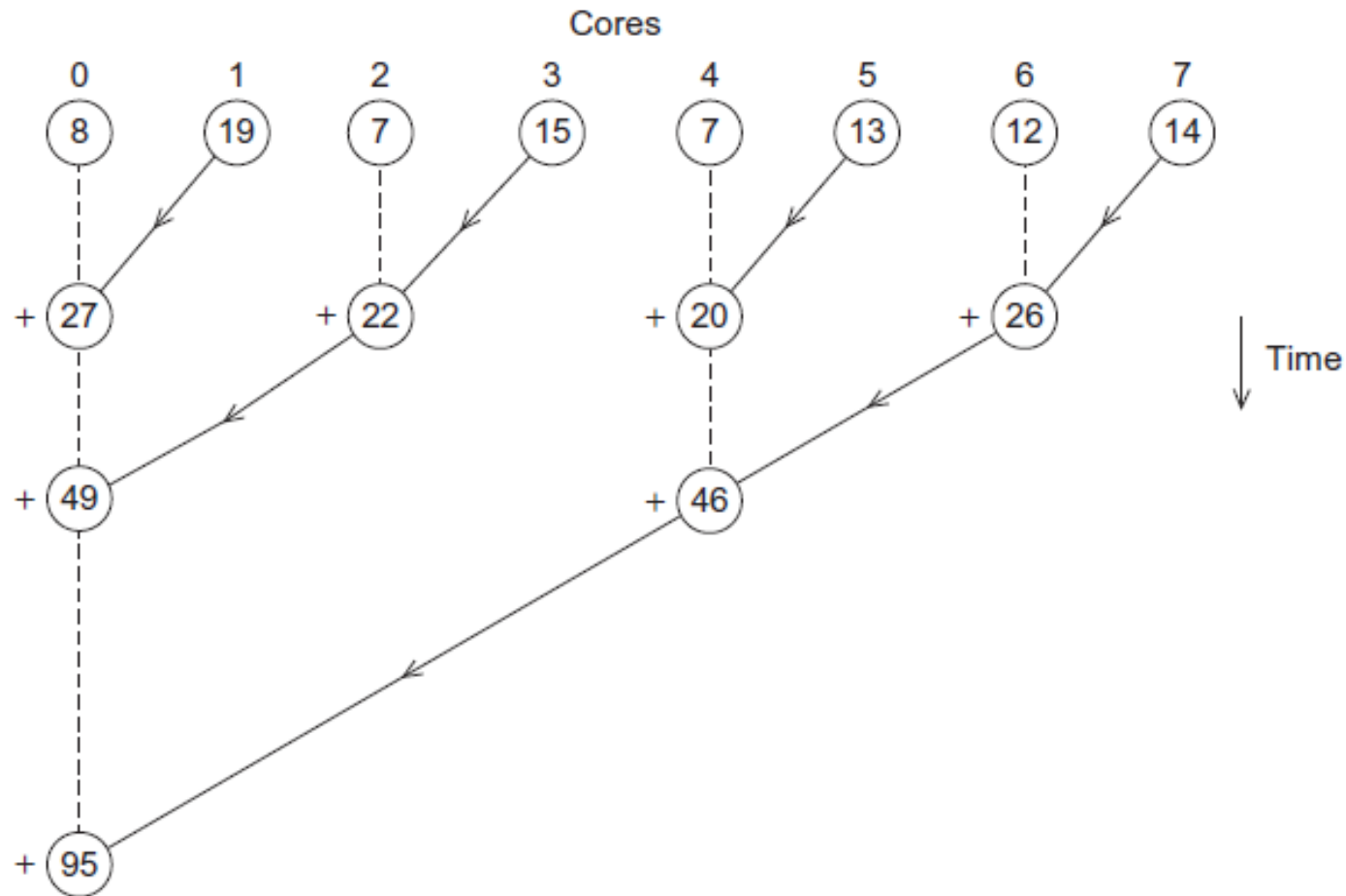
- 在每个核完成代码的执行后, 是一个私有变量 `my_sum` 包含由其调用计算的值的总和 `Compute_next_value`.
- 例如, 8个核, $n = 24$, 然后调用 `Compute_next_value` 返回:

1, 4, 3, 9, 2, 8, 5, 1, 1, 1, 1, 5, 2, 7, 2, 5, 0, 4, 1, 8, 6, 5, 1, 2, 3, 9

思考：
有没更好的方法
计算全局和？



形成全局和的多个核



分析

- 在第一个示例中, 主内核执行**7次接收和7次求和**。
- 在第二个示例中, 主内核执行**3次接收和3次求和**。
- 提升超过**2倍**!

分析 (续)

- 随着核数量的增加, 差异更为显著。
- 如果我们有1000个核:
 - 第一个示例将要求核0执行999次接收和999次求和。
 - 第二个示例只需要10次接收和10次求和。
- 几乎100倍提升!

怎样编写并行程序？

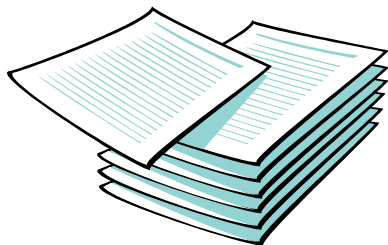
- 任务并行

- 将待解决问题所需要执行的各个任务分配到各个核上执行。

- 数据并行

- 将待解决问题所需处理的数据分配给各个核。
 - 每个核都对数据的一部分执行类似的操作。

5道题
100个学生
4个助教



p 教授的评分助理



助教1

助教2

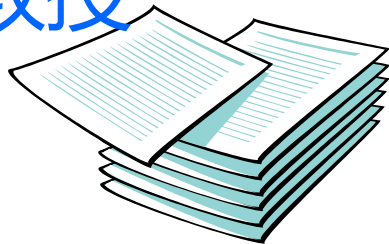
助教3

助教4

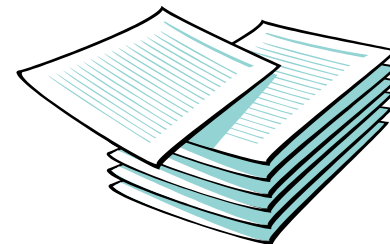
别问我在哪？正在跟你捉迷藏呢

分工--数据并行性

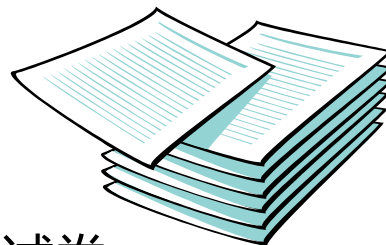
P教授



20份试卷

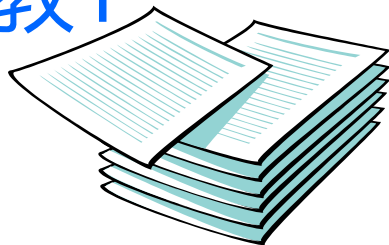


20份试卷 助教3

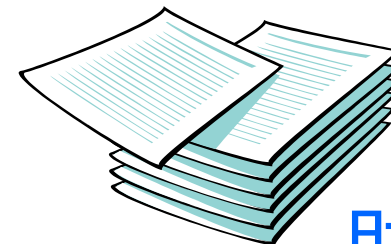


20份试卷 助教2

助教1



20份试卷



20份试卷 助教4

分工--任务并行性

P教授



问题1



助教3

问题4



助教2

问题3

助教1



问题2



助教4

问题5

协调

- 多个核通常需要协调他们的工作。
- 通信—一个或多个内核将其当前部分总和发送到另一个内核。
- 负载平衡—在内核之间均匀地共享工作, 这样就不会有个别核负载过重。
- 同步—因为每个核心都在自己的空间工作, 速度可能不同, 有时需要让快者等待慢者。



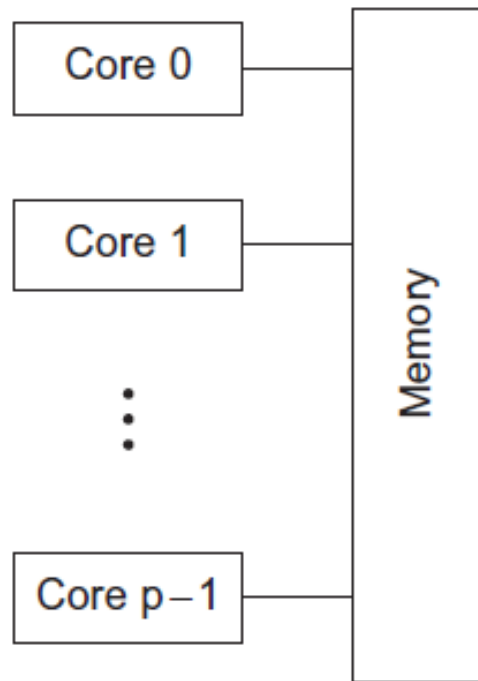
我们将做什么

- 学习编写显式并行的程序。
- 使用C语言。
- 对C使用二个不同的扩展。
 - 消息传递接口 (MPI)
 - OpenMP

并行系统的类型

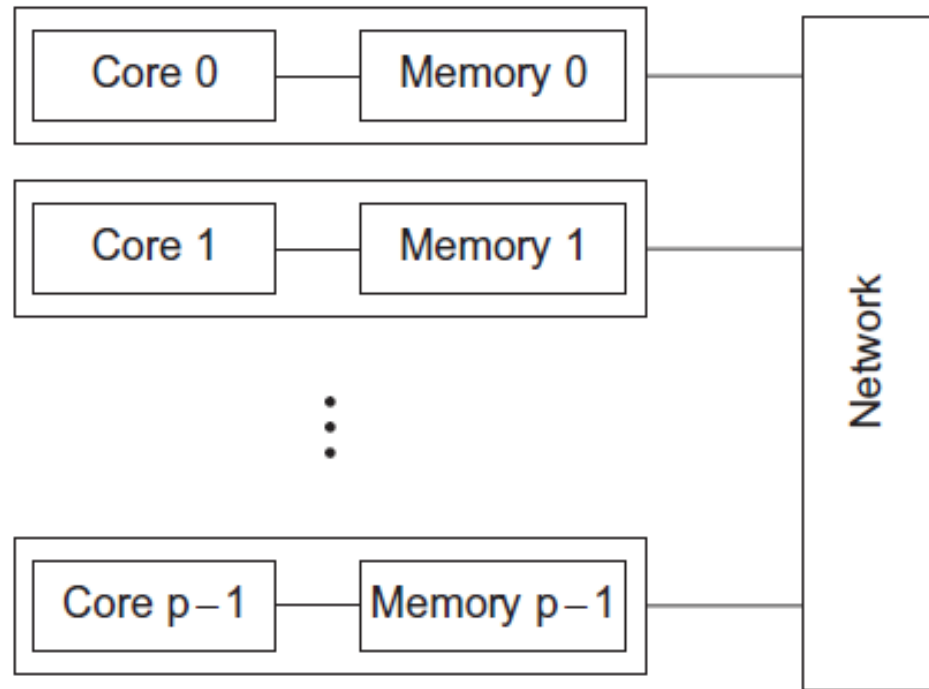
- 共享内存
 - 各个核可以共享访问计算机内存。
 - 通过检测和更新共享内存中的数据来协调各个核。
- 分布式内存
 - 每个核都有自己的私有内存。
 - 核之间显式通信（类似通过网络发送消息）。

并行系统的类型



(a)

共享内存



(b)

分布式内存

术语

- 并发计算：一个程序的多个任务在同一个时段内可以同时执行。
- 并行计算：一个程序通过多个任务紧密协作来解决某个问题
- 分布式计算：一个程序需要与其他程序协作来解决某个问题。

注意**并发**与**并行**的区别，并发只是利用了**CPU**单个核的多个时间片，表面上看似并行，实际**并非真正的并行**。

小结

- 传统处理器性能提升速度不断降低->多核处理器
- 串行程序（单核），并行程序（多核）
- 翻译程序难以完成并行化
- 协调：通信、负载平衡、同步
- MPI（分布式内存），OpenMP（共享内存）
- 并发计算，并行计算，分布式计算