基于优化的 COW 虚拟块设备的虚拟机按需部署机制

陈 彬 肖 侬 蔡志平 王志英
(国防科学技术大学计算机学院 长沙 410073)

摘 要 基于 COW(Copy-on-Write)读写模式的虚拟块设备有利于实现大规模虚拟机环境下虚拟机的快速部署.文中为虚拟机管理器中的 COW 虚拟块设备设计了一种优化方法,能够提高 COW 磁盘的访问性能以及生成多个小尺寸的 COW 磁盘映像文件,以降低通过网络部署虚拟机的开销.基于优化的 COW 虚拟块设备,文中提出了虚拟机环境下的虚拟机按需部署机制及关键技术问题的解决方案.基于 Linux 平台和 QEMU 虚拟机,实现了基于优化 COW 虚拟块设备的虚拟机按需部署原型系统.实验表明,优化的 COW 虚拟块设备、基于 COW 磁盘有效工作集的优化部署以及 COW 磁盘回收等方法能够有效地支持虚拟机环境下低开销的、按需的虚拟机部署.

关键词 虚拟机;虚拟机管理器;虚拟机按需部署;COW 磁盘;有效工作集
中图法分类号 TP303 **DOI 号**: 10.3724/SP.J.1016.2009.01915

On-Demand Deployment of Virtual Machines Based on Optimized COW Virtual Block Device

CHEN Bin XIAO Nong CAI Zhi-Ping WANG Zhi-Ying
(*School of Computer, National University of Defense Technology, Changsha* 410073)

Abstract The COW (Copy-on-Write) virtual block device (VBD) is very helpful for fast virtual machine (VM) deployment in large-scale virtual machine environments. This paper presents an optimization approach for the COW VBD in virtual machine monitor (VMM), which can improve the access performance of COW disks and can make many small-sized COW disk images to reduce the cost of VM deployment over the network. Based on the optimized COW VBD, the paper also presents an on-demand VM deployment scheme and the solutions to the key problems of the scheme. Based on Linux system, the authors have implemented the optimized COW VBD in QEMU virtual machine monitor and an on-demand VM deployment prototype system. Experiments show that the optimized COW VBD, the effective working set-based deployment optimization and the COW disk reclamation approach, etc can effectively support low-cost, on-demand VM deployment in virtual machine environments.

Keywords virtual machine; virtual machine monitor; on-demand virtual machine deployment; COW disk; effective working set

1 引 言

近年来,系统虚拟机(system virtual machine)[1]因其在解决异构性、可移动性和系统管理等问题方面具有的优势,成为研究和应用的热点[2-5],而基于虚拟机实现软件部署已成为快速构建大规模虚拟计算环境的一种较好的解决方案.通过虚拟机管理器

收稿日期:2009-07-15;最终修改稿收到日期:2009-08-27.本课题得到国家"九七三"重点基础研究发展规划项目基金(2007CB310900)、国家自然科学基金重点项目(60736013)、国家自然科学基金项目(60603062)和湖南省自然科学基金项目(06JJ3035)资助.**陈 彬**,男,1975 年生,博士研究生,主要研究方向为虚拟化技术、分布式计算.E-mail: chenbin@nudt.edu.cn. **肖 侬**,男,1969 年生,教授,博士生导师,主要研究领域为网格计算、网络存储、体系结构和虚拟化技术. **蔡志平**,男,1975 年生,博士,副教授,主要研究方向为虚拟化技术和网络安全. **王志英**,男,1956 年生,教授,博士生导师,主要研究领域为高性能计算机体系结构、虚拟化技术.

VMM(Virtual Machine Monitor),虚拟机的存储设备的整个状态能够被封装到宿主机中的虚拟磁盘映像文件中.虚拟磁盘映像文件可以被传输到其它机器上,然后通过 VMM 可以快速启动或恢复相同的虚拟运行环境,省去了用户重新安装/配置软件的环节.通过 VMM 和分发易于拷贝/移动的虚拟磁盘映象文件来替代传统的软件安装/配置过程,能够大大加速软件部署进程,降低软件部署开销,对于大规模计算环境的快速构建、失效恢复等具有重要的意义.然而,传统的将安装有所有软件的单个虚拟磁盘映像分发给每个用户的部署方法存在如下几个方面的局限性:

(1)磁盘空间利用.不同用户有不同的软件需求,给每个用户分发所有软件会导致磁盘空间的浪费和无谓的传输开销;

(2)传输开销.一个安装有操作系统和各种应用软件的虚拟磁盘映像大小通常在几个 GB(甚至几十个 GB)以上,将这种大尺寸的映像文件通过网络分发给每个用户,会导致较高的传输开销;

(3)新增软件部署.如果安装新软件或进行软件更新,需要重新分发更新后的整个虚拟磁盘映像文件.

为了避免分发单个大尺寸的虚拟磁盘映像带来的高开销,可以利用基于 COW(Copy-on-Write)机制[2]的块共享技术将单个虚拟磁盘映像映射成多个虚拟磁盘映像(简称 COW 磁盘),然后通过按需分发 COW 磁盘来降低传输开销.在 COW 模式下,最初生成的 COW 磁盘称为根 COW 磁盘,通常用于安装操作系统和最常用的软件.基于根 COW 磁盘可以先后生成多个子/孙 COW 磁盘,每个 COW 磁盘安装有一种或多种软件.在安装软件生成每个 COW 磁盘的过程中,先前生成的所有祖辈 COW 磁盘都处于只读状态,对虚拟磁盘的所有写操作将针对当前 COW 磁盘进行.COW 磁盘之间的父子关系或生成顺序记录在每个 COW 磁盘的文件头中,可以用图 1(a)所示的多层次单分枝的树形结构来表示.COW 磁盘的这种层次结构对虚拟机中的客户操作系统(Guest OS)是透明的,客户操作系统看到的始终是一整块磁盘.

通过按需地向用户分发较小尺寸的 COW 磁盘,可以大幅降低通过网络部署虚拟机的开销.然而,较小尺寸的 COW 磁盘会导致较深的 COW 磁盘层次结构.另一方面,由于传统的基于回溯追踪(backward-tracing)[2,6-7]的 COW 磁盘访问方法的开销随着 COW 磁盘层次深度的增加而线性增长[3],在 COW 磁盘层次较深的情况下,会导致较低的 COW 磁盘访问性能,因此 COW 磁盘层次应该尽可能少,然而这又会导致生成大尺寸的 COW 磁盘,如图 1(b)所示.对于使用软件种类较多的计算环境,传统的 COW 磁盘访问机制使得难以在 COW 磁盘大小和 COW 磁盘层次深度之间做出合适的折衷.

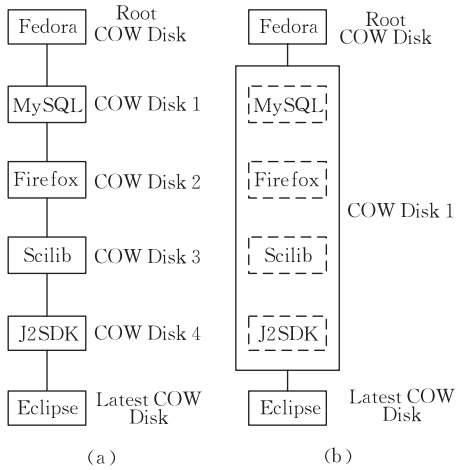


图 1 COW 磁盘层次结构

本文为 VMM 中基于 COW 读写模式的虚拟块设备(简称 COW 虚拟块设备)设计了一种优化的块定位方法,能够提高 COW 磁盘的访问性能,并且使得 COW 磁盘的访问性能不再受 COW 磁盘层次深度的影响.基于优化的 COW 虚拟块设备,本文提出在单个软件的基本粒度上生成多个小尺寸的 COW 磁盘,然后按需地分发到用户端,以降低网络环境下虚拟机部署的开销.为解决大型软件部署、软件的持久更新等关键问题,本文设计了基于 COW 磁盘有效工作集的优化部署、COW 磁盘回收等关键技术.基于 Linux 平台和 QEMU[6]虚拟机,实现了优化的 COW 虚拟块设备和虚拟机按需部署的原型系统,并通过实验验证了优化方法和各项关键技术的有效性.本文第 2 节详细介绍 COW 虚拟块设备的优化;第 3 节介绍基于优化的 COW 虚拟块设备的虚拟机按需部署机制;第 4 节介绍原型系统的实现和实验;第 5 节介绍相关工作;最后第 6 节总结全文.

2 COW 虚拟块设备的优化

由于虚拟磁盘在 COW 读写模式下被映射成多个 COW 磁盘,虚拟磁盘的磁盘块将分布到多个 COW 磁盘中去,如图 2 所示,因此 VMM 中的 COW 虚拟块设备驱动必须提供在 COW 磁盘层次中定位磁盘块所在 COW 磁盘的方法.本节分析了

传统的回溯追踪块定位方法,提出了优化的基于全局位图文件 gbf 的块定位方法,并对两种方法的时间和内存开销进行了分析和比较.

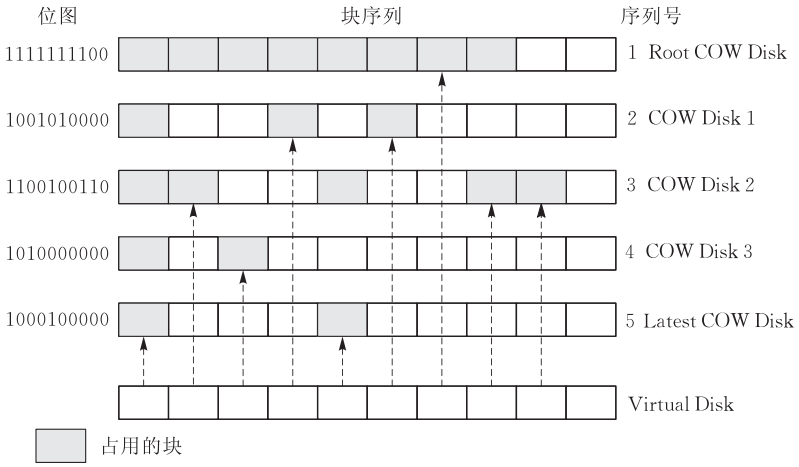


图 2　基于回溯追踪法的块定位

2.1　基于回溯追踪法的块定位

通常每个 COW 磁盘有一个块位图,用来表示该 COW 磁盘中包含的块.块位图可以是 COW 磁盘映像文件的一部分,也可以是一个单独的文件.一个块可能由于被多个 COW 磁盘更新而存在多个版本,每个版本位于不同的 COW 磁盘中,较新的版本位于更靠近 COW 磁盘层次底部的 COW 磁盘中.对于虚拟磁盘上的每个块,块位图用一个位(bit)来表示该 COW 磁盘中是否包含该块的一个版本.

回溯追踪法利用每个 COW 磁盘的块位图来确定磁盘块的最新版本所在的 COW 磁盘,它从 COW 磁盘层次底部的最新 COW 磁盘(Latest COW Disk)开始由下至上查询每个 COW 磁盘的块位图中相应的位,直至发现该位被置 1 或已查询至根 COW 磁盘,如图 2 所示.为了提高查询效率,VMM 通常事先将每个 COW 磁盘的块位图映射到宿主机的内存中以避免位图查询中频繁的磁盘操作.由于定位块的信息分布在多个 COW 磁盘的块位图中,回溯追踪法在 COW 磁盘层次较深的情况下会带来较大的开销:

(1)时间开销.设块 n 的最新版本位于序列号为 seq_n 的 COW 磁盘中,COW 磁盘层次深度为 d,查询 COW 磁盘块位图一次的时间开销为 t,则定位块 n 的时间开销 T_n 可表示为函数 $T_n=(d-seq_n)\times t$,因此定位一个块的时间开销是和 COW 磁盘层次深度成正比的.对于越靠近根 COW 磁盘的块,定位的开销越大.例如,和单个虚拟磁盘映像相比,位于根 COW 磁盘中的应用或数据的访问性能将下降 d 倍,并且会随着 COW 磁盘层次深度的增加而不断下降.

(2)内存开销.设每个 COW 磁盘的块位图大小为 s,COW 磁盘层次深度为 d,则位图映射的内存开销 M 可表示为函数 $M=s\times d$,因此位图映射的内存开销将随着 COW 磁盘层次深度的增加而线性增长.例如,若虚拟磁盘大小为 40GB,块大小为 4KB,COW 磁盘层次深度为 100,则每个 COW 磁盘的块位图大小为 1.25MB,将全部 COW 磁盘的块位图映射到内存中需要占用 125MB 的内存.

2.2　基于 gbf 的块定位

块定位信息的分布性使得回溯追踪法在 COW 磁盘层次较深的情况下会带来较大的开销,因此我们为 COW 虚拟块设备引入了一个全局位图文件 gbf(global bitmap file),为块的定位提供一个集中的访问点.gbf 使用 COW 磁盘的序列号来表示块所在的 COW 磁盘,而不是用一个位来表示 COW 磁盘中是否存在对应的块,如图 3 所示.根据生成顺序,每个 COW 磁盘被分配一个序列号;根 COW 磁盘的序列号为 1,其它 COW 磁盘按照生成顺序依次被分配后续的序列号;序列号 0 保留给空闲块;最大的序列号保留给用户的工作 COW 磁盘 WCD(Work COW Disk).WCD 是在用户端创建的最新 COW 磁盘,用于保存虚拟机运行期间对虚拟磁盘的更新.COW 磁盘的序列号可以表示为一个二元组 $(offset\_seq\_no,\ group\_index)$,其中 $0\leq group\_index<2^{gdx\_width}$,$gdx\_width$ 是组号 $group\_index$ 的位宽;组内序列号 $offset\_seq\_no$ 是 COW 磁盘序列号在所对应组中的偏移量,且满足 $0\leq offset\_seq\_no<2^{osn\_width}$,$osn\_width$ 是 $offset\_seq\_no$ 的位宽.则 COW 磁盘的序列号 seq_no,osn_width,$offset\_seq\_no$,$group\_index$ 满足下列关系:

$$seq\_no = offset\_seq\_no + group\_index \times 2^{osn\_width} \quad (1)$$

若 $gdx\_width=2$ 且 $osn\_width=8$，则最大序列号为 1023，即 COW 磁盘层次深度的最大值为 1023. 可根据实际需要，对 gdx_width 和 osn_width 的大小进行调整以增加或减小最大序列号的值.

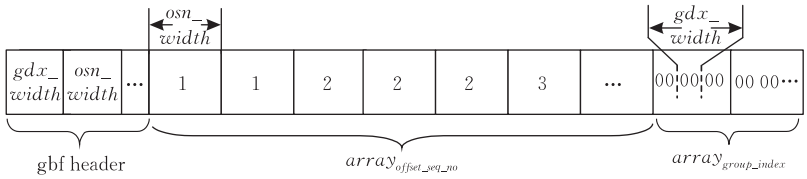


图 3 gbf 结构

gbf 和根 COW 磁盘同时创建，并被初始化为空. 每当客户操作系统对虚拟磁盘进行写操作时，就用最新 COW 磁盘或 WCD 的序列号来更新 gbf 中被写块对应的位置. gbf 主要包含如下 3 部分：

(1) gbf header. 包含一些全局信息，如 COW 磁盘数目、gdx_width、osn_width 等.

(2) $array_{offset\_seq\_no}$. 每个块所在 COW 磁盘的组内序列号数组.

(3) $array_{group\_index}$. 每个块所在 COW 磁盘所对应组号的数组.

基于 gbf，在 COW 磁盘层次结构中定位一个块的最新版本的过程如下，且如图 4 所示：

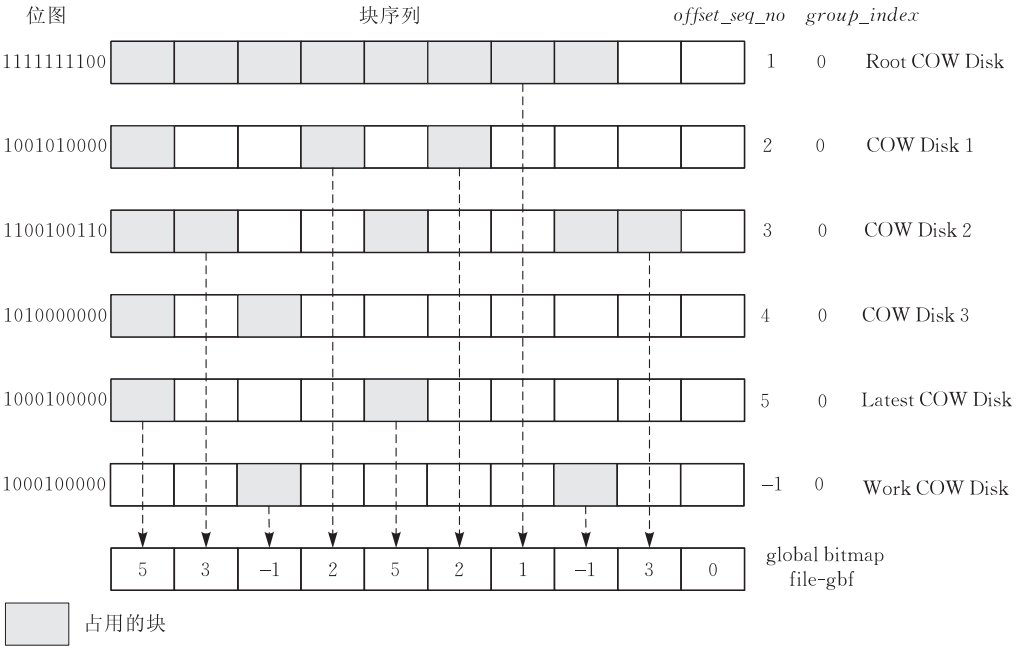


图 4 基于 gbf 的块定位

(1) 当 VMM 中的 COW 虚拟块设备驱动接收到客户操作系统对块 n 的请求后，查询 gbf 块 n 所在 COW 磁盘的组号 $group\_index$ 和组内序列号 $offset\_seq\_no$；

(2) COW 虚拟块设备驱动从序列号可表示为 $(offset\_seq\_no,\ group\_index)$ 的 COW 磁盘中读取块 n，然后将块 n 交给客户操作系统.

从上述过程可以看到，对任何块的定位只需要查询 gbf 两次，因此相比较于回溯追踪法，基于 gbf 的块定位方法能够提高 COW 磁盘的访问性能. 特别是在 COW 磁盘层次较深的情况下，性能的优势会更加明显. 为了避免读写 gbf 导致频繁的磁盘操作，在虚拟机启动时将 gbf 映射到内存中以提高访问效率是非常必要的. 对于一个 40GB、4KB 块大小的虚拟磁盘，gbf 文件大小约为 12.5MB，将整个 gbf 映射到宿主机内存中不会导致较大的内存开销；而且，用于映射 gbf 的内存大小是固定的，不会随着 COW 磁盘层次深度的增加而增加.

3 基于优化的 COW 虚拟块设备的虚拟机按需部署

基于 gbf 块定位方式的 COW 虚拟块设备使得 COW 磁盘的访问性能不再受到 COW 磁盘层次深

度的影响,因而能够生成大量小尺寸的 COW 磁盘,以降低网络环境下虚拟机部署的开销.

在服务器端,基于单个软件的基本粒度,管理员逐一安装所有必要的软件以生成多个小尺寸的 COW 磁盘. COW 磁盘的生成粒度可以根据需要进行调整,例如,如果网络速度较快,可以将多个小规模的软件合并安装到一个 COW 磁盘中以提高部署的效率. COW 磁盘生成完之后,就可以通过如下的 COW 磁盘按需分发和按需取块过程实现虚拟机的按需部署:

1. 初始部署阶段. 从服务器端,将根 COW 磁盘和 gbf 分发到每个用户端;

2. COW 磁盘按需分发. 从服务器端,将用户所需软件的 COW 磁盘分发到用户端;

3. 在用户端,如果虚拟机处于关机状态,首先派生工作 COW 磁盘 WCD,然后启动虚拟机;

4. 最后,用户通过客户操作系统使用在本地缓存的 COW 磁盘中安装的软件;如果虚拟机请求的块不在本地缓存的 COW 磁盘中,则通过按需取块[2,8]的方法从服务器端获取缺失的块;

5. 如果用户还需要服务器端的其它软件,转步 2;如果用户需要服务器端新部署的软件,首先关闭虚拟机,然后将用户端的老版本 gbf 和服务器端的新版本 gbf 进行同步,转步 2.

在用户端虚拟机运行过程中,所有从服务器端获取的 COW 磁盘都处于只读状态,所有的写操作将针对 WCD 进行. 如果未设置保留 WCD,默认情况下,WCD 在虚拟机关闭之后会被删除;否则,用户就能够从最近保存的 WCD、内存等状态中快速恢复挂起的虚拟机. 用户数据存放在网络存储或其它的存储设备上. 对于每个未在本地缓存的COW磁盘,会在本地生成一个对应的影子 COW 磁盘(Shadow COW Disk)[2]. 通过按需取块过程从服务器端获取的块将保存到对应的影子 COW 磁盘中.

对于新增软件,如新种类的软件、软件的升级版本等,管理员首先安装它们以生成对应的 COW 磁盘,然后向用户发布. 在生成新的 COW 磁盘的同时,也会生成新版本的 gbf. 如果用户不需要或服务器端不强制使用新发布的软件,旧版本 gbf 仍然可以继续使用. 否则,在向用户分发新生成的 COW 磁盘之前,必须将用户端的旧版本 gbf 和服务器端的新版本 gbf 进行同步. 在支持稀疏文件(sparse file)的文件系统上,如 Linux Ext2/Ext3 文件系统等,gbf 是稀疏文件,即 gbf 实际占用的磁盘空间通常小于文件大小;另一方面,由于大多数操作系统会尽可能保证磁盘块分配的连续性,且在每个 COW 磁盘的生成过程中,VMM 只用该 COW 磁盘的序列号对 gbf 进行更新,因此 gbf 的文件内容具有局部同一性的特点,4.2.4 节的实验结果表明,具有这种特点的 gbf 是易于压缩的. 因此,通过先压缩再传输进行 gbf 同步的开销是较低的.

按需取块过程可以通过两个 C/S 模式的网络程序来实现. 设客户程序 cow_client 和 VMM 运行在用户端,服务程序 cow_server 运行在服务器端,request_blocks 函数在发生缺块时发送取块请求,read_blocks 函数从服务器端存储的 COW 磁盘中读取用户端请求的磁盘块,send_blocks 函数发送读取的磁盘块,save_blocks 将接收到的磁盘块缓存到本地,则按需取块过程中 VMM、cow_client 以及 cow_server 之间的交互过程如图 5 所示.

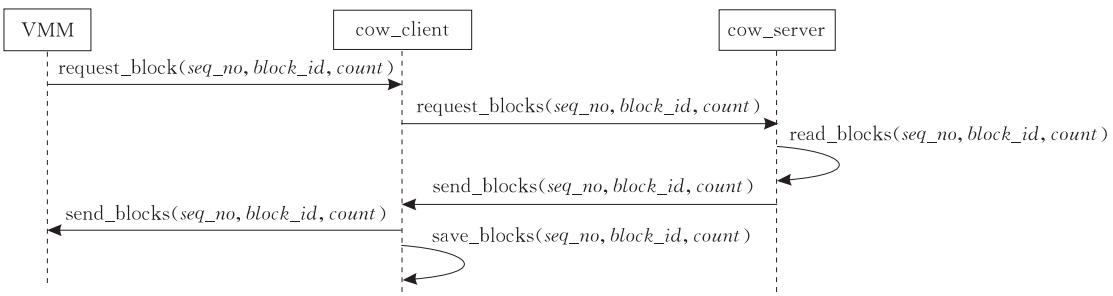


图 5　按需取块过程

基于 COW 磁盘的按需分发和按需取块,我们设计了基于 COW 磁盘有效工作集的优化部署、COW 磁盘回收方法来解决大型软件部署、软件的持久更新等关键问题.

3.1　大型软件的部署优化

安装大型软件生成的大尺寸 COW 磁盘会带来较高的部署开销,特别是那些需要部署到每个用户端的 COW 磁盘,例如,根 COW 磁盘. 对于大型软件,用户在一次虚拟机会话中用到的软件功能通常只占所有功能的一小部分[9],因此这一小部分软件功能对应的磁盘块也通常只占整个 COW 磁盘的一小部分,我们将这部分被用户访问的磁盘块构成的

集合称为 COW 磁盘工作集. 如果只向用户分发小尺寸的 COW 磁盘工作集,而不是整个 COW 磁盘,则能够极大地降低大型软件的部署开销.

COW 磁盘工作集随时间变化且因人而异,因此我们针对大型软件定义了 COW 磁盘有效工作集的概念. COW 磁盘有效工作集,是指 COW 磁盘中用于运行软件最常用功能的磁盘块的集合. COW 磁盘有效工作集是静态的,大小取决于软件最常用功能的定义. 软件最常用功能的定义既要包括该软件用户群中多数用户常用的功能,又不能涵盖功能太多使得有效工作集过大而失去意义,因此它的定义最好有软件用户的参与. 设大型软件 A 的 COW 磁盘为 COW_a,则可通过按需取块过程获取 COW_a 的有效工作集:

1. 在无任何 COW 磁盘缓存的用户端,从服务器端获取最新版本的 gbf,派生 WCD,然后启动虚拟机;

2. 启动软件 A,逐一运行为 A 定义的最常用功能;

3. 关闭虚拟机,用户端 COW_a 的影子 COW 磁盘中缓存的磁盘块构成的集合即为 COW_a 的有效工作集;

4. 将获得的影子 COW 磁盘作为 COW_a 的有效工作集存储到服务器端,当用户请求软件 A 时,则将 COW_a 的有效工作集分发给用户.

对于根 COW 磁盘,步 2 可采取不同的执行方式以获得不同大小的有效工作集. 最简单的是省略步 2,则获得的有效工作集只包含系统启动所需要的磁盘块;如果在步 2 中运行一些根 COW 磁盘中安装的常用软件,例如文件管理器、终端等程序,则可在有效工作集大小不超过一定范围的条件下,使用户获得更好的软件使用体验. 由于根 COW 磁盘通常需要在初始部署阶段部署到每个用户的机器上,如果根 COW 磁盘的有效工作集较小,则基于 COW 磁盘有效工作集的部署方法将大大地降低根 COW 磁盘的部署开销.

如果一种大型软件的 COW 磁盘有效工作集中的大部分磁盘块由于软件更新而被更新,则需要重新生成该软件的 COW 磁盘有效工作集. 一些工具可以用来自动化 COW 磁盘有效工作集的生成,以降低 COW 磁盘有效工作集的维护开销,如 Linux 平台上的宏记录/回放工具 JW_Record_Playback①.

通过部署较小尺寸的 COW 磁盘有效工作集,而不是整个 COW 磁盘,不仅能够较好地满足用户的软件使用需求,也能够大幅降低大尺寸 COW 磁盘的部署开销. 对于网络带宽有限的计算环境,基于 COW 磁盘有效工作集的部署方法可以扩展到其它更多中等规模的软件,以提高虚拟机部署的效率.

3.2 软件持久更新

频繁或长时间累积的软件升级更新会导致生成大量的 COW 磁盘,并因此带来两方面的问题:一方面,虽然 gbf 可以支持大量的 COW 磁盘,但仍然有最大数目(最大序列号)的限制,而且大量的 COW 磁盘也带来管理上的不便[10];另一方面,软件升级后,旧版本软件的 COW 磁盘占用的宿主机上的磁盘空间并未被释放,累计的大量旧版本软件的 COW 磁盘会造成严重的磁盘空间浪费. 通过 COW 磁盘合并的方法可以减少 COW 磁盘的数目,但合并后的 COW 磁盘尺寸却会大幅增加.

基于前期工作对虚拟磁盘映象回收机制的研究[11],我们设计了一种针对优化的 COW 虚拟块设备的 COW 磁盘回收方法,以回收旧版本软件占用的磁盘空间并将 COW 磁盘数目控制在一定范围之内. 设所有采用全量更新方式的软件都是通过先卸载旧版本再安装新版本的方式进行更新,安装新版本软件 A' 对旧版本软件 A 升级后生成了 COW 磁盘 $COW_{a'}$,则通过如下方法回收旧版本软件 A 的 COW 磁盘 COW_a,且如图 6 所示.

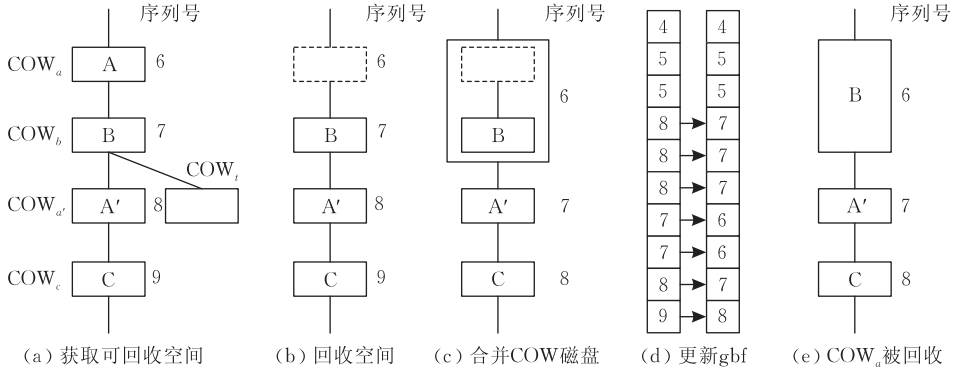


图 6 COW 磁盘回收过程

① JW_Record_Playback. http://members.home.nl/wijnenjl/index7o.html

1. 获取可回收空间. 通过如下方法获取可回收块集合 *free_blocks*;

1.1. 基于 $COW_{a'}$ 的父亲 COW 磁盘 COW_b 创建临时 COW 磁盘 COW_t,然后从 COW_t 启动虚拟机,卸载旧版本软件 A,用零填充虚拟磁盘的剩余空间(如图 6(a)所示);

1.2. 比较零填充前后的 gbf,将那些从 COW_a 的序列号变化为 COW_t 的序列号的块加入可回收块集合 *free_blocks* (如图 7 所示);

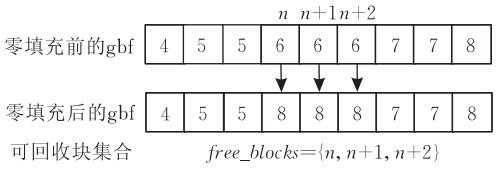


图 7　获取可回收块集合

1.3. 关闭虚拟机,删除 COW_t.

2. 回收空间. 对于 COW_a,用零填充 *free_blocks* 集合中的块,同时更新 COW_a 的块位图;然后使用工具(如 GNU 的 cp 等)稀疏化 COW_a 映像文件以释放旧版本软件 A 占用的宿主机上的磁盘空间(如图 6(b)所示);

3. 合并 COW 磁盘. 将 COW_a 和邻居 COW 磁盘 COW_b 进行合并,然后删除 COW_a(如图 6(c)所示);

4. 更新 gbf. 将 gbf 中所有大于 COW_a 序列号的序列号减 1(如图 6(d)所示).

在合并 COW 磁盘之前,由于旧版本软件的 COW 磁盘占用的宿主机上的磁盘空间已大部分被释放,因此合并后的 COW 磁盘大小不会大幅增长. COW 磁盘合并是根据被合并父子 COW 磁盘的块位图将父亲 COW 磁盘的内容合并到儿子 COW 磁盘中. 合并之后,由于部分 COW 磁盘的序列号发生变化,gbf 必须更新且所有用户端的旧版本 gbf 必须强制性地和服务器端的新版本 gbf 进行同步. 旧版本软件的 COW 磁盘在 COW 磁盘合并后可以被删除,COW 磁盘的数目将减少,如图 6(e)所示. 上述 COW 磁盘回收的基本方法也能够很容易地扩展到同时回收多个 COW 磁盘的情况.

对于采用增量更新方式的软件,例如软件补丁,可通过直接安装并生成对应 COW 磁盘的方式进行部署. 在其它 COW 磁盘被回收的过程中,一旦软件的原始安装包和增量更新包对应的 COW 磁盘成为父子关系,就将它们合并.

4　实现与实验

基于 Linux 平台和 QEMU 虚拟机,我们实现了优化的 COW 虚拟块设备和虚拟机按需部署的原型系统,并通过实验验证了优化方法和各项关键技术的有效性.

4.1　实　现

4.1.1　优化的 COW 虚拟块设备

我们在 QEMU 中开发了一个新的 COW 虚拟块设备驱动 mcow 来实现基于 gbf 的块定位方法. 由于 VMM 和上层文件系统之间的语义隔阂[12],VMM 层的块设备驱动通常是以物理扇区为单位进行读写. 我们利于文件系统格式这种通用抽象[12],在 mcow 中实现了以文件系统块为基本单位的块大小可变的读写模式,提高了读写效率且大幅减小了 gbf 文件的尺寸. mcow 也实现了和外部通信的接口以支持按需取块.

4.1.2　虚拟机按需部署工具和服务

基于 mcow,我们开发了相关的工具和服务以支持虚拟机环境下虚拟机的按需部署. 用户通过统一的 Web 界面选择所需软件的种类和版本,按需部署服务根据用户的选择调用 COW 磁盘传输服务,将对应的 COW 磁盘传输到用户机器上. gbf 同步在后台自动完成,对用户是透明的.

我们用 C 语言开发了网络程序 cow_client 和 cow_server 来实现按需取块过程. 运行在用户端的 cow_client 通过分别和本地的 QEMU 以及服务器端的 cow_server 建立的 TCP 链接转发 mcow 发出的缺块请求以及接收从 cow_server 发来的磁盘块,实现同步的按需取块过程. 我们还开发了工具 cow_merger 来实现 COW 磁盘回收过程. 当服务器端的 COW 磁盘数目超过一定阈值时,可利用 cow_merger 来回收旧版本软件的 COW 磁盘,将 COW 磁盘数目控制在一定范围内.

4.2　实　验

实验中的服务器和用户机均配置有主频为 1.8GHz 的 Intel 双核处理器和 1GB 内存,并通过 100Mbps 的以太网相连. 宿主和客户操作系统均为 Fedora 7. 经过扩展的包含 mcow 驱动的 QEMU 0.9.0 作为 VMM 部署在服务器端和用户端. 虚拟机配置了 512MB 内存和 10GB 的虚拟磁盘. 我们在服务器端准备了 16 种常用软件用于生成 COW 磁盘,并将它们的安装包存放在一个 1GB 大小的虚拟磁盘映像中,该虚拟磁盘在虚拟机启动时会被自动挂载. 表 1 列出了实验中使用的软件.

表 1 创建 COW 磁盘的软件包

序列号	COW 磁盘	压缩尺寸/MB
1	F7	661.8
2	firefox 2.0	21.8
3	jre 1.5.0_10	28.3
4	audacity 1.3	7.2
5	gimp 2.2	69.8
6	scilab 4.12	16.2
7	mysql 5.0	25.0
8	openoffice.writer	84.0
9	emacs 22.0	19.4
10	pidgin 2.2	7.5
11	skype 2.0	23.3
12	adobe reader 7.01	50.0
13	mplayer 1.0	9.8
14	stardict 3.0	3.5
15	eclipse(cpp) 3.3	45.2
16	realplayer 10.0	12.1
17	opera 9.21	5.3

4.2.1 COW 磁盘大小与部署开销

在 Linux/QEMU 的 COW 模式下生成的 COW 磁盘是稀疏文件，传输大尺寸的稀疏文件会带来较大的不必要的传输开销. 根据我们的测量结果，在服务器端和用户端之间传输一个 10GB 未经压缩的虚拟磁盘映象平均需要 22.1min，这种开销对普通用户来说通常是不可接受的. 可以在服务器端利用 tar 和 bzip2 等工具将所有 COW 磁盘进行压缩，按需分发到用户端后，再解压缩恢复成可被 VMM 直接使用的稀疏文件. 压缩产生的开销是一次性的，并且对用户是透明的. 为了支持按需取块，服务器端仍然保留未压缩的 COW 磁盘.

从表 1 可以看出，除了根 COW 磁盘 F7 之外，其它 COW 磁盘压缩后的大小都不超过 100MB，因此相比较于单个大尺寸的虚拟磁盘映像，基于单个软件粒度生成的 COW 磁盘的传输开销将大大降低. 为了比较的公平性，我们在用户端测量了每个 COW 磁盘的解压缩时间，并和传统的软件安装/配置过程的开销进行了对比. 我们选择表 1 中序列号 2～9 的软件进行了测量，测量结果如图 8 所示. 从图 8 可以看出，所有 COW 磁盘解压缩恢复成稀疏文件的时间不超过 30s，每种软件的 COW 磁盘解压缩时间都低于甚至远低于传统的软件安装/配置过程的时间开销.

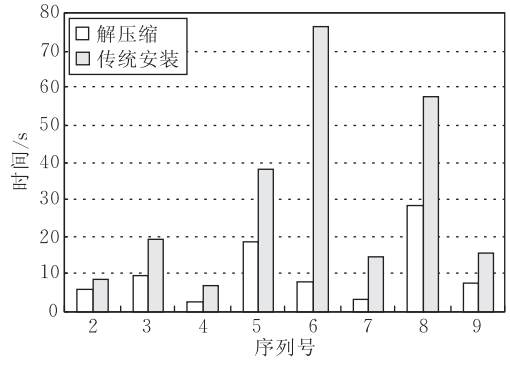


图 8 COW 磁盘解压缩和传统软件安装过程的比较

4.2.2 COW 虚拟块设备性能比较

我们利用开源的数据库基准测试程序 OSDB (Open Source Database Benchmark)[①]来测试优化的 COW 虚拟块设备 mcow 在不同深度的 COW 磁盘层次下的性能. OSDB 通过多种数据库操作来测试一种数据库系统的性能，因此会在底层的块设备驱动层产生大量的 I/O 操作. 实验采用 OSDB 0.21 和 PostgreSQL 8.2 数据库系统. 为了比较，在相同的条件下也测试了 QEMU 中原有的 COW 虚拟块设备驱动 cow. 将 PostgreSQL 和 OSDB 使用的测试数据包含在根 COW 磁盘中，而将 OSDB 测试程序安装到一个单独的 OSDB-COW 磁盘中. OSDB-COW 磁盘分别以第 4,8,12,16 4 种不同的次序生成，从而产生不同深度的 COW 磁盘层次结构. 对于每种次序，对两种驱动分别运行 OSDB 对根 COW 磁盘中的 PostgreSQL 数据库进行各种操作. 在每一轮 OSDB 测试中，既进行了 IR (Information Retrieval)测试，也进行了 OLTP(Online-Transaction Processing)测试. 图 9 给出了测试结果，横坐标是 COW 磁盘层次深度，纵坐标是每秒读写的数据库元组(tuple)数目. 从测试结果可以看出，mcow 的性能优于 cow；特别是在 COW 磁盘层次较深的情况下，mcow 的性能优势更加明显.

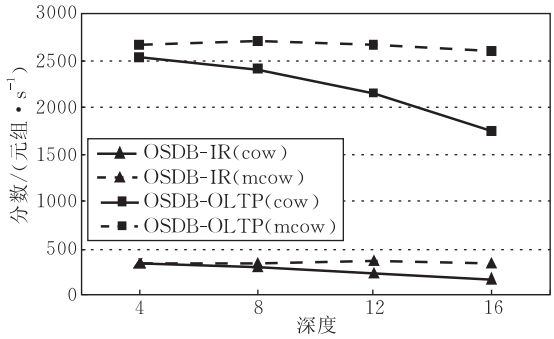


图 9 优化前后 COW 虚拟块设备驱动的性能比较

4.2.3 COW 磁盘有效工作集

我们对根 COW 磁盘 F7 以及序列号为 5,8,12 的 3 个尺寸较大的 COW 磁盘采用按需取块的方法获得它们的有效工作集. 表 2 给出了它们最常用软件功能的定义、COW 磁盘及其有效工作集的大小(未压缩)比较.

① Open Source Database Benchmark. http://osdb.sourceforge.net

表 2　COW 磁盘及其有效工作集的大小比较

序列号	COW 磁盘	最常用软件功能	实际大小/MB	有效工作集/MB	比例/%
1	F7	启动、终端、文件管理器	2596	189.2	7.3
5	gimp 2.2	常规功能*、图层(新建、复制、删除、合并)滤镜(模糊、增强、扭曲、通用、噪音)、选择工具、涂画工具、变换工具	157.3	13.0	8.3
8	openoffice.writer	常规功能*、输出、查找替换、表格、绘图、插入(页眉页脚、特殊字符、图片)、字符、列、段落、项目符号和编号、页面、选项	227.6	76.0	33.5
12	Adobe reader 7.01	常规功能*、另存为文本、属性、全部选定、查找、搜索、首选项、页面布局、基本工具、缩放工具	124.8	69.2	55.4

注:常规功能:打开、关闭、保存、复制、剪切、粘贴、另存.

从表 2 可以看出,虚拟机正常启动只需要根 COW 磁盘约 7.3% 的磁盘块,因此通过部署根 COW 磁盘的有效工作集,能够极大地降低初始部署阶段根 COW 磁盘的部署开销.其它 3 种软件的 COW 磁盘有效工作集占 COW 磁盘大小的比例平均为 32.4%,最大为 55.4%,因此通过部署大型软件的 COW 磁盘有效工作集,不仅能获得软件常用功能的本地访问,而且能够大幅降低部署开销.

4.2.4　gbf 同步开销

在生成每个 COW 磁盘之前,我们都对 gbf 做了备份,因此最终在服务器端会生成 17 个版本的 gbf,每个版本较之前一个版本新增加了一种软件.表 3 给出了各个版本的 gbf 使用 tar 和 bzip2 压缩前后的大小及压缩率.从表 3 可以看出,gbf 压缩率平均为 0.92%,最大为 1.4%,因此 gbf 是易于压缩的.压缩后的 gbf 尺寸非常小,最大不超过 16KB,因此通过先压缩再传输的方式进行 gbf 同步的开销是很低的.

表 3　压缩前后的 gbf 大小及压缩率

序列号	实际大小/KB	压缩/KB	压缩率/%
1	843	2.1	0.2
2	855	4.0	0.5
3	883	5.2	0.6
4	891	6.2	0.7
5	913	6.8	0.7
6	946	6.9	0.7
7	955	7.9	0.8
8	996	8.3	0.8
9	1013	8.9	0.9
10	1018	9.7	1.0
11	1029	11.0	1.1
12	1061	11.9	1.1
13	1070	13.2	1.2
14	1072	13.6	1.3
15	1110	14.6	1.3
16	1131	15.2	1.3
17	1142	15.7	1.4

4.2.5　COW 磁盘回收

对表 1 中序列号为 3,5,12 的 3 个 COW 磁盘中的软件,我们采用先卸载旧版本再安装新版本的方式,将它们的新版本 jre1.5.0_14,gimp 2.4,adobe reader 8.1.2 分别安装到序列号为 18,19,20 三个新创建的 COW 磁盘中,然后利用 COW 磁盘回收方法以及 cow_merger 工具对 3 个旧版本软件的 COW 磁盘进行了回收,使得 COW 磁盘数目保持不变.表 4 对回收过程中被合并 COW 磁盘合并前后的大小(未压缩)进行了比较.

由于旧版本软件的 COW 磁盘中部分磁盘块可能包含全局或其它软件的文件系统元数据,因此这部分磁盘空间不能被释放,从而导致合并后的 COW 磁盘大小略有增加.从表 4 可以看出,合并之后的 COW 磁盘大小增幅很小,因此旧版本软件的 COW 磁盘占用的宿主机上的磁盘空间已大部分被释放.每回收一个 COW 磁盘,我们在客户操作系统中检查新版本软件是否能正确运行.实验表明,COW 磁盘被回收之后,升级更新后的新版本软件都能够正确运行.

4.2.6　小　结

通过在原型系统上的实验,我们可以得出下面的结论:

(1)优化的 COW 虚拟块设备驱动 mcow 的性能优于 QEMU 中原有的 cow 驱动,在 COW 磁盘层次深度较深的情况下,性能优势更加明显;

(2)相比较于单个大尺寸的虚拟磁盘映像,基于单个软件粒度生成的 COW 磁盘的传输开销将大大降低;通过部署 COW 磁盘有效工作集能够大幅降低大型软件的部署开销;

(3)利用 COW 磁盘回收方法能够有效回收旧版本软件占用的空间和控制 COW 磁盘的数目,支持软件的持久更新.

表 4 COW 磁盘合并前后的大小比较

合并前			合并后			增幅/MB
序列号	COW 磁盘	实际大小/MB	序列号	COW 磁盘	实际大小/MB	
4	audacity 1.3	29.7	3	audacity 1.3	29.9	0.2
6	scilab 4.12	130.5	4	scilab 4.12	131.4	0.9
13	mplayer 1.0	31.2	10	mplayer 1.0	31.6	0.4

5 相关工作

斯坦福大学的 Collective[2,13-14] 项目基于虚拟机提出了虚拟装置(Virtual Appliance,VAP)的概念,试图通过虚拟装置降低计算环境中软件管理的开销,建立更加易用和易于管理的全局计算设施.由于每个 VAP 是包含完整虚拟机状态的虚拟机映象,VAP 是比 COW 磁盘更加重量级的实体,因此通过网络部署 VAP 的开销较大. Sapuntzakis[2] 等提出了有效的技术来优化虚拟机在低速网络链路上的迁移.然而,他们提出的 COW 磁盘优化技术难以实现 COW 磁盘大小和磁盘访问性能之间的平衡,且难以支持持久的软件更新.

卡内基梅隆大学和 Intel 联合发起的研究项目 ISR[8,15] 利用分布式文件系统和虚拟机技术,使得用户能够从网络上不同的站点访问个人的计算环境.由于虚拟机状态的数据量较大,ISR 利用按需取块、基于内容寻址的存储(CAS)等多种优化技术来减少通过网络传输的虚拟机状态的数据量. GVFS[16] 扩展了分布式文件系统 NFS 以支持网格环境下虚拟机状态的按需传输,例如客户端代理管理的磁盘缓存、元数据处理等. ISR 和 GVFS 针对的都是单个大尺寸的虚拟磁盘映像.

VioCluster[17] 和 SODA[18] 利用虚拟机分别来实现集群和网格环境下的软件部署,它们通过对虚拟磁盘中的软件进行精心的裁减以使得尺寸不致过大,然后再分发给每个计算节点.这种方法缺乏更加细粒度的软件可重用性以及适应不同软件需求的灵活性. Appstream 的 VID[9] 技术基于软件功能将单个虚拟磁盘映像从逻辑上分成多个小片段,然后按需地将这些片段流到用户的桌面上. VID 实际上是一种基于单个大尺寸虚拟磁盘映像的按需取块方案,但它以更加主动和可控的方式来实现,以提高分发效率. PDS[19] 在操作系统和应用之间插入一个虚拟化器(virtualizer)来有选择性地截获部分 Windows 系统调用,以提供一个虚拟执行环境来按需地部署和执行 asset——一种类似于 COW 磁盘的软件映像. Desktop2Go[20] 改进了现有多计算资源的软件管理方式,将应用层虚拟化技术引入到软件服务系统之中,设计并实现了程序的虚拟运行环境. PDS 和 Desktop2Go 都是通过截获大量 Windows 系统调用来实现虚拟的运行环境,实现机制较为复杂且和平台相关.

Mirage[10] 设计了一种新的虚拟磁盘映像格式 MIF 来优化虚拟机环境下大量虚拟磁盘映像的存储,并能够透明地对虚拟磁盘映像中的软件执行查找、更新等操作. Parallax[3] 是一个基于块级别虚拟化的分布式虚拟机存储系统,每个物理主机上运行一个存储虚拟机,通过访问一个后端的共享块存储(shared block store)为其它的前端虚拟机提供存储服务. Mirage 和 Parallax 都是在块的级别上执行软件管理任务,需要专门的块存储,且这种细粒度可能会带来额外的开销,例如,虚拟磁盘映像组合/分解的开销. Machine Bank[21] 将虚拟磁盘的磁盘块存储在基于内容寻址的存储系统中,并通过按需取块的方法将虚拟磁盘映像从服务器端流到客户端,然而,从基于内容寻址的块存储中按需取块比访问本地缓存的 COW 磁盘具有较差的性能.

6 结论及展望

COW 虚拟块设备对实现低开销的、按需的虚拟机部署,降低虚拟机环境下软件管理的开销具有重要意义.本文为虚拟机管理器 VMM 中的 COW 虚拟块设备设计了一种基于全局位图文件 gbf 的块定位方法,提高了 COW 磁盘的访问性能且使得 COW 磁盘访问性能不再受 COW 磁盘层次深度的影响.基于优化的 COW 虚拟块设备,本文提出在单个软件的基本粒度上生成多个小尺寸的 COW 磁盘,然后按需地分发到用户端,以降低网络环境下虚拟机部署的开销;并设计了基于 COW 磁盘有效工作集的优化部署、COW 磁盘回收方法来解决大型软件部署以及软件的持久更新等关键问题.

基于 Linux 平台,我们在 QEMU 中开发了新的驱动 mcow 来实现优化的 COW 虚拟块设备,并

基于 mcow 实现了虚拟机按需部署的原型系统.实验表明,mcow 的性能优于 QEMU 中原有的 cow 驱动;基于 COW 磁盘有效工作集的部署方法能够有效降低大型软件的部署开销;COW 磁盘回收方法能够将 COW 磁盘数目控制在一定范围内且减少了磁盘空间的浪费,支持软件的持久更新.

相同的 COW 磁盘经过一段时间后可能会在网络环境下多个用户的机器上缓存.如果将这些分布在多个用户机器上的 COW 磁盘组织成一个 P2P 对等模式的虚拟磁盘映像共享系统,将有可能降低服务器压力,对进一步提高计算环境的可用性、软件的可重用性以及分布环境下虚拟机迁移的效率等方面都具有重要的意义,这也是我们下一步即将开展的工作.

参　考　文　献

[1] Smith James E, Nair Ravi. The architecture of virtual machines. IEEE Computer, 2005, 38(5): 32-38

[2] Sapuntzakis C, Chandra R, Pfaff B, Chow J, Lam M, Rosenblum M. Optimizing the migration of virtual computers//Proceedings of the 5th Symposium on Operating Systems Design and Implementation. Boston, MA, USA, 2002: 377-390

[3] Meyer Dutch T, Aggarwal Gitika, Cully Brendan, Lefebvre Geoffrey, Feeley Michael J, Hutchinson Norman C, Warfield Andrew. Parallax: Virtual disks for virtual machines//Proceedings of the ACMSIGOPS/EuroSys European Conference on Computer Systems 2008 (EuroSys'08). Glasgow, Scotland, UK, 2008: 41-54

[4] Figueiredo R, Dinda P, Fortes J. A case for grid computing on virtual machines//Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS 2003), Providence, RI, USA, 2003: 550-559

[5] Bradford Robert, Kotsovinos Evangelos, Feldmann Anja, Schioberg Harald. Live wide-area migration of virtual machines including local persistent state//Proceedings of the 3rd ACM/USENIX Conference on Virtual Execution Environments (VEE'07). San Diego, California, USA, 2007: 169-179

[6] Bellard Fabrice. QEMU, A fast and portable dynamic translator//Proceedings of the USENIX Annual Technical Conference (USENIX'05). Anaheim, CA, USA, 2005: 41-46

[7] Dike J. A user-mode port of the Linux kernel//Proceedings of the USENIX Annual Linux Showcase and Conference. Atlanta, GA, 2000: 63-72

[8] Satyanarayanan M, Gilbert B, Toups M et al. Pervasive personal computing in an Internet suspend/resume system. IEEE Internet Computing, 2007, 11(2): 16-25

[9] Appstream Inc. Appstream Virtual Image Distribution. Whitle Paper. September 2005

[10] Darrell Reimer, Arun Thomas, Glenn Ammons, Todd Mummert, Bowen Alpern, Vasanth Bala. Opening black boxes: Using semantic information to combat virtual machine image sprawl//Proceedings of the 4th ACM/USENIX Conference on Virtual Execution Environments (VEE'08). Seattle, Washington, USA, 2008: 111-120

[11] Chen Bin, Xiao Nong, Cai Zhi-Ping, Chu Fu-Yong, Wang Zhi-Ying. Virtual disk image reclamation for software updates in virtual machine environments//Proceedings of the 4th IEEE International Conference on Networking, Architecture, and Storage (NAS'09). Zhangjiajie, China, 2009: 43-50

[12] Chen P M, Noble B D. When virtual is better than real//Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS'01), 2001: 133-138

[13] Sapuntzakis C, Brumley D, Chandra R, Zeldovich N, Chow J, Norris J, Lam M S, Rosenblum M. Virtual appliances for deploying and maintaining software//Proceedings of the 17th USENIX Large Installation System Administration Conference. San Diego, USA, 2003: 181-194

[14] Chandra R, Zeldovich N, Sapuntzakis C, Lam M S. The collective: A cache-based system management architecture//Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05). Boston, MA, USA, 2005: 259-272

[15] Nath Partho, Kozuch Michael A, O'Hallaron David R, Harkes Jan et al. Design tradeoffs in applying content addressable storage to enterprise-scale systems based on virtual machines//Proceedings of the USENIX Annual Technical Conference (USENIX'06). Boston, Massachusetts, USA, 2006: 71-84

[16] Zhao M, Zhang J, Figueiredo R. Distributed file system virtualization techniques supporting on-demand virtual machine environments for grid computing. Journal of Cluster Computing, 2006: 45-56

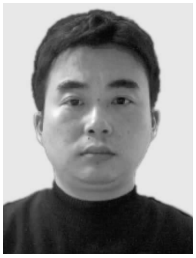
[17] Ruth Paul, McGachey Phil, Xu Dong-Yan. VioCluster: Virtualizatioin for dynamic computational domains//Proceedings of the IEEE International Conference on Cluster Computing, 2005: 1-10

[18] Jiang X, Xu D. SODA: A service-on-demand architecture for application service hosting utility platforms//Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC 2003). Seattle, WA, 2003: 174-183

[19] Alpern Bowen, Auerbach Joshua, Bala Vasanth, Frauenhofer Thomas, Mummert Todd, Pigott Michael. PDS: A virtual execution environment for software deployment//Proceedings of the 1st ACM/USENIX Conference on Virtual Execution Environments (VEE'05). Chicago, Illinois, USA, 2005: 89-99

[20] Zhang Youhui, Wang Xiaoling, Hong Liang. Portable desktop applications based on P2P transportation and virtualization//Proceedings of the 22nd Large Installation System Administration Conference(LISA'08). San Diego, USA, 2008: 133-144

[21] Tang S, Chen Y, Zhang Z. Machine Bank: Own your virtual personal computer//Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS'07). Long Beach, California, USA, 2007: 1-10



CHEN Bin, born in 1975, Ph. D. candidate. His main research interests include virtualization technology, distributed computing.

XIAO Nong, born in 1969, professor, Ph. D. supervisor. His main research interests include grid computing, networked storage, computer architecture, virtualization technology.

CAI Zhi-Ping, born in 1975, Ph. D, associate professor. His main research interests include virtualization technology, network security.

WANG Zhi-Ying, born in 1956, professor, Ph. D. supervisor. His main research interests include high-performance computer architecture, virtualization technology.

Background

As an important system-level virtualization technology, virtual machine (VM) has many advantages over traditional methods in addressing issues such as heterogeneity, system management, mobility, etc. Software deployment is one of its important applications. Using the VM technology to help with software deployment will undoubtedly speed up the deploying process by replacing the normal installation process with an easily-moved virtual disk (VD) image. However, distributing the traditional one-piece large-sized VD image to each user will cause great transfer cost over the network and unnecessary disk space wastage, especially in large-scale virtual machine environments. The COW (Copy-on-Write) virtual block device (VBD) supported by many virtual machine monitors (VMM) can be used to split the one-piece large-sized VD image into multiple smaller-sized VD images (COW disk). However, it is difficult to make tradeoff between disk access performance and the size of COW disk for existing COW VBDs. This paper presents an optimization approach for the COW VBD driver in VMM, which supports making many small-sized COW disks without impairing the access performance of virtual disk. Based on the optimized COW VBD, this paper further proposes making COW disks at a basic granularity of single kind of software to make COW disks as small as possible and then distributing the small-sized COW disks to users on demand. The authors also design an effective working set-based approach to optimize the deployment of those kinds of large-scale software and a COW disk reclamation approach to support frequent or long-term software updates in virtual machine environments.

This work is supported by the National Basic Research Program of China (973) under grant No. 2007CB310900, the National Natural Science Foundation of China under grant No. 60736013, the National Natural Science Foundation of China under grant No. 60603062 and the Natural Science Foundation of Hunan Province of China under grant No. 06JJ3035. Related work of the research group has been published in some top-level conferences, such as DAC2008 (the 45th Design Automation Conference), and some prestigious journals, such as IET Computer & Digital Techniques, Chinese Journal of Software, Chinese Journal of Electronic, etc.