

# 一种基于 LVM 和软件 RAID5 的文件存储改进方案

林斌斌, 叶荣华

(浙江师范大学 数理与信息工程学院, 浙江 金华 321004)

**摘要:** 存储虚拟化技术在方便服务器存储管理的同时存在一些弊端, 将所有文件存于单个逻辑卷中可能会造成磁盘空间浪费或影响大文件的读写性能, 将文件分类存储于多个逻辑卷会涉及存储空间划分, 由于逻辑卷不支持在线缩小文件系统, 为某个逻辑卷分配过多空间亦会造成磁盘的浪费。针对上述情况, 提出了一种基于 LVM 和软件 RAID5 的服务器文件存储改进方案, 并设计了相应的算法。算法可对文件进行分类存储于多个逻辑卷, 加入一个共享逻辑卷, 若文件写入的逻辑卷满足约定使用率条件则将文件迁移至共享逻辑卷中创建或写入, 避免过度预分配造成的浪费。对方案的实现进行了仿真实验, 结果表明: 分类存储可以提高大文件的读性能和存储利用率, 方案在文件迁移情况下对小文件写操作的响应时间影响较小。

**关键词:** 软件 RAID5; LVM; 存储; 算法

中图分类号: TP301

文献标识码: A

文章编号: 1673-629X(2015)05-0099-05

doi: 10.3969/j.issn.1673-629X.2015.05.024

## An Improved Scheme of Server File Storage Based on LVM and Software RAID5

LIN Bin-bin, YE Rong-hua

(College of Mathematics Physics and Information Engineering, Zhejiang Normal University,  
Jinhua 321004, China)

**Abstract:** Storage virtualization technology offers a convenient management for server storage, but there are also some drawbacks at the same time. When all files are stored in a single logical volume, it may cause the waste of disk space or affect the read and write performance of large files. When the files are stored on multiple logical volume according to their types, a new problem can be the division of storage space, allocating too much space for a logical volume will result in a waste of disk space because logical volume doesn't support shrinking file system online. For the above reasons, design an improved scheme for server storage and a corresponding algorithm for the scheme based on software RAID5 and LVM technology. The algorithm can classify files and make them stored in multiple logical volumes, a shared logical volume is added for the rest of the logical volumes, if the logical volume which a file is written to meets the appointed usage rate threshold condition, the file will be moved to the shared logical volume to create or be written, avoiding the waste caused by excessive pre-allocation for a single logical volume. The experimental results show that classified storage for files can improve the reading performance of large files and utilization of storage, this program has a less effect to the response time of small file's writing when the file is in migration situation.

**Key words:** software RAID5; LVM; storage; algorithm

## 0 引言

以往服务器在考虑到存储系统的可扩展性时会使用存储虚拟化<sup>[1-3]</sup>技术, 将整个存储网络虚拟化为一个存储池, 再在存储池中规划逻辑卷使用, 这种做法的

优点是可以统一管理存储, 其对应的缺点是单个逻辑卷不能跨文件系统。不分种类的对文件在单个逻辑卷上存储可能会造成磁盘空间的浪费, 由于逻辑卷并不支持在线缩小文件系统, 为某个逻辑卷分配了过多的

收稿日期: 2014-06-21

修回日期: 2014-09-23

网络出版时间: 2015-04-22

基金项目: 国家自然科学基金资助项目(61272007); 浙江省自然科学基金(Y1110483)

作者简介: 林斌斌(1990-), 男, 硕士研究生, 研究方向为计算机应用; 叶荣华, 教授, 研究方向为 Web 服务。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20150422.0950.003.html>

空间亦会造成磁盘空间的浪费。一个存储池上划出的逻辑卷所对应的具体物理磁盘亦不确定,这样就不能针对性地对提高某类文件的读写速率配置对应的磁盘。为了通过提高磁盘的利用率来降低服务器的资金投入,也为了提升不同类型文件的读写性能,文中提出了一种基于软件 RAID5<sup>[4-5]</sup> 和 LVM 技术<sup>[5-6]</sup> 的服务器文件存储的改进方案。该方案的特点:

- (1) 将文件分类存储于文件系统参数不同的逻辑卷提高磁盘的利用率;
- (2) 可为存储某类文件的逻辑卷配置优化文件读写的硬盘,增强系统 I/O 性能;

(3) 增加一个共享逻辑卷,在其余逻辑卷使用率过高时分担负载,避免过度预分配造成的磁盘浪费。

## 1 服务器存储系统的方案结构

### 1.1 方案的存储系统结构

该方案基于 LVM 和软件 RAID5 技术,系统由服务器主机、IP SAN<sup>[7-8]</sup> 交换机以及多个 RAID5 阵列组成。LVM 架构图如图 1 所示,方案结构图如图 2 所示。其中服务器的网络端口与交换机连接,交换机连接多个已装入磁盘柜的磁盘阵列。

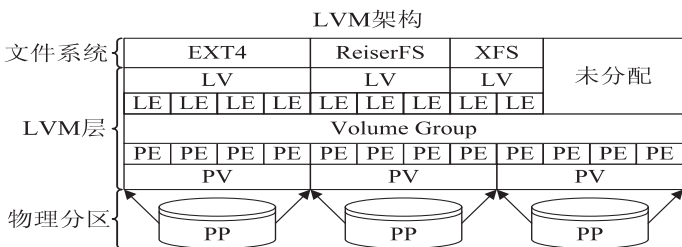


图 1 LVM 架构图

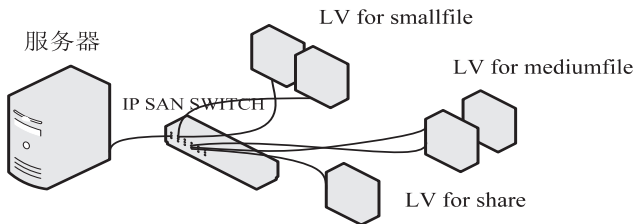


图 2 服务器存储方案结构图

对系统中的这些磁盘阵列进行分组,假设三组,选取其中两组分别存放小文件和大文件,剩下的一组用于共享。在各组磁盘阵列上建立 LV (逻辑卷) 并进行格式化和目录挂载,步骤如下:

- (1) 将单个组的所有磁盘阵列调成 PV (物理卷);
- (2) 在每个 PV 上创建 VG,注:在建立的过程中需指定 PE 的大小,在 LVM1 版本中,LVM 默认使用 4 MB 的 PE 块,而 LVM 的 VG 最多仅能含有 65 534 个 PE;
- (3) 从 VG 中切割出 LV (此方案中不取 VG 的一部分作为 LV,而是将每个 VG 全部转化为一个 LV,为了 VG 中划分的 LV 和具体的物理硬盘产生对应关系,从而能为提高某类文件的读写性能配置不同参数的硬盘);
- (4) 格式化 LV,并挂载使用,对于存储小文件的 LV,其文件系统的 blocksize 可为 1 kB,这样可以减少硬盘空间的浪费;若 LV 用于存储稍大的文件,其对应的文件系统 blocksize 设为 4 kB,使用较大的 blocksize 可提高 I/O 性能。

### 1.2 存储系统的扩展方法

构建软件 RAID5 并扩展 LV 的方法步骤:

- (1) 将已放置硬盘的磁盘柜接入交换机,参照磁

盘柜的说明书配置磁盘柜网卡参数,并将其设置为 iSCSI<sup>[8-9]</sup> Target,使其作为本地硬盘来使用;

- (2) 使用命令 `#mdadm --create --auto=yes /dev/md7 --level=5 --raid-devices=4 --spare-devices=1 /dev/sdc{6,7,8,9}` 将磁盘柜中的硬盘组成一个软件 RAID5 阵列 (假设用了 4 个盘,且硬盘显示为 `/dev/sdc`<sup>[6-8]</sup>),阵列的构建需要时间,可等数分钟后使用命令 `#mdadm --detail /dev/md7` 查阅阵列的详细信息;

- (3) 通过命令 `#pvcreate /dev/md7` 将阵列调成 PV,接着用命令 `#vgextend smallfilevg /dev/md7` 将 PV 加入 VG (smallfilevg 是 VG 名);

- (4) 使用 `vgdisplay` 可查看 VG 下的 Free PE 数量,假设显示为 8 000,使用命令 `#lvresize -l +8 000 /dev/smallfilevg/smallfilelv` 增加 LV 的容量;

- (5) 用命令 `#resize2fs /dev/smallfilevg/smallfilelv` 将 LV 的容量扩充至整个文件系统。

存储系统的 VG 均可独立进行在线扩展,当存储系统中的某个 LV 需要增大容量时,扩展步骤如下:

- (1) 将硬盘装入 LV 对应的阵列柜;
- (2) 在 LV 对应的磁盘阵列中选一个做扩容,设新增的硬盘为 `/dev/sdf`,而阵列为 `/dev/md0`,将硬盘加入

这个阵列的命令为#mdadm /dev/md0 --add /dev/sdf,再将阵列设备由原先的 3 个(假设为 3 个)增加到 4 个,命令为#mdadm --grow /dev/md0 --raid-devices = 4,使阵列容量得到扩充;

(3)在完成上述步骤的情况下,因为文件系统是建立在 LVM 上的,还需要改变 LV 的大小,用来包含新添的硬盘,先使用命令#pvresize /dev/md0 将阵列剩余的空间全分配给物理卷,再使用命令#lvresize -l + 4 000 /dev/smallfilevg/smallfilelv 调整逻辑卷的大小(PE 大小为 128 MB,刚添加的硬盘容量为 500 GB),然后通过命令#resize2fs /dev/smallfilevg/smallfilelv 调整 LV 上文件系统的大小。

2 存储系统的文件分类及迁移算法

2.1 算法规则的设定

针对上一部分提出的系统结构设计了对应的算法,算法的相关约定如下:

(1)在 LV 上创建新文件时其当前使用率需低于 75%,当 LV 的使用率超过 75%,转到共享 LV 上建新文件,创建完后在原 LV 中创建一个指向共享 LV 中文件的链接文件,通过这种策略降低原 LV 上磁盘阵列的 I/O 负载。考虑到随着时间的推移,用户对 LV 中已存在文件的写入和更新会使文件增大,而这些文件的总存储容量不能超过 LV 容量,否则会造成服务中止。所以规定进程向目录中的已存在文件写入时该文件所处 LV 的使用率应小于 95%,若 LV 的使用率已大于 95%,需要将正在写入的文件先迁移到共享 LV 上,再对文件写入,并在写完成后在原 LV 上创建一个指向共享 LV 中文件的链接文件。两个阈值可根据 LV 上存储文件的类型设定不同值。

(2)另外系统维护一份内容为各个文件的排他锁(X 锁)状态的文件,遵守一级封锁协议<sup>[10]</sup>。进程在修改文件数据之前,先对文件加 X 锁,直到修改完后才释放,可防止进程在修改文件时有其他进程同时对文件写入,从而丢失此次更新。进程对文件仅做读操作时不加锁。一级封锁协议不可防止读‘脏’数据和不可重复读问题,但可以防止丢失更新。

2.2 算法的操作流程

进程对文件的操作分为读操作和写操作,需要将实现算法逻辑的程序嵌入这些操作调用的组件(组件的设计思想:若服务器通过网络提供文件存储服务,可将算法逻辑嵌入 J2EE<sup>[11-12]</sup>组件,用一份 XML<sup>[13]</sup>配置文件保存文件类型与目录之间的映射关系以及目录与使用率阈值的映射关系,组件通过 JDOM 技术读取配置文件中的信息,并将读到的信息转化存储于两个 HashMap 中,在第一个 HashMap 中 key 为文件类型,value 为文件目录,在第二个 HashMap 中 key 为目录,value 为使用率阈值。用户上传文件时会调用 J2EE 组件,组件判断文件类型,根据第一个 HashMap 得到其对应的目录,根据目录可以从第二个 HashMap 中得到使用率阈值,经过算法逻辑的处理后,文件被存储于对应的目录中)。当进程对文件进行读操作时,大致的流程如下:

- (1)首先根据目标文件的扩展名(如.txt,.jpeg等)确定其所在的目录;
- (2)判断目录下该文件名对应的文件是否存在;
- (3)找到文件并读出。

算法的写操作部分较复杂,算法的写操作流程如图 3 所示,写操作伪代码如图 4 所示。

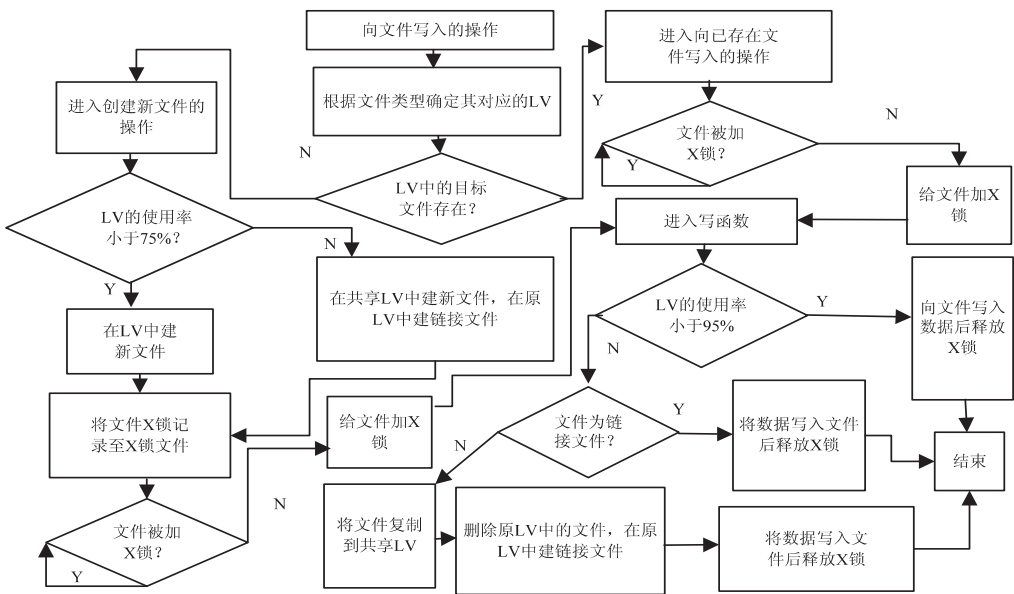


图3 算法的写操作流程

```
#judgefile(),用于判断文件类型
Function judgefile(){
If[文件为小文件类型];Then
    smallwrite(); //调用smallwrite()
Eli[文件为大文件类型];Then
    Mediumwrite();
Else Otherwrite();
Fi }
# write(),写入数据和文件迁移
Function write(){
If[LV的使用率<95%];Then
    将数据写入文件;FKEY=1;
    释放文件的X锁;
Else //使用率>95%
If[文件为链接文件];then
    将数据写入文件;
    FKEY=1; //在写完后置1
    释放文件X锁;
Else //文件非链接文件
    在共享LV中建同名文件;将文件复制到共享LV;移除原LV中文件;在原LV建链接文件;FKEY=1;释放文件X锁;
Fi
Fi }
```

```
# smallwrite(),用于新建文件
Function smallwrite(){
If[目标文件存在];Then
Until [SFKEY==1] //FKEY代表
Do
    //完成状态
    While[X锁未锁&&SFKEY!=1]
    Do
        文件加X锁;write();//调用write()
    Done
Done
Else //新建文件的操作
If[LV的使用率<75%];Then
    在LV中新建文件;
Else
    在共享LV中新建文件;
    在LV中建链接文件;
Fi
    记录文件的X锁;
Until[SFKEY ==1]
Do
    While[X锁未锁&&SFKEY!=1]
    Do
        给文件加X锁;write();
    Done
Done
Fi }
```

图 4 写操作伪代码

当 LV 使用率过高时可对其进行扩展,扩展后可对属于该 LV 但存在于共享 LV 上的数据进行回迁,流程如下:

- (1)查找原 LV 目录下的所有链接文件记录,写入一个文本文件备用;
- (2)删除原 LV 中某个链接文件,复制共享 LV 中的对应文件至原 LV,在复制前对文件加 X 锁;
- (3)复制完后,释放文件 X 锁,删除共享 LV 的对应文件;
- (4)重复步骤 2,3,直到回迁任务完成。

3 仿真实验及分析

3.1 实验配置

实验设备:惠普 ProBook 4 416s,320 G 的硬盘 1 (4 416s 自带,5 400 转),usb3.0 扩展卡,Seagate 500 G 移动硬盘 2(5 400 转,支持 usb3.0)。

实验步骤如下:

- (1)电脑已装 CentOS 6.3 操作系统,在硬盘 2 上,通过命令#fdisk /dev/sdb 再按菜单提示建 9 个 20 G 的

分区,每三个分区建一个软件 RAID5 阵列。建好的阵列分别是/dev/md1,/dev/md2,/dev/md3。md1 用于存放小文件,md3 用于存放稍大文件,md2 用于共享。将每个阵列转化为 PV,加入 VG,建 LV,再挂载于不同的目录,通过仿真方式建立三个基于软件 RAID5 和 LVM 的逻辑卷,这三个逻辑卷分别为 smallfilelv,meidiumfilelv,emergencylv。在硬盘 1 文件系统下的/project 目录内保存一份记录文件 X 锁的文件 xkey.txt。

- (2)使用 shell script 语言编写程序,设置不同的环境进行数据测量,仿真实验主要验证算法的可行性,以及算法实现后的效果。

3.2 读操作实验

实验 1:选取 mediumfilelv 这个 LV 用于测试不同大小文件在不同 blocksize 的 ext4 文件系统的读速率,在文件系统格式化后将 LV 挂载于/mediumfile 这个目录,在目录中建立大小分别为 1 kB,1 MB,10 MB,50 MB 的文件,再通过读程序对 LV 中的文件进行读取测量读速率,读程序可根据文件类型判断文件所在的 LV,在判断文件存在后读取文件。测量数据如表 1 所示。

其中如 77.8 kB/s 为多次测量的平均值。在实验中,在不同 blocksize 情况下读 1 kB 小文件测得的读速率差异不明显,且在相同 blocksize 的情况下,其读速率也不稳定,如在 blocksize 为 1 kB 的情况下,其读速率在 20 kB/s 和 200 kB/s 之间跳动。当文件大小为 1 MB~50 MB 时,其随文件系统 blocksize 的增大,读速率有较明显的提高。分析:将小文件存储于较大 blocksize 的文件系统对其读速率提高不明显,但会造成硬盘空间的浪费,将小文件存储于文件系统 blocksize 为 1 kB 的 LV 中可节约硬盘空间,并可为 LV 对应的磁盘阵列配置平均寻道时间更短的硬盘来提高小文件读速率。将大文件存储于文件系统 blocksize 为 4 kB 的 LV 中对比文件系统 blocksize 为 1 kB 的 LV 中,其读速率有明显提高。

表 1 实验 1 测量结果

blocksize	被读文件大小			
	1 kB	1 MB	10 MB	50 MB
1 kB	77.8 kB/s	35.7 MB/s	34.4 MB/s	39.2 MB/s
2 kB	107.5 kB/s	34.6 MB/s	43.9 MB/s	44.6 MB/s
4 kB	74.3 kB/s	37.8 MB/s	43.4 MB/s	45.7 MB/s

3.3 写操作实验

实验 2:选取 smallfilelv 存储文件,emergencylv 用于共享。在测试过程中两个 LV 的文件系统 blocksize 均为 1 kB。模拟两种环境,在环境 1 下,smallfilelv 的

使用率低于 75%,建新文件和向已存在文件的写入都不发生文件的迁移;在环境 2 下,smallfilelv 的使用率大于 95%,建新文件和向已存在文件写入都会发生文件的迁移。模拟环境 1 测量:首先将 smallfilelv 的使用



率控制在 75% 以内,在 smallfilelv 中建大小分别为 1 kB,1MB,5MB,20MB 的四个文件,用写程序向这些文件写入 1 kB,1MB,5 MB 的数据测量每个写操作完成

的时间。模拟环境 2 测量:执行与环境 1 时相同的写入操作,测量写操作完成所需的时间。两种环境下测得的数据如表 2 所示。

表 2 实验 2 测量结果

datesize	被写文件大小			
	1 kB	1 MB	5 MB	20 MB
1 kB	0.66 s/0.83 s	0.44 s/0.83 s	0.54 s/1.50 s	0.51 s/4.57 s
1 MB	0.54 s/0.70 s	0.58 s/1.04 s	0.56 s/1.73 s	0.51 s/4.71 s
5 MB	1.25 s/1.52 s	1.38 s/1.70 s	1.32 s/2.27 s	1.19 s/5.33 s

表中数据如 0.66 s/0.83 s,0.66 s 和 0.83 s 分别代表在环境 1 和环境 2 下写操作完成的时间。分析:环境 2 中的写操作比环境 1 中的多了文件迁移操作,两者间的差值可视为文件的迁移时间,较大文件的迁移时间较长。由于整个被写文件的迁移只发生在 LV 使用率高于 95% 的情况下,所以当 LV 使用率接近 95% 或很高时需对其进行扩展。

4 结束语

文中设计了一种基于软件 RAID5、LVM 技术的服务器文件存储的改进方案,为方案设计了相应的算法,并给出了组件的设计思想。用户可个性化地定制文件类型与存储目录的对应关系以及目录与使用率阈值的映射关系,通过对不同文件的分类存储提高硬盘的利用率,可针对不同文件的读写配置相应的硬盘提高文件的 I/O 性能。增加一个共享 LV,在其余 LV 使用率满足约定阈值时,可将正在进行写操作的文件迁移。共享 LV 可避免为其余 LV 过度预分配造成磁盘的浪费,当某个 LV 上的文件有迁移行为时,表明 LV 需要扩展,将硬盘按需分配。对方案的实现进行了仿真实验和分析,实验结果显示文件分类存储可提高稍大文件的读性能,方案在文件迁移情况下对小文件写操作的响应时间影响较小。

参考文献:

[1] Clark T. Storage virtualization:technologies for simplifying da-

ta storage and management [ M ]. [ s. l. ] : Addison – Wesley Professional,2005.

[2] 葛苏慧,梁宏涛,房正华. 高校共享数据中心虚拟化技术的架构[J]. 计算机技术与发展,2014,24(4) :174–177.

[3] 张冬晖,万晓冬,李育龄. 基于光盘库管理系统的虚拟化技术研究[J]. 计算机技术与发展,2013,23(6) :12–14.

[4] Perumal S,Kritzinger P. A tutorial on RAID storage systems [ R/OL ]. 2004–05–06. [http://pubs.cs.uct.ac.za/archive/00000131/01/perumal2004\\_RAIDTutorial.pdf](http://pubs.cs.uct.ac.za/archive/00000131/01/perumal2004_RAIDTutorial.pdf).

[5] 鸟哥,王世江. 鸟哥的 Linux 私房菜[M]. 北京:人民邮电出版社,2010:463–489.

[6] Kiwi K H. 逻辑卷管理[EB/OL]. 2007. <http://www.ibm.com/developerworks/cn/linux/1-lvm2/>.

[7] Clark T. 存储区域网络设计[M]. 邓劲生,李宝峰,李蕾,译. 北京:电子工业出版社,2005:94–129.

[8] Lu Yingping,Du D H C. Performance study of iSCSI-based storage subsystems [ J ]. IEEE Communications Magazine, 2003,41(8) :76–82.

[9] 彭聪,陕振,张淑萍. iSCSI Target 研究与性能测试[J]. 计算机工程与设计,2010,31(4) :889–892.

[10] 李月军. 数据库原理与设计[M]. Oracle 版. 北京:清华大学出版社,2012:158–175.

[11] 周平. JavaEE 大学教程[M]. 北京:清华大学出版社,2012.

[12] 李刚. 轻量级 JavaEE 企业应用实战[M]. 北京:电子工业出版社,2014.

[13] Duckett J. Web 编程入门经典[M]. 杜静,敖富江,译. 北京:清华大学出版社,2011.