第42卷 第6期 西 安 交 通 大 学 学 报 Vol.42 №6
2008年6月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Jun. 2008

支持多种Linux版本的动态内核性能测试技术

冯国富，魏恒义，储鹰，董小社
（西安交通大学电子与信息工程学院，710049，西安）

摘要：基于Intel IA32架构，提取了相对稳定的Linux操作系统框架.该框架主要集中于源代码树中的/arch目录中，由一些简短、高效的与体系结构相关的汇编代码构成.据此，设计了性能测试方案，包括基于地址表格改写的入口插接方案和基于代码拼接的出口插接方案.方案在运行的操作系统中，通过动态改写系统内存的插接技术实时地插入测试代码，测试代码可以使用高级语言来书写.实验表明，所提方案适用于通用的Linux 2.4和Linux 2.6内核系列，其时间开销仅为基于中断动态插接技术的6%左右，且能发现系统硬件和操作系统的关键性能问题.

关键词：操作系统内核；性能测试；动态插接
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2008)06-0674-05

Dynamic Instrumentation Technology for Kernel Performance Evaluation Supporting Multi-Version Linux

FENG Guofu, WEI Hengyi, CHU Ying, DONG Xiaoshe
(School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: According to the IA32 architecture and the source code which resides in the directory /arch under the Linux source code tree, a stable operating system framework attaching to the architecture tightly is abstracted. Then a dynamic instrumentation technology based on the framework is proposed for the Linux kernel performance evaluation. The major scheme of the performance evaluation technology includes both the entry and the exit instrumentation techniques by means of address table rewriting and code splicing respectively. The evaluation codes based on this technology can be written in a high level language and be deployed into the system memory dynamically. The experiment results show that the technology is suitable for both 2.4.x and 2.6.x series kernels, overhead of the kernel performance evaluation tool based on this technology is only about 6% of that based on interrupt mechanism, and the technology can be applied to find performance problems caused by either the hardware or the operating system kernel.

Keywords: operating system kernel; performance evaluation; dynamic instrumentation

当前，计算机系统常包含多执行单元、高速缓存、指令调度、分支预测及多级缓存等功能，编译又多采用复杂的编译规则，使得系统的性能更加依赖于统计因素，难以用预测的方法来全面评价，有时甚至产生难以理解的性能问题[1]，由此通过性能的实际测试对性能进行评价显得更为重要[2].

Linux操作系统内核性能测试主要是对系统调用、中断和异常进行测评.一般的性能测试工具，如LMbench[3]、Gprof等，常忽略对中断和异常的测试，而中断和异常对系统性能影响很大，且具有随机性和隐蔽性[2].

目前，用于内核性能测试的技术可分为内核补丁、集成于内核及内核动态插接等3种，它们通常依赖于特定内核版本，前2种还需重新编译内核，安装

收稿日期：2007-09-06. 作者简介：冯国富(1971—)，男，博士生；董小社(联系人)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2003AAIZ2610，2004AA111110，2006AA01A109).

和测试过程繁琐. KProbe(Kernel Dynamic Probe)是一种在 DProbe 基础上开发的基于内核补丁的探测工具，SystemTap 在 KProbe 的基础上又做了进一步封装. 一些基于内核补丁的工具，会有选择地集成于内核，如 KProbe. 内核动态插接技术的主要优点是被测系统无需重新配置和编译内核，安装及卸载比较方便. GILK[4]是一种在 IA32 平台上运用运行代码拼接技术的内核调试与性能监视工具，但它仅支持 Linux 2.0.x～Linux 2.2.x 的内核版本.

本文基于动态插接技术研究了一种可以运行于当前 Linux 2.4 及 Linux 2.6 系列内核版本的低开销、通用的内核性能测试技术.

1　操作系统功能框架提取

硬件的稳定发展决定了操作系统中存在一个相对稳定的框架. 基于 IA32 架构的 Linux 2.4 及 Linux 2.6 系列内核的抽象总结如图 1 所示，其中的阴影部分是操作系统的功能实现部分，常用与平台无关的 C 语言来实现，而其他部分为相对稳定的操作系统功能框架，用与架构相关的汇编语言来实现.

1.1　操作系统功能实现

Linux 内核的发展主要体现在以下几方面.

(1)功能扩展. 增加和完善系统调用，如 Linux 1.0.9 只有 134 个系统调用，而 Linux 2.4.7、Linux 2.4.35 和 Linux 2.6.22 分别有 221、239 和 323 个系统调用.

(2)完善设备管理，支持新设备. 例如，从 Linux 2.6.12 版本开始，Linux 支持可信平台模块(TPM)芯片，并对 DVB、USB、网络和声卡等驱动程序进行升级.

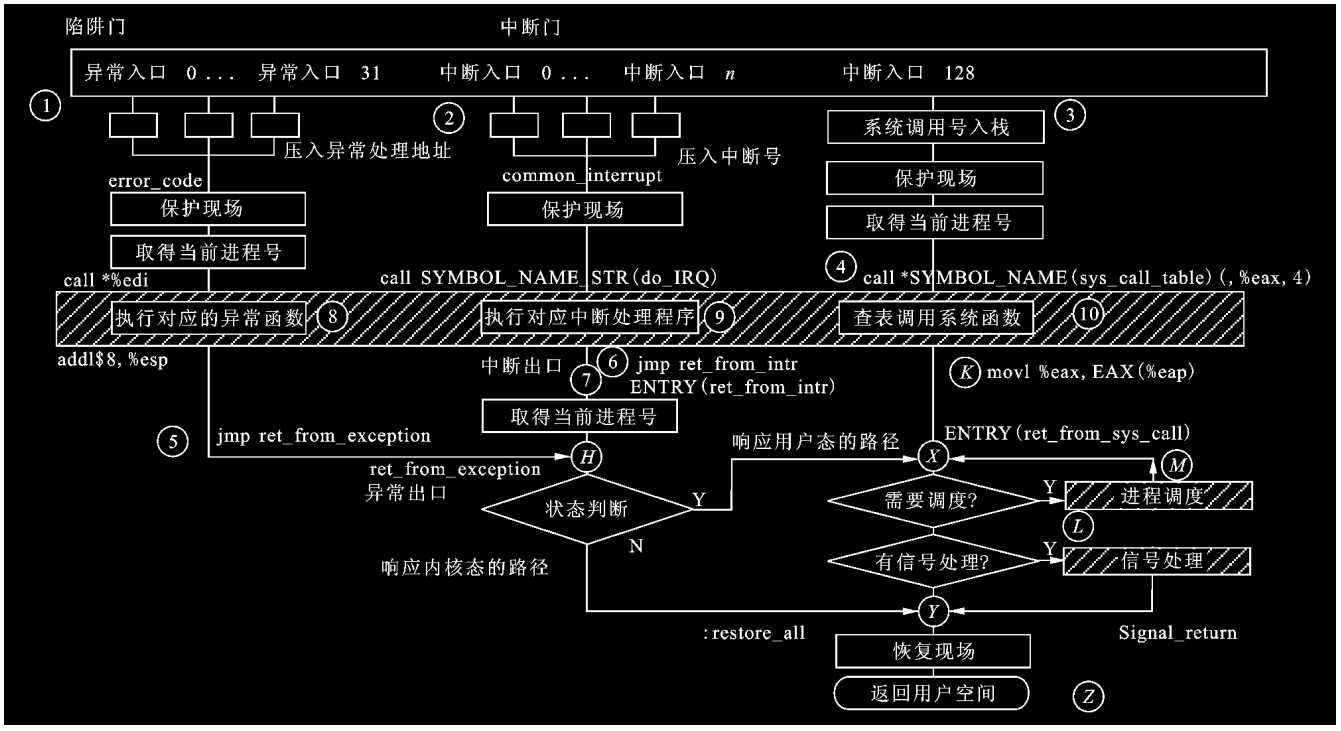
(3)辅助功能. 例如，Linux 2.6.12 改进了 IPv6、SELinux、软挂起和设备映射等功能，Linux 2.6.18 改进了内核源代码管理方式，完善了 SELinux.

上述的改进部分体现在图 1 中的阴影部分，它们不会影响操作系统功能框架.

1.2　操作系统功能框架

为了保护用户在软件上的投资，硬件架构发展应保持相对稳定，如 IA32 架构近年来虽然经历了 P6、Netburst、Core 等多种微架构的变更，但其基本架构是稳定的. 操作系统功能框架(图 1 中非阴影部分)是进入和退出 Linux 内核的控制核心，主要集中在源代码树中的/arch/i386 目录之下，是一些简短、高效的汇编代码. 分析表明，这部分代码在 Linux 2.4 和 Linux 2.6 等多种版本中保持相对稳定，只在部分细节和表达方式上有所变化.

在上述框架上可以设计跨内核版本的内核性能测试，在相对稳定的功能框架上对变动、发展的内核进行测试也符合测试的内在要求.



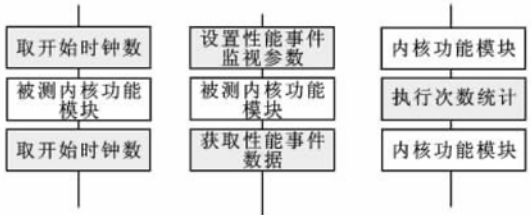
从内核 2.6.1 起，始标号⑦与 H 交换了位置，并根据是否为抢占内核在其间会插入一个关中断指令

图 1　Linux2.4、Linux 2.6 内核框架示意图

2 功能框架下的动态内核测试技术

2.1 内核性能数据获取技术

本文通过在内核待测功能模块的入口和出口插入测试代码来完成测试任务，并将测试代码定义为测试装置(图2中阴影部分)．测试装置包括时间开销、性能事件数据、统计执行次数3类，如图2所示．这些数据主要从Intel提供的硬件监视单元中获取，如Pentium4提供了48个事件的检测器和18个事件的计数器，时间开销可从TSC寄存器读取．



(a)时间开销 (b)性能事件数据 (c)统计执行次数

图2 内核性能测试装置

2.2 基于系统内存改写的低开销动态插接技术

IA32提供了动态插入代码的方法，如INT 1和INT 3，但其中断机制处理过程繁琐、时间开销大．内核测试中的每个功能模块往往是一个函数或代码段，内核进入这些模块通常使用跳转或调用指令．通过动态修改这些跳转或调用指令的目标地址可改变程序流向插入代码．这样做无需经过中断机制，因此时间开销比较小．

本文通过综合Hack中断描述符表(IDT)、Hook系统调用、Linux动态内核补丁及文献[4]的代码拼接技术，设计了一种用于内核性能测试的动态插接方案．插入装置可使用高级语言来书写．

2.3 测试装置安装方案

2.3.1 基于地址表格改写的入口测试装置安装

在IA32架构下，Linux主要通过中断门和异常门经安全检查进入内核，通过修改IDT控制表格内容、改变程序流向达到在IO中断和异常入口安装测试装置的目的(见图1中的①②③)．通过sidt指令可读取中断描述符表寄存器(IDTR)的值，由中断号可进一步求出每个中断入口的地址．

系统调用入口，可在总入口处(见图1③)使用上述方法，也可采用系统调用劫持技术，通过修改原系统调用函数表格中的系统调用函数地址，达到对单独系统调用插入代码的目的(见图1④)．

2.3.2 基于代码拼接的出口测试装置安装 各中断、异常和系统调用的内核出口子路径通过跳转指令汇总为总出口路径，而总路径上安装测试装置(见图1)对系统性能影响较大，因此可在子路径向总路径跳转处安装测试装置．

在中断出口插入测试装置有二种方案，一种是在jmp ret_from_intr处(见图1⑥))插入，另一种是在中断的后半部bottom half结束处插入．以前者为例，每个中断出口用一个5 B的长跳转指令(如图1的⑥ 跳向⑦)，可修改该长跳转指令的目标地址将内核流程指向测试装置，测试完成后再跳回标号⑦．

异常出口是一个总出口(见图1⑤)，它使用了一条3 B的短跳转指令，只能在相对的256 B范围内跳转，需在该指令的256 B范围内安装一条5 B的长跳指令，并作为跳向测试装置的跳板．一些不常用的异常入口或错误处理代码段均可作为跳板的位置(非抢占的Linux 2.6内核异常出口可以不用跳板)．

在系统调用出口(见图1④)，可以参考缓冲区溢出原理，将内核堆栈中原系统调用返回地址修改为测试装置地址插入测试代码．各种事件计数点可以安装在各个出口点上，Linux 2.6和Linux 2.4判断中断号的方法有所区别，但其编码并不重叠，所以还是可以正确区分．

2.4 内核地址定位

不同版本或不同编译选项的内核映像内存地址分布是有差异的．对于System.map中没有的地址标号，可用GDB读入/boot/vmlinux文件对该段内存的内核代码进行静态反汇编得到地址信息，也可通过IDTR获取IDT表地址，再由该表中的地址信息通过内存关键词来搜索其他地址点．许多Hook系统调用的黑客软件，均采用这种技术来确定系统调用表的地址，攻击不同内核版本的主机．

2.5 不依赖于内核版本的测试方案分析

分析、对比和实验表明，在Linux 2.4.7～Linux 2.6.22版共计50余种标准内核版本中，就功能框架而言，这些内核版本有26个版本是与其上一个版本完全相同的，5个版本仅是增加、完善了一些系统调用，14个版本只在部分细节有变化，3个版本只是源代码表达方式有所不同，所有这些变化不影响本文测试方案．

3 验证与测试

3.1 测试环境与测试方案设计

(1)验证动态插接技术的可行性：获取典型的Linux中断、异常、系统调用的时间开销．

(2)验证本文方案测试计算机硬件变化对性能的影响:在相同操作系统内核配置、不同微架构处理器系统下,测试指定内核功能的时间开销.

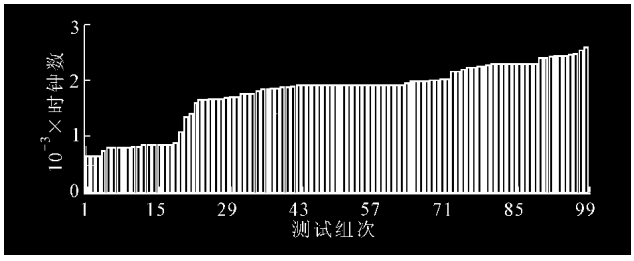
(3)验证本文方案测试操作系统内核变化对性能的影响及在多内核版本上运行的能力:测试 Linux 2.4、Linux 2.6 内核版本系统调用的时间开销及其差异,并验证了方案在不同内核版本上的通用性.

(4)测试本文技术的性能开销.

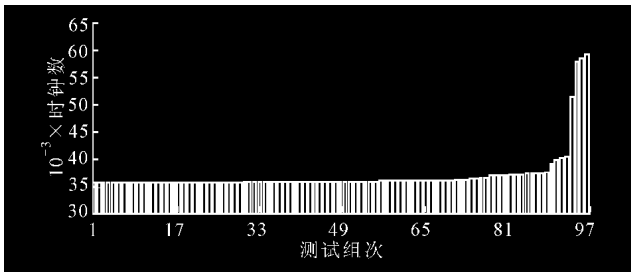
3.2　测试结果与分析

3.2.1　可行性测试　测试环境为 Intel P4,主频为 3.0 GHz,内存为 512 MB. 在 Red Hat 9.0 (Linux 2.4.20-8)上启动 X-Windows 的过程中,连续采集 100 组待测内核事件所用时钟数,去除最大值和最小值,并按时钟数排序. 图 3 为 brk 系统调用、时钟中断、缺页异常所用时钟数开销的测试结果.

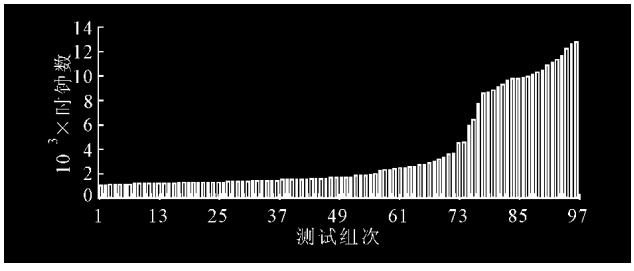
如图 3 所示,不同测试组次间内核功能时钟数开销明显划分为几个等级. 除文首所述基于统计的因素外,操作系统对相同中断和异常,其处理方式还存在复杂性,如不同时刻发生的时钟中断会响应不同的定时器要求,有些操作有时会放在后半部执行,



(a)brk 系统调用的时钟数开销



(b)时钟中断的时钟数开销



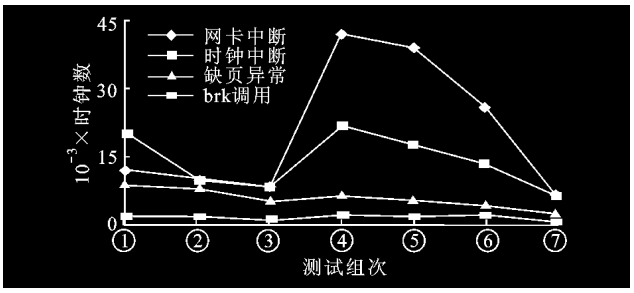
(c)缺页异常的时钟数开销

图 3　测试内核模块的时钟数开销

缺页异常也会根据不同情况采取不同的处理方式等. 对于这类差异,可以通过在测试装置中读取传入内核堆栈的参数进一步区分和细化,但限于篇幅本文在此不做讨论.

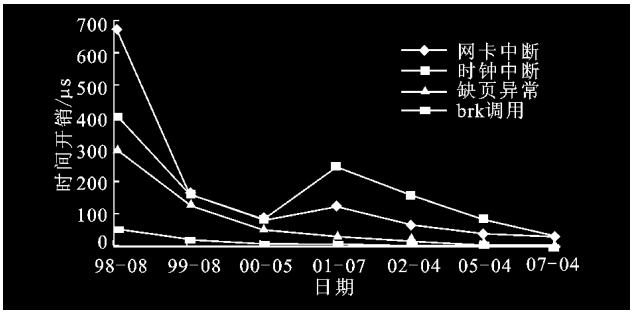
3.2.2　同操作系统版本不同硬件配置的测试　表 1 列出 7 种不同待测系统的硬件配置、微架构及处理器出厂年份. 对所列系统的网卡中断、时钟中断、缺页异常、brk 系统调用所用时钟数开销及时间开销进行测试,结果如图 4 所示.

图 4 数据显示,无论是 P6 还是 Netburst,对于同一微架构的系统,随着主频的提升,时间开销逐步减少. 当从 P6 微架构向 Netburst 微架构过渡时,同样的内核操作所用时钟数和时间开销不仅没有减少,反而大幅攀升,这时只能靠提升主频来掩盖时钟数开销的异常. 在向 Core 微架构过渡时的时钟数开销呈现回落,大约回到P6微架构水平. 表1中采用



①～⑦:对应表 1 中的 7 种型号

(a)不同内核功能时钟数开销



(b)不同内核功能时间开销

图 4　不同微架构硬件典型内核事件测试

表 1　测试使用系统的硬件配置

生产时间	微架构	型号	主频/MHz	L2 缓存/kB	主存/MB
1998-08	P6	赛扬 P2①	300	128	256
1999-08	P6	P3②	600	256	256
2000-05	P6	P3③	933	256	256
2001-07	Netburst	P4④	1 681	256	256
2002-04	Netburst	P4⑤	2 406	512	512
2005-04	Netburst	P4⑥	3 000	2 048	512
2007-04	Core	E6320⑦	1 860	4 096	2 048

Core微架构的E6320主频虽然不及Netburst微架构P4 3.0 GHz的2/3,但时间性能却超过了P4 3.0(且仅考虑单核情况).造成这种差异的原因是,Netburst微架构采用深度流水线和较小容量的level-1缓存,而P6微架构的P2和P3则采用较温和的深度超标量动态调度流水线[2].

3.2.3 同硬件不同操作系统的测试 基于主频为3.0 GHz、内存为512 MB的Intel P4服务器,在Red Hat 8、Red Hat 9及Fedora 4～Fedora 6共5种不同Linux发行版本上测试系统调用时的系统状态切换时间开销如图5所示.由于Linux 2.6内核中采用快速系统调用指令sysenter,使得在相同硬件配置下的用户进程可以从高级语言层次快速地访问内核.

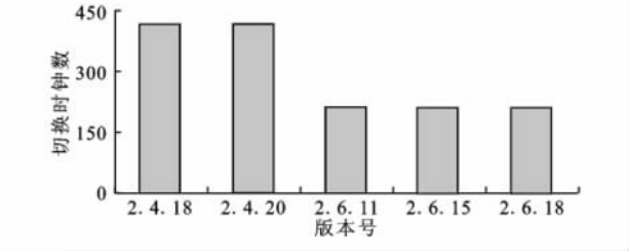
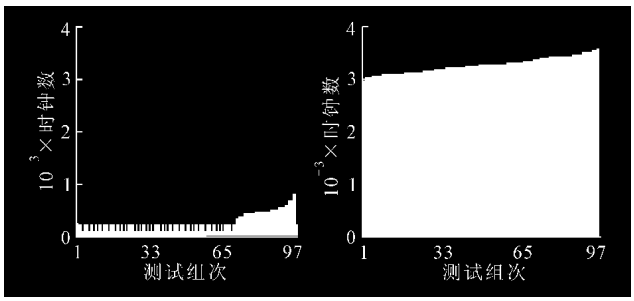


图5 不同内核系统的状态切换时钟数开销

3.2.4 性能开销分析 基于主频为1.7 GHz、内存为256 MB的Intel P4服务器和Linux 2.6.18内核版本,测得的本文方法及KProbe的时钟数开销如图6所示.二者的时钟数开销的平均值分别为197.24和3 275.77,本文方法约是后者的6%.Kprobe由于使用调试中断来插入代码,所以时间开销比较大.



(a)本文方法 (b)KProbe

图6 本文方法与KProbe的性能比较

4 结束语

本文基于IA32架构设计了一种动态Linux内核性能测试技术.实验表明该技术可以较小的系统开销对多个Linux内核版本进行性能测试,并能够确定由硬件配置或操作系统变化带来的关键性能等问题.

参考文献:

[1] ANDERSON J M, BERC L M, DEAN J, et al. Continuous profiling: where have all the cycles gone? [C]//16th ACM Symposium on Operating System Principles. New York, USA: ACM, 1997: 357-390.

[2] MOORE B, SLABACH T, SCHAELICKE L. Profiling interrupt handler performance through kernel instrumentation [C]// Proceedings of the International Conference on Computer Design. Los Alamitos, CA, USA: IEEE Computer Society, 2003:156-163.

[3] BROWN A, SELTZER M. Operating system benchmarking in the wake of lmbench: a case study of the performance of netbsd on the Intel x86 architecture [C]//ACM SIGMETRICS International Conference on the Measurement and Modeling of Computer Systems. New York, USA: ACM, 1997:214-224.

[4] PEARCE D J, KELLY P H J, FIELD T, et al. GILK: a dynamic instrumentation tool for the Linux kernel [C]// Proceedings of the 12th International Conference on Computer Performance Evaluation. Berlin, Germany: Springer-Verlag,2002:220-226.

(编辑 苗凌)