第31卷 第20期 计 算 机 工 程 2005年10月
Vol.31 № 20 Computer Engineering October 2005
·软件技术与数据库· 文章编号：1000—3428(2005)20—0077—03 文献标识码：A 中图分类号：TP311.11

一种基于对象存储中的元数据组织管理方法

谈华芳[1,2]，孙丽丽[1,2]，侯紫峰[3]
(1. 中国科学院计算技术研究所，北京 100080；2. 中国科学院研究生院，北京 100039；3. 联想研究院，北京 100085)

摘 要：提出了一种动态分区元数据组织管理方法。它混合了动态和静态的方法在 MDS 机群中分布元数据，并使用散列的技术索引元数据，利用共享存储来存放元数据。整个方法使得元数据访问可以高效地完成，机群的失败接管和扩展获得好的性能。
关键词：基于对象存储系统；元数据管理；动态分区

A Method of Metadata's Organization and Management in Object-based Storage

TAN Huafang[1,2], SUN Lili[1,2], HOU Zifeng[3]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080; 2. Graduate School of the Chinese Academy of Sciences, Beijing 100039; 3. Lenovo Corporate Research & Development, Beijing 100085)

【Abstract】 This thesis presents a new method called dynamic hashing partition (DHP). It mixes the dynamic and static methods to distribute the metadata in the MDS cluster, uses hash technology to index metadata and employs share common storage space to store the metadata. The whole method makes the MDS cluster to achieve good performance of MDS cluster failover and scalability.
【Key words】 Object-based storage; Metadata management; Dynamic hashing partition (DHP)

1 概述

当前，计算机系统中比较流行的网络存储系统是 NAS 和 SAN。在 NAS 中，虽然提供了数据共享和安全能力，但所有请求都要经过服务器，因此当客户端数目或来自客户端的请求较多时，NAS 服务器将成为瓶颈。在 SAN 中，虽然它提供了高速、直接的访问，但是它的共享能力有限，安全性差强人意。因此人们提出了结合两者优点的基于对象存储系统。

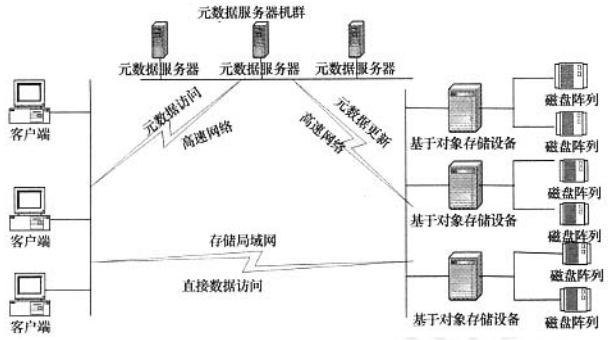

图 1 分布式对象存储系统架构

在一个大的分布式基于对象的存储系统中，如图 1 所示，元数据和数据分开管理。基于对象的存储设备（OBSDs）管理低水平的存储任务，对外呈现对象访问接口。一个分离的 MDS 机群管理元数据的操作和文件到对象的映射，并执行安全策略策。在这样的存储系统中，通过使用专门的 MDS 机群管理元数据来提升整个系统的性能；并且在数据存放和获取时，不需要引入 MDS，允许在 OBSDs 和客户端之间十分高效地进行并发的数据传输。在这种存储架构下，MDS 不需要执行任何的数据操作。整个系统将不能再依赖数据的移动隐藏元数据操作的开销。元数据操作将会存在于关键路径上。所以，为了保证整个存储系统的性能，防止出现瓶颈，元数据的处理负载在整个 MDS 机群中的分布必须相对均匀。并且当出现负载不平衡时，很容易进行动态调整。最后，当整个机群过载时，系统可以很容易加入新的 MDS 进行扩展。

对于一个分布式系统来说，可用性也是十分关键的问题。为了保证整个系统的连续可用性，不允许任何一个 MDS 的失败影响整个系统的运行，因此元数据的组织管理必须保证可以很容易地实现失败接管。

本文提出了一个新的称为动态散列分区的方法来管理组织元数据。它使得 MDS 机群可以获得高效的元数据访问、动态负载平衡、失败接管和扩展。

2 相关研究

（1）目录子树分割法[1]：这种方法把名字空间分割成不同的目录子树，每一棵目录子树被一个 MDS 管理。目录子树分割法提供了一个自然的方法来分割名字空间。它的一个优点是当一个特定的 MDS 存放一个已给文件的路径上的一些或所有目录时，对一个文件元数据的访问可以通过联系相对少的元数据服务器来完成。但是这个方法在单个文件、目录，或者目录子树成为热点时，将遭受严重的瓶颈问题。

（2）纯散列分割法[2]：它通过基于文件标识、文件名或者其它相关值的散列来广泛地在 MDS 机群中分布元数据。这种方法与目录子树分割法相比，带来了更均衡的工作负载分布。如果散列基于完整的路径名，则元数据将会完全随机分布。但是，当一个目录重命名时，则大量的元数据不得不被移动。如果散列仅仅使用文件名（代替完整的路径名），则在不同目录下同名的文件将会散列到相同的 MDS 上。当

作者简介：谈华芳（1977－），女，博士生，主研方向：分布式文件系统和存储系统；孙丽丽，硕士生；侯紫峰，博导、研究员
收稿日期：2004-09-07 E-mail：tanhf@lenovo.com

并发访问不同路径下的同名文件时，有可能形成瓶颈。当增加 MDS 到机群中或从机群中移出 MDS 时，上述两种方法都会遇到困难。一定数量的不成比例的元数据将不得不被移动。

（3）Lazy Hybrid (LH) 元数据管理方法[3]：LH 使用路径名生成散列值，以散列值为索引在元数据查找表（MLT）中来查找元数据存放的位置。这样当增加和移出 MDS 时，只需要修改 MLT 中的某个或某几个条目即可。LH 使用一个唯一的双入口访问控制列表结构允许文件的访问许可被直接决定，这样与路径名散列相结合可以不需要遍历整个路径，就可以访问元数据。但是如果要与类 Unix 客户端的虚拟文件系统兼容，则这种优势没法体现。因为在类 Unix 虚拟文件系统中，查找一个文件是按照路径名一级一级往下遍历进行的。并且最主要的是，虽然它使用了懒惰更新和重配置的方法把目录重命名，链接和访问许可改动而引入开销的分散，但是所需的开销并没有减少，反而增加了。

（4）散列分割法（HAP）[4]：它也采用了散列的方法来分布元数据，但是它集中在减少元数据在 MDS 之间移动。这种方法的优点是可以很容易地实现机群系统的负载平衡、失败接管和可扩展性。因为数据存放在共享存储上，所以进行上述操作时不需要移动元数据。而且有了逻辑分区这个概念使得元数据分割和实现元数据二级映射更加容易。

但是这种方法的缺点也是明显的：1）同名的文件会散列到相同的逻辑分区上，由同一个 MDS 来管理，当并发访问不同目录下同名的文件时会带来瓶颈；2）如果修改目录名，则不光需要改变当前被修改的目录的元数据存放的位置，而且以此目录为根的子树上所有文件的元数据在逻辑分区中的位置都需要发生改变，因此引入了很大的开销；3）逻辑分区个数和每个逻辑分区的存储区域是固定的。因此随着系统的发展创建新的逻辑分区时，需要增加新的存储硬件存放映射到新分区上的元数据，并且需要在整个机群范围内重新分布元数据，带来了巨大的开销。

（5）动态绑定访问[5]：它使用动态的方法来映射元数据到 MDS 上，并且在系统运行过程中，可以根据各个 MDS 的负载情况进行动态负载平衡。

这种方法的缺点是：1）绑定服务器会成为单一故障点，随着系统规模的扩大，也将会是一个瓶颈；2）如果客户端是第一次访问元数据或者是负载平衡后的首次访问元数据，客户端没法知道当前的元数据由哪台 MDS 管理，则请求会发给错误的 MDS，因此在 MDS 间需要转发元数据的访问请求，使得请求的延迟增大，并且 MDS 的额外负载增加。

3 动态散列分区

动态散列分区先根据文件的路径信息和文件名使用静态散列的方法对元数据实现静态分区，然后把分区动态地映射到 MDS 上。但是这个方法与 HAP 方法是不同的，分区时使用的信息保留了文件在目录树中位置信息，不会让不同目录下同名的文件映射到相同的分区，带来潜在的瓶颈。而且由于我们选择的表示位置的信息具有特殊性，不会像 LH 那样在目录重命名时带来巨大的开销。并且逻辑分区会随着系统的发展动态分裂和合并，不会像 HAP 那样扩展机群时会受逻辑分区个数的限制，需要重建。为了获得元数据的有效查找，我们选用了一种好的散列索引结构，且为了减少元数据的移动，把元数据存放在所有 MDS 都可以访问到的共享存储上。

3.1 元数据到 MDS 的映射

元数据到 MDS 的映射采用两级映射的方法，先把元数据映射到一个逻辑分区，然后逻辑分区再根据一定的策略映射到 MDS。元数据到逻辑分区的映射使用静态散列映射的方法，而逻辑分区到 MDS 上的映射则根据当前 MDS 的负载情况和处理能力使用动态映射的方法。为了避免预置的逻辑分区个数限制机群的扩展，也为了在一个逻辑分区内能有效地查找元数据，逻辑分区会随着它管理的元数据对象的增多动态分裂。当然，当分区中包含的元数据对象很少时，也会进行合并。那样元数据到逻辑分区的映射关系会改变，但是这种映射关系的改变并不经常发生，因此元数据到逻辑分区的映射是相对固定。

3.1.1 元数据到逻辑分区的映射

元数据到逻辑分区的映射是由文件名加父目录唯一标识的散列结果来决定。这个散列结果又称为全局标识，用 GlobalId 表示；唯一标识就像本地文件系统的索引节点号一样，在一个文件系统中唯一地标识一个文件/目录，用 ExclusiveId 表示。逻辑分区号用 LogPid 表示。方程 1 描述了这个映射函数：

$$\text{GlobalId}=H(\text{filename}, \text{ExclusiveId}_{\text{parent}})$$
$$\text{LogPid}=f(\text{GlobalId}) \quad (1)$$

这里，H 代表散列函数；f 代表转换全局标识到逻辑分区号的映射函数。

我们使用了文件名加表示文件在目录树中位置信息的父目录唯一标识来进行散列映射。由于唯一标识在整个名字空间中是唯一的，只要选择好的散列函数和映射函数，元数据就会完全均匀地分布在不同的分区上。并且不会随着目录名字的改变而改变，因此目录重命名不会引入开销。

但是这种映射方法与 LH 和 HAP 相比，唯一不好的地方是，访问元数据时需要在客户端遍历路径名上的目录。为了减少这种开销，把全局标识铸造在文件句柄中，这样，一旦获得文件的句柄后，就可以直接访问元数据了。所以只是在首次访问时，速度会比较慢。而且，由于在类 Unix 客户端的接口方式是被嵌入在类 Unix 的虚拟文件系统接口中的，因此它的这个不足就被掩盖了。

3.1.2 逻辑分区到 MDS 的映射

逻辑分区到 MDS 的映射是一个动态映射，用 MDSId 表示 MDS 号，MPWi 表示 MDSId＝i 的 MDS 的处理能力，用 MLWi 表示 MDSId＝i 的 MDS 的当前的负载能力。方程 2 描述这个映射关系：

$$\text{MDSId}=\text{ML}(\text{LogPid}, \text{MPWi}, \text{MLWi}) \quad (2)$$

其中，$i \in \{0,Mn\}$，Mn 为当前总的 MDSs 个数，ML 代表映射函数。如果各个 MDS 的处理能力相同，则整个映射关系可以简化为：

$$\text{MDSId} = \text{ML}(\text{LogPid}, \text{MLWi}) \quad (3)$$

对于 MDS 的处理能力，可根据 MDS 的 CPU 主频、内存的大小以及带宽决定。MDS 的负载情况可根据 CPU、内存使用率来决定，也可以由 MDS 上的元数据访问率决定。在某一时刻的逻辑分区到 MDS 的映射关系可以由一个元数据查找表（MLT）来描述。表 1 给出了一个 MLT 的实例，为了访问在逻辑分区 33 上的元数据，客户端应该发请求给 MDS2。

表 1 一个 MLT 的实例

逻辑分区号	MDSID
0~15	0
16~31	1
32~47	2
48~63	3

当下列任何情况发生时，这个映射关系会发生改变。

（1）元数据请求负载发生改变导致某个 MDS 上负载过重。

（2）某个 MDS 失效或需要进行下线维护、升级。

（3）有新的 MDS 加入机群。

（4）逻辑分区进行动态分裂或合并。

根据上述映射方法，当由于文件的访问频率不断变化使得某个 MDS 上的负载过重时，这个 MDS 上的一部分逻辑分区会被转移到负载相对较轻的 MDS 上来管理。对于失败接管的处理实现起来，也将会很简单、有效。不需要一个备用的服务器来接管某个失败的 MDS 的工作，而是使用机群化的方法实现。当一个 MDS 失败时，整个机群根据当前各个活动的 MDS 负载的情况把由失败的 MDS 管理的几个逻辑分区分配给其它的 MDSs 来管理即可。当 MDS 机群由于负载太重不能有效地处理元数据请求时，新的 MDS 可以被动态加入进来，然后一部分逻辑分区就会转移到新的 MDS 上。

3.1.3 逻辑分区动态分裂、合并

当逻辑分区中包含的元数据对象达到一定规模后，无论采用什么样的索引结构，查找的性能都会下降。并且如果逻辑分区个数固定，则机群的扩展性会受到限制。因为一个逻辑分区在一个时刻只能映射到一个 MDS 上。因此我们引入了动态逻辑分区分裂。当一个逻辑分区包含的元数据对象数目超过一定阈值时，自动把逻辑分区一分为二。在这个过程中需要修改映射函数 f，把原分区中一部分元数据映射到新的分区，并且需要修改 ML 把一个新创建的逻辑分区加载到一个 MDS 上。为了实现简单，往往用一个逻辑分区查找表来描述元数据到逻辑分区的映射，当一个分区动态分裂后，只需要修改表中与原分区相关的条目，并增加描述新分区的条目即可。并且当逻辑分区由于删除操作导致元数据对象减少到一定程度时，可以把两个逻辑分区合并成一个分区。

3.2 元数据的有效查找

通过上面的映射方法，可以把一个元数据对象映射到一个逻辑分区。但是，根据元数据映射方法，元数据访问中的局部性被牺牲了。为了支持在一个逻辑分区内部元数据对象的有效查找，需要有一种先进的索引结构支持。并且为了获得好的查找性能，需要足够的内存来让索引表驻留在内存中。基于散列的索引和基于树的索引是两个主要的索引结构。由于 B 树索引具有 O(logn)的查找复杂性和 50％的空间使用率。而散列表具有 O(1)的查找时间复杂度和比 B 树索引更高的空间使用率，因此散列表更合适组织元数据对象。

把一个逻辑分区内的元数据对象使用密集索引的方法进行组织管理。元数据对象记录被顺序地登录在内容文件中，包含相同全局标识的元数据对象记录用链接的形式把它们连在一起；而元数据对象记录所在的地址、全局标识和唯一标识作为索引项被登记在索引表中。用按桶散列的方法组织这个索引表。在一个逻辑分区中的元数据查找过程如下：

（1）根据全局标识，获得索引项所在的桶号，BucketId= BH(GlobalId)。BH 代表索引项到桶的映射函数。

（2）根据 BucketId 找到对应的桶。如不是第一访问，则请求中包含唯一标识，转入（3）；否则转入（4）。

（3）在桶中顺序查找，找到全局标识和唯一标识相匹配的项，获得元数据对象的地址 addr。使用 addr 访问内容文件中的元数据。

（4）在桶中顺序查找，找到全局标识相匹配的项，获得元数据对象的地址 addr。

（5）使用 addr 读出内容文件中的元数据对象记录。

1)如果记录中包含的父目录唯一标识和文件名信息与请求中相符，则返回。

2)若不符，则令 addr 为记录包含的下一个相同全局标识的记录链接地址，转入（5）继续查找。只要记录中的链接地址不为空。

因此如用 ln 表示一个桶中包含的索引项，用 lm 标识内容文件中包含相同全局标识值的链接表的长度，则初次查找并且命中的探测次数（与索引项和元数据对象记录比较的次数）：ln/2＋lm/2；初次查找不命中的探测次数：ln＋lm；非首次访问带唯一标识的查找命中和不命中的探测次数分别为：ln/2，ln。可看到初次访问的探测次数由指标文件中对应的桶的容量和散列到相同全局标识冲突的概率决定，而带唯一标识的访问，则完全由桶的容量决定。所以应该选择一个合适的散列函数 H，让全局标识的冲突尽量减少，从而提高初次访问的效率；并且控制每个桶的容量。由于逻辑分区可以动态分裂，因此当每个桶内的记录数增多时，可扩充桶数，把每个桶分成两个桶。新生成的桶由新创建的分区来管理。

3.3 元数据的存储管理

通过对元数据查找的过程可以看出，元数据的存储分为两部分：用于索引的索引表和存放真正元数据对象的内容文件。我们把这两个文件分开存放在 MDS 机群共享的存储上。在逻辑分区建立时，在共享存储上申请足够的空间来存放索引表，然后由这个分区以独占的方式访问这块空间。而对于内容文件，在写入时才进行块分配，内容文件所在的空间可以被所有的分区访问。这样，当逻辑分区发生分裂时，只需要移动一部分索引表的内容，而内容文件不需要任何的移动。并且当逻辑分区的映射发生改变时，不需要移动任何数据。

4 结论

本文在分析了现有元数据组织管理方法优缺点的基础上，提出了一种动态散列分区的元数据组织管理方法。这种方法结合了动态和静态的元数据分布方法，使得元数据负载可以均匀地在机群中分布。并且我们考虑了系统的动态变化，让逻辑分区，这个机群进行负载平衡，动态扩展和失败接管的最小管理单元，可以动态地分裂和合并，满足了机群系统随着存储系统的发展可以自适应调解的需求。而且我们提供的索引结构，可以让系统高效地查找元数据对象，提高系统的性能。最后我们使用共享存储来存放元数据能减少元数据的移动，避免了在机群进行负载平衡，动态扩展和失败接管时的大量的元数据移动开销。

参考文献

1 Morris J H, Satyanarayanan M, Conner M H, et al. Andrew: A Distributed Personal Computing Environment. Communications of the ACM, 1986, 29(3): 184–201

2 Corbett P F, Feitelso D G. The Vesta Parallel File System. ACM Transactions on Computer Systems, 1996,14(3): 225–264

3 Brandt S A, Xue Lan, Miller E L, et al. Efficient Metadata Management in Large Distributed File Systems. In: Proceedings of the 20th IEEE /11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003-04:290–298

4 Yan Jie, Zhu Yaolong, Xiong Hui. A Design of Metadata Server Cluster in Large Distributed Object-based Storage. In: 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies, 2004-04:13-16

5 田 颖，许 鲁. 分布式文件系统中的负载平衡技术. 计算机工程，2003,29(19):42-44