第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)03—0025—03　　文献标识码：A　　中图分类号：TP311

基于对象存储的新型元数据管理策略

方 圆[1]，杜祝平[1]，周功业[2]

(1. 解放军信息工程大学信息工程学院，郑州 450002；2. 华中科技大学武汉光电国家实验室，武汉 430074)

摘 要：在对已有对象存储元数据管理策略进行研究的基础上，提出一种基于对象存储的新型元数据管理策略。该策略将命名空间的目录子树分割为等粒度的中子树，将中子树的根目录名和文件名的组合作为哈希参数进行哈希运算，元数据服务器根据其所得哈希值确定存储路径。实验结果表明，该策略在处理元数据重命名操作和修改文件名时，可以避免大量元数据迁移及网络开销问题。

关键词：元数据；对象存储；元数据管理；元数据服务器；负载均衡；热点

New Kind of Metadata Management Strategy Based on Object Storage

FANG Yuan[1], DU Zhu-ping[1], ZHOU Gong-ye[2]

(1. Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002, China;
2. Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China)

【Abstract】On the basis of the research of previous object storage metadata management strategies, this paper puts forward a new kind of metadata management strategy. The method is to divide namespace directory into neutron trees, and compute with hash function using the combination of the root directory name and file name. Metadata Server(MDS) determines its storage path according to the hash value. Experimental result shows that this new kind of metadata management tactics can avoid the massive metadata migration problems and network overhead problems in dealing with metadata rename operation and modify filename.

【Key words】metadata; object storage; metadata management; Metadata Server(MDS); load balance; hot spot

DOI: 10.3969/j.issn.1000-3428.2012.03.009

1 概述

目前，对象存储系统元数据服务器主要是对文件级元数据进行存储管理，而元数据具有本身数据小而数量多的特点，操作系统中 50%~80%[1]的文件操作与元数据息息相关，因此，元数据管理高效与否在一定程度上决定着存储系统性能的高低。本文对对象存储元数据管理策略进行研究，提出一种新型的元数据管理策略方法。

2 传统元数据分配策略

元数据作为数据索引信息，存放在元数据服务器上。元数据本身是很小的，但随着系统的增大，系统内的元数据会逐渐增多，怎样来提高元数据的检索速度，这就需要对元数据服务器(Metadata Server, MDS)中元数据的分配策略进行研究。只有元数据分配更加平均，才能使得集群资源得到充分利用，元数据服务器之间的负载达到均衡，从而提高系统的性能。

2.1 子树分割法

子树分割法分为静态子树分割法和动态子树分割法。静态子树分割法[2]需要管理员参与，预先把名字空间分割成不同的目录子树，然后再根据目录子树的结构来对元数据进行分布，在 MDS 集群中每个 MDS 单独管理各自负责的完整目录子树。

动态子树分割法是能够动态的将负载均衡分配到命名空间各个目录子树之间，并允许 MDS 之间对目录层次权限和元数据分区进行修改。缺点在于：在访问层次较深的元数据时，需要对整个层次树进行遍历，开销较大。重命名时产生大量的元数据迁移；子树重命名后很难实现元数据之间的动态平衡。

2.2 静态哈希算法

静态哈希算法[3]主要是以文件的全路径名或者目录路径节点作为哈希函数参量进行哈希运算来将文件元数据直接分布到不同的 MDS 上。其缺点是：无法很好处理目录重命名操作，因为当目录重命名后，MDS 集群间会有大量的元数据被重新分配，而且对一个目录下的文件进行访问时也会产生一些路径遍历和网络开销。

2.3 Lazy Hybrid 法

Lazy Hybrid(LH)法[4]是一种建立在路径名哈希映射基础上的可升级的元数据管理机制。其具体方法是使路径名生成散列值，以散列值为索引在元数据查找表(Metadata Lookup Table, MLT)中来查找元数据存放的位置。当 MDS 集群中有 MDS 增加或移除时，只需修改 MLT 上个别条目就可完成操作。其缺点在于：并没有真正做到分散目录重命名，链接和访问许可所带来的网络开销没有从根本上减少或消除，而且由于目录重命名操作带来大量的文件元数据迁移，导致系统需要的开销并没有减少，反而增加。

3 目录元数据分层管理策略

根据存储管理的机制可知，数据的访问需知道其文件属性，而文件属性又须先知道其目录路径属性，因为文件的访问不仅与根目录的元数据信息有关，还与目录路径中目录层

基金项目：国家“863”计划基金资助项目(2009AA01A402)

作者简介：方 圆(1980－)，男，硕士研究生，主研方向：元数据管理策略，网络与信息安全；杜祝平，副教授；周功业，教授

收稿日期：2011-07-05　**E-mail：**heeroyui01@163.com

次和访问权限有很大的联系。

3.1 目录元数据分层管理策略的设计方法

对象存储系统 MDS 中元数据主要分为目录元数据和文件元数据 2 种。本文结合元数据服务器内的元数据分割，提出一种新型的元数据分配策略，并根据这种策略来对目录元数据和文件元数据进行分层管理与存储，实现多元数据服务器来管理元数据[5]，从而避免单元数据服务器成为对象存储的“瓶颈”。本文把这种数据分层的管理的方法，称为目录元数据分层管理法(Directory Matedate Hierarchy Manage, DMHM)。

下面分 2 步对 DMHM 策略的方法进行描述：

(1)DMHM 策略元数据树的分割

DMHM 策略的运行必须进行目录子树分割，将目录元数据所在的目录子树分割为中等粒度的若干等子树，以它们的父路径名和文件名的组合进行哈希运算，并根据哈希值来分配元数据。这样能够使元数据均匀分布并能保存大部分目录路径的等级。

通过对目录子树的粒度进行参数设置来对 DMHM 策略进行说明：Depth 为节点的深度；Nodenum 为节点的度数；MDS_i 为存储该节点的元数据服务器：以度数为 2 深度为 3 的目录子树为例来对分割中粒度子树进行说明，如图 1 所示。

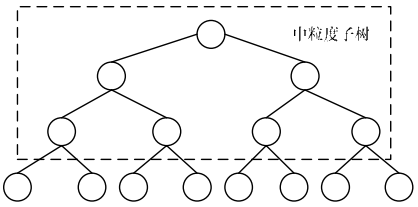

图 1 Tree Depth=3, Node Num=2 时的子树

假定在一个目录子树 A 中的节点 Y 下创建 X，如 X 是文件，将 X 存放在 Y 的 MDS 中；如 X 是目录，并且节点 Y 的深度和度数都未达到最大值，其深度和度数将自动增 1，X 存放在 Y 的 MDS 中，否则以父目录节点为 X 创建另一个子树。

(2)DMHM 策略中元数据的哈希运算

DMHM 策略为了避免更改路径名时，其目录路径的哈希值发生很大变化，在对目录子树进行完分割后，选取目录子树的父目录路径名和文件名(根目录名称)的组合为哈希参数来运行哈希算法。以粒度为 Tree Depth=2, NodeNum=2 的中粒度子树为例进行说明，如图 2 所示。

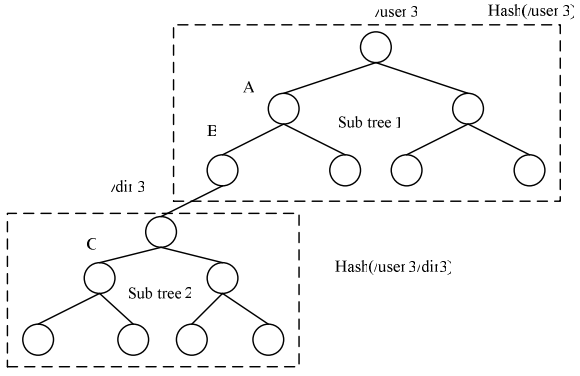

图 2 DMHM 策略分割子树

子树 Sub tree1 的绝对路径为/user3，同哈希路径放置算法相同，本文使用/user3 作为子树 Sub tree1 的哈希参数；子树 Sub tree2 的文件名为/dir3，因此子树 Sub tree2 的哈希参数取父目录与文件名称的组合为/user3/dir3。使用这种哈希策略，可以避免由于子树中非目录节点的名称变化引起的哈希值变化。

3.2 DMHM 策略的元数据更新

每个用户和元数据服务器都保存有一张 MTL 表[6]和路径树。MTL 表中存放元数据子树哈希值与元数据服务器之间的映射关系。路径树为整个文件系统中数据树的存放形式。当更新元数据服务器与元数据子树哈希值的映射时，服务器中的 MTL 表发生改变，更新后的 MTL 表将广播到各个元数据服务器中。MTL 映射表如表 1 所示。

表 1 MTL 映射表

Value	IP address of MDS
89	X.X.X.1
200	X.X.X.2
1 021	X.X.X.1
5	X.X.X.3

MTL 映射表设立一个版本号，当更新映射表后，通过广播消息将更新的映射表传递给各个元数据服务器。当客户端连接到元数据服务器时，若客户端的 MTL 映射表版本号与元数据服务器的不同，将获得新版本的 MTL 映射表。

4 元数据动态负载均衡策略

DMHM 策略中分割的子树根据其根目录的哈希值分配到元数据服务器中。因为系统的负载情况是动态的，所以本文采用元数据负载均衡策略[7-8]来调整元数据服务器之间的负载分配。假定更新间隔为 P，每个子树有一个存取计数 $Access_{counter}$，MDS_i 为元数据服务器 i 中子树的存取计数 $Access_{counter}$ 总和。再根据元数据服务器的性能给每个元数据服务器分配一个权值 w_i。

元数据负载均衡调整算法步骤如下：

Step1 根据下式计算每个元数据服务器的存取次数：

$$MDS_i = \sum_{subtree \in MDS_i} Access_{counter}$$

Step2 根据公式 $x_i = \frac{w_i}{\sum w_i} \sum MDS_i$ 计算各个元数据服务器的理想负载。

Step3 For i=0 to n

$P_i=X_i-MDS_i$

Step4 将{i|p_i<0}放入集合 Busy 中，{S|P_s>0}放入集合 Free 中。

Step5 For (i ∈ Busy && i != NULL) { //从 Busy 集中 i 逐个比较，直到集合为空。

$p_i=-p_i$

For (s ∈ Free && s ! =NULL) {

If (|p_i-p_s|>2%×x_i && p_i-p_s<0) {

在 Free 集合中寻找下一个 s;

break; //进入下一个循环 }

Else {

在 MDS_s 寻找一系列子树，使其 $ACCESS_{counter}$ 之和接近于 P_s;

将这系列子树从 MDS_s 移入到 MDS_i;

将 i 移出集合 Busy

首先计算每个元数据服务器的存取次数 MDS_i，并根据权值 W_i 计算每个元数据服务器的理想存取次数。间隔时间 T 后，根据算法，再调整子树在元数据服务器集群中的分布。

5 DMHM 策略热点问题

在多 MDS 系统中“热点”问题表现为客户的访问过于集中于某些个别元数据。

由于用户对数据的存取是无意识的，用户可能会同时或在很短时间内存取同一元数据。这样，就会有大批量请求同时请求同一元数据服务器服务，一些请求就会延迟，部分请

求会被系统抛弃。

为了监测出热点元数据，定义一个结构来记录活跃元数据的存取情况，如表 2 所示。

表 2　信息存取表

名称	符号
存取总次数	$Access_{total}$
被复制次数	$Number_{replica}$
访问间隔时间	T
单位时间内平均访问次数	$Access_{average}$

$Access_{total}$ 表示间隔时间内该元数据子树的总存取次数，$Number_{replica}$ 代表元数据的复制份数，T 表示记录时间。当存取元数据时，首选检查改元数据子树信息存取表，如果间隔时间 T 比定义的间隔时间 T_p 大，根据 $Access_{average}=Access_{total}/Num_{replica}$ 更新 $Access_{average}$，并置 $Access_{total}$ 为 0。

通过复制热点数据，并分布到其他元数据服务器中来消除热点数据。对热点数据分为下面 4 个处理状态：

(1)复制热点数据

当系统监测到某元数据子树的平均存取频率超过热点数据存取阈值 $P_{threshold}$，该元数据子树自动复制。

(2)元数据子树复件的分布

被复制的元数据子树按随机数分布到其他元数据服务器中，这种随即策略可以使得复制子树均匀分布，避免某些元数据服务器负载过重。

(3)存取元数据子树

元数据子树被监测为热点数据后，保存该元数据子树的复件分布列表，客户可以从列表中随机选取要服务的元数据服务器。

(4)删除元数据子树复件

当 $Access_{average}$ 低于热点数据存取阈值 $P_{threshold}$ 时，从元数据服务器中随机删除一个元数据子树复件，并更新该元数据子树的复件分布列表。

6　测试与实验

6.1　实验环境

由配置为 Pentium 4 1.7 GHz 处理器，2 GB 内存、160 GB 硬盘和 Fedora 9.0 操作系统的 5 台计算机组成的 NMSOFS 系统为实验环境，其中 1 台作为客户端、3 台作为元数据服务器、1 台作为对象存储设备。最后用 Iozine 作为测试工具进行测试。

6.2　测试目标

用 5 台计算机充当元数据服务器对 LH 法、静态子树法和 DMHM 策略进行元数据的分布、吞吐率、迁移率的测试，并对结果进行研究分析。

6.3　测试过程

限于篇幅，选取部分测试，对 DMHM 策略进行说明。在不考虑“热点”问题的测试元数据迁移率的实验中，随机对文件或目录名称进行更新，更新率在 1%~5%之间。

由图 3 可看出，静态子树法的迁移率为 0，这是由于静态子树法只对发生迁移的目录子树进行更新，其他目录子树上元数据不发生迁移，因此不存在变化。由于 LH 法使用哈希算法，因此更新目录名称时对象文件的哈希值将发生变化，从而导致大量元数据发生迁移。DMHM 策略不对子树的根目录进行更新，更新时分割后子树中需要变化的元数据很少，不需要大规模重新分布。当增加客户节点数时，系统每秒的操作吞吐率会发生变化，对 3 种策略进行吞吐率的测试。

如图 4 所示，静态子树法吞吐率最低，LH 法和 DMHM 策略在客户节点数比较小时吞吐率差别不大，随着客户节点的增多，DMHM 策略比 LH 法吞吐率逐渐增大，DMHM 策略更能提高系统的性能。

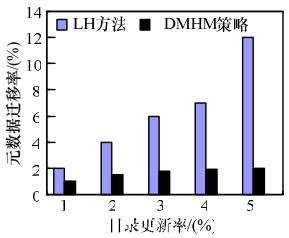

图 3　元数据迁移率

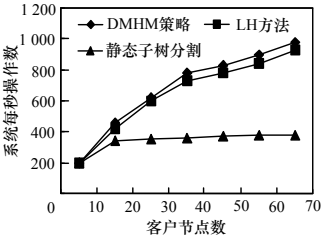

图 4　元数据吞吐率

以上实验结果表明，DMHM 在提高系统吞吐率、实现负载均衡以及减少元数据迁移等方面具有明显的优势。在有大量存储节点和用户的集群文件系统中，不仅对象存储器是可寻址的，元数据到元数据服务器也是可寻址的。

7　结束语

目前对象存储已成为存储技术的重要发展趋势，其元数据管理效率在一定程度上决定着存储系统的整体性能。本文在对已有的对象存储元数据管理策略进行比较研究的基础上，提出了一种新型的元数据管理策略。实验结果表明，DMHM 策略在集群系统中能均匀分布元数据，并根据负载均衡策略，在系统中迁移元数据和消除热点数据。在下一步的 DMHM 策略研究工作中，将重点针对其负载问题和大规模数据的测试进行更深入的研究。

参考文献

[1] Ousterhout J K, Costa H D, Harrison D, et al. A Trace-driven Analysis of the Unix 4.2 BSD File System[C]//Proceedings of the 10th ACM Symposium on Operating Systems Principles. [S. l.]: ACM Press, 1985.

[2] Morris J H. Satyanarayanan M, Conner M H, et al. Andrew: A Distributed Personal Computing Environment[J]. Communications Environment, 1986, 28(3): 184-201.

[3] Corbett P F, Feitelso D G. The Vesta Pacallel File System[J]. Transactions on Computer Systems, 1996, 14(3): 225-264.

[4] Brandt S A, Lan Xue, Miller E L, et al. Efficient Metadata Management in Large Distributed File Systems[C]//Proceedings of the 20th IEEE NASA Goddard Conference on Mass Storage Systems and Technologies. [S. 1.]: IEEE Press, 2003: 290-298.

[5] 刘　群, 冯　丹, 王　芳. 高可靠性元数据服务器研究[J]. 计算机工程, 2008, 34(17): 88-90.

[6] Mummert L, Satyanarayanan M. Long Term Distributed File Reference Tracing: Implementation and Experience[J]. Software-Practice and Experience, 1996, 26(6): 705-736.

[7] 李　登. 分布式系统负载均衡策略研究[D]. 长沙: 中南大学, 2002.

[8] 王广芳. 分布式计算机系统(DCS)负载均衡算法 20 年[J]. 计算机工程与设计, 2002, 16(5): 58-63.

编辑　索书志