第35卷 第16期 计 算 机 工 程 2009年8月
Vol.35 No.16 Computer Engineering August 2009
·博士论文· 文章编号：1000—3428(2009)16—0017—03 文献标识码：A 中图分类号：N945

动态地址映射虚拟存储系统

柯 剑[1,2]，朱旭东[1,2]，那文武[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 100039)

摘 要：针对静态资源管理方式存储资源利用率低、无法满足多类应用不同需求的问题，设计实现一种基于动态机制的虚拟存储系统ASD。采用写时分配策略解决存储资源利用率低的问题，采用地址动态映射机制实现按照数据特性管理存储资源。测试表明，与静态管理方式相比，以ASD为基础的ext2文件系统的资源利用率和读写性能有明显改善。

关键词：存储虚拟化；写时分配；动态地址映射；数据布局

Dynamic Address Mapping Virtualization Storage System

KE Jian[1,2], ZHU Xu-dong[1,2], NA Wen-wu[1,2], XU Lu[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

【Abstract】There are two problems in static storage resource management that its storage resource utilization is inefficient and it can not meet the requirements of multiple-class applications. This paper designs and implements a virtual storage system named ASD(Allocate-on-demand Storage Device) to solve the problems, which uses the allocate-on-demand policy to relieve the problem of underutilization of storage resource, and uses the dynamic mapping mechanism to manage storage resource in accordance with the access characteristics of applications. The results of test indicate that, compared with the static management method, the performance and resource utilization of Linux ext2 file system based on ASD platform has distinguished improvement.

【Key words】storage virtualization; allocate-on-demand; dynamic address mapping; data layout

1 概述

虚拟化存储技术具有简化存储空间管理、降低管理费用的优点，因此，得到了广泛的应用。LVM[1]是Linux平台上普遍使用的虚拟化存储系统，它使用资源预留和静态地址映射技术，具有逻辑清晰、结构简单、易于实现等优点。但其忽略了存储系统以下3个特性，导致存储资源未被充分利用：

(1)应用对存储空间的需求是一个渐进增长的过程。应用在运行初期对存储的需求较少，随着运行时间的持续，用户对存储的需求逐渐增加。

(2)应用访问特征的多样性。以文件系统操作为例，I/O访问通常是元数据与用户数据的混合型，用户数据在全部I/O中只占13%~41%，其余均为系统数据和元数据[2]。为了保证数据的一致性，文件系统中元数据一般采用同步读写方式，元数据访问的性能极大地影响了整个应用的性能。

(3)存储介质的异构性增加。随着SSD、MEMS等设备成本的下降，它们在存储系统中的使用逐渐增加。存储系统的物理资源是由具有性能量级差的异构存储介质所构成的。

资源预留分配方式在虚拟设备创建时就预留了一定的物理存储空间，只能被该虚拟设备使用。在运行初期应用存储需求较少时，大量预先分配的存储资源未被利用，造成存储资源的浪费。其次，采用静态映射，存储系统无法按照数据特性进行存储资源的分配和调整，存储系统中快速的SSD设备只能按照磁盘设备的速度提供服务，造成SSD设备没有被充分利用。为了解决上述问题，本文设计和实现了一种写时分配动态地址映射的虚拟存储系统ASD(Allocate on-demand Storage Device)，按照应用特性分配存储资源，提高资源的有效利用率和应用的性能。

2 写时分配动态映射机制

写时分配指在虚拟设备创建时不预留物理存储资源，而在发生数据写操作时进行物理资源的分配。动态映射指逻辑地址到物理地址映射关系不是事先确定的，而是在物理资源分配时按照实际获取的物理资源的地址建立映射关系，并且可以动态修改。如图1所示，逻辑设备(VD)在创建时只有逻辑空间，随着用户数据量的增加，实际分配的物理资源分布在各个物理设备上，逻辑地址与物理地址不是公式化地一一映射。

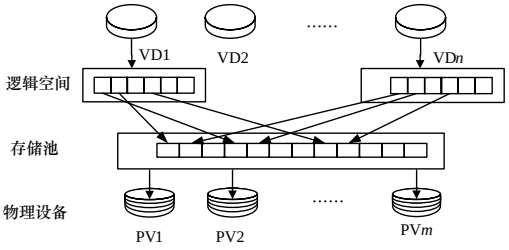


图1 逻辑地址到物理地址的动态映射

为了便于进行存储资源的动态分配和回收，写时分配技术中物理资源被切分为固定大小的存储块，其大小称为映射粒度，物理资源的分配和回收以存储块为单位进行。

基金项目：国家“863”计划基金资助项目(2007AA01Z402)；国家“973”计划基金资助项目(2004CB318205)

作者简介：柯 剑(1971－)，男，博士研究生，主研方向：网络存储，虚拟存储；朱旭东、那文武，博士研究生；许 鲁，研究员、博士生导师

收稿日期：2009-01-02 **E-mail：**kejian@ict.ac.cn

动态地址映射的基本过程为：在数据写请求发生时，首先查找逻辑地址到物理地址的映射表。如果没有查到映射信息，说明该逻辑地址第 1 次被写，需要对其分配存储块，此时从空闲物理块池中获取一块存储空间，建立逻辑地址到物理地址的映射。

相比资源预留静态映射技术，动态映射机制有如下优点：

(1)资源利用率高，减少系统初期投资。用户对空间的需求是随着时间而逐渐增加的，在系统构建初期只需要购置少量存储空间即可以满足需求。通过在线扩容技术，当用户的实际数据需求达到预定阈值时，扩充存储池资源，这样可以减少初期投资，减少资源空闲，提高存储的利用率。

(2)采用动态映射可以按照应用请求的特性进行存储资源的分配和重组。比如，可以根据数据的特性分配不同性能的存储资源，或者按照数据热度进行数据布局重组，为高热度数据分配高性能存储资源，从而提高系统的性能。

3 系统设计与实现

本文在 Linux 2.6.20 平台上实现了 ASD 系统。系统主体部分是一个 Linux 可加载模块逻辑块设备驱动程序，以虚拟块设备的形式提供存储服务。ASD 的核心功能部件包括地址映射管理和物理资源管理 2 个部分。地址动态映射机制负责逻辑地址到物理地址的查找和映射关系的建立，物理资源管理负责物理资源的组织和空闲资源的分配。

3.1 物理资源管理

如图 1 所示，ASD 使用存储池管理物理存储资源。每个存储池是一个相对独立的存储管理单元，具有资源分配策略、存储块映射粒度等不同属性。分配策略有线性、条带等不同方式；存储块粒度在 2^{12}~2^{30} 之间可选，这些属性可在创建存储池时按照应用的要求设置。

当物理设备加入 ASD 系统中时，按照存储块粒度进行切分，加入到某个存储池中。最终多个存储池构成了整个 ASD 系统的存储资源空间。

ASD 系统使用 B+树结构管理空闲物理资源。每一个存储池的空闲物理资源构成一棵 B+树，每个叶子为一个存储块，以存储池 ID 和存储块物理地址为键值。多个存储池空闲资源 B+树组成的链表构成整个存储系统的空闲存储资源。

3.2 地址动态映射机制

动态映射机制在实际数据写发生时分配空闲物理块，建立逻辑地址到物理地址的映射。动态映射因为地址映射的不确定性而不能采用公式化的函数地址映射，只能通过映射表的方式管理映射信息。

映射表使用了与空闲物理资源管理相同的 B+树结构。每一个虚拟设备维护一棵映射表树，每个叶子为一项映射信息，以虚拟设备 ID 和设备内逻辑地址为键值。映射表项由键值、物理设备 ID、物理地址等信息组成。

逻辑设备读写流程如下：

(1)按照 VD 的 ID 和逻辑地址查找映射表树。

(2)如果查到映射表项，则按照物理地址转发当前请求。

(3)如果未查到映射表项，对于读请求，直接返回 0 数据；对于写请求，则分配空闲存储块，建立映射表项并插入映射表树中。

(4)按照物理地址转发当前请求。

(5)返回。

3.3 映射表元数据

映射表信息作为元数据持久保存，有 2 种方式：正向映射表和反向映射表。正向映射表依据逻辑地址构建映射表元数据，需要保存每一个逻辑地址是否映射、映射的物理地址内容等信息。反向映射表则根据物理资源的使用情况保存映射元数据。

因为要保存 64 位逻辑空间地址的映射信息，正向映射表需要较大的存储空间，开销较大；用户对逻辑空间的访问具有局部性的，不必在运行初期就保存全部逻辑地址映射信息，只需保存实际映射过的逻辑地址的映射信息即可；逻辑空间是由多个物理设备组成的，实际中物理设备空间远远小于逻辑空间，所以，ASD 系统采用反向映射表保存映射元数据。

反向映射表的基本原理是：每一个物理设备保存自己的存储块的映射信息。在 ASD 的实现中，反向映射表的每一项由使用标记、UUID、逻辑设备 ID 和逻辑地址 4 项组成，需要 16 Byte，当映射粒度为 1 MB 时，2 TB 的物理设备需要

$$2\ \text{TB} \div 1\ \text{MB} \times 16\ \text{Byte} = 32\ \text{MB}$$

的物理空间保存映射元数据。

为了便于动态扩充物理存储空间，ASD 系统将物理空间按 64 GB 粒度进行分段，每段的前部存放该段的映射元数据。

映射表的查找、插入等操作位于 I/O 关键路径上，因此，映射表必须常驻内存。在 ASD 系统启动时扫描物理设备，首先使用系统元数据构建虚拟设备，然后使用映射元数据构建虚拟设备的正向映射表。

构建算法如下：

(1)读取系统元数据，构建虚拟设备和存储池等系统结构。

(2)扫描物理设备上每个 64 GB 段的映射元数据区，读取段内每个存储块的映射信息。

(3)如果该存储块已被映射，根据虚拟设备 ID 查找该块所属的虚拟设备，构建正向映射表项，将其插入映射表树中。

(4)如果该存储块未被使用，则将其构建到该池的空闲资源树中。

4 测试及分析

首先测试和分析以 ASD 系统为基础的 ext2 文件系统的性能和资源利用情况，其次评估动态映射机制引起的时空开销。测试环境为 Intel PentiumD 2.8 GHz、2 GB 内存、3 块 250 SATA 磁盘，操作系统为 Linux2.6.20.4。

4.1 文件系统应用测试

ext2 文件系统采用分组管理存储空间的方法[3]。如图 2 所示，将整个存储空间按照 128 MB 粒度划分为多个组，每个组头部存放组描述符等元数据，占用约 3 MB 的磁盘空间。

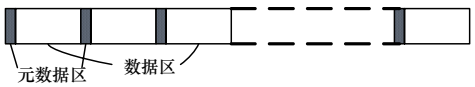


图 2 静态映射方式 ext2 文件系统数据分布

(1)资源利用率分析

依据 ext2 文件系统管理存储空间的方法，文件系统的元数据占用空间可以按照如下公式计算：

$$M = \text{文件系统大小} \div \text{组大小} \times \text{每组元数据大小}$$

在资源预留方式中，文件系统的逻辑空间必须不大于物理存储空间。写时分配方式只需要对实际发生的写请求分配存储空间，因此，只要有能够存放元数据的物理空间就可以创建文件系统。以 1 TB 的 ext2 文件系统为例，假设应用在一段时期内只能产生 200 GB 数据，那么只需要 200 GB+24 GB 的物理空间即可满足运行要求，节省初期 800 GB 物理

空间的资金投入，同时减少了空闲资源占用。

(2)元数据分布对系统性能的影响

在静态映射方式中，元数据与用户数据是混合存放的，分布在整个存储空间，无法针对元数据特性分配存储资源。使用动态映射机制时则可以按照数据特性为元数据和用户数据分配不同类型的存储资源。根据这个特点，本文设计了元数据与用户数据分离存放的布局方案，目的是利用快速设备提升元数据操作的性能，进而提升文件系统的整体性能。分离布局方案如下：元数据存放在 DRAM-SSD 等随机设备中，用户数据仍采用传统的磁盘存储。

动态映射中元数据可以按照需要分布在任何存储位置上，如存储空间的中间、前部或尾部。考虑到文件系统的扩展性和元数据的可分离性，选择将元数据存放在地址空间前部。图 3 显示了以 ASD 为平台的 ext2 文件系统的数据布局，ASD 采用了线性存储资源分配方式。

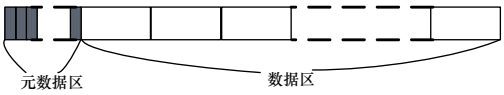
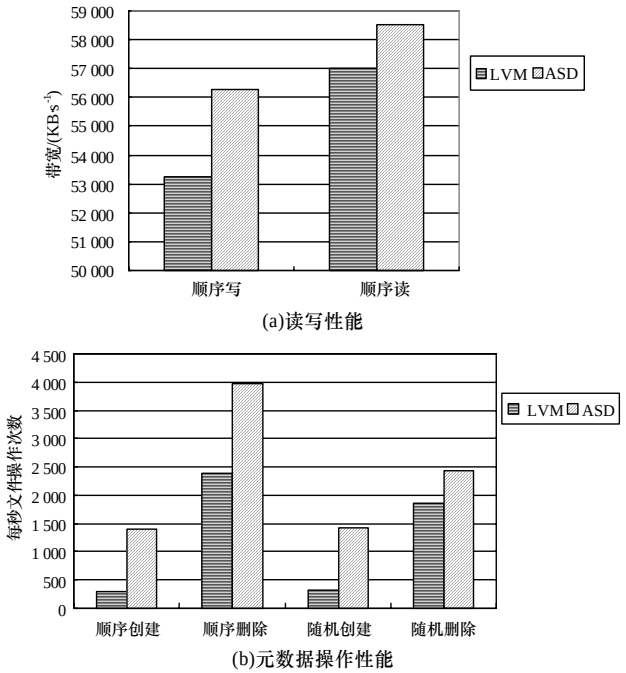


图 3　动态映射方式 ext2 文件系统数据分布

将 ASD 与 LVM 进行了对比测试，元数据区使用 RAM 盘模拟 DRAM-SSD 设备，用户数据区使用与 LVM 相同的磁盘存储。映射粒度为 1 MB，测试工具为 bonnie++ 1.03c[4]。

图 4 表明，元数据分离方案的 ext2 文件系统的性能有显著提升，尤其是元数据操作性能，最高达到 363.8%的提升。如此显著的性能提升一方面与元数据分离方案有关，另一方面与使用内存模拟 SSD 有很大关系。在实际中 DRAM-SSD 的性能远低于内存，但是依照比例计算，即使 DRAM-SSD 只有内存速度的 1/10，应用性能的提升也将达到 30%以上。



(a)读写性能

(b)元数据操作性能

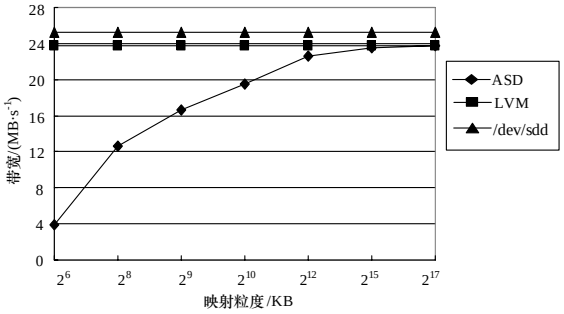
图 4　2 种布局方案文件系统性能对比

4.2　系统开销分析

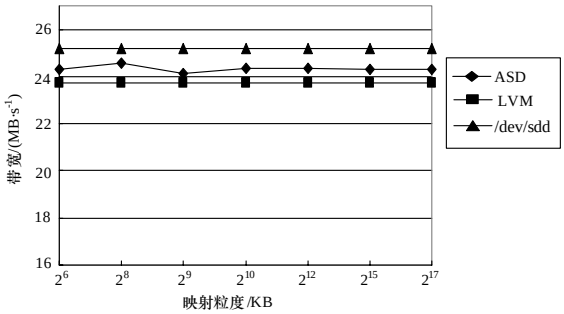
在动态映射方式中，映射过程需要查找映射表和保存映射元数据，地址映射时间受映射表操作效率的影响。映射粒度的大小决定了映射表项的数目和存储资源分配的灵活性，较小的映射粒度有较灵活的资源分配粒度，但是会导致较多的映射表查找和映射元数据保存等操作，较大的映射粒度则相反。本测试对不同映射粒度大小的 ASD 系统进行性能测试，分析映射粒度对性能的影响。

首先使用物理设备/dev/sdd 进行基线测试，然后对 ASD 和 LVM 使用相同的物理设备进行测试。此外，顺序读写每次都会引起映射表查找和分配操作，对性能影响最大，因此，只测试顺序读写这种极端情况下的性能损失。测试工具为 Iometer[5]，读写数据量为 5 GB。

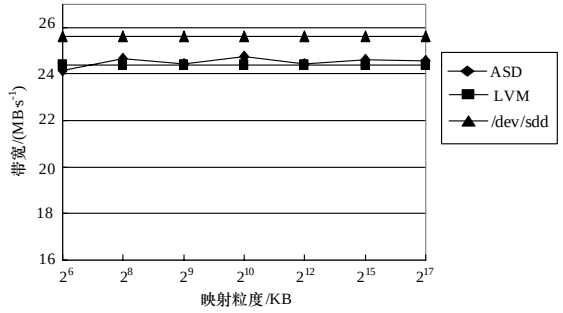
新写会引起物理资源分配、映射建立和映射元数据保存等额外操作，时间开销较大。而读和再次写只有查找开销，开销较小。因此，分别测试了 ASD 中新写、再次写和读的性能，分析地址映射过程中各阶段开销对性能的影响。测试结果如图 5 所示。由图 5(a)可以看到，对于新写操作，ASD 映射粒度的大小对性能有显著影响。较小的映射粒度对性能影响较大，随着映射粒度的增加，ASD 设备的性能逐渐提升。当映射粒度大于 4 MB 后，ASD 的性能与 LVM 基本相同。从图 5(b)、图 5(c)可以看到，ASD 设备再次写和读操作的性能没有损失。测试结果表明，影响性能的主要因素是映射的建立和映射信息的保存等开销，映射表的查找开销对性能没有影响。



(a)新写带宽



(b)再次写带宽



(c)读带宽

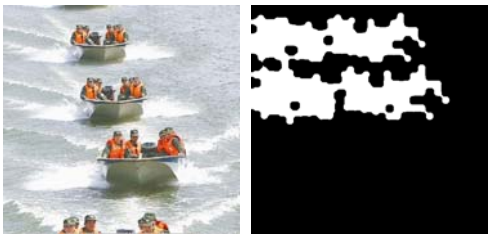
图 5　不同映射粒度 ASD 性能测试

(下转第 22 页)

4 实验结果

本文算法共有 5 个参数，即 B, T_{ind}，r，T_s 和 T_{sift}。在实验中，选择 $B=7$，$T_{ind}=400$，$r=8$，$T_s=1.8$ 及 $T_{sift}=0.4$。算法检测效果和耗费时间依赖于图像的内容，细节丰富的图像有较多的 SIFT 特征点，需花费更多的匹配时间，能得到更好的检测效果。

第 1 个实验的目的是验证算法的检测能力。为说明算法在实际中的应用，以一副新闻图片作为待检测图片，图 2 显示了 2007 年 5 月 25 日《安徽日报》头版题为《武警苦练船艇操作技能》的新闻配图和其检测结果。待检测图片中最远一排的冲锋舟及水波部分和其前排的内容相同但大小不同，猜测是经过篡改并特意伪造出“近大远小”的三维透视效果，检测结果证实了这种猜测。



(a)待检测图片　　(b)检测结果

图 2　一幅新闻图片和检测结果

第 2 个实验考虑篡改图像经过润色操作和 JPEG 压缩的情形。原图和篡改图片如图 3 所示。

(a)原图　　(b)篡改图像

图 3　原始图像和篡改图像

原图像中一部分草地区域被复制并旋转 10°以 120%比例被拉伸后粘贴到人所在的位置。把篡改图像加入不同数量的高斯白噪声，进行不同程度的模糊，以不同的质量因子 JPEG 压缩，并将多种润色操作组合，其中，加噪和模糊使用 Photoshop 完成，JPEG 压缩使用 Matlab 的 imwrite 函数完成。在加入数量多达 2.5%的高斯噪声或在 JPEG 压缩质量因子不低于 70 的情况下，检测算法都能得到正确结果。因为 SIFT 特征点检测方法对模糊操作不够稳健，所以算法对较大程度模糊的抵抗能力较差。在多种润色方法组合操作后，算法仍然能得到令人满意的检测结果。

5 结束语

拷贝-变换-移动篡改是比拷贝-移动篡改更为常见也更加有效的篡改技术，原有的拷贝-移动篡改是拷贝-变换-移动篡改的特殊情形，且针对拷贝-移动篡改的检测框架无法检测出拷贝-变换-移动篡改。本文研究了拷贝-变换-移动篡改模型并提出了基于 SIFT 特征点的检测算法。算法有以下 3 个特点：(1)利用同一幅图像中互相匹配的 SIFT 特征点对标明篡改存在。(2)从“种子点”出发，逐步生长出篡改区域。(3)利用 SIFT 点的特征避开光滑区域的干扰，误判区域少。本文算法对常见的润色操作和 JPEG 压缩也有较强的稳健性。

需要指出的是，如能确定篡改过程中仅有尺度变化，则可将本文算法中的圆形邻域改为方块形邻域，此时不需要在圆形区域内插值，计算复杂度低，计算速度快得多。下一步工作是研究怎样更加充分地利用彩色图像 3 个通道的信息使算法的检测结果更精细。

参考文献

[1] Fridrich J, Soukal D, Lukáš J. Detection of Copy-move Forgery in Digital Images[C]//Proceedings of Digital Forensic Research Workshop. Cleveland, USA: [s. n.], 2003: 5-8.

[2] Popescu A C, Farid H. Exposing Digital Forgeries by Detecting Duplicated Image Regions[D]. Hanover, UK: Department of Computer Science, Dartmouth College, 2004.

[3] Ju Shenggen, Zhou Jiliu, He Kun. An Authentication Method for Copy Areas of Images[C]//Proceedings of the 4th International Conference on Image and Graphics. Chengdu, China: [s. n.], 2007: 303-306.

[4] Lowe D. Distinctive Image Features from Scale-invariant Keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 91-110.

[5] Ng T T, Chang S F, Hsu J, et al. Columbia Photographic Images and Photorealistic Computer Graphics Dataset[D]. NY, USA: Columbia University, 2005.

编辑　索书志

(上接第 19 页)

实际应用中可以根据应用的特点选择合适的映射粒度。对于写多读少的应用，可以采用较大的映射粒度；对于读写混合且再次写占一定比例的应用，可以选择较小的映射粒度，这些措施可以有效降低映射表操作开销对性能的影响。

5 结束语

本文设计和实现了一种按照应用特性、基于动态映射按需分配机制管理存储资源的虚拟化存储系统。测试结果表明，动态机制是一种改善存储利用率和提升存储系统性能的有效方法。

在未来研究中准备引入应用访问特性在线识别技术，通过区分 I/O 请求数据特性和应用动态行为，指导数据的适应性分配和存储布局的动态重组。

参考文献

[1] Ruemmler C, Wilkes J. Unix Disk Access Patterns[C]//Proc. of USENIX Winter Technical Conference. San Diego, USA: [s. n.], 1993.

[2] Teigland D, Mauelshagen H. Volume Managers in Linux[C]//Proc. of USENIX Annual Technical Conference. Boston, USA: [s. n.], 2001.

[3] Poirier D. The 2nd Extended File System[Z]. [2008-07-05]. http://www.nongnu.org/ext2-doc/ext2.html.

[4] Russell Coker. Bonnie++[Z]. [2008-07-05]. http://www.coker.com.au/bonnie++/.

[5] Dan Bar Dov. Iometer[Z]. [2008-07-05]. http://www.iometer.org/.

编辑　张　帆