一种自动分析网络软件时延的工具*

刘海鹏, 张根度

(复旦大学 计算机科学与工程系,上海 200433)
E-mail: 990309@fudan.edu.cn
http://www.fudan.edu.cn

摘要: 网络软件实现的复杂性及运行的动态性增加了精确分析时延出现原因的难度.利用先进微处理器提供的性能监视功能和操作系统网络软件实现的特点,NetSlice 提供了一种新颖、模块化且具有良好扩展性的自动分析网络软件时延的方法.首先介绍 NetSlice 的体系结构、各组成部分的功能和关系,然后分析其性能开销和应用策略,最后给出 Linux 操作系统中 TCP 发送过程的时延分析结果.实验结果表明,NetSlice 非常有助于发现造成网络软件时延的真正原因.

关 键 词: 网络协议;时延;性能监视计数器;Linux 操作系统
中图法分类号: TP393 文献标识码: A

随着因特网的快速发展和多媒体应用的日益普及,服务质量(QoS)正在成为网络研究的热点问题.分析和研究网络系统软件的时延是保证 QoS 的重要因素之一.同时,在高性能计算环境、分布式系统以及安全网络操作系统等方面的研究中,也非常需要对网络系统软件的性能进行深入的剖析.但是,网络软件实现的复杂性和基于软件分析工具的缺乏,使得深入研究和分析网络系统软件及相应协议的通信时延成为一个颇为棘手的问题.

虽然借助于硬件逻辑分析仪[1,2]可以开展这方面的研究,但是,昂贵的硬件设备和使用的复杂性极大地限制了研究的普及.近年来,处理器的飞速发展和性能监视(performance-monitoring,简称 PM)功能的提供与完善,如 IBM 公司 Power2 的 PM[3]、Intel 公司奔腾及后续处理器的性能监视计数器[4](performance-monitoring counter, 简称 PMC)等,使得用软件手段来分析和研究网络系统软件的时延成为可能.利用 PMC 的硬件功能,Rabbit 项目[5]为 Linux 操作系统上应用程序的性能分析提供了实用函数库.Intel 公司的 VTune 性能分析软件[6]则用于 Windows 平台上应用软件的性能分析和优化.LTT[7](Linux trace toolkit)虽然提供了操作系统和网络软件性能的测量方法,但是,一方面 LTT 未利用处理器提供的性能监视功能,另一方面对网络软件性能的测量仅限于简单事件的跟踪和记录,并未对网络软件时延提供一种有效的手段.

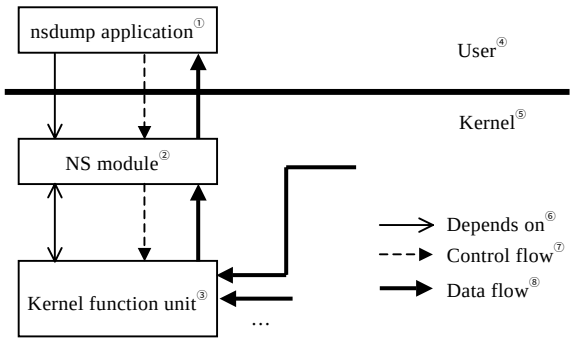
NetSlice 提供了一种测量和分析网络系统软件时延的全新方法.NetSlice 在使用 PMC 的同时,巧妙地利用了网络系统软件实现的特点,即数据包在网络层次中的垂直运行.这样,可以通过将网络软件不同模块的 PMC 测量数据存储于当前处理的数据包中,从而实现用软件的方法来测量网络软件的时延和分析时延原因.在具体实现上,NetSlice 首先选择了奔腾III处理器和 Linux 操作系统.在 Linux 中,数据包及其操作信息保存在套接字缓冲区[8](sk_buff),套接字缓冲区随数据包的产生而分配,随数据包的终止而释放.

1 NetSlice 体系结构

NetSlice 的体系结构由核心功能单元、NS 模块和 nsdump 应用程序三部分组成,如图 1 所示.核心功能单元

* 收稿日期: 2001-01-07; 修改日期: 2001-05-18
基金项目: 国家 863 高科技发展计划资助项目(863-300-02-03-99)
作者简介: 刘海鹏(1970－),男,河南焦作人,博士生,主要研究领域为计算机网络,无线移动网络,操作系统;张根度(1937－),男,江苏无锡人,教授,博士生导师,主要研究领域为计算机网络与分布式系统,宽带多媒体网络.

提供了时延和性能事件测量以及 NS 模块调用最基本的功能,需随着内核一起编译和启动运行.NS 模块一方面为应用程序所需系统调用提供丰富的核心函数支持,另一方面提供核心功能单元所需的扩展功能.NS 模块既可以独立编译、动态加载,也可以直接编译到内核中去.Nsdump 应用程序直接供终端用户使用.



①nsdump 应用程序,②NS 模块,③核心功能单元,
④用户态,⑤核心态,⑥依赖,⑦控制流,⑧数据流.

Fig.1 The architecture of NetSlice
图 1 NetSlice 的体系结构

核心功能单元和 nsdump 应用程序均与 NS 模块相互依赖,但是二者之间没有依赖关系,可以根据用户或内核的变化而独立修改、编译.Nsdump 应用程序发送控制信息至 NS 模块,设置 NS 模块的属性,或由 NS 模块根据请求生成新的信息来修改核心功能单元的默认配置.套接字缓冲区释放时调用核心功能单元提供的宏指令,将测量数据发送到 NS 模块,NS 模块经过处理,根据情况实时或延时将数据转发至 nsdump 应用程序.NetSlice 体系结构的设计一方面使其具备足够的灵活性和可扩展性,另一方面使系统必须的额外开销最小.

1.1 核心功能单元

核心功能单元提供了 5 个宏指令,用于测量网络软件时延或 PMC.宏指令的描述如下:

(1) NET_SLICE_CONF(size)(简称 NSC).配置测量数据块缓冲区,size 是缓冲区大小.必须在 Linux 系统分配套接字缓冲区前调用.

(2) NET_SLICE_INIT(skb,dir)(简称 NSI).初始化数据块缓冲区,包括数据块标志、测量开始时间等.skb 是结构 sk_buff 型指针,dir 是数据包流动方向.在整个测量开始时调用.

(3) NET_SLICE_START(id,skb)(简称 NSS).测量被测代码时延或 PMC 之前调用,id 是被测代码的标识号.

(4) NET_SLICE_STOP(id,skb)(简称 NST).测量被测代码时延或 PMC 之后调用.

(5) NET_SLICE_EXIT(skb)(简称 NSE).处理测量的数据,Linux 系统释放套接字缓冲区前调用.

核心功能单元为 NS 模块提供了注册、注销和必须的控制函数.如果 NS 模块被编译为内核的一部分,NS 模块则在系统启动时向核心功能单元注册;反之,则在模块加载时注册.NS 模块注册时,向核心功能单元提供数据接收回调函数.修改后的内核在套接字缓冲区终止时不再调用系统原有的释放函数,而是直接调用宏指令 NSE.这样,数据接收函数在被间接调用时,将包含测量数据块的套接字缓冲区送至 NS 模块进行处理;如果 NS 模块未注册,则宏指令 NSE 只是简单地调用系统原有的释放函数,不对测量数据进行任何处理.总之,核心功能单元在网络系统软件和 NS 模块间起着桥梁的作用.

1.2 NS模块

NS 模块在 NetSlice 体系结构中起着关键作用,NS 模块的设计和实现直接影响 NetSlice 的性能.从理论上讲,NS 模块的功能是简单的,即接收和缓存核心功能单元发送的套接字缓冲区,然后有效地将其中的测量数据块传送到 nsdump 应用程序.但是在具体实现上,还有许多因素需要考虑.

1.2.1 测量数据格式和数据传送方式

由于网络系统软件的执行时延往往在微秒甚至纳秒级,因此,设计高效并且能提供足够信息的测试数据格式是非常重要的.在 NS 模块实现中,一次测量数据称为一个数据块,数据块由块头和零个或多个数据记录组成.块头和数据记录的格式如图 2(其中①~⑧分别为传输层协议、数据传输方向和记录类别、性能事件号、传输层头长度、网络层头长度、网络层数据总长度、测试开始时 TSC[4]的高 32 位和测试开始时 TSC 的低 32 位)和图 3(其中①~③分别为测试区域代号、记录魔幻数和时延或性能事件测试值)所示.数据块的存储空间随着套接字缓冲区一起分配和释放,且位于数据包尾部.

在设计 NS 模块和 nsdump 应用程序的数据传送方式时,有中断和查询两种方式可以选择.Tcpdump[9]采用查

询方式,LTT 则采用中断方式.为了支持实时监测和显示,NS 模块选择了查询方式.同时,在具体实验时,NS 模块接收数据块缓冲区的大小可以设置为容纳一次测试的所有数据块.这样,NS 模块便可以支持 nsdump 应用程序一次接收全部数据,从而使 nsdump 应用程序的用户/核心切换开销对测试系统的额外影响为零.

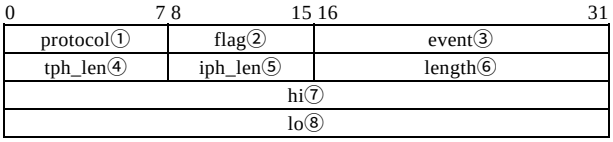


Fig.2 Header format of the data block
图 2 数据块头格式

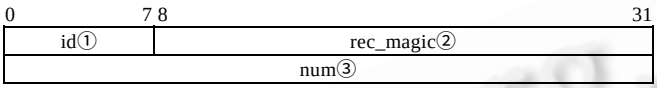


Fig.3 Record format of the data block
图 3 数据块记录格式

1.2.2 NS 模块的配置

NS 模块提供丰富的配置,以支持 nsdump 应用程序的各种测试要求.重要配置参数有:

- NSIOCSRCVBUF 设置接收数据缓冲区大小
- NSIOCSEVENT 设置监视的性能事件
- NSIOCSEMODE 设置性能事件的监视类别,计数或计持续时间
- NSIOCSSTATE 使能或禁止 NS 模块工作
- NSIOCGPACKETS 得到 NS 模块正确接收的数据块数
- NSIOCGDROPS 得到 NS 模块丢弃的数据块数

1.2.3 基本文件操作的核心函数

Nsdump 应用程序通过/dev/ns 虚拟字符设备访问 NS 模块.为此,NS 模块提供了 4 个基本文件操作函数,即 open,ioctl,read 和 close.

为了尽可能减少对被测网络系统的影响,NS 模块采用了将收到的套接字缓冲区进行简单处理后直接缓存的方式.同时,为了防止 NS 模块消耗过多的内存空间,NS 模块根据系统启动时所配内存大小,自动调整使用内存的上下限.

1.3 Nsdump应用程序

Nsdump 应用程序在编制时借鉴了 tcpdump 的命令参数格式和约定,因此使用时在许多方面和 tcpdump 类似.用户既可以实时在终端显示测试数据,也可以指定在收到若干数据之后一起显示.如果不希望立即看到数据,也可以设置将收到的数据直接保存于某一文件,供日后分析使用.另外,为了方便使用,Nsdump 应用程序具有对测量数据进行自动分析和统计的功能.

2 NetSlice 性能分析

NetSlice 性能测试环境为两台微机通过双绞线直接连接而构成的局域网.用于测试的微机型号一台是奔腾Ⅲ450MHZ,另一台是赛扬 366MHZ.两台机器网卡均是 D-Link DFE-530TX.实验采用 HBench-OS[10]的 Lat_tcp 和 Netperf[11]作为系统开销的测试程序,Lat_tcp 和 Netperf 的测试结果为用户层的环路时延*.

2.1 指令和宏指令开销

奔腾Ⅲ处理器提供的汇编指令 rdpmc 和 rdmsr[12]都可以用来读 PMC 的值.但是在同一处理器上,指令 rdpmc 的执行时延远远小于 rdmsr 的开销,因此,NetSlice 采用了 rdpmc 指令.表 1 给出了在奔腾Ⅲ上进行实验测得的相

* 如果不特别指明,本文时延测量时用户层数据均为 4 字节.

关机器指令开销.

Table 1 Latency of machine instructions
表 1 机器指令的执行时延

Machine instructions①	cpuid	rdmsr	rdpmc	rdtsc	wrmsr
Latency② (ns)	173	180	82	69	220

①机器指令,②时延.

宏指令 NSC 实际上是一个加法运算,其额外开销可以忽略不计,其他宏指令的执行时延见表 2.表中时延的变差系数(coefficient of variation)除了标注以外,其余均小于 0.5%.图 4 给出了宏指令部分性能事件的开销次数.图中宏指令 NSE 各事件测量值的变差系数在 0.6%~2.1%之间波动,其余变差系数不超过 0.5%.

Table 2 Latency of macro-instructions
表 2 宏指令的时延

Acronym①	Macro-Instructions②	Latency③ (ns)		Event symbol⑥
		Value④	Symbol⑤	
NSI	NET_SLICE_INIT()	87	τ_α	$n(\alpha,e)$
NSS	NET_SLICE_START()	132	–	–
NST	NET_SLICE_STOP()	154	–	–
NSE	NET_SLICE_EXIT()	904 (1.4%)	τ_γ	$n(\gamma,e)$
NSO	Inter-Overhead between NET_SLICE_START() and NET_SLICE_STOP()⑦	150	τ_o	$n(o,e)$
NSA	Overhead of NET_SLICE_START() and NET_SLICE_STOP()⑧	303	τ_β	$n(\beta,e)$

①简称,②宏指令,③时延,④值,⑤符号,⑥事件符号,⑦NET_SLICE_START 和 NET_SLICE_STOP 之间的内开销,⑧NET_SLICE_START 和 NET_SLICE_STOP 的总开销.

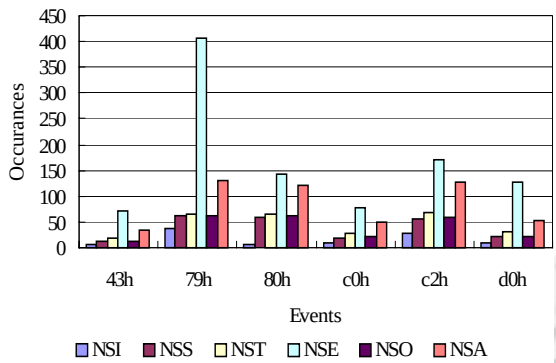


Fig.4 Event occurrences of macro-instructions
图 4 宏指令事件数

在图 4 中,执行宏指令 NSE 的机器周期数(事件 79H)为 408,即 907 纳秒,这与表 2 中的相应时延值非常吻合.在宏指令 NSE 的执行中,取机器指令数为 144(事件 80H),处理器解码数是 126(事件 D0H),指令撤消数为 77(事件 C0H).这一方面反映了处理器不同单元的处理差异,另一方面表明该宏指令的实现代码存在进一步优化的余地.

表 2 和图 4 的测试结果表明,宏指令的额外开销不但小,而且非常稳定.

2.2 系统的额外开销

为了测试 NetSlice 对网络系统软件的整体额外开销,本节对下述配置进行了测试.

Ⅰ 除了将网卡驱动程序由 CONFIG_VIA_RHINE 取代 CONFIG_EEPRO100 以外,其余内核配置参数采用 Linux 2.4.0[13]的默认配置.

Ⅱ 配置Ⅰ基础上增加核心功能单元,NS 模块为动态加载.

Ⅲ 配置Ⅱ基础上增加对宏指令开销进行测量的若干宏指令(表 2 和图 4 测试结果的配置).具体增加的宏指令为:1 个 NSI,7 对 NSA(包括对自身开销的测量),1 个 NSE.宏指令的位置与第 4 节中测试 TCP 各模

块时延的位置相同,以保证宏指令的开销尽可能地反映真实情况.

3 种配置的测试结果见表 3,表中括号内的数值是变差系数.表 3 表明,核心功能单元的引入和宏指令的增加不但使系统的额外存储开销很小,而且对系统性能的影响也小.

Table 3 The overhead of NetSlice on systems

表 **3** NetSlice 对系统的额外开销

Configuration①	system.map (Bytes)	vmlinux (Bytes)	bzImage (Bytes)	lat_tcp (μs)	netperf (μs)
Ⅰ	442 947	2 612 768	920 909	103.7 (0.1%)	104.92
Ⅱ	443 497	2 613 537	921 299	104.11 (0.2%)	105.78
Ⅲ	443 497	2 613 537	922 471	105.11 (0.4%)	105.89

①配置.

2.3 用户层环路时延

在实际使用时,网络时延测试程序(例如 lat_tcp)的运行次数远远大于 1,因此,NetSlice 可以用来测量相临两个数据块的时延.此时延的实际值便是用户层环路时延.图 5 中 lat_tcp 是在配置Ⅰ时使用测试程序 lat_tcp 测得的数值;lat_tcp_ns0 是在配置Ⅲ时 NetSlice 的测量值,lat_tcp_ns 是使用第 3 节公式(5)计算后的实际值.在帧长小于 512 字节时,lat_tcp_ns 和 lat_tcp 测值相对误差小于 0.5%.在帧长为 1 518 字节时绝对误差最大,为 2.86 微秒,但是相对误差仍小于 1%.这说明,NetSlice 宏指令的引入对系统网络性能的负面影响不大.进一步的理论分析指出,由于宏指令未使用奔腾Ⅲ的序列化指令[14](serializing instructions),因而宏指令的加入不可能显著影响原有代码的并行执行.

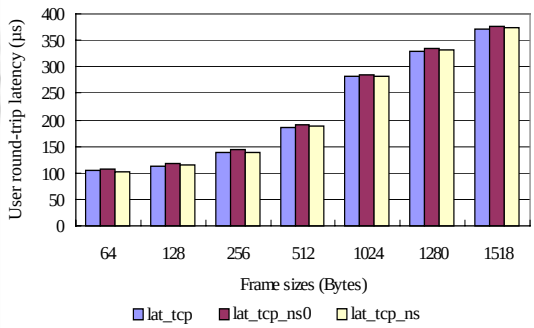


Fig.5 Event occurrences of macro-instructions

图 5 宏指令事件数

2.4 丢包问题

由于在设计 NS 模块时保证其有足够的数据块缓冲区,同时对每次运行网络时延测试程序的次数予以限制,因此不论 nsdump 应用程序实时接收数据,还是延时接收数据,都不存在丢包问题.当然,如果失去这两个假设前提条件,nsdump 应用程序在运行时会出现丢包现象.然而分析 NetSlice 和网络系统软件的运行机制,即便存在丢包现象,测试结果除了环路时延以外均是正确的.实际测试结果也证明了这一点.

3 应用策略

NetSlice 宏指令的典型应用如图 6 所示.图中忽略了宏指令 NET_SLICE_CONF(),因为它总位于 Linux 系统函数 *alloc_skb*()中.在本节,使用符号 T'和 N'代表测量的时延和性能事件值;使用符号 T 和 N 代表实际的时延和性能事件值.图 6 中计算代码 A 或 B 的 T,Ne 值使用公式(1)和(2);代码 X 的 T 值和 Ne 值使用公式(3)、(4)来计算.公式(3)和(4)中 K 为被测量代码 X 间的宏指令对 NSA 的个数.测量用户层环路时延的实际值使用公式(5)计算,式中 K 为环路代码中宏指令对 NSA 的个数.

$$T_i = T_i' - \tau_o , \tag{1}$$

$$N_{(i,e)} = N'_{(i,e)} - n_{(o,e)} , \tag{2}$$

$$T_i = T_i' - \sum_{j=1}^{K} \tau_\beta - \tau_o , \tag{3}$$

$$N_{(i,e)} = N'_{(i,e)} - \sum_{j=1}^{K} n_{(\beta,e)} - n_{(o,e)} , \tag{4}$$

$$T = T' - \tau_\alpha - \sum_{j=1}^{K} \tau_\beta - \tau_\gamma - \tau_o . \tag{5}$$

```
NET_SLICE_INIT(skb)

NET_SLICE_START(X,skb)
NET_SLICE_START(A,skb)
    Measured code A
NET_SLICE_STOP(A,skb)

    …

NET_SLICE_START(B,skb)
    Measured code B
NET_SLICE_STOP(B,skb)

    Measured code C, D etc.
NET_SLICE_STOP(X,skb)

NET_SLICE_EXIT(skb)
```

Fig.6 An example of using macro-instructions
图 6 宏指令使用例子

4 应用实例

在 Linux 操作系统中,TCP 数据的发送要经过 tcp,ip,link,nic 等模块的处理[8].涉及发送的上下文环境包括:应用进程发送数据、网卡发送完数据帧后的硬中断处理和随后的软中断处理.图 7 显示了 TCP 发送数据包时不同模块的实际时延和性能事件值.由于用户到核心的数据拷贝和校验(图 7 中的 csum)发生在 tcp 处理模块,因此,图 7 中 tcp 的时延和性能事件值包含 csum.图 7 中的 write 表示应用进程自套接字缓冲区分配后到 link 模块完成发送之间的代码.

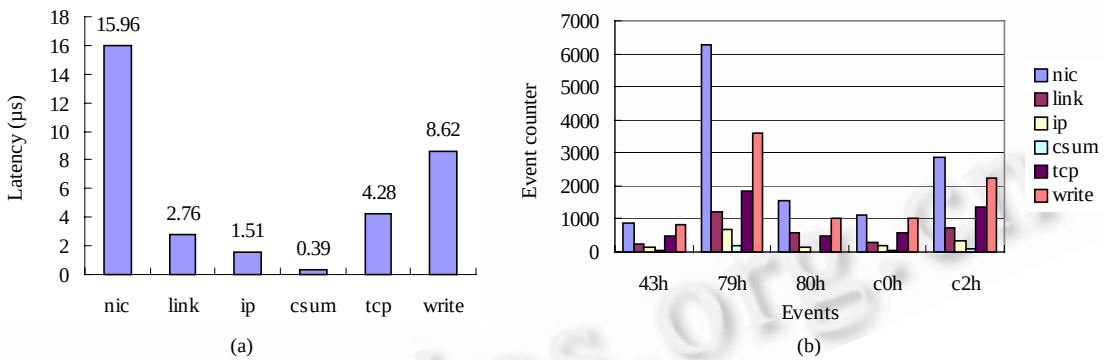


Fig.7 Latency and performance events for sending TCP data
图 7 TCP 发送数据时的时延和性能事件值

分析图 7(a)得知,网卡从接收数据到完成数据的发送需要 15.96 微秒时间.然而在这段时间内,处理器完全可以并行执行完应用测试进程的接收数据代码,并使进程进入睡眠.图 7(b)中性能事件数 79H 清楚地显示出这一期间处理器工作的时钟数为 6268,即 13.93 微秒.换句话讲,在这段时隙,处理器有 2.03 微秒的空闲时间.图 7(a)显示,Link 模块的处理时间高达 2.76 微秒,远大于 ip 模块的处理时间;同时,图 7(b)中 Link 模块的事件数 79H 为 1217,而 ip 模块只是 693.由于 Linux 实现 Link 模块的代码相对比 ip 模块简单,因此这一现象值得进一步研究.

5 结 论

利用当前先进处理器的性能监视功能和网络系统软件实现的特点,本文提出了一种新颖的用于自动分析网络软件时延的方法.实验显示,NetSlice 不仅使系统的额外开销很小,而且宏指令的执行时延非常稳定,完全可以用来分解网络软件的时延和分析时延原因.对 Linux 操作系统上发送 TCP 数据的通信时延分析,不仅揭示了网络系统软件存在的问题,而且很好地证明了 NetSlice 的实用性和灵活性.由于 NetSlice 基于 Linux 操作系统和模块化设计,因此使其便于扩展和用户化.当前,我们正在利用 NetSlice 对在 Linux 上实现的多协议标记交换

MPLS 性能进行分析和研究.

致谢 复旦大学信息科学与工程学院的凌力老师对本文的完成提出了很多有益的建议,在此表示感谢.

References:

[1] Clark, D.D., Jacobson, V., Romkey, J., *et al*. An analysis of TCP processing overhead. IEEE Communications Magazine, 1989, 27(6):23~29.

[2] Kay, J., Pasquale, J. The importance of non-data touching overhead in TCP/IP. In: Chlamtac, I., Sidhu, D., eds. Proceedings of the ACM SIGCOMM'93. San Francisco, CA: ACM Press, 1993. 259~268.

[3] Welbon, E.H., Chan-Nui, C.C., Shippy, D., *et al*. POWER2 performance monitor. IBM Journal of Research and Development, 1994,38(5):545~554.

[4] Intel Corporation. Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide. 1999.

[5] Rabbit : a performance counters library for Intel/AMD processors and Linux. 2000. Http://www.scl.ameslab.gov/Projects/Rabbit.

[6] Intel VTune performance analyzer. 2001. http://developer.intel.com/software/products/vtune/index.htm.

[7] Yaghmour, K., Dagenais, M. Measuring and characterizing system behavior using kernel-level event logging. In: Small, C., Arnold, K., eds. Proceeding of the USENIX 2000 Annual Technical Conference. San Diego, CA: Usenix Press, 2000.

[8] Rusling, D.A. The linux kernel. 1996. http://www.linuxhq.com/guides/TLK/tlk.html.

[9] Jacobson V., Leres, C. Tcpdump——dump traffic on network. UNIX man pages, 1998.

[10] 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. In: Vernon, M., Gibson, G., eds. Proceedings of the ACM SIGMETRICS'97. Seattle, WA: ACM Press, 1997. 214~224.

[11] Netperf. 1999. http://www.netperf.org.

[12] Intel Corporation. Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference. 1999.

[13] Linux kernel. 2001. http://www.kernel.org/pub/linux/kernel/v2.4.

[14] Intel Corporation. Intel Architecture Optimization Reference Manual. 1999.

A Tool for Automated Analysis of Network Software Latency*

LIU Hai-peng, ZHANG Gen-du

(*Department of Computer Science and Engineering, Fudan University, Shanghai* 200433, *China*)

E-mail: 990309@fudan.edu.cn

http://www.fudan.edu.cn

Abstract: The complexity and the dynamic executing behavior of network software add to the difficulties for identifying the precise causes of latency. Using performance monitoring functions provided by the advanced microprocessor and implementation characteristics of network software, NetSlice provides a novel, modular, and extensible way for automated analysis of network software latency. The architecture of NetSlice and its components are introduced at first. Then the performance of NetSlice is studied in detail and the typical application strategy is also given. To demonstrate the utility of NetSlice, the results of analyzing the latency of TCP sending process on Linux operating system are presented. The experimental results show that NetSlice can shed considerable light on the causes of latency in network software.

Key words: network protocol; latency; performance-monitoring counter; Linux operating system

* Received January 7, 2001; accepted May 18, 2001
Supported by the National High Technology Development 863 Program of China under Grant No.863-300-02-03-99