

第1 - 2章

定量设计与分析基础

1

本章主题

- 1.1 引言
- 1.2 计算机的类别
- 1.3 定义计算机体系结构以及计算机设计的任务是什么？
- 1.4 技术趋势
- 1.5 集成电路的功率趋势
- 1.6 成本趋势
- 1.7 可靠性
- 1.8 性能的测量、报告与总结
- 1.9 计算机设计的定量原则
- 1.10 综合考量

5个计算市场

特性	PMD	桌面端	服务器	集群/仓库级计算机	物联网/嵌入式
系统价格	100美元 - 1000美元	\$300-\$2500	\$5000-\$1000万美元	\$100000-2亿美元	\$10-\$100,000
.....的价格 微处理器模块	10 - 100美元	每个处理器50 - 500美元	\$200-\$2,000 每个处理器	50 - 250美元	\$0.01-\$100 每个处理器
关键系统设计问题	成本、能源、媒体性能响应能力	性价比、能源、图形性能	吞吐量、可用性、可扩展性、能源	性价比、吞吐量、能源比例性	价格、能源、特定应用性能

3

个人移动设备

- 具备多媒体用户界面的无线设备，例如智能手机、平板电脑等
- 应用程序 > 基于网络、面向媒体 > 响应性、可预测性
- 要求：

桌面计算

- 从金额上看，第一个且至今仍是最大的市场是桌面计算。
(桌面的笔记本电脑 > 50% 市场)

□ 要求：

- > 优化的性价比
- > 高性能且低成本的微处理器首先出现在桌面系统中

□ 新挑战：

- > 以网络为中心的交互式应用程序
- > 如何评估性能？

5

服务器

- 服务器在提供更大规模、更可靠的文件和计算服务方面的作用日益凸显。

△ 要求：

- > 首先，可用性至关重要。
- > 服务器系统的第二个关键特性是强调可扩展性。 → 内存、存储和I/O带宽至关重要。
- > 最后，服务器旨在实现高效的吞吐量。

6

停机成本

应用程序的成本 每小时停机成本	1% 每年87.6小时	0.5% 每年43.8小时	0.1% 每年8.8小时
经纪业务	645万美元	5.65亿美元	2.83亿美元	5650万美元
信用卡授权	260万美元	2.28亿美元	1.14亿美元	2280万美元
包裹运输服务	\$150,000	1300万美元	660万美元	130万美元
家庭购物频道	\$113,000	990万美元	490万美元	100万美元
目录销售中心	\$90,000	790万美元	390万美元	\$800,000
航空公司预订中心	\$89,000	790万美元	390万美元	\$800,000
蜂窝服务激活	\$41,000	360万美元	180万美元	\$400,000
在线网络费用	\$25,000	2200000美元	1100000美元	\$200,000
自动取款机服务费	\$14,000	1200000美元	\$600,000	\$100,000

集群/仓库级计算机

- 软件即服务（搜索、社交网络、视频分享、多人游戏、在线购物）
- └ JWS—数万台服务器协同工作
- 要求：
 - > 性价比、功耗（节省10%可节省700万美元）
 - > 可用性至关重要
 - > 可扩展性（注意与服务器的区别）
 - └ (注意与超级计算机的区别)

9

嵌入式系统

嵌入式系统



10

其他分类

- └ 计算机市场中增长最快的部分，其处理能力和成本范围最广。
- 要求
 - > 实时性能（软实时和硬实时）
 - > 严格的资源限制
 - > 有限的内存大小、较低的功耗.....
- └ 处理器核心与专用电路的结合使用。
 - > 数字信号处理器、移动计算、手机、数字电视

- 量子计算机与化学计算机，标量处理器与向量处理器
- └ 非统一内存访问（NUMA）/UMA寄存器机与堆栈机与累加器机哈佛架构与冯·诺伊曼架构/非冯·诺伊曼架构（类脑芯片：达尔文2）

□ 精简指令集计算机（RISC）与复杂指令集计算机（CISC）

M蜂窝架构

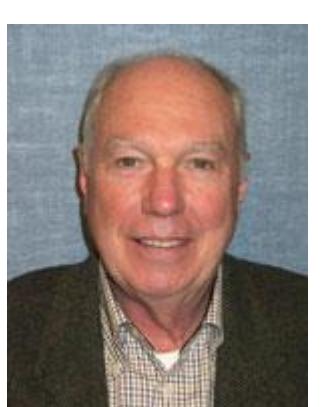
11

独眼巨人64（即蓝色基因/C）



计算机的类别

- 弗林分类法：一种基于指令流和数据流数量对计算机体系结构进行的分类 - 单指令单数据 (SISD)



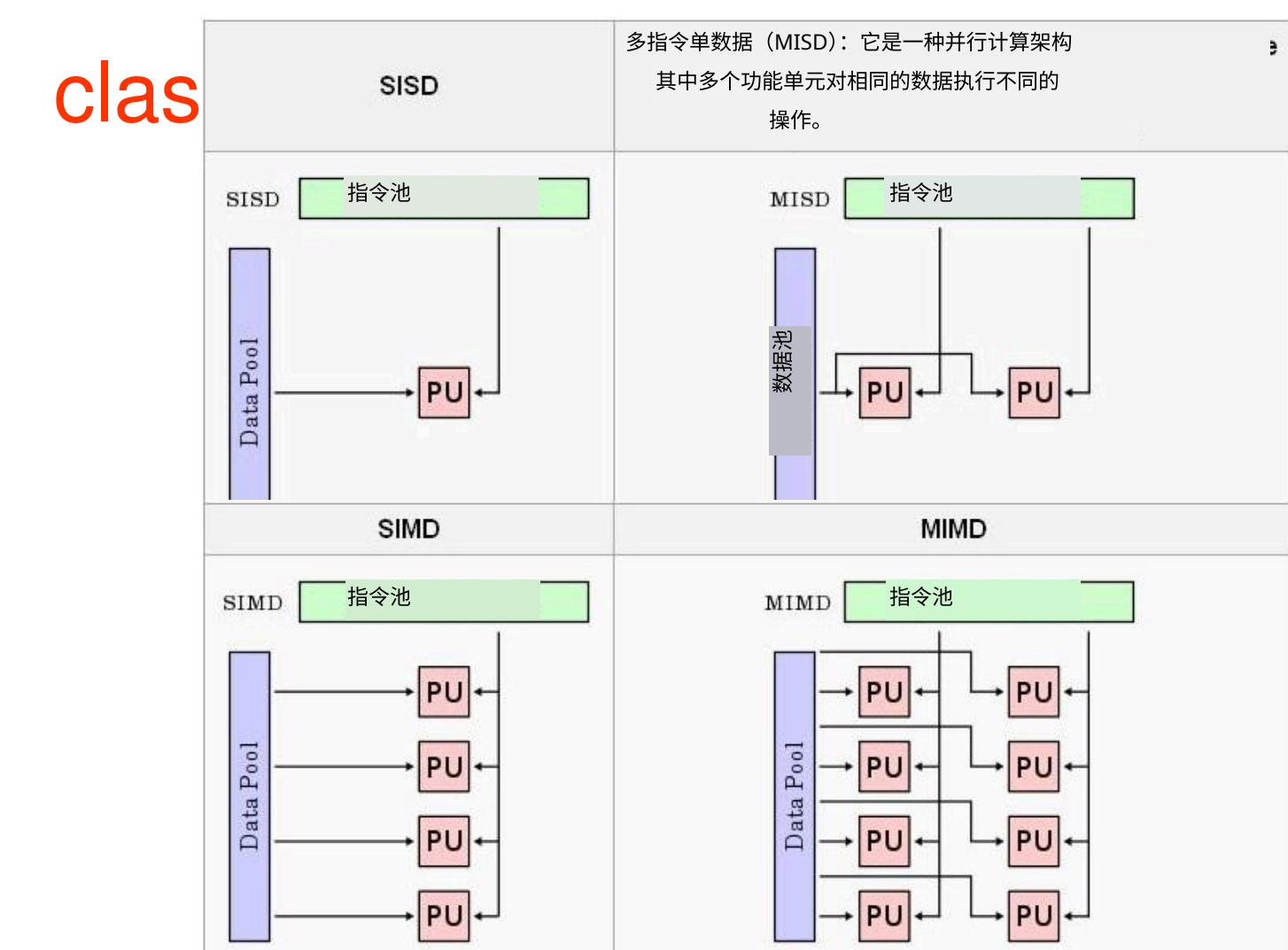
- 单处理器
- 多指令单数据 (MISD) - ??? - 单指令多数据 (SIMD) - 示例：伊利阿克IV型计算机、CM-2 » 简单的编程模型 » 低开销 » 灵活性 » 全定制 - 多指令多数据 (MIMD) - 示例：SPARCCenter、T3D » 灵活性高 » 使用现成的微处理器

12

并行性和并行体系结构的类别

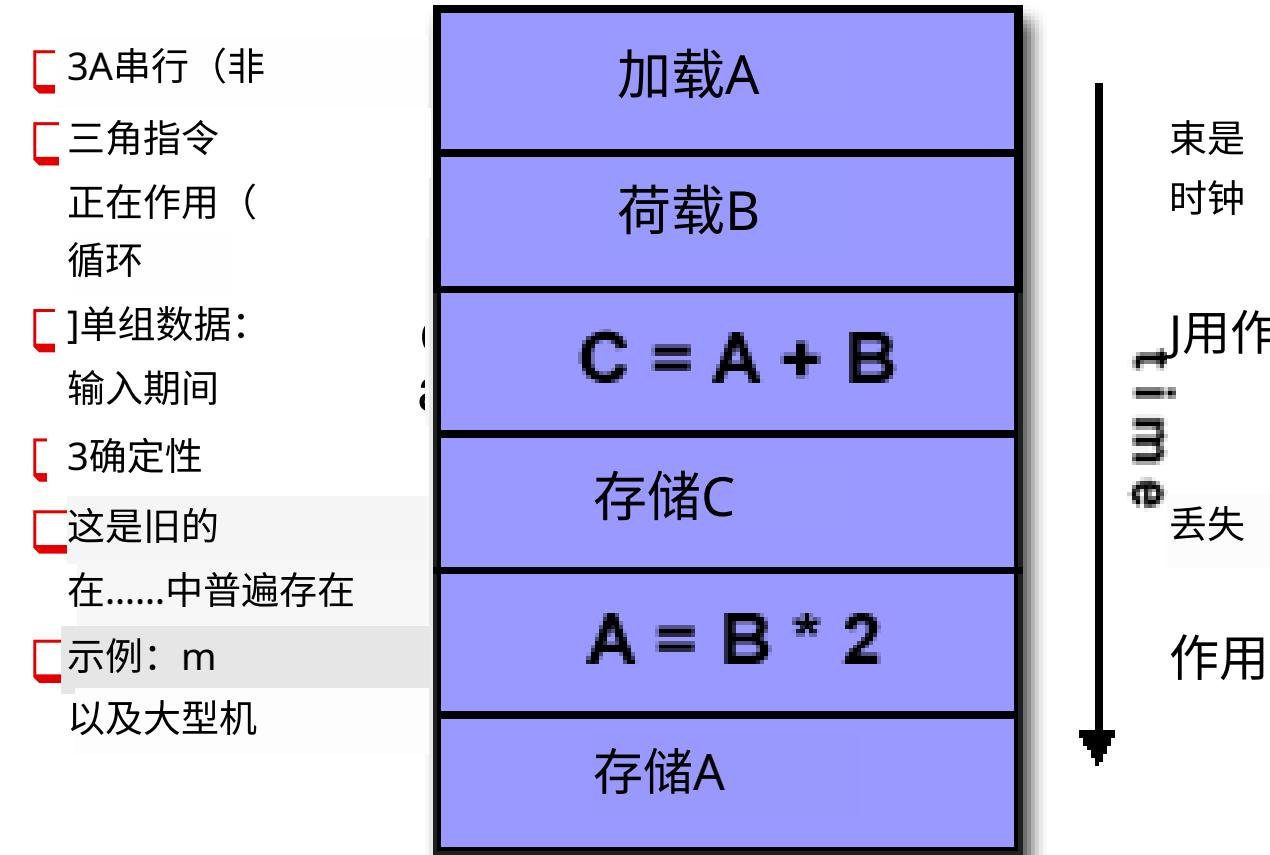
- 应用程序中的两种并行性
 - 数据级并行 (DLP)
 - 任务级并行 (TLP)
- 硬件以4种主要方式利用
 - > 指令级并行
 - > 向量架构与图形处理器
 - 线程级并行
 - > 请求级并行

14



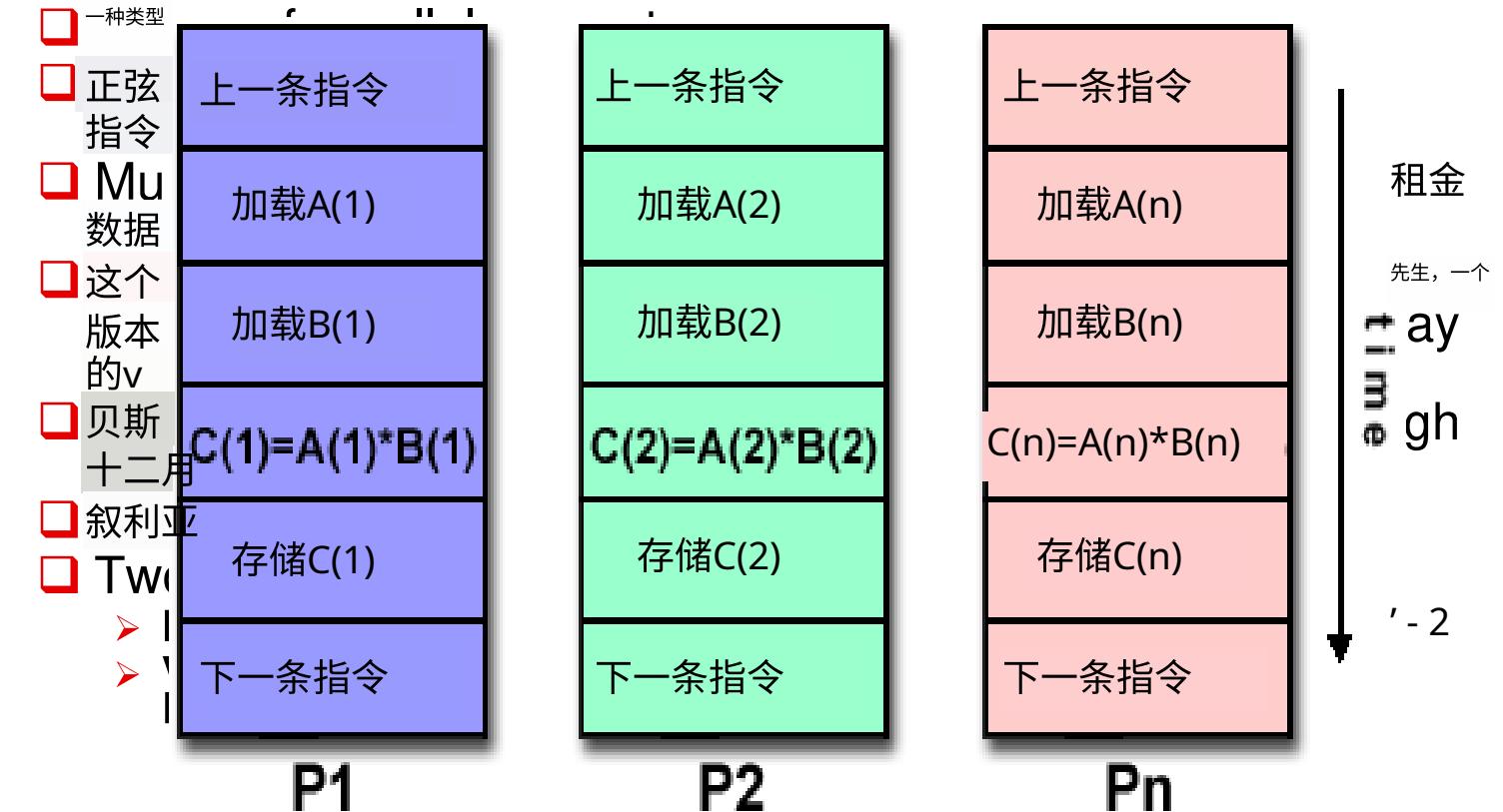
16

SISD



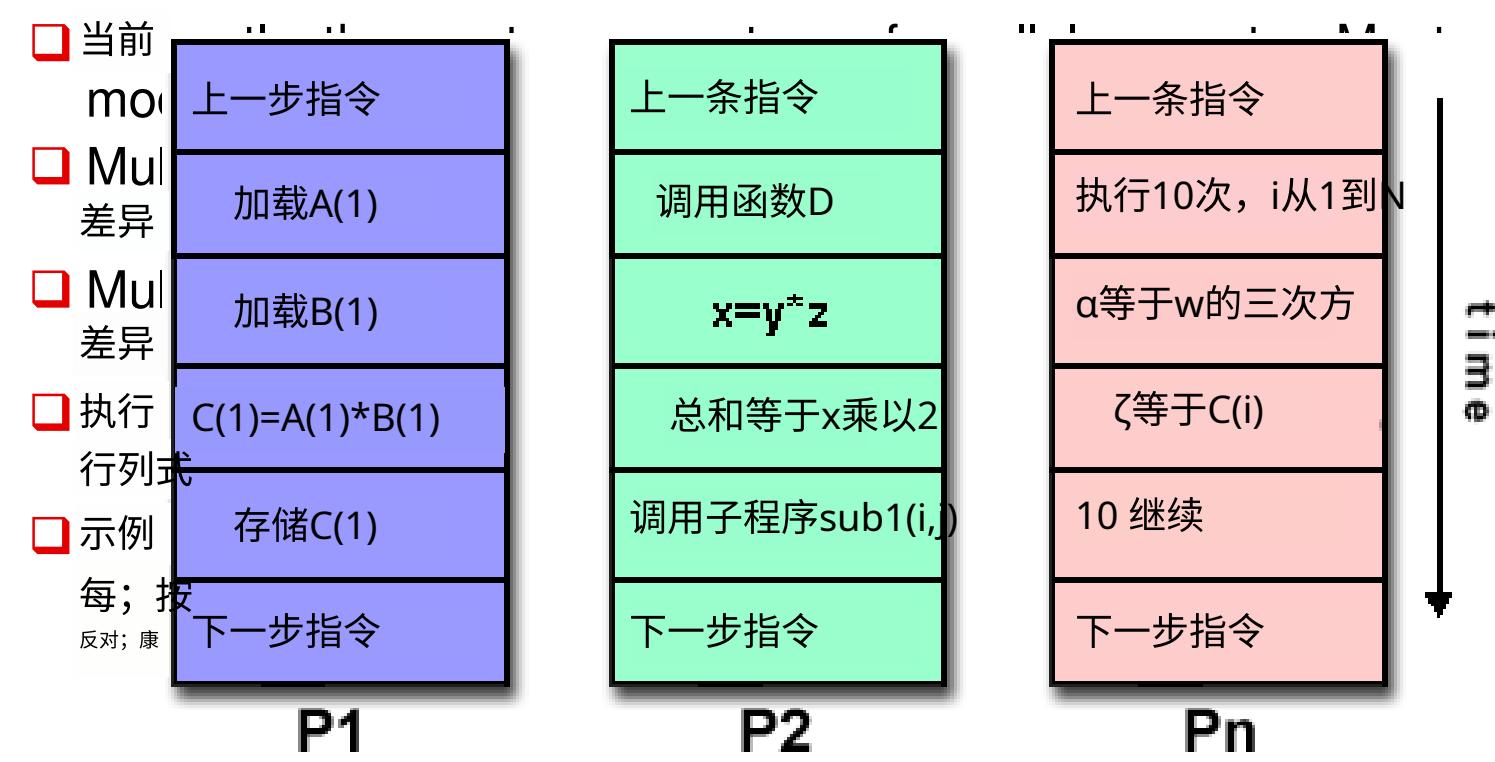
17

SIMD



18

MIMD



19

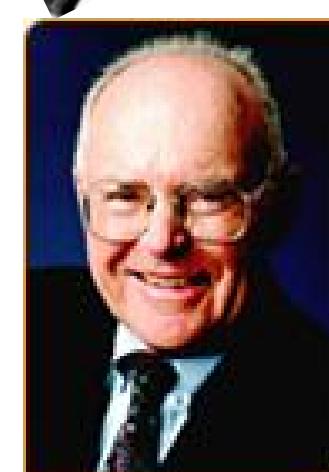
技术趋势

□ 摩尔定律

- 1965年, 他预测行业能够在计算机芯片上放置的组件数量每都会翻一番。1975年, 他将预测更新为每两年翻一番。它已成为
- 半导体行业在降低成本的同时提供更强大芯片的指导原则
- 电子产品的成本。
- 晶体管密度 $2x$ 每18 - 24个月提升 (每年1.4 - 1.5倍)
- 3 动态随机存取存储器 (DRAM) 密度
- 1977 - 1997年: 每年1.46倍
- 1997 - 2017年: 每年1.34倍
- 2017 - 2022年: 每年1.1倍



戈登·摩尔



21

技术趋势

设计师通常会为下一代技术进行设计。

□ 集成电路逻辑技术

- 晶体管密度: 每年增长35% (每4年增长4倍)
- 芯片尺寸: 每年缩小10% - 20%
- 每颗芯片的晶体管数量: 每年增长40 - 55%, 或在10至24个月内翻一番

□ 半导体动态随机存取存储器

- 每块DRAM的容量: 每年增长25%-40% (每2 - 3年翻一番)
- 内存速度: 每年约 10%
- > 可能会停止并被取代

□ 半导体闪存 - 个人移动设备中的标准存储设备

- 每个闪存芯片的容量: 每年增长50 - 60% (每2年翻一番)
- 比DRAM便宜15 - 20倍

□ 磁盘技术

- 密度: 1990年前每年增长30%; 1990 - 1996年每年增长60%
- > 1996 - 2004年每年增长100%; 2004年后每年增长30%
- 容量: 约每年 60%

□ 网络

- > 带宽: 10Mb → 100Mb → 1Gb

10年 5年

23

20

戈登·摩尔谈摩尔定律

□ 摩尔的简要生平

<http://www.intel.com/pressroom/kits/bios/moore.htm>

□ 戈登·摩尔谈摩尔定律

<http://www.sichinamag.com/Article/html/2007-09/2007919032802.htm>

□ Video on conversation with Moore

<http://you.video.sina.com.cn/b/7076856-1282136212.html>

22

动态随机存取存储器 (DRAM) 速率的提升

AQA 版本	年份	动态随机存取存储器 (DRAM) 增长	对动态随机存取存储器 (DRAM) 容量影响的特征描述
1	1990	每年60%	每3年增长四倍
2	1996	每年60%	每3年增长四倍
3	2003	每年40%-60%	每3至4年增长四倍
4	2007	每年40%	每2年翻一番
5	2011	每年25% - 40%	每2 - 3年翻一番

24

导入说明

□经验法则

> 成本降低率 ~ 密度增长率

□技术门槛

> 技术持续改进，这种改进的影响可能是离散式的飞跃。

25

微处理器的性能里程碑

微处理器	16位 地址/总线, 微 编码	32位 地址, 总线, 微编码	五级流 水线, 片上指令与数据缓存、浮点运算单元	两路 超标量、64位 总线	乱序三发射 超标量	乱序超级流水 线、片上1.2 级缓存
产品	英特尔80286	英特尔80386	英特尔80486	英特尔奔腾	英特尔奔腾Pro	英特尔奔腾4
年份	1982	1985	1989	1993	1997	2001
芯片尺寸 (mm ²)	47	43	81	90	308	217
晶体管	134,000	275,000	1,200,000	3,100,000	5,500,000	4200万
引脚	68	132	168	273	387	423
延迟 (时钟周期)	6	5	5	5	10	22
总线宽度 (位)	16	32	32	64	64	64
时钟频率 (兆赫兹)	12.5	16	25	66	200	1500
带宽 (每秒百万条指令)	2	6	25	132	600	4500
延迟 (纳秒)	320	313	200	76	50	15

27

与大人物的错误预测

□“没有人会想在家里拥有一台电脑。”

-肯·奥尔森, 总裁
数字设备公司董事长兼创始人, 1977年

29

功率趋势

□随着设备尺寸缩小, 功率也带来了挑战>> 首款微处理器: 1/10瓦 --> 2GHz奔腾4: 135瓦

□挑战:

- > 功率分配
- > 散热
- > 防止热点

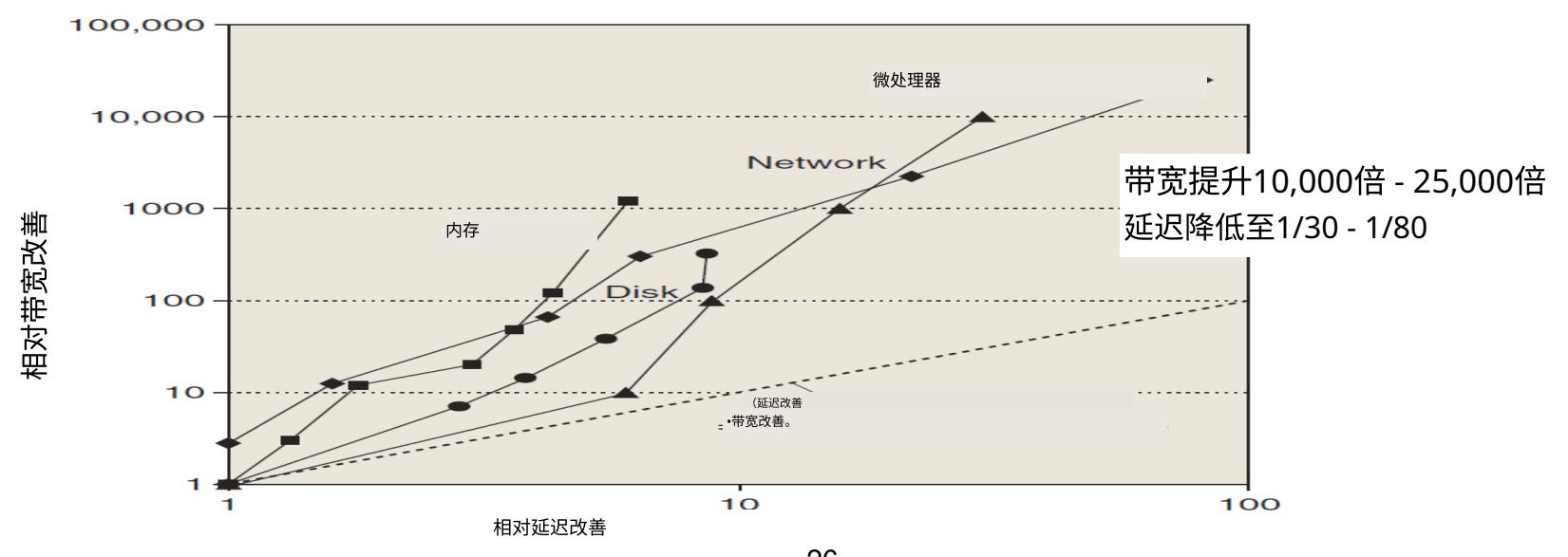
性能趋势: 带宽优先于延迟

□带宽/吞吐量: 给定时间内完成的总工作量 延迟/响应

□时间: 事件开始到完成之间的时间。

□经验法则

> 带宽增长率 ~ 延迟改善²



26

集成电路技术面临的挑战

□集成电路特性: 特征尺寸(特征尺寸)

> 10 微米, 2001年为 1971 → 0.18 微米

2006年为 >→ 0.09 微米, 2011年为 → 0.032 微米

> 7 nm 正在进行中。

> 经验法则: 晶体管性能随 $\frac{1}{d}$ 的减小呈线性提升。
特征尺寸。

□集成电路密度的提高既是机遇也是挑战:

> 导线的信号延迟与其电阻和电容的乘积成正比增加。
电阻和电容。

□镜面反射延迟——主要设计限制

28

章节主题

1.1 为什么要学习这门课程?

1.2 当前计算机市场中的计算机类别

1.3 定义计算机体系结构以及
计算机设计的任务是什么?

1.4 技术趋势

1.5 集成电路中的功率与能源趋势

1.6 成本趋势

1.7 可靠性

1.8 性能的测量、报告与总结

1.9 计算机设计的定量原则

1.10 综合运用

30

节能技术

□充分闲置——核心空闲时停止时钟。

□动态电压 - 频率调整 (DVFS)

> 几种时钟频率和电压

□针对典型情况进行设计

> 低功耗模式 (LPM) - 节省电量

> 处于低功耗模式时无法访问DRAM或磁盘

□超频

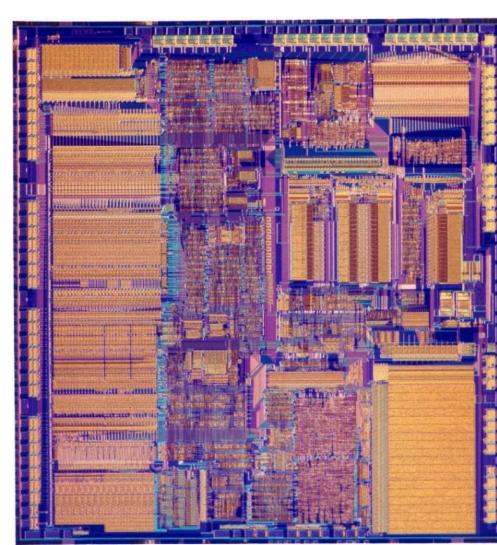
> 关闭除一个核心外的所有核心，并以更高的时钟频率运行。

□快速停机

> 使用更快但能效较低的处理器以实现
系统的其余部分进入睡眠模式

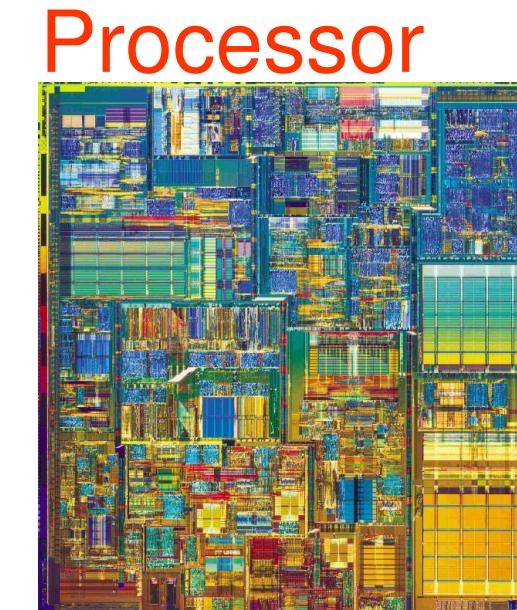
比较性能/功耗

386处理器



1986年5月
@16兆赫兹内核
275,000个 1.5μ 晶体管
~ 1.2 SPECint2000

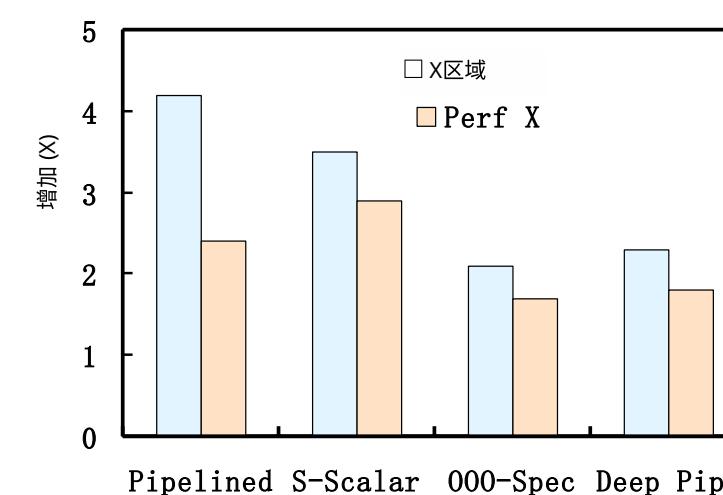
奔腾® 4



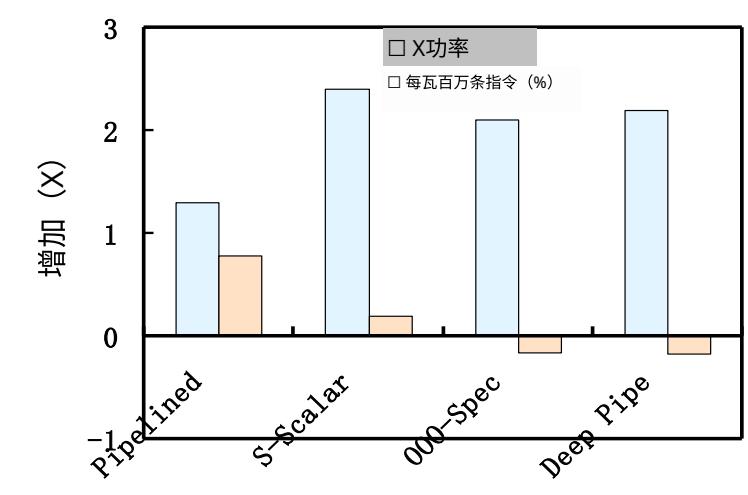
17年
200x
200x/11x
1000x
2003年8月27日
@3.2吉赫兹内核
5500万个 0.13μ 晶体管
1249 SPECint2000

33

功率效率



功率效率已下降



性能缩放与面积的0.5次方成正比

34

两个概念

□ 动态功耗：晶体管开关过程中的功耗。

$$> \text{功耗}_{\text{dynamic}} = 1/2 * \text{电容负载} * \text{电压}^2 * \text{开关频率}$$

$$> \text{Energy}_{\text{dynamic}} = \text{电容负载} * \text{电压}^2$$

□ 静态功耗：晶体管因漏电处于关断状态时的功耗
 $> \text{功耗}_{\text{static}} = \text{静态电流} * \text{电压}$

经验法则

□ 10% 电压降低会导致

频率降低 > 10%

功率降低 30%

性能降低不到 10% 经验法则

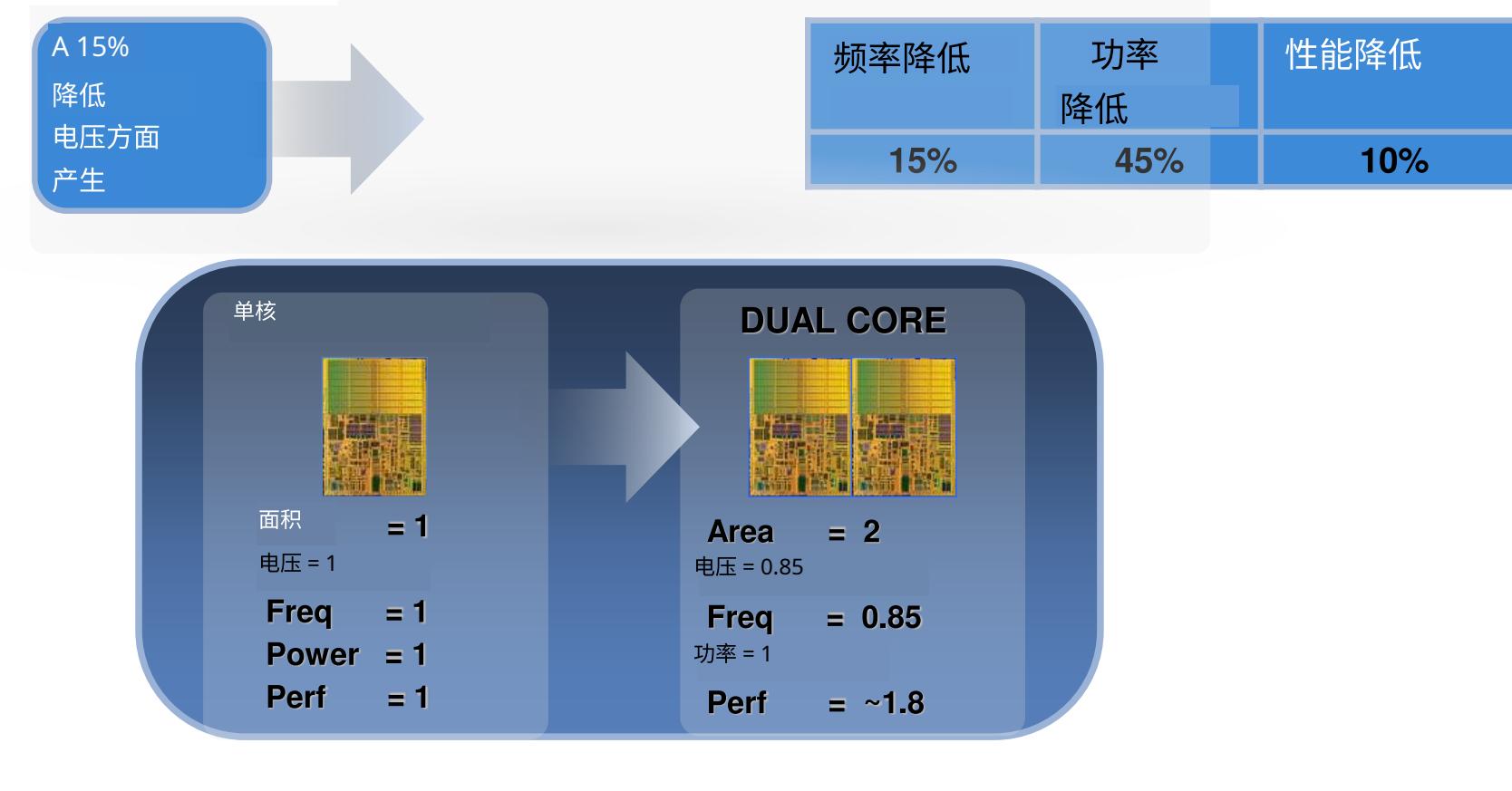
电压	频率	功率	性能
1%	1%	3%	0.66%

35

36

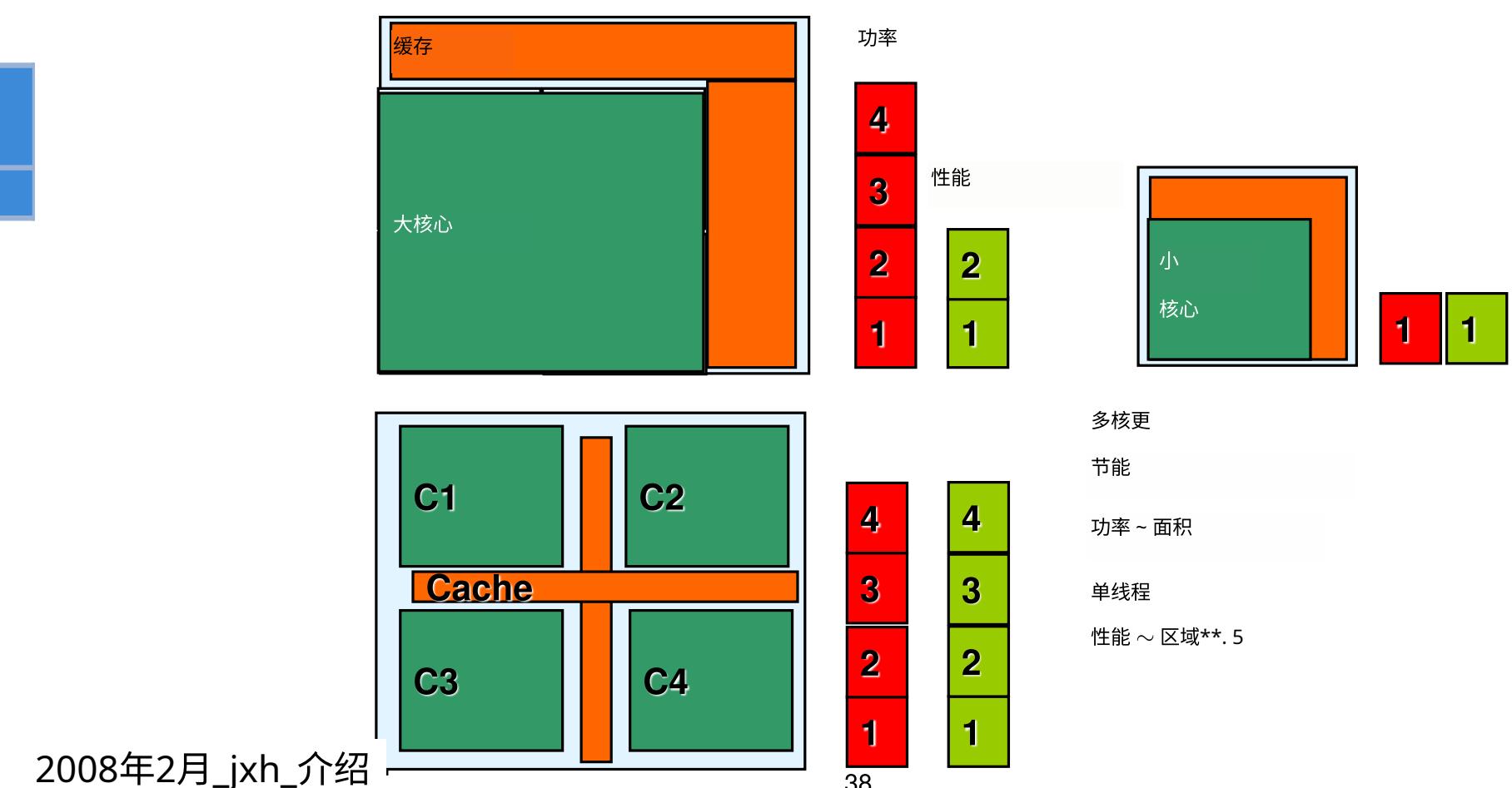
支持电压调节的双核处理器

经验法则



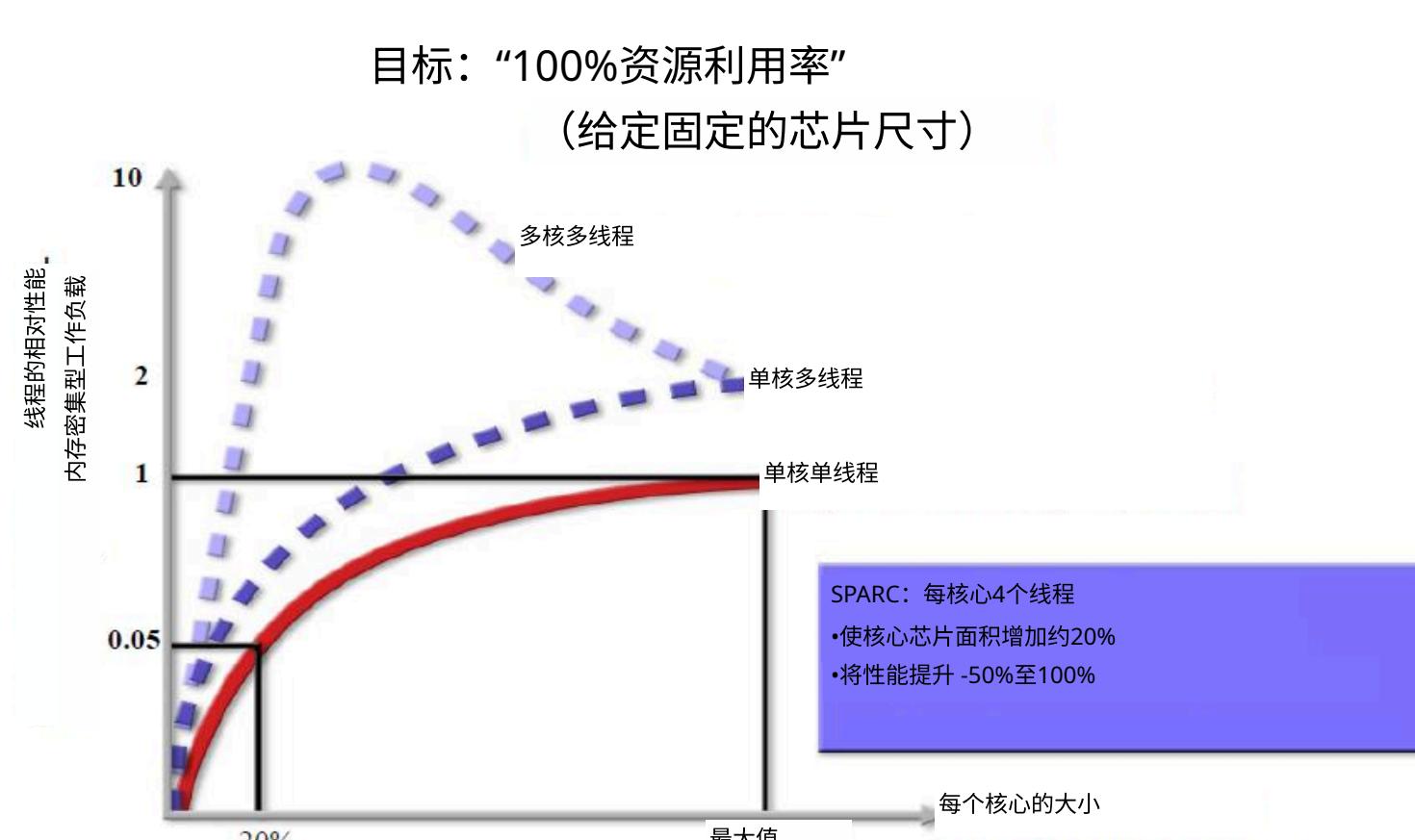
37

多核每瓦能提供更高性能



38

为何使用多核？



39

章节主题

1.1 为什么要选这门课程?

1.2 当前计算机市场中的计算机类别 1.3 定义计算机体系结构以及计算机设计的任务是什么?

1.4 技术趋势

1.5 集成电路的功耗趋势

1.6 成本趋势

1.7 可靠性

1.8 测量、报告和总结性能。

1.9 计算机设计的定量原则

1.10 综合应用

40

主要主题：降低成本

□ 成本趋势

> 了解组件的成本趋势对设计师来说很重要，因为我们

是为未来而设计！

□ 成本的影响因素：

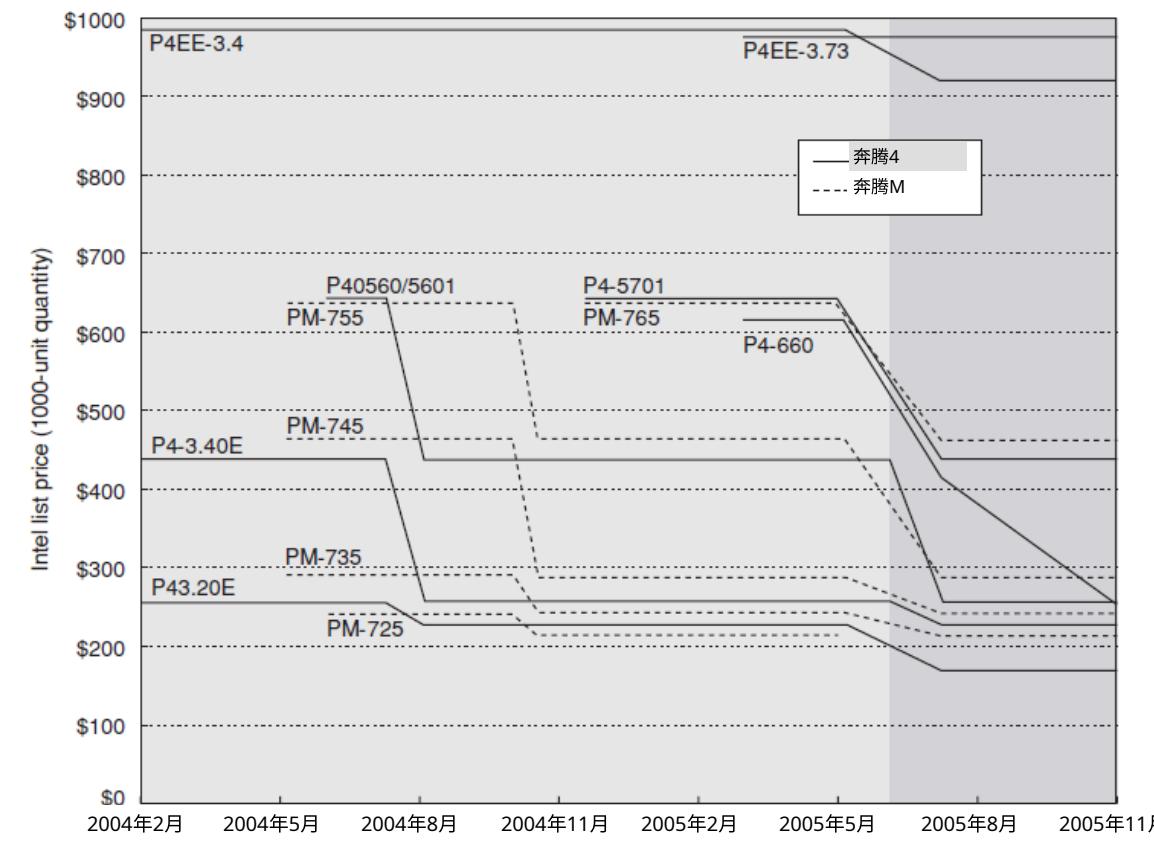
> 时间——在制造技术没有重大改进的情况下，组件价格会随时间下降

> 产量——由于制造效率的提高，产量增加会降低成本。

> 商品化——组件供应商之间的竞争将降低整体产品成本。

41

奔腾4和奔腾M的价格



43

42

经验法则

□ 时间：学习曲线——产量

> 产量翻倍，成本减半。（适用于芯片、电路板或系统）

○ 产量：

> 产量每翻一番，成本降低约 10%。

□ 商品：

» 供应商竞争

> 供应商竞争

> 销量增加，但利润有限。

44

集成电路的成本

$$\text{集成电路} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}} \text{ 的成本}$$

$$\text{裸片} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}} \text{ 的成本}$$

$$\text{每晶圆裸片数} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2} \times \text{Die area}}$$

$$\text{裸片良率} = \text{晶圆良率} \times \left(1 + \frac{\text{Defects per unit area} \times \text{Die area}}{\alpha}\right)^{-12}$$

45

成本与价格

组件成本

> 原材料成本。

□ 直接成本：

• 制造单个产品所产生的成本。使组件成本增加20%至

40%。I 毛利率（间接成本）：> 与单个产品无关的间接费用，即研发、营销、制造设备、税收等。

仅4% - 12%的收入用于研发

□ 平均销售价格 (ASP)：

> 组件成本 + 直接成本 + 间接成本。

□ 标价：

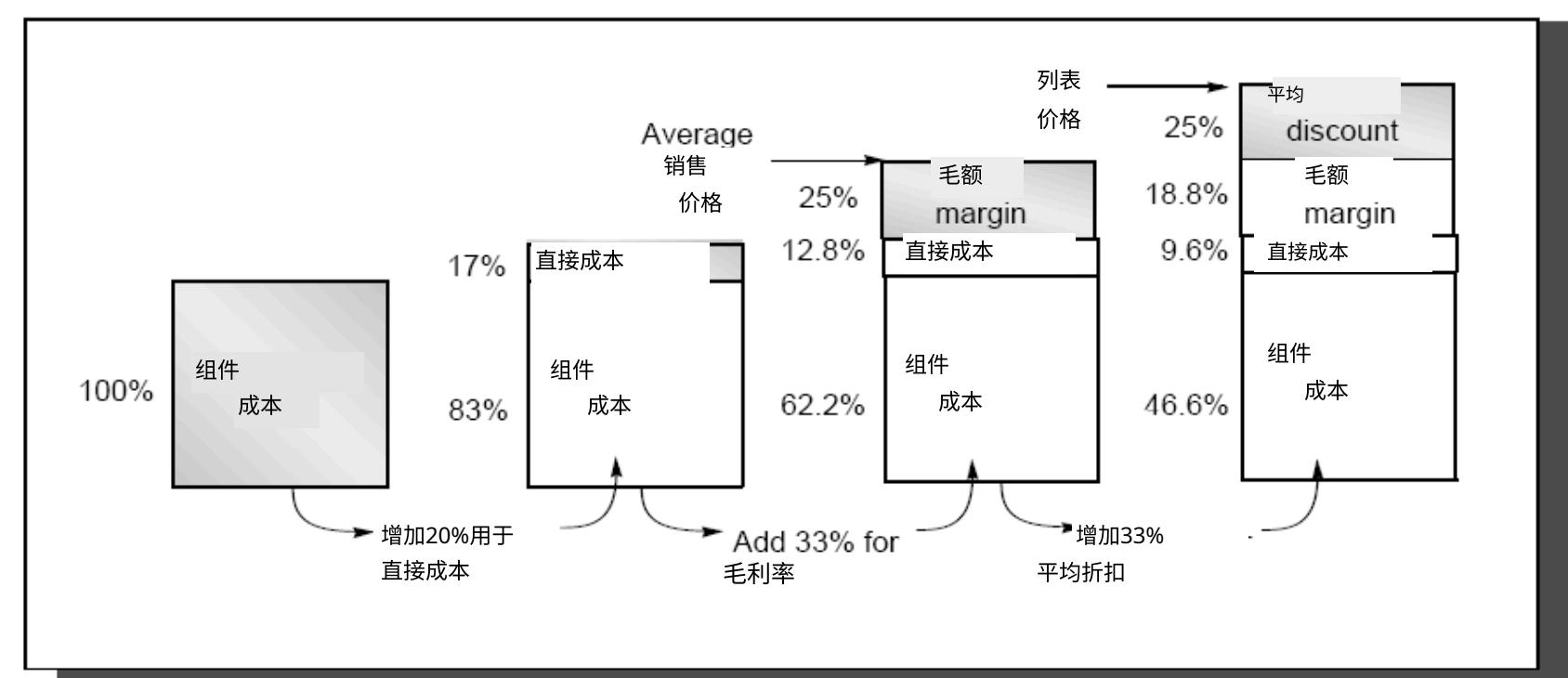
> 非平均销售价格 (ASP)。商店会在ASP基础上加价以获取利润。期望达到标价的50%至75%。

系统中的成本分配

系统	子系统	占总体的比例
机柜	金属薄板, 塑料	2%
	电源, 风扇	2%
	电缆, 螺母, 螺栓	1%
	运输箱, 手册	1%
	小计	6%
处理器板	处理器	22%
	动态随机存取存储器 (128 MB)	5%
	显卡	5%
	支持基本输入输出和网络功能的主板	5%
	小计	37%
输入/输出设备	键盘和鼠标	3%
	显示器	19%
	硬盘 (20GB)	9%
	DVD驱动器	6%
	小计	37%
软件	操作系统 + 基础办公套件	20%

46

一台售价1000美元的个人电脑的价格构成



47

48

成本与价格

这能让你了解设计决策将如何影响售价。

> 即成本增加 \$1,000 会使售价提高 3000 美元至 4000 美元。

此外，考虑销量与价格的关系：

> 一般来说，售出的电脑数量越少，价格就越高。

> 此外，销量下降会导致成本增加，进而进一步提高价格。

因此，成本的微小变化可能会导致价格意外大幅上涨。

49

章节主题

1.1 为什么要学习这门课程？

1.2 当前计算机市场中的计算机类别

1.3 定义计算机体系结构以及计算机设计的任务是什么？

1.4 技术趋势

1.5 集成电路功耗趋势

1.6 成本趋势

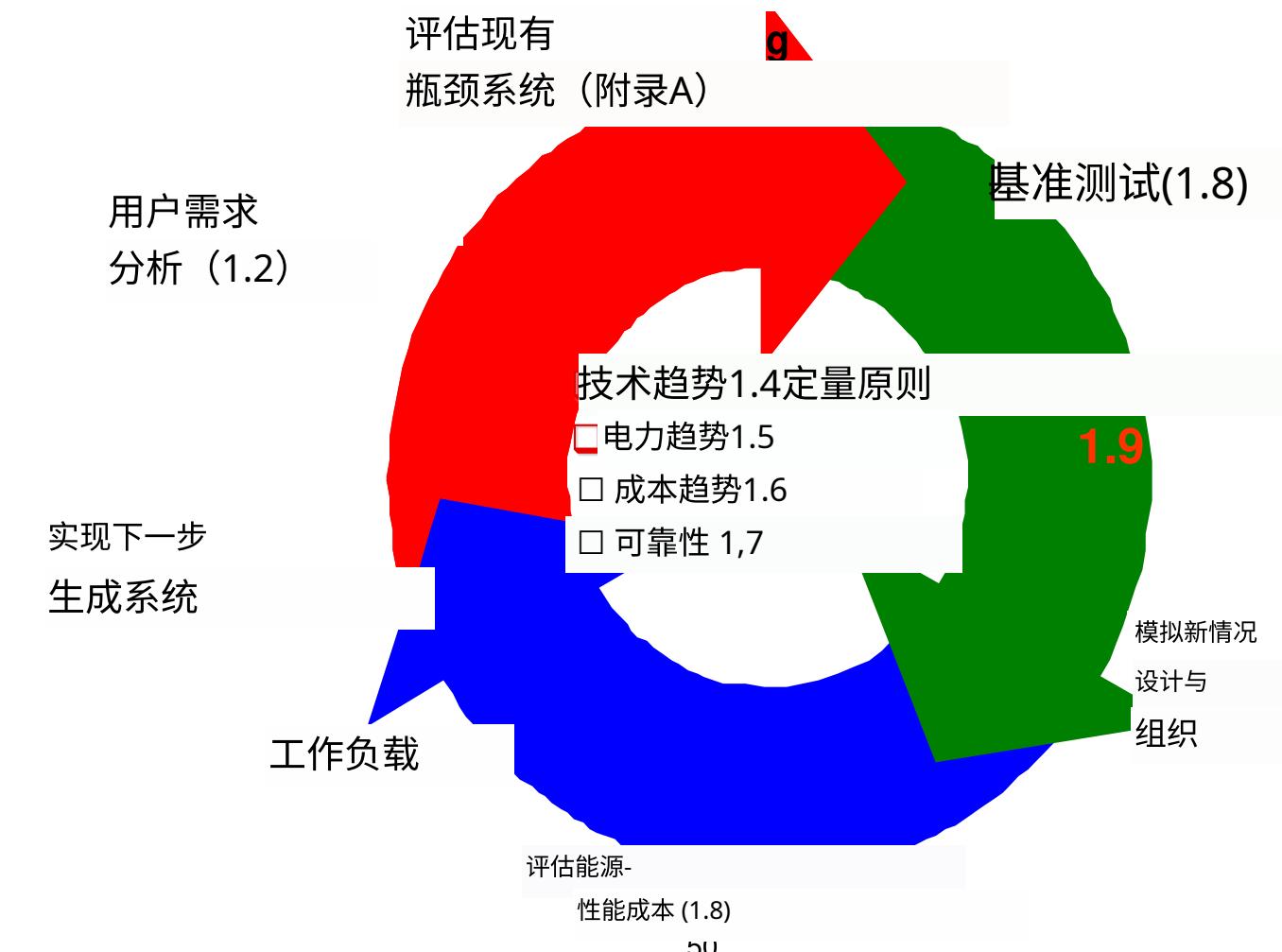
1.7 可靠性

1.8 性能的测量、报告与总结

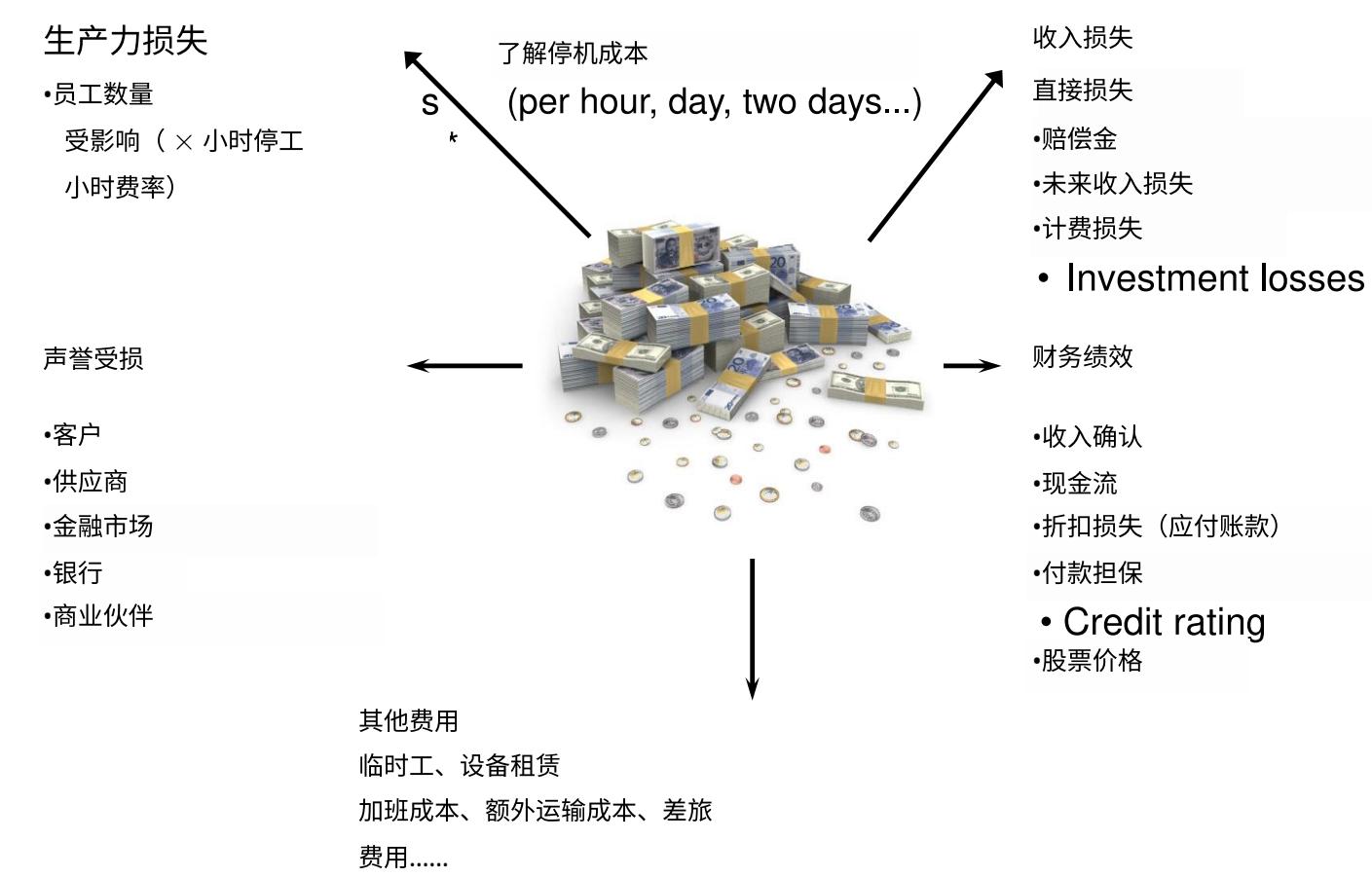
1.9 计算机设计的定量原则

1.10 综合考量

计算机设计工程生命周期

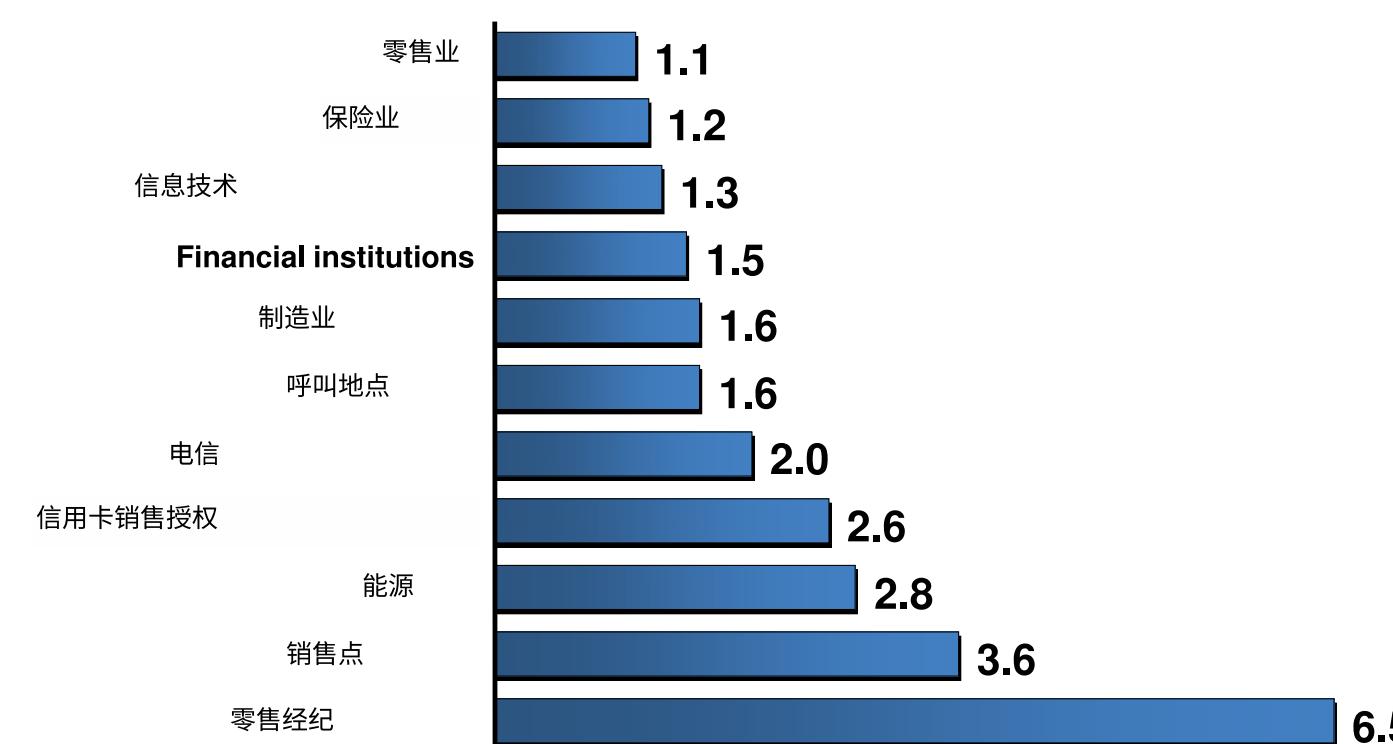


为什么要进行业务连续性管理？



51

为何需要高可靠性？每小时损失数百万美元的收入



Source: Meta Group, 2005

53

信息可用性（五个9）

%正常运行时间	%停机时间	每次停机时间 年	每周停机时间
98%	2%	7.3天	3小时22分钟
99%	1%	3.65天	1小时41分钟
99.8%	0.2%	17小时31分钟	20分钟10秒
99.9%	0.1%	8小时45分钟	10分钟5秒
99.99%	0.01%	52.5分钟	1分钟
99.999%	0.001%	5.25分钟	6秒
99.9999%	0.0001%	31.5秒	0.6秒

54

可靠性的演变 - 1

可靠性（可靠性）20世纪60年代

> ---容错与系统可靠性领域

定义

> 一般来说，可靠性（系统定义）是指个人或系统在常规环境以及恶劣或意外环境中执行并维持其功能的能力。

> 电气与电子工程师协会将其定义为“.....系统或组件在规定条件下和规定时间内执行所需功能的能力”。

电气与电子工程师协会

55

可靠性演变 - 2

可信性（可信性）

> 1985年，让 - 克洛德·拉普里

J.C. 拉普里。“可靠计算与容错：概念与术语”，载于1985年第15届IEEE国际容错计算研讨会论文集

> 国际可靠系统与网络会议

> 国际可靠分布式系统研讨会

> 国际软件可靠性工程研讨会。

56

可靠性的定义1

□ 计算机系统的可靠性是所提供的服务的质量，使得可以合理地依赖该服务。

> J.C. 拉普里。《可靠计算与容错：概念与术语》，载于《第15届IEEE国际容错计算研讨会论文集》，1985年

> 可靠性、可维护性、可用性、安全性是对于系统同一属性（即其可靠性）不同认知的定量度量。

> 一般概念

57

可靠性的定义2

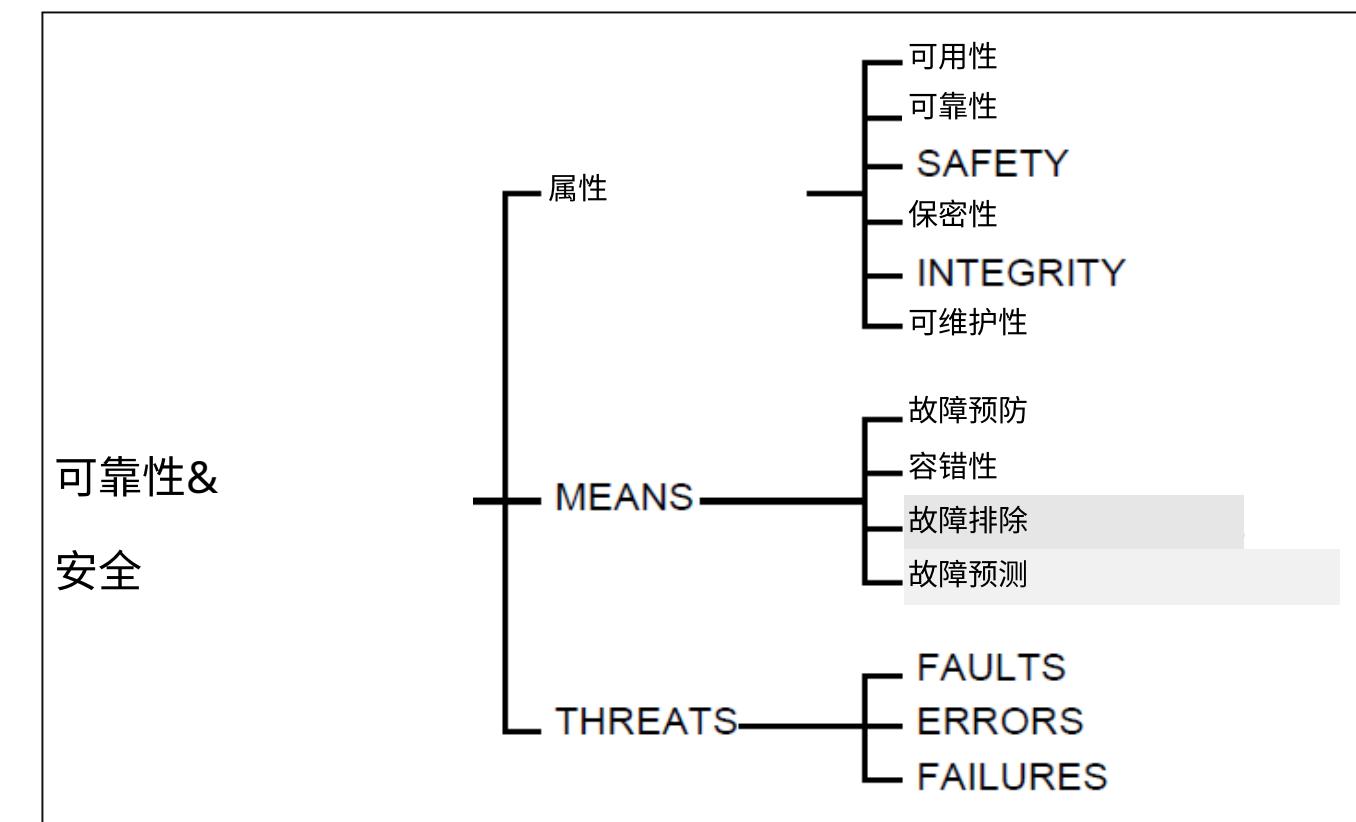


图1 - 可信性树

相关概念

□ J 可用性 - 准备好提供正确的服务

□ R 可靠性 - 正确服务的连续性

□ 安全性 - 对用户和环境无灾难性后果

□ 完整性 - 无不当的系统变更

□ J 可维护性 - 进行修改和修复的能力

□ 机密性，即无未经授权的信息披露

□ Security (安全保障) 是保密性、完整性和可用性的综合体。

59

可信性(可靠性)

□ 服务水平协议(SLA) / 服务水平目标(SLO)

➤ 服务完成情况

➤ 服务中断

□ 故障：

> $S_{accomplishment} \rightarrow S_{interruption}$

□ 恢复：

> $S_{interruption} \rightarrow S_{accomplishment}$

60

示例：磁盘系统

□ 假设：

> 10 磁盘，平均无故障工作时间1000000小时

> 1 SCSI控制器，平均无故障工作时间500000小时

> 1 电源，1 fan，两者平均无故障工作时间 (MTTF) 均为200000小时

> 1 小型计算机系统接口 (SCSI) 电缆，平均无故障工作时间 (MTTF) 为100000小时

□ 问题：整个系统的平均修复时间 (MTTR)

□ 答案：

> 故障率 $system = 10 * 1/1000,000 + 1/500,000 +$

$1/200,000 + 1/100,000$

$= 23000/1000,000,000$

> $MTTF_{system} = 1/Failure_{system} = 43500$ 小时 ~ 5 年

62

61应对故障

的方法

□ 冗余：

> 时间冗余：再次重复操作，查看是否仍存在错误。

> 资源冗余：让其他组件接管发生故障的组件。

可靠性

□ 与.....比较

> 可信计算

> 值得信赖的计算

> 可信度

章节主题

- 1.1 为什么要学习这门课程?
- 1.2 当前计算机市场中的计算机类别
- 1.3 定义计算机体系结构以及计算机设计的任务是什么?

- 1.4 技术趋势
- 1.5 集成电路的功耗趋势
- 1.6 成本趋势
- 1.7 可靠性
- 1.8 性能的测量、报告与总结
- 1.9 计算机设计的定量原则
- 1.10 综合考量65

不同的认知

- 性能对不同的人而言有不同的含义，因此对其评估需谨慎



- 不同的认知
- 用户和设计师之间的绩效评估标准不同

67

性能指标——CPU时间

- 测量CPU计算的时间（不等待I/O）
 - > 衡量设计师对CPU速度的认知
- CPU时间进一步分为：
 - > 用户时间 - 在用户模式下花费的时间
 - > 系统时间 - 在操作系统 (OS) 中花费的时间
- Unix时间命令将CPU时间报告为：
 - > 90.7u 12.9s 2:39 65%
 - > 90.7用户CPU秒（在用户程序中）
 - > 12.9系统CPU秒（在系统调用中，例如printf）
 - > 2分钟，39秒挂钟时间
 - > 65%的挂钟时间用于CPU运行

69

响应时间与吞吐量

- 如果你改善响应时间，通常也会提高吞吐量
 - > 用更快的版本替换计算机的处理器
- 你也可以在不改善响应时间的情况下提高吞吐量
 - > 向使用多处理器的系统中添加额外的处理器
 - > 用于处理不同的任务（例如处理航空公司预订系统）

性能测量与报告

机器

- > 执行时间（延迟）
- > 吞吐量
- > MIPS - 每秒百万条指令
- 使用程序集比较计算机
 - > 选择用于评估性能的程序——基准测试套件
 - > 不同的均值：算术均值、调和均值和几何均值

66

性能指标——响应时间

□ 时钟时间

- 启动程序并看时钟 -
 - > 当程序结束时，那就是总的时钟时间
 - > 也称为响应时间或经过时间或
 - > 衡量用户对系统速度的感知

□ 时钟时间存在的问题

- > 如果有多个程序在运行怎么办
同一台机器？
- > 如果程序要求用户输入怎么办？



68

性能指标 - 吞吐量

- 给定时间内完成的工作量
 - > 衡量管理员对系统性能的感知。
- 我们经常使用吞吐量来衡量
 - > 每天的代码行数
 - > 通过线路传输的每秒比特数
 - > 提供服务的网页数量
- 与延迟相反
 - > 生成 1 行代码所需的时间
 - > 通过线路发送 1 比特所需的时间
 - > 等待接收网页所花费的时间
- 通常，处理器性能仅以相对延迟来衡量
 - > 程序A的运行速度比程序B快10倍
 - > 但是，对于许多应用程序而言，吞吐量比延迟重要得多
- > 金融市场、政府统计数据（人口普查）

70

另一个行业指标：每秒百万条指令 (MIPS)

每秒百万条指令 - 每秒执行的百万条指令数

$$MIPS = \frac{\text{指令数量}}{\text{总运行时间}} - \frac{\text{X}}{1,000,000}$$

- 当比较具有相同指令集的两台机器 (A, B) 时，MIPS是一个合理的比较（有时……）
- 但是，每秒百万条指令 (MIPS) 可能是一个“无意义的性能指标……”

示例：每秒百万条指令（MIPS）可能毫无意义

机器A有一条用于执行平方根计算的特殊指令。执行该指令需要100个时钟周期。

机器B没有这条特殊指令——必须使用简单指令（例如加法、乘法、移位）以软件方式执行平方根计算，每条简单指令的执行时间为1个时钟周期。

机器A： $1/100 \text{ MIPS} = 0.01 \text{ MIPS}$

机器B：1 MIPS

73

性能指标总结

□ 响应（执行）时间

> 用户感知

> 系统性能

•唯一无可置疑的性能衡量标准

□ CPU时间

> 设计者感知

> CPU性能

□ 吞吐量

> 管理员感知

□ 每秒百万条指令

> 商家感知

75关于合成的内容

□合成基准测试：

> 尝试以相同方式“测试”系统的程序，以匹配大量程序的平均操作频率和操作数。

> 韦斯顿基准测试（Whetstone）和德莱斯顿基准测试（Dhrystone）。

与内核类似，但不是真正的程序！

> 编译器和硬件优化可以人为地提高这些基准测试的性能，但对实际程序则不然。

> 这些基准测试不会对优化给予奖励！

> $\text{SQRT}(\text{EXP}(x)) = \sqrt{e^x} = e^{x/2} = \text{EXP}(X/2)$

77

SPEC

□SPEC - 系统性能评估合作组织

> 由少数工作站供应商于1988年成立，他们意识到市场迫切需要现实、标准化的性能测试。

> 已发展成为成功的性能标准化机构，拥有40多家成员公司。

> <http://www.spec.org>

□ SPEC的理念

› SPEC的目标是确保市场拥有一套公平且有用的指标，以区分候选系统。

> 基本的SPEC方法是为基准测试者提供一套基于现有应用程序的标准化源代码套件

另一种观点：功耗与效率

□嵌入式系统的关键因素：

> 成本

> 物理尺寸

> 内存

> 功耗

□图1.27

> AMD ElanSC520

> AMD K6 - 2E

> IBM PowerPC 750CX

> NEC VR 5432

> NEC VR 4122

NEC VR 4122是大型的

因其最佳表现而获胜者

性能/瓦特

尽管它是性能第二低的处理器。

74

选择要评估性能的程序

□理想的性能评估：

> 运行其程序和操作系统命令的用户的随机样本。

□多种不同类型的基准测试

> 实际应用程序——科学与工程

•修改（或编写脚本）的应用程序——专注于特定功能

•内核 --- 关键程序片段

> 玩具程序 --- 小程序，通常测量值很小

> 合成程序 - 为代表程序的某些方面而创建（例如，指令类型的组合）

> 数据库 -- 一个独立的世界

> 真正重要的是你的应用程序的性能表现

76

性能基准测试笔记

□基准测试可以聚焦于系统的特定方面 > 浮点和整数

算术逻辑单元、内存系统、输入输出、操作系统

•通用基准测试可能会产生误导，因为硬件和编译器供应商可能仅针对这些程序优化其设计

•最佳的基准测试类型是实际应用程序，因为它们反映了最终用户的需求

•某些架构在一些应用程序上表现良好，而在其他应用程序上表现不佳

•编译可以通过利用特定架构的特性来提升性能。针对特定应用的编译器优化正变得越来越流行。

78

SPEC基准测试：桌面基准测试

单核CPU密集型基准测试

> SPEC89

> SPEC92

> SPEC95

> SPEC2000

> SPEC CPU2006 (12 CINT2006, 17 CFP2006) □ 图

形密集型基准测试

> SPEC2000

•SPEC视图性能测试

> 用于对支持 OpenGL 图形库的系统进行基准测试

•SPECapc

> 由大量使用图形的应用程序组成。

79

80

SPEC INT 95基准测试描述

基准测试	参考时间 (秒)	应用领域	特定任务
099.go	4600	游戏; 人工智能	与自身进行围棋对弈。
124.m88ksim	1900	模拟	模拟摩托罗拉88100处理器运行Dhrystone和内存测试程序。
126.gcc	1700	编程与编译	将预处理后的源文件编译为优化的SPARC汇编代码。
129.压缩	1800	压缩	使用自适应Lempel-Ziv编码压缩大型文本文件(约16MB)。
130.li	1900	语言解释器	Lisp解释器。
132.渐进式JPEG	2400	成像	使用各种参数执行JPEG图像压缩。
134.perl	1900	Shell解释器	执行文本和数字操作(变位词/质数分解)。
147.vortex	2700	数据库	构建并操作三个相互关联的数据库。

81

SPEC浮点运算95基准测试描述

基准测试	参考时间 (秒)	应用领域	具体任务
101.汤姆猫v	3700	流体动力学/几何变换	围绕一般几何域生成二维边界拟合坐标系。
102.游泳	8600	天气预报	使用有限差分近似求解浅水方程。(CFP95中唯一的单精度基准测试。)
103.su2cor	1400	量子物理学	基本粒子的质量是在夸克-胶子理论中计算得出的。
104.hydro2d	2400	天体物理学	流体动力学的纳维-斯托克斯方程用于计算星系喷流。
107.mgrid	2500	电磁学	三维势场的计算。
110.applu	2200	流体动力学/数学	通过主元法求解矩阵系统。
125.三维湍流模拟	4100	模拟	模拟立方区域内的湍流。
141.大气污染模拟	2100	天气预报	计算网格内温度和污染物的统计数据。
145.fpppp	9600	化学	执行多电子导数计算。
146.wave	3000	电磁学	在笛卡尔网格上求解麦克斯韦方程组。

82

新的SPEC Int 2000基准测试

164.gzip	C	压缩
175.vpr	C	FPGA电路布局与布线
176.gcc	C	C编程语言编译器
181.mcf	C	组合优化
186.crafty	C	游戏玩法：国际象棋
197.parser	C	文字处理
252.eon	C++	计算机可视化
253.perlbmk	C	PERL编程语言
254.gap	C	群论，解释器
255.涡旋	C	面向对象数据库
256.bzip2	C	压缩
300.twolf	C	布局与布线模拟器

83

168.wupwise	Fortran 77	物理学/量子色力学
171.swim	Fortran 77	浅水建模
172.mgrid	Fortran 77	多重网格求解器：三维势场
173.applu	Fortran 77	抛物/椭圆型偏微分方程
177.mesa	C	三维图形库
178.galgel	Fortran 90	计算流体动力学
179.艺术	C	图像识别/神经网络
183.地震	C	地震波传播模拟
187.人脸识别	Fortran 90	图像处理：人脸识别
188.ammp	C	计算化学
189.lucas	Fortran 90	数论/素性测试
191.fma3d	Fortran 90	有限元碰撞模拟
200.sixtrack	Fortran 77	高能核物理加速器设计
301.apsi	Fortran 77	气象学：污染物分布

84

SPEC基准测试 服务器基准测试

④ SPECrate——多处理器的处理速率

- SPEC CPU2000——面向吞吐量的基准测试
- SPECrate——多处理器的处理速率
- SPECFSF——文件服务器基准测试
- SPECWeb——Web服务器基准测试
- 事务处理 (TP) 基准测试
- TPC基准测试——事务处理委员会
 - TPC - A, 1985年
 - TPC-C, 1992年
 - TPC-H → TPC-R → TPC-W

85

SPEC基准测试 嵌入式基准测试

□ EDN嵌入式微处理器基准测试联盟 (即EEMBC, 发音为“embassy”)。

基准测试类型	内核数量	示例基准测试
汽车/工业	16	6个微基准测试(算术运算、指针追踪、内存性能、矩阵运算、查表、位操作)、5个汽车控制基准测试和5个滤波器或快速傅里叶变换(FFT)基准测试
消费级	5	5个多媒体基准测试(JPEG压缩/解压缩、滤波和RGB转换)
网络连接	3	最短路径计算、IP路由和数据包流操作
办公自动化	4	图形和文本基准测试(贝塞尔曲线计算、抖动处理、图像旋转、文本处理)
电信	6	过滤和数字信号处理基准测试(自相关、快速傅里叶变换、解码器、编码器)

86

运行基准测试

□ 关键因素：其他实验者的可重复性。□ 细节，还是细节，更多的细节！！！列出你实验的所有假设和条件。

> 即程序输入、程序版本、编译器版本、优化级别、操作系统版本、主内存大小、磁盘类型等。

□ A 系统的软件配置会显著影响基准测试的性能结果。

比较两台机器

机器	CPI	时钟周期	平均指令时间 (secs)
机器A	1.2	2 ns	
机器B	2.5	1 ns	

□ 执行指令的总CPU时间 = # * 平均指令时间
□ 假设执行1,000,000,000条指令

□ 机器A: $1,000,000,000 * 2.4 \text{ 纳秒} = 2.4 \text{ 秒}$
□ 机器B: $1,000,000,000 * 2.5 \text{ 纳秒} = 2.5 \text{ 秒}$
□ 哪台机器更快？机器A
□ 快多少？ $2.5 / 2.4 = 1.04 \text{ times faster}$

87

88

性能比较

•通常，我们希望比较不同机器或不同程序的性能。为什么呢？

- 帮助工程师了解哪个“更好”
- 为营销提供新闻稿中的“万灵药”
- 帮助客户理解为什么他们应该购买<my machine>

•性能和执行时间互为倒数

最大化性能意味着最小化响应（执行）时间

$$\text{性能} = \frac{1}{\text{Execution Time}}$$

常用短语

|-对于给定的工作负载，“ P_1 的性能优于 P_2 ”意味着 L, P_1 执行 L 所需的时间比 P_2 少

性能(P_1) > 性能(P_2)

\Rightarrow 执行时间 (P_1, L) < 执行时间 (P_2, L)

□ “处理器X比Y快n倍”是

$$n = \frac{\text{Execution time } Y}{\text{Execution time } X}$$



89

跨多个程序比较性能

	计算机A	计算机B	计算机C
程序1 (秒)	1	10	20
程序2 (秒)	1000	100	20
程序3 (秒)	1001	110	40

- > A 在程序1中比 B 快10倍
- > B在程序2中比A快10倍
- > A在程序1中比 C 快20倍
- > C 在程序2中比 A 快50倍
- > 在程序1中比 C 快2倍
- > C 在程序2中比 B 快5倍

| [以上每个陈述都是正确的...

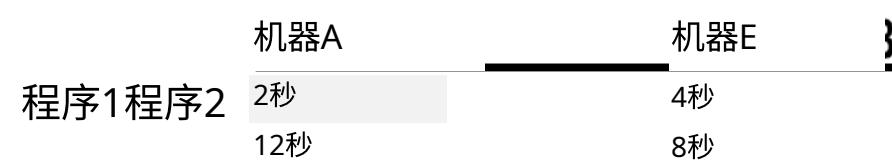
...但我们想知道哪台机器是最好的？

91

示例：第二个答案

□ 两台机器在两个基准测试上的计时

- > 机器A比机器B快多少？



□ 尝试2：运行时间比率，以机器B的时间为基准进行归一化

- > 程序1: 2/4 程序2: 12/8
- > 机器A运行程序1的时间是机器B的1/2，运行程序2的时间是机器B的3/2
- > 平均而言， $(1/2 + 3/2)/2 = 1$
- > 换句话说，机器A比机器B快1.0倍

93

哪个是正确的？

□ 问题：

- > 我们如何能得到三个不同的答案？

□ 解决方案

- > 因为，虽然它们都是合理的计算.....
- >每个都回答了一个不同的问题

□ 我们需要更精确地理解和提出这些性能与指标问题

90

让我们尝试一个更简单的例子

□ 两台机器在两个基准测试上的计时

- > 机器A比机器B快多少？
- 机器A 机器B
程序1 2秒 4秒
程序2 12秒 8秒

□ 尝试1：运行时间的比率，以机器A的时间为基准进行归一化

- > 程序1: 4/2 程序2: 8/12
- > 机器A在程序1上的运行速度快2倍，在程序2上快2/3倍
- > 平均而言，机器A比机器B快 $(2 + 2/3)/2 = 4/3$ 倍

□ 事实证明，这种“平均”的方法可能会误导我们

92

示例：第三个答案

□ 两台机器在两个基准测试上的计时

- > 机器A比机器B快多少？



□ 尝试3：运行时间比率，总计（总和）时间

- > 机器A运行两个程序用时14秒
- > 机器B运行两个程序用时12秒
- > 因此，机器A所用时间是机器B的14/12
- > 换句话说，机器A比机器B快6/7

94

算术和调和

□ 总执行时间：一个一致的汇总指标

- > 算术平均值是跟踪总执行时间的执行时间的平均值。

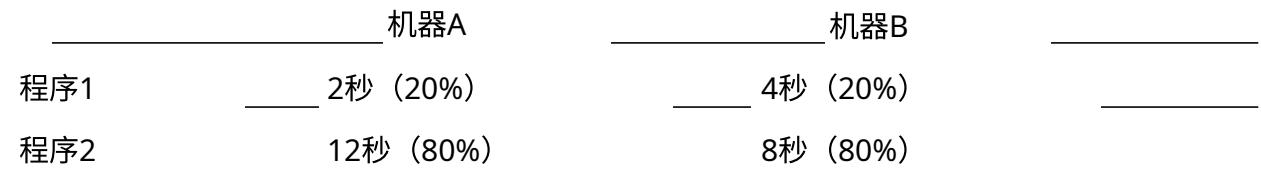
$$\frac{1}{n} \sum_{i=1}^n Time_i$$

- > 如果性能以速率表示，那么跟踪总执行时间的平均值就是调和平均值

$$\frac{n}{\sum_{i=1}^n \frac{1}{Rate_i}}$$

算术问题

- 应用程序被运行的概率并不相同
- 较长的程序在平均值中占比更重
- 例如，两台机器在两个基准测试中计时



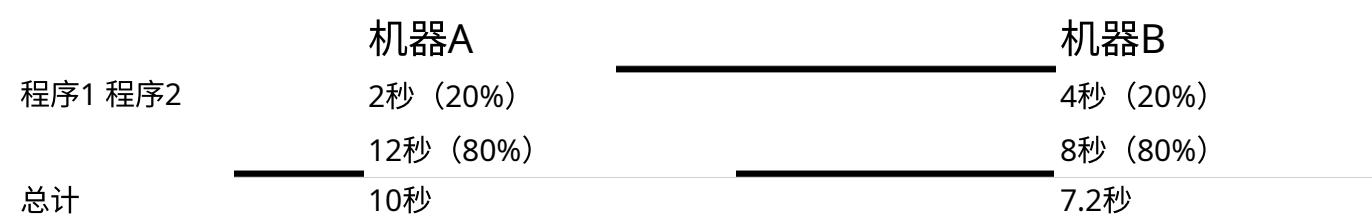
- 如果我们采用算术平均值，程序2的“权重”比程序1更大

> 程序2的改进对平均值的影响比程序1按比例的改进更大

- 但也许程序1运行的可能性是程序2的4倍

97

使用加权和或加权平均值



使我们能够确定相对性能 $10/7.2 = 1.38$

--> 机器B比机器A快1.38倍

加权执行时间

- 通常，人们运行某些程序的频率比其他程序更高。
- 因此，我们应该对更常用程序的执行时间进行加权

$$\sum_{i=1}^n Weight_i \times Time_i$$

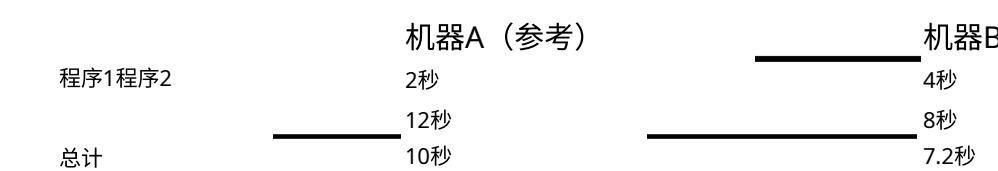
- 加权调和平均数

$$\frac{1}{\sum_{i=1}^n \frac{Weight_i}{Rate_i}}$$

98

另一种解决方案

- 将每个程序的运行时间归一化为一个参考值



	机器A (相对于B的标准)	机器B (相对于A的标准)
程序1	0.5	2.0
程序2	1.5	0.666
平均值?	1.0	1.333

- 因此，当我们把A归一化到B并取平均值时，看起来A&B是相同的。

- 但当我们把B归一化到A时，看起来B要好33%！

99

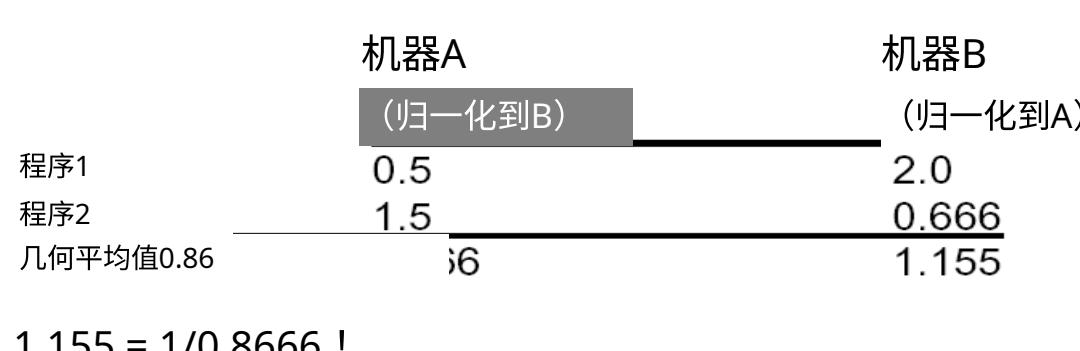
示例

程序	权重				
	AB	C	W(1)	W(2)	W(3)
Program P1 (secs)	1.00	10.00	20.00	0.50	0.909
Program P2 (secs)	1000.00	100.00	20.00	0.50	0.091
算术平均值: W(1)	500.50	55.00	20.00		
算术平均值: W(2)	91.91	18.19	20.00		
算术平均值: W(3)	2.00	10.09	20.00		

$W(B)_1 = \frac{1}{10 \times (1/10 + 1/100)} = 0.909$	等时间加权	$W_i := \frac{1}{Time_i \times \sum_{j=1}^n \left(\frac{1}{Time_j} \right)}$
$W(B)_2 = \frac{1}{100 \times (1/10 + 1/100)} = 0.091$		

101

使用几何平均数



$$1.155 = 1 / 0.8666 !$$

缺点：

- 几何平均值无法预测运行时间
- 进行归一化处理。
- 现在每个应用程序的权重相同。
- 优点：参考计算机在相对性能

性能比较总结

总执行时间或算术平均值

- 一致的结果
- 工作负载中的程序并非总是运行相同的次数

加权算术平均值

- 考虑工作负载中的使用频率
- 解决方案取决于以哪台机器作为参考。

归一化几何平均值——SPEC比率

- 无论以哪台机器作为参考，结果都一致。
- 几何平均值无法预测运行时间

理想解决方案：测量实际工作负载并根据程序的执行频率对其进行加权。

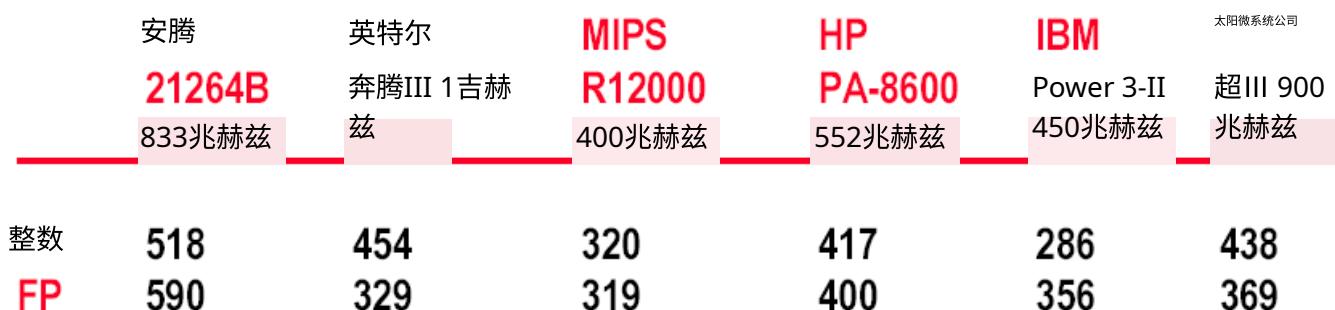
真正重要的是您的应用程序的性能

103

104

新的SPEC性能数据

- 12项 (SpecInt) 和 14项 (SpecFP) 基准测试的几何平均值
 - > 相对于SPARC 10/40的性能测量结果
- 2000年性能数据 (《微处理器报告》, 2000年12月)



105



106

章节主题

- 1.1 为什么要学习本课程?
- 1.2 当前计算机市场中的计算机类别
- 1.3 定义计算机体系结构以及计算机设计的任务是什么?
- 1.4 技术趋势
- 1.5 集成电路的功率趋势
- 1.6 成本趋势
- 1.7 可靠性
- 1.8 测量、报告和总结性能
- 1.9 计算机设计的定量原则
- 1.10 综合运用 107

利用并行性

- 提高性能最重要的方法 □ 并行级别 > 系统级别：使用多个处理器 > 指令级别：- 流水线技术

- > 操作级别：- 组相联缓存
- 流水线功能单元



1.9 量化原则

- 利用并行性
- 局部性原理
- 关注常见情况
- 阿姆达尔定律
- MCPU性能方程

Principle of Locality

Any example ?

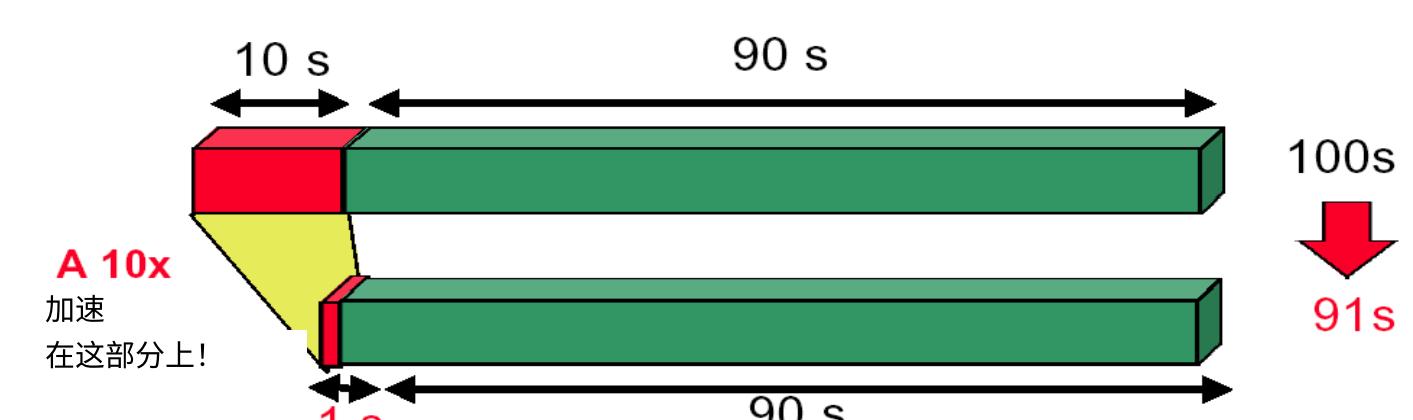
- 程序特性：程序倾向于重用它们最近使用过的数据和指令。
- 经验法则：
 - > 一个程序仅在10%的代码中花费90%的执行时间。
- 时间局部性
 - > 最近访问过的项很可能在不久的将来被再次访问。空间局部性 > 地址彼此接近的项往往在时间上被紧密引用。

109 关注常见情况

阿姆达尔定律

- 计算机设计中最重要且普遍适用的原则
- 计算机设计。
 - 功率、资源分配、性能
 - 可靠性
- > 经验法则：简单即快速。
- > 常见情况通常更简单，并且可以完成更快。
- 一条基本定律，称为阿姆达尔定律，可以用来量化这一原理。

- 通过某种更快执行模式所能获得的性能提升受限于使用某种更快执行模式所能获得的性能提升受限于更快模式可使用的时间比例使用。
- 示例



改进后的执行时间 =

$$\frac{\text{受改进影响的执行时间改进量}}{\text{+不受影响的执行时间}}$$

- 提高时钟频率不会影响内存访问时间
- 使用浮点处理单元不会加快整数算术逻辑单元操作

113 加速方 程

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExecTime}_{\text{old}}}{\text{ExecTime}_{\text{new}}} = \frac{1}{\left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)}$$

示例：

> 一个使用增强型CPU（比原CPU快10倍）的服务器系统用于Web服务。假设原CPU有40%的时间忙于计算，有1/0.60%的时间处于等待状态。□ 答案：比例 enhanced = 0.4，速度 up_enhanced = 10

$$\text{Speedup} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

115

阿姆达尔定律定义了加速比

加速比

$$= \frac{\text{Performance with enhancement}}{\text{Performance without enhancement}} = \frac{\text{Execution time w/o enhancement}}{\text{Execution time with enhancement}}$$

□ 如果我们知道两个因素：> 增强部分比例：原机器中可转换为利用增强功能的计算时间比例

增强功能的优势。

> 增强模式下的加速比：增强执行模式带来的改进：

$$\text{执行时间}_{\text{new}} = \text{执行时间}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}}{\text{Speedup}} \right)$$

114

另一个示例

□ 浮点 (FP) 平方根的实现方式在性能上存在显著差

异

□ 两项改进建议

> 一项建议是改进FPSQR硬件，并将此操作速度提高10倍。

> 另一种选择是尝试使图形处理器中的所有FP指令运行速度提高1.6倍；

□ 假设

> 浮点平方根 (FPSQR) 占关键图形基准测试执行时间的20%。

> 浮点指令总共占应用程序执行时间的50%。

> 设计团队认为他们以相同的工作量进行这两项改进。

□ 比较这两种设计方案。

116

示例解答

$$\text{Speedup}_{\text{FPSQR}} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

117

□ 已知：

$$\begin{aligned} &> \text{Failure rate}_{\text{system}} = 10^*1/1000,000 + 1/500,000 + 1/200,000 + \\ &\quad \text{十万分之一} \\ &= \frac{10 + 2 + 5 + 5 + 1}{1,000,000} \\ &= 23000/1000,000,000 \end{aligned}$$

那么，停电 % = 5/23 = 0.22

118

阿姆达尔定律意味着什么？

□ 如果一项改进仅适用于部分任务，那么总加速比将不超过 $1 / (1 - F)$ 。

□ 提供指导

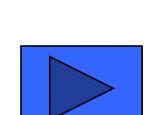
> 关于一项改进将在多大程度上提升性能
> 关于如何分配资源以提高性价比

□ 可用于比较

> 两种方案的整体系统性能
> 两种CPU设计方案

□ 我们可以通过以下方式提高性能

> 增加增强部分的比例
> 或，提高增强部分的加速比



119

120

CPU性能方程

□ 处理器性能的“铁律”：

- > 通常，直接衡量使用新增强功能在时间上的改进是困难的。

□ CPU性能方程

$$\text{CPU时间} = \text{程序的CPU时钟周期数} \times \text{时钟周期时间}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

CPU时间的计算

$\text{CPU时间} = \text{指令数量} \times \text{每条指令的时钟周期数} \times \text{时钟周期时间}$

Or
$$\text{CPU时间} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

$$\text{CPU时间} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

架构 → 实现 → 实现情况
编译器设计师 处理器设计师 芯片设计师

性能组成部分	度量单位
程序的CPU执行时间	程序执行的秒数
指令计数	程序执行的指令数
每条指令的时钟周期数 (CPI)	平均每个指令的时钟周期数
时钟周期时间	每个时钟周期的秒数

121 相关技术

□ CPU性能取决于3个特性：

- > 时钟周期 (或频率) (CCT)
- > 每条指令的时钟周期数 (CPI)
- > 指令数量。 (IC)

	指令数量	CPI	时钟频率
程序	x		
编译器	x	(X)	
指令集	x	x	
组织		x	x
技术			x

□ 一个难题：孤立地改变其中一个而不影响其他方面是很困难的。

123

CPU时间计算示例

□ 假设我们进行了以下测量：

- 浮点运算 (除了FPSQR) 的频率 = 25%
- > 浮点运算的平均CPI = 4.0
- > 其他指令的平均CPI = 1.33
- > FPSQR的频率 = 2%
- > FPSQR的CPI = 20

□ 两种设计方案

- > 将FPSQR的CPI降至2
- > 将所有浮点运算的平均CPI降至2.5。

□ 使用CPU性能方程比较这两种设计方案。

125

将结果与阿姆达尔定律得出的结果进行比较

□ 这与我们使用阿姆达尔定律得到的加速比相同：

$$\begin{aligned} \text{Speedup}_{\text{new FP}} &= \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} = \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}} \\ &= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{new FP}}} = \frac{2.00}{1.625} = 1.23 \end{aligned}$$

122
其他格式的
CPU性能方程

$$\text{CPU clock cycles} = \sum_{i=1}^n \text{IC}_i \times \text{CPI}_i$$

$$\text{CPU time} = \left(\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

$$\text{CPI} = \frac{\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i}{\text{Instruction count}} = \sum_{i=1}^n \frac{\text{IC}_i}{\text{Instruction count}} \times \text{CPI}_i$$

124

问题的答案

$$\begin{aligned} \text{CPI}_{\text{original}} &= \sum_{i=1}^n \text{CPI}_i \times \left(\frac{\text{IC}_i}{\text{Instruction count}} \right) \\ &= (4 \times 25\%) + (1.33 \times 75\%) = 2.0 \end{aligned}$$

$$\begin{aligned} \text{CPI}_{\text{with new FPSQR}} &= \text{CPI}_{\text{original}} - 2\% \times (\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{of new FPSQR only}}) \\ &= 2.0 - 2\% \times (20 - 2) = 1.64 \end{aligned}$$

$$\text{CPI}_{\text{new FP}} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625$$

□ 我由于整体浮点增强的CPI略低，其性能将略有提升。

126

□ 假设一个应用程序中有四种类型的操作。对原始功能单元进行增强后，每种类型的操作都获得了如下表所示的性能提升。

操作类型	指令计数 (共100条)	增强前的CPI	CPI 之后 增强 nt
Op1	10	2	1
Op2	30	20	15
Op3	35	10	3
Op4	25	4	1

1) 改进后各操作的增强加速比分别是多少？

2) 仅改进操作2或操作4后，应用程序的整体加速比分别是多少？

□ 提示：仅考虑使用一个增强功能单元来计算整体加速比。有两种情况。

127

128

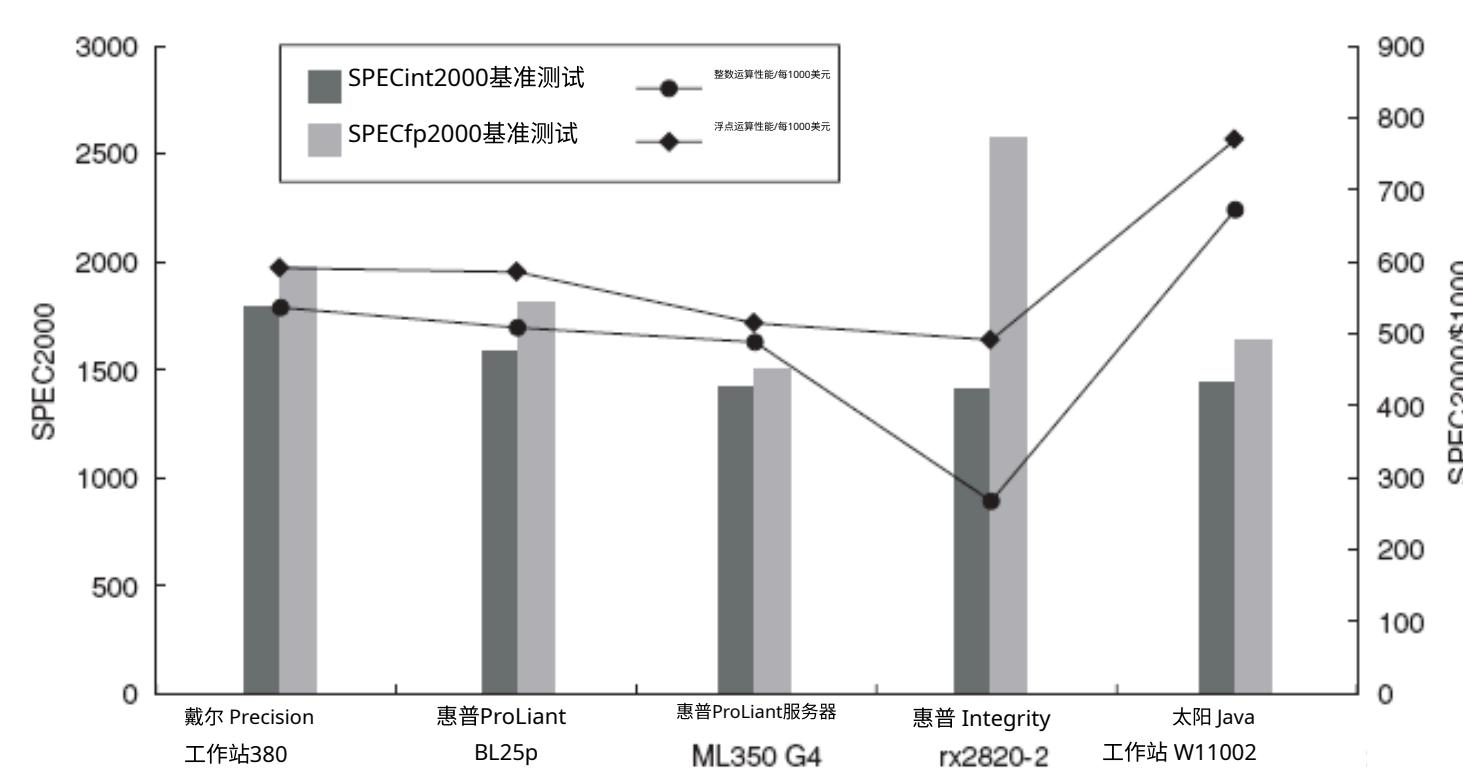
性能与性价比

□ 导致价格差异较大的因素

- > 不同的可扩展性水平
- > 使用较便宜的磁盘和内存
- > CPU 成本不同
- > 软件差异
- > 低端系统在风扇、电源、支持芯片组中使用个人电脑通用零部件
- > 商品化效应

129

性价比



130

五款台式和机架式系统

供应商/型号	处理器	时钟频率	二级缓存	类型	价格
戴尔Precision工作站380	英特尔奔腾4至强	3.8 GHz	2 MB	台式	\$3346
惠普ProLiant BL25p	AMD皓龙252	2.6 GHz	1 MB	机架式	\$3099
惠普ProLiant ML350 G4服务器	英特尔奔腾4至强处理器	3.4 GHz	1 MB	台式机	\$2907
惠普Integrity rx2620 - 2服务器	安腾2处理器	1.6 GHz	3 MB	机架式服务器	\$5201
太阳微系统公司Java工作站W1100z	超微皓龙150处理器	2.4 GHz	1 MB	桌面	\$2145

• 1GB纠错码同步动态随机存取存储器，80GB硬盘

- 可扩展性：

- 太阳微系统公司Java工作站 < 戴尔 < 惠普BL25p
- 处理器成本：芯片尺寸、二级缓存、时钟频率、缓存大小
- 软件差异

131

测量 - 1

□ 针对服务器

TPC - C：在线事务处理（OLTP）的标准行业基准

- 合理近似值
- 测量整个系统的性能
- 测量规则非常完善
- 供应商投入了大量精力
- 同时报告性能和性价比

132

TPC - C与TPC - E

□ TPC (事务处理性能委员会)

> 10 成员 [http://www\(tpc.org](http://www(tpc.org)

□ TPC - C:

- > 在线事务处理基准测试
- > 模拟一个完整的环境，在该环境中，一群终端操作员针对数据库执行事务

□ TPC - E

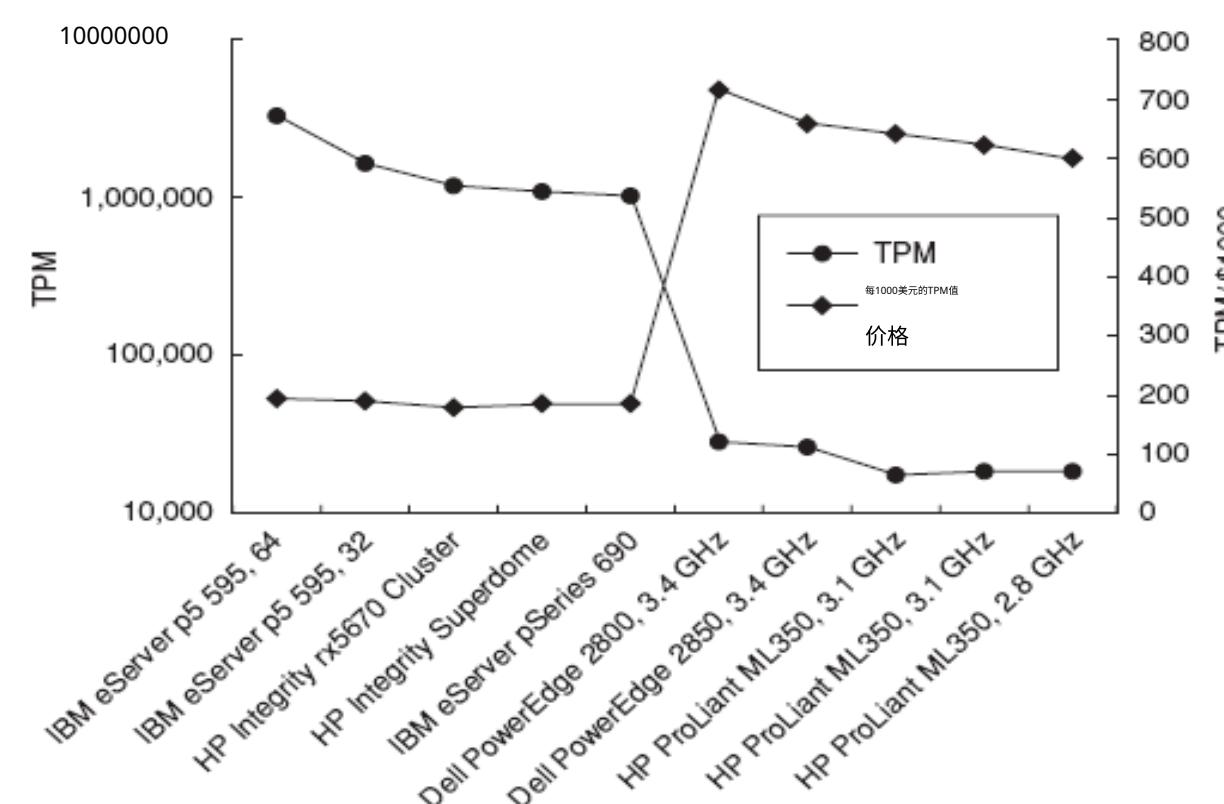
- > 模拟经纪公司的在线事务处理工作负载

□ TPC - C&TPC - E

http://www.searchsv.com.cn/showcontent_13524.htm

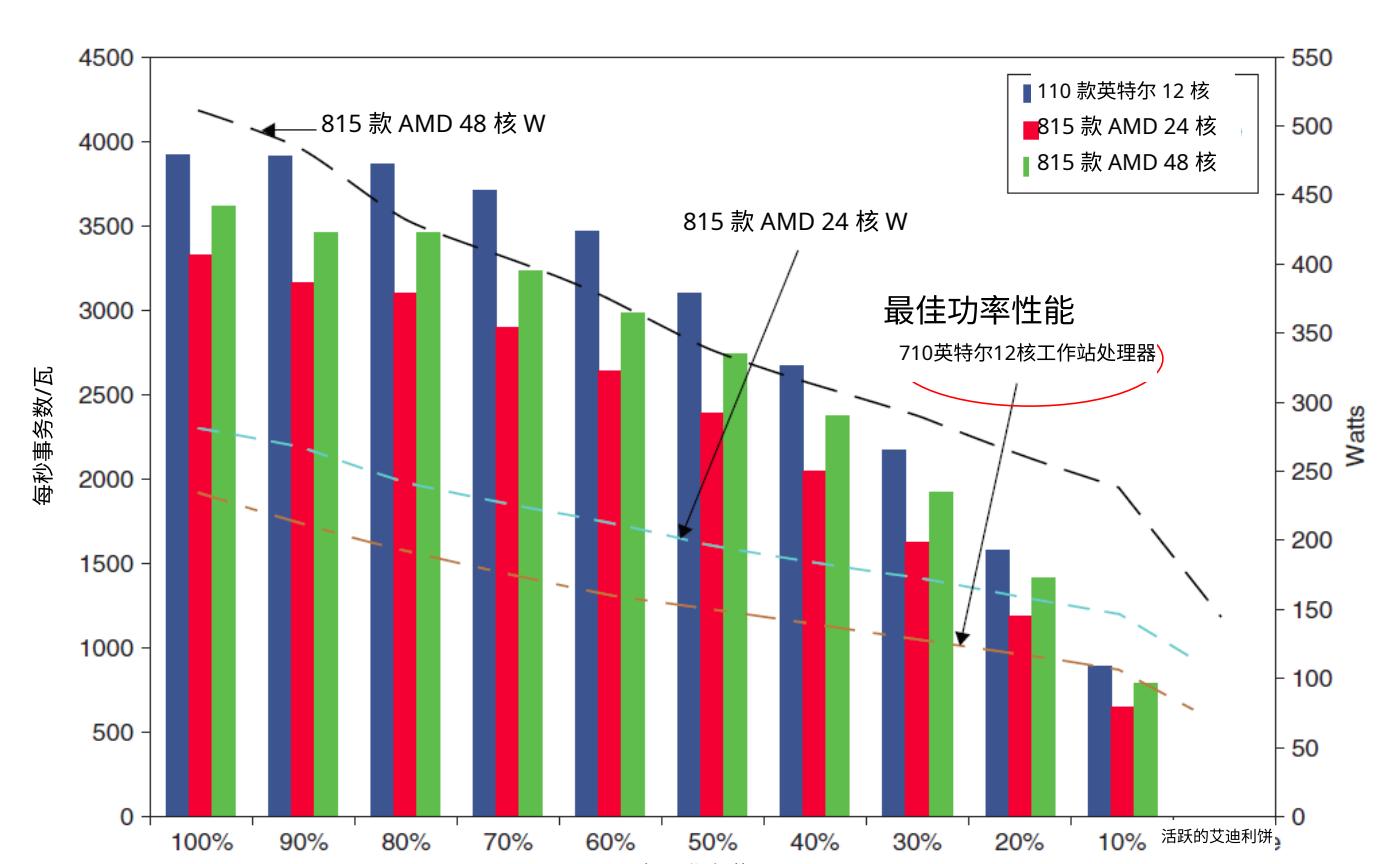
133

性价比TPC - C 2005.7



134

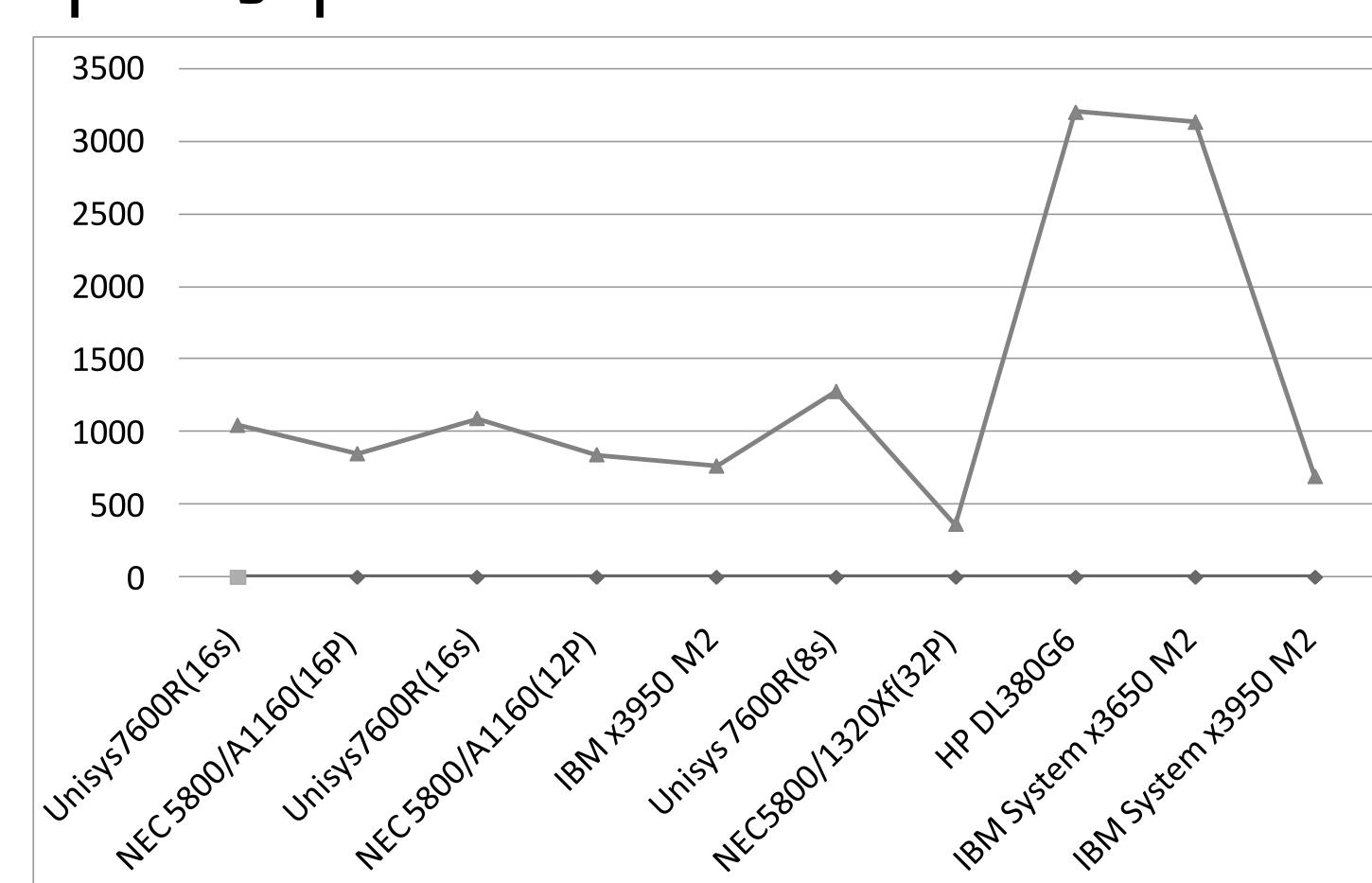
3台服务器的性能表现



2012年2月13日

135

tpsE与tpsE/100000



136

□陷阱（易犯的错误）：

- >成为阿姆达尔定律的牺牲品。
- 不要试图改进某个小的单元 F
- > 单点故障可靠性的一个例子——别忘了那个单风扇！

137

	处理器 + 机柜	内存	存储	软件
IBM eServer p5 595	28%	16%	51%	6%
IBM eServer p5 595	13%	31%	52%	4%
惠普Integrity rx5670集群	11%	22%	35%	33%
惠普Integrity Superdome服务器	33%	32%	15%	20%
IBM eServer pSeries 690	21%	24%	48%	7%
高性能计算机中位数	21%	24%	48%	7%
戴尔PowerEdge 2800服务器	6%	3%	80%	11%
戴尔PowerEdge 2850	7%	3%	76%	14%
惠普ProLiant ML350	5%	4%	70%	21%
惠普ProLiant ML350	9%	8%	65%	19%
惠普ProLiant ML350	8%	6%	65%	21%
性价比计算机的中位数	7%	4%	70%	19%

138

谬误

□基准测试永远有效

- > “基准测试技巧”“基准测试工程”

□磁盘的额定平均故障间隔时间为1200000小时，约140年，因此磁盘几乎不会发生故障。

- > 对于ATA磁盘，实际平均故障间隔时间（MTTF）比制造商宣称的要差2 - 4倍；对于SCSI磁盘，要差4 - 8倍。

(ATA: AT附件接口)

(SCSI: 小型计算机系统接口)

□峰值性能与观测到的性能相符。

139