FUPS-DV:用于桌面虚拟化的全时抢占 CPU 调度算法

夏虞斌[1,2],杨 春[2],程 旭[2]
(1.复旦大学并行处理研究所,上海 200433; 2.北京大学微处理器研究开发中心,北京 100871)

摘 要: 桌面虚拟化通常运行混合负载,且更注重交互式性能,现有的虚拟机调度算法无法很好适应这两个特点.本文提出了一种全时抢占 CPU 调度算法,通过灰盒技术探测虚拟机内部信息用于辅助虚拟机调度,并结合远程桌面的负载特性进行优化.评测表明,5 台 Windows XP 虚拟机同时运行混合负载,优化后播放幻灯片的显示延迟降低了至少 60%.

关键词: 虚拟机; 桌面虚拟化; 瘦客户计算; 调度算法; Xen
中图分类号: TP368 **文献标识码:** A **文章编号:** 0372-2112 (2011) 08-1721-06

FUPS-DV: Full-Time Preemption CPU Scheduling for Desktop Virtualization

XIA Yu-bin[1,2], YANG Chun[2], CHENG Xu[2]
(1. *Parallel Processing Institution of Fudan University*, *Shanghai* 200433, *China*;
2. *Microprocessor Research and Develop Center of Peking University*, *Beijing* 100871, *China*)

Abstract: Desktop virtualization usually runs mixed workload, and is more sensitive to the interactive performance. Current virtual machine scheduler cannot meet the two demands. This paper presents a full-time preemption CPU scheduler. It uses grey-box technology to inspect information inside virtual machine to support virtual machine scheduling, and considers the characteristics of remote desktop workload for optimization. The evaluation results show that when 5 Windows XP virtual machines running mixed workload concurrently, the display latency of slides presentation is reduced by at least 60% with our optimization.

Key words: virtual machine; desktop virtualization; thin client computing; scheduling; Xen

1 引言

桌面虚拟化(Desktop Virtualization)是一种将虚拟化技术与瘦客户系统相结合的技术.桌面系统以 VM(Virtual Machine)的形式在服务器上运行,使用远程显示协议(如 RDP、VNC 等)对外提供远程显示服务,用户通过网络访问其专用的桌面计算环境.

与传统的用于网络服务器整合(Network Server Consolidation)的虚拟化系统相比,桌面虚拟化系统具有以下两个特点:首先,用户对于操作延迟的要求更高,其次,VM 通常运行混合性负载,其中有些偏重交互性能,如文档编辑、网页浏览等;有些则偏重计算性能,如编译、压缩解压缩等.在实际应用中发现,以 Xen[1] 为虚拟化平台,当多个运行混合负载的 VM 同时运行在一台物理主机上时,远程桌面的操作延迟大大增加,远高于同等负载下的非虚拟化方案[2].

上述问题的原因在于:现有 VM 调度算法中的抢占机制对于延迟的优化仅适用于非混合负载的情况;而对于混合负载,由于 VMM(Virtual Machine Monitor)无法获知 VM 内部的任务信息,导致 VM 调度算法很难进行有效的资源分配.本文设计并实现了用于桌面虚拟化的全时抢占 CPU 调度算法 FUPS-DV(FUll-time Preemption Scheduling for Desktop Virtualization).

FUPS-DV 充分利用了桌面系统的特性,扩充了原有调度算法的抢占机制,对远程显示相关事件进行优化,提高响应速度;使用灰盒技术探测客户 VM 内部的任务信息对交互式任务进行识别和优化,细化抢占粒度,提高抢占精度;通过对抢占率参数的调节来控制抢占时间,保证不同 VM 间的调度公平性;不需要对客户操作系统进行任何修改,且不依赖于特定的远程显示服务.

收稿日期:2010-01-30;修回日期:2011-02-18
基金项目:国家 863 高技术研究发展计划(No.2006AA010202)

实验使用 Windows XP 作为客户 OS,使用真实应用程序进行了评测.评测数据表明,在 5 台 VM 同时运行混合负载的情况下,优化后 RDP 和 VNC 的显示延迟相比优化前分别降低了 77.3% 和 60%.同时,该优化也能够保证客户操作系统对于资源占用的公平性.该优化并不依赖于特定的虚拟化平台,因此除 Xen 以外的其他虚拟机平台同样适用.

2 相关工作

在虚拟机调度方面,Cherkasova 等[3]在先前的研究中分析了 Xen 三个调度器各自的特点.Govindan 等[4]在研究中发现了与本文类似的问题,他们通过引入通信友好的 CPU 调度方法,改进了 Xen 环境中设备驱动虚拟机的响应性.但是其改进的调度方法仅在保证各个虚拟机 CPU 份额的基础上根据 I/O 请求情况优先调度驱动虚拟机,当驱动虚拟机 CPU 份额不足时,仍然会导致客户虚拟机长时间等待 I/O 请求的问题.Ongaro 等[5]研究了调度器的不同参数配置对 I/O 性能的影响,他们在实验评测中只使用了简单的 ping 来代表延迟敏感应用程序,而本文则使用真实的应用程序.

Kim 等[6]提出了在 VMM 层通过灰盒技术获取 VM 内部任务信息的方法,设计了任务可知的调度器 TAVS (Task-Aware Virtual-machine Scheduler),改进了 VM 在高整合程度下的 I/O 性能.本文在实现上借鉴了其探测任务信息的方法.不同的是,Kim 的工作集中分析了 I/O 任务的行为,其主要目的为提高 I/O 性能;而本文则针对更高层次的桌面应用的特点,其主要目的是为提高交互式应用的性能.本文在实验评测中对 TAVS 和 FUPS-DV 进行了对比评测.结果显示,对于运行混合负载的 VM,FUPS-DV 的交互性能高于 TAVS.

3 全时抢占机制

本节分析了在高负载环境下,桌面虚拟化的延迟是由于 Xen 现有的抢占机制并不能很好的适应其异步事件机制而导致;进而提出了全时抢占机制,通过与异步事件机制更好的配合来降低操作延迟.

3.1 原有抢占机制缺陷

为了提供对已有驱动程序的二进制兼容,同时也为了保证驱动程序的错误隔离,Xen 采用了分离式 I/O 模型.在此模型中,设备驱动分为前端和后端两部分,前端驱动运行在客户虚拟机(guest domain)中,后端驱动运行在驱动虚拟机(driver domain,通常为 domain-0)中,驱动虚拟机有权限通过本地驱动程序对硬件直接进行操作.前后端驱动程序通过事件通道(event channel)进行控制通信,并利用共享内存进行数据传输.因此,每次 I/O 操作都需要客户虚拟机和驱动虚拟机协同完成.

事件通道是一种异步的 I/O 机制,客户虚拟机只有当被调度运行时,才有机会对事件进行处理;在此之间,事件处于未决(pending)状态.因此,事件的发生与实际发送(delivering)之间存在一定的延迟.

为缩短异步事件机制所导致的处理延迟,Xen 的默认调度算法(Credit 算法)加入了 Boosting 优化:当某个 VCPU(Virtual CPU)由于事件发生而被唤醒时,有机会抢占当前运行的 VCPU.Boosting 优化的抢占只发生在 VCPU 被唤醒的时刻,只能用于仅运行 I/O 负载的 VM;对于运行混合负载的 VCPU 来说,事件发生时很可能在运行队列中排队,不满足被唤醒的条件,因此上述抢占优化失效,如图 1(a)所示.

3.2 全时抢占

为减少事件处理的延迟,本文提出了全时抢占的机制:即当有事件发生时,即使 VCPU 正在运行队列中等待,也有机会进行抢占.实际应用中,事件处理只需要花费很少的时间(如图 1 中阴影部分所示),因此理想的调度行为应当是让 $d4$ 进行抢占,处理完事件后,VMM 立即回收 CPU,如图 1(b)所示.这样一方面能够使得 VM 尽快完成对事件的处理,另一方面也能减少抢占对于其他 VM 的影响,不破坏调度的公平性.

全时抢占机制有两个核心要素:即抢占的开始点(何时触发抢占)、以及抢占的结束点(何时回收 CPU).可以通过对这两个核心要素设定不同的策略,来满足不同应用的需求.在桌面虚拟化应用中,有两个需求:第一,要尽可能降低用户的操作延迟,提高交互式性能;第二,必须保证 VM 调度的公平性.针对这两点需求,全时抢占的触发条件必须与用户的操作以及屏幕显示相关,尽可能降低用户可感知的延迟;同样,全时抢占的持续时间应当尽可能短,即在处理完交互式任务后就应当立即结束抢占,防止抢占时间过长导致其他 VM 的延迟增加.

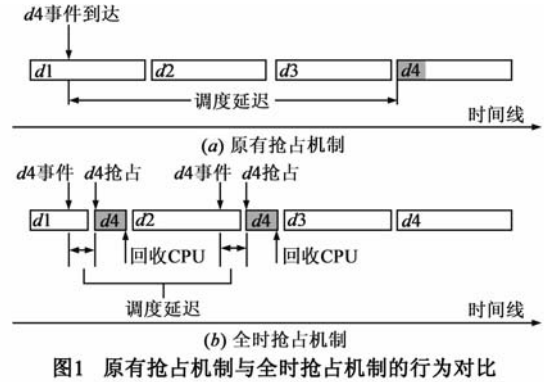


图1 原有抢占机制与全时抢占机制的行为对比

4 FUPS-DV 的设计与实现

FUPS-DV 利用桌面虚拟化的特点实现了全时抢占,有以下三个特点:(1)将全时抢占应用于特定端口(即

远程显示网络端口)的数据包接收,通过降低关键事件的处理延迟来提高响应速度;(2)对 VM 的任务信息进行跟踪,利用应用程序的运行特点判断其交互置信度;在抢占状态下只允许运行交互性高的应用程序,从而减少抢占对系统的负面影响;(3)引入 VM 抢占时间控制参数"抢占率"(preemption ratio),管理员可通过设置该参数防止 VM 过度抢占,从而进一步保证调度公平性.在具体实现中,主要修改了 Xen 的事件发送模块、内存管理模块和调度模块,总代码行数小于 800 行.下面将分别对这三点进行进一步的阐述.

4.1　特定端口事件触发抢占

FUPS-DV 通过网络端口来识别出特定的网络事件,并通过触发抢占来进行优化;具体的端口号由管理员进行设置,以适应不同的远程显示服务.具体实现中,通过修改 Domain-0 的后端驱动,使其在接收到网络包之后,记录网络包的目标 VM 与网络端口,并通过共享内存传递给 VMM,如图 2 所示.在 Domain-0 切换出去的时刻,VM 调度器根据上述网络包信息进行判断,若目标 VM 接收到了之前注册为远程显示服务端口的网络包,则在其 VCPU 抢占率低于预设值的条件下,将该 VCPU 提前到运行队列的头部,使其能尽快获得运行机会.VCPU 抢占率将在 4.3 节中进一步阐述.

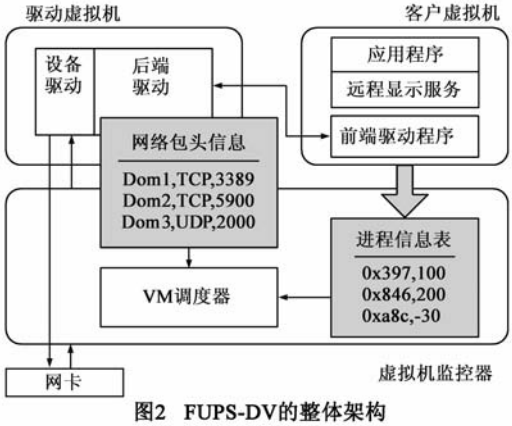


图2　FUPS-DV的整体架构

4.2　任务信息的探测与识别

为了保证 CPU 的公平性,FUPS-DV 仅允许交互性高的任务进行抢占,这需要 VMM 有能力区分 VCPU 上运行的应用程序,并根据其交互性区分对待.目前常见的桌面操作系统,为了提高交互式性能,一般都会对交互式任务进行识别和优化.识别方法一般基于以下观察:交互式应用通常会将大部分时间用于等待用户输入,因此一般消耗较少的 CPU.优化方法则是通过抢占,在交互式任务需要运行的时刻予以优先执行.为了与桌面操作系统中常见的调度器进行配合,FUPS-DV 在 VMM 层通过使用灰盒技术来探测客户操作系统内部的任务信息用于辅助虚拟机调度,同时结合远程桌面的负载特性进行了优化.

4.2.1　VM 内部的任务识别

在 x86 体系结构中,处理器使用 CR3 寄存器存储进程虚拟地址的页目录.由于每次进行任务切换时都会陷入 VMM,因此 VMM 能够获取任务每一次运行的开始时刻和结束时刻,从而识别出任务信息.对于支持 Nest Page Table 或 Extended Page Table 的平台,可通过设置 VMexit 的配置寄存器,使客户虚拟机在切换 CR3 寄存器时发生 VMexit 陷入 VMM.

4.2.2　任务的交互置信度

交互置信度是 FUPS-DV 对任务交互程度的量化值.通过对 CR3 寄存器的监视,VMM 能够得知每个任务每次消耗 CPU 的时间.若该时间小于阈值,则增加其交互置信度;反之则减少.在具体实现中,对于 VCPU 的第一个任务和最后一个任务需要进行特殊处理:首先,当 VCPU 运行后,可能由于事件立即导致抢占,因此第一个任务可能是 CPU 密集型应用,其运行时间却很短;其次,若某个 VCPU 被其他 VCPU(如 domain-0)抢占,其任务的运行时间也可能很短,但却不一定是交互式任务.

4.2.3　抢占回收

FUPS-DV 根据任务的交互置信度,判断其是否能在抢占状态下运行,从而决定抢占的结束时刻.如果 VCPU 在抢占状态下运行了非交互式任务,则回收其 CPU,结束抢占状态.在具体实现中,使用以下两种方法来检测 VCPU 当前是否运行非交互式应用:

方法一:在 VCPU 切换任务时,检测下一个是否为交互式任务;如果为非交互式任务,则在其被调度前一刻回收 CPU.方法二:若 VCPU 进入抢占状态时正运行非交互式任务,则进行定时检查.通过设置定时器,每个 0.1 毫秒判断当前 VCPU 处于用户态还是内核态,若处于用户态且运行的是非交互式任务,则回收 CPU 并结束其抢占状态.方法二是对方法一的补充,两者结合能有效防止非交互式任务在抢占状态下的运行.

4.3　调度公平性

通过对 VM 内部任务的检测,可以有效的控制每次抢占的时间,但并不能完全保证 CPU 分配的公平性.首先,运行混合负载的 CPU 会频繁的进行抢占,由于抢占并没有考虑 VCPU 之前的优先级和状态,因此无节制的抢占会导致连续中断其他 VCPU 的运行;其次,若某个任务积累了较高的交互置信度后开始大量占用 CPU,则可能导致该任务长时间的抢占 CPU.虽然该问题发生的次数并不多,但通过实验发现,其负面效果并不可忽略.再次,恶意的应用程序也可能通过频繁抢占来获得更多的 CPU 和 I/O 资源.

为了保证调度的公平性,本文提出了两种限制方法.第一,限制单次抢占的时间,即 VCPU 从进入抢占状

态到退出抢占状态的运行时间不得超过某个上限,一旦超过则回收 CPU.第二,引入了抢占率来约束 VCPU 的抢占行为.抢占率用来表示允许抢占的频率,即抢占的 CPU 时间占运行 CPU 的总时间的比例.这两个时间都会阶段性的清零,从而仅仅统计近期的抢占行为.如果抢占率较低,那么只有交互式应用程序,而不是 I/O 密集型应用,能够通过获得较好的响应速度.相反,如果抢占率较高,则表示该 VM 能够使用更多的 I/O 资源.如果抢占率设置为 0,则与原来默认的调度算法行为完全一致.

4.4 对 Credit 算法的其他修正

FUPS-DV 也考虑并解决了 Credit 调度算法的已知问题[6].Credit 调度算法会将切换出的 VCPU 排在同优先级队列的末尾.因此当某个 VM 的 I/O 较为密集时,其 VCPU 会频繁被驱动 VM 抢占,并排在同优先级队列的末尾;这导致该 VM 的 CPU 占用量急剧下降,破坏了 CPU 分配的公平性,同时也导致 VM 内部的应用程序性能大大下降.为了提高公平性,FUPS-DV 采用了如下优化:若某个 VCPU 是被驱动 VM 抢占而切换出,则该 VCPU 排在同优先级队列的头部,从而保证当驱动 VM 运行完后,该 VCPU 能够立刻继续运行.

5 实验评测

实验使用了 VNC-IPA[2]来进行交互式会话的录制回放,以多用户并发、同步回放相同会话作为测试负载,记录所有操作的延迟作为评测标准.使用实际应用测试来评测 FUPS-DV 的优化效果,并且与 TAVS 进行了对比.其中,TAVS 的实现来自于 Kim[6]提供的源码.实验的服务器和客户终端的配置如表 1 所示.

表 1 软件/硬件配置列表

	硬 件	软 件
物理服务器	AMD Opteron 246,4GB 内存,100Mbps 网卡,146GB 硬盘	Xen-3.2.0-stable
驱动虚拟机	VCPU×1,416MB 内存	Linux-2.6.18/RedHat EL-5.3
桌面系统虚拟机	VCPU×1,256MB 内存	WindowsXP/GPLPV 驱动
客户终端	北大众志网络计算机	Linux 2.4.17/BusyBox-1.0

实验使用的参数如下:交互置信度上限为 500,下限为 100,正向增量为 5,负向增量为 10;交互置信度阈值为 0.交互时间片阈值为 0.5ms;单次抢占时间片上限为 10ms,抢占率为 0.125.用户可根据不同的运行环境调整运行时参数,获得适应于特定环境的最佳性能.RDP 环境下的优化端口设置为 3389,VNC 环境为 5900.

实验负载有以下几种:

(1)cpuBomb:一个尽可能占用 CPU 的程序,用于模拟 CPU 密集型应用.

(2)PPT:全屏播放图片组成的 PPT,等待每一页刷新完毕后,等待 3s 翻下一页.该应用会产生大量的屏幕更新,因此对于延迟更加敏感.

(3)解压缩:使用 WinRAR 解压缩一个 700MB 的文件,作为实际的 CPU 密集型应用.

(4)混合负载:在 VM 内同时运行 PPT 和 WinRAR 解压缩,用于进一步模拟真实的使用场景.

5.1 优化效果评测

图 3 和图 4 分别给出了 RDP 和 VNC 环境下不同负载的平均操作延迟.其中,1 台 VM 运行 Windows XP,用于评测操作延迟;其他 5 台 VM 分别运行 cpuBomb,用于模拟高负载情况.数据表明,在原有的 Credit 调度算法下,操作延迟受其他 VM 的干扰很严重.对于非混合与混合负载,RDP 环境的操作延迟分别增加了 3.3 倍和 3.4 倍,VNC 环境下,混合负载的延迟增加了 2.5 倍.

TAVS 在非混合负载下的抗干扰效果非常明显,操作延迟仅增加了 1.9%和 41.6%.然而,对于混合负载,TAVS 的性能却低于 Credit 调度器.这是由于远程显示服务端口并不能与服务进程一一匹配,导致交互应用被误判为 CPU 密集型应用的可能性增加.但由于内核处理完后并不一定进行任务切换,因此 CPU 密集的任务可能运行在抢占状态下.

FUPS-DV 避免了上述情况的发生.在其他 5 个 CPU 密集型 VM 并存的情况下,运行混合负载的操作延迟仅增加了 13.8%和 50.9%,增幅分别为 Credit 调度器增幅的 4.0%和 20.4%,TAVS 的 3.0%和 16.0%.

图 5 显示了同时运行 5 个混合负载的 VM 时,播放 PPT 的平均延迟,用于模拟真实的应用环境.优化后 RDP 和 VNC 的显示延迟相比优化前分别降低了 77.3%和 60%.这证明 FUPS-DV 优化在多台交互式 VM 同时运行的情况下也是有效的.

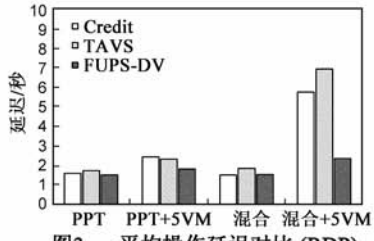


图3 平均操作延迟对比 (RDP)

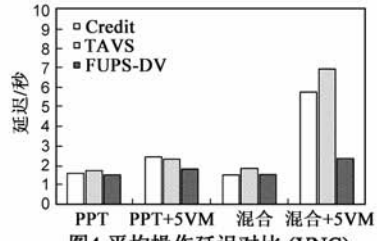


图4 平均操作延迟对比 (VNC)

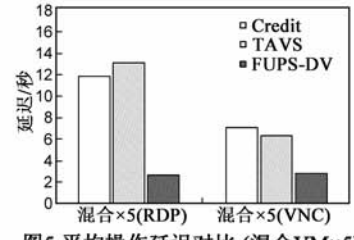


图5 平均操作延迟对比 (混合VM×5)

5.2 交互置信度

图 6 和图 7 分别给出了 RDP 和 VNC 环境下,不同任务的交互置信度随时间的变化.可以看到,在 RDP 环境下,PPT 在每次翻页时的 CPU 占用量会显著上升(导致交互置信度值下降);而 CSRSS 和 System 进程则大部分时间都处于最高值.VNC 环境下,PPT 的 CPU 占用量明显低于 RDP 环境;同样,VNC 的服务进程也占用了一定的 CPU,但占用的数量很小,所以交互置信度的变化很小.在两个环境下,对于 WinRAR 的评测都基本准确,集中分布在最低值.

图 6 也说明了对于同样的操作,RDP 环境下的 CPU 占用率高于 VNC.这也能说明为何使用 Credit 调度器,RDP 环境下的操作延迟受其他 VM 的干扰程度大大高于 VNC 环境.同样,在高负载情况下 TAVS 的混合负载延迟较高,很大程度上是因为 PPT 被误判为 CPU 密集型应用.

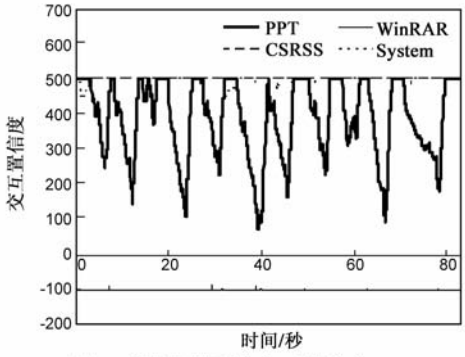


图6　混合负载下的交互置信度 (RDP)

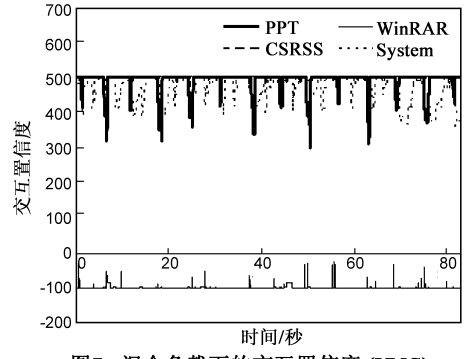


图7　混合负载下的交互置信度 (VNC)

5.3 CPU 分配公平性

图 8 和图 9 显示了优化前后,混合负载下的 CPU 占用情况.在 Credit 调度器下,WinXP 的 CPU 占用量颠簸较为严重.这是因为在交互过程中,由于 I/O 较为频繁,驱动虚拟机会频繁抢占 WinXP,导致后者只运行很短的时间就重新在运行队列中排队,大大减少了其 CPU 分配量.使用 FUPS-DV 后,CPU 占用非常平稳,而且所有 VM 的 CPU 占用量基本相同;这证明 FUPS-DV 可以很好的保证 CPU 分配的公平性.

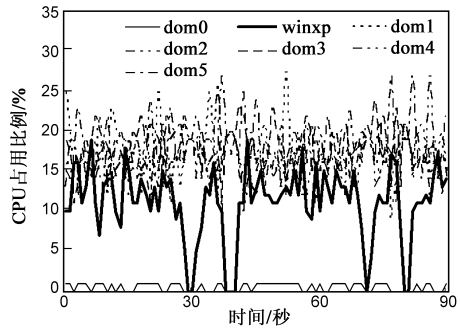


图8　使用Credit调度器的CPU占用 (RDP)

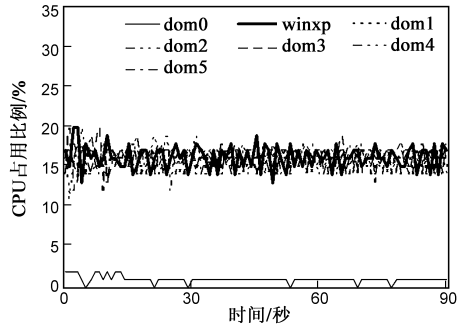


图9　使用FUPS-DV的CPU占用 (RDP)

6 结论和展望

桌面虚拟化由于其安全性、灵活性、多样性和易于管理等特点,已经得到越来越广泛的应用.传统的虚拟化技术并不能很好的适应桌面负载的混合性,无法满足用户对交互式性能的需求.分析了 Xen 原有调度算法交互性能低的原因在于其抢占机制与异步事件机制的不匹配,提出了全时抢占机制及其两个关键要素,并在此基础上设计并实现了 VM 调度算法 FUPS-DV.FUPS-DV 不需要对客户操作系统进行任何修改,且不依赖于特定的操作系统或远程服务平台.实验以 Windows XP 作为客户操作系统,分别以 RDP 和 VNC 作为远程服务,使用真实应用程序进行了评测.评测数据表明,在 5 台虚拟机同时运行混合负载的情况下,RDP 和 VNC 的显示延迟分别降低了 77.3% 和 60%.同时,实验也证明 FUPS-DV 能够保证客户操作系统对于资源占用的公平性.

FUPS-DV 目前只适用于单核系统,下一步的工作是将其应用于多核环境.在多核环境下,可以划分专门数量的处理器核优先处理交互请求,并可动态决定该类处理器核的数量,从而充分利用多核系统所带来的并行优势,进一步优化桌面虚拟化的交互性能.此外,将进一步通过量化的方法,确定不同参数对于交互性能的影响,以及参数之间的相互影响.同时,在实验方面,也将使用更符合实际应用的场景,从而更准确的评测优化效果以及额外负载.

参考文献

[1] Barham P, Dragovic B, et al. Xen and the art of virtualization [A]. Proceedings of the 19th Symposium on Operating Systems Principles (SOSP)[C]. Bolton Landing: ACM Press, 2003. 164 – 177.

[2] Yang C, Niu Y, Xia Y B, Cheng X. Performance analysis of interactive desktop applications in virtual machine environment [J]. The Chinese Journal of Electronics (CJE), 2008, 17(2): 242 – 246.

[3] Cherkasova L, Gupta D, Vahdat A. Comparison of the three CPU schedulers in Xen[J]. ACM SIGMETRICS Performance Evaluation Review, 2007, 35(2): 42 – 51.

[4] Govindan S, Nath A R, et al. Xen and co: communication-aware CPU scheduling for consolidated Xen-based hosting platforms[A]. Proceedings of the 3rd International Conference on Virtual Execution Environments (VEE)[C]. San Diego: ACM Press, 2007. 126 – 136.

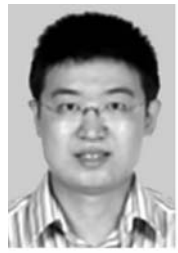
[5] Ongaro D, Cox A L, Rixner S. Scheduling I/O in virtual machine monitors[A]. Proceedings of the 4th International Conference on Virtual Execution Environments (VEE)[C]. Seattle: ACM Press, 2008. 1 – 10.

[6] Kim H, Lim H, et al. Task-aware virtual machine scheduling for I/O performance[A]. Proceedings of the 5th International Conference on Virtual Execution Environments (VEE)[C]. Washington, ACM Press, 2009. 101 – 110.

作者简介

夏虞斌 男,生于 1982 年. 2004 年获得复旦大学学士学位,2010 年获得北京大学计算机系博士学位. 现于复旦大学并行处理研究所做博士后. 研究方向包括虚拟化、操作系统、体系结构.

E-mail: xiayubin@fudan.edu.cn



杨 春 男,生于 1980 年. 2001 年获得北京大学学士学位,2008 年获得北京大学计算机系博士学位. 研究方向包括瘦客户计算、虚拟化、操作系统与系统软件等.

E-mail: yangchun@mprc.pku.edu.cn

程 旭 男,生于 1964 年. 北京大学教授、博士生导师,国家 863 项目集成电路设计专家组成员. 1994 年获得哈尔滨理工大学博士学位. 研究方向包括高性能微处理器设计、嵌入式系统、指令级并行、编译器优化、软硬件协同设计等.

E-mail: chengxu@mprc.pku.edu.cn