

基于共享存储池的元数据服务器机群的设计研究

苏勇^{1,2}, 周敬利¹, 余胜生¹, 姜明华^{1,2}, 刘钢¹

¹(华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

²(武汉科技学院 计算机科学学院, 湖北 武汉 430073)

E-mail: suyong527@hotmail.com; sy@wuse.edu.cn

摘要: 在大型分布式对象存储系统中, 元数据服务系统是一个潜在的访问瓶颈。本文提出一种通过分层式文件系统构建共享存储池的采用两次分布式哈希函数方式的元数据服务系统。其具有不需要人工干预的故障恢复性和易扩展特性, 而且最大程度减少了MDS之间大量元数据的物理迁移。实验测试证明系统具有良好的I/O性能。

关键词: 元数据服务器; 共享存储池; 哈希; 对象存储

中图分类号: TP393

文献标识码: A

文章编号: 1000-1220(2007)04-0734-04

Research and Design of MDS in Distributed Storage System

SU Yong^{1,2}, ZHOU Jing-li¹, YU Sheng-sheng¹, JIANG Ming-hua^{1,2}, LIU Gang¹

¹(Department of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China)

²(Department of Computer Science, Wuhan University of Science and Engineering, Wuhan 430073, China)

Abstract: In the large-scale distributed object-based storage system, the metadata service system is a latent visit bottleneck. This paper introduces a metadata service system which constructs a shared storage pool by hierarchy file system and is based on two distributed hash. It has characters of failure- recovery without manual intervention and easy expansibility. What's more, it reduces the physical migration of massive number metadata between MDS. The experiment tests proves the system that have good I/O performance.

Key words: metadata server; shared storage pool; hash; object-based storage

1 引言

随着日新月异的应用对存储技术的要求, 近几年有了一种新兴的存储技术—基于对象存储^[1,2] (Object-Based Storage, OBS), 它把对象作为直接存储的基本单位, 从而像块一样提供良好的存储性能, 又因为它为每个对象提供了操作的接口, 所以可以像文件那样支持跨平台访问。在对象存储系统中分离了数据和元数据的管理, 不像传统的文件存储系统, 其数据和元数据都由同一台机器存储和管理。但是虽然相对于整个系统的存储量, 元数据的存储量是较小的, 但是元数据的访问量占整个系统的访问量的50%~80%, 因此对元数据的访问是一个潜在的瓶颈, 只有提供高性能的元数据访问服务, 才能提高整个存储系统的性能。

在对象存储系统中, 其元数据访问一般由多个元数据服务器组成的元数据服务器 (Meta Data Server, MDS) 机群系统, 如图1所示。这种方式的好处在于通过多个MDS的协同工作, 能够提供较好的元数据访问服务。但是目前这种集群元数据服务系统存在以下问题:

(1) MDS机群系统一般通过目录子树分配方法, 把存储

系统的目录树元数据分配到每个MDS上^[3]。这是一种静态的元数据分配方法, 其最大的缺点是当某个文件或目录成为访问热点时, 将使访问某单个MDS成为严重的瓶颈。

(2) 当MDS机群系统进行负载均衡, 或者需要增加MDS进行扩展时, 会在MDS之间产生大量的元数据移动, 这对整个存储系统性能有非常大的影响。

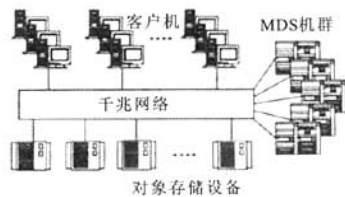


图1 基于对象存储系统

Fig.1 Object-based storage system

本文主要针对以上问题设计了一种基于共享存储池的元数据服务系统 (Shared Storage Pool Metadata Service Sys -

收稿日期: 2006-02-21 基金项目: 国家自然科学基金项目 (60373088) 资助; 国家重点实验室项目 (51484040504JWO518) 资助。 作者简介: 苏勇, 男, 1975年生, 博士研究生, 讲师, 主要研究方向为网络存储; 周敬利, 女, 1946年生, 教授, 博士生导师, 主要研究方向为计算机多媒体网络通信, 高性能网络接口和计算机网络存储; 余胜生, 男, 1946年生, 教授, 博士生导师, 主要研究方向为多媒体系统技术、计算机接口技术; 姜明华, 男, 1966年生, 博士研究生, 主要研究方向为网络存储; 刘钢, 男, 1975年生, 博士研究生, 主要研究方向为网络存储。

tem,简称 SMDS),它采用由网络存储器构建共享存储池,并且元数据采用两次 Hash 方式进行分布式存储和访问.采用共享存储池的元数据存储,可以有效避免在当 MDS 进行负载均衡时和增加 MDS 进行扩展时造成的大量的元数据在 MDS 之间的移动;元数据采用 Hash 方式进行分布式存储和访问,可以有效避免当某个目录成为访问热点时造成的单个 MDS 成为严重的瓶颈.

2 系统设计

SMDS 分为三层模型,由高到低分别是映射管理器(Mapping Manager,MM)、逻辑分区管理器(Logical Partion Manager, LPM)和公共存储池(Common Storage Pool, CSP),如图 2 所示.

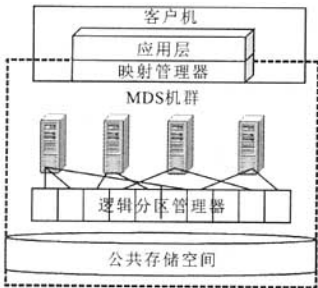


图 2 SMDS 模型图

Fig.2 SMDS model

公共存储池被划分成若干个逻辑分区,并且加以编号;每个 MDS 由逻辑分区管理器控制只能加载和访问其指定的逻辑分区;映射管理器主要由分布式 Hash 方式实现对象文件元数据到逻辑分区和 MDS 的映射.

在 SMDS 系统中元数据访问过程如下:首先,映射管理器将对象文件名通过 Hash 函数得到一整数值,这个值将映射到一个保存有对象文件的元数据的逻辑分区号;接着,映射管理器根据这个逻辑分区号映射到唯一的 MDS,然后客户端将直接向该 MDS 发送含有对象文件路径 Hash 值的查询请求;最后 MDS 通过逻辑分区管理器访问公共存储池中的逻辑分区,将请求的元数据返回给客户端^[4].

2.1 公共存储池

共享公共存储池是由高速网络和网络存储器构建,如 NAS(Network Attached Storage)等设备.并且将公共存储池划分成若干个逻辑分区.每个逻辑分区存储全局元数据中的一部分,并且唯一的分配给一个 MDS 授权访问.MDS 通过逻辑分区管理器接口只能访问其托管的逻辑分区.

2.2 逻辑分区管理器

逻辑分区管理器采用分层式文件系统技术设计,由提供虚拟统一 IO 空间的 UniformIO_ FS 和提供负载均衡的 LoadBalance_ FS 两个模块组成如图 3 所示.

为了访问某一物理文件系统(如 UFS、NFS 等),进程发出的系统调用请求先被翻译为 VFS 调用,然后 VFS 再将调用请求发给适当的物理文件系统.此时 VFS 被认为是上层

文件系统,而物理文件系统被认为是下层文件系统,上下两层文件系统的调用是通过虚拟节点接口(Vnode Interface)实现的.虚拟节点(Vnode)的提出促进了文件系统功能模块化,并产生了堆叠式虚拟节点(Stackable Vnode).

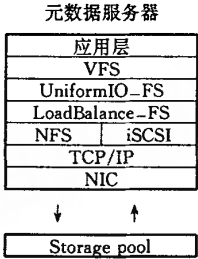


图 3 分层式文件系统模型

Fig.3 Hiberarchy file system model

分层式文件系统是一种面向模块的文件系统,它所提供的是虚拟节点对象,虚拟节点是一个抽象的对象,代表了存放在文件系统中的对象.作为对象,虚拟节点具有属性和行为特征,属性通常指数据变量,行为指对数据变量进行的操作.分层式文件系统具有很强的模块性,因此具有可扩展性;它在开发时容易调试,而且不需要对物理文件系统未作任何修改;分层式文件系统作为系统模块加载到系统的核心层,而不是用户层,但是它不像 UFS 与底层的设备驱动过于紧密结合而导致使得开发和移植的难度很大的缺点,并且同时也避免了用户层文件系统过多的上下文切换和数据拷贝降低了该类文件系统的性能缺陷.

UniformIO_ FS 模块主要是对被授权的逻辑分区进行用户视图的逻辑路径到网络存储设备上的物理路径的映射关系,提供虚拟统一 IO 空间.每当一个新的客户端访问 MDS 机群时,就从共享的网络存储设备中生成用户目录树,并将此用户目录树的根指针存放在用户队列中当前用户的节点中.用户任务处理线程每收到一个用户的请求,都调用 UniformIO_ FS 进行逻辑路径到物理路径的转换,然后按照物理路径访问 NAS 存储设备,将请求的结果通过高速 IP 通道直接返回给用户.

LoadBalance_ FS 模块主要是负责解决 MDS 机群中的负载均衡问题,其工作原理将在第三节详细介绍.

2.3 映射管理器

映射管理器主要解决对象文件与其元数据的映射问题.为此,必须要知道两方面的信息:元数据由哪个 MDS 管理;以及元数据在逻辑分区中的定位.映射管理器通过两次 Hash 求值得到这两种信息:

- ① 首先对对象文件名的 Hash 求值,映射到逻辑分区号,再查询元数据查询表(Metadata Look-Up Table,MLT)可获得 MDS ID,从而知道是哪个 MDS 管理元数据;
- ② 然后将路径 Hash 求值用来在 MDS 上定位所需的元数据.

映射管理器的工作原理可用下列公式(1)和(2)描述:

$$P_i = f(H(\text{filename})) \quad P_i \in \{1, P_n\} \quad (1)$$

H 表示对文件名的 Hash 函数, f 表示对 Hash 值进行逻辑分区映射, P_i 表示逻辑分区号, P_n 表示逻辑分区总数.

$$\text{MDS}_i = \text{mlt}(P_i) \quad \text{MDS}_i \in \{1, \text{MDS}_n\} \quad (2)$$

mlt 表示对逻辑分区 P_i 映射到某台元数据服务器 MDS_i , 它是一个容量较小并且较少更新的全局表, 保存着 Hash 映射值与 MDS ID (MDS 机群中每个 MDS 用 ID 编号) 的映射, 其功能主要是通过 mlt 查询实现, MLT 表结构见表 1.

表 1 元数据查询表示例
Table 1 Example of metadata lookup table

逻辑分区号(P_i)	元数据服务器(ID)
0-10	0
11-20	1
21-30	2
31-40	3

在 SMDS 中, 客户端通过对文件名的 Hash 求值映射到目的 MDS ID, 然后只向一个目的 MDS 发出元数据的查询请求, 这符合目前 MDS 机群中普遍采用 "a single message to a single metadata server"^[4] 的原则

比如客户要访问一个文件 "/a/b/filec", 那么客户首先对 "filec" 进行 Hash 求值来判断在 MDS 机群中由哪个 MDS 管理其元数据, 然后再对 "/a/b/filec" 进行 Hash 求值用来在 MDS 上定位其元数据. 另外对于某些常用的文件名, 如 "help"、"setup" 等, 其 Hash 值必须进行处理, 要使其尽量分布到不同的 MDS 上, 这样可避免不同目录下的常用文件名的文件高频率的访问造成的瓶颈. SMDS 的处理方法是对常用文件名的文件在选取 MDS 时, 不是用文件名进行 Hash 映射而是用路径进行 Hash 映射, 这样可避免不同目录下常用文件名的文件元数据集中在某台 MDS 造成瓶颈.

3 SMDS 的负载均衡设计

SMDS 主要是为大量客户端提供并发的元数据读写操作, 而每一个 MDS 同时接受客户端访问的数目是有限的, 并且每个 NAS 存储设备的容量也是有限的, 所以我们要考虑负载均衡的问题. 在 SMDS 系统中负载均衡涉及两个方面: 1. MDS 的负载均衡; 2. 逻辑分区的负载均衡. 由于元数据是通过文件名 Hash 来选择了逻辑分区存储, 而一个 MDS 要负责若干个对逻辑分区元数据访问请求管理, 因此 MDS 形成瓶颈的概率要比逻辑分区大得多. 基于这个原因, 我们设计 LoadBalance_FS 模块主要对 MDS 进行动态负载均衡. 其工作原理可用下列公式 (3)、(4) 来描述:

$$\text{MPW}_i = \sum_{i=1}^{M_n} \text{PW}_i / M_n \quad (3)$$

$$\text{PW}(i+1) = \text{PW}(i) * \alpha + \text{PW}_{\text{current}}(1-\alpha) \quad (4)$$

其中 MPW_i 表示 ID 为 i 的 MDS 的负载; M_n 表示 MDS 的总数; PW_i 表示逻辑分区 i 的负载, 负载以其被访问的频率

作为参数; $\text{PW}(i)$ 表示在 i 时刻时逻辑分区的负载, $\text{PW}_{\text{current}}$ 表示在当前时刻的逻辑分区的负载, α 为 0 到 1 之间的固定参数, α 用来平衡逻辑分区负载的旧值和当前值对负载新值的影响.

在动态负载均衡实现中, $\text{PW}_{\text{current}}$ 由 LoadBalance_FS 负责统计, 然后通过公式 (4) 计算 PW_i , 再用公式 (3) 计算各个 MDS 的负载. 当每个 MDS 负载相当或者 MDS 负载没有超过设置的极限值时, 则不需要进行负载均衡, 否则则要对 MDS 托管的逻辑分区进行重新分配, 分配原则按照公式 (1) 进行, 然后刷新 MLT. MLT 由 MDS 机群中一台特别的 MDS 维护, 并且及时的将每次更新的 MLT 分发给客户端.

4 SMDS 的故障恢复和扩展性

传统的 MDS 机群一般采用元数据直接存储在 MDS 上, 这样一旦某台 MDS 发生故障或 MDS 机群负载超过阈值需要扩展时, 必然需要将原 MDS 的元数据重新分配到新增加的 MDS 上, 这是一种 MDS 之间元数据的物理迁移, 这对存储系统的整体性能会有较大的影响.

而 SMDS 采用的是由网络存储设备构成的公共存储空间, 因此当某台 MDS 发生故障时, 其原先托管的逻辑分区将重新进行映射, 映射按照公式 (3) 进行, 由其它的 MDS 接管它的逻辑分区, 这样 MDS 机群不会因为某台 MDS 的故障而不能正常工作. 并且这种故障恢复是不需要人工干预, 而且也没有 MDS 之间元数据的物理迁移的.

在扩展性方面存在两种情况:

- ① MDS 机群处理元数据请求的负载超过某一阈值时.
- ② 元数据存储超过公共存储空间的阈值.

以上两种情况都需要对 MDS 机群进行扩展, 但是扩展的方式不同. 在第一种情况下 SMDS 可以通过增加 MDS 来解决, 增加 MDS 后仍然只需要按照公式 (3) 进行重新映射, 分担元数据请求负载, 而且也不需要 MDS 之间元数据的物理迁移. 但是在第二种情况下, 就需要增加网络存储设备对 SMDS 的扩展, 而且这种扩展需要对新增加的逻辑分区进行元数据的物理迁移, 为了避免这种情况需要在构建 SMDS 时做好规划, 在成本允许的前提下尽量保证有较大的公共存储空间.

5 性能测试和分析

系统的测试实验环境如下: 两台 Xeon-2G Hz CPU、Linux 7.1 操作系统、512M 内存、IBM 60G 硬盘、AGE -1000SX 网卡的主机作为 MDS; 一台 Pentium4 1.6G CPU、Linux 7.1 操作系统、256 内存、AGE -1000SX 网卡的主机作为客户机; 两台 Pentium4 2G CPU、Linux 7.1 操作系统、1G 内存、Seagate 80G 硬盘、AGE -1000SX 网卡的 NAS 作为网络存储设备.

测试中采用文件系统测试软件 Bonnie+ 对 MDS 分两种情况测试: 一是测试客户机通过 MDS 的 UFS 文件系统访问 MDS 本地磁盘; 二是测试客户机通过 MDS 的 NFS 文件

系统+UniformIO_FS和LoadBalance_FS两个模块访问NAS存储设备.我们把第一种测试设为A组,第二种测试设为B组.

测试时考察客户机在MDS文件系统内核层加载UniformIO_FS和LoadBalance_FS两个模块访时访问NAS和不加载两模块时访问MDS本地磁盘,对数据访问性能的比较. Bonnie++ 测试是在随机读、随机写和随机定位3种模式下对文件系统的传输速率和CPU占用率进行测试,而在随机读/写中又分为单字节读/写和块读/写两种,反映的是文件系统的输入输出性能指标.测试时,考虑到系统采用Hash方式进行元数据存取,因此选择随机块读取、随机块写入、随机块重写作为测试项.测试结果如图4所示.

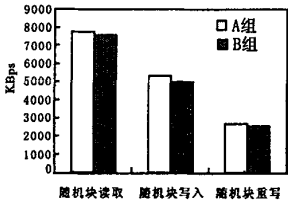


图 4 A 组与 B 组性能对比

Fig. 4 Performance of A group vs B group

图4中,在随机块读取时,当加载了UniformIO_FS和LoadBalance_FS两个模块时速度下降了2%;在随机块写入时,当加载了UniformIO_FS和LoadBalance_FS两个模块时速度下降了3.3%;在随机块重写时,当加载了UniformIO_FS和LoadBalance_FS两个模块时速度下降了3.8%.虽然加载了UniformIO_FS和LoadBalance_FS两个模块使系统

性能受到2%~3.8%影响,但是这与构建基于共享存储池元数据服务系统所提供的统一的虚拟存储空间以及当系统故障恢复时减少大量元数据物理迁移和良好的系统扩展性相比,这种性能影响是可以接受的.

6 结束语

SMDS是在基于共享存储的基础上采用两次Hash方式对元数据进行分布式存储,设计中充分考虑了系统的负载均衡,具有不需要人工干预的故障恢复性,并且最大程度减少了MDS之间大量元数据的物理迁移,有良好方便的系统扩展性.下一步我们将重点对NAS的I/O性能做进一步的优化,并对整个系统各模块做进一步的性能测试和优化.

References:

[1] Mesnier M, Ganger G R, Riedel E. Object-based storage[J]. IEEE Communications Magazine, 2003, 41(8):84-9.

[2] Azagury A, Dreizin V F. Towards an object store[C]. In: Proc. 20th IEEE/11th NASA Goddard Conf. on Mass Storage Systems and Technologies, 2003.

[3] Levy E, Silberschatz A. Distributed file systems: concepts and examples[J]. ACM Computing Surveys, Dec. 1990,22(4):321-374.


[4] Yan Jie, Zhu Yao-long, Xiong Hui. A design of metadata server cluster in large distributed object-based storage[C]. In: 12th NASA Goddard, 21st Conference on Mass Storage Systems and Technologies, 2004,4,13-16.

[5] Scott A. Brandt,Ethan L. Miller,Darrell D. E. Long and Lan Xue. Efficient metadata management in large distributed storage systems[C]. Proceedings of the 20 th IEEE/11 th NASA Goddard Conference on Mass Storage Systems and Technologies , pages 3,2003.

基于共享存储池的元数据服务器机群的设计研究

作者: [苏勇](#), [周敬利](#), [余胜生](#), [姜明华](#), [刘钢](#), [SU Yong](#), [ZHOU Jing-li](#), [YU Sheng-sheng](#), [JIANG Ming-hua](#), [LIU Gang](#)

作者单位: [苏勇, 姜明华, SU Yong, JIANG Ming-hua \(华中科技大学, 计算机科学与技术学院, 湖北, 武汉, 430074; 武汉科技学院, 计算机科学学院, 湖北, 武汉, 430073\)](#), [周敬利, 余胜生, 刘钢, ZHOU Jing-li, YU Sheng-sheng, LIU Gang \(华中科技大学, 计算机科学与技术学院, 湖北, 武汉, 430074\)](#)

刊名: [小型微型计算机系统](#) 

英文刊名: [JOURNAL OF CHINESE COMPUTER SYSTEMS](#)

年, 卷(期): 2007, 28(4)

被引用次数: 1次

参考文献(5条)

1. Mesnier M, Ganger G R, Riedle E [Object-based storage](#) 2003(08)
2. Azagury A, Dreizin V F [Towards an object store](#) 2003
3. Levy E, Silberschatz A [Distributed file systems: concepts and examples](#) 1990(04)
4. Yan Jie, Zhu Yao-long, Xiong Hui [A design of metadata server cluster in larger distributed object-based storage](#) 2004
5. Scott A Brandt, Ethan L Miller, Darrell D, E. Long Lan Xue [Efficient metadata management in large distributed storage Systems](#) 2003

引证文献(1条)

1. 吴婷, 鞠时光, 蔡涛 [基于DBMS的元数据管理策略](#)[期刊论文]-[计算机应用研究](#) 2010(4)

本文链接: http://d.g.wanfangdata.com.cn/Periodical_xxwxjsjxt200704034.aspx

授权使用: 中科院计算所(zkyjsc), 授权号: 449af2bb-d26f-4ee4-9d29-9e4001287e5a

下载时间: 2010年12月2日