ISSN 1000-0054 清华大学学报（自然科学版）2005 年 第 45 卷 第 10 期 34/36
CN 11-2223/N J Tsinghua Univ (Sci & Tech), 2005, Vol. 45, No. 10 1429-1432

基于资源动态变化的桌面 Linux 交互性能测试方法

吴邦欲[1]， 张 昀[2]， 顾 明[1]， 孙家广[1]

（1. 清华大学 计算机科学与技术系，北京 100084；2. 共创开源软件有限公司，北京 100083）

摘 要：传统的性能测试方法或者仅关心操作系统或CPU的局部吞吐量性能，或者仅关注应用软件的使用性能，不能反映Linux系统对桌面交互应用的影响。该文提出了基于动态变化的系统资源环境，利用事件处理延迟时间对Linux桌面应用进行交互式性能测试的方法。建立了桌面应用的静态模型，改进了已有的动态模型，完成了在不同的内存负载下应用软件启动交互性能测试。基于该方法的测试结果表明Linux桌面应用交互性能在软件启动方面不佳，在桌面应用环境下的swap效率较低。由于Linux体系结构和算法源自传统Unix，先天比较适合服务器。因而Linux要成为性能卓越的桌面操作系统，除了应用软件自身需要改进外，系统资源的管理调度也需要优化。

关键词：操作系统；交互性能；桌面应用；启动延迟

中图分类号：TP 316.8 **文献标识码**：A

文章编号：1000-0054(2005)10-1429-04

A methodology for interactive performance measurement in desktop Linux based on dynamic resource environment

WU Bangyu[1], ZHANG Yun[2], GU Ming[1], SUN Jiaguang[1]

(1. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;
2. Co-Create Open Source Software Corporation Limited, Beijing 100083, China)

Abstract: The conventional methodology for Linux performance measurements focus only on the OS, the CPU, or applications, so they can not demonstrate the influence of the OS on the application performance. This paper presents a methodology using event handling latency to measure the interactive performance of applications in a dynamic resource environment. A static model of the desktop application was built and the existing dynamic model was improved. The experimental results on startup latency of applications measured with dynamic free memory show that both the startup latency and the swap efficiency are poor. Linux provides poor interactive performance for desktop applications with the architecture and algorithm designed for the traditional Unix server. Therefore, Linux must be optimized to improve resource management for desktop application.

Key words: operating system; interactive performance; desktop application; startup latency

随着Linux在桌面应用领域的推进，逐步暴露出Linux对桌面应用的不适应性，特别是交互性能欠佳，用户使用不流畅。原因在于Linux在体系结构和算法上秉承了传统的适合服务器的Unix操作系统，因而在桌面领域不能很好地发挥性能。

桌面应用是典型的交互式应用，系统资源动态变化时的交互性能反映了桌面应用的持续运行性能，事件处理延迟是交互性能的重要表现。目前缺乏专门针对Linux桌面交互性能的研究，利用传统的测试方法[1-5]不能有效分析Linux操作系统的结构和算法对桌面应用交互性能的影响。首先大部分测试侧重在系统吞吐量，关注Cache、TLB等与CPU体系结构相关的性能以及文件访问、网络传输、系统调用延迟等局部微观性能，不是用户所感受到的实际交互性能[1,2,4,5]。其次，有的测试目标尽管是交互性能，但也仅仅限于应用软件的使用性能，测试在静态的系统环境下进行，不能动态反映系统资源变化对交互性能的影响[3]。另外，利用CPU空闲区间[3]或事件集合状态[6]间接获取延迟时间的测试手段容易受进程调度和I/O量大等行为复杂事件的影响，不适合测试延迟时间长的交互式事件。这些问题导致测试结果不能有效指导设计人员从最终用户的角度同时优化系统与应用。

本文提出了在动态的系统资源环境下，利用延迟对Linux桌面应用进行交互式性能测试的方法，建立了桌面交互应用的静态模型，改进了已有的动态模型[3]，以不同的内存负载下应用软件启动交互性能为例验证了测试方法。

收稿日期：2004-09-13

基金项目：国家“八六三”高技术项目（2003AA414031）

作者简介：吴邦欲(1973-)，女(汉)，湖北，博士研究生。

通讯联系人：孙家广，教授，E-mail：sunjg@mail.tsinghua.edu.cn

1 桌面应用动态特性和交互模型分析

桌面系统是典型的人机交互系统。与服务器系统相对固定的运行模式、相对稳定的服务软件相比，桌面用户需要频繁运行和终止应用软件，即使对相同软件的操作行为，因时因地差异也很大，因而系统行为的可预测性差，处理事件比传统的批处理系统离散性强。从宏观的角度，系统不可能根据用户当前处理的事件预测下一阶段要处理的事件；从微观角度，由于事件之间的相互独立性和不可预测性导致指令级的分支预测功能、指令 Cache 的使用、指令 TLB 等 CPU 体系结构的作用发挥受到一定程度的影响。

因此只有在动态的资源环境下的性能测试才能真实反映桌面应用的实际使用效果。

1.1 桌面交互应用静态模型

桌面图形环境下的应用软件以菜单的形式展示所提供的功能，菜单树可以静态地表示交互式桌面应用，见图 1。每个菜单项可看作菜单树的一个节点，根节点 A 表示应用本身，非叶节点 M_i 表示各层子菜单，叶节点 E_i 表示经过若干层菜单索引后软件要处理的目标事件。应用软件能处理的事件可采用人工驱动的深度优先搜索获得，非叶节点即事件索引路径节点。

一般，广而浅的菜单树优先于窄而浅的菜单树，菜单树高度(depth)较小，而树的度(degree)则远远大于高度，是一棵“矮”而“胖”的树。为了让用户“所见即所得”，减少事件搜索时间，为常用操作设置了“快捷键”，则菜单树的非叶节点被动态剪枝，这些树枝的叶节点便过继给父节点。应用软件与菜单树映射关系模型表示如图 1 所示。

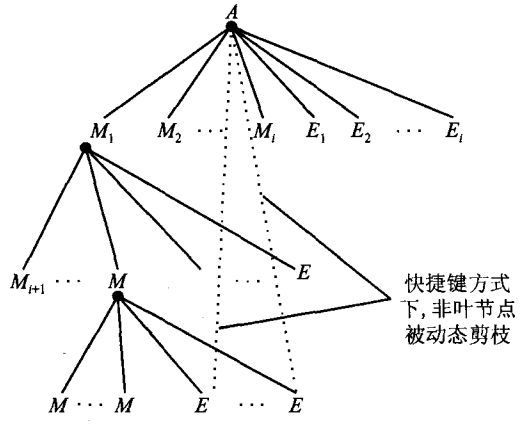

图 1 应用软件与菜单树映射关系模型

定义 F_a 为软件能处理的所有事件集合，即

$$F_a=\{E_1,E_2,\cdots,E_n\},\quad n \text{ 为事件总数.}$$

对某一类桌面用户而言，使用软件的轮廓(profile)基本是固定的，对应的搜索树只是整棵菜单树的一棵子树。定义 F_i 为第 i 类用户需要请求的交互事件集合，U_a 为所有用户都需要请求的交互事件集合，即

$$U_a = F_1 \cap F_2 \cap \cdots \cap F_k,\quad k \text{ 为用户类总数.}$$

显然 U_a 中的事件延迟具有普遍性，对这一类事件的测试分析更有助于改善系统整体交互性能。

1.2 桌面交互应用动态模型

动态模型首先与用户行为密切相关。基于前面分析的静态模型，在已有的动态模型的基础上[3]增加了事件索引时间。将用户请求一个交互事件的动态行为细化为一系列操作，即：

1) 逐层索引菜单，定位事件；

2) 提出事件处理请求并等待事件完成；

3) 用户思考贯穿整个过程。

定义

$$T_{\text{total}}=T_{\text{index}}+T_{\text{think}}+T_{\text{handle}},$$

式中：T_{total}为事件总延迟时间，T_{index}为事件索引时间，T_{think}为用户思维时间，T_{handle}为事件处理时间。

动态模型其次受系统资源动态变化的影响，不同的资源约束条件产生不同的 T_{total}。只有在 T_{total}超过了用户期望值，用户才能感知到延迟的存在[3]。随着延迟的增大，用户感受到的延迟时间会“大于”实际存在的时间值。此时，用户感觉到的延迟时间为：

$$T_{\text{user}}=\begin{cases}0, & T_{\text{total}}\leqslant T;\\ (T_{\text{total}}-T)^{\beta}, & T_{\text{total}}<T.\end{cases}$$

其中：T 为用户对某一事件延迟的期望值，T 是事件 E_i 的函数；β 为反映用户感觉的指数函数。

因此，测试研究的重点应放在超出用户期望的关键事件上，只有这部分事件才是影响 Linux 交互性能的关键所在。T_{index}和 T_{think}的研究涉及到统计学以及人机工程，T_{handle}是本文测试的重点。

2 内存资源动态变化时启动延迟性能测试

2.1 测试方法

基于对桌面交互应用静态和动态模型的分析，本文测试内存负载动态变化时关键应用的是启动延迟交互性能。原因如下：

1) 站在用户的角度，“启动”事件的延迟决定用户对应用软件的第一感觉，桌面用户经常需要启动软件，而服务器应用则不必要。从静态模型分析得到

的集合 U_a 中分离出高延迟事件集合 U_{ah}。

U_{ah} = {启动，存盘，打开文件，打印 ……}.

其中“启动”是所有用户向应用提出的第一个请求，但在实际测试时很容易受到忽视。

2) 从 Linux 系统的实际情况出发，在使用过程中，用户发现桌面应用的启动延迟很大，应用软件的设计者经常使用过渡性界面显示来分散用户注意力，如 Evolution、Openoffice 等。来自开发社区的信息表明，大型应用软件的设计者已经意识到了启动延迟存在的问题。

3) 站在系统分析的角度，软件启动能全面地反映系统性能。由于软件启动涉及到系统的诸多环节，如目标文件大小、共享库布局、文件系统性能、内存管理算法、进程调度算法等因素，测试启动延迟有助于分析 Linux 操作系统对桌面应用的适应性。

4) 站在测试的实用性角度，由于桌面环境以图形界面为主，窗口环境下的工作集很大，交互性能对内存的动态变化相对敏感；磁盘访问速度的提升远远落后于 CPU，内存成为影响性能的瓶颈。

充分利用 Linux 有源代码的优势，在软件启动开始和完成后分别加入计时直接获取时间延迟，比文[3,6]测量抗干扰能力强，结果精确。内存的压力以 swap 空间值来反映，swap 值越大，内存越紧张。

测试系统的硬件环境为 2.4 GHz Pentium 4 处理器，80 GB 的 Maxtor 6Y080LO 硬盘，RAM 主存容量 512 MB。操作系统为 RedHat9.0 简体中文版，内核版本 kernel2.4.20-8。表 1 给出了应用软件 Benchmark 的情况描述。

表 1 测试采用的应用软件 Benchmark

应用软件	版本	信息描述	文件大小 MB	初始化后占用内存 MB	系统引导完成时的启动延迟 s
Xemacs	21.2.1	软件开发和工作环境	4.09	1.52	107
Xpdf	2.0.1	Pdf 文件阅览器	0.96	0.01	12
Evolution	1.2.2	邮件客户端	0.47	9.52	113
Openoffice	1.0.2	Openoffice 办公套件	0.24	6.45	220

2.2 测试结果和分析

图 2 至图 5 分别反映了系统有 swap 行为时软件的启动延迟性能，内存资源足够时的启动时间如表 1。测试结果定量验证了 Linux 的交互性能不佳，也说明利用延迟测试交互性能的方法是正确的。

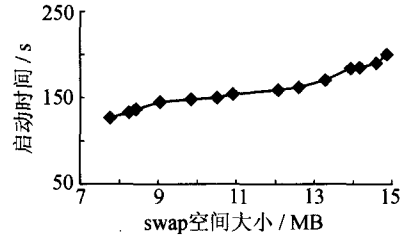

图 2 Xemacs 启动延迟性能测试

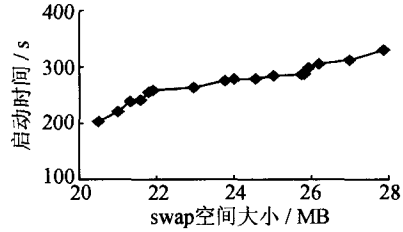

图 3 Evolution 启动延迟性能测试

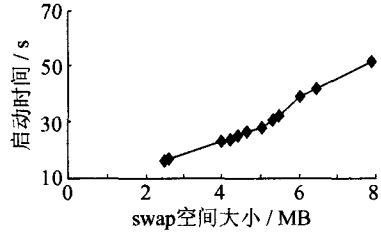

图 4 Xpdf 启动延迟性能测试

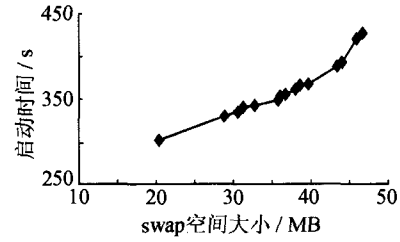

图 5 Openoffice 启动延迟性能测试

Linux 系统下的桌面应用软件启动速度相当慢，启动延迟过大，并且随着内存压力的增加而增加。速度最快的是 Xpdf，12 s，Openoffice 最慢，高达 220 s，最常用的应用软件的启动延迟均超过了 100 s，远远超出了用户期望。

针对关键事件的交互性能测试有助于发现应用软件自身的问题。对照表 1 分析发现，启动延迟主要受初始化时软件占用内存资源的影响，其次受目标文件大小的影响。初始化占用内存越多，意味着与内存相关的初始化工作越多，相应地存储访问指令也越多，启动延迟越大；在同等紧张的系统内存下，启动触发的 swap 操作越多，启动延迟时间增长越快，

Evolution、Openoffice 的测试结果说明了这个问题。软件目标码越大,代码加载及指令执行时间越长,导致启动延迟大,Xemacs 是这种情况的典型。应用软件需要在内存分配、目标码布局方面有所调整优化。

系统资源动态变化环境下的交互性能测试除了反映交互事件对资源变化的敏感,也有助于分析操作系统资源管理调度机制对交互应用的支持。测试发现,swap 效率低,进一步恶化了应用软件的启动速度。Linux 操作系统的 swap 操作首先由存储管理模块中的核心算法决定需要 swap 读或写的页面,然后将读写请求交给负责 I/O 操作的 swap 设备管理模块。设 T_{swap} 为页面 swap 读写的总时间,T_{mm} 为存储管理模块的处理时间开销,$T_{I/O}$ 为 swap 模块的处理时间开销,$T_{startup}$ 表示启动延迟,T_0 表示没有 swap 发生时的启动时间,则有

$$T_{swap} = T_{mm} + T_{I/O}; \quad \text{且 } T_{swap} = T_{startup} - T_0;$$

根据上述两式得到:$T_{mm} = T_{startup} - T_0 - T_{I/O}$;

则 T_{mm} 在 swap 操作中占用的时间开销比例为

$$e = (T_{startup} - T_0 - T_{I/O})/(T_{startup} - T_0).$$

时间开销分析以与 swap 空间相当的大块文件突发写访问性能为参照,以延迟最小和最大的 Xpdf、Openoffice 为例,如图 6 和图 7 所示。比较图 4 与图 6、图 5 与图 7,可见对于大小相同的磁盘文件操作,swap 比普通文件速度慢得多,可见 swap 功能的软件开销之大。

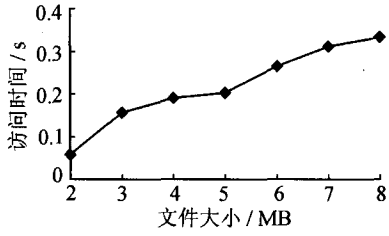

图 6 与 Xpdf 对应的文件操作性能

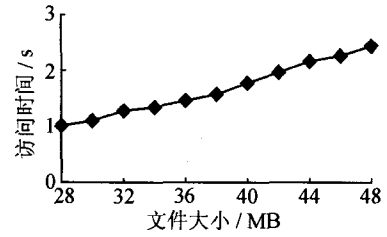

图 7 与 Openoffice 对应的文件操作性能

为了与其他系统有一个横向比较,采用相同的测试方法对 Windows2000 做了测试,结果发现:Windows 的桌面软件启动速度远远高于 Linux,Word 仅需要 4～5 s。当窗口最小化时,对应的空间大部分被转移到 swap 缓存,当内存紧张时,缓存中的空间立即被释放,同时以进程为对象将长时间内用户没有使用的空间迅速换出到磁盘,因此 Windows 针对桌面应用的 swap 效率比较高。

测试还发现,Linux 软件在启动后的交互性能不差,这表明,应用软件作为独立的个体本身,其性能表现卓越,但由于软件启动涉及到的因素较多,包括系统层面和应用层面,任何环节间的不协调都会对启动产生负面影响。这一事实说明了依托社区的分散式软件开发模式存在一些问题,Linux 缺乏统一的标准、明确清晰的战略规划。Linux 要成为性能卓越的桌面系统,除了应用软件自身需要改进外,系统资源的管理调度也需要优化。

3 结 论

针对 Linux 桌面应用交互性能差、传统测试方法无法将系统与应用性能有机结合的现状,提出了在动态变化的系统资源环境下、利用延迟对 Linux 桌面应用进行交互式性能测试的方法。利用该方法,能进一步测试 CPU 等其他系统资源动态变化时的桌面交互性能,而且针对延迟大、对资源敏感的交互事件更有意义。进一步的研究包括在测试方法的指导下建立一套桌面应用交互性能度量体系。

参考文献 (References)

[1] Lee D, Crowley P, Baer J et al. Execution characteristics of desktop applications on Windows NT [A]. Proc 25th Annual International Symposium on Computer Architecture [C]. Barcelona, Spain: ACM Computer Society, 1998. 27－38.

[2] Chen J, Endo B, Chan Y, et al. The measured performance of personal computer operating systems [A]. Proc 15th Symposium on Operating Systems Principles [C]. Colorado, USA: ACM Transactions on Computer Systems, 1995. 299－313.

[3] Endo Y, Wang Z, Chen J, et al. Using latency to evaluate interactive system performance [A]. Second USENIX Symposium on Operating Systems Design and Implementation [C]. Berkeley, USA: USENIX Association, 1996. 185－199.

[4] Santa C. SYSmark for Windows NT [M]. NJ, USA: Press Release by IDEAS International, 1995.

[5] The Standard Performance Evaluation Corporation. SPEC CPU2000 V1.2 [EB/OL]. http://www.spec.org.

[6] Kristiān Flautner, Rich Uhlig, Steve Reinhardt. Thread-level parallelism and interactive performance of desktop applications [A]. Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems [C]. MA, USA: ACM SIGPLAN Notices Archive, 2000. 129－138.