第 34 卷 第 5 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.5 Computer Engineering March 2008
·博士论文· 文章编号：1000—3428(2008)05—0033—03 文献标识码：A 中图分类号：TP393

用于网络存储系统的存储空间动态分配方法

韦 理，周悦芝，夏 楠
(清华大学计算机科学与技术系普适计算教育部重点实验室，北京 100084)

摘 要：提出一种在网络存储系统中动态分配存储空间的方法，可以在保持用户视图的逻辑空间大小不变的情况下，按需分配用户实际所用存储空间。与传统基于逻辑卷管理器的动态分配方法相比，该方法无须修改文件系统元数据，具有更好的可扩展性，已应用于 TransCom 系统中。实践表明，它实现简单、开销较小，可满足实际使用需求。

关键词：网络存储系统；逻辑卷管理器；按需分配

Approach to Allocate Storage Space Dynamically in Network Storage System

WEI Li, ZHOU Yue-zhi, XIA Nan
(Key Laboratory of Pervasive Computing, Education Ministry,
Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

【Abstract】 This paper presents a new approach to allocate storage space in network storage system. It employs an Allocate on Demand(AoD) strategy, which can adjust real storage space used without modifying the logical volume size from a user's perspective. It does not need to reconfigure file system size as in traditional methods if storage space is extended. It is much more flexible and scalable than traditional logical volume manager based methods. Primary results from the application in TransCom system show that the overhead is small and acceptable, it can meet the practical requirement.

【Key words】 network storage system; Logical Volume Manager(LVM); Allocate on Demand(AoD)

网络存储系统，如FC-SAN系统[1]和以iSCSI[2]为代表的IP-SAN系统，可以使网络内的多个用户共享存储资源。它已在政府、企业、学校和研究机构中得到了广泛的部署和应用。

1 背景介绍

逻辑卷管理器(Logical Volume Manager, LVM)[3]方法是网络存储系统管理和分配存储空间的一种常用方法。它的基本原理是将多个驱动器上的存储空间分解成许多基本单元，再将这些存储单元通过一定的映射机制映射成为逻辑卷(Logical Volume, LV)。LV在用户视图下为一具有一定大小的文件系统卷，它所对应的虚拟空间是逻辑空间，通过一定的机制映射到物理的存储空间。当用户对存储空间的需求改变时，可以通过管理员手工增删存储单元改变逻辑空间到存储空间的映射关系。

但是 LVM 对 LV 大小的改变对文件系统并不透明，因此，当 LV 的大小改变后，还须对原文件系统的关键元数据进行修改，使之与扩展后的 LV 相适应。这种将 LVM 和修改文件系统元数据相结合的用户存储空间扩展方法存在以下问题：

(1)现代文件系统越来越复杂，不同文件系统来源于不同的厂商，元数据结构并不一致，资料也未必公开，因此，修改元数据方法的通用性、鲁棒性都很难保证。

(2)由于修改元数据时只能修改保存在磁盘中的元数据，在运行过程中已加载到内存中的元数据很难直接修改，因此，修改过元数据后，往往要重新挂载文件系统才能生效。

(3)LVM 虽然可以调整存储空间的大小，但是依然需要用户(管理员)手工干预，无法按用户需求动态调整存储空间。

针对上述问题，本文提出一种在网络存储系统中动态分配存储空间的方法(Storage Space Allocation on Demand, SSAoD)。

2 SSAoD 的概述与设计

SSAoD 方法的基本思想是在系统初始化时为文件系统分配足够大的逻辑空间，如 1 TB，然后将逻辑空间通过一种动态的映射机制映射到存储空间，也就是在开始时，只分配用于保存文件系统元数据所必需的存储单元，因此，已分配的存储空间远远小于对应的逻辑空间大小(1 TB)。当已分配的存储空间写满时，SSAoD 采用按需分配(Allocate on Demand, AoD)的策略动态为用户增加新的存储单元，来扩充所需存储空间。按需分配的策略具体通过动态扩展(增长)逻辑存储单元到物理存储单元映射表的方法来实现。

与传统基于 LVM 的方法不同，SSAoD 中空闲的逻辑空间并没有相对应的物理存储单元。本方法在动态扩展逻辑存储单元到物理存储单元的映射表过程中，无须修改上层文件系统的元数据，可以透明地支持多种文件系统。由于 SSAoD 方法按照实际使用空间的情况来分配存储空间，因此可以提高系统存储资源的利用率。这样不但能够满足系统用户需求

基金项目：国家“863”计划基金资助项目(2004AA111020, 2004AA114062)
作者简介：韦 理(1980－)，男，博士研究生，主研方向：透明计算，网络计算机；周悦芝，助理研究员、博士；夏 楠，博士研究生
收稿日期：2007-04-06 **E-mail：**knight99@mails.tsinghua.edu.cn

的动态增长，还可以提高系统所能支持的用户数和可扩展性。

SSAoD 方法的总体思想如图 1 所示，在实际系统中存在 3 种类型节点：用户(users)，元数据服务器(metadata server)以及存储结点(storage nodes)。用户是指使用存储空间的机器或实际用户，他们通过存储网络，如光纤网或 IP 网，访问存储结点中的存储单元。存储结点(子系统)是网络中可以访问到的磁盘或者磁盘阵列，这些设备通过特定的网络接口接入存储网络。

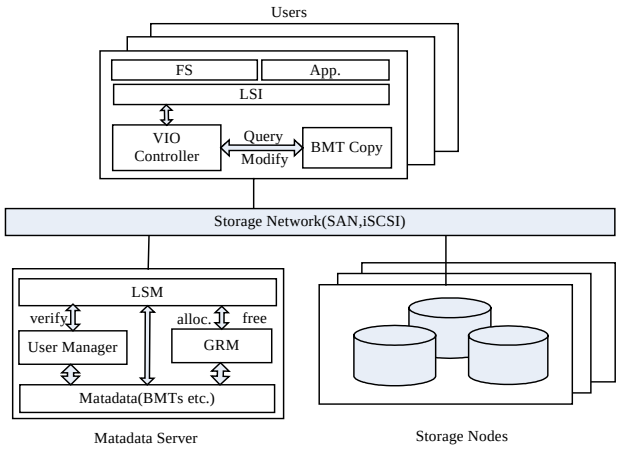


图 1　SSAoD 总体思想

用户机器上运行客户端代码，它的主要功能是处理操作系统或应用程序对逻辑卷的访问请求。它包括 2 个功能组件：LSI(Logic Space Interface)和 VIOC(Virtual IO Controller)。LSI 为用户提供访问逻辑空间的接口，接收用户或者上层应用的 IO 请求，并转送给 VIOC 处理，同时将来自 VIOC 的结果返回给用户或上层应用。VIOC 与元数据服务器和存储结点进行交互，完成用户的请求。为了降低与元数据服务器的交互开销，VIOC 还保存了元数据服务器上块映射表(Block Map Table, BMT)的副本。该数据结构用于将逻辑空间映射到存储空间。VIOC 从 LSI 获得 IO 请求后，首先查询 BMT 副本，按照一定的算法向元数据服务器或者存储结点提交请求。元数据服务器保存了系统中与 SSAoD 相关的主要元数据，是整个系统的管理中心。元数据服务器还包括用户管理器(user manager)、全局资源管理器 (global resource manager, GRM)、逻辑空间管理器 (logical space manager, LSM)这 3 个模块。用户管理器负责用户的授权和认证，维护用户和 LV 的映射关系。全局资源管理器管理系统中存储设备的相关信息，监控各节点的运行状态，维护各存储设备的可用空间，并为用户分配存储空间。LSM 维护每个用户逻辑空间到存储空间的映射表 BMT。

3　AoD 算法

3.1　主要数据结构

系统中主要的数据结构如图 2 所示，用户表(User Table, UT)记录了每个用户的基本信息，每个用户对应一个逻辑卷表(Logical Volume Table, LVT)。LVT 记录了该用户所能访问的 LV，每个 LV 对应一个 BMT。这里块(block)是指一个或多个连续扇区的集合，它是 AoD 的最小分配单位。BMT 维护了 LV 的逻辑块到物理块之间的映射关系。如图 2 所示，地址(LBA)为 100 的逻辑块对应的是设备号为 PID1 的物理存储设备上地址(PBA)为 1000 的物理块。GRM 维护存储设备表(Storage Device Table, SDT)以及空闲空间位图表(Free Space BitMap, FSBM)。SDT 记录了网络中存储设备的基本信息和当前的使用状态，包括总容量和目前可用空间。每个存储设备对应一个 FSBM 表，该表以位图形式记录相关联存储设备上每个存储单元的占用情况，“1”表示已占用，“0”表示未占用。

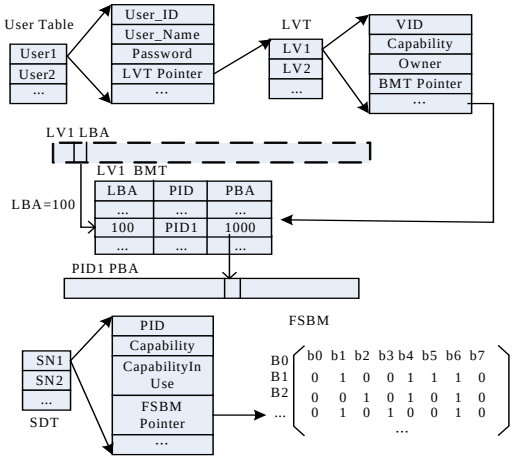


图 2　主要的数据结构及其关系

3.2　AoD 算法

如图 1 所示，用户通过 LSI 访问存储设备，LSI 收到的 IO 请求由 VIOC 进行转发。VIOC 首先查询 BMT 副本，查找所请求逻辑块所对应的设备号和物理块地址，然后向相应存储结点发送对应物理块的读写请求。由于 LV 的实际占用存储空间远小于其逻辑空间，因此存在没有对应实际存储单元的逻辑块。当用户文件系统或应用程序试图写这样的逻辑块时，VIOC 会首先发送分配请求报文到元数据服务器。元数据服务器根据存储设备的占用情况，选择占用率最低的物理设备中的空闲块进行分配，更新 BMT，并返回所分配的物理块所在的存储设备号和物理地址信息。VIOC 收到上述信息后，更新 BMT 副本信息，和元数据服务器上的 BMT 保持一致，然后发送写请求到对应的存储设备，写入对应物理块。

用户数据读写算法和元数据服务器上进行空间动态分配的算法如下所示。

算法 1 用户数据读写

```
void* read(LBA) {
  If (ExistInBMT(LBA)){
    <PID,PBA>=GetPhysicalInfo(Logical_
    Block); //获得存储设备 ID 和地址
    buffer = ReadRemoteBlock(PID,PBA);
    return buffer;}
  else return NULL;} //BMT 不存在返回失败
bool write(LBA, buffer){
  If (ExistInBMT(LBA)){
    <PID,PBA> = GetPhysicalInfo(LBA);
    return WriteRemoteBlock(PID,PBA);}
 else{
    <PID,PBA>= AllocateRequest(LBA,VID,
UserID); //向 metadata server 请求新块;
    UpdateBMT(PID,PBA,LBA);
    return WriteRemoteBlock(PID,PBA);}}
```

算法 2 存储空间动态分配

(1)全局资源管理器注册和分配算法

```
void OnRegister(PID, DeviceInfo){
```

```
    UpdateSDT(PID,DeviceInfo); //更新 SDT
    CreateFSBM(PID,DeviceInfo->
Capability);} //创建对应设备的 FSBM
void* Alloc(){
    PID = MinUtility(SDT); //获得使用率最低的存储设备
    PBA = RequestPBA(SDT[PID]->FSBM);
     SDT[PID]->CapabityInUse+=CHUNKSIZE;//修改已使用空间
  SDT[PID]->FSBM->PBA=1; } //设置为以占用
void* Free(PID,PBA){
     SDT[PID]->CapabityInUse-=CHUNK_SIZE; //修改已使用空间
    SDT[PID]->FSBMPointer->PBA=0; }//设置为空闲
```

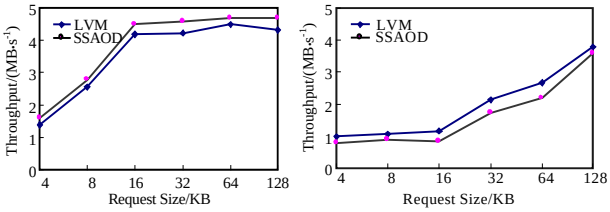
(2)LV 管理算法

```
Packet=ReceiveAllocateRequest();
if (verify(Packet ->UserID)==false return ErrorCode; //验证用户
//合法性
<PID,PBA> = SG_Manager->Alloc();//请求全局资源管理器分配
//存储单元
    AddEntry(VID->BMT, PID, PBA, Packet->LBA);//添加 BMT
//表项
return SendAck(PID,PBA);//返回存储单元信息
```

4 实现与性能评价

本文在TransCom系统[4]中采用了SSAoD的方法来实现多用户存储空间的动态分配。TransCom系统由客户机和服务器通过通信网络连接组成，采用客户机本地计算、服务器集中管理的模式。客户机本地没有存储设备，依赖网络中的存储设备为其提供存储空间。本文将其和传统LVM的方法的读写性能进行了比较。实验环境如下：用户机器使用 1 GHz CPU，128 MB内存和 100 Mb/s网卡；元数据服务器使用 3 GHz CPU，2 GB内存，1 Gb/s网卡，80 GB硬盘(7 200 rpm)，并用另一台同样配置的主机作为存储设备；存储访问协议使用iSCSI；客户机操作系统是红旗Linux 5.0，文件系统是ext2。

实验结果如图 3 所示。可以看出，SSAoD 方法读文件数据的性能略高于 LVM，这是由于 SSAoD 仅存储了数据的磁盘块，因此数据存储位置相对集中，从而减少了寻道时间，如图 3(a)。SSAoD 在写数据方面性能略低于 LVM，这是由于 SSAoD 要对首次写入的逻辑磁盘块进行实时分配，但是第 2 次写入同一块时就没有这项开销，虽然在写性能上略有下降，但并不大，如图 3(b)。上述初步的实验结果表明 SSAoD 开销完全在用户的可接受范围内，可满足实际系统的需求。



(a)读文件性能比较　　(b)写文件性能比较

图 3　实验结果

5 结束语

本文在分析传统 LVM 扩展存储空间方法缺点的基础上，提出一种可满足动态扩展需求的存储空间动态分配方法。该方法具有对上层文件系统的透明性，可提高存储系统的资源利用率和扩展能力。本文暂未对在多用户条件下 SSAoD 元数据更新的原子性、数据一致性，以及存储资源的有效回收利用和性能优化等问题进行深入讨论，这是下一步的研究方向。

参考文献

[1] Tate J, Lucchese F, Moore R. Introduction to Storage Area Network[Z]. (2006-02-04). http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf.

[2] Krueger M, Haagens R, Sapuntzakis C, et al. Small Computer Systems Interface Protocol over the Internet Requirements and Design Considerations[S]. RFC 3347, 2002.

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

[4] 张尧学. 透明计算: 概念、结构和示例[J]. 电子学报, 2004, 32(12A): 169-174.

（上接第 32 页）

度大大加快，能运行更多的测试向量，但缺点是调试比较麻烦，因为信号不能存储，有些故障很难再现。

形式验证是国际上研究的热点[5]，主要采用形式化语言逻辑上进行推断，采用公式证明。龙芯 1 号IP的验证采用半形式化的验证，也就是设置一些AMBA信号的相关组合情况，如果不满足就可以断言不满足AMBA规范，国际上已经有了相应的语言和工具，比较流行的是Verisity公司的E语言和Synopsys公司的Vera语言，都已应用在龙芯 1 号IP的验证中，从而使得验证工作更有效。

随机测试主要是进行 IP 的功能验证，根据龙芯 1 号的指令集，在一定约束下，随机产生指令序列，这样使得代码覆盖率大大提高，从而保证整个 IP 的正确性。

5 结束语

本文介绍了龙芯 1 号IP核的结构，建立了验证平台的虚拟环境，对比传统的IP验证流程，说明了龙芯 1 号IPcore验证方法的充分性，极大提高了IPcore的可信度，给相应的研究工作提供了有益的参考。由于龙芯 1 号IP的复杂性，因此还有很多地方需要提高和改进，比如在功能验证阶段研究如何提高仿真速度[6]、测试向量自动生成、软硬件协同仿真等问题，都是今后进一步研究的方向。

参考文献

[1] Verisity Design Inc.. Functional Verification Automation for IP, Bridging the Gap between IP Developers and IP Integrators[Z]. [2007-03-28]. http://www.verisity.com/resources/whitepaper/ip white paper.html.

[2] AMBA Specification (Rev2.0)[Z]. [2007-03-28]. http://www.arm.com/products/solutions/AMBAHomePage.html

[3] 胡伟武, 唐志敏. 龙芯一号处理器结构设计[J]. 计算机学报, 2003, 26(4): 385-394.

[4] 范东睿, 黄海林, 唐志敏. 嵌入式处理器 TLB 设计方法研究[J]. 计算机学报, 2006, 29(1): 73-80.

[5] Hoon C, Yim M K, Lee J Y. Formal Verification of an Industrial System-on-a-chip[C]//Proceedings of 2000 International Conference on Computer Design. Austin, Texas: [s. n.], 2000.

[6] 冯子军. 仿真 RAM 大小对仿真速度的影响[C]//计算机科学技术研讨会论文集. 大连, 中国: [s. n.], 2004.