ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.4, April 2010, pp.849−862
doi: 10.3724/SP.J.1001.2010.03492

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于虚拟机的高效磁盘 I/O 特征分析方法*

沈玉良[1,2+], 许 鲁[1]

[1](中国科学院 计算技术研究所,北京 100190)

[2](中国科学院 研究生院,北京 100049)

Efficient Disk I/O Characteristics Analysis Method Based on Virtual Machine Technology

SHEN Yu-Liang[1,2+], XU Lu[1]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail:shenyuliang@nrchpc.ac.cn, http://www.bwstor.com.cn

Shen YL, Xu L. Efficient disk I/O characteristics analysis method based on virtual machine technology. *Journal of Software*, 2010,21(4):849−862. http://www.jos.org.cn/1000-9825/3492.htm

Abstract: Disk I/O may be the bottleneck in computer systems because of the mechanism characters in disk system. In order to tune the system performance effectively, the collection of the disk I/O workload characteristics will be the essential step for the performance optimization. Compared with other I/O characteristics collection methods, this paper presents an on-line I/O characteristics analysis method based on Xen 3.0 virtual machine system. In virtual machine environments, this method can be transparent to the unmodified operating system. With this method, several essential I/O characteristics metrics can be collected, such as disk I/O block size, I/O latency, I/O arrival interval, I/O spatial locality statistics, I/O time locality statistics and the I/O operation hotspot distribution. Through the testing and analysis, this method is found to have little system overhead and impact on the application system I/O performance. In addition, the I/O characteristics analysis results are demonstrated on the large file copy workload and the Filebench benchmark workload generated by the filemicro and varmail personalities.

Key words: virtual machine; disk I/O characterization; performance optimization; on-line I/O analysis; file system

摘 要: 由于磁盘系统的机械运动本质,磁盘系统 I/O 往往会成为计算机系统的性能瓶颈.为了有效地提高系统性能,收集和分析应用系统的磁盘 I/O 特征信息将成为性能优化工作的重要基础.与以往 I/O 特征分析方法不同,给出了一种基于 Xen 3.0 虚拟机系统的磁盘 I/O 特征在线分析方法.在虚拟机环境下,该磁盘 I/O 特征采集方法可以透明地应用于任意无须修改的操作系统.该方法可以高效地在线采集多种基本 I/O 特征数据,其中包括:磁盘 I/O 块大小、I/O 延迟、I/O 时间间隔、I/O 空间局部性、时间局部性以及磁盘 I/O 操作热点分布.通过测试和分析,该在线 I/O 分析方法有着较小的系统开销,并且对应用系统 I/O 性能的影响很小.此外,还给出了在大文件拷贝、基于 Filebench 的 filemirco 和 varmail 等工作负载下的 I/O 特征分析结果.

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z184 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2004CB318205 (国家重点基础研究发展计划(973))

Received 2008-03-07; Revised 2008-07-02; Accepted 2008-10-09; Published online 2009-06-05

关键词: 虚拟机;磁盘 I/O 特征;性能优化;在线 I/O 分析;文件系统
中图法分类号: TP316 文献标识码: A

目前,与 CPU、内存技术的快速发展相比,存储系统性能增长速度相对较慢,并且成为计算机系统的主要性能瓶颈之一.因此,应用系统整体性能的表现越发地依赖于存储系统的 I/O 性能.而且不同的存储系统配置有可能直接影响应用系统的性能,比如通过优化文件系统的 Layout 可以提高系统读取性能[1].那么,研究和分析应用系统的磁盘 I/O 访问特征就成为存储系统优化的重要基础.

当然,我们可以从应用本身到物理设备的多个层次获取应用的 I/O 行为特征.然而无论在应用程序中,还是在操作系统驱动程序中收集应用系统的 I/O 行为,我们都需要应用程序提供商或操作系统提供相关的支持.因此,这些磁盘 I/O 特征分析方式都只适用于特定的应用或操作系统.同时,以往很多磁盘 I/O 特性数据收集和分析方法主要是在用户操作系统中插入 I/O 操作跟踪程序,或者在系统平台加入 I/O 分析硬件设备.目前,许多商业操作系统没有提供相应的 I/O 操作跟踪工具,并且,相应的硬件 I/O 分析设备又相当少见.从应用环境来看,在操作系统中加入 I/O 操作跟踪程序的方法往往只适合于开发测试系统.在生产环境中,企业用户通常会担心由 I/O 操作跟踪程序可能引入的安全和稳定性问题.

近年来,虚拟机技术成为系统研究和应用的热点.目前,虚拟机技术在开发调试、服务器整合、高可用技术以及系统容灾等方面有着广泛的应用前景.在虚拟机环境中,虚拟机系统使用软件技术来模拟磁盘 I/O 访问接口.因此,我们可以在虚拟机软件中提供 I/O 操作分析程序.这样,它就可以独立于用户操作系统运行,而且适用于各种用户操作系统和应用环境.值得一提的是,应用系统提供商通常不会提供数据 I/O 的特征信息.这是因为,即便是同一应用系统,不同的用户使用方式和系统设置也可能会导致其 I/O 访问特征发生变化.另外,不同版本的应用系统往往需要不同版本的 I/O 特征分析程序.由此我们可以看出,基于虚拟机系统的磁盘 I/O 特征分析将会具有应用系统无关性和广泛的适应性等优势.

存储系统的优化工作需要的往往不是 I/O 操作 Trace,而是磁盘 I/O 行为的特征数据,比如 I/O 块的尺寸分布、时间局部性信息、空间局部性信息以及 I/O 延迟分布等信息.这些特征量往往是系统优化和存储系统参数调整的重要依据.如果能够在线获取磁盘 I/O 的特征信息,那么不仅可以方便系统管理员的优化工作,而且可以为自动调整系统参数提供支持.因此,本文将着重研究基于虚拟机系统的高效磁盘 I/O 特征在线分析方法.

针对上述情况,本文设计和实现了基于开源虚拟机 Xen 的在线磁盘 I/O 特性分析方法.该方法可以有效地在线统计磁盘 I/O 块大小分布、I/O 操作间隔分布情况、I/O 延迟、I/O 操作热点分布以及 I/O 访问的空间/时间局部性等信息.

本文第 1 节介绍 Xen 虚拟机系统中设备 I/O 访问的系统结构.第 2 节描述基于虚拟机的磁盘 I/O 特性分析方法的设计和实现.第 3 节给出针对本 I/O 特征分析方法的测试结果.随后,利用本文的分析方法,我们针对大文件拷贝、Filebench 测试程序进行测试实验,而且依据磁盘 I/O 特征分析结果给出相关分析评价.第 4 节介绍相关研究工作.第 5 节给出结论和未来的工作方向.

1 虚拟机系统 Xen 的系统结构

本文以 Xen 3.0.5 作为分析虚拟机磁盘访问模式和分析磁盘 I/O 特征数据的系统实验平台.在 Xen 中,用户操作系统对磁盘等 I/O 设备访问的系统结构如图 1 所示.在 Xen 虚拟机系统中,Dom0 负责虚拟化管理和 I/O 设备模拟等工作.同时,用户应用程序和操作系统运行在 HVM(hardware virtual machine) Domain 之内.用户操作系统通过其磁盘和网络设备驱动程序访问虚拟的磁盘和网络设备.这时,Xen 的 Hypervisor 可以截获 HVM Domain 对相关 I/O 端口和内存映射的 I/O 区域的访问.然后,Xen 虚拟机软件利用事件传递通道将访问请求发送给 Dom0 中的 I/O 设备模拟程序.在 I/O 设备模拟程序中,系统可以模拟磁盘 IDE(integrated drive electronics)接口或 SCSI(small computer system interface)接口.I/O 设备模拟程序会根据 I/O 访问请求内容访问本地磁盘设备上或者网络存储设备上的数据.最后,I/O 设备模拟程序会把 I/O 请求的处理结果返回给 HVM Domain.在 Dom0 中实

现I/O设备模拟机制的好处主要在于系统可以重复利用Dom0内核的设备驱动程序.从而不必像VMware ESX Server那样,在虚拟机软件中实现多种I/O设备的驱动程序.值得注意的是,Dom0中的磁盘设备虚拟化过程独立于用户操作系统,并且在用户空间实现.因此,我们选择在Dom0中的I/O设备模拟程序中实现本文的磁盘I/O特征分析机制.

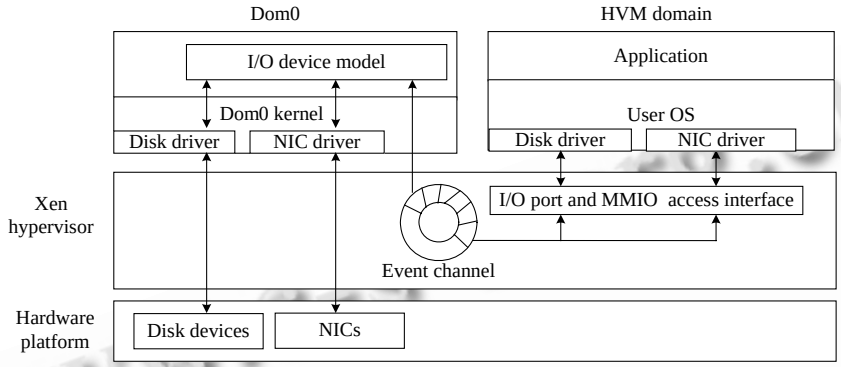


Fig.1 Architecture of I/O device emulation in Xen virtual machine system

图1 Xen虚拟机系统中的I/O设备模拟结构图

需要说明的是,虚拟机系统的I/O虚拟化操作不可避免地增加了用户操作系统的磁盘I/O访问路径和响应时间.比如:在物理平台上,用户操作系统通过I/O端口可以直接访问硬件平台.而在Xen系统中,用户操作系统对I/O端口的访问会首先被Xen虚拟机软件所截获.随后,相关的I/O请求被转发给Dom0.然后,Dom0系统中的设备模拟程序处理该I/O请求.最后,Xen系统再把Dom0的处理结果返回给用户操作系统.由此可见,引入虚拟机软件后,磁盘I/O操作的虚拟化开销是影响磁盘I/O性能的主要因素.磁盘I/O的虚拟化开销是虚拟机软件整体虚拟化开销的主要组成部分.此外,如果同一平台上多个虚拟机系统的磁盘I/O请求落在相同的物理设备上,有可能导致更多更复杂的磁盘寻址操作产生.这样就会进一步增加磁盘I/O响应时间并降低磁盘I/O操作性能.

然而,虚拟机软件和硬件平台为用户操作系统提供了CPU和内存的虚拟化支持,同时还进行了系统I/O设备和总线的模拟.因此,用户操作系统本身无法检测和判断到其运行在虚拟机系统还是物理硬件平台之上.据此我们可以看出,虚拟机软件的引入对于用户操作系统磁盘I/O性能的主要影响体现在增加了磁盘I/O访问路径和相应时间上.

为了有效减小磁盘I/O虚拟化的影响以及提高虚拟机系统的磁盘I/O性能,我们可以在软、硬件两方面采用相关技术.在硬件方面,我们可以采用带有I/O虚拟化支持的物理平台、高性能的SCSI磁盘、RAID(redundant array of inexpensive disks)磁盘阵列技术或者更理想地采用SAN(storage area network)存储域网络进行数据集中存储.在软件技术方面,我们也有许多优化技术可以用来提高虚拟机系统的磁盘I/O性能.比如在虚拟机软件内增加以内存或本地磁盘为介质的数据缓存机制,这样可以有效缩短磁盘I/O响应时间和磁盘I/O访问路径.同时,我们还可以通过分析和调整磁盘数据的分布来缩短虚拟机系统的磁盘寻址距离和时间.

对于各种软件方面的存储优化技术,我们都需要获取虚拟机系统的磁盘I/O信息,尤其是典型的磁盘I/O特征信息.随后,本文将着重介绍基于虚拟机系统的磁盘I/O特征信息的在线获取方法.

2 磁盘I/O特征数据的采集方法

如前一节所述,在Xen中,HVM Domain内的用户操作系统使用和访问虚拟的磁盘设备.此时,HVM Domain看到的是标准的IDE或SCSI接口.用户操作系统可以像使用物理设备一样,通过标准的磁盘I/O指令访问虚拟磁盘设备.Dom0中的I/O设备模拟程序会截获来自HVM Domain的磁盘设备I/O访问请求.随后,I/O设备模拟程序会进行相应的I/O虚拟化处理.我们将在I/O设备模拟程序中插入磁盘I/O特征测量程序,并用其来获取磁盘I/O特征数据.

2.1 磁盘I/O特征数据量

磁盘 I/O 特征数据的主要应用之一就是给系统性能优化提供依据.在此类应用中,往往并不需要精确记录每个磁盘 I/O 请求的相关信息,而是需要分析系统工作负载的磁盘 I/O 行为特性,如磁盘 I/O 数据块大小的分布情况、读写分布情况等.由于各 I/O 特征量的统计结果数据规模相对较小,我们就可以将 I/O 特征数据的统计结果以在线的方式呈现给用户.

为了便于统计磁盘 I/O 行为特征,我们把磁盘看成由扇区(sector)组成的一维数组.这样,每个 I/O 读写请求就是对数组中连续多个扇区的访问.

2.1.1 磁盘 I/O 读写分布

磁盘 I/O 的读写操作比例是最常见的 I/O 特征量,并且是优化存储系统性能的重要依据.读操作和写操作往往在同一个存储系统中有着不同的系统代价和优化方案.因此,我们会记录读写操作计数,并且分别记录读写操作的块大小、I/O 延迟、I/O 时间间隔、空间局部性信息、并发 I/O 数量以及 I/O 操作热点等信息.

2.1.2 磁盘 I/O 块大小

磁盘 I/O 的块大小分布是重要的磁盘 I/O 特征量之一.这是因为磁盘 I/O 块大小分布会影响存储系统的优化策略.比如,应用系统负载的 I/O 块大小往往决定磁盘 RAID 系统条带大小的优化设置[2].

为了有效记录磁盘 I/O 块大小的分布,我们创建了一个包含 N_{block} 个元素的整数数组.每个数组元素负责记录磁盘 I/O 块大小落在一定范围内的磁盘 I/O 计数.每个数组元素所代表的范围是 M 个扇区.也就是说,数组的第 i 个元素负责记录磁盘 I/O 块大小在$(i-1)\times M$ 和 $i\times M$ 个扇区之间的磁盘 I/O 操作计数.需要说明的是,数组的第 N_{block} 个元素表示磁盘 I/O 块大小比$(N_{block}-1)\times M$ 个扇区大的磁盘 I/O 计数.系统管理员可以根据需要设置 N_{block} 和 M 的大小.我们采用 N_{block} 的默认设置为 512,M 的默认设置为 8.

2.1.3 磁盘 I/O 延迟

磁盘 I/O 延迟是关系到存储系统性能的重要指标.I/O 延迟的大小不仅取决于应用系统的 I/O 数量,而且可以反映出存储系统的忙闲程度.比如:在一个同时服务于其他应用系统的存储系统上,我们将会得到较长的 I/O 延迟时间.

为了准确记录磁盘 I/O 延迟时间,我们以 CPU 周期数来记录每次 I/O 的起始和结束时间.那么,我们用每次 I/O 的结束时间减去其起始时间,就可以得到该次 I/O 请求的延迟时间.为了有效地在线统计磁盘 I/O 延迟,我们创建包含 $N_{latence}$ 个元素的整数数组.数组元素分别代表同等大小的连续时间区域.数组第 i 个值负责记录延迟时间刚好落在第 i 个时间区域内的 I/O 访问计数.

2.1.4 磁盘 I/O 时间间隔

磁盘 I/O 的时间间隔可以体现出应用系统 I/O 访问请求的频繁程度.我们同样以 CPU 周期数作为记录时间的基本单位.为了有效地在线统计磁盘 I/O 时间间隔,我们创建了包含 $N_{interval}$ 个元素的整数数组.每个数组元素代表同等大小的连续时间区域.数组第 i 个值负责记录磁盘 I/O 时间间隔刚好落在第 i 个时间区域内的 I/O 访问计数.

2.1.5 磁盘 I/O 的空间局部性信息

磁盘 I/O 的空间局部性特征是进行存储系统性能优化的重要依据之一.比如,数据访问的预读机制就是利用了磁盘 I/O 的空间局部性原理.我们把两次 I/O 请求的空间距离定义为前一次磁盘 I/O 请求的结束逻辑地址和后一次磁盘 I/O 请求的起始逻辑地址之差.我们把各磁盘 I/O 请求的空间距离记录在包含 N_{space} 个元素的空间距离数组中.每个数组元素负责记录对应空间距离的 I/O 操作计数.由此可以看出,对于顺序访问的两个磁盘 I/O 请求之间的空间距离为 1.空间距离为$-BLOCK_SIZE+1$,表示某 I/O 流中有连续的两个磁盘 I/O 请求访问同一起始数据块.

然而,磁盘 I/O 负载在多数情况下并不是单独的一个 I/O 顺序流.通常情况下,对磁盘的访问会是多个 I/O 流交叉在一起.为了在多个 I/O 流交织的情况下体现出各磁盘 I/O 流的空间局部性信息,我们定义了由 L 个逻辑块地址组成的数组,分别记录了 L 个最近访问过的结束逻辑块地址.当一个 I/O 请求到达时,我们首先在这 L 个结

束逻辑块地址中线性查找与该I/O请求起始地址最接近的逻辑地址.然后,计算该I/O请求与此最接近的逻辑地址之间的空间距离,并根据结果增加空间距离数组中对应元素的访问计数.最后,我们把找到的结束逻辑地址更新为此次I/O请求的结束逻辑地址.我们采用 L 的默认值为16.这样,本磁盘I/O空间局部性分析程序的时间复杂度为 $O(L)$.

当少于 L 条的顺序I/O流交织在一起时,上述方法可以有效地显示出其空间局部性特征.这样,本方法就可以适用于应用系统中存在多条磁盘I/O流的情况,并可以较好地展示磁盘I/O的空间局部性特征.

2.1.6 磁盘I/O的时间局部性信息

磁盘I/O的时间局部性也是存储系统性能优化的重要指标.数据缓存系统的实现就是依赖于数据I/O访问的时间局部性特性.数据缓存系统将刚访问过的I/O数据暂存在高速设备中,以期其在短期内会被多次访问,从而提高系统性能.数据缓存是各种存储设备的重要性能优化机制.因此,磁盘I/O访问的时间局部性特征信息将是系统管理员优化系统性能的重要依据.

经过简单分析我们看到,直接记录每个sector访问时间的方法在空间上几乎是无法接受的.比如:假定每次I/O的访问时间以32位来表示,则对于1TB数据,我们在最坏情况下就至少需要8GB的内存空间.为此,我们设计了一种节省内存空间的时间局部性特征获取方法,如图2所示.

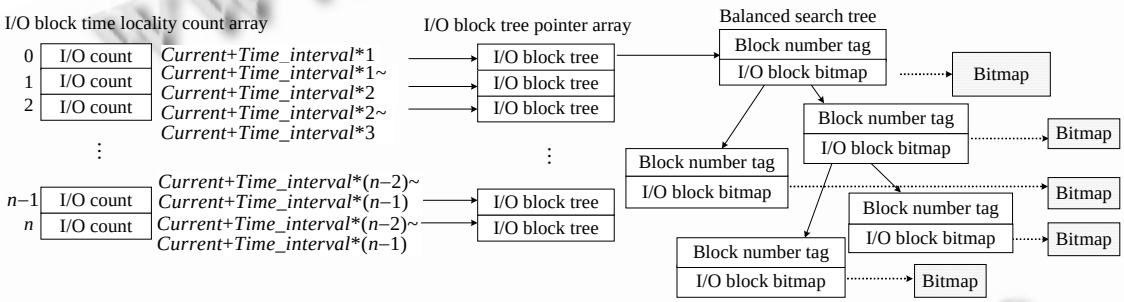


Fig.2 Main data structure of time locality analysis for disk I/O

图2 磁盘I/O时间局部性分析的数据结构示意图

正如图2所示,我们采用一个包含 n 个元素的I/O block tree指针数组来存储磁盘I/O的近期访问情况.为了有效地保留磁盘I/O访问信息,我们采用一个平衡二元查找树(I/O block tree)来记录一个大小为 *TIME_INTERVAL* 的时间段内的磁盘I/O访问信息.这样,我们就可以用 n 个I/O block tree保留最近 n 个 *TIME_INTERVAL* 时间段的磁盘I/O信息.我们以一个大小为 *Bitmap_size* 的I/O块位图(I/O block bitmap)来记录一个I/O块是否被访问.位图的某二进制位为1表示一个大小为 *Block_size* 个扇区的数据块被访问过,否则表示未被访问过.这样,一个位图就可以表示一个大小为 *Bitmap_sectors* 个扇区的连续对齐磁盘区域的访问情况,其中,*Bitmap_sectors* 的大小为 *Bitmap_size*×*Block_size* 个扇区.我们以该位图中所表示的数据块的最小扇区号作为该位图的标记(tag).在一个 *TIME_INTERVAL* 内的所有位图信息以TAG为关键字被组织成一个平衡二元查找树.如下面的算法1是磁盘I/O请求到达时,本磁盘I/O特征分析程序利用I/O block tree来收集磁盘I/O时间局部性信息的主要算法.其中值得说明的是,*Block_size*=1<<*Block_shift*,*Bitmap_size*=1<<*Bitmap_shift*,*Array_size* 表示I/O block tree数组的大小.另外,我们采用 d 表示磁盘I/O的时间距离,并用 *io_count* 来记录对应各种时间距离的I/O操作访问计数.

本I/O时间局部性分析程序在最坏情况下的时间复杂度为 $O(k\times n\times \log P)$,其中,$P$ 为整个磁盘空间内包含 *Bitmap_sectors* 的个数,n 为磁盘I/O请求访问数据块的数量,k 为I/O block tree指针数组的大小.

算法 1.

输入:I/O block tree的指针数组 *io_tree*,当前I/O block tree的索引值 *current*,

磁盘I/O请求访问的起始数据块 *uint*64_*t sector_number*,

数据块数量 *uint*64*_t sector_size*

Procedure *update_time_info*(*struct io_block_tree***io_tree*,*uint*32*_t current*,
*uint*64*_t sector_number*,*uint*64*_t sector_size*)

```
bit_in_bitmap=sector_number>>Block_shift;
//表示在 Bitmap 中磁盘 I/O 请求访问的起始块号
bits_size=sector_size>>Block_shift;
//表示在 Bitmap 中磁盘 I/O 请求访问的二进制位的数量

if (sector_number%Bitmap_sectors+sector_size>Bitmap_sectors)
    oversize=1;                    //表示该磁盘 I/O 请求访问的数据块跨 BITMAP 边界
else
    oversize=0;                    //表示该磁盘 I/O 请求访问的数据块在一个 BITMAP 内

found=find_bitmap(io_tree[current],sector_number);
//在 current 指向的 I/O block tree 中查找对应的 BITMAP
if (found==NULL){                  //没有找到
        为 current 查找树分配新的 BITMAP,由 found 保存;
   insert_bitmap(io_tree[current],found,sector_number);
   //把新分配的 BITMAP 插入到 current 查找树
}

if (oversize==1){                  //在 current 查找树中查找被访问的后一个 BITMAP
    found_next=find_bitmap(io_tree[current],sector_number+Bitmap_sectors);
    if (found_next==NULL){         //没有找到对应的 BITMAP
            为 current 查找树分配新的 BITMAP,由 found_next 保存;
       insert_bitmap(io_tree[current],found_next,sector_number+Bitmap_sectors);
       //把新分配的 BITMAP 插入到 current 查找树
    }
}

i=0;d=0;                           //初始化序列号 i 和时间距离 d
while (i<=bits_size){
    if ((oversize==1) && (bit_in_bitmap+i>Bitmap_size))
        found=found_next;

    //设置 bit_in_bitmap+i 数据块在 current 中的访问状态为 1,并返回原值
    origin=set_bit(found,(bit_in_bitmap+i)%Bitmap_size);

    if (origin==1){                //该数据块在 current 中被访问过
        continue;                  //继续下一个数据块
```

```
        else {
            next=current;
            //next 保留下一个需要查找的 I/O block tree
            do {
                next=(next+1)%Array_size;
                    在 next 指向的 I/O block tree 中查找对应于 bit_in_bitmap+i 的二进制位;
                    若该二进制位的值为 1,则 get=1,否则 get=0;
            } while ((get==0) && (next!=current));
            if (next==current){
                //表示在所有保留的 I/O block tree 中都没有该数据块被访问过的信息
                d=Array_size;
                break;              //直接设置 d 为最大值
            }
        }

        tmp=distance(current,next,Array_size);
        //计算找到数据块的 I/O block tree 和 current 之间的距离
        d=d>tmp?d:tmp;              //记录并保存最大的时间距离
    }
    io_count[d]++;                  //记录此次磁盘 I/O 的时间局部信息
End
```

此外,我们还设置了以 *TIME_INTERVAL* 为时间周期的定时处理函数.我们把I/O block tree指针数组看作一个循环数组来保存I/O信息.定时处理函数首先将 *current* 索引值减1,然后,清空 *current* 指向的I/O block tree,用来保存未来一个 *TIME_INTERVAL* 内的磁盘I/O访问信息.

在磁盘I/O的时间局部性信息采集中,我们没有区分磁盘I/O的读写操作.这主要是因为磁盘I/O读写操作之间往往也会出现时间局部性特征,比如,某个应用刚刚写过的数据块往往有可能在短期内被读取.

2.1.7 活跃磁盘I/O

我们把应用程序已经提交但没有完成的I/O请求称为活跃磁盘I/O.活跃磁盘I/O的数量在一定程度上反映了应用程序的并发度.例如:当应用程序采用异步I/O方式或者多线程的编程模型实现时,系统负载就可能同时产生较多的活跃I/O.为了有效记录活跃I/O数量信息,我们分别针对读操作和写操作设置了 *active_queue* 数组,并用其来记录系统内出现的活跃I/O数量.

2.1.8 磁盘I/O的热点探测

我们把磁盘看作一个由扇区组成的连续数组.然而,实际应用系统的磁盘I/O操作数并不会平均分布在整个扇区数组上.那么,探测和发现磁盘I/O热点可能为系统管理员或者应用系统开发人员的存储系统优化工作提供有效帮助.

由于系统磁盘空间往往会大于数百GB,若我们以单个扇区为单位记录磁盘I/O操作热点,就很有可能造成不必要的内存空间浪费.为了有效探测磁盘I/O的热点区域并节约磁盘I/O特征分析程序的内存空间,我们把整个磁盘空间映射到包含 *HOTSPOT_LEVEL* 个元素的整数数组.每个元素负责记录落在大小为 *HOTSPOT_UNIT* 个连续扇区内的I/O访问次数.用户可以根据需要设置 *HOTSPOT_LEVEL* 或 *HOTSPOT_UNIT* 的大小.我们采用的默认设置是 *HOTSPOT_LEVEL* 为1 024或者 *HOTSPOT_UNIT* 为8 192.这样,一般情况下,磁盘I/O热点分析程序运行时的时间复杂度为 $O(1)$.

2.2 磁盘I/O特征采集程序的运行

本 I/O 特征分析程序可以在管理员的指令下在任意时刻启动或停止.而且,在统计磁盘 I/O 计数、磁盘 I/O 块大小、磁盘 I/O 时间间隔、磁盘 I/O 活跃数量、I/O 延迟时间等信息时,本程序的时间复杂度为 $O(1)$.同时,在分析磁盘 I/O 空间/时间局部性信息以及 I/O 操作热点分布时,本程序的时间复杂度也较低.而且,本程序占有内存空间也较小.因此,本程序可以适合于在线磁盘 I/O 特征的分析.

本程序主要由磁盘 I/O 请求到达、磁盘 I/O 请求完成、监控停止时以及周期性运行的 4 个处理函数组成.在磁盘 I/O 请求到达时,我们首先判断该请求的读写特性,然后分别进行相关的信息统计.其中包括磁盘 I/O 计数、磁盘 I/O 块大小、磁盘 I/O 空间局部信息、磁盘 I/O 时间间隔、磁盘 I/O 活跃数量以及磁盘 I/O 的起始时间等信息.在磁盘 I/O 请求完成时,我们主要统计磁盘 I/O 延迟时间以及更新活跃磁盘 I/O 数量等信息.在磁盘 I/O 监控停止时,我们会输出全部磁盘 I/O 特征信息,并清空相关统计信息的数据结构.此外,我们还建立了周期性的处理函数来定期地更新磁盘 I/O 的时间局部性信息以及输出磁盘 I/O 特征信息.

对于特殊的磁盘 I/O 特征分析需求,本文的 I/O 特征分析程序也能够同时生成详细的磁盘 I/O 指令的 Trace.借助该磁盘 I/O 指令 Trace,系统管理员可以进行更加深入的磁盘 I/O 特征分析.

3 测试结果与评价

我们首先通过 IOMeter 测试和评价了本文磁盘 I/O 特征分析程序的运行效率及其对应用系统的 I/O 性能的影响.随后,以大文件拷贝、Filebench 测试程序作为应用负载,我们展示了应用本文磁盘 I/O 特征分析方法得到的测试结果,并且进行了相应的分析和讨论.

由于本文的磁盘 I/O 特征分析方法和测试都是基于虚拟机系统进行的,因此,我们将对虚拟机系统进行简要的说明.与非虚拟机系统不同,虚拟机系统具有一层介于物理平台和客户操作系统之间的虚拟机核心软件(virtual machine monitor,简称 VMM).虚拟机软件负责管理对物理平台的访问,以使得物理平台资源能够被多个客户操作系统所共享使用.目前,虚拟机软件在企业服务器整合、软件调试和测试、模拟传统平台、系统容灾等方面的应用取得了重要发展.虚拟机系统的主要优点包括:

- 资源效率.资源效率通常是选用虚拟机技术的常见原因之一.充分利用空闲资源正是虚拟机技术的主要应用目标之一.它可以使我们在更少的物理资源上完成相同的任务,减少能耗和资源浪费现象的发生.
- 计算系统的易配置.虚拟计算系统的配置管理可以变得十分简化,几乎一切都可以通过鼠标点击完成.比如,我们可以轻易地通过管理接口给虚拟机系统增、减逻辑资源.虚拟机系统的逻辑资源也可以通过系统模板进行大批量的部署和管理.
- 快照.快照是逻辑磁盘的虚拟复制.我们可以通过快照使系统快速回到创建快照时的即时状态.这样,我们可以在必要的时候方便地让虚拟机系统回到以往的运行状态下.
- 远程复制.在高可用系统和远程容灾系统中,经常需要用到远程复制技术.这要求我们在不影响生产系统正常运行的情况下,将生产系统的信息完整复制到远程的站点.虚拟存储和虚拟机技术可以有效地保证我们在不影响生产系统运行的情况下,方便地实现远程复制.
- 资源使用的统计报告.通过虚拟机软件,我们可以准确而详细地获取用户操作系统的资源访问情况.任何关于 CPU、内存、磁盘和网络的访问信息,我们都能够以独立于用户操作系统的方式获取.

与此同时,虚拟机系统仍有一些不足之处:

- 性能损失.由于虚拟机系统进行了物理资源的隔离和虚拟化处理,这样就不可避免地带来虚拟化的性能开销.因此,对于某些特别关注性能的应用,目前可能并不适合运行在虚拟机系统之上.
- 管理和维护的复杂性.管理和维护的复杂性通常是虚拟机技术应用所面临的另外一个问题.大量的逻辑资源聚合在少量的物理资源之上,就不可避免地增加了系统的复杂性和宕机的风险.同时,由于虚拟机软件的存在,使得我们在系统的部署方面增加了一层复杂性.

- 硬件故障风险.由于多个用户系统运行在同一个物理平台上,因此,整个物理平台的工作负载显著增加.这样就会增加物理平台出现故障的风险.

值得注意的是,正如本文第1节所述,虚拟机系统的磁盘I/O和物理平台上的磁盘I/O的主要差别在于由虚拟化操作引入的性能损失.我们可以认为虚拟机系统的磁盘I/O行为和物理平台上的磁盘I/O行为基本一致,因此,采用本文的磁盘I/O特征分析方法取得的结果,如磁盘I/O计数、读写分布、I/O空间局部性、时间局部性以及磁盘I/O操作热点分布等I/O特征信息,都可以较为真实地反映实际系统行为,并对系统性能优化有着重要的参考意义.

3.1 测试环境

本文采用的性能测试硬件平台是Intel 965P平台,Core2 1.86GHz CPU,1G DDR2 667MHz内存,希捷120G硬盘和Marvell 88E8053网卡.软件系统采用Xen 3.0.5虚拟机软件,服务域采用Fedora Core 5操作系统,用户域采用Fedora Core 4操作系统或Windows XP Professional操作系统.

3.2 性能结果与评价

3.2.1 磁盘I/O特征分析程序的性能测试与分析

基于IOMeter(2006.10.31版),本文针对Xen 3.0.5虚拟机下的Fedora Core 4客户操作系统进行了磁盘I/O特征分析程序的性能测试和比较.在本次测试中,我们通过IOMeter标准测试程序,生成了磁盘顺序读、磁盘顺序写、读写各50%的磁盘随机读写3种不同类型的I/O工作负载.IOMeter测试程序通过其运行在客户操作系统中的dynamometer测试引擎来生成I/O工作负载,同时测量和记录磁盘I/O操作的性能.其测试结果包括每秒钟磁盘I/O操作数、磁盘I/O带宽以及I/O操作的平均延迟时间.其中,无特征分析程序的测试是我们在相同的物理平台上直接运行标准的Xen 3.0.5虚拟机软件,即没有插入我们的I/O特征分析程序.如图3所示,我们分别对比测试了在有无I/O特征分析程序的两种条件下,客户操作系统磁盘I/O整体性能的损失情况.

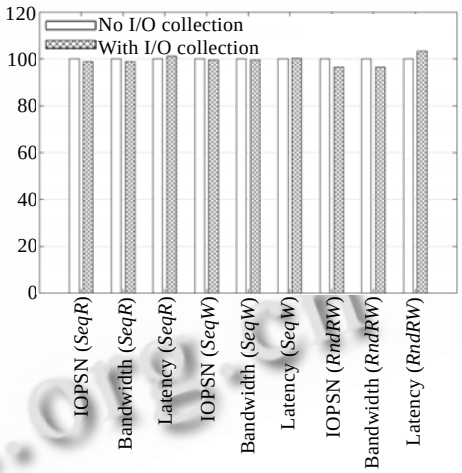


Fig.3 Performance comparison of disk I/O characteristics

图3 磁盘I/O特征分析程序的性能比较

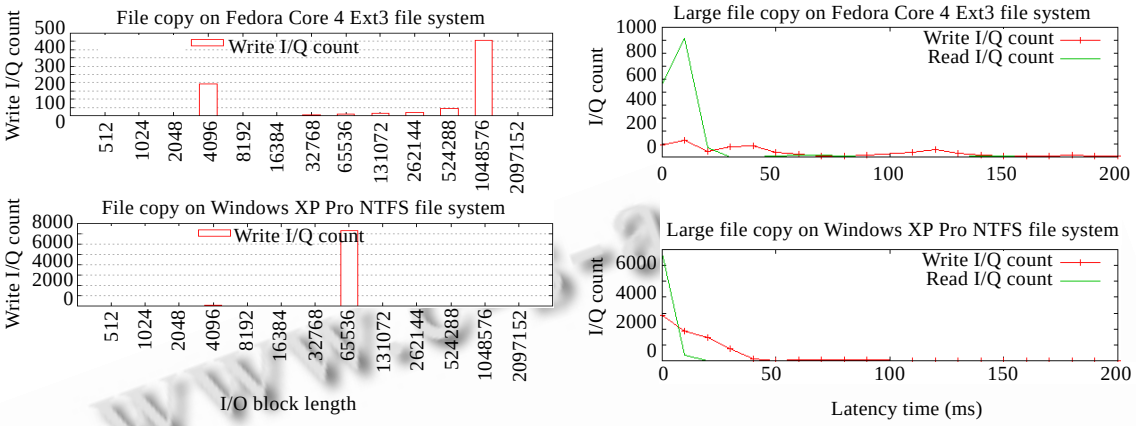
在图3中,我们假定在无磁盘I/O特征分析程序的情况下,系统磁盘I/O的性能为100.这样我们可以看到,在各种磁盘I/O操作负载下,开启在线I/O特征分析程序对应用系统的磁盘读写性能的影响至多为4%.经测试,在运行单个磁盘I/O特征分析程序实例时,Xen的服务域CPU占用率增加最多3%,内存空间增加少于8MB.由此我们可以看出,本I/O特征分析程序具有较小的系统开销,对应用系统磁盘I/O性能的影响也较小.因此,本I/O特征分析程序可以适用于在线磁盘I/O特征信息的采集和分析.

3.2.2 大文件拷贝的磁盘I/O特征分析

相关研究工作表明,文件系统的大部分磁盘空间通常被大文件所占用[3].这时,大文件拷贝操作的性能将可能对系统的整体性能产生较大影响.因此,我们采用本文的磁盘I/O特征分析方法对不同系统下的大文件拷贝操作进行了对比测试.我们分别在Fedora Core 4和Windows XP Pro下,对470MB的大文件拷贝操作进行了对比测试.在Fedora Core 4下,我们采用Ext3文件系统,文件块大小为4KB.在Window XP Pro下,我们采用NTFS文件系统,文件块大小也为4KB.具体的磁盘I/O特征分析结果如图4所示.

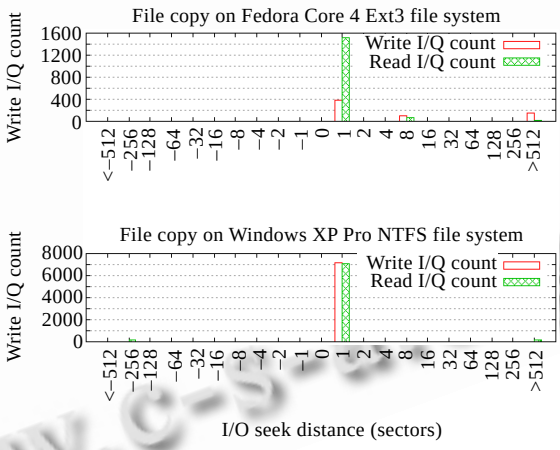
由图4我们可以看出,在相同的工作负载情况下,两个操作系统的磁盘I/O行为有着很多差异.Windows XP Pro系统的最大磁盘写操作块为64KB,而Fedora Core 4系统的最大磁盘写操作块大小为1MB.而且,两者的绝大多数写操作都是发送最大块的写请求.正是由于两者写操作块大小的这种差别,Windows XP Pro中的写操作

数量比 Fedora Core 4 中的写操作数量多出一个数量级.即:Windows XP Pro 的磁盘写操作数为 7 420,而 Fedora Core 4 的磁盘写操作数仅为 745.基于同样原因,在图 4(b)中我们可以看到,Windows XP Pro 的 I/O 延迟时间要普遍少于 Fedora Core 4 系统.同时,在图 4(c)中我们可以注意到,两者相同之处是大部分文件系统写操作的磁盘空间分配都是连续的.



(a) Distribution of disk I/O block size
(a) 磁盘 I/O 块大小的分布情况

(b) Distribution of disk I/O latency
(b) 磁盘 I/O 延迟时间的分布情况



(c) Distribution of disk I/O seek distance
(c) 磁盘 I/O 的空间寻址距离的分布情况

Fig.4 I/O characteristics of large file copy workload
图 4 大文件拷贝的 I/O 特征分析结果

3.2.3 Filebench 的磁盘 I/O 特征分析

Filebench 是由 SUN 公司开发的开源文件系统性能测试程序.根据描述应用程序进程和线程的工作负载模型文件,Filebench 可以方便地模拟出多种复杂应用程序的文件系统访问行为.Filebench 的开发者对多种应用程序的工作 Trace 进行了分析,并建立了相应的工作负载模型.每个工作负载模型包括一个或多个文件 I/O 工作流,以及多个工作流之间的同步关系.文件 I/O 工作流的描述包括 THINK 时间和文件的创建、读写、删除、追加等操作,以及每个操作的 I/O 尺寸、操作比率、I/O 模式等描述信息.我们只需指定所需的工作负载,Filebench 就可以模拟出相应应用程序的文件操作行为.

我们采用 Filebench 1.1.0 程序中自带的应用程序工作负载模型进行了应用系统模拟,其中,filemicro 工作负

载模型定义了一系列典型文件操作,比如创建大规模目录树及文件、复制大规模目录树及文件、单线程顺序读写大文件、多线程随机读写大文件、多线程顺序读写大文件等等.另外,Varmail 工作负载模型模拟了在 NFS 文件服务器上电子邮件服务的文件读写行为,其中包括多线程的文件读取、追加以及目录删除等操作.在 Fedora Core 4 客户操作系统下,我们采用 varmail 和 filemicro 工作模型,分别对大小为 4GB 的 Ext3 和 Ext2 文件系统进行了对比测试,并试图观察文件系统的磁盘 I/O 行为.其磁盘写操作分布情况如图 5 所示.

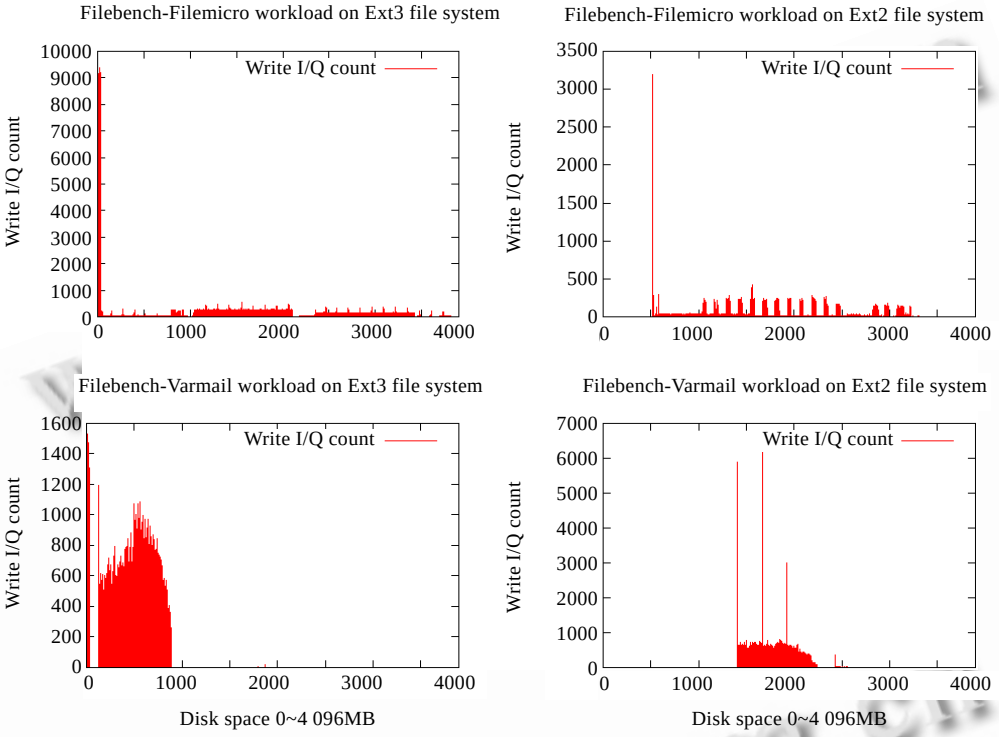


Fig.5 Distribution of disk write operations

图 5 磁盘写操作分布情况

根据图 5 我们可以发现,在 Ext3 文件系统下的磁盘写操作热点在磁盘分区的前 36MB 区域,然而在 Ext2 文件系统中却没有此现象.由于两个文件系统的主要区别在于维护文件系统一致性的日志机制,也就是说,Ext3 文件系统增加了维护文件系统元数据一致性的 Journal 日志文件机制,因此我们可以猜想在 Ext3 文件系统中的 Journal 日志文件位于磁盘分区的前 36MB 区域.经过对 Linux 2.6 内核中 Ext3 文件系统的源代码以及 mkfs 1.37 源代码的分析我们发现,Ext3 文件系统的 Journal 日志文件的大小为 8 192 个文件块,即 32MB,并且该文件的起始地址小于 4MB,由此我们可以看出,在 Ext3 文件系统上,Journal 日志文件所在区域确实是在磁盘分区的前 36MB 区域.

为了进一步分析和判断日志文件的写操作频繁程度,我们利用 I/O 操作热点分析程序进行了测试.经过细化 I/O 操作热点分布程序的探测粒度,我们得到的测试结果是:在 Filemicro 工作负载下,日志文件写命中 5 099 次,占写操作总数(15 204 次)的 33.54%;在 varmail 工作负载下,日志文件的写操作命中 5 172 次,占写操作总数(44 747 次)的 11.56%.

由此可以看出,在典型工作负载下,Ext3 文件系统的日志文件区域的写操作频率明显高于其他磁盘区域.因此,提高文件系统中日志文件区域的写操作性能将会有助于提高文件系统的整体性能.以往关于文件系统日志的研究结果同时也印证了上述的分析结论[4].

与此同时,在典型工作负载下,磁盘 I/O 操作的时间局部性特征也是我们所关心的重要的磁盘 I/O 特征信息.

为此,我们在 Ext3 文件系统上分别采用 filemicro 和 varmail 工作负载进行了相关测试.其中,时间间隔 *TIME_INTERVAL* 采用的是 0.2s.通过测试我们知道,系统在 filemicro 工作负载下的总读写 I/O 操作数为 322 521,其中,在 3.2s 内重复访问数据块的 I/O 操作数为 100 040,占总 I/O 数量的 31%.系统在 varmail 工作负载下,总读写 I/O 操作数为 191 148,其中在 3.2s 内重复访问数据块的 I/O 操作数为 75 645,占总 I/O 数量的 39.6%.如图 6 所示,在 filemicro 工作负载下,系统重复访问数据块的 I/O 数量逐渐递减,并且在 0.4s 内重复访问数据块的 I/O 数量占 91.7%.同样,在 varmail 工作负载下也呈现 I/O 数量逐渐递减的趋势,并在 0.6s 内重复访问数据块的 I/O 数量占 85.3%.这些结果都恰好印证了数据 I/O 访问的时间局部性原理.同时,这些在线获取的时间局部性特征信息对磁盘数据缓存的设置有着实际参考意义.

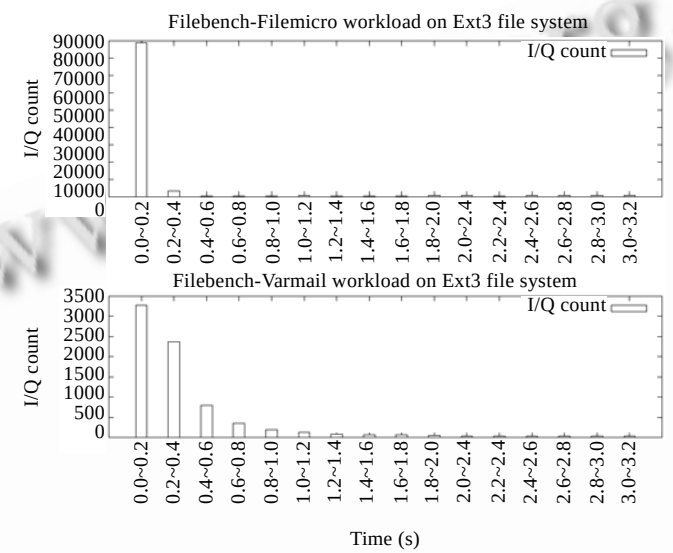


Fig.6 Time locality analysis results of disk I/O operations

图 6 磁盘 I/O 操作时间局部性的分析结果

4 相关研究工作

虚拟机技术一直是计算机系统研究的热点.与非虚拟机系统不同的是,虚拟机系统一般具有一层介于物理平台和用户操作系统之间的虚拟机软件.虚拟机软件负责管理对物理平台的访问,以使得物理平台能够被多个用户虚拟系统所共享使用.在虚拟机软件中,我们可以方便地监控应用系统的磁盘 I/O 行为,而且不必依赖于某种特定的操作系统.Xen 是最初由 Cambridge 大学 Computer Laboratory 发起的开源虚拟机项目.它的开发得到了 Intel,HP,IBM 等公司的支持.Xen 是支持同时运行多个虚拟系统的高性能 VMM[5].它支持 x86_32,x86_64,IA64 等多种平台,并支持 Intel VT[6]和 AMD SVM(secure virtual machine)技术.目前,以 Xen 为代表的高性能虚拟机软件,如 VMware ESX Server[7],Microsoft Virtual Serve 等,越来越受到业界的关注,并逐步应用到企业生产环境中.同时,国内关于虚拟机技术和应用的研究也十分活跃[8].虚拟化的开销一直是虚拟机技术应用的关键.目前,磁盘 I/O 的虚拟化开销是虚拟机整体虚拟化开销的主要来源之一[9].因此,研究和分析虚拟机系统的磁盘 I/O 特征,将会给优化虚拟机系统的整体性能提供重要帮助.

近些年来,关于存储 I/O 特性的研究一直是计算机系统研究的热点问题.然而,大部分研究都是基于 I/O 操作 Trace 来进行工作负载的分析.基于 I/O 操作 Trace 的分析工作往往是在线采集原始数据,然后以离线的方式来分析 I/O 访问特征信息.然而,基于虚拟机的 I/O 特性分析方法能够以在线的方式进行应用系统的磁盘 I/O 特性分析.另外,在许多存储系统的有关工作中,如利用 IOMeter 来模拟应用系统工作负载,我们需要的往往是工作负载的各种磁盘 I/O 特征,而不是具体的 I/O 操作 Trace.

近来,关于在工作负载在线分析的研究工作成为 I/O 特征研究领域的重要组成部分.比如,IBM 的 Moilanen

在Linux系统中,在线跟踪平均I/O块大小、读写比例以及平均寻址距离等信息[10],然后根据跟踪结果利用遗传算法生成当前工作负载的fingerprinting,并用以自动调节文件系统的参数.Falk提交的Linux内核块设备补丁程序实现了在线统计I/O块大小、I/O完成时间等功能[11].Pablo物理I/O工具通过在Linux内核中加入补丁程序来获取I/O请求块大小、I/O请求延迟等信息,然后生成相应的直方图[12].在IBM的System Z中,Linux资源分析工具能够在线跟踪I/O块大小、I/O延迟以及I/O队列中I/O数量[13].与本文相比,我们提供了更加丰富的I/O统计信息.更为重要的是,本文基于虚拟机的在线I/O特征分析方法独立于用户操作系统,可以方便地应用于多种系统环境.VMware公司的Ahmad以直方图的方式在VMware ESX Server上研究了在线磁盘I/O特征分析方法[14].与之相比,我们分别在Xen虚拟机系统的IDE和SCSI接口中实现了I/O特征分析程序.而且,本文的I/O特征分析程序提供了更多、更重要的I/O特征统计信息,如磁盘I/O的时间局部性信息以及磁盘I/O操作的热点分布.这些信息是进一步研究磁盘数据缓存等优化技术的重要依据.

此外,磁盘性能评测程序也可以获取一部分磁盘I/O的特征信息.比如,IOMeter通过在用户操作系统上生成块级别I/O操作的工作负载来测量和记录磁盘系统的性能.另外,IOzone是在文件系统级别的存储性能测试工具.此外,还有Bonnie++标准测试程序包含一系列的文件级别或块级别的测试用例,并用以比较不同存储系统的性能.类似的磁盘性能评测程序还有很多,它们都是通过在用户操作系统上生成相关的工作负载来测量磁盘系统的性能指标,如磁盘带宽、每秒I/O数量、平均延迟时间等.这些磁盘性能评测程序的目的在于测试和比较磁盘系统性能,因此其工作负载往往都是自身生成的标准工作负载.它们并不能用于获取各种应用负载的磁盘I/O特征信息.

值得一提的是,许多操作系统都会带有相关的磁盘性能监控程序,比如Linux系统中的iostat命令,它可以有效监控I/O设备的运行状况和系统时间,从而获取磁盘设备的每秒读写数据量.Windows系统中的FileMon可以实时地监控文件系统行为,如每次文件的创建、读写、删除等操作.DiskMon可以有效监控Windows系统中的磁盘I/O信息,如磁盘I/O起始位置和长度等信息.在虚拟机系统中运行此类程序也可以获取和本文类似的用户系统磁盘I/O信息.但是,这些程序都针对某一种操作系统而开发,不能广泛适用于各种操作系统.而且,由于这些评测程序需要运行在用户操作系统之中,它们势必会占用用户系统的计算资源和存储资源.

5 结束语

磁盘I/O性能一直以来都是计算机系统的性能瓶颈.随着电子技术的不断发展,CPU和内存的性能迅速提高,其与磁盘性能之间的性能差距逐渐加大.因此,存储系统性能分析和优化技术一直是计算机系统研究的重点.本文介绍的基于Xen虚拟机的在线磁盘I/O特征分析方法可以有效地统计磁盘I/O读写比例、I/O块大小、I/O时间间隔、活跃I/O数量、I/O延迟、I/O空间/时间局部性信息以及磁盘I/O操作热点分布等多种I/O特征量.通过测试我们看到,本文的I/O特征分析方法具有较小的系统开销和应用系统性能影响,适于进行在线I/O特征分析.随后,本文还给出了面向典型应用负载的磁盘I/O特征测试结果和分析评价.

我们可以看到,基于虚拟机的在线磁盘I/O分析机制是一种灵活而高效的磁盘I/O特征分析工具.其在线分析的特点能够有效地帮助存储系统动态调节性能参数.在我们正在进行的基于磁盘的网络数据缓存机制研究中,我们将基于本磁盘I/O特征分析方法对数据缓存的尺寸、数据替换策略、缓存块大小等参数做出优化调整.此外,在企业计算环境中多数虚拟机系统都采用网络存储设备进行数据存储.因此,我们将采用基于虚拟机的在线I/O分析方法来研究多台虚拟机系统的共同I/O行为特征.此研究将对优化网络存储系统性能有着重要帮助.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是中国科学院计算技术研究所存储分中心的同学和老师表示感谢.

References:

[1] Roselli D, Anderson TE. Characteristics of file system workloads. Technical Report, CSD-98-1029, Berkeley: University of California, 1998.

[2] Liu J, Yang XJ, Tang YH, Wang YX. The optimal strip unit size of heterogeneous disk array. Chinese Journal of Computers, 2004,27(6):819−827 (in Chinese with English abstract). http://cjc.ict.ac.cn/qwjs/view.asp?id=1612

[3] Agrawal N, Bolosky WJ, Douceur JR, Lorch JR. A five-year study of file-system metadata. In: Proc. of the 5th Conf. on File and Storage Technologies (FAST 2007). San Jose: USENIX, 2007. 31−45. http://www.usenix.org/event/fast07/

[4] Prabhakaran V, Arpaci-Dusseau AC, Arpaci-Dusseau RH. Analysis and evolution of journaling file systems. In: Proc. of the USENIX 2005 Annual Technical Conf. Anaheim: USENIX, 2005. 105−120. http://www.usenix.org/publications/library/proceedings/usenix05/

[5] Pratt I, Fraser K, Hand S, Limpach C, Warfield A. Xen 3.0 and the art of virtualization. In: Proc. of the Linux Symp., Vol.2. Ottawa, 2005. 65−77. http://www.linuxsymposium.org/2005/

[6] Uhlig R, Neiger G, Rodgers D, Santoni AL, Martins FCM, Anderson AV, Bennett SM, Kagi A, Leung FH, Smith L. Intel virtualization technology. IEEE Computer, 2005,38(5):48−56.

[7] Waldspurger CA. Memory resource management in VMware ESX server. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation (OSDI 2002). Boston: USENIX, 2002. 181−194. http://www.usenix.org/events/osdi02/

[8] Huai JP, Li Q, Hu CM. Research and design on hypervisor based virtual computing environment. Journal of Software, 2007,18(8): 2016−2026 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2016.htm

[9] Clark B, Deshane T, Dow E, Evanchik S, Finlayson M, Herne J, Matthews JN. Xen and the art of repeated research. In: Proc. of the USENIX 2004 Annual Technical Conf.: Free and Open Source Software (FREENIX) Track. Boston: USENIX, 2004. 135−144. http://www.usenix.org/events/usenix04/cfp/freenix.html

[10] Moilanen J. I/O workload fingerprinting in the genetic-library. In: Proc. of the Linux Symp., Vol.2. Ottawa, 2006. 165−172. http://www.linuxsymposium.org/2006/linuxsymposium_procv2.pdf

[11] Falk E. Introduce block I/O performance histograms. 2006. http://lwn.net/Articles/209770/

[12] Reed DA, Aydt RA, Madhyastha TM, Noe RJ, Shields KA, Schwartz BW. An overview of the Pablo performance analysis environment. 1992. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.2102

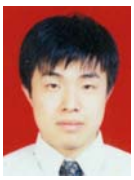
[13] Parziale L, Belardi M, Held M, Ma LY, Peckover L, Reed K. Linux on IBM system z: Performance measurement and Tuning. 2008. http://www.redbooks.ibm.com/redbooks/pdfs/sg246926.pdf

[14] Ahmad I. Easy and efficient disk I/O workload characterization in VMware ESX server. In: Proc. of the 2007 IEEE Int'l Symp. on Workload Characterization (IISWC 2007). Boston: IEEE, 2007. 149−158. http://iiswc.org/iiswc2007/

附中文参考文献:

[2] 刘军,杨学军,唐玉华,王勇献.异构盘阵中最优 Strip Unit Size 选择技术.计算机学报,2004,27(6):819−827. http://cjc.ict.ac.cn/qwjs/view.asp?id=1612

[8] 怀进鹏,李沁,胡春明.基于虚拟机的虚拟计算环境研究与设计.软件学报,2007,18(8):2016−2026. http://www.jos.org.cn/1000-9825/18/2016.htm



沈玉良(1977－),男,黑龙江哈尔滨人,博士生,主要研究领域为虚拟机系统,网络存储.

许鲁(1962－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为计算机网络存储系统.