北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Computer Organization and Architecture

Chapter 2

Performance Issues

School of Computer Science（National Pilot Software Engineering School）
AO XIONG (熊翱)
xiongao@bupt.edu.cn

Preface

We have learned:

- Basic Concepts and Computer Evolution
  - Introduction of Computer and this course 计算机和本课程的介绍
  - Why study computer organization and architecture 为什么要学习计算机组织与架构
  - What are organization and architecture 什么是组织与架构
  - What are structure and function 什么是结构和功能
  - A Brief History of Computers 计算机发展历史

Review

- 组成和架构
- 结构和功能
- 计算机的顶层功能
- 计算机的基本组成部分
- 提高计算机性能的关键因素

Review

- 组织与架构
  - 架构，也称体系架构，是程序员可以看到的这些属性。指令集，数据表示的位数、I/O机制、寻址技术等等，是体系架构相关的内容。
  - 组织，指的是这些特征是怎么实现的。通过什么控制信号来实现控制，采用什么接口，用什么内存技术，这些是计算机组成的内容。
- 结构和功能
  - 结构是组成计算机的各个组件之间相互关联的方式。
  - 功能则是作为整个结构的一个组成部分的单个组件可以进行的操作。
- 计算机的顶层功能包括数据处理，数据存储，数据移动以及控制。
- 计算机的组成部分包括：运算器，控制器，存储器，输入，输出。
- 平衡的思想

Preface

We will focus the following contents today:

- Performance Issue
  - What are the factors that affect computer performance? 什么因素影响计算机的性能
  - Amdahl' laws Amdahl定律
  - How to measure and compare the performance of computers? 如何评测计算机的性能

Outline

- Designing for Performance　性能设计
- Amdahl’s Law Amdahl 定律
- Basic Measures of Computer Performance 计算机性能基本测量

Designing for Performance 性能设计

- Performance and storage capacity of the computer are constantly improving, and the price is constantly decreasing 计算机性能和容量不断提高，价格下降
- Computers have been applied to all fields of life, and people's requirements for computers are getting higher and higher 计算机应用到生活的各个领域，人们对计算机的要求也越来越高
- Today, the applications require the great power of today’s microprocessor-based 今天，这些应用需要基于微处理器的强大功能
  - Image processing 图像处理
  - Speech recognition 声音识别
  - Videoconferencing 视频会议
  - Multimedia authoring 多媒体创作
  - Simulation modeling 仿真建模
  - Large model training 大模型训练

Designing for Performance 性能设计

- The factors that affect a computer performance directly 直接影响计算机性能的因素
  - Microprocessor speed 微处理器的速度
  - Performance balance 性能平衡
  - Improvements in chip organization and architecture 芯片组织和架构的提升

Microprocessor Speed 微处理器的速度

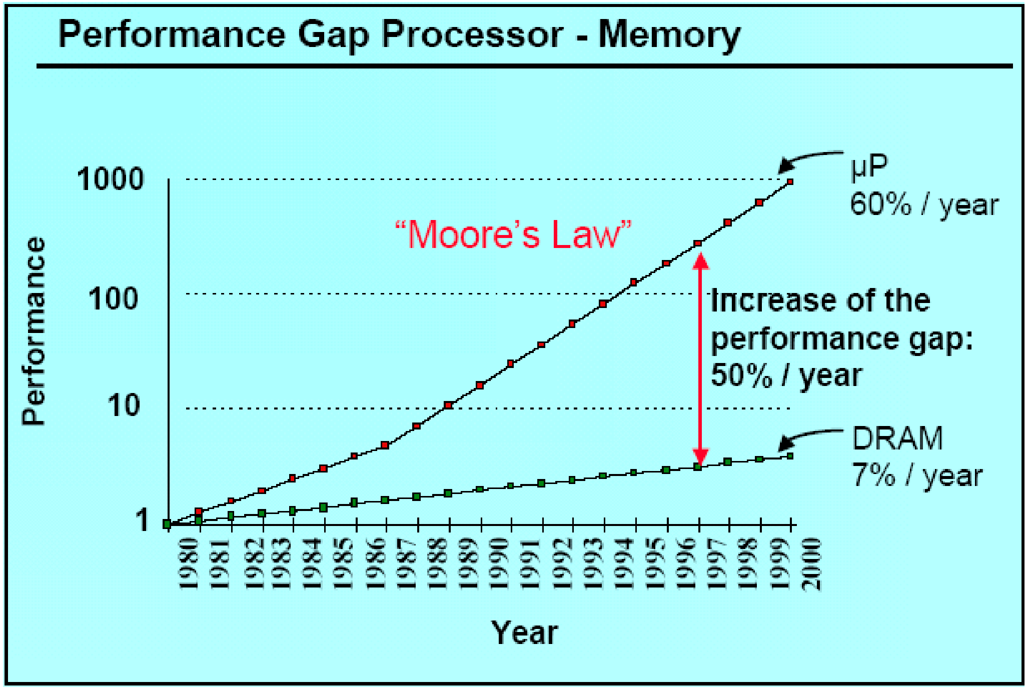
- According to Moore's theorem, by integrating more transistors on the chip and reducing the distance between circuits, the performance of microprocessors can be improved by 4-5 times every three years 根据摩尔定理，通过在芯片上集成更多的晶体管和缩短电路之间的距离，微处理器的性能每三年可以提高4-5倍
- Capacity of dynamic RAM(DRAM) is the same as that of processors, increasing by 4 times every three years 动态随机访问存储器（DRAM）的容量和处理器一样，每三年增加4倍
- Fully utilizing the processing power of the processor is the key to improving computer performance 充分发挥处理器的处理能力，是提高计算机性能的关键

Microprocessor Speed 微处理器的速度

- The processor designers must come up with ever more elaborate techniques for feeding a constant stream of work to do 处理器设计人员必须想出更加精细的技术，为处理器提供源源不断的工作
  - Pipelining 流水线
  - Branch prediction 分支预测
  - Superscalar execution 超标量执行
  - Data flow analysis 数据流分析
  - Speculative execution 推测执行

Performance gap of CPU and memory CPU和内存的性能鸿沟

While CPU power has raced ahead at speed, other critical components have not kept up 虽然**CPU**的能力以极快的速度向前发展，但其他关键组件并没有跟上



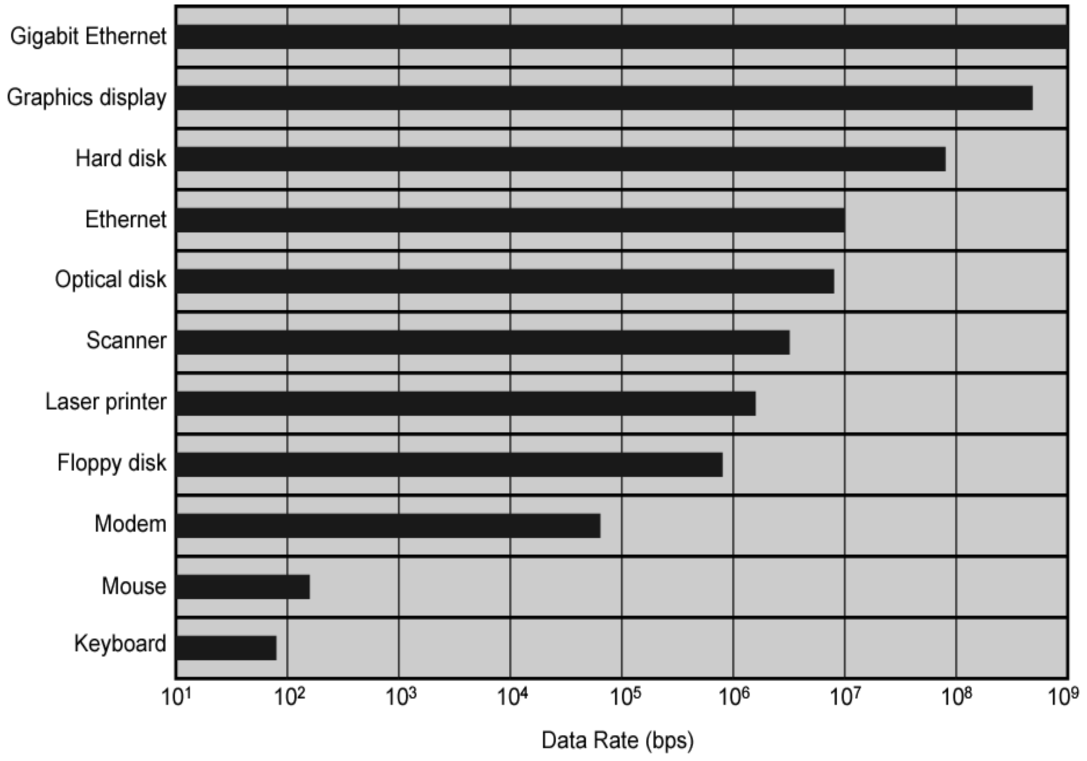
Performance of memory will seriously affect the performance of CPU!

内存的性能将会严重影响CPU的能力的发挥

Solutions about Memory 存储器解决方案

- Increase number of bits retrieved at one time 增加一次访问的位数
  - Make DRAM "wider" rather than "deeper" 使得DRAM更宽，而不是更深
- Change DRAM interface 改变DRAM的接口
  - Cache 高速缓存
- Reduce frequency of memory access 减少存储器访问频率
  - More complex cache and cache on chip 更复杂的cache和片上cache
- Increase interconnection bandwidth 提高互联的带宽
  - High speed buses 高速总线
  - Hierarchy of buses 多级总线

Typical I/O Device Data Rates 典型I/O设备的速率



- Same problem in I/O access. I/O存在相同的问题
- Lots of I/O device types I/O设备种类多
- Large data rate difference. 数据速率差异大

Solutions about I/O devices I/O设备解决方案

- Caching 缓存
- Buffering 缓冲
- Higher-speed interconnection buses 高速互联总线
- More elaborate bus structures 更复杂的总线结构
- Multiple-processor configurations 多处理器配置

Key factor is balance 关键因素是平衡

Key to solving the problem of computer performance is to balance, that is, to find the best combination scheme among various components 解决计算机性能问题的关键是平衡，也就是要在各个组成部件之间找到一个最佳的组合方案

- Processor components 处理器组件
- Main memory 主存
- I/O devices I/O设备
- Interconnection structures 互联结构

Process speed 处理器速度

Three approaches to achieving increased processor speed 三种提高处理器速度的方法

- Increase hardware speed of processor 提高处理器硬件速度
  - Fundamentally due to shrinking logic gate size 基本上依赖缩小逻辑门的尺寸
- Increase size and speed of caches 提高cache的大小和速度
  - Dedicating part of processor chip 芯片专用部分给cache
- Change processor organization and architecture 改变处理器的组织和架构
  - Increase effective speed of execution 提高执行的有效速度
  - Parallelism 并行

Problems with Clock Speed and Logic Density 时钟速度和逻辑密度问题

- Power 能耗
  - Power density increases with density of logic and clock speed 功率密度随逻辑密度和时钟速度的增加而增加
  - Dissipating heat 散热
- RC delay 阻容迟滞
  - Delay increases as RC product increases 电容和电阻的增加，产生了延迟
  - Wire interconnects thinner, increasing resistance 导线越细，电阻越大
  - Wires closer together, increasing capacitance 导线靠得越近，电容越大
- Memory latency 存储器延迟
  - Memory speeds lag processor speeds 存储器速度拖了处理器速度的后腿

Increasing cache capacity 增加Cache容量

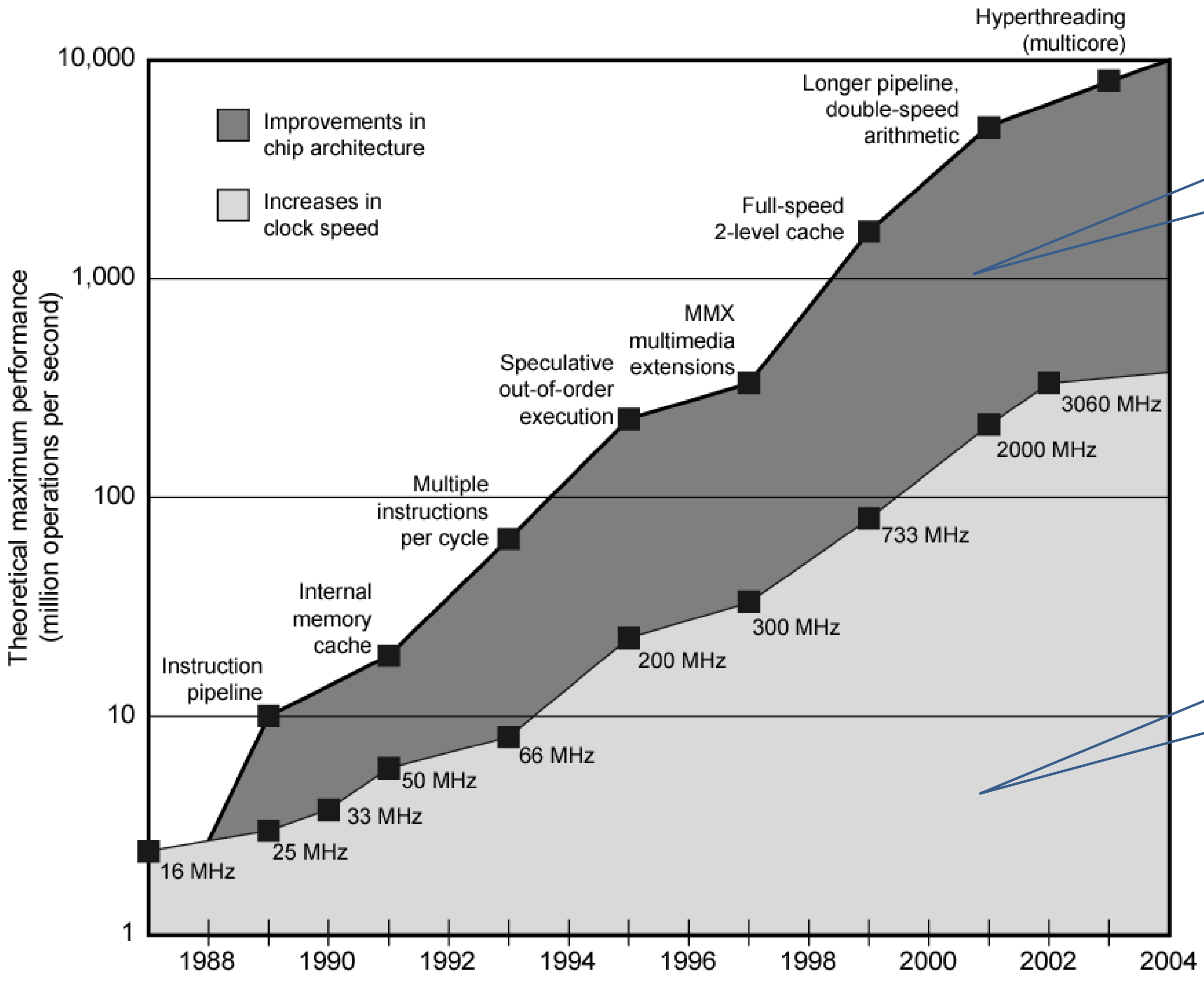
- Typically two or three levels of cache between processor and main memory 处理器和主内存之间通常有两到三级缓存
- Chip density increased 芯片密度提升
  - More cache memory on chip 更多的cache在片上
  - Faster cache access cache访问更快
- Pentium chip devoted about 10% of chip area to cache 奔腾芯片将大约10%的芯片面积用于缓存
- Pentium 4 devotes about 50% 奔腾4投入约50%（给cache）

More complex execution logic 更复杂的执行逻辑

Enable parallel execution of instructions 指令的并行执行

- Pipeline works like assembly line 类似于装配的流水线
  - Different stages of execution of different instructions at same time along pipeline 沿流水线同时执行不同指令的不同阶段
- Superscalar allows multiple pipelines within single processor 超标量允许在单个处理器中使用多个流水线
  - Instructions that do not depend on one another can be executed in parallel 相互独立的指令可以并行执行

Intel microprocessor performance Intel处理器性能



芯片结构导致的CPU性能提升

时钟速度提升导致的CPU性能提升

Diminishing returns 收益递减

- Internal organization of processors complex 增加处理器内部组织的复杂度
  - Can get a great deal of parallelism 能够获得大量的并行性
  - Further significant increases likely to be relatively modest 进一步大幅度增长越来越难
- Benefits from cache are reaching limit 通过cache得到的收益达到极限
- Increasing clock rate runs into power dissipation problem 增加时钟频率会导致功耗问题
  - Some fundamental physical limits are being reached 一些基本的物理极限正在达到

New approach – multiple cores 新方法-多核

- Multiple processors on single chip 单个芯片上多个处理器核
  - Large shared cache 大的共享cache
- Within a processor, increase in performance proportional to square root of increase in complexity 在处理器中，性能的提高与复杂性增加的平方根成正比
- If software can use multiple processors, doubling number of processors almost doubles performance 如果软件可以使用多个处理器，那么处理器数量翻倍几乎会使性能翻倍

CISC and RISC 复杂指令集和简单指令集

- The Intel x86 represents the complex instruction set computers (CISC) x86代表着CISC，复杂指令集计算机
  - Personal computer 个人计算机
  - Mainframes 大型机
  - Supercomputers 超级计算机
- The ARM architecture represents the reduced instruction set computers (RISC) ARM结构代表着RISC，简单指令集计算机
  - SUN SPARC，MIPS
  - Apple Macintosh
  - embedded chip 嵌入式系统
  - Mobile phone 手机

General Purpose GPU 通用GPU

- GPU is a processing unit for parallel processing of graphic data GPU是对图形数据进行并行处理的处理单元
- GPUs can perform parallel operations on multiple sets of data and are often used in vector processors, not just for processing graphic data GPU可以对多组数据进行并行操作，经常用于向量处理器，不仅仅是处理图形数据
- Can be used for various applications that require repeated calculations, such as model training 可以用于各种需要重复计算的应用，例如模型训练
- GPGPU utilizes a graphics processor that processes graphics tasks to calculate general-purpose computing tasks that were originally handled by the central processing unit GPGPU利用处理图形任务的图形处理器来计算原本由中央处理器处理的通用计算任务
- Data processing performance far exceeds traditional CPUs 数据处理性能远大于传统的CPU

Outline

- Designing for Performance 性能设计
- Amdahl’s Law Amdahl 定律
- Basic Measures of Computer Performance 计算机性能基本测量

Background 背景

- Computers are a whole 计算机是一个整体
- The improvement of individual performance does not necessarily lead to an overall performance improvement, such as CPU 单个部分性能的提升并不一定带来整体性能的提升，例如CPU
- Parallel processing can improve performance, but is limited by the parallelism of software 并行处理可以提升性能，但受限于软件的并行性

Amdahl’s Law 阿姆达定律

- First proposed by Gene Amdahl 首先由阿姆达尔提出
- For program running on single processor and multi processors
  - Fraction *f* of code infinitely parallelizable with no scheduling overhead f表示代码中可无限并行执行的指令比例，无调度开销
  - Fraction (1-*f*) of code inherently serial 1-f表示需要串行执行的比例
  - T is total execution time for program on single processor T表示在单个处理器上执行的时间
  - N is number of processors that fully exploit parallel portions of code N是处理器数量，能够完全利用代码的并行部分

$$Speedup = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}} = \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{N}} = \frac{1}{(1-f)+\frac{f}{N}}$$

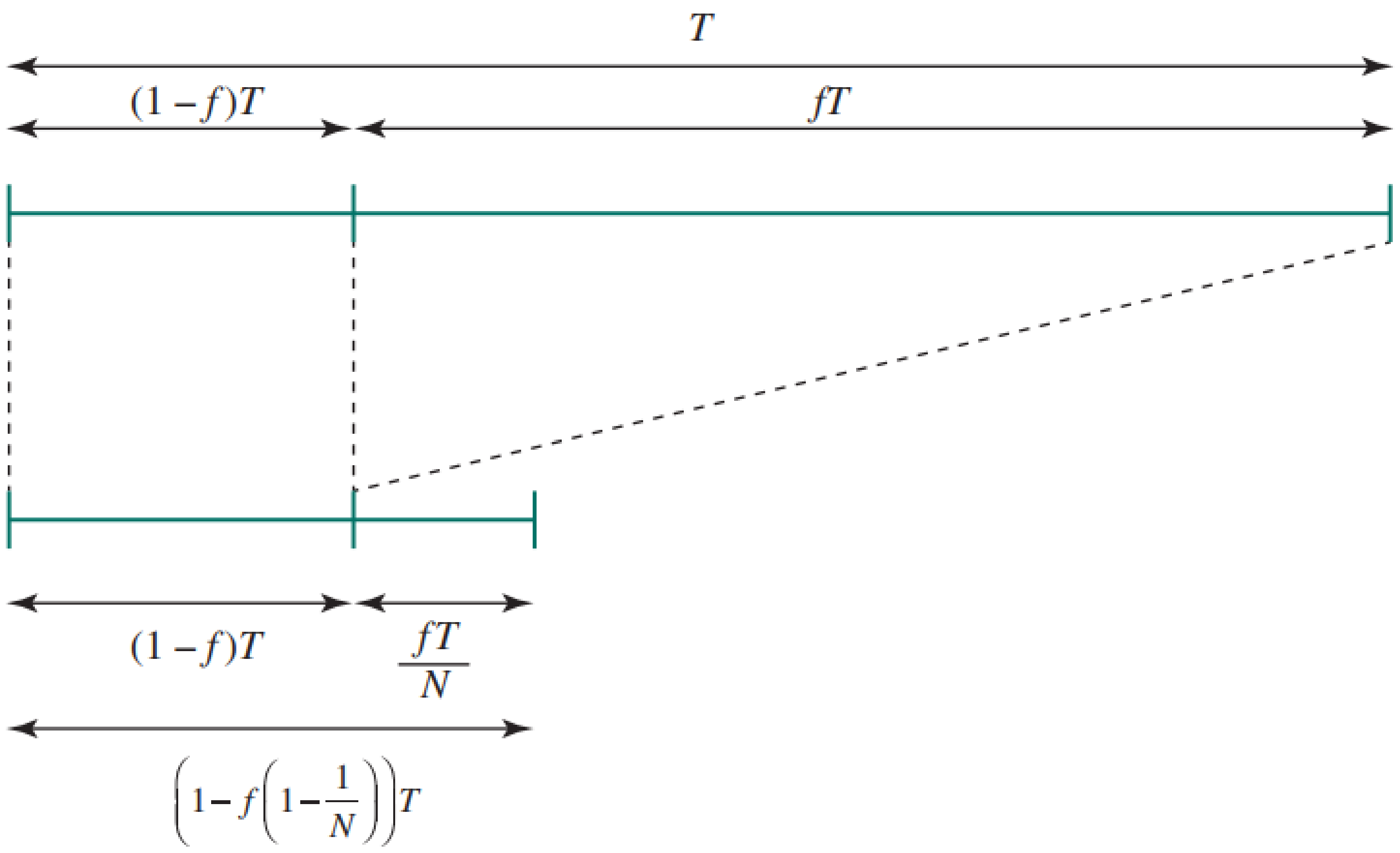
Conclusions of Amdahl’s Law 阿姆达定律的结论

$$Speedup = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}} = \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{N}} = \frac{1}{(1-f)+\frac{f}{N}}$$

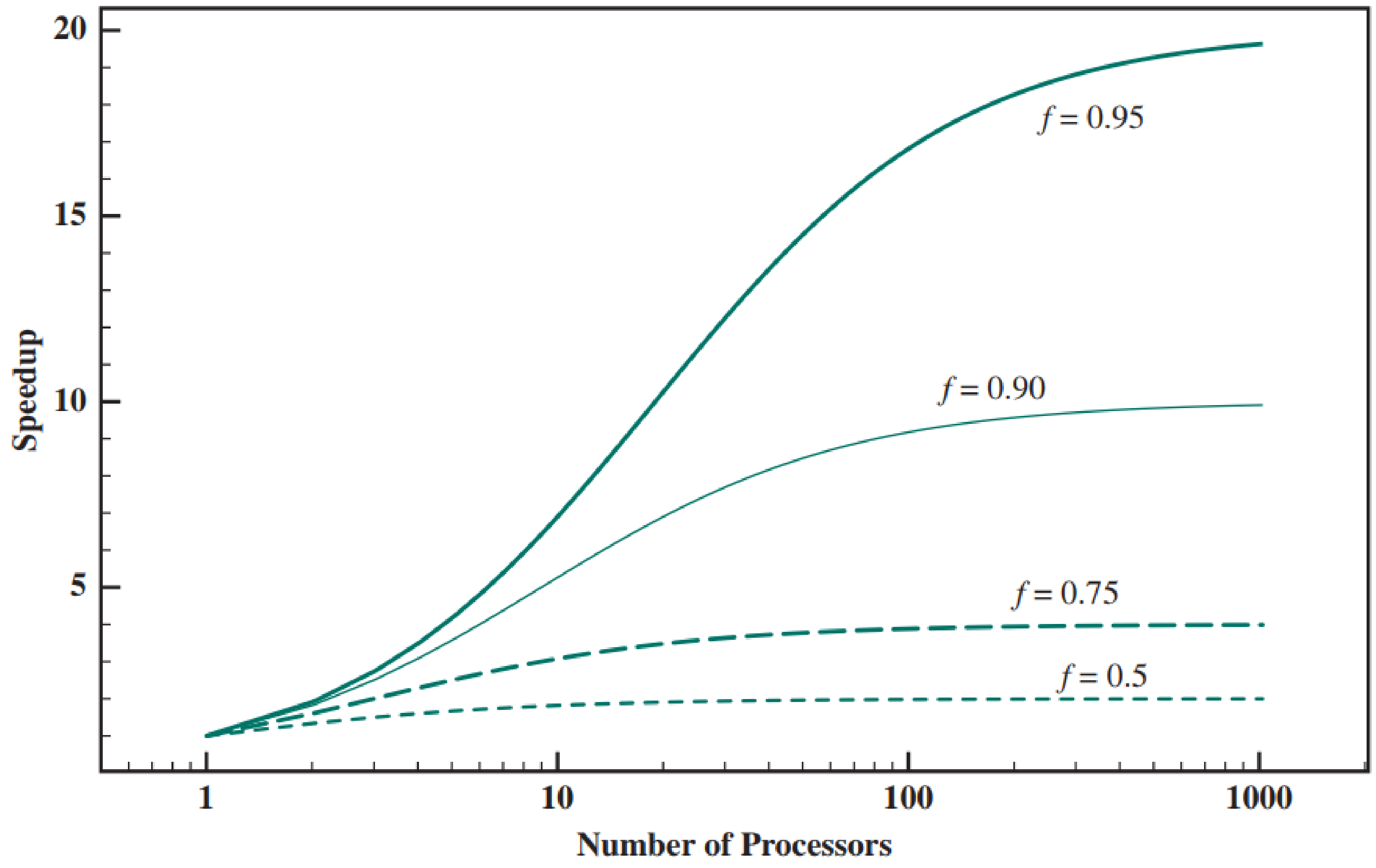
- f small, parallel processors has little effect f很小，并行处理器几乎没有效果
- N ->∞, speedup bound by 1/(1 – f) N无穷大，加速比接近1/(1 – f)
  - *Diminishing returns for using more processors 增加处理器的收益下降*

$$\text{Speedup} = \frac{1}{(1-f)+\frac{f}{SU_f}}$$

Amdahl’s Law 阿姆达定律



Amdahl’s Law 阿姆达定律



Amdahl’s Law for Multiprocessors 多处理器环境下的Amdahl 定律

Example1

- 描述阿姆达定律的含义和意义

$$Speedup = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}} = \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{N}} = \frac{1}{(1-f)+\frac{f}{N}}$$

Example2

- 根据下述条件，计算下述计算机的加速比
  - 假定程序中串行执行的部分为20%，对于一个8核处理器，理想的加速比是多少?
  - 一个4核处理器，如果希望加速比不低于3，那么程序中能够并行执行的指令至少需要占多少比例?

Example2

– 假定程序中串行执行的部分为20%，对于一个8核处理器，理想的加速比是多少?

  Speedup=1/(1-f+f/N)=1/(0.2+0.1)=3.33

– 一个4核处理器，如果希望加速比不低于3，那么程序中能够并行执行的指令至少需要占多少比例?

  3=1/ (1-f+f/N)

  f=8/9

Outline

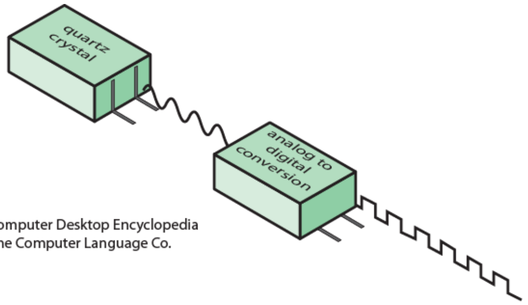
- Designing for Performance 性能设计
- Amdahl’ s Law Amdahl 定律
- Basic Measures of Computer Performance 计算机性能基本测量

About Performance Assessment 关于性能评估

- Evaluating the performance of different processor is difficult because of different architectures used 由于采用了不同的架构，因此客观评价处理的性能很困难
  - One processor is suitable for specific scenario 处理器针对特定的场景进行了优化
  - Instruction set, program language, compiler have an impact on the performance of processors. 指令集，编程语言，编译器对处理器的性能也有很大影响
- How to evaluate the performance of the processor? 怎么评价处理器的性能呢？

System clock 系统时钟

- System clock speed 系统时钟速度
  - Related to execution time of Instruction 和指令执行时间相关
  - Generated by the crystal oscillator，after analog-to-digital conversion, forms a digital voltage pulse signal 水晶振子产生后经过模数转换，形成数字脉冲信号
  - In Hz or multiples of 以Hz表示
  - Clock rate, clock cycle, clock tick, cycle time 时钟速率，时钟周期，时钟节拍，周期时间



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

Clock speed 时钟速度

- Operations need to be synchronised 操作需要同步
- Instruction execution in discrete steps 以离散步骤执行指令
  - Fetch, decode, load and store, arithmetic or logical 取指、解码、加载和存储，算术或逻辑运算
  - Usually require multiple clock cycles per instruction 通常每个指令需要多个时钟周期
- Instruction is not fully serial 指令执行并不是完全串行
- Pipelining gives simultaneous execution of instructions 流水线可以同时执行指令集
- So, clock speed is not the whole story 时钟速度并不是全部

Instruction execution rate 指令执行速度

- MIPS：Million Instruction per Second 每秒百万指令数
- CPI，Cycle per Instruction 指令平均周期数
- CPU constant frequency f CPU固定频率f
- constant cycle time τ=1/f 指令周期τ
- Instruction count Ic 指令数量Ic
- CPI

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

- MIPS

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

- MFLOPS

MFLOPS=程序中浮点运算的数量/(执行时间*10^6）

CPI : the average cycles per instruction CPI：每指令平均周期数

每指令平均周期数

第 i 种机器指令对应时钟周期数

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

程序的机器指令条数

第 i 种机器指令对应条数

Parameter in CPI CPI中的参数

- I_c: The number of machine instructions executed for that program, not the number of instructions in the object code of the program I_c:该程序执行的机器指令数，而不是程序目标代码中的指令数
- CPI_i be the number of cycles required for instruction type i. CPI_i: 第i种机器指令对应时钟周期数
- I_i be the number of executed instructions of type i for a given program I_i: 第i种机器指令对应条数

Program execute time and cycle time 程序执行时间和时钟周期

- The processor time T needed to execute a given program can be expressed as：执行给定程序所需的处理器时间T可以表示为：

$$\underbrace{T}_{\text{程序的执行时间}} = \underbrace{I_c}_{\text{程序的机器指令条数}} \times \underbrace{CPI}_{\text{每条指令平均周期数}} \times \overset{\text{时钟周期}}{\tau}$$

- A processors driven by a clock with a constant frequency f or, equivalently, a constant cycle time , where 处理器由一种固定频率的时钟来驱动，或者说由固定时钟周期来驱动

$$\overset{\text{时钟周期}}{\tau} = 1/\overset{\text{时钟频率}}{f}$$

MIPS - Millions of instructions per second MIPS：每秒百万指令数

每秒百万条指令

程序的机器指令条数

时钟频率

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

$$T = I_c \times CPI \times \tau$$

程序的执行时间

每条指令平均周期数

MFLOPS

- Millions of floating point instructions per second (MFLOPS) 每秒百万条浮点型指令数
- Heavily dependent on instruction set, compiler design, processor implementation, cache & memory hierarchy 严重依赖于指令集、编译器设计、处理器实现、缓存和内存层次结构

$$\text{MFLOPS rate} = \frac{\textit{Number of executed floating-point operations in a program}}{\textit{Execution time} \times 10^6}$$

Example3

- 根据下述信息，计算程序的CPI

指令类型	指令数量	该指令的时钟周期数量
整数运算	50000	1
浮点运算	20000	3
数据传送	25000	2
控制指令	5000	2

Example3

指令类型	指令数量	该指令的时钟周期数量
整数运算	50000	1
浮点运算	20000	3
数据传送	25000	2
控制指令	5000	2

- CPI= （50000*1+20000*3+25000*2+5000*2) /(50000+20000+25000+5000)=1.7

Example4

- 一台主频为100MHz的计算机，执行一段程序，程序中各种指令的数量和周期数如下表所示。计算这段程序的CPI以及MIPS。

指令类型	指令数量（M）	该指令的时钟周期数量
算术和逻辑运算	5	1
数据存取	8	2
控制指令	2	4
其它	1	3

Example4

指令类型	指令数量（M）	该指令的时钟周期数量
算术和逻辑运算	5	1
数据存取	8	2
控制指令	2	4
其它	1	3

- CPI=（5*1+8*2+2*4+1*3）/(5+8+2+1)=32/16=2
- MIPS=f/(CPI*10^6)=100* 10^6 /2* 10^6 =50

Disadvantages of MIPS MIPS的缺点

- MIPS only reflects the execution efficiency of machine MIPS仅反映了机器执行的执行效率
- Instructions are executed in parallel 指令是并行执行的
- Different machines have optimized different types of instruction sequences 不同机器对不同类型的指令序列进行了优化
- MIPS alone cannot be used to evaluate processor performance 不能仅用MIPS来评价处理器的性能
- Benchmarks 基准程序

Benchmarks 基准测试程序

Benchmarks：Programs designed to test performance

为测试性能而设计的程序

- Written in high level language 用高级语言编写
  - Portable 便于携带
- Represents style of task 代表任务的类型
  - Systems, numerical, commercial 系统程序，数值计算、商业
- Easily measured 容易测量
- Widely distributed 广泛发布

Benchmarks 基准测试程序

- System Performance Evaluation Corporation 系统性能评估公司（SPEC）
- Most famous：SPEC 2006
  - Suitable for testing applications of computing intensive 适合于测试计算密集型应用
  - 17 floating-point programs written in C, C + + and FORTRAN 17个用C，C++，Fortran写的浮点运算程序
  - 12 integer programs written in C and C ++ 12个用C和C++写的整数运算程序
  - total code of more than 3 million lines 代码量超过300万行

Result process and analysis 结果的处理和分析

- Run several different test programs, and then arithmetically process the test results to obtain an average measurement 运行多个不同的测试程序，并对测试结果进行算术处理，得到平均的测量
- Average execution speed 平均执行速度

$R_A=\frac{1}{m}\sum_{i=1}^{m} Ri$ Ri：第i个基准程序的高级语言指令执行速度

- Harmonic mean 调和平均值

$R_B=m/\sum_{i=1}^{m} 1/Ri$ 1/Ri：第i个基准程序的平均指令执行时间

Speed Metric 速度度量

- Ability of a computer to complete a single task 执行单个程序的能力
- $r_i = T_{refi}/T_{suti}$
- T_{refi}: execution time of benchmark program i on the reference system 基准程序的参考运行时间
- Tsuti : execution time of benchmark program i on the system under test 基准程序的实际执行时间
- Use geometric mean for more than one benchmark program 对于多个基准程序，用几何平均计算
- $r_G = (\prod_{i=1}^{n} ri)^{1/n}$

Example5

- 根据下述条件，用两种方法（平均执行速度和调和平均值）比较两个处理器的性能，并分析其中的意义

基准程序	处理器1的MIPS	处理器2的MIPS
基准程序1	2	3
基准程序2	3	4
基准程序3	5	3

Example5

基准程序	处理器1的MIPS	处理器2的MIPS
基准程序1	2	3
基准程序2	3	4
基准程序3	5	3

- 平均执行速度：处理器1的平均执行速度为（2+3+5）/3=10/3，处理器2的平均执行速度为（3+4+3）/3=10/3。从这个角度看，两个的性能一样
- 调和平均值：处理器1的调和平均值为90/31，处理器2的调和平均值为36/11，这样处理器2的性能更好

Summary and Question

- 小结
  - 计算机的性能设计要素
  - 计算机性能评估的方法
- 问题
  - 问题1：直接影响计算机性能的因素包括哪些
  - 问题2：阿姆达定律的核心意义是什么？

Summary and Question

- 直接影响计算机性能的因素有哪些?
  - 处理器的运行速度
  - 芯片内组织和架构
  - 各部件之间的平衡
- 阿姆达定律的含义
  - 程序的并行执行能力，是影响多核处理器加速比的关键因素

Assignments

- Review Questions:
  - 2.1， 2.2，2.4，2.6
- Problems
  - 2.1，2.2， 2.5，2.7

谢谢大家！