

第七章 系统性能评价简介

计算机系统性能评价

一、系统性能评价技术

从技术上,主要有分析、模拟、测量三种技术。

1. 分析技术

在一定假设条件下,计算机系统参数与性能指标参数之间存在着某种函数关系,按其工作条件列出方程,通过数学方法求解。

较常采用排队论、随机过程、均值分析等方法进行近似求解,比如流水线性能、多处理器系统性能分析、软件可靠性静态评估等。

特点是理论严密,对基础理论的掌握要求较高。优点是节约人力/物力,可应用于设计中的系统。

2. 模拟技术

模拟技术是基于实验数据的系统建模,主要有:

- (1) 按系统的运行特性建立系统模型;
- (2) 按系统工作负载情况建立工作负载模型;
- (3) 编写模拟程序,模拟被评价系统的运行。

上述方法都需要进行试验,得出实验值,即获取测试数据(几乎所有基于模拟的评价方法都依赖于测试数据或实验值),再进行统计、分析、建模。比如网络负载与性能分析、软件可靠性动态评价。

特点是既可以应用于设计中或实际应用中的系统,也可以与分析技术相结合,构成一个混合系统。

3. 测量技术

该技术是对已投入使用的系统进行测量,通常采用不同层次的基准测试程序评估。

- 实际应用程序
- 核心程序
- 合成测试程序

二、系统性能评价对象

- CPU的性能 (比如IPC或CPI)
- 内存的性能
(如内存容量、总线数据宽度、读写延迟和带宽等)
- I/O的性能
(如I/O总线数目和带宽、磁盘通道的数目和带宽、磁盘的性能(转速、寻道时间、扇区缓存容量等))
- 网络的性能
- 操作系统的性能
(系统调用/中断/进程切换/线程调度开销、存储映射/文件系统的缓冲区性能和吞吐量等。)
- 编译器的性能 等
CPU按流水线方式工作, 编译器对CPU性能影响很大

对**CPU性能指标**, 可以从**多个方面**进行评价, 如:

- 主频(同种类型的CPU比较才具有参考价值)
- 运算部件数
- 流水线长度
- 指令的并行度
- 定点部件性能
- 浮点部件性能
- Cache大小和命中率等

具体评价指标大多围绕“**速度**”, 而衡量“**速度**”的直接参数则是**程序执行时间**。

三、CPU性能公式

与程序执行的时间相关的两大因素:

- (1) 时钟频率(MHz);
- (2) 执行程序所用的总时钟周期数。

程序执行的CPU时间:

$$\begin{aligned}\text{CPU时间} &= \text{总时钟周期数} \times \text{时钟周期} \\ &= \text{总时钟周期数} / \text{时钟频率}\end{aligned}$$

引入参数**IC**(程序执行的指令数)和**CPI**(每条指令所需时钟数)两个参数,则:

$$\text{CPU时间} = \text{CPI} \times \text{IC} \times \text{时钟周期} = \text{CPI} \times \text{IC} / \text{时钟频率}$$

该公式通常称为CPU性能公式。公式中的三个参数反映了与体系结构相关的三种技术:

- (1) **时钟频率**: 反映计算机实现、工艺和组织技术;
- (2) **CPI**: 反映计算机实现、指令集结构和组织;
- (3) **IC**: 反映计算机指令集结构和编译技术。

但对于性能公式中的**CPI**, 不同指令所需的时钟可能是不同的。因此:

假设系统有**n**种指令: 第**i**种指令的时钟数为**CPI_i**, 在程序中第**i**种指令出现的次数为**IC_i**, 由此可得指令的平均时钟数:

$$CPI = \sum_{i=1}^n (CPI_i \times IC_i) / IC = \sum_{i=1}^n (CPI_i \times IC_i / IC)$$

式中: **(IC_i / IC)**反映了第**i**种指令在程序中所占比例, 所以**CPI**反映了不同指令平均所需时钟数。

根据CPU性能公式:

$$\text{CPU时间} = \text{CPI} \times \text{IC} \times \text{时钟周期} = \text{CPI} \times \text{IC} / \text{时钟频率}$$

CPU性能公式可细化为:

$$\begin{aligned}\text{CPU时间} &= \sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i) \times \text{时钟周期} \\ &= \sum_{i=1}^n (\text{CPI}_i \times \text{IC}_i) / \text{时钟频率}\end{aligned}$$

可利用上述性能公式进行系统性能分析

例1: 假设指令集中条件分支指令有两种不同设计方法:

(1) CPU_A: 通过比较指令设置条件码, 然后测试条件码进行分支;

(2) CPU_B: 在分支指令中包括比较过程。

假设: 在两种CPU中, 条件分支指令都占用2个时钟周期, 所有其它指令占用1个时钟周期。

对于CPU_A, 假设执行的指令中分支指令占20%; 由于每个分支指令之前都需要有比较指令, 因此比较指令也占20%。由于CPU_A在分支时不需要比较, 因此假设它的时钟周期时间比CPU_B快1.25倍。

问: 哪一个CPU更快? 如果CPU_A的时钟周期时间仅比CPU_B快1.1倍, 哪一个CPU更快?

解：不考虑系统中其他问题，直接用CPU性能公式。占用2个时钟周期的分支指令占总指令的20%，剩下的指令占用1个时钟周期。所以指令的平均周期数：

$$CPI_A = 20\% \times 2 + 80\% \times 1 = 1.2$$

则CPU性能为：

$$\text{总CPU时间}_A = IC_A \times 1.2 \times \text{时钟周期}_A$$

根据假设(CPU_A比CPU_B快1.25倍)有：

$$\text{时钟周期}_B = 1.25 \times \text{时钟周期}_A$$

CPU_B中**没有**独立的比较指令，所以CPU_B的程序量为CPU_A的80%，假设仍有20%的分支指令，则分支指令的比例为：

$$20\% \div 80\% = 25\%$$

这些分支指令占用2个时钟周期,而剩下的75%的指令占用1个时钟周期,因此指令的平均周期数:

$$CPI_B = 25\% \times 2 + 75\% \times 1 = 1.25$$

因为CPU_B没有独立比较指令,故指令数:

$$IC_B = 80\% \times IC_A$$

因此:

$$\text{时钟周期}_B = 1.25 \times \text{时钟周期}_A$$

$$\begin{aligned} \text{总CPU时间}_B &= IC_B \times CPI_B \times \text{时钟周期}_B \\ &= 0.8 \times IC_A \times 1.25 \times (1.25 \times \text{时钟周期}_A) \\ &= IC_A \times 1.25 \times \text{时钟周期}_A \end{aligned}$$

$$\text{对比: 总CPU时间}_A = IC_A \times 1.2 \times \text{时钟周期}_A$$

尽管CPU_B执行指令条数较少,但因为CPU_A的时钟周期更短,所以CPU_A比CPU_B快。

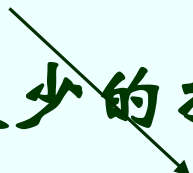
如果CPU_A的时钟周期时间比CPU_B快1.1倍,则:

$$\text{时钟周期}_B = 1.1 \times \text{时钟周期}_A$$

CPU_B的性能为:

$$\begin{aligned}\text{总CPU时间}_B &= IC_B \times CPI_B \times \text{时钟周期}_B \\ &= 0.8 \times IC_A \times 1.25 \times (1.1 \times \text{时钟周期}_A) \\ &= IC_A \times 1.1 \times \text{时钟周期}_A\end{aligned}$$

因此CPU_B由于执行更少的指令,比CPU_A运行更快。


$$\text{对比: 总CPU时间}_A = IC_A \times 1.2 \times \text{时钟周期}_A$$

例2:如果FP操作的比例为25%, FP操作的平均CPI=4, 其它指令的平均CPI=1.33(包括FPSQR操作), FPSQR操作的比例为2%, FPSQR操作CPI=20。假设有两种设计方案:把FPSQR操作的CPI减为2或所有FP操作的CPI减为2, 试用CPU性能公式比较两种方案。

解: (1) 没有采取提高措施以前原系统的CPI为:

$$CPI_{原} = \sum(CPI_i \times IC_i / IC) = 4 \times 25\% + 1.33 \times 75\% = 2$$

(2) 采用方案1(把FPSQR操作的CPI减为2):

其他指令(75%), 除去FPSQR操作的平均CPI值 **X** :

$$(73\% \mathbf{X} + 2\% \times 20) / 75\% = 1.33 \quad \text{则有} \quad \mathbf{X} = 0.818$$

$$CPI_{方案1} = 4 \times 25\% + 0.818 \times 73\% + 0.02 \times 2 = 1.64$$

(3) 采用方案2(把FP操作的CPI减为2):

$$\text{CPI}_{\text{方案2}} = 2 \times 25\% + 1.33 \times 75\% = 1.5$$



$$\text{CPI}_{\text{方案1}} = 4 \times 25\% + 0.818 \times 73\% + 0.02 \times 2 = 1.64$$

所以, 提高所有FP指令处理速度(方案2)比提高FPSQR操作的速度更有效。

四、系统性能评价标准

即:系统性能评价的公认的量化指标

(1) 时钟频率(主频): 用于同类处理机之间

如:PentiumII/450比PentiumII/300快50%。

影响系统速度的因素很多,主频仅有参考价值

(2) 指令执行速度法 (MIPS - 定点运算)

$$\text{MIPS} = \frac{\text{指令条数}}{\text{执行时间} \times 10^6} = \frac{F_z}{\text{CPI}} = \text{IPC} \times F_z$$

每一时钟周期
执行的指令条
数

例1: PentiumII 450处理机:

$\text{IPC} = 2$ (或 $\text{CPI} = 0.5$), $F_z = 450\text{MHz}$

$\text{MIPS} = \text{IPC} \times F_z = 2 \times 450 = 900 \text{ MIPS}$

指令执行平均
时钟周期数

例2: Pentium III 500有3条指令流水线, 则有:

$$\text{MIPS} = 3 \times 500\text{MHz} = 1500 \text{ (15亿次/s)}$$

(Pentium III增强的SIMD:既能处理整数, 也能处理浮点数)

例3: 一个由8台机器组成的Cluster系统, 每台机器是由4个PentiumIII 500组成SMP系统; 计算该系统的MIPS。

解: $\text{MIPS} = 500\text{MHz} \times 8 \times 4 \times 3 = 48 \text{ (GIPS-480亿次/s)}$

由于MIPS是直接根据主频得到的理论值, 可将其视为系统峰值速度。可用同样方法计算浮点运算性能:

$$\text{MFLOPS} = \frac{\text{程序中浮点操作次数}}{\text{执行时间} \times 10^6}$$

MIPS指标的主要缺点是不能反映以下情况:

① 不能反映不同指令对速度的影响

即使在同一台计算机上MIPS会因程序不同而变化(不同指令所需时钟周期数不同)。

② 不能反映指令使用频率差异的影响

③ 不能反映程序量对程序执行速度的影响

大型软件并且处理大数据量与小型软件处理少量数据相比,由于Cache的命中率的不同而导致执行时间的差异。

(3) 等效指令速度:吉普森(Gibson)法

MIPS不能反映不同指令使用频率差异的影响等,为此,根据不同类型的指令在程序出现的频率,乘以不同的系数,求出统计平均值。

i : 指令种类; T_i : 第 i 类指令执行时间

W_i : 第 i 类指令的使用频率;

• 等效指令执行时间: $T = \sum_{i=1}^n W_i \times T_i$

• 等效指令速度: $MIPS = 1 / \sum_{i=1}^n \frac{W_i}{MIPS_i}$

说明: $(1/MIPS_i)$ 表示第 i 类指令执行时间;

$(1/MIPS_i) \times W_i$ 为第 i 类指令等效执行时间;

$1 / \sum_{i=1}^n \frac{W_i}{MIPS_i}$ 即为等效的MIPS

例1: 假设在程序中浮点开平方操作FPSQR的比例为2%, 它的CPI为100; 其他浮点操作FP的比例为23%, 它的CPI=4.0; 其余75%指令的CPI=1.33, 计算该处理机的等效CPI。如果FPSQR操作的CPI也为4.0, 重新计算等效CPI。

解:

$$\text{等效CPI}_1 = 100 \times 2\% + 4 \times 23\% + 1.33 \times 75\% = 3.92$$

$$\text{等效CPI}_2 = 4 \times 25\% + 1.33 \times 75\% = 2.00$$

虽然FPSQR操作的CPI从100提高到4(为原来的25倍), 按照阿姆达尔定律, 因FPSQR操作只占2%的比例, 整体性能也仅约提高一倍。

例2: DJS-130小型机, 产品说明书提供的速度参数为每秒50万次, 即MISP=0.5, 该参数实际为执行定点加法指令的速度。定点加法指令速度为乘法和除法运算指令执行速度的1/100。

— 计算等效指令速度:

应用统计, 各类指令出现的频率为:

加/减法: 50%、乘法: 15%、除法: 5%

程序控制: 15%、其他15%

假设: 程序控制类指令和其它指令与定点加法指令的速度相同。

$$\text{MIPS} = \frac{1}{\sum_{i=1}^n \frac{W_i}{\text{MIPS}_i}}$$

各类指令出现的频率为：

加/减法50%、乘法15%、除法 5%

程序控制：15%、其他15%

$$= \frac{1}{\frac{0.5}{0.5} + \frac{0.15}{0.5 \times (1/100)} + \frac{0.05}{0.5 \times (1/100)} + \frac{0.15}{0.5} + \frac{0.15}{0.5}}$$
$$\approx 0.024$$

即为产品速度参数的 $0.024 \div 0.5 \approx 1/21$ 。

例3: 在一台40MHz处理机上运行的程序有200000条指令，主要由四种指令组成。每种指令所需的指令数如下：

指令类型	CPI	比例
算术和逻辑	1	60%
访存指令	2	8%
转移指令	4	12%
Cacha不命中	8	10%

(1) 计算运行程序的CPI

(2) 根据(1)所得CPI, 计算MIPS和CPU时间。

解: (1) $CPI = 1 \times 60\% + 2 \times 18\% + 4 \times 12\% + 8 \times 10\% = 2.24$

$$(2) \text{ MIPS} = \frac{f}{CPI \times 10^6} = \frac{40 \times 10^6}{2.24 \times 10^6} = 17.86$$

$$\text{CPU}_{\text{时间}} = \frac{IC}{\text{MIPS} \times 10^6} = \frac{200000}{17.86 \times 10^6} = 0.112\text{s}$$

吉普森(Gibson)法的主要缺点:

- (1) 同类指令在不同的应用中被使用的频率不同;
- (2) 程序量和数据量对Cache影响;
- (3) 流水线结构中指令执行顺序对速度的影响;
- (4) 编译程序对系统性能的影响。

例: 寄存器相加指令 $\text{ADD } R_i, R_j$; 完成 $R_i \leftarrow R_i + R_j$

程序段1:

.....

$\text{ADD } R_0, R_1$

$\text{ADD } R_2, R_3$

$\text{ADD } R_4, R_5$

.....

程序段2:

.....

$\text{ADD } R_0, R_1$

$\text{ADD } R_2, R_0$

$\text{ADD } R_3, R_2$

.....

程序段2的执行速度与硬件或编译是否对其执行顺序进行优化和优化方法有关。

(4) 数据处理速率PDR(processing data rate)法

因为不同程序中各类指令的使用频率不同, 所以固定比例方法存在很大局限, 且数据长度与指令功能的强弱对速度影响很大, 也不能反映计算机中cache、流水线、交叉存储等结构的影响。

数据处理速率PDR法采用计算“数据处理速率”

PDR值的方法来衡量机器性能。PDR值越大, 机器性能越好, PDR与每条指令和每个操作数的平均位数以及每条指令的平均运算速度有关, 计算方法如下:

$$\text{PDR}=\text{L}/\text{R}$$

其中: $\text{L}=0.85\text{G}+0.15\text{H}+0.4\text{J}+0.15\text{K}$

$$\text{R}=0.85\text{M}+0.09\text{N}+0.06\text{P}$$

式中: G是每条定点指令的位数;

H是每条浮点指令位数;

J是定点操作数的位数;

K是浮点操作数的位数;

M是平均定点加法时间;

N是平均浮点加法时间;

P是平均浮点乘法时间;

分子L所涉及参数的单位是位数(指令位数、数据位数),分母R所涉及参数的单位是时间(运算时间),可理解为单位时间处理的位数

(5) 基准程序测试法

基本思想是：

把应用程序中出现最频繁的那部分核心程序作为评价计算机性能的标准程序，在不同的机器上运行，测量其执行时间，作为各类机器性能评价的依据，称为基准程序Benchmark。

基准程序法是公认的测试性能较好的方法，有多种基准程序，如测试整数性能的基准程序(Dhrystone)、测试浮点性能的基准程序(Linpack、Whetstone)、以及SPEC基准程序测试等。

基准程序测试法反映机器持续性能，**主要指标有：**

① 算术平均值 A_m

即多个测试程序运算的算术平均值。

假设 n 个测试程序, R_i 表示第 i 个程序的执行速度, T_i 表示执行时间, 则算术平均值为:

$$A_m = \frac{1}{n} \sum_{i=1}^n R_i = \frac{1}{n} \sum_{i=1}^n \frac{1}{T_i}$$

如果用程序平均执行时间来表示性能, 则:

$$A_m = \frac{1}{n} \sum_{i=1}^n T_i$$

如果考虑不同程序在总任务中所占的时间比例不同 (用 w_i 表示比例), 则有加权平均值:

$$A_m = \sum_{i=1}^n w_i \times R_i = \sum_{i=1}^n w_i \times \frac{1}{T_i}$$

② 调和平均值

调和平均值用 H_m 表示, 则有:

$$H_m = \frac{n}{\sum_{i=1}^n \frac{1}{R_i}} = \frac{n}{\sum_{i=1}^n T_i}$$

加权的调和平均值可表示为:

$$H_m = \frac{n}{\sum_{i=1}^n \frac{w_i}{R_i}} = \frac{n}{\sum_{i=1}^n w_i \times T_i}$$

\rightarrow n 个程序在单位时间内所做平均工作

③几何平均值

设几何平均值用 G_m 表示, 则有:

$$G_m = \sqrt[n]{\prod_{i=1}^n R_i} = \sqrt[n]{\prod_{i=1}^n \frac{1}{T_i}}$$

加权的几何平均值可表示为:

$$G_m = \prod_{i=1}^n (R_i)^{w_i} = \prod_{i=1}^n \left(\frac{1}{T_i}\right)^{w_i}$$

— SPEC 基准测试程序

(System Performance Evaluation Corporation)

(系统性能评估测试)

由IBM、AT&T、BULL、Compaq、CDC、DG、DEC、Fujitsu、HP、Intel、MIPS、Motolola、SGI、SUN、Unisys等30余家厂商联合推出。

SPEC 基准测试的结果表示：

- SPECint 整数程序测试结果的几何平均值
- SPECfp 浮点程序测试结果的几何平均值
- SPECmark 所有测试结果的几何平均值

① SPEC89:

1989年10月首次宣布, 包含10个测试程序, 程序量超过15万行, 4个定点程序, 6个浮点程序; 测试结果用SPECint'89和SPECfp'89表示。

② SPEC92:

1992年又增加了10个测试程序, 共计6个定点程序和14个浮点程序, 测试结果用SPECint'92和SPECfp'92表示。

③ SPEC95:

1995年推出SPECint'95和SPECfp'95。

SPECint95: 采用8个真实应用, 包括仿真技术、人工智能、图像处理、压缩算法、编译器、解释器和数据库。

SPECfp95: 采用10个真实的应用来评测系统的单处理器的浮点运算性能, 应用包括流体力学、天气预报、量子物理、天文、电子等领域。

采用SPEC95对部分PentiumII、PentiumIII、Celeron的测试结果:

处理机	SPECint'95	SPECfp'95	处理机	SPECint'95	SPECfp'95
PentiumII 450	18.7	13.7	Celeron 366	14.1	10.70
PentiumIII 500	20.6	14.7	Celeron 400	15.1	11.20
PentiumIII 550	22.3	15.6	Celeron 433	16.1	11.60
Celeron 300A	12.0	9.66	Celeron 466	17.0	12.00

比较:

1992年SUN的Super SPARC: SPECint=40, SPECfp=45

④ SPEC2000:

SPECint2000:

在SPECint95的基础上增加为12个应用,包括:压缩算法、编译器、优化组合、棋类游戏、字处理、可视化、PERL语言、群论解释器、面向对象数据库和仿真技术。

SPECfp2000:

将SPECfp95的10个应用增加到14个应用,包括:量子运动、浅水模型、三维电势场、抛物线/椭圆偏微分方程、三维图像库、计算流体力学、图像识别/神经网络、地震波传播仿真、图像处理/人脸识别、计算化学、数论、有限元碰撞仿真、高性能物理加速器设计和污染分布计算等。

SPEC2000评价方法:

12个整数程序使用C或C++语言,不使用CPU的浮点单元

14个浮点程序,使用 FORTRAN 77/90 和C语言,主要运算是浮点数。

取值与之前的SPEC有较大区别。参考平台为Sun工程工作站Ultra5/ 或Ultra 10 300MHz。

$$\text{SPEC值} = \frac{\sum_{n=1}^n \frac{\text{1个程序在Sun工作站上的运行时间}}{\text{在被评测机上的运行时间}} \times 100}{n}$$

⑤ SPEC2006

CPU2006是SPEC标准的CPU精密基准测试集。

CPU2006将逐渐完全取代CPU2000, 成为评估

Windows、Linux下部件、系统性能的新标准。

分为测试整数性能的CINT2006和测试浮点性能的CFP2006两个部分, 分别包含12个测试场景和17个测试场景, 均为基于应用程序的测试。

以**Sun Ultra Enterprise 2**工作站作为基准参考系统, 以此为参考, 其他测试系统与之相比得出相对性能指数。该工作站使用UltraSPARCII处理器(296MHz)。

无论是计算机的厂商还是计算机用户都需要有某种方法来衡量计算机的性能,作为设计、生产、购买和使用的依据。但是由于计算机系统是一个极其复杂的系统,其体系结构组成和实现都有若干种策略;而且其应用领域也千差万别,所以很难找到统一的规则或标准去评测所有的计算机。所以还需不断努力,在理论和实践中去总结归纳。