计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
48(6): 1089-1100, 2011

一种海量数据分级存储系统 TH-TS

敖 莉[1] 于得水[1] 舒继武[1,2] 薛 巍[1,2]
[1](清华大学计算机科学与技术系 北京 100084)
[2](清华大学信息科学与技术国家实验室(筹) 北京 100084)
(shujw@tsinghua.edu.cn)

A Tiered Storage System for Massive Data: TH-TS

Ao Li[1], Yu Deshui[1], Shu Jiwu[1,2], and Xue Wei[1,2]
[1](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)
[2](*National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing* 100084)

Abstract With the rapid growth of storage scale, the key issues in building massive storage systems are to reduce the total cost of ownership (TOC) and to improve system performance. In this paper, the design and implementation of a tiered storage system, called TH-TS (Tsinghua Tiered Storage), are presented. In TH-TS, we design an approach for migrating files efficiently in tiered storage system (CuteMig), which promotes the files based on their promotion costs and benefits with the incremental scanning technique to acquire files' access information and adaptively demotes files based on the free space of a high-end storage tier to ensure that the high-end storage system will have free space. It eliminates file replacement before promotion, and when files are promoted, it reserves their replicas in the low-end storage system. This approach solves the shortcomings of the traditional on-demand promotions and replacement demotion approaches, resulting in improved system performance and less amount of migrated data. The evaluation results reveal that TH-TS could effectively migrate files among different tiers, and reduce the average response time by 10% compared with LRU, and 39% compared with GreedyDualSize, and that TH-TS could reduce the amount of promoted data by 32% and 59%, and the amount of demoted data is also reduced by 47% and 66% compared with LRU and GreedyDualSize.

Key words tiered storage system; file classification; migration approach; adaptive demotion; migration schedule

摘 要 随着数据存储规模的飞速增长,降低存储系统的总拥有成本,提高数据访问性能成为构建海量存储系统的关键.设计并实现了一个海量数据分级存储系统 TH-TS(Tsinghua Tiered Storage),由多级存储设备构成一体化的数据存储环境.该系统提出了 CuteMig 数据迁移方法:采用基于升级成本和升级收益的升级迁移策略和基于剩余空间的文件自适应降级选择策略,解决了传统 on-demand 迁移方法中迁移数据量大、访问性能不佳的问题.评测结果表明,TH-TS 采用 CuteMig 迁移方法的系统平均 I/O 响应时间比传统的 LRU 和 GreedyDualSize 方法分别降低了 10%和 39%左右,数据升级迁移量分别降低了 32%和 59%左右,降级迁移量分别降低了 47%和 66%左右.

关键词 分级存储系统;文件分级;迁移策略;自适应降级;迁移调度

中图法分类号 TP333.2

收稿日期:2009-04-20;**修回日期**:2010-06-23

基金项目:国家自然科学基金项目(60873066);高等学校博士学科点专项科研基金项目(20070003092);国家"八六三"高技术研究发展计划重大专项基金项目(2009AA01A403).

通信作者:舒继武(shujw@tsinghua.edu.cn)

分级存储系统是针对基于服务需求和成本构建的层次存储系统[1]. 它由具有不同性能、可用性和单位价格等指标的存储级别构成,数据存放在不同的存储级别中(固态磁盘、光纤盘阵、IDE 盘阵、SATA 盘阵和磁带库)[2]. 该系统可满足海量数据存储的高性能、大容量和低成本等要求.

分级存储系统的核心是数据迁移技术. 该技术在不同存储层次之间迁移数据,同时保证迁移过程中数据访问的一致性. 数据迁移分为离线迁移和在线迁移两种. 离线迁移需要将应用停止服务后再进行迁移,它避免了迁移过程中对数据一致性的维护. 由于目前企业级应用都要求 7×24 h 在线,离线迁移已不适合大规模存储系统的需要,因此在线迁移成为迁移技术的研究热点. 目前,已有的在线数据迁移技术都存在如下缺陷:

1) 迁移条件缺乏自适应机制. 一些分级存储系统的迁移策略是由管理员预先制定好的[3-4]. 如在生命周期管理的体系结构 STEPS[3] 中,具体迁移策略由管理员手工设定,文件在创建时就按照一定的放置策略放入不同的存储池中,在文件的生命周期内,由预先设定好的迁移策略将文件在不同存储池之间迁移. 该迁移方法简单易操作,但不能很好地适应动态变化的负载.

2) 迁移代价高. 如基于 Lustre 的分级存储管理系统[5] 中,文件从离线设备迁移到在线设备都是由访问缺失触发的,因此造成一次访问缺失的代价很大,且不支持文件的在线迁移.

3) 传统的文件迁移方法的升级策略都是 on-demand 类型. 如果被访问的文件没有在高端存储系统中命中,则将其从低端存储系统迁移到高端存储系统中. 该方法的缺点是没有考虑文件的其他信息,比如文件大小、访问间隔等,造成升级的文件过多. 具有代表性的两种文件迁移方法为:①LRU(least-recently-used),优先将最近最不常使用的文件进行替换. LRU 的缺陷在于平等地对待全部文件,没有考虑到文件的大小,而文件大小决定了文件的迁移代价[6]. ②GreedyDualSize[7],基于文件的 recency, size 和 migration cost 对文件进行替换. 该方式升级迁移的数据量大[8-9],而且文件升级前需要通过降级来替换文件,增加了文件访问的响应时间.

针对已有在线数据迁移技术存在的不足,本文提出了一种高效的数据迁移方法——CuteMig,该方法采用基于升级成本和升级收益的文件分级策略,动态地考虑了文件大小和文件访问频度,将升级成本与收益比值满足条件的文件进行迁移,既保证了升级必要的热点文件达到较高的命中率,也解决了 on-demand 升级方式迁移数据量大的问题. 同时 CuteMig 采用基于剩余空间的文件自适应降级选择策略,根据高端存储系统的剩余空间情况主动地选取文件来降级,保证了高端存储系统中始终有剩余空间,解决了传统替换策略在升级前必须先执行 DEMOTE[10] 操作进行替换的问题.

在 CuteMig 迁移方法的基础上,我们为一种物理数值模拟的海量数据存储,设计并实现了一个分级存储系统 TH-TS,该系统基于并行文件系统 PVFS2[11-12],采用增量扫描的方式获取文件访问频度信息,建立升级和降级队列管理迁移任务,减少了迁移决策的开销,提高了系统的迁移效率. 评测结果表明,TH-TS 可以根据文件访问频度在不同数据服务器之间有效地迁移数据,同时 CuteMig 迁移方法和传统迁移方法 LRU 和 GreedyDualSize 相比,升级迁移量下降了 32%和 59%;降级迁移量下降了 47%和 66%,且 CuteMig 的平均 I/O 响应时间比 LRU 最多可降低 10%,比 GreedyDualSize 最多可降低 39%.

1 THTS 体系结构

1.1 硬件架构

TH-TS 按功能划分,包括客户端、元数据服务器和数据服务器 3 部分,如图 1 所示:

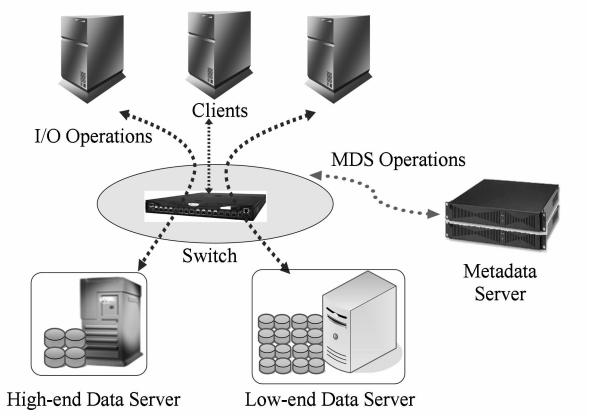


Fig. 1 Hardware framework of TH-TS.

图 1 TH-TS 硬件架构图

在图 1 中,元数据服务器和客户端之间的通路称为元数据路径,数据服务器和客户端之间的通路称为数据路径. 元数据服务器负责把位于不同数据服务器上的数据文件组织成统一的文件系统视图,为客户端软件提供元数据操作服务,同时执行文件扫描、数据分级、迁移决策和迁移速率控制等操作,

实现对迁移操作的单点管理;数据服务器保存每个文件分片后的数据文件,为客户端软件提供文件I/O操作,同时执行元数据服务器发来的文件迁移指令;文件系统客户端软件实现虚拟文件系统层和MPI-IO层的各种文件操作.

1.2　软件体系结构

TH-TS的软件体系结构也分成3部分:客户端软件、元数据服务器软件和数据服务器软件,如图2所示.TH-TS的软件体系结构的重要特征是层次化,客户端软件、元数据服务器软件和数据服务器软件均由若干层构成.这种层次化的结构保证了接口和实现分离,层与层之间通过接口进行交互,对某一层代码的修改对于其他层是透明的,降低了模块间的耦合性.

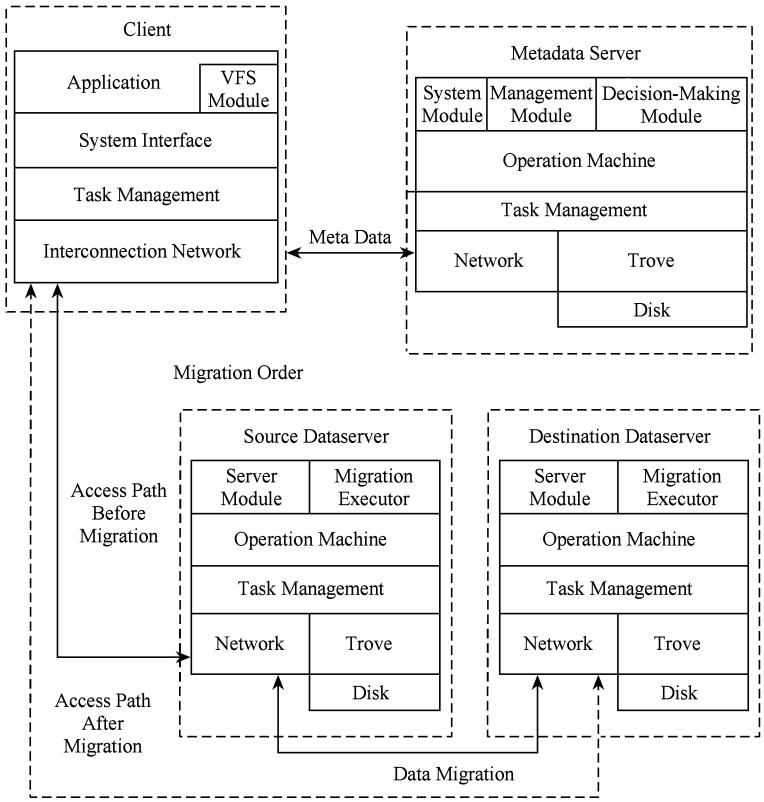


Fig. 2　Software framework of TH-TS.

图2　TH-TS软件体系结构图

1.2.1　客户端软件

客户端软件分为应用层、系统接口层、任务管理层和网络通信层,如图2所示.为了提供客户端软件的灵活性,并在系统访问性能和应用程序简易性之间取得平衡,客户端软件设计的重点就是实现系统接口模块和VFS模块之间的交互与转化.

客户端软件的模块图如图3所示:

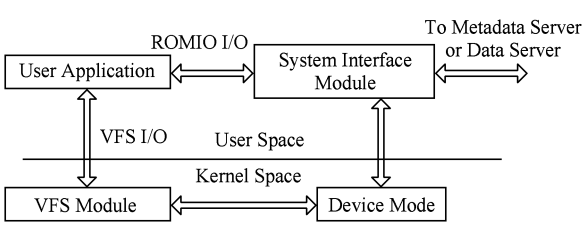


Fig. 3　The software module of client.

图3　客户端软件模块图

TH-TS设计了一个虚拟设备节点,实现了系统接口模块和VFS模块之间的交互与转化.在内核空间实现的VFS模块,通过系统接口模块中的系统接口,实现文件的VFS层操作,供用户通过VFS层对并行文件系统中的文件进行访问.具体交互和转化过程为:VFS层收到应用发来的请求之后,VFS模块将其转化成对系统接口模块的访问请求,并将该请求写入到虚拟设备节点中(该节点由客户代理软件在初始化时创建),同时系统接口模块产生一个子进程调用,从虚拟设备节点中读取该请求,一旦读取到请求,按照请求内容调用对应的系统接口,并将系统接口的执行结果写回到虚拟设备节点中,最后VFS模块从虚拟设备节点中读取调用结果并将其返回给应用程序.

1.2.2 元数据服务器软件

元数据服务器是整个 TH-TS 系统中负责迁移和数据管理的主控节点,其设计主要包括:1)获取数据服务器的文件访问频度信息;2)管理并调度迁移任务;3)与数据服务器交互来控制迁移的执行.为了实现以上功能,元数据服务器软件设计了文件迁移决策模块,它包含了增量扫描器、文件访问表管理器以及迁移调度控制器 3 个子模块.

1) 增量扫描器.定期获取数据服务器记录的文件访问信息.每隔一个扫描周期,增量扫描器中的文件扫描线程向数据服务器发出扫描指令,获取文件访问信息,包括文件的 *inode* 数值、文件在本周期内的访问次数和文件大小等.

2) 文件访问表管理器.负责维护一个文件访问表,保存每个文件的访问次数、文件大小、文件访问间隔和当前位置等信息.文件访问表管理器使用增量扫描获取的文件访问信息来更新文件访问表,并判断是否有新的文件需要迁移,若有文件需要迁移,根据文件的迁移目标将该文件放入升级队列或者降级队列.

3) 迁移调度控制器.负责迁移任务的调度和处理.升级队列和降级队列分别保存需要升级和降级的候选文件.由于升级任务的紧迫性,每当升级队列中有需要进行升级的升级候选文件,升级线程就对其进行升级操作;由于降级任务的紧迫性相对较小,降级线程加入了速率控制技术,当系统负载较小时才对降级候选文件进行处理,系统负载较重时不进行降级操作,避免降级任务同前端负载争用带宽资源.

为了处理除迁移决策模块功能之外的过程,元数据服务器还包含元数据系统模块和元数据管理模块.如图 4 所示.

元数据系统模块负责提供执行元数据操作的接口,包括文件创建、文件删除、目录创建、目录删除、文件查找等操作.元数据管理模块负责提供管理元数据的接口,把多个数据服务器组织成一个统一的文件自管理系统,包括管理目录项、获取系统负载情况、显示文件系统统计信息等操作.除此之外,元数据服务器软件还包括操作状态机、任务管理层、网络通信层和存储管理层.操作状态机维护一张操作映射表,映射表中每个表项的内容包括操作码、字符串类型的操作名称以及执行该操作请求的操作状态机入口地址.其任务管理层与网络通信层与客户端软件相同.

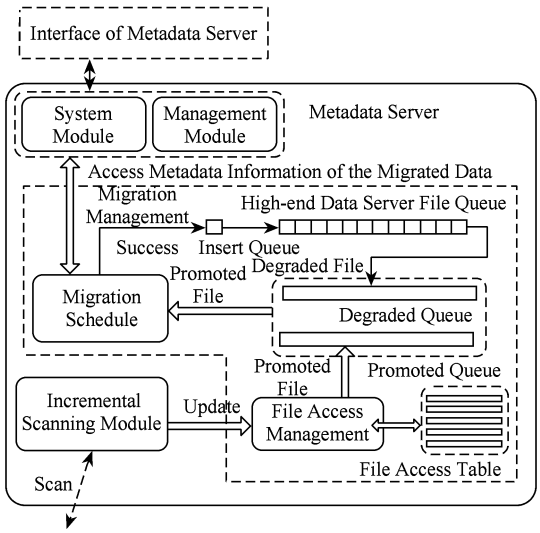


Fig. 4 The software module of metadata server.

图 4 元数据服务器软件模块图

1.2.3 数据服务器软件

数据服务器负责向客户端软件提供 I/O 服务、记录数据文件的访问频度信息、执行元数据服务器发来的迁移请求.数据服务器软件的设计主要包括:

1) 数据分片.为了提高 I/O 性能,文件数据按照一定分片规则分布在不同数据服务器上.数据服务器需要保证客户端软件和元数据服务器软件按照分布信息可以获取正确的数据.

2) 记录数据文件的访问频度信息.数据服务器软件需定时记录数据文件的访问频度信息,并在收到元数据的扫描指令后把数据文件的访问频度信息返回给元数据服务器.

3) 数据服务器不仅在收到元数据服务器的迁移指令后开始执行迁移,而且还需把要迁移的数据文件写入目标数据服务器的数据文件中,以完成迁移任务.

为了实现以上 3 个技术点,数据服务器软件设计了数据分片策略、I/O 记录模块和迁移执行模块来完成其功能.

2 CuteMig 数据迁移

分级存储系统中数据迁移是核心技术.为提高分级存储系统的 I/O 性能,达到高的 I/O 命中率及少的数据迁移量,我们设计了 CuteMig 数据迁移方法,主要包括以下 3 种关键技术.

2.1 基于升级成本和升级收益的文件分级策略

基于升级成本和升级收益的文件分级策略是根

据文件大小和文件访问频度信息分别计算文件升级的成本和升级后的收益,使用二者的比值对文件进行分级,并根据文件分级结果决定是否对文件升级,以提高系统的整体性能.

1) 文件升级的成本定义为升级需要传送的数据量:

$$Cost = filesize. \tag{1}$$

传送数据过程增大了高端和低端存储系统的 I/O 负载,也通过竞争带宽资源增大了前端应用的响应时间,文件越大迁移过程对前端应用的影响也越大.

2) 文件升级后的收益定义为文件升级后被访问的吞吐率.其计算过程包括如下两个步骤:

步骤 1. 计算文件的平均访问时间间隔:

$$avg\_interval = \begin{cases} INFINITE, \text{ if the file has never been accessed before;} \\ current\_interval, \text{ else if } avg\_interval \text{ equals to } INFINITE; \\ \alpha \times avg\_interval + (1-\alpha) \times current\_interval, \text{ else;} \end{cases} \tag{2}$$

其中文件的当前访问间隔为

$$current\_interval = current\_access\_time - last\_access\_time. \tag{3}$$

式(2)中,如果文件以前没有被访问过,它的平均访问时间间隔设为 $INFINITE$;如果文件是第 2 次被访问,将它的平均访问间隔就是当前访问间隔,否则按遗忘因子 α 将当前访问间隔和旧的平均访问间隔加权求和,得到新的平均访问间隔.该方法既考虑了文件当前的访问间隔,也通过遗忘因子 α 把文件过去的访问间隔信息反映到平均访问间隔中.

步骤 2. 计算文件的升级收益.设 $access\_num$, $access\_bytes$, $filesize$, $avg\_interval$ 分别表示文件的总访问次数、总访问字节数、文件大小和文件的平均访问间隔,文件升级后经过 T 时间的升级收益表示为

$$RequestAfterPro = \frac{T}{avg\_interval} \times \frac{access\_bytes}{access\_num}. \tag{4}$$

$T/avg\_interval$ 表示文件在 T 时间内的预期访问次数,$access\_bytes/access\_num$ 表示文件每次访问的平均字节数.式(4)中升级收益是随着时间 T 不断累积的,因此平均收益为

$$Benefit = \frac{RequestAfterPro}{T} = \frac{access\_bytes}{avg\_interval \times access\_num}. \tag{5}$$

由于 $access\_num$, $access\_bytes$, $avg\_interval$ 都是文件的历史访问特征,可近似认为文件的访问特征在短时间内不会发生变化[8],因此 $Benefit$ 是根据文件历史访问特征预测的文件近期内的平均收益.

为保证升级热点文件提高访问性能的同时,尽量降低迁移的数据量,我们使用迁移成本和收益的比值表示文件的迁移优先级 $MigLaziness$,即

$$migLaziness = \frac{Cost}{Benefit} = \frac{filessize \times avg\_interval \times access\_num}{access\_bytes}. \tag{6}$$

该优先级越小文件被升级的概率越大.每当文件被访问,即更新平均访问间隔、总访问大小、总访问字节数等迁移相关信息,计算升级迁移的优先级值.该值越小说明升级的成本越小,而且升级后的收益越大,如果文件的迁移优先级小于升级阈值,即对该文件执行升级操作.

CuteMig 中基于升级收益和成本的文件分级策略在文件访问频度和文件大小之间作了权衡,只有迁移优先级小于升级阈值的文件才会被升级,既保证了升级必要的热点文件达到较高的命中率,也解决了 on-demand 升级方式迁移数据量大的问题,同时该策略根据文件访问信息动态改变迁移策略,提高了迁移方法的自适应性.

另外,该方法给系统带来的额外开销也较低.在文件迁移过程中计算文件升级所需要的信息仅在文件被访问时才更新,避免了定期扫描所有文件以进行降温的过程[8].平均收益的计算中每个文件需要增加 $access\_num$, $access\_bytes$, $avg\_interval$ 三项元数据信息,设每个变量使用 4 B 表示,则每个文件只需增加 12 B 的元数据.对于包含 500 万个文件的文件系统,元数据的额外存储开销仅为 60 MB.

2.2　基于剩余空间的文件自适应降级选择策略

基于剩余空间的文件自适应降级选择策略根据文件的访问情况和高端存储设备的剩余空间,主动地选择需要降级的文件,以保证高端存储系统中始终有剩余空间.该方法解决了传统替换策略在升级前必须先执行 DEMOTE[1] 操作进行替换的问题.

1) 维护一个 LRU 栈.所有升级到高端存储系统上的文件都放入该 LRU 栈中.每当高端存储系统上的文件访问完成后,则将其放入 LRU 栈的 MRU 端,同时检查 LRU 栈中 LRU 端的文件,根据该文件的上次访问时间和当前时间来计算它的未访

问时间.

2) 根据文件未访问时间与降级阈值的比值进行降级判断. 如果它的未访问时间大于降级阈值 *Demotion_threshold*,那么将其放入降级候选队列中,由降级调度程序处理. 降级阈值在初始化时被赋值为初始值 *Init_demotion*(该值为可变参数),之后它按照图 5 中的方法自适应地变化:每当降级候选队列中的文件被访问,把该文件重新放入 LRU 栈,同时把降级阈值设置为该文件的本次访问和上次访问的时间间隔,作为对降级阈值的惩罚;每当降级线程从降级候选队列中成功降级了一个文件,把降级阈值设置为 LRU 栈中 LRU 端文件的未访问时间,作为对降级阈值的奖励,以使 LRU 栈中更多的文件进入到降级候选队列中来.

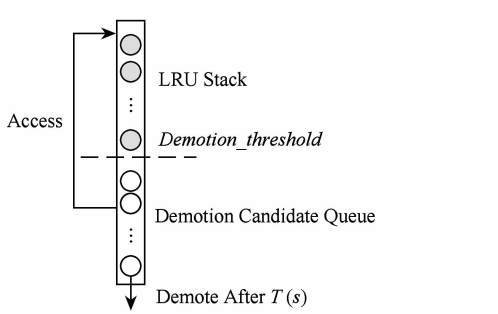


Fig. 5 Sketch map of LRU stack and demotion candidate queue.

图 5 LRU 栈和降级候选队列示意图

3) 降级频率. 如图 5 所示降级后候选队列保存了 LRU 栈中大于降级阈值的文件. 每隔时间 T,调度程序从降级候选队列的队首取出降级候选文件执行降级操作. T 的计算公式如下:

$$T = Demotion\_threshold \times freeratio^{K}. \quad (7)$$

式(7)中 K 是可调参数,*freeratio* 是高端存储系统的剩余空间占其总空间的比例,取值范围是[0,1],因此 T 的取值范围是[0,*Demotion_threshold*]. 降级的频率根据 *Demotion_threshold* 和 *freeratio* 自适应变化:*Demotion_threshold* 和 *freeratio* 越小,T 值越小,降级操作执行得越频繁;*Demotion_threshold* 和 *freeratio* 越大 T 值越大,降级操作执行得越不频繁. 式(7)保证了高端存储系统始终有剩余空间,当高端存储系统的空间所剩无几,即 *freeratio* 值很小,两次降级的时间间隔 T 也很小时,降级调度程序会频繁地从降级候选队列中降级文件,同时按照图 5 中的方法不断调低降级阈值,使得更多的文件进入到降级候选队列中. 随着降级文件的增多,*freeratio* 的数值逐渐增大,同时由于大量文件进入到降级候选队列中,对这些文件的访问会使降级阈值受到惩罚而增加,这样 T 随着降级阈值和 *freeratio* 的增加而增加,降级频率随之降低. 因此该策略是一个反馈过程,最终降级阈值和高端存储系统的剩余空间都会稳定在一个范围内.

2.3 迁移调度控制

为了避免迁移过程影响前端应用,TH-TS 采用了迁移调度控制,按照迁移目标的不同,将迁移任务分为升级迁移和降级迁移,并用双候选队列技术,使用升级和降级队列分别管理调度这两种迁移任务. 这种迁移任务区分的方法保证了紧迫的升级任务可以迅速执行,同时不紧迫的降级任务在负载较轻时才执行.

TH-TS 把迁移任务分成两类:将数据从低端存储系统迁移到高端存储系统的过程称为升级迁移;将数据从高端存储系统迁移到低端存储系统的过程称为降级迁移. 这两类迁移的目标不同:升级迁移是为了把热点数据迁移到高端存储系统中,以提高系统的访问性能;降级迁移是为了把非热点数据迁移到低端存储系统中,以使高端存储系统拥有足够的剩余空间,来保存后续可能升级的热点文件. 对于所有升级到高端存储系统的文件,迁移调度控制器维护一个 LRU 队列来保存这些文件的 *inode* 值,该队列简称为 LRU-PF(LRU queue of promoted files). 降级线程定期检查位于 LRU-PF 队列的 LRU 端的文件,如果该文件达到了降级要求,则它成为降级候选文件;每次对文件访问表更新后,如果被更新的文件达到了升级要求,则它成为升级候选文件. 升级或降级候选文件的基本信息如 *inode* 值、数据的当前位置和数据的目标位置等被放入升级或降级队列中,由迁移调度控制器中的升级线程和降级线程进行调度.

3 性能评测

3.1 测试方法与实验平台

文件迁移是针对长期访问的大规模文件系统进行的一种优化. 我们使用 berkeley trace 来评测 TH-TS 分级存储系统在真实使用环境中迁移文件的有效性和不同迁移方法中 I/O 访问的平均响应时间.

测试系统由一台文件系统 client、一台元数据服务器和两台数据服务器组成(如图 1 所示),其中一台数据服务器作为高端存储系统,另一台数据服务器作为低端存储系统. 客户端和元数据服务器的

配置相同,如表 1 所示,数据服务器的具体配置情况如表 2 所示.

Table 1　Configuration of Client and Metadata Server

表 1　客户端和元数据服务器的配置

Parameter	Parameter's Value
CPU	Intel Itanium2 1 GHz×2
Memory/GB	2
OS	Linux (kernel: 2.6.9)
SCSI Disks	HP SCSI Disk(36 GB 15 KRPM)×1

Table 2　Configuration of Data Server

表 2　数据服务器的配置

Parameter	Parameter's Value of High-end Data Server	Parameter's Value of Low-end Data Server
CPU	Itanium2 1.5 GHz×2	Intel Xeon 700 MHz×1
Memory/GB	16	1
OS	Linux (kernel: 2.6.9)	Linux (kernel: 2.6.20)
SCSI Disks	(36 GB 10 KRPM)×3	(36 GB 10 KRPM)×1

我们把客户端、元数据服务器和高端数据服务器接到千兆以太网交换机上,把低端数据服务器接到百兆网交换机上,以进一步使高端数据服务器和低端数据服务器产生性能差异.经过测试,高端和低端数据服务器在 TH-TS 系统中的最大吞吐率分别为 9.3 MBps 和 3.4 MBps.

3.2　基于 Berkeley Trace 的测试

3.2.1　Berkeley Trace 及 BKReplayer 介绍

测试采用的 Trace 是由加州大学伯克利分校的 Roselli 等人在 1997 年采集的系统调用层的文件访问 Trace,采集过程接近 3 个月[13-14].其中我们选择 Berkeley Trace 中 1996-12-01—1996-12-30 之间采集的 1 个月的 Research Trace 作为模拟的测试数据.表 3 总结了这组 Trace 的相关信息:

Table 3　The Information of Research Trace

表 3　Research Trace 信息

Statistics Information	Research Trace
Average number of files	188 239
Total files' size/KB	4 314 900
Length of trace/d	30
Environment	13 HP-UX server

我们统计了 Research Trace 中每个文件的总访问字节数,并按照总访问字节数由高到低的顺序对文件进行排序,统计前 20%的热点文件的总访问字节数占全部访问字节数的百分比,如图 6 所示,Research Trace 的空间局部性很强,占总文件数 0.05%的文件的 I/O 访问量即占到了总文件的 90%.

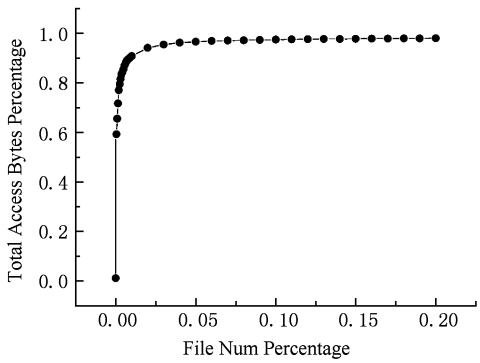


Fig. 6　File access density in Research Trace.

图 6　Research Trace 文件访问密集度

BKReplayer 是一款用来播放 Berkeley 格式 Trace 的播放器.它以行为单位从 Berkeley Trace 文件中读取 Trace,并在规定的时间将该行 Trace 对应的 VFS 请求发送给 TH-TS 的客户端软件.

BKReplayer 的设计难点在于如何恢复初始文件系统的状态[15-17].由于 Trace 在采集时文件系统已经存在很多文件,而且在 Trace 采集过程中常常包含对这些文件的操作,因此在 Trace 播放前需要恢复出文件系统的初始状态.通常采用的恢复方法是快照[18],但是 Berkeley Trace 没有提供 Trace 采集过程开始前文件系统的快照.因此我们在 BKReplayer 中采用了前处理方法来恢复文件系统的初始状态.

3.2.2　TH-TS 的平均响应时间测试

本次实验中,为了测试 TH-TS 迁移文件的有效性,我们在 TH-TS 的客户端软件上使用 BKReplayer 播放 Research Trace 中第 31 天前 12 h 的 Trace,当高端存储系统的大小分别为 1 GB 和 2 GB、CuteMig 的升级阈值为 1 000、降级阈值初始值(*Demotion_threshold*)为 100 000、遗忘因子 α 为 0.5、式(7)中的可调参数 K 为 2 时(4.2.4 节将讨论升级阈值和降级初始阈值的选取对迁移的影响),元数据服务器软件采用 TH-TS 系统的 CuteMig 迁移方法与传统的 LRU, GreedyDualSize 两种迁移方法进行比较,得到数据升降级迁移量如图 7 所示.同 LRU 和 GreedyDualSize 两种方法相比,CuteMig 显著降低了迁移的数据量.以高端存储系统大小为 1 GB 为例, CuteMig 的升级迁移量比 LRU 和 GreedyDualSize 分别减少了 32.2% 和 59.3%, CuteMig 的降级迁移量比 LRU 和 GreedyDualSize 分别减少了 47.5%和 66.3%.

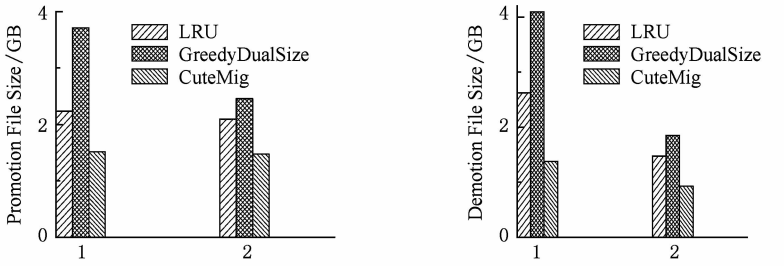


Fig. 7 Amount of promoted and demoted data of LRU, GreedyDualSize and CuteMig Approaches.

图 7 LRU, GreedyDualSize 和 CuteMig 的升降级迁移量

表 4 显示了 CuteMig, LRU 和 GreedyDualSize 3 种迁移方法的 I/O 访问平均响应时间和字节命中率的比较结果. 同 LRU 和 GreedyDualSize 相比, 尽管 CuteMig 的字节命中率比两者略低, 但因迁移量显著下降, 因此 CuteMig 的平均响应时间要低于两者. 同 LRU 相比, CuteMig 的响应时间最多下降了 10.1%, 同 GreedyDualSize 相比, CuteMig 的响应时间最多下降了 39.4%.

Table 4 Average Response Time and Byte Hit Rate of Different Approaches

表 4 不同迁移策略的平均响应时间和字节命中率

Approaches	Avg Response Time/ms	Bytes Hit Ratio/%
LRU-1 GB	0.89	96.7
LRU-2 GB	0.83	96.9
GreedyDualSize-1 GB	1.32	94.3
GreedyDualSize-2 GB	1.16	94.8
CuteMig-1 GB	0.80	91.2
CuteMig-2 GB	0.78	91.4

分析原因如下: 1) 传统的文件迁移方法 LRU 和 GreedyDualSize 的升级策略都是 on-demand 类型的, 即如果被访问的文件没有命中在高端存储系统中, 则将其从低端存储系统(磁带、慢速磁盘或远程的存储节点)迁移到高端存储系统中. 该方法的缺点是没有考虑文件的其他信息, 比如文件大小、访问间隔等, 造成升级的文件过多. 2) 如果高端存储系统已经没有剩余空间, 同时又有文件需要升级, LRU 和 GreedyDualSize 则先使用一定的替换算法从高端存储系统中替换出一个文件进行降级操作, 再对要升级的文件执行升级. 3) CuteMig 则根据文件大小和文件访问频度信息分别计算文件升级的成本和升级后的收益, 使用二者的比值对文件进行分级, 并根据分级结果决定是否对文件升级; 同时根据高端存储系统的剩余空间情况自适应地选取文件来降级, 保证高端存储系统始终有剩余空间, 避免了高端存储系统的剩余空间耗尽之后每次升级都必须先进行替换的情况. 因此, LRU 和 GreedyDualSize 比 CuteMig 的迁移数据量大, 而且文件升级前需要通过降级来替换文件, 增加了文件访问的响应时间.

图 8 对比了 CuteMig 与 LRU, GreedyDualSize 在高端存储系统大小分别为 1 GB 和 2 GB 是整个迁移过程中的平均响应时间变化情况. 图中 X 坐标为 Trace 播放的时间, 对应的 Y 坐标表示 Trace 中从 0

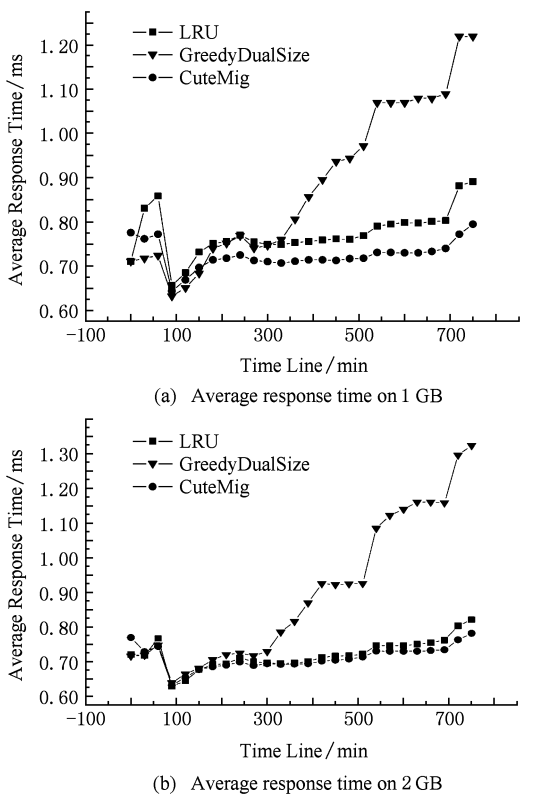


Fig. 8 Average response time of different approaches on 1 GB and 2 GB.

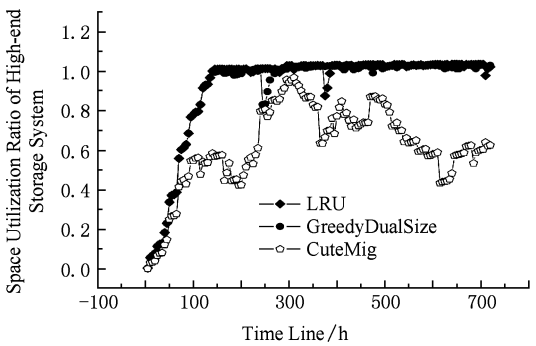
图 8 1 GB 和 2 GB 情况下不同策略的平均响应时间

到该时间内的平均响应时间.比如横坐标为 700 的点对应的 Y 坐标表示 Trace 中 0～700 min 之间所有 I/O 访问的平均响应时间.

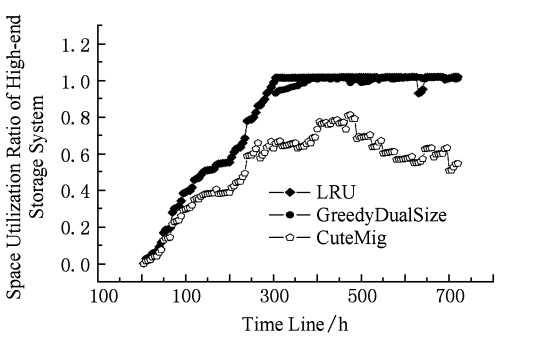
测试刚开始时,CuteMig 由于命中率稍低,平均响应时间要略高于 LRU 和 GreedyDualSize.随后,由于 CuteMig 的迁移量少于其他两种方法,补偿了命中率的降低带来的性能损失,因此平均响应时间最终低于后两者,说明 CuteMig 方法带来了总体性能的提升.

3.2.3　TH-TS 的空间利用率测试

为了评价 TH-TS 系统的空间利用率,我们测试了 LRU,GreedyDualSize 和 CuteMig 三种方法在 30 d 的模拟过程中,高端存储系统的空间利用率的变化情况.图 9 为高端存储系统大小分别为 1 GB 和 2 GB 时,LRU,GreedyDualSize 和 CuteMig 三种方法中高端存储系统的空间利用率:



(a)　Space utilization on 1 GB



(b)　Space utilization on 2 GB

Fig. 9　Space utilization in high-end storage system of different approaches.

图 9　不同策略的高端存储系统空间利用率

测试结果显示 LRU,GreedyDualSize 预热结束之后空间利用率达到 100%,因此二者对应的曲线大部分时间是重合的.由于进行了自适应降级操作,CuteMig 方法高端存储系统上的空间利用率一直没有达到 100%,同时 CuteMig 保证了只降级高端存储系统中的非热点文件,热点文件仍被保留,因此 CuteMig 的空间利用率维持在一个合理范围.而 LRU 和 GreedyDualSize 中的降级是在替换时才进行的,因此 LRU 和 GreedyDualSize 两种方法中高端存储系统的利用率在预热结束后就达到了 100%,以后如果有文件需要升级,则必须要先替换出一个文件后,增大了响应时间.

3.2.4　CuteMig 参数的选取

为了评测 CuteMig 迁移策略中各参数对系统迁移数据量、字节命中率等性能的影响,我们设计了一个文件迁移模拟器,模拟 CuteMig 中文件的迁移行为和系统状态,包括升级文件大小、降级文件大小和 I/O 访问的字节命中率.模拟过程中遗忘因子 α 取 0.5.

图 10 显示了高端存储系统空间大小为 1 GB 和 2 GB 时,使用不同的升级阈值获得的字节命中率和升级迁移量的变化情况.当升级阈值从 0 开始增长时,命中率和迁移量都随着升级阈值的增加而增加.升级阈值大于 4 000 后,1 GB 和 2 GB 两种情况下命中率开始稳定.因为阈值在 4 000 左右时已经把热点文件都执行了升级,此后再增大阈值命中率已经没有大的提升,反而迁移量会不断增多.高端存储系统的空间大小对命中率和迁移量也有影响,在升级阈值相同的情况下,高端存储系统空间越大命中率越高,同时升级迁移量越少.这是因为高端存储系统的空间越大,文件在高端存储系统中停留时间也就越长,避免了文件被降级之后由于访问密集又再次被升级的情况,提高了命中率同时也减少了迁移量.

为了选择合适的升级阈值,保证命中率较高的同时,迁移量较少,我们定义函数 $F(p)=(1-BytesHitRatio_p)\times PromotionSize_p$ 来评价升级阈值.其中 $BytesHitRatio_p$ 和 $PromotionSize_p$ 分别表示升级阈值为 p 时的字节命中率和升级迁移量.$(1-BytesHitRatio_p)$ 越小命中率越高,$PromotionSize_p$ 越小升级迁移的迁移量越少.因此 $F(p)$ 越小升级阈值 p 越理想.图 11 显示了不同阈值 p 下 $F(p)$ 的变化情况.随着升级阈值从 0 开始增长,命中率和迁移量都开始增加,但是命中率的增加幅度要高于迁移量的增长幅度,因此 $F(p)$ 显著下降,当升级阈值为某值时达到最小(1 GB 时该值为 1 000,2 GB 时该值为 4 500),之后随着升级阈值增大,升级迁移量逐渐增大,但是命中率基本保持稳定,因此 $F(p)$ 也逐渐增长.为便于比较,在系统性能测试的模拟分析中,我们统一选择 1 000 作为高端存储系统为 1 GB 和 2 GB 时的升级阈值,保证了较高命中率的同时,升级迁移量也较少.

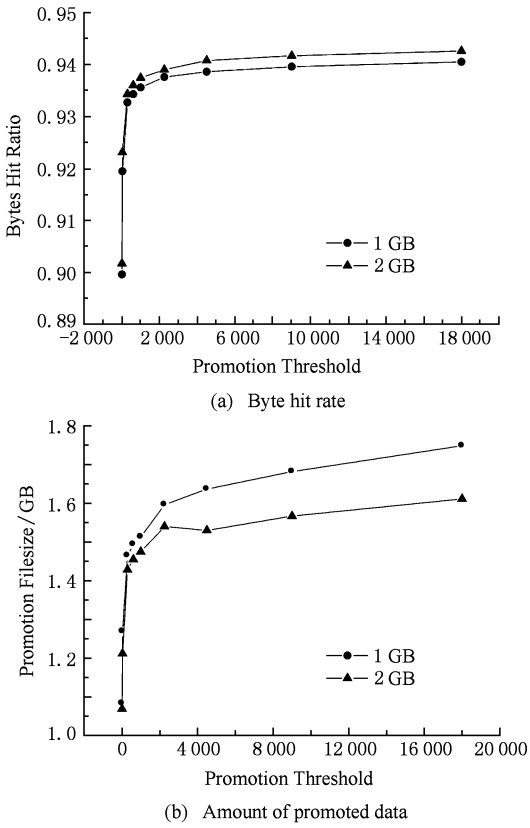


Fig. 10 Variation of the byte hit rate and the amount of promoted data with different promotion threshold.

图 10 不同升级阈值下的字节命中率和升级迁移量的变化

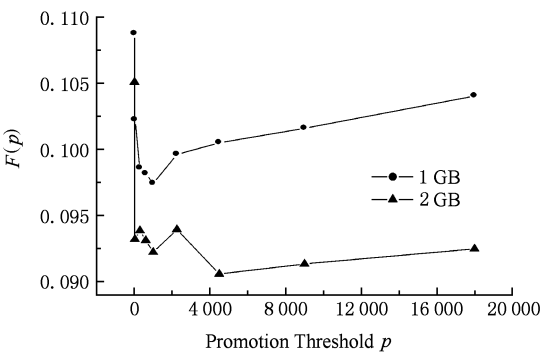


Fig. 11 Variation of $F(p)$ with different promotion threshold.

图 11 不同阈值下 $F(p)$的变化

在基于剩余空间的文件自适应降级选择策略中,我们为降级阈值设置了一个初始值,目的是保证算法开始时,会有文件因为未访问时间大于降级阈值而进入到降级候选队列中,使降级阈值的自适应变化过程能够启动. 为了验证初始值对降级阈值变化的影响,我们分别使用不同的降级阈值初始值对 Berkeley Research Trace 进行了模拟,观察初始值对降级阈值自适应变化过程的影响效果(实验中式(7)中的可调参数 K 设为 2).

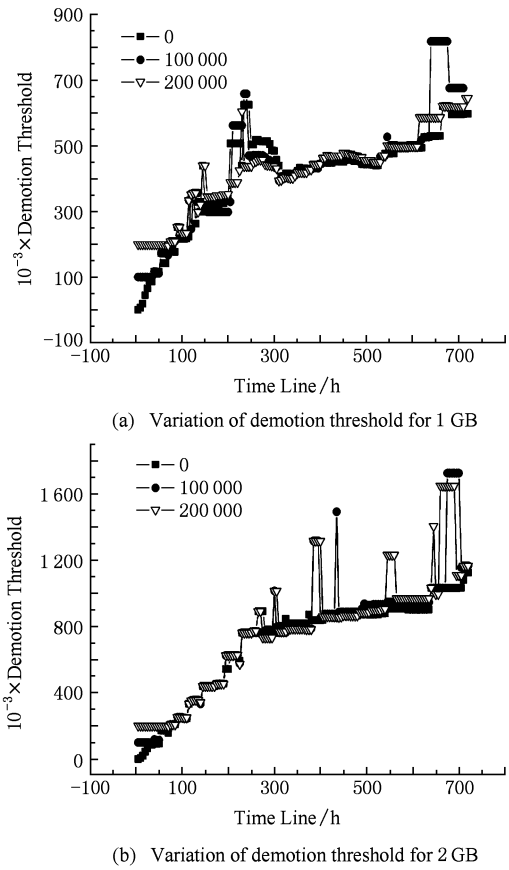


Fig. 12 Variation of demotion threshold with different initial values for 1 GB and 2 GB high-end storage systems.

图 12 高端存储系统为 1 GB 和 2 GB 时不同初始值下降级阈值的变化

在图 12 中,降级阈值的初始值分别取 0,100 000,200 000. 尽管初始值不同,随着时间的增长,3 种初始值情况下的降级阈值逐渐接近,当模拟到 300 h 之后,3 种初始值情况下的降级阈值变化曲线基本重合. 这是对降级阈值自适应调整的结果. 当降级阈值初始值比较小时,会有很多文件都被放入降级候选队列中,对降级候选队列中的文件访问也会比较频繁,造成降级阈值频繁地改变,降级阈值会逐渐接近候选队列中文件的访问间隔的最大值. 因此 3 种初始值情况下,降级阈值最终会重合到一起. 只要初始值选择在合适范围内,自适应降级策略都会将降级阈值逐渐调整到一个固定的区间内变化,所以该策略对降级阈值的初始值是不敏感的. 另外,随着高端存储系统空间大小的增长,经调整稳定之后的降级阈值也不断增加:高端存储系统为 1 GB 时,降级

阈值在[400 000,600 000]区间内变化;高端存储系统为 2 GB 时,降级阈值在[800 000,1 200 000]区间内变化.这是因为高端存储系统的空间越大对降级的需求越不紧迫,对降级候选队列中候选文件的降级操作执行得越不频繁.根据图 5 中所示的降级阈值自适应变化算法,降级阈值得到奖励的机会越少,但得到惩罚的机会增多,因此降级阈值会逐渐增大,直到达到一个稳定范围内.

根据以上分析我们可以得出结论:降级阈值对初始值不敏感,最终的稳定值取决于高端存储系统的空间大小,说明基于剩余空间的文件自适应降级选择策略对不同初始值和不同的高端存储系统空间大小有很强的自适应性.

4　结　　论

本文介绍了海量数据分级存储系统 TH-TS 及实现的几个关键技术.该系统具有一般分级存储系统具备的功能.与同类系平台相比,本系统具有如下特点:1)采用基于升级成本和升级收益的升级迁移策略,解决了传统迁移方法中 on-demand 方式升级的缺点,提高了系统的访问性能并减少了迁移的数据量;2)提出基于剩余空间的文件自适应降级选择策略,解决了传统替换策略在升级前必须先执行 DEMOTE 操作进行替换的问题;3)将迁移划分为升级和降级,针对迁移任务的不同,设计了双候选队列,采用不同的策略分别处理升、降级任务,提高了系统的迁移效率.

结合以上关键技术实现的分级存储系统 TH-TS 既具有好的可用性,又表现了较高的性能.本文应用 Trace 播放器对 TH-TS 系统进行了性能测试.性能评测结果表明,TH-TS 可以有效地根据数据访问信息执行数据迁移,同时也说明同传统迁移方法相比,数据迁移方法 CuteMig 显著降低了数据迁移量并提升了系统性能.

参　考　文　献

[1] Michael P. ILM and tiered storage [EB/OL]. USA: Storage Networking Industry Association, 2006 [2010-03-30]. http://www.snia.org/about/resources/DMF-ILM_and Tiered_Storage_20060221.pdf

[2] Zhu Shengyu. Getting started with IL [EB/OL]. Beijing: Storage Networking Industry Association, 2005 [2010-03-30]. http://www.sniachina.org/css/pdf/edu/tutorial_download/Getting_Started_with_ILM_v6.pdf

[3] Verma A, Pease D, Sharma U, et al. An architecture for lifecycle management in very large file systems [C] //Proc of the 22nd IEEE/the 13th NASA Goddard Conf on Mass Storage Systems and Technologies. Los Alamitos, CA: IEEE Computer Society, 2005: 160-168

[4] Beigi M, Devarakonda M, Jain R, et al. Policy-based information lifecycle management in a large-scale file system [C] //Proc of the 6th IEEE Int Workshop on Policies for Distributed Systems and Networks. Los Alamitos, CA: IEEE Computer Society, 2005: 139-148

[5] He Dingshan, Zhang Xianbo, Du D H C, et al. Coordinating parallel hierarchical storage management in object-based cluster file system [C] //Proc of the 23rd IEEE/the 14th NASA Goddard Conf on Mass Storage Systems and Technologies. Los Alamitos, CA: IEEE Computer Society, 2006

[6] Gibson T J, Miller E L. An improved long-term file usage prediction algorithm [C] //Proc of the 25th Annual Int Conf on Computer Management and Performance (CMG'99). Piscataway, NJ: IEEE, 1999: 639-448

[7] Cao Pei, Irani S. Cost-aware WWW proxy caching algorithms [C] //Proc of the USENIX Symp on Internet Technologies and Systems. Los Alamitos, CA: USENIX Association, 1997: 193-206

[8] Ari I, Gottwals M, Henze D. Performance boosting and workload Isolation in storage area networks with SANCache [C] //Proc of the 23nd IEEE/the 14th NASA Goddard Conf on Mass Storage Systems and Technologies. Los Alamitos, CA: IEEE Computer Society, 2006

[9] Ari I, Gottwals M, Henze D. SANBoost: Automated SAN-level caching in storage area networks [C] //Proc of the 13th IEEE Int Conf on Autonomic Computing. Los Alamitos, CA: IEEE Computer Society, 2004: 164-171

[10] Wong T M, Wilkes J. My cache or yours? Making storage more exclusive [C] //Proc of the USENIX Annual Technical Conf. Los Alamitos, CA: USENIX Association, 2002: 161-175

[11] Clemson University and Argonne National Laborary. PVFS2 project [EB/OL]. [2010-03-30]. http://www.pvfs.org

[12] Harish R. Design and Implementation of the System Interface for PVFS2 [D]. Clemson: Clemson University, 2002

[13] Roselli D, Anderson T E. A comparison of file system workloads [C] //Proc of the USENIX Annual Technical Conf. Los Alamitos, CA: USENIX Association, 2000: 41-54

[14] Roselli D, Anderson T E. Characteristics of file system workloads [R]. Berkeley, CA: University of California at Berkeley, 1998

[15] Joukov N, Wong T, Zadok E. Accurate and efficient replaying of file system traces [C] //Proc of the 4th USENIX Conf on File and Storage Technologies. Los Alamitos, CA: USENIX Association, 2005: 337-350

[16] Aranya A, Wright C P, Zadok E. Tracefs: A file system to trace them all [C] //Proc of the 3rd USENIX Conf on File and Storage Technologies. Los Alamitos, CA: USENIX Association, 2004: 129-145

[17] Ellard D, Seltzer M. New NFS tracing tools and techniques for system analysis [C] //Proc of the Annual USENIX Conf on Large Installation Systems Administration. Los Alamitos, CA: USENIX Association, 2003: 73-86

[18] Agrawal N, Bolosky W J, Douceur J R, et al. A five-year study of file-system metadata [C] //Proc of the 5th USENIX Conf on File and Storage Technologies. Los Alamitos, CA: USENIX Association, 2007: 31-45

Ao Li, born in 1983. Master. Her main research interests include network storage system.

Yu Deshui, born in 1983. Master. His main research interests include network storage system.

Shu Jiwu, born in 1968. PhD, professor and PhD supervisor. Senior Member of China Computer Federation. His main research interests include network storage, storage security, parallel processing technologies, and so on.

Xue Wei, born in 1974. Received his PhD in electrical engineering from Tsinghua University in 2003. His main research interests include power system analysis and simulation, parallel algorithm design, cluster computing and network storage.