Lab1-处理器性能评测

何昊 1600012742

1. 我们针对系统评测的不同角度会采用不同的评测程序。在目前已有的评测程序中，为下列评测目标找到某些合适的评测程序（列出即可）。

|  |  |
| --- | --- |
| CPU整点性能 | SPEC int |
| CPU浮点性能 | SPEC fp |
| 计算机事务处理能力 | TPC[[1]](#footnote-1) |
| 嵌入式系统计算能力 | EEMBC[[2]](#footnote-2) |
| 2D处理能力 | GFXBENCH2D[[3]](#footnote-3) |
| 3D处理能力 | SPEC viewperf |
| 并行计算性能 | SPEC OMP, NAS Parallel Benchmarks[[4]](#footnote-4) |
| 系统响应速度 | TPC-C(Database), Speedometer(Web)[[5]](#footnote-5) |
| 编译优化能力 | 使用其他评测程序，对不同编译器编译同样代码生成的程序进行性能评测 |
| 操作系统性能 | HBench-OS[[6]](#footnote-6) |
| 多媒体处理能力 | ALPBench[[7]](#footnote-7) |
| IO处理能力 | IOMeter[[8]](#footnote-8) |
| 浏览器性能 | BrowserBench[[9]](#footnote-9) |
| 网络传输速率 | Speedtest[[10]](#footnote-10) |
| Java运行环境性能 | SPECjvm2008 |
| 邮件服务性能 | SPECmail2009 |
| 文件服务器性能 | SPEC SFS |
| Web服务器性能 | SPECweb2009 |
| 服务器功耗和性能 | SPEC SERT |

1. 阅读文献（Reinhold P.Weicker, An Overview of Common Benchmarks, IEEE Computer, December 1990.）并回答下面的问题
2. 简述用于性能评测的MIPS指标之含义，以及它是如何被计算的。

MIPS是Million Instructions Per Second的缩写，其原始含义是指一个CPU每秒能够执行的百万指令数量。然而，在RISC体系结构出现后，RISC和CISC计算机的性能无法直接使用每秒执行的指令数量来客观地比较。因此，另一种常见的计算方法被称为VAX MIPS，也就是说，如果一台计算机执行同样程序能够比VAX11/780计算机快X倍，它的性能就是X VAX MIPS。对于实际的计算机生产商而言，他们在商业上所宣称的MIPS可能是以上两种之一，也可能是一些其他的计算方法，可信度不高。因此，工业界目前并没有一个被广泛接受的MIPS标准与计算方法。

1. 使用linux下的剖视工具（例如gprof）对dhrystone和whetstone进行剖视，参考论文Table 1形式给出数据，你的结果和该论文是否一致，为什么？
2. 对Whetstone的剖析

首先，在编译whetstone的过程中，必须将相关数学库外部链接进whetstone程序，然而gprof并不支持对共享库进行剖析。因此，我修改了whetstone的源代码，对所有的三角函数和数学函数进行了手动封装，从而获得了剖析结果。

Procedure profile for Whetstone

|  |  |  |
| --- | --- | --- |
| Procedure | Percent | What is done here |
| Main program | 44.31 |  |
| p3 | 17.99 | FP arithmetic |
| p0 | 12.49 | Indexing |
| pa | 16.25 | FP arithmetic |
| User code | 91.04 |  |
| Trigonometric functions | 4.57 | Sin, Cos, Atan |
| Other math functions | 4.04 | Log, Exp, Sqrt |
| Library functions | 8.61 |  |
| Total | 99.62 |  |

我获得的结果与论文里的有很大不同。首先，用户代码所占据的比例远远大于库函数，尽管库函数被调用的次数与之相差无几。这可能是因为在这几十年中，对数学函数进行数值计算的方法得到了很大的改进，有的甚至得到了硬件的支持，使得诸如开方，三角函数等运算的速度远远快于1990年的水平。此外，在用户代码中，PA函数占据的时间远远高于论文里的结果，P3, P0和PA三个函数在我的运行结果中运行时间大致相同。根据查看反编译汇编的结果，PA的指令数目比另外两个函数高，还需要用到更多的乘法和除法指令，因此我认为在原论文里PA运行如此之快(只占据1.6%的时间)相当不合理。对于三角函数函数和其他数学函数之间的调用比值，与原论文大致相同。

1. 对Dhrystone的剖析

类似前述方法，由于gprof并不能直接剖析位于共享库内的strcpy和strcmp，故我自己直接实现了strcpy和strcmp函数，替代掉原来的库函数调用，得到了如下的剖析结果。

Procedure profile for Dhrystone

|  |  |  |
| --- | --- | --- |
| Procedure | Percent | What is done here |
| Main program | 7.25 |  |
| User Procedures | 35.22 |  |
| User code | 42.47 |  |
| Strcpy | 29.31 | String Copy |
| Strcmp | 28.22 | String Comparison |
| Library functions | 57.53 |  |
| Total | 100 |  |

Dhrystone的剖析结果也与原论文中的有所不同。Main函数与其他用户函数的执行时间的比值与原论文相同。而最大的异常在于对于Strcpy和Strcmp的时间占用出奇之高。这应该是由于简陋版本的strcpy和strcmp实现效率过低所致。

1. 论文中讨论了处理器之外可能对性能造成影响的因素，请分别使用两种不同的语言（例如C和Java）使用同一算法实现快速排序、矩阵乘法、求Ackermann函数，验证文中的观点。（请保留你的程序，我们在后面可能还会用到它）
2. **性能评测**

**基于某个给定的计算机系统平台，使用dhrystone、whetstone、SPEC CPU2000开展评测、分析、研究并给出报告。**

**计算机组织与体系结构实习Lab 1**

**处理器性能评测报告模板**

1. **工作背景和评测目标**
2. **评测环境**

|  |  |
| --- | --- |
| **项目** | **详细指标和参数** |
| 处理器型号及相关参数（频率、架构、缓存等） |  |
|  |  |
| 内存 |  |
| 外存 |  |
| 操作系统及其版本 |  |
| 编译器版本  （及编译参数） |  |
| 库函数及其版本 |  |

1. **评测步骤及要求**
2. 在linux下基于dhrystone-2.1所提供的Makefile编译dhrystone
3. 分别采用108、3\*108、5\*108、7\*108、9\*108为输入次数，运行编译生成的两个程序，记录、处理相关数据并做出解释。
4. 对dhrystone代码做少量修改，使其运行结果不变但“性能”提升。
5. 采用dhrystone进行评测有哪些可改进的地方？对其做出修改、评测和说明。
6. 在linux下使用编译器分别采用-O0、-O2、-O3选项对whetstone程序进行编译并执行，记录评测结果。
7. 分别采用106、107、108、109为输入次数，运行编译生成的可执行程序，记录、处理相关数据并做出解释。
8. 进一步改进whetstone程序性能（例如新的编译选项），用实验结果回答。
9. 完成SPEC CPU2000的安装。
10. 修改自己的config文件，分别用低强度优化（例如O2）和高强度优化（例如O3）完成完整的SPEC CPU2000的评测，提交评测报告文件。
11. **评测结果及简要分析（表格样式可自己调整）**

|  |  |  |  |
| --- | --- | --- | --- |
| **评测程序** | | **评测结果** | |
| 配置1 | 配置2（可增/删） |
|  | |  |  |
| **Dhrystone** |  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |
| **Whetstone** |  |  |  |
|  |  |  |
|  |  |  |  |
| **SPEC2000 INT** |  |  |  |
|  |  |  |
| **SPEC2000 FP** |  |  |  |
|  |  |  |  |
|  |  |  |  |

**性能评测分析：**

1. 从xx程序的实验数据来看……
2. xx处存在xxxx问题，因为……
3. ……
4. **小结**

略。

1. http://www.tpc.org [↑](#footnote-ref-1)
2. https://www.eembc.org [↑](#footnote-ref-2)
3. http://hdrlab.org.nz/benchmark/gfxbench2d/ [↑](#footnote-ref-3)
4. https://en.wikipedia.org/wiki/NAS\_Parallel\_Benchmarks [↑](#footnote-ref-4)
5. https://browserbench.org/Speedometer2.0 [↑](#footnote-ref-5)
6. https://www.eecs.harvard.edu/margo/papers/sigmetrics97-os/hbench/ [↑](#footnote-ref-6)
7. Li, Man-Lap, et al. "The ALPBench benchmark suite for complex multimedia applications." IEEE International. 2005 Proceedings of the IEEE Workload Characterization Symposium, 2005.. IEEE, 2005. [↑](#footnote-ref-7)
8. http://www.iometer.org [↑](#footnote-ref-8)
9. https://browserbench.org [↑](#footnote-ref-9)
10. https://www.speedtest.net [↑](#footnote-ref-10)