第 40 卷 第 11 期 计 算 机 工 程 2014 年 11 月
Vol. 40 No. 11 Computer Engineering November 2014

·专栏· 文章编号：1000-3428(2014)11-0031-05 文献标识码：A 中图分类号：TP393.09

一种虚拟机监控器的时间片动态调整机制

赵玉艳[1]，陈海宝[1,2]，赵生慧[1]

（1. 滁州学院计算机与信息工程学院，安徽 滁州 239000；2. 华中科技大学计算机科学与技术学院，武汉 430074）

摘 要： 在同一物理主机甚至同一 CPU/core 上，虚拟化技术使得多个虚拟机能够以公平共享物理资源的方式运行。然而，随着共享同一 CPU/core 的虚拟机数量的增多，每个虚拟机所经历的物理 CPU/core 访问延迟也在显著增加，造成异构型应用（运行过程中既有网络 I/O 需求，也有计算需求）在性能方面的负面影响。为解决上述问题，引入一种应用类型感知的虚拟机管理器时间片动态调整机制。该机制可以根据虚拟机中应用的类型动态调整虚拟机的时间片长度。实验结果表明，与 Xen Credit 调度机制相比，时间片动态调整机制可使异构型应用（Nginx Web 服务器）具有更好的响应能力和吞吐能力。

关键词： 应用类型驱动；虚拟机监控器；动态时间片；虚拟化；异构型应用

中文引用格式： 赵玉艳，陈海宝，赵生慧. 一种虚拟机监控器的时间片动态调整机制[J]. 计算机工程，2014，40(11)：31-35.

英文引用格式： Zhao Yuyan，Chen Haibao，Zhao Shenghui. A Dynamic Time Slice Adjusting Mechanism of Virtual Machine Monitor[J]. Computer Engineering，2014，40(11)：31-35.

A Dynamic Time Slice Adjusting Mechanism of Virtual Machine Monitor

ZHAO Yuyan[1]，CHEN Haibao[1,2]，ZHAO Shenghui[1]

（1. School of Computer and Information Engineering，Chuzhou University，Chuzhou 239000，China；
2. School of Computer Science and Technology，Huazhong University of Science and Technology，Wuhan 430074，China）

【Abstract】 Multiple Virtual Machines（VMs） can be hosted in the same CPU core with virtualization technologies，in a fair share manner of the physical resources among the VMs. However，as the number of VMs sharing the same core/CPU increase，the CPU access latency perceived by each VM also increases，which translates into longer network I/O processing latency experienced by heterogeneous application including both network I/O and computation. To mitigate such impact，an application type driven dynamic time slice adjusting mechanism is presented. The evaluation of a prototype in Xen shows that，compared with Credit scheduler of Xen，this mechanism improves the connection rate and response time of Nginx Web server.

【Key words】 application type driven；Virtual Machine Monitor（VMM）；dynamic time slice；virtualization；heterogeneous application

DOI： 10.3969/j.issn.1000-3428.2014.11.006

1 概述

云计算模式可以显著降低用户的投资以及运营成本，因为用户可以简单地以按使用付费的模式租用云资源来托管和运行应用。支持云托管的一项关键技术是虚拟机合并，也就是利用虚拟化技术将单台物理机划分为多个虚拟机，每个虚拟机被分配来运行客户的应用。尽管每个虚拟机通常被分配至少一个虚拟核（例如文献[1]中的 VCPU），但是虚拟核和物理核之间的映射并不总是一对一的。例如，在商业云平台 Amazon EC2[2] 中，其计算实例是以 EC2 计算单元（ECU）为单位进行分配。分配了一个计算单元（ECU）的 EC2 实例（虚拟机）的计算能力粗略地等同于一台配置了 1 GHz CPU 的物理机器。在一个配备了 3 GHz CPU 的物理机中，可能会有 3 个虚拟机共享一个物理 CPU。在这样的情况下，虚拟

基金项目： 安徽省自然科学基金资助面上项目（1408085MF126）；安徽省教育厅自然科学研究基金资助重大项目（KJ2011ZD06）；滁州学院优秀青年人才基金资助重点项目（2013RC006）；滁州学院科研启动基金资助项目（2014qd016）。

作者简介： 赵玉艳（1982－），女，讲师、硕士，主研方向：数据挖掘，云计算；陈海宝，讲师、博士研究生；赵生慧，教授、博士。

收稿日期： 2014-03-07 **修回日期：** 2014-04-24 **E-mail：** zhyy@chzu.edu.cn

机监控器(Virtual Machine Monitor,VMM)的 CPU 调度器(例如 Xen 默认的 Credit 调度器)以轮转的方式调度可运行的虚拟机,每个虚拟机被准许获得相同数量的物理 CPU 时间来确保共享 CPU 的虚拟机之间的公平性。然而,文献[3]通过实验指出 CPU 在多个虚拟机之间共享会对网络 I/O 处理延迟敏感的应用产生明显负面影响。一个需要处理网络 I/O 事件的虚拟机必须等到它访问物理 CPU 时才可以处理这个 I/O 事件。而在这种情况下,CPU 访问延迟趋向于每个虚拟机所分得默认物理 CPU 时间片的倍数(例如 Xen 中的30 ms),并且不能对虚拟机中的应用隐藏这样的延迟。这种影响非常不利于单纯的网络 I/O 密集型应用(如只支持对静态页面访问的 Web 服务器。需要指出,在这种场景中,CPU 大部分时间都处于空闲状态),同时这种影响也不利于既包含网络 I/O 部分,也包含计算部分的异构型应用。与单纯 I/O 密集型应用相比,异构型应用需要消耗相对多的 CPU 时间来完成计算任务。为避免产生歧义,现约定如下:术语“异构型应用”在本文中特指这样一类应用,在它们运行的过程中即对网络 I/O 有需求,也对计算有需求。例如,支持动态页面访问的 Web 服务器,相对于静态页面访问来说,一个 Web 服务器在收到客户的动态页面请求后,它需要通过一定的计算量来处理这个请求,在得到请求的结果后,对客户进行响应,在这种情况下,支持动态页面访问的 Web 服务器就是一个异构型应用。

如果对共享 CPU 的虚拟机使用默认的 CPU 时间片,运行着 Web 服务器的虚拟机可能不会及时地访问物理 CPU 来处理和响应请求。为了避免通信处理延迟所造成的影响,一种方法是选择使用一个独占物理 CPU 的虚拟机。然而,这可能会导致较高的成本,因此,它并不是成本敏感型客户的首选。本文关注于缓解共享 CPU 的虚拟机(例如 EC2 中的小型或者微型计算实例)所受到的网络 I/O 处理延迟影响,并针对上述问题,引入应用类型感知的虚拟机监控器时间片动态调整机制以提高应用性能。

2 相关研究

虚拟机管理器的 I/O 性能优化是一个热点研究领域。例如,文献[4]通过修改客户操作系统优化 KVM(一种开源的虚拟机管理器)的 I/O 性能。文献[5]对 Xen 的 I/O 虚拟化驱动进行了研究。缓解网络 I/O 处理延迟的一个途径是修改虚拟机监控器(VMM)的 CPU 调度器,让网络 I/O 密集型虚拟机(运行网络 I/O 密集型应用的虚拟机)的优先级高于 CPU 密集型虚拟机(运行 CPU 密集型应用的虚拟机)。例如,文献[6]在面对运行网络 I/O 密集型应用的虚拟机和运行 CPU 密集型应用的虚拟机时,优先调度前者。然而,这种方法在 CPU 份额分配上引入了短期的不公平。类似地,文献[7]使用了局部加速来帮助 I/O 密集型虚拟机抢占一个运行中的虚拟机以响应到来的 I/O 事件,但是它很难进行配置来保持共享 CPU 的虚拟机之间的公平性,因此,并不适用于建立在虚拟化技术之上的云计算系统。文献[8]扩展了 Xen 的 Credit 调度器来支持软实时应用,但是它可能给运行延迟敏感型应用的虚拟机更多的 CPU 时间,从而打破了虚拟机之间的公平性。Xen 的 Credit 调度器[9]使用了 BOOST 机制通过临时加速 I/O 密集型应用虚拟机(设置一个高的优先级)来缩短 I/O 响应时间。尽管这个机制对于单纯 I/O 密集型虚拟机非常有效,但是在异构型应用运行于虚拟机的情况下,一旦一个虚拟机通过 BOOST 机制获得调度来处理 I/O 请求,工作负荷中的计算密集部分会消耗它在当前调度周期中的 CPU 份额(Xen 中的术语是 Credits)。因为 Credit 调度器在虚拟机间是 CPU 公平的,当异构型应用的计算密集部分消耗完虚拟机所分得的 CPU 份额后,BOOST 机制在当前调度周期的后面部分中就会失去作用,从而导致高的 I/O 处理延迟。

针对 BOOST 机制在处理异构型应用时所面临的问题,本文提出一种基于时间片的动态调整机制来提高这类异构型应用的性能。

3 时间片动态调整机制

本节首先以一个例子来分析虚拟机共享 CPU 对异构应用 I/O 处理延迟的负面影响,然后用一个实验进行验证,最后引入时间片动态调整机制。

3.1 CPU 共享问题的分析及验证

在这个例子中,有 3 个单核虚拟机(VM1,VM2 和 VM3)共享一个物理 CPU 核,VM1 和 VM2 运行计算密集型应用,而 VM3 运行着一个异构型应用。VM3 中的应用等待客户请求的到来,然后以数据或控制信息响应这个请求。假设虚拟机监控器(VMM)中的 CPU 调度器使用一个被许多商业虚拟机平台采用的按比例分享的调度策略(例如,用在 Amazon EC2,RackSpace 和 GoGrid 商业云中的 Xen)。如图 1 所示,一个目的地是 VM3 的请求发送到物理机后,它首先被缓存在 VM3 之外(例如,在虚拟机监控器中或者在特权驱动器域中),一直等到 VM3 被调度。当 VM3 被调度后,它会处理这个请求并产生响应。如果 CPU 的时间片是 30 ms,请求的响应延迟可以高达 60 ms。而这样一个高延迟影响了 VM3 中应用的响应性以及请求处理率。

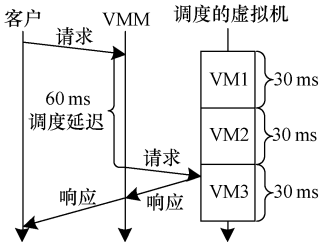

图 1　默认调度时间片示意图

为了展示延迟问题,设计了如下实验,在一台物理机(IBM System x3650:Intel Xeon E5530,32 GB RAM,500 GB Disk)上安装 Xen4.2,并使用默认的 Credit 调度器。每台虚拟机都运行了开源软件 lookbusy[10],用以保证虚拟机的 CPU 在实验期间处于忙碌状态。不失一般性,本文实验使用虚拟机中运行的 lookbusy 和网络连通性探测命令 ping 合成了一个异构型应用。然后,分别在 3 种情况下(1 台单核虚拟机独占一个物理核;2 台单核虚拟机共享一个物理核;3 台单核虚拟机共享一个物理核),从另外一台物理机上使用 ping 命令测试一个虚拟机的数据包往返时间 RTT(Round-Trip Time)。

图 2 展示了 ping 一台虚拟机时 RTT 的累积分布函数(CDF)。实验结果清晰地表明 ping RTT 随着共享 CPU 的虚拟机数量增加而增加。

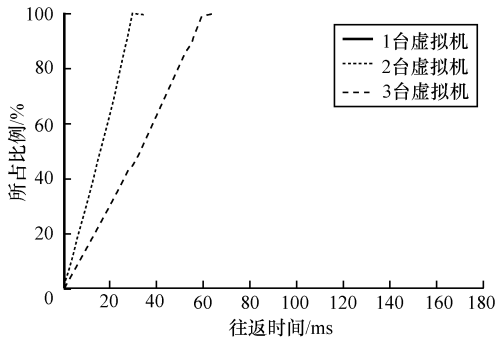

图 2　ping RTT 的累计分布函数

3.2　机制和算法

由 2.1 节可知,对于异构型应用,如果将调度时间片减小,虚拟机可以更频繁地访问物理 CPU,相应的 I/O 处理延迟也随之降低。然而,对于计算密集型应用来说,降低调度时间片会增加上下文切换的次数,从而会带来一定应用性能上的开销。也就是说,对于运行了计算密集型应用的虚拟机,并不适合于缩短它们的时间片。

为了解决上述问题,本文基于 Credit 调度器提出了一种应用感知的动态时间片调整机制,根据运行在虚拟机中应用的类型来动态调整此虚拟机的调度时间片,从而达到提高应用性能的目的,如 Web 服务器的连接率和响应时间。

(1)动态调整机制:动态调整机制的思路简述如下。首先,系统中只存在 2 种长度的时间片,一种是默认时间片(也就是 Credit 调度器中采用的 30 ms);另一种是短时间片。在本文中,短时间的长度是一个经验值,由管理员根据系统的实际情况设置(第 3 节的实验部分将比较短时间片被设置为不同的长度时,对应用性能的影响)。其次,系统在开始运行时,所有虚拟机都使用默认的时间片长度。如果虚拟机运行着同构型应用(如单纯的网络 I/O 密集型应用或者单纯计算密集型应用),则继续用默认的时间片。因为在默认时间片下,Xen Credit 调度器能够有效保证单纯计算密集型应用的性能,同时其 BOOST 机制也能保证单纯网络 I/O 密集型应用的性能[11]。而如果虚拟机中运行着异构型应用,系统则把此虚拟机的时间片调整为短时间片。

(2)应用类型推断算法:在实现时间片动态调整机制之前,调度器需要获取运行在虚拟机中的应用类型,由于虚拟机管理器和虚拟机之间存在着语义鸿沟的问题,虚拟机管理器并不能直接获取虚拟机内部运行应用的信息(如应用类型)。消除语义鸿沟一般采用的是侵入式方法:在虚拟机操作系统内核中增加一个超级调用,通过这个超级调用来向虚拟机管理器传递虚拟机内部应用的类型信息。本文采用一种非侵入式的方法,即不需要修改虚拟机操作系统内核,VMM 只需监控一些信息从而推断出应用的类型。

本文的设计包括一个监控器用来分析和推断运行在虚拟机中的应用类型以及一个时间片调整算法。图 3 显示了调度器的框架,它可以监控虚拟机的行为以及动态调整时间片。

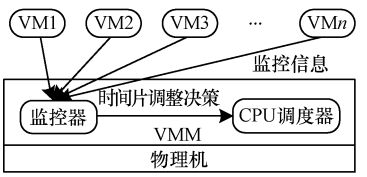

图 3　调度器框架

行为监控器被周期性调用以监控每个虚拟机数据包接收频率(在一个监控间隔内每个虚拟机接收数据包的数量)以及虚拟机的 VCPU 利用率。其中,数据包接收频率用来推断运行在虚拟机上的应用是否为网络 I/O 密集型应用,虚拟机的 VCPU 利用率则用来推断运行的是否为计算密集型应用。将 VCPU 利用率和数据包接收频率结合起来,则可用来推断虚拟机中运行的是否为异构型应用。

类型推断算法如下:

(1)全局变量

N:需要考虑的监控间隔的数量

PacketThreshold:区分网络 I/O 密集型应用的阈值

CPUThreshold:区分计算密集型应用的阈值

(2)每个虚拟机所属的变量

AppType:应用类型,HETERO(异构)或 HOMO(同构)

PacketArray[N]:数组,记录虚拟机在每个间隔中收到的数据包量

CPUArray[N]:数组,记录虚拟机在每个间隔中的 VCPU 利用率

CurrentPointer:数组指针,指明当前数组元素,初始值为 0

```
Begin
  for i = 0 to N - 1 {
    if (PacketArray[i] > PacketThreshold) {
      for j = 0 to N - 1 {
        if (CPUArray[j] > CPUThreshold) {
          Set APPType to HETERO;
                Goto UpdatePointer;
            }
        }
    }
    Set VMType to HOMO;
  UpdatePointer:
    Increment CurrentPointer;
    Set CurrentPointer to 0 if greater than N - 1;
    Reset PacketArray[CurrentPointer] to 0;
    Reset CPUArray[CurrentPointer] to 0;
    Set the next time of activating the monitor;
End
```

为了避免因监控数据短暂变化所引起的类型变化,算法在进行类型推断时考虑了每台虚拟机在 N 个监控周期的数据包接收频率以及 VCPU 利用率,N 在本文中设置为 5。在虚拟机的生命周期中,根据行为监控的反馈信息,每个虚拟机的类型可以变化。因为随着监控频率的增加,监控开销也会增加,所以文中采用的监控间隔为 1 s。

3.3 原型实现

本工作以 Xen4.2 中的 Credit 调度器为基础设计并实现了一个原型系统。

在 Credit 调度器的源代码(sched_credit.c 文件)中通过添加程序段实现监控虚拟机 VCPU 利用率的功能。程序段首先统计每个虚拟机在一个监控周期内获得的物理 CPU 时间(也就是虚拟机中每个 VCPU 在一个监控周期中占用物理 CPU 时间之和),然后利用下面的公式计算虚拟机的 VCPU 利用率:$Utilization = 100 \times cpuTime/(T \times VCPUsNum)$,其中,$cpuTime$ 是虚拟机在一个监控周期内总共获得的物理 CPU 时间;T 是在一个监控周期应获得的物理 CPU 时间;$VCPUsNum$ 是虚拟机的 VCPU 数量。

在监控虚拟机数据包接收量方面,首先在特权域 dom0 的网络后端设备(netback)源代码(netback.c)中添加程序段,监控虚拟机的数据包接收数量并将相应信息写入 dom0 和虚拟机监控器(Virtual Machine Monitor,VMM)之间的共享内存页(shared info page),然后,虚拟机管理器在需要时可以从共享内存页中读取该信息。

获取以上 2 个参数值后,类型推断算法根据系统设定的阈值判断每个虚拟机中运行应用的类型。本文设定的虚拟机 VCPU 利用率阈值为 50%,包接收率的阈值设置为每秒 30 个数据包。在实际应用中,系统管理员可以根据系统的运行情况修改上面 2 个阈值。

为了实现时间片动态调整机制,在 Credit 调度器的源代码中添加了如下的控制逻辑:(1)在虚拟机结构体 *csched_dom* 中添加 3 个变量:分别是 *tslice_ms*,*tick_period_ms*,*ticks_per_tslice*。其中,*tslice_ms* 表示时间片的长度;*tick_period_ms* 表示 Credit 值的记账周期;*ticks_per_tslice* 表示每个时间片包含记账周期的数量,这 3 个变量的关系是:第一个变量的值等于后 2 个变量值的乘积,即 $tslice\_ms = tick\_period\_ms \times ticks\_per\_tslice$。(2)根据类型算法得到的应用类型,在 3 个变量中设置相应的值。例如,如果应用类型是异构型,那么将 *tslice_ms* 设置为短时间片,如 12(单位是 ms),*tick_period_ms* 和 *tick_period_ms* 则分别对应的设置为 4(单位是毫秒)和 3。如果应用类型是同构型,3 个变量设置为系统的默认值。(3)*csched_schedule* 函数开始执行的时候,从虚拟机结构体 *csched_dom* 中读取 *tslice_ms* 的值,同时将 *csched_tick* 函数(作用是定时扣除 Credit 值)中定时器的定时周期设置为 *tick_period_ms*。

另外,由于并未修改 Xen Credit 调度器的 CPU 时间分配和消耗机制,因此继承了 Credit 调度器的 CPU 公平性保证策略,也就是能够保证物理 CPU 时间在虚拟机之间得到公平共享。

4 实验与分析

本文实验的目的是测试动态时间片机制对异构型应用的性能影响。在实验中,依次将动态调节机制中的小时间片设定为 24 ms,18 ms,12 ms,6 ms,3 ms。然后在每种时间片下执行一遍实验,最后将每种时间片下的性能数据与默认时间片(30 ms)下的性能数据进行比较。

4.1 实验环境设置

物理服务器(IBM System x3650:Intel Xeon E5530,32 GB RAM,500 GB Disk)运行了 4 个共享同一物理核的单核虚拟机(VM1,VM2,VM3 和 VM4)。为了保证 CPU 分配的公平性,在实验开始前利用 Xen 的控制端命令“xm sched-credit -d ‘vmName’ -w 256 -c 25”,将每个虚拟机的 *weight* 值设置为默认值(256),*cap* 值

设置为25,也就是每个虚拟机最多获得25% 的 CPU 时间。虚拟机 VM1 运行了 Nginx 和 lookbusy 合成的一个异构型应用;虚拟机 VM4 运行了网络 I/O 密集型应用(Nginx);为了增大对 VM1 和 VM4 中应用的干扰,从而能够更清晰地体现出本文所提机制的优势,在剩余的2 个虚拟机(VM2 和 VM3)中分别运行了计算密集型应用(lookbusy)。2 台客户端机器分别同时使用 Httperf[12] 产生对 VM1 和 VM4 中一个 4 KB 静态 Web 页面的请求,以测量 VM1 和 VM4 中 Web 服务器的性能。

实验中使用 Nginx Web 服务器的静态页面查询和 lookbusy 来合成一个异构型应用,使其既有网络 I/O 密集型部分,也有计算密集型部分。其中,Nginx 对网络 I/O 处理延迟非常敏感,网络 I/O 处理延迟会影响 Nginx 服务器的响应时间和请求处理的吞吐量。Lookbusy 用来产生 CPU 负载(使虚拟机的 VCPU 利用率保持在 50%)。

为了进行比较,将并发连接请求率从每秒 100 个逐渐增加到每秒 500 个,步长为 100,然后在每种连接请求率下分别测试了不同方法(默认时间片和小时间片)下的 Web 服务器响应时间和连接率等2 个指标。

4.2　结果分析

图 4 和图 5 分别展示了 VM1 的响应时间和连接率。

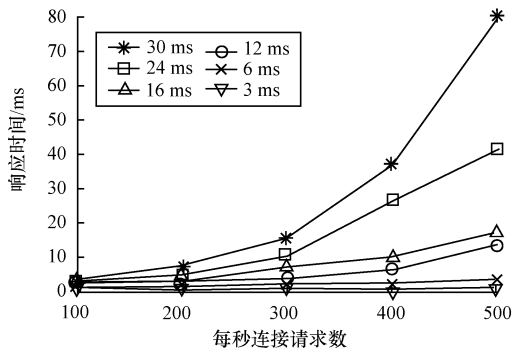

图 4　VM1 中不同连接请求率下的响应时间对比

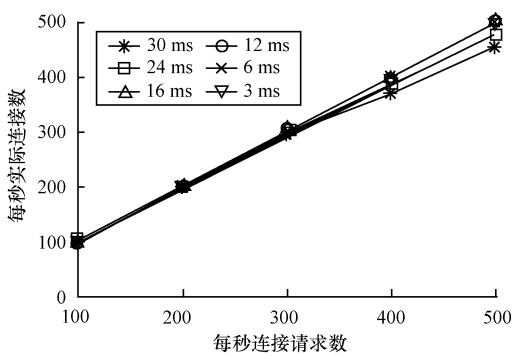

图 5　VM1 中不同时间片下的连接率对比

从图 4 中可以看到,随着连接请求率的增加,默认时间片(30 ms)下的 Web 服务器响应时间从每秒 100 个连接下的 3.4 ms 迅速升高到每秒 500 个连接下的 79.9 ms。相比较而言,小时间片下的 Web 服务器响应时间则有明显优势,尤其是在 3 ms 的短时间片下,响应时间只从 0.6 ms(每秒 100 个连接)升高到 1.1 ms(每秒 500 个连接)。

从图 5 中的数据可以得到 3 个信息:(1)相对于默认时间片(30 ms)而言,随着连接请求率的增加,小时间片在连接率方面的优势逐渐体现出来。(2)连接请求率在不大于每秒 300 个时,默认时间片和短时间片在连接率方面性能相同,这主要是因为,连接请求率没有超出各种时间片下的服务器处理能力。当超出服务器处理能力后,也就是,连接请求率大于每秒 300 个时,不同时间片下的性能差距才体现出来。(3)在 18 ms,12 ms,6 ms 和3 ms 等4 种短时间片下连接率等于连接请求率(4 种短时间下的数据重叠在一起),这表明系统在分别采用上述 4 种短时间片下,连接请求率并未超出服务器的处理能力;同时也表明在时间片为 18 ms 时,服务器已经能够有效地处理客户端发来的请求,因此,不需要进一步将时间片设置得更小。

因为 VM2 和 VM3 中运行的是计算密集型应用,本文所提动态机制并不会调节它们的时间片长度,也就是它们所运行的计算密集型应用性能与默认环境下的性能相当。此外,由于 VM4 中运行的是单纯的网络 I/O 密集型应用,动态时间片调整机制也不改变它的时间片长度。在 BOOST 机制的作用下,其 Web 服务器的性能(请求连接率和响应时间)同样与默认环境下的性能相当。因此,此处省略了 VM2,VM3 和 VM4 的性能对比图。

以上的实验结果表明,与默认时间片相比,短时间片能够有效提高虚拟机中异构型应用的性能。

5　结束语

本文设计了一个应用类型感知的虚拟机管理器动态时间片调整机制,并在 Xen Credit 调度器基础之上实现了一个原型系统。本文基于以下思想:对于运行异构型应用的虚拟机,将其默认时间片(如 Xen Credit 调度器中默认的 30 ms)分成多个小时间片(如 10 个 3 ms 的时间片),从而达到在一个调度周期中多次对其调度的目的。也就是说,相对于默认时间片的虚拟机,短时间片虚拟机可以更频繁地访问物理 CPU 来处理 I/O 事件。原型系统的实验结果表明,在 Web 应用的响应时间和连接率方面优于 Xen Credit 调度器。

(下转第 41 页)

记忆效应的加入使得在节点间观点差值大于置信值时,节点可通过多次记忆值累加直至突破记忆阈值 M_t 的方式进行有限次观点更新,从而促进了网络一致性观点的形成。

参考文献

[1] 李　翔.复杂动态网络传播动力学[J].力学进展,2008,38(6):723-732.

[2] 刘　云,丁　飞,张振江.舆论形成和演进模型的研究综述[J].北京交通大学学报:自然科学版,2010,34(5):83-88.

[3] 熊　熙,胡　勇.基于社交网络的观点传播动力学研究[J].物理学报,2012,61(15):104-110.

[4] Krawiecki A,Hołyst J A,Helbing D. Volatility Clustering and Scaling for Financial Time Series Due to Attractor Bubbling[J]. Physical Review Letters,2002,89(15).

[5] Krapivsky P L. Kinetics of Monomer-monomer Surface Catalytic Reactions [J]. Physical Review A, 1992, 45(2):1067.

[6] Sood V, Redner S. Voter Model on Heterogeneous Graphs[J]. Physical Review Letters,2005,94(17).

[7] Galam S. Minority Opinion Spreading in Random Geometry [J]. The European Physical Journal B-Condensed Matter and Complex System,2002,25(4):403-406.

[8] Krapivsky P L,Redner S. Dynamics of Majority Rule in Two-state Interacting Spin Systems[J]. Physical Review Letters,2003,90(23).

[9] Slanina F,Lavicka H. Analytical Results for the Sznajd Model of Opinion Formation[J]. The European Physical Journal B-Condensed Matter and Complex System, 2003,35(2):279-288.

[10] Deffuant G,Neau D,Amblard F,et al. Mixing Beliefs Among Interacting Agents [J]. Advances in Complex Systems,2000,3(1):87-98.

[11] Hegselmann R, Krause U. Opinion Dynamics and Bounded Confidence Models, Analysis, and Simulation [J]. Journal of Artificial Societies and Social Simulation,2002,5(3):216-219.

[12] Yang H X,Wang W X,Lai Y C,et al. Convergence to Global Consensus in Opinion Dynamics Under a Nonlinear Voter Model [J]. Physics Letters A, 2012, 376(4):282-285.

[13] Nardini C,Kozma B,Barrat A. Who's Talking First? Consensus or Lack Thereof in Coevolving Opinion Formation Models [J]. Physical Review Letters, 2008, 100(15).

[14] Kozma B,Barrat A. Consensus Formation on Adaptive Networks[J]. Physical Review E,2008,77(1).

[15] Grauwin S,Jensen P. Opinion Groups Formation and Dynamics: Structures That Last from Non Lasting Entities[J]. Physical Review E,2012,85(6).

[16] 宋冰冰,王海艳.马太效应综述[J].社会心理科学,2005,20(1):11-12.

[17] Erdös P, Rényi A. On Random Graphs I [J]. Publicationes Mathematicae Debrecen,1959,6:290-297.

[18] Albert R,Barabási A L. Statistical Mechanics of Com-plex Networks[J]. Reviews of Modern Physics,2002,74(1):47.

编辑　索书志

(上接第 35 页)

目前原型系统采用的短时间片长度是由系统管理员根据经验提前设定的,系统根据虚拟机的应用类型,动态地在指定的短时间片和默认时间片之间进行切换。因此,下一步的研究工作就是设计一种无需系统管理员干涉的短时间设定机制,并且考虑如何将动态时间片与 CPU 运行队列的动态优先级结合起来,进一步提升此类应用的性能。

参考文献

[1] 薛海峰,卿斯汉,张焕国. XEN 虚拟机分析[J].系统仿真学报,2008,19(23):5556-5558.

[2] 亚马逊公司. AWS 产品与解决方案[EB/OL].[2014-03-01.]http://aws.amazon.com/cn/ec2.

[3] Xu C,Gamage S,Kompella R,et al. vTurbo:Accelerating Virtual Machine I/O Processing Using Designated Turbo-Sliced Core [C]//Proceedings of USENIX Annual Technical Conference. [S. l.]: USENIX Press, 2013: 243-254.

[4] 张彬彬,汪小林,杨　亮,等.修改客户操作系统优化 KVM 虚拟机的 I/O 性能[J].计算机学报,2010,33(12):2312-2320.

[5] 胡冷非,李小勇.基于 Xen 的 I/O 准虚拟化驱动研究[J].计算机工程,2009,35(23):258-262.

[6] Govindan S,Nath A R,Das A,et al. Communication-aware CPU Scheduling for Consolidated Xen-based Hosting Platforms [C]//Proceedings of the 3rd Interna-tional Conference on Virtual Execution Environments. [S. l.]: ACM Press,2007:126-136.

[7] Kim H,Lim H,Jeong J,et al. Task-aware Virtual Machine Scheduling for I/O Performance [C]//Proceedings of ACM International Conference on Virtual Execution Environments. [S.l.]:ACM Press,2009:101-110.

[8] Lee M,Krishnakumar A S,Krishnan P,et al. Supporting Soft Real-time Tasks in the Xen Hypervisor [C]// Proceedings of the 6th International Conference on Virtual Execution Environments. Pittsburgh, USA: [s. n.],2010:97-108.

[9] Xen Project Advisory Board. Xen Credit Scheduler[EB/OL].[2014-03-01]. http://wiki.xen.org/wiki/Credit_Scheduler,

[10] Devin Carraway. A Synthetic Load Generator [EB/OL]. [2014-03-01]. http://www.devin.com/lookbusy/.

[11] Gamage S, Kangarlou A, Kompella R R, et al. vSlicer: Latency-aware Virtual Machine Scheduling via Differ-entiated-frequency CPU Slicing[C]//Proceedings of the 21st International Symposium on High-performance Parallel and Distributed Computing. [S.l.]: ACM Press, 2012: 3-14.

[12] HP Inc. A Tool for Measuring Web Server Performance [EB/OL]. [2014-03-01]. http://www.hpl.hp.com/research/linux/httperf/.

编辑　顾逸斐