

并行处理及分布式系统

计算机科学与技术教研室

张　全

中国超级计算机，从屈辱到世界第一，看呆美国的逆袭



理论教学（40学时）

- 第一章　为什么要学习并行计算（2学时）
- 第二章　并行硬件和并行软件（4学时）
- 第三章　MPI分布式内存编程（10学时）
- 第四章　Pthreads共享内存编程（10学时）
- 第五章　OpenMP共享内存编程（10学时）
- 第六章　分布式系统概述（4学时）

实验教学（8学时）

序号	项目编码	项目名称	学时
1	001063010201	MPI程序设计实验	2
2	001063010202	Pthreads程序设计实验	2
3	001063010203	OpenMP程序设计实验	2
4	001063010204	综合实验（OneAPI）	2

考核方式与成绩评定

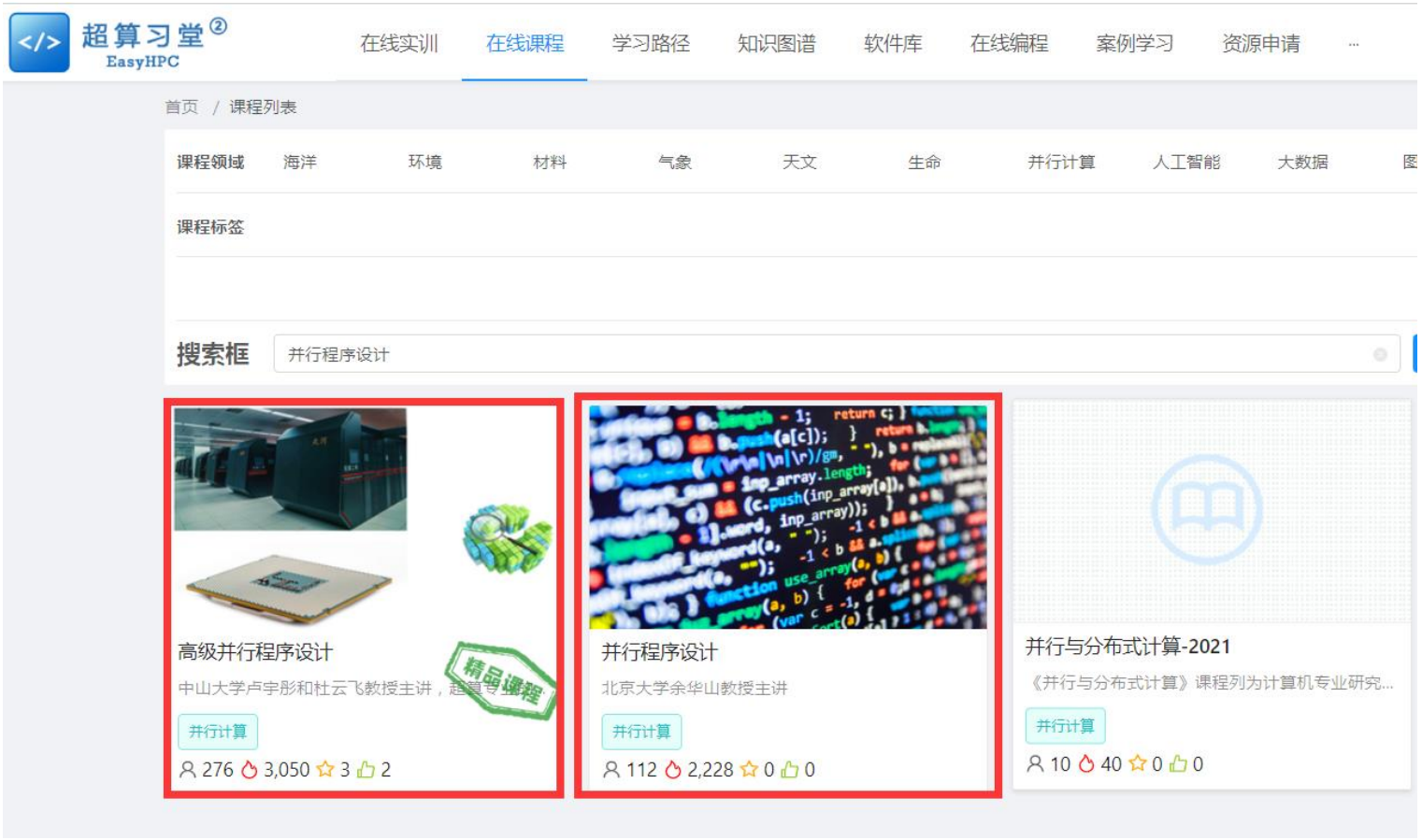
成绩核算方法		
成绩构成	分项成绩	考核方式
课堂（80%）	期末成绩（80%）	期末考试（闭卷）
	平时成绩（20%）	作业（80%）
		考勤（20%）
实验（20%）	平时成绩(100%)	实验报告（80%）
		考勤（20%）

教材及主要参考资料

- （一）教材：
  - 并行程序设计导论，帕切克著，邓倩妮等译，机械工业出版社，2019年3月第1版第9次印刷
- （二）参考书：
  - 分布式计算——原理，算法与系统，克谢姆卡雅尼著，余宏亮，张冬艳译，高等教育出版社，2012.6

课程资源

- https://easyhpc.net/



课程资源

- https://easyhpc.net/



学科竞赛

全国并行应用挑战赛

http://www.hpc-cpc.com/

第四届“神威杯”国产CPU并行应用挑战赛		
队伍编号	单位	队名
金奖		
CPC005	西南石油大学	攻壳机动队
银奖		
CPC041	广东工业大学	从0到1队
CPC049	山东大学	乘风破浪的幼儿园园霸队
铜奖		
CPC007	中国科学院高能物理研究所	真空中的方形鸡队
CPC044	北京大学	代码都队
CPC030	中国科学技术大学	鸿雁队
CPC037	青海大学	HDACP队
CPC012	清华大学	Earth814队
并行基金奖		
CPC013	信息工程大学	科学大道队
CPC026	西安交通大学	404扛把子队

“神威·国实杯”第六届国产CPU并行应用挑战赛 - 晋级名单		
队伍编号	队伍名称	单位/学校
CPC20220079	哞哞草原	山东科技大学
CPC20221276	知乎AT优化少年	并行优化爱好者
CPC20221585	桃李特攻队	清华大学
CPC20223247	郑大摸鱼队	郑州大学
CPC20223396	shell	齐鲁工业大学
CPC20223460	划水不队	山西大学&西北工业大学
CPC20224673	所以爱会消失对不队	山东大学
CPC20224875	003	拓竹科技&华声医疗&顺丰科技
CPC20225116	宛香都	山东大学
CPC20226376	楷骐今天吃嘉然	浙江大学
CPC20226566	TUT1	太原理工大学
CPC20227205	Omega	西南石油大学
CPC20228765	Lambda	西南石油大学
CPC20229101	友谊地久天长	燕山大学
CPC20229520	山东建投数据科技	建投数据科技（山东）有限公司
CPC20229988	灵犀	齐鲁工业大学&齐鲁工业大学（山东省科学院）

·按照队伍编号排名

学科竞赛

ACM中国-国际并行计算挑战赛



IPCC ACM中国 国际并行计算挑战赛

晋级名单公示

队伍编号	所在单位
IPCC20216734	武汉大学
IPCC20216032	上海交通大学
IPCC20211925	中山大学
IPCC20213994	上海交通大学
IPCC20217801	山东大学
IPCC20214918	山西大学
IPCC20215898	山东大学
IPCC20213737	中国科学技术大学
IPCC20216508	浙江大学
IPCC20214683	西北农林科技大学
IPCC20212037	技术爱好者
IPCC20214170	中国科学技术大学
IPCC20215707	中国科学院大学 中国科学院计算机网络信息中心
IPCC20215883	清华大学
IPCC20210620	西南石油大学
IPCC20212314	天津大学

学科竞赛

全国并行应用挑战赛



学科竞赛

全国并行应用挑战赛

https://cas-pra.sugon.com/



学科竞赛

亚洲大学生超级计算机竞赛**(ASC)**



超算与并行计算团队

- 超算与并行计算团队成立于**2019**年**7**月，团队主要针对“国产**CPU**并行应用挑战赛（**CPC**）”、“世界大学生超级计算机竞赛（**ASC**）”、“全国并行应用挑战赛（**PAC**）”、“中科院先导并行计算挑战赛”等。
- 团队依托于学校图像处理与并行计算实验室，目前分并行计算和机器学习两个方向对新生进行培养，具有完善的本硕培养体系。
- 团队获得“神威杯”国产**CPU**并行应用挑战赛金奖**1**项、并行基金奖**1**项

第四届“神威杯”国产CPU并行应用挑战赛		
队伍编号	单位	队名
金奖		
CPC005	西南石油大学	攻壳机动队
银奖		
CPC041	广东工业大学	从0到1队
CPC049	山东大学	乘风破浪的幼儿园园霸队
铜奖		
CPC007	中国科学院高能物理研究所	真空中的方形鸡队
CPC044	北京大学	代码都队
CPC030	中国科学技术大学	鸿雁队
CPC037	青海大学	HDACP队
CPC012	清华大学	Earth814队
并行基金奖		
CPC013	信息工程大学	科学大道队
CPC026	西安交通大学	404扛把子队



攻壳机动队：张鸿宇、马聪、蒋讯、涂然（夺得金奖，10万元）
指导教师：彭博、张全

超算与并行计算团队

全国一等奖		
队伍编号	学校/单位	队伍名称
CPC20225116	山东大学	宛香都
全国二等奖		
队伍编号	学校/单位	队伍名称
CPC20221585	清华大学	桃李特攻队
CPC20229520	建投数据科技(山东)有限公司	山东建投数据科技
全国三等奖		
队伍编号	学校/单位	队伍名称
CPC20220079	山东科技大学	哞哞草原
CPC20223247	郑州大学	郑大摸鱼队
CPC20224875	拓竹科技&华声医疗&顺丰科技	003
CPC20227205	西南石油大学	Omega
CPC20228765	西南石油大学	Lambda
CPC20229988	齐鲁工业大学	灵犀
并行基金奖		
队伍编号	学校/单位	队伍名称
CPC20224673	山东大学	所以爱会消失对不队
CPC20226566	太原理工大学	TUT1



Omega队：周昕航，赵洪杰，高士其，贾俊祥（全国三等奖，1万）
Lambda：夏卓昭，张强，李锡涛，胡蕊（全国三等奖，1万）
指导教师：彭博、张全

超算与并行计算团队

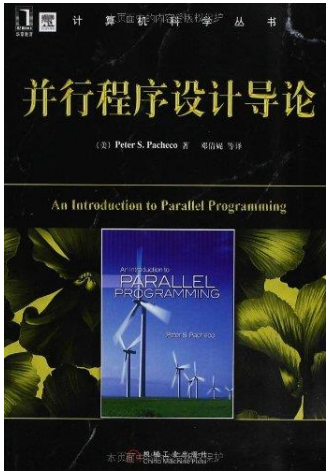
“神威·国实杯”第七届国产CPU并行应用挑战赛

全国总决赛榜单公布

全国特等奖		
队伍编号	学校/单位	队伍名称
CPC20238571	清华大学	桃李特攻队
全国一等奖		
队伍编号	学校/单位	队伍名称
CPC20233669	齐鲁工业大学(山东省科学院)	灵犀
CPC20239081	山东大学	蛋仔派队
全国二等奖		
队伍编号	学校/单位	队伍名称
CPC20231253	山东大学	深藏blue队
CPC20239946	西北工业大学	东大村小分队
CPC20232122	西南石油大学、四川大学	梦溪湖
CPC20230333	中国科学技术大学	鸿雁队
CPC20234972	西南石油大学	我们优化的都队
全国三等奖		
队伍编号	学校/单位	队伍名称
CPC20238250	中国科学院大学、中国科学院计算机网络信息中心	天才吴国太
CPC20239790	华东师范大学	大夏雷猴
CPC20230080	青海大学	优化入门
CPC20230766	西南石油大学	油理油情
CPC20230425	山东科技大学	叁幺幺
CPC20232442	齐鲁工业大学(山东省科学院)	亿金就学习
CPC20238496	华中科技大学	我想打PAC



梦溪湖队：张强、李锡涛、杨嘉苓、江国庆（全国二等奖，1万）
我们优化的都队：马皓严、张莹桥、罗潇棋、邢远杰（全国二等奖，1万）
油理油情队：廖洪樟、周广森、郑云鹤、关忠林（全国三等奖）
指导教师：彭博、张全、李艳



第一章

为什么要做并行计算

提纲

- 为什么我们需要不断提高的性能
- 为什么我们要建立并行系统
- 为什么我们需要写并行程序
- 我们如何编写并行程序?
- 我们要做什么？
- 并发、并行、分布式!

微处理器的发展历程1/2



1942: Digital Electric Computer

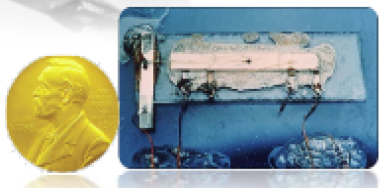
(Atanasoff and Berry)



1956

1947: Transistor

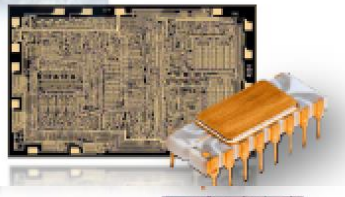
(Shockley, Bardeen, and Brattain)



2000

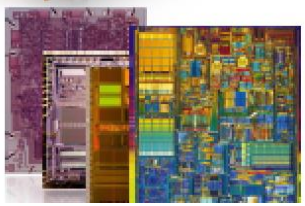
1958: Integrated Circuit

(Kilby)



1971: Microprocessor

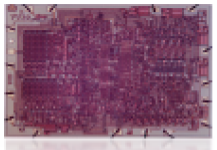
(Hoff, Faggin, Mazor)



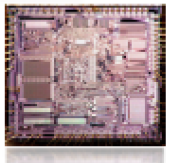
1971- Exponential growth

(Moore, 1965)

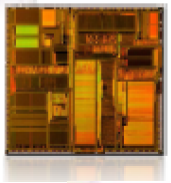
微处理器的发展历程2/2



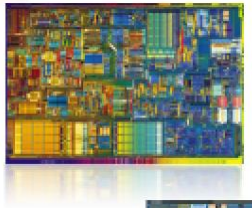
1971: 4004,
2300 trans, 740 KHz



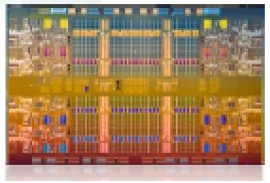
1982: 80286,
134 thousand trans, 8 MHz



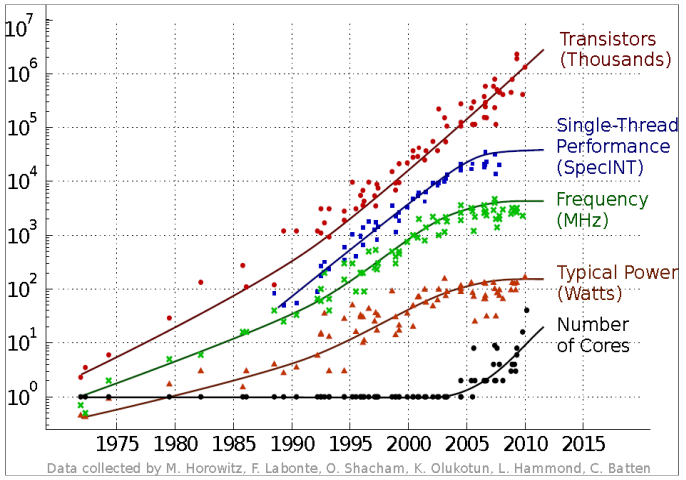
1993: Pentium P5,
1.18 mill. trans, 66 MHz



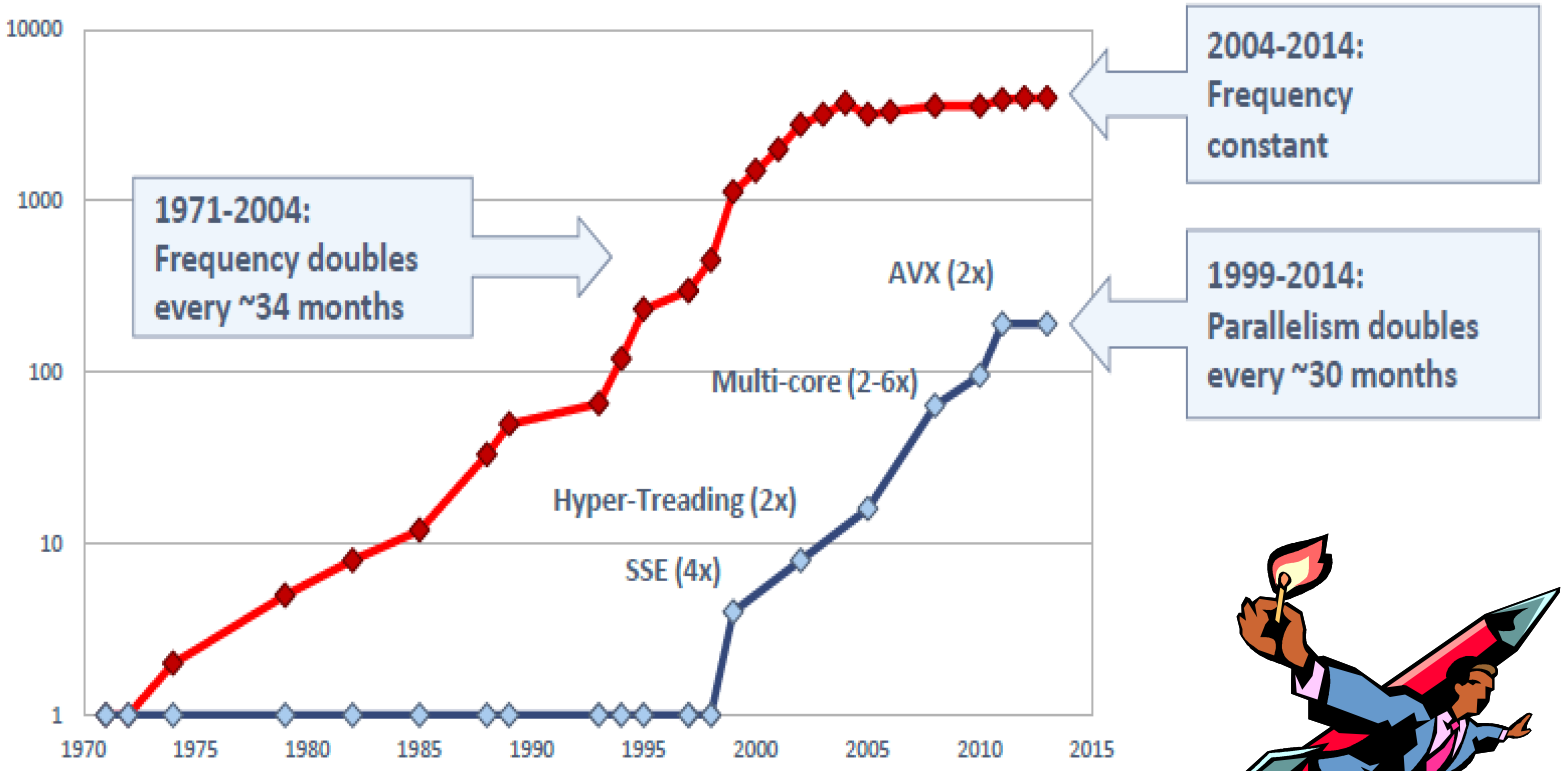
2000: Pentium 4,
42 mill. trans, 1.5 GHz



2010: Nehalem
2.3 bill. Trans, 8 cores, 2.66 GHz



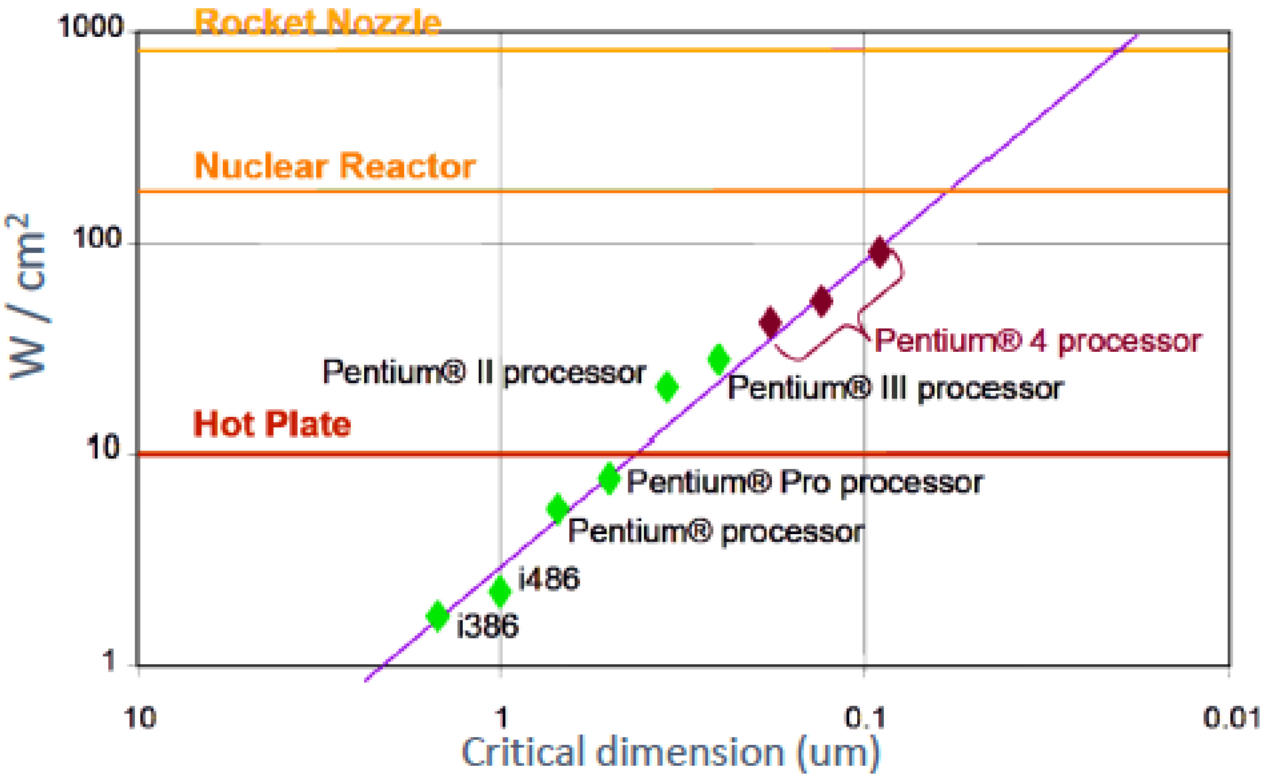
依靠主频提升性能的时代结束



- **1970-2004**：主频每**34**个月翻一番
- **1999-2014**：并行度每**30**个月翻一番

2004年发生了什么？

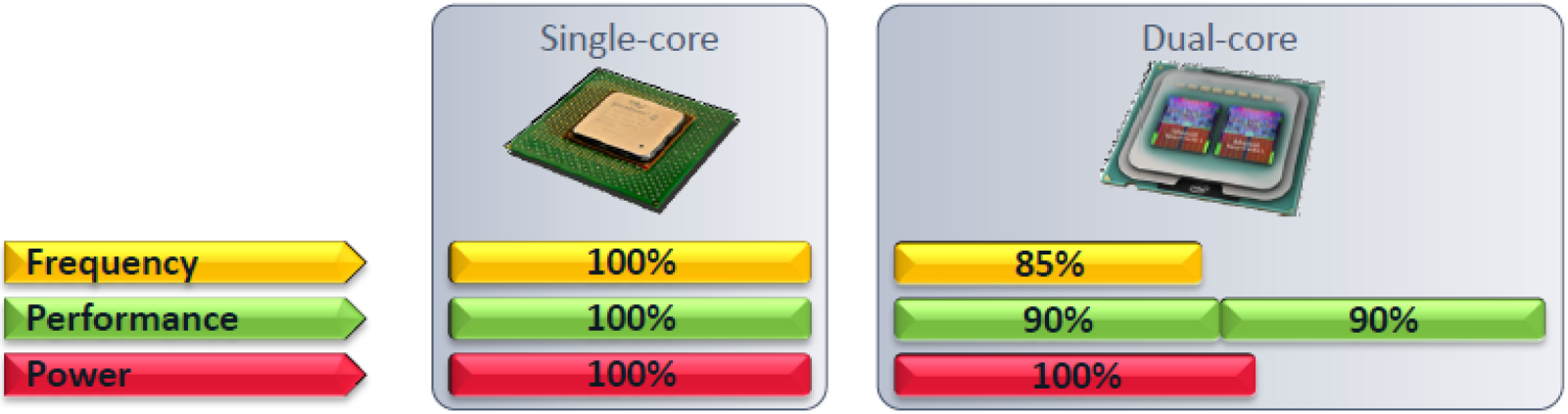
- 核反应堆堆芯的热密度:能量墙(**Power Wall**)
  - 传统的散热解决方案(散热器+风机)不能满足散热。
- 工业界解决方案:多核并行!



为什么并行？

微处理器的能量密度正比例与时钟频率的三次方:[1]

$$P_d \propto f^3$$



聪明的解决方案:与其设计和制造更快的微处理器，不如把多个处理器放在一个集成电路上。

[1] Brodtkorb et al. State-of-the-art in heterogeneous computing, 2010

程序员登场

- 如果程序员没有并行算法设计技能，添加再多的处理器并没有多大帮助
- 程序员不知道如何使用它们

- 串行程序不能从多处理器方案中获益(在大多数情况下)。



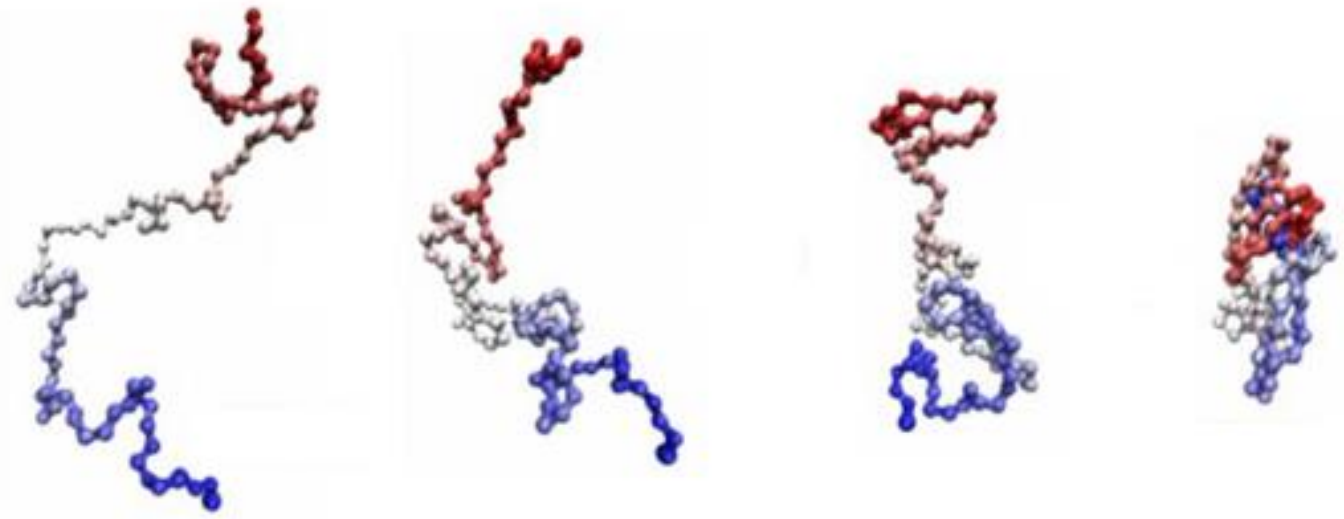
为什么我们需要不断提高的性能

- 计算能力在增加，但我们的计算问题和需求也在增加。
- 由于技术进步，我们从未想过的问题得到了解决，比如解码人类基因组
- 更复杂的问题仍有待解决。

气候模型



蛋白质折叠



新药研发





能源研究





数据分析





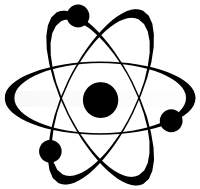


为什么我们要建立并行系统

- 到目前为止，性能的提高归因于晶体管密度的增加。
- 但也存在一些内在问题。



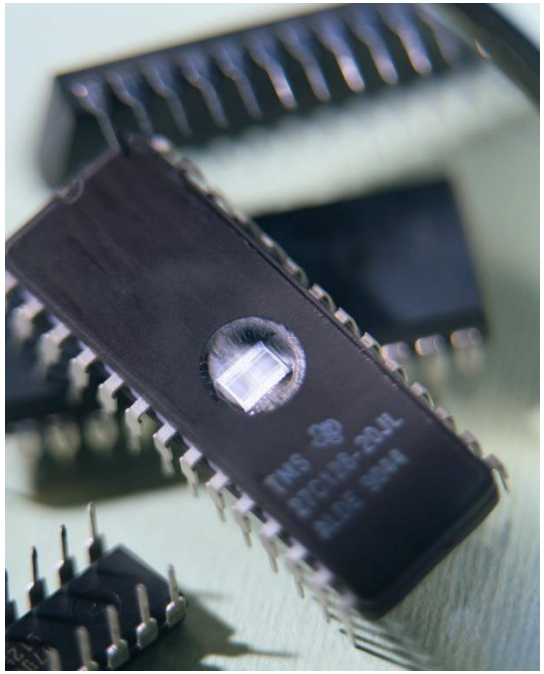
一些物理常识



- 更小的晶体管 = 更快的处理器
- 更快的处理器 = 更高的能耗
- 功率消耗增加 = 热量增加
- 增加的热量 = 不可靠的处理器.

解决方案

- 从单核系统转向多核处理器。
- “core” = central processing unit (CPU)



- 引入并行性!!

为什么我们需要写并行程序

- 运行一个串行程序的多个实例通常不是很有用。
- 想想运行你最喜欢的游戏的多个实例。
- 你真正想要的是让它跑得更快。



串行问题的处理方法

- 重写串行程序，使它们是并行的。
- 编写自动将串行程序转换为并行程序的翻译程序。
  - 很难做到。
  - 成功案例有限。

更多问题

- 一些编码结构可以被自动程序生成器识别，并转换为并行结构。
- 然而，结果很可能是一个非常低效的程序。
- 有时，最好的并行解决方案是退一步，设计一个全新的算法。

Example

- 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;
}
```

Example (继续)

- 有p个核，p比n小得多。
- 每个核执行一个近似n/p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;
}
```

每个核都使用自己的私有变量，独立于其他核并执行这段代码。

Example (继续)

- 在每个核心完成代码执行后，有一个私有变量my_sum，累加了Compute_next_value函数计算的值。
- Ex.8 个核，n = 24，然后调用Compute_next_value返回:

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

Example (继续)

- 一旦所有的核都完成了它们的私有my_sum变量的计算，它们通过将结果发送给指定的“主核”将最终结果相加，得到一个全局和。

Example (继续)

```
if (I'm the master core) {
    sum = my_x;
    for each core other than myself {
        receive value from core;
        sum += value;
    }
} else {
    send my_x to the master;
}
```

Example (继续)

Core	0	1	2	3	4	5	6	7
my_sum	8	19	7	15	7	13	12	14

Global sum

8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95

Core	0	1	2	3	4	5	6	7
my_sum	95	19	7	15	7	13	12	14

但是等等!

有一个更好的方法来计算全局和。



更好的并行算法

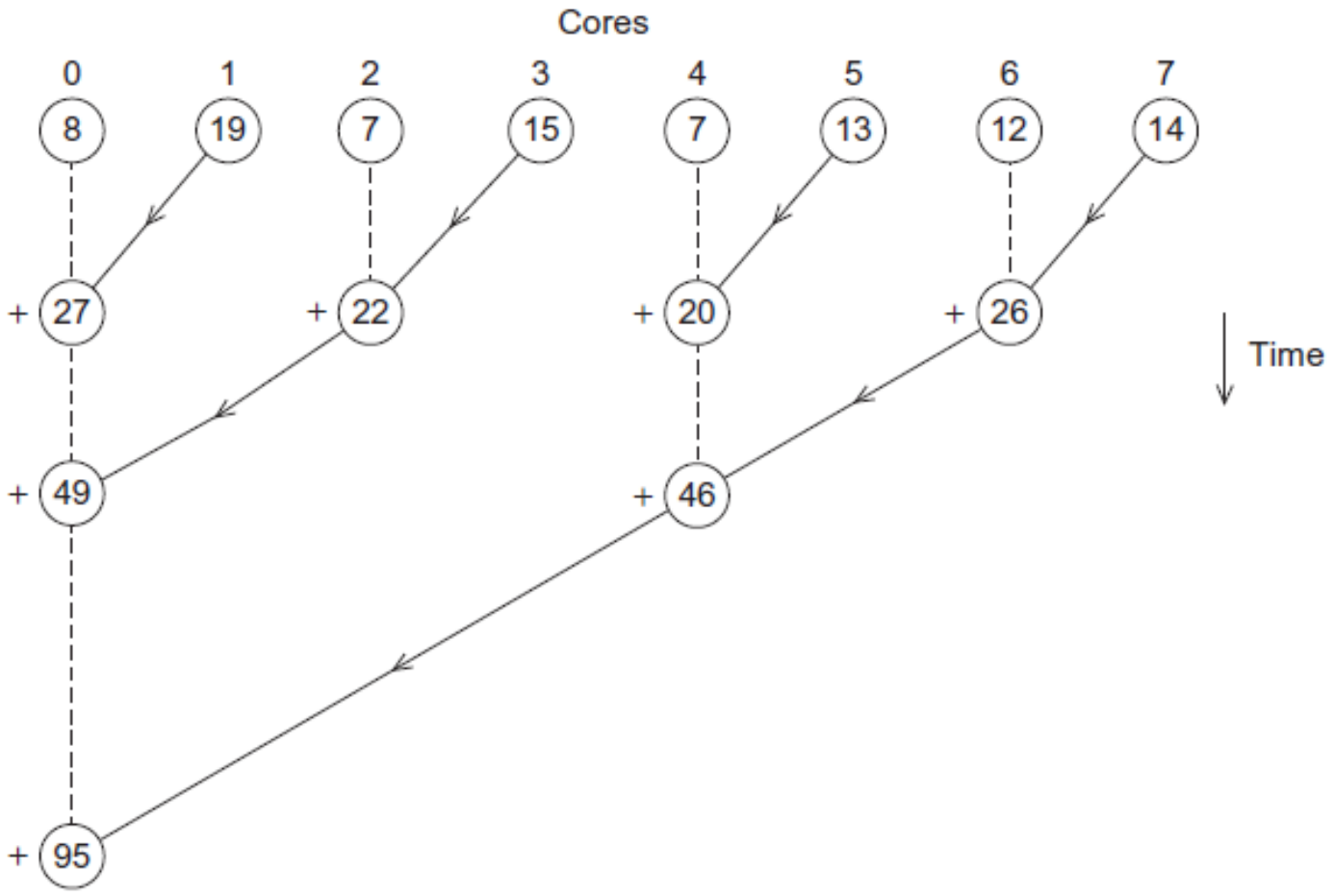
- 不要让主核做所有的任务。
- 在其它核分担任务.
- 将两个核配对，使Core 0的结果与Core 1的结果相加。
- Core 2将其结果与Core 3的结果相加，以此类推。
- 使用奇数和偶数的核对

更好的并行算法(继续)

- 现在，只对性能相当的核重复这个过程。
- Core 0 + core 2.
- Core 4 + core 6, 以此类推.

- 现在，core被4整除，重复这个过程，直到core 0得到最终结果。

多个核形成一个全局和



分析

- 在第1个例子中，主核执行7次接收和7次加法操作。
- 在第2个示例中，主核心执行3次接收和3次加法.
- 加速超过了2倍!

分析 (继续)

- 当内核数量更多时，这种差异更加显著。
- 如果我们有1000个核:
  - 第一个示例将要求master执行999接收和999加法.
  - 第二个示例只需要10个接收和10个加法.
- 这几乎加速了100倍!

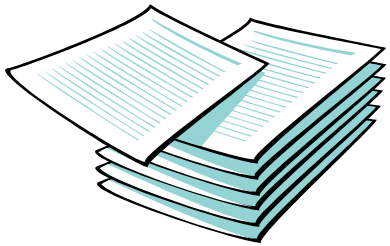
我们如何编写并行程序?

- 任务并行
  - 将任务划分为不同的子任务，由不同的核来完成，多个核协同完成总任务
- 数据并行
  - 将数据划分为不同是子数据，由不同的核来完成，多个核协同完成总的数据计算。
  - 每个核心对它负责的子数据执行类似的操作。

教授 P

15 个问题

300 份试卷



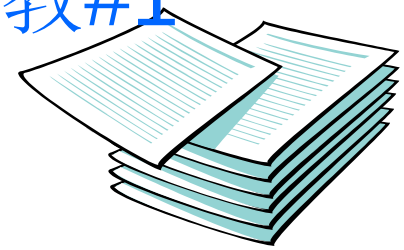


教授 P的助教

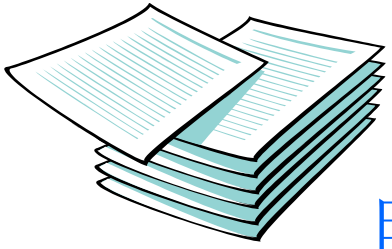


分工原则-数据并行

助教#1

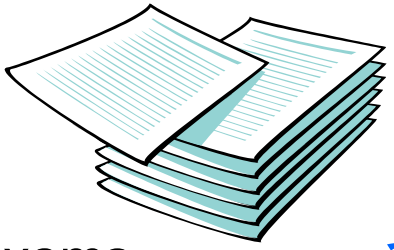


100 exams



100 exams

助教#3



100 exams

助教#2

分工原则-任务并行

助教#1



Questions 1 - 5



助教#3

Questions 11 - 15



助教#2

Questions 6 - 10

分工原则-数据并行

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

分工原则-任务并行

```
if (I'm the master core) {
    sum = my_x;
    for each core other than myself {
        receive value from core;
        sum += value;
    }
} else {
    send my_x to the master;
}
```

Master core'Tasks

1) Receiving
2) Addition

Slave core'Tasks

1) Sending

协同（Coordination）

- 核与核之间通常需要进行协同它们的任务.
- 通信（ Communication ）：一个或多个核将其当前的部分总和发送给另一个核.
- 负载平衡（ Load balancing ）：在内核之间均匀地分配工作，这样一个内核就不会负载过重。
- 同步（Synchronization）：因为每个核都有自己的工作节奏，所以要确保核不会比其他核走得太远。

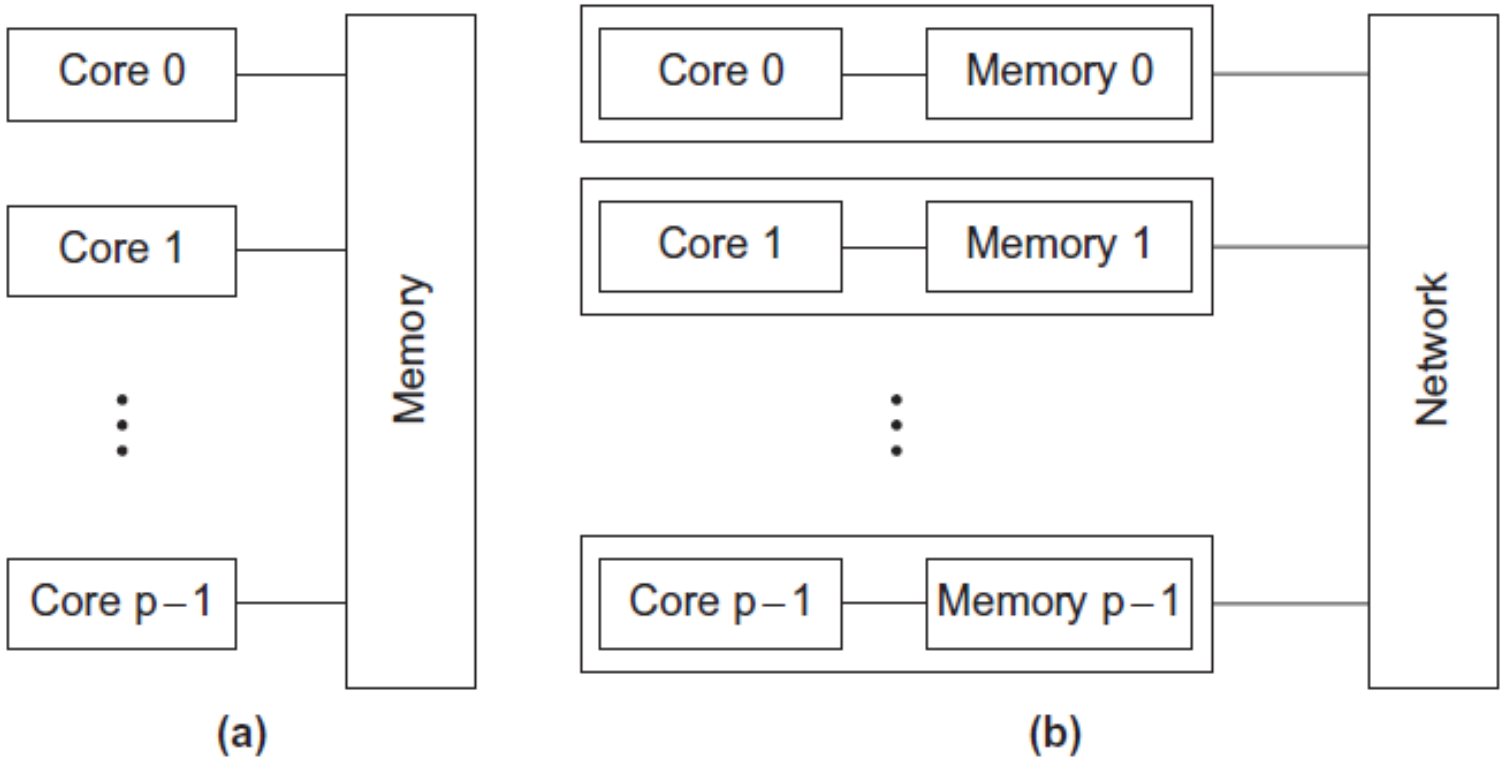
我们要做什么

- 学习编写并行程序.
- 使用C语言.
- 使用C的三种不同的扩展.
  - Message-Passing Interface (MPI)
  - Posix Threads (Pthreads)
  - OpenMP

并行系统类型

- 共享存储（Shared-memory）
  - 所有核可以共享计算机存储器的访问权限。
  - 通过内核检查和更新共享内存位置实现协同。
- 分布式存储（Distributed-memory）
  - 每个核都有自己的私有内存.
  - 核必须通过网络发送消息进行显式通信，从而实现协同。

并行系统类型



Shared-memory　　Distributed-memory

术语

- 并发计算（Concurrent computing）：多个任务可以在任何时刻进行。
- 并行计算（Parallel computing）：多个任务紧密合作来解决一个问题。
- 分布式计算（Distributed computing）：可能需要与其他程序合作来解决问题。

几点认识（1）

- 处理器的发展把我们带到了多核技术的门口
- 串行程序通常不能从多核中获益.
- 从串行程序代码中自动生成并行程序并不是实现多核计算机高性能的最有效方法.

几点认识（2）

- 学习编写并行程序需要学习如何协调内核.
- 并行程序通常非常复杂，因此，需要扎实的程序基础和开发技能.