

文章编号:1007-130X(2007)003-0117-02

基于 LDAP 的对象存储系统元数据的组织与管理^{*}

LDAP-Based Organization and Management of the Metadata in Object-Based Storage Systems

王 芳,张顺达,冯 丹,曾令仿

WANG Fang, ZHANG Shun-da, FENG Dan, ZENG Ling-fang

(华中科技大学计算机科学与技术学院信息存储系统教育部重点实验室,湖北 武汉 430074)

(National Laboratory of Storage System, Huazhong University of Science and Technology, Wuhan 430074, China)

摘 要:本文介绍了面向对象系统中元数据服务器的设计及元数据的组织和管理。该系统中元数据服务器使用了轻量级目录访问协议(LDAP)作为存放元数据的平台,针对这个平台设计了相应的数据分配算法和数据转换模块,并对其性能进行了分析和优化。

Abstract: This paper introduces the design of a metadata server in object-based storage systems and the organization and management of metadata. The metadata server in our object-based storage system uses the Lightweight Directory Access Protocol (LDAP) to store metadata. And we design data allocation and data conversion modules especially for the platform. We also analyze and optimize the performance of the metadata.

关键词:对象存储系统;元数据管理;轻量级的目录访问协议

Key words: object-based storage system; metadata management; LDAP

中图分类号: TP333

文献标识码: A

1 引言

在以往的 DAS、NAS、SAN 中,由于块设备不能管理元数据,使得服务器/元数据服务器成为瓶颈。而在基于对象存储(OBS)系统中,由于与块/扇区有关的元数据管理(大约有 90% 的负载)已交由 OSD 负责,元数据服务器只管理与文件目录有关的元数据(10% 的负载),使系统瓶颈得到极大的缓解。良好的元数据管理将减少不必要的开销,加快访问速度,均衡负载,提高系统性能。

对象存储模型显现出分布式存储系统的体系结构特征。对象存储系统由客户端、元数据服务器和对象设备三大部分组成,它采用三方通讯方式提供存储服务,即客户端向元数据服务器发送读写文件请求,元数据服务器返回文件对应的对象信息和权限,客户端根据对象信息以赋予的权限访问相应的设备。传统系统中元数据和数据在同一台设备、同一个机器和同一个文件系统中^[1]。基于对效率的考虑,元数据通常被单独存放在距离其描述的数据较近的物理位置上^[2]。在一些分布式文件系统中,数据被存放在

可以通过网络直接访问的智能设备上,而元数据由专门的元数据服务器管理^[3]。我们不难发现,元数据服务器在对象存储系统中的位置非常重要,是整个系统潜在的瓶颈。元数据服务器的中心任务就是元数据的组织和管理。

2 元数据服务器的设计

2.1 元数据服务器的功能分析

对象存储系统中,元数据服务器的主要任务是完成元数据管理功能,如文件目录结构维持、用户管理、命名、权限管理,通过文件系统调用向用户提供与传统文件系统相同的文件服务。主要的关键技术可描述为:

(1)元数据的存储、访问和管理:将元数据存放到轻量级目录访问协议(LDAP)后台数据库 BDB 中,通过 LDAP 协议 API 访问数据库,方便高效地对元数据进行存储、访问和管理。由于 LDAP 针对目录访问进行了优化,在元数据管理方面具有一定优势。加之封装了较多常用功能,使

^{*} 收稿日期:2005-12-26;修订日期:2006-03-22

基金项目:国家 973 计划资助项目(2004CB318201);国家自然科学基金资助项目(60303032)

作者简介:王芳(1972-),女,河北深县人,副教授,研究方向为计算机存储技术与网络存储系统。

通讯地址:430074 湖北省武汉市华中科技大学计算机科学与技术学院;Tel:(027)87557649;E-mail:zhangshunda@126.com

Address: School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, P. R. China

开发和维护更加方便。其可扩展性和安全性也十分优异。

(2)缓冲区管理:针对元数据访问的特点,在系统中特制一个 Buffer Cache,将最近访问过的元数据缓存到缓冲区中,以便下次访问命中时直接从缓冲区中取走数据,不必再产生磁盘 I/O。缓冲区使用类似 Hash 链表加双向链表的结构,使用 LRU 算法淘汰过时数据,提高内存利用率。

(3)文件到对象的映射:映射通过策略库调出一定的算法,根据 OSD 的负载情况、空闲空间的大小、文件本身的特点、QOS 的要求等因素决定对象 ID 的分配。

(4)对象在 OSD 间的分布:根据文件的大小决定如何放置到具体的 OSD 设备中。大文件(超过一定阈值,如 1M、2M 等)就采用分片的方式,即将文件平均分成若干份,散列在 OSD 中;小文件使用 Hash 的方法,将文件全路径名哈希成一个整形数,根据这个数字确定放置对象的设备。分片和哈希各有自身的优势,将它们分别应用到大文件和小文件中可以最大限度地发挥算法的特点,使系统性能达到最优。

2.2 元数据服务器的软件组成

元数据服务器由文件管理器、资源管理器和轻量级目录访问协议(LDAP)数据库三大部分组成。文件管理器主要负责与客户端的通信、对象分配、元数据管理;资源管理模块主要负责与设备通信、管理设备和用户、设备间负责均衡和调度;LDAP 数据库存放元数据信息和设备信息,也是文件管理模块和设备管理模块的公用平台。

与元数据组织和管理关系较为密切的是文件管理器,其具体的结构如图 1 所示。

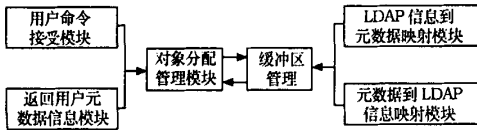


图 1 文件管理器各模块

文件管理器各主要模块包括:

(1)客户命令接受模块。负责从网络接受客户发送的读写请求。

(2)对象分配、管理模块。负责从客户命令接受模块中提取信息,根据相关信息结合 LDAP 中的信息分配对象和管理元数据。在现阶段具体就是通过文件类型和大小的判断,决定文件到对象的映射。大文件使用分片算法,小文件使用哈希算法。

(3)返回客户数据信息模块。将预分配好的对象信息封装成标准的 OSD 命令,通过网络传送给客户。

(4)元数据到 LDAP 信息映射模块。将新建的对象信息转换成 LDAP 标准格式,并提交到 LDAP 后台数据库中。比如,文件 /test/usr/src/prog.c 对应的对象会被存放到类似 dn: cn = prog.c, cn = src, cn = usr, cn = test, cn = root, dc = example, dc = com 的记录中去。而具体的对象数据结构会被存放到此条记录下的属性项中去,如 cn、sn 和自定义的属性等。再通过如 ldap_open()、ldap_bind()、ldap_add()等 API 函数向后台数据库提交结果。

(5)LDAP 信息到元数据映射模块。是元数据到 LDAP 信息映射模块的逆过程,供对象分配、管理模块调用

LDAP 信息时使用。

(6)缓冲区管理模块。当用户发送的信息到达服务器端时,首先将所得到的文件名哈希成一个整形数,以这个数为下标找到对应的数组元素。再以这个数组元素为指针找到元数据信息。如果其中的文件名与用户发送的文件名吻合,说明就是用户要找的元数据,则可直接取走;否则顺着链表向后找。若找到则取走元数据,若找不到则访问 LDAP 数据库,并将返回的元数据填充至链表头。此时,如果链表节点数达到设定的最大值,则将最后链表末尾项丢弃。这样就简单地实现了 LRU 算法。

2.3 元数据的存放和转换

元数据存放在服务器后台挂接的 LDAP 服务器中。LDAP (Lightweight Directory Access Protocol, 简称 LDAP)是基于 X. 500 标准的,但它简单多了,并且可以根据需要定制。与 X. 500 不同,LDAP 支持 TCP/IP,这对访问因特网是必须的。就像 Sybase、Oracle、Informix 或 Microsoft 的数据库管理系统(DBMS)是用于处理查询和更新关系型数据库那样,LDAP 服务器也是用来处理查询和更新 LDAP 目录的。换句话说,LDAP 目录也是一种类型的数据库,但不是关系型数据库。LDAP 最大的优势是:可以在任何计算机平台上用很容易获得且数目不断增加的 LDAP 的客户端程序访问 LDAP 目录,而且也很容易定制应用程序为其加上 LDAP 的支持。LDAP 协议是跨平台的和标准的协议,因此应用程序就不必为 LDAP 目录放在什么样的服务器上操心。

Linux ext2 文件系统的最大容量为 4TB^[4],而 LDAP 通过多台服务级联可以达到 PB 级容量。对象存储系统中的元数据不包括块信息,大部分元数据读多写少,长度固定且较小,适合数据库特别是 LDAP 数据库模型。另外,LDAP 允许你根据需要使用 ACL(一般都称为 ACL 或访问控制列表)控制对数据读和写的权限,在安全方面比单纯的文件系统更有优势。综合 LDAP 的跨平台、容量、性能和安全考虑,我们的元数据存储平台使用 LDAP 而不是传统的文件系统。

对象分配、管理模块与元数据到 LDAP 信息映射模块和 LDAP 信息到元数据映射模块通过 get(struct object *) 和 put(struct object) 函数交互信息,保证了文件请求预处理/元数据管理模块和 LDAP/DBMS 自定义映射机制模块间的相对独立,前者不涉及任何 LDAP 信息的处理(strcut object 结构包括对象 id、顺序号、对应设备的 IP 地址、对象长度、对应的文件名、对应客户 id、对应客户组 id、创建时间、修改时间、更新时间等信息)。

3 性能分析及优化

尽管理论上轻量级目录访问协议(LDAP)具有更高的性能,但在实际应用当中由于受到实际系统的影响,在元数据读写量不大的情况下其性能反而不如 ext2 文件系统。这也是由于 ext2 使用了 Buffer Cache,在 TB 级环境中对文件的读写有优化。针对当前的应用环境,我们在元数据服务器中加入了自建的缓冲区,使性能得到了提升。测试条

(下转第 135 页)

生联系;文献[8]试图通过扩展一个面向对象的系统来对角色继承进行研究。然而,他们提出的角色继承的概念和继承机制都是基于对象技术,其本质只是用系统中扮演着各种角色的 Agent 来代替了原来的对象。正如我们在 2.3 节中分析的,多 Agent 系统中继承具有了新的特点,表达方式、分析过程都不同于对象技术中的继承。因此,在多 Agent 系统的分析和设计中,需要从 Agent 的特点和角色的概念出发研究继承的概念和继承机制,通过对继承的识别、描述和分析,在需求分析阶段得到清晰自然的系统结构层次图,促进设计阶段的软件重用。

但是,现有的研究还不足以使继承机制像对象技术中那样有效地支持软件系统的开发,还有一系列的问题需要解决。比如,本文仅对单继承进行了分析,实际上多继承也普遍存在于多 Agent 系统中;另外,本文讨论了怎样在需求分析阶段对继承进行分析和建模。下一步应该讨论在设计阶段对继承机制提供支持,甚至在面向 Agent 的程序设计语言中实现继承。

参考文献:

- [1] Mao Xinjun, Yu E. Organizational and Social Concepts in Agent Oriented Software Engineering[A]. Proc of Agent Oriented Software Engineering[C]. 2004. 1-15.
- [2] Ferber J, Gutknecht O, Michel F. From Agents to Organizational View of Multi-Agent Systems[A]. Proc of the 4th Int'l Workshop on Agent Oriented Software Engineering IV[C]. 2003. 214-230.
- [3] Zambonelli F, Jennings N R, Wooldridge M. Developing Multiagent Systems: The Gaia Methodology[J]. ACM Trans on Software Engineering Methodology, 2003, 12(3): 317-370.
- [4] Deloach S A, Wood M F, Sparkman C H. Multiagents Systems Engineering[J]. International Journal of Software Engineering and Knowledge Engineering, 2001, 11(3): 231-258.
- [5] Bresciani P, Giorgini P, Giunchiglia F, et al. TROPOS: An Agent-Oriented Software Development Methodology[J]. Journal of Autonomous Agents and Multi-Agent Systems, 2004, 8(3): 203-236.
- [6] Ferber J, Gutknecht O. A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems[A]. Proc of 3rd Int'l Conf on Multi-Agent Systems[C]. 1998. 128-135.
- [7] Depke R, Heckel R, Kuster J M. Improving the Agent-Oriented Modeling Process by Role[A]. Proc of the 5th Int'l conf on Autonomous Agents[C]. 2001. 640-647.
- [8] Gottlob G, Schrefl M, Rock B. Extending Object-Oriented Systems with Roles[J]. ACM Trans on Information Systems, 1996, 14(3): 268-296.

(上接第 118 页)

件为 8K 个目录(记录),分为 2K 组,每组深度为 4 层,1K 个线程并发访问。

测试结果如图 2~图 5 所示。可以看出,未加缓冲区时,对于 ext2:第一次 10s 左右,第二次 5s 左右,后面的数据稳定在 0.2s 左右。对于 LDAP:第一次 10s 左右,第二次 5s 左右,后面的数据稳定在 2s 左右。加上缓冲区后 LDAP 性能优于 ext2。对于 ext2 来说,自制的缓冲区效果

并不明显,这是因为 ext2 的 Buffer Cache 已经十分完善。而对于 LDAP 来说性能提高很明显。

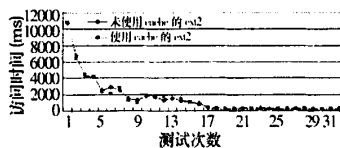


图 2 ext2 使用与未使用 Cache 的比较

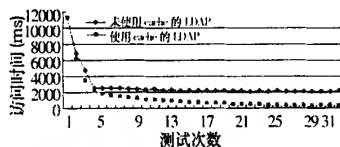


图 3 LDAP 使用与未使用 Cache 的比较

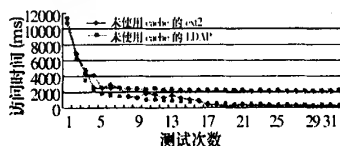


图 4 ext2 与 LDAP 未使用 Cache 的比较

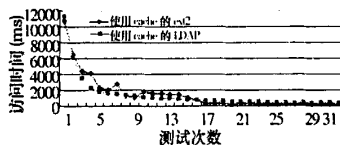


图 5 ext2 与 LDAP 使用 Cache 的比较

4 结束语

本文介绍了面向对象系统中元数据服务器的设计及元数据的组织和管理。该系统中元数据服务器使用了轻量级目录访问协议(LDAP)作为存放元数据的平台,针对这个平台设计了相应的数据分配算法和数据转换模块,并使用缓冲技术优化了性能。在容量、性能和安全等方面都得到了较好的效果。

在今后工作中,我们将在实测数据的基础上进一步改进系统,提高性能。

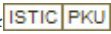
参考文献:

- [1] Morris J H, Satyanarayanan M, Conner M H, et al. Andrew: A Distributed Personal Computing Environment[J]. Communications of the ACM, 1986, 29(3): 184-201.
- [2] McKusick M K, Joy W N, Leffler S J, et al. A Fast File System for UNIX[J]. ACM Trans on Computer Systems, 1984, 2(3): 181-197.
- [3] Gibson G A, Meter R V. Network Attached Storage Architecture[J]. Communications of the ACM, 2000, 43(11): 37-45.
- [4] Wang Feng, Brandt S A, Miller E L, et al. OBFS: A File System for Object-Based Storage Devices[A]. Proc of the 21st IEEE / 12th NASA Goddard Conf on Mass Storage Systems and Technologies[C]. 2004.

基于LDAP的对象存储系统元数据的组织与管理

作者: [王芳](#), [张顺达](#), [冯丹](#), [曾令仿](#), [WANG Fang](#), [ZHANG Shun-da](#), [FENG Dan](#), [ZENG Ling-fang](#)

作者单位: [华中科技大学计算机科学与技术学院信息存储系统教育部重点实验室](#), 湖北, 武汉, 430074

刊名: [计算机工程与科学](#) 

英文刊名: [COMPUTER ENGINEERING AND SCIENCE](#)

年, 卷(期): 2007, 29(3)

被引用次数: 1次

参考文献(4条)

1. Morris J H. [Satyanarayanan M. Conner M H Andrew: A Distributed Personal Computing Environment](#) 1986(03)
2. McKusick M K. [Joy W N. Leffler S J A Fast File System for UNIX](#) 1984(03)
3. Gibson G A. [Meter R V Network Attached Storage Architecture](#) 2000(11)
4. Wang Feng. [Brandt S A. Miller E L OBFS: A File System for Object-Based Storage Devices](#) 2004

相似文献(9条)

1. 学位论文 [张顺达](#) 对象存储系统的元数据管理 2006

随着网络技术和信息数字化的快速发展,面向海量数据的大型应用纷纷涌现,进一步对存储系统性能提出更为苛刻的要求。尽管磁存储技术仍在不断发展中,但受到块级存储访问接口制约,无法改变I/O性能远落后于CPU和内存速度的状况。对象存储系统(Object-Based Storage System)以对象为接口,将有望解决这些问题。容纳海量用户数据的对象存储系统中高效的元数据管理成为了新的挑战和研究课题。

对象存储系统由客户端、元数据服务器和各个对象存储节点三部分组成。用户数据存放在直接联网访问的智能存储节点上。元数据服务器在对象存储系统中的位置非常重要,是整个系统潜在的瓶颈。在这种具有分布式系统结构特征的对象存储系统中,文件被映射到一个或多个对象存储节点上。合理的对象分布策略对系统性能显得尤为重要。针对常用对象分布策略哈希(Hashing)算法和分片(Fragment-Mapping)算法存在的优缺点,提出一种能够结合两者优点、又尽量避免其缺点的柔性对象分布算法,同时分析了影响对象存储系统性能的主要因素。

元数据服务器的设计及元数据的组织和存储是面向对象系统中元数据管理的重要组成部分。元数据服务器使用了轻量级目录访问协议(Lightweight Directory Access Protocol, LDAP)作为存放元数据的平台,针对这个平台设计了相应的数据分配算法和数据转换模块,针对元数据访问特征,构建缓冲机制优化元数据访问性能。通过测试验证了柔性对象分布算法和元数据组织管理模式在对象系统中是行之有效的,并对系统性能的提升起到了重要作用。

2. 学位论文 [姜成龙](#) 对象文件系统中元数据管理技术研究 2005

随着信息技术的进一步发展,以及网络的大规模应用,带来了数据的爆炸性增长,也给网络存储带来了巨大的发展机会。如何构建一个扩展性强、可靠性高、易管理的高性能存储系统成为目前研究的一个重要课题。

基于对象的存储技术是存储领域的新兴技术,它提出了一种新型的存储结构,数据对象是这种存储结构的核心,数据对象封装了用户数据(文件数据)和这些数据的属性(元数据),他们分别由不同的系统管理。以对象存储结构为基础构建的大型分布式文件系统,可扩展性强、可靠性高,能提供较强的并发数据处理能力。元数据服务管理在对象存储文件系统中尤为重要,采用集群管理元数据是大型对象存储系统中的一种趋势,本文致力于研究对象存储结构中的元数据集群管理技术,所做的主要工作如下: 1. 分析研究基于对象存储系统的体系结构,设计并实现了一个小型的对象存储文件系统原型OCFS。

2. 研究对象存储文件系统中的元数据管理,设计原型改进的文件系统OCFS II,对元数据管理集群实行层次化管理,分别以目录路径索引服务器DPIS集群和元数据服务器MDS集群管理目录元数据和文件元数据。

3. 在研究集群负载均衡的基础上,设计和实现OCFS II元数据管理集群静态负载均衡与动态反馈重分配相结合的负载均衡方案。通过静态元数据分割算法和元数据分布存储,实现元数据服务负载均衡;采用动态反馈服务器负载信息,实现不均衡负载重新分配。保证元数据管理集群的负载均衡,并解决了“热点”数据访问问题。

4. 设计实现了OCFS II元数据管理集群可用性保障方案。目录路径索引服务器DPIS集群中采用共享容错磁盘阵列和节点容错机制解决共享存储数据和节点故障问题;元数据服务器MDS集群采用备份服务器保证服务器节点出现故障时元数据服务工作的接替和数据备份的重建。实现了元数据管理集群在单点失效和特定的多点失效情况下的容错和恢复,保证了系统的可靠性和可用性。

3. 学位论文 [刘仲](#) 基于对象存储结构的可伸缩集群存储系统研究 2005

随着处理器和网络技术的飞速发展,大大的提高了Linux集群计算的计算能力。Linux集群计算在高性能科学计算、商业应用和海量信息服务等领域得到了广泛应用,逐渐发展成为高性能计算中的流行方法。而受传统存储结构的限制,其计算能力得不到充分体现。构建满足Linux集群计算需求的可伸缩、高性能、跨平台、安全、共享数据的存储结构对现有的存储结构提出了巨大的挑战。

新兴的对象存储结构能够利用现有的处理技术、网络技术和存储组件提供空前的可伸缩性和聚合吞吐量,为构建新一代的大规模并行存储系统提供了基础。本文在全面深入了解对象存储结构与现有对象存储系统的基础上,对基于对象存储结构的大规模集群存储系统涉及的几个关键技术进行了深入研究,提出了新颖有效的实用算法。主要的贡献如下:

(1) 提出一种基于确定性算法分布目录对象和数据对象的可伸缩集群文件系统的框架结构,改进了现有对象存储系统的元数据和数据对象的管理方法。基于确定性算法自主计算数据分布的方法简化了大规模存储系统的管理,支持元数据服务器、存储节点的动态均衡扩展。

(2) 首次提出目录路径属性与目录对象分离的元数据管理方法,扩展了现有的对象存储结构。该方法能够有效避免因为目录属性的修改而导致的大量元数据更新与迁移;通过减少前缀目录的重迭缓存提高了元数据服务器Cache的利用率和命中率;通过减少遍历目录路径的开销和充分开发目录访问的存储局部性,减少了磁盘I/O次数;通过元数据服务器的动态负载均衡避免单个服务器过载。实验结果表明该方法在提高系统性能、均衡元数据分布以及减少元数据迁移等方面具有明显的优势。

(3) 首次在研究数据对象的分布中引入MonteCarlo方法,提出一种基于动态区间映射的数据对象布局算法,支持权重分布和副本,在均衡数据分布和最少迁移数据方面都是统计意义上最优的,有效解决了动态存储系统的数据均衡分布问题,提高了系统的可扩展性。该算法的基本思想是将数据对象与随机数对应起来,将存储节点与容纳随机数的区间对应起来,将离散空间中的数据对象分布问题转化为连续空间中的区间分割问题。根据系统中存储节点的规模和权重将单位区间分割成不同长度的区间,并在区间与存储节点之间建立映射关系,通过两次映射确定数据对象的存储位置。理论分析和实验结果表明数据对象分布具有统计意义上的均衡性、自适应性和迁移最优性,定位数据对象速度快。

(4) 提出一种可伸缩分布式节点地址计算算法。该算法使得数据对象分配地址的计算不依赖于中央节点计算或者访问一个集中式目录, 所有计算节点和存储节点独立地进行地址计算, 并且对数据对象的访问操作或新增节点引起的系统规模变化不需要原子更新到其他计算节点, 计算节点通过视图校正算法自主学习, 自动适应新的系统规模。消除了现有的集中式访问性能瓶颈, 使系统具有高可伸缩性。

(5) 分别提出基于镜像和分组的高可用数据对象布局算法, 借鉴RAID的方法在算法一级上实现数据的冗余分布。在数据对象和存储节点失效时, 利用冗余数据重构数据对象和存储节点, 保证存储系统的高可用性。采用马尔可夫激励模型分别对基于镜像和分组的高可用数据对象布局算法的存储系统进行定量的可用性分析, 计算结果表明两种方法能够有效保证存储系统的高可用性。

(6) 对基于动态区间映射的数据对象布局算法进行扩展, 提出支持节点组的数据对象副本布局算法, 支持多个节点同时批量扩展, 扩展后的数据对象布局算法仍然保证数据对象的分布是负载均衡的, 支持权重分布和副本。理论分析和实验结果表明数据对象分布具有统计意义上的均衡性、自适应性和迁移最优性, 定位数据对象速度快。

(7) 利用上述研究成果, 在Linux操作系统上设计与实现了一个基于对象存储的集群文件系统原型。

4. 学位论文 [刘群 基于可扩展对象的海量存储系统研究](#) 2006

信息存储是人类社会永恒的需求。随着计算机技术的发展和应用的普及, 信息存储容量成爆炸性地增长, 现有网络存储系统已无法满足人们对于存储的需要。基于对象存储 (Object-Based Storage, OBS) 技术适时崛起, 利用现有的存储组件、处理技术和网络技术, 通过简单方式来获得前所未有的高吞吐量, 成为下一代网络存储的主流。它采用包含数据和属性的“对象”作为接口, 既有“块”接口的快速, 又有“文件”接口的便于共享, 并分离了存储数据的逻辑视图和物理视图, 将存储数据的逻辑视图保留在元数据服务器中, 而物理数据存放在基于对象存储设备 (Object-Based Storage Device, OSD) 中。同时, 它将传统文件分解为系列数据对象, 分发到一个或多个OSD 中。虽然对象给存储系统带来了一种新的理念, 但现有的与对象相关的存储系统中对象都仅定义为非定长的数据单位, 束缚了“对象”这个有着丰富内涵的词汇。

基于可扩展对象的海量存储系统 (Based on Scalable Object Mass StorageSystem, BS0-MSS) 吸取了OBS 的优点, 在“对象”现有的含义基础上扩充, 使它不仅仅包括用户数据, 还将目录、文件、存储设备管理等纳入对象之中, 形成层次结构的对象体系结构, 实现对象的分布存储、层次管理的模式, 并建立基于存储对象统一访问模式, 将块、对象和文件三种存储接口进行融合与统一。这样不仅具有统一逻辑视图、数据共享、主动服务、并行访问、统一存储和易管理等特点, 而且有着其他存储结构难以达到的高可扩展性和高性能。

通过建立系统广义随机Petri 网模型, 对BS0-MSS 进行性能评价, 模拟结果显示无论增加存储对象 (Storage Object, SO) 还是客户端, 系统性能都随之增加。并采用测试工具iozone 对系统原型与Lustre 系统作对比测试, 测试结果表明写性能超过Lustre, 读性能略比Lustre 好, 并验证了BS0-MSS 的广义随机Petri 网模型。

首次将存储系统与元胞自动机相结合, 利用元胞自动机的原理, 解析BS0-MSS动力演变规律。构建了一个通用框架的BS0-MSSCA 概念模型框架, 并在此基础上, 分析了两种具体元胞自动机模型。基于存储对象负载分配模型是将SO 解析为元胞, 模拟了一个简单的负载均衡分配的动态变化, 高度概括了BS0-MSS 的演变过程。

基于数据对象访问行为模型则分析数据对象的访问频率对系统的影响, 结合数据对象访问的特征和主动性, 通过机械学习适当调整数据对象的访问行为频率, 使系统朝着稳定方向发展。通过分析基于存储对象的负载分配模型和基于数据对象的访问行为模型的演变过程, 可以看出系统具有主动性、共享性、并行性、相关性等特性, 是一个自组织管理的对象存储系统。

大规模分布式存储系统中, 元数据高性能服务、负载均衡以及扩展性已成为一个重要的研究热点。在元数据服务器中, 将元数据分解为目录对象和文件对象, 目录对象为定位性元数据, 提供文件所在位置和访问控制; 文件对象为描述性元数据, 描述文件的数据特性。每一个元数据服务器 (Metadata Server, MDS) 负责所有目录对象和自身的文件对象, 这样充分利用MDS 中Cache, 提高Cache 的命中率, 减少磁盘I/O 次数, 并且能够动态扩展MDS。同时, 以目录对象ID 和文件名关键字的哈希值作为局部元数据查找表 (Local Metadata Lookup Table, LMLT) 的索引, 获得相应的MDS_ID。一旦目录权限改变、更名、移动目录、修改权限等都不会造成元数据的迁移。通过Bloom Filter 算法将每个MDS 的LMLT 压缩成一个摘要, 能够实现快速的元数据查找。同时采用主从备三重链式结构的MDS 服务, 不仅在未提高硬件成本下能够保证系统高可靠性和可用性, 而且根据热点访问进行迁移, 实现负载均衡。SO 是BS0-MSS 重要组成单位, 它与OSD 不同之处是本身具有“接口”与“状态”标识, 由数据、属性和方法组成, 这样对现有的T10 OSD 标准进行了扩充。由于数据对象是通常在二维空间中命名, 传统文件系统管理大量数据对象的效率是极其低, 采用线性哈希查找算法, 由负载因子控制分裂和合并, 与传统文件系统的树结构查找相比, 哈希法查找时间复杂度为O(1)。同时, 针对Ext2 文件系统中数据访问至少需两次以上的磁盘操作特性, 将数据的块地址和长度链接在一起, 作为对象的扩展属性, 连同数据对象一起存储到磁盘中, 这样无论数据对象大小为多少, 磁盘访问次数仅为两次。在BS0-MSS 中, 负载与众多因素相关, 如请求队列长度、CPU 处理能力、内存大小、网络带宽、磁盘带宽和磁盘容量等。负载柔性放置策略不仅考虑网络的影响, 而且考虑SO 之间存在差异, 并设置权重, 权重大的SO 担负较多的负载。依据SO 属性中信息统计出负载特征, 以系统响应时间为代价, 自适应选择SO 数目, 采用不同大小的分条进行存储, 使BS0-MSS 具有更高的性能、可扩展性和自适应负载均衡能力。

5. 期刊论文 [单颖、姚念民、赵建明、Shan Ying、Yao Nianmin、Zhao Jianming 基于对象存储系统体系结构的研究](#) - 计算机研究与发展2009, 46 (z2)

随着基于对象存储系统的快速发展, 存储系统服务性能要求越来越高。在分析传统基于对象存储系统体系结构的基础上, 提出一种体系结构SOBSS, 通过扩展元数据服务器功能, 简化体系结构内部数据交互模式。通过与传统基于对象存储系统体系结构的性能比较, 实验结果表明, 采用SOBSS体系结构在提高系统性能方面有明显的优势。

6. 学位论文 [孙丽丽 共享对象存储并行文件系统的元数据管理研究](#) 2005

当前的高性能计算已经由传统的主机方式逐渐向机群方式演变。机群体系结构的采用一方面使得系统的计算能力大大加强, 另一方面也对当前的存储系统提出了更高的要求; 在保证数据共享和易管理性的前提下, 要求存储系统在存储容量和I / O性能方面具有很好的可扩展性。传统的基于主机的存储架构已经远远不能满足这些要求, 研究新的存储体系结构和相应的文件系统具有十分积极的意义。本文基于对象存储系统, 提出一种新的共享对象存储设备的并行文件系统 (命名为SOPFS) 设计, 其目标是为高性能计算机群提供高性能、可扩展、高可用的机群存储系统。在文中给出了SOPFS的总体描述, 内容包括分布式元数据管理、并行数据访问等关键技术; 结合SOPFS中动态散列分区的元数据组织方法, 设计实现了元数据的访问管理; 针对高性能并行计算中经常出现的对同一文件 / 目录的高并发访问情形, 提出了一种动态的元数据复制策略, 通过多个存放元数据副本的MDS同时响应应对同一文件元数据的并发访问请求, 提高了高并发访问情形下的元数据访问效率; 利用SOPFS的结构优势, 即文件系统的数据通路与控制通路分离, 提出了懒惰的元数据更新策略, 使得文件元数据的更新独立于文件数据读写过程, 进一步保证了高的I / O吞吐率; 在SOPFS中设计实现了元数据的事务日志机制, 以提高系统的可用性和提供系统的快速失败恢复能力。

7. 学位论文 [吕松 对象存储结点的设计与实现](#) 2006

随着知识经济的推进和信息时代的日益临近, 同时在网络技术革新的推动下, 存储行业既迎来大好的市场前景又面临巨大技术挑战。数据量的指数级增长和基于高速网络的数据应用要求的进一步提高, 对数据存储技术提出了革新的要求, 不仅针对网络存储的体系结构, 而且对存储设备的接口也提出了新的要求。

基于对象存储提出了对象这样新的数据存储单位, 增加了数据的自我管理能力和, 同时方便了数据的共享和访问, 将对象存储系统中各个不同功能部件的职能进行了重新的分工, 将对象数据的物理存储管理任务移交给存储设备, 提高了存储结点的智能性。用户和存储设备结点之间直接进行数据传输, 从而提高了数据访问速度以及系统的可扩展性。

基于对象的存储设备结点利用通用处理器的计算能力, 实现专用OSD Controller (Object-Based Storage Device Controller) 的处理功能。对象处理模块通过可装载方式插入到Linux操作系统内核, 在内核态下实现, 减少I/O过程中内核切换的开销。基于对象存储结点通过Iscsi通道和MDS、Client进行数据交互, 顺应了现在网络存储向低成本IP存储发展的趋势。对象存储结点实现了最新T10标准中的常用OSD命令集, 并对该标准进行相应的扩展。对象存储结点在Ext2文件系统基础上构建了对象存储管理, 实现了对象存储接口的物理基础和外部访问接口。测试结果表明, 对象接口很好地提高了设备的并行性和传输效率, 同时单台对象存储结点虽然增加了元数据管理负担, 但是性能仍然和文件接口的FTP相当。

8. 期刊论文 [谈华芳、孙丽丽、侯紫峰、TAN Huafang、SUN Lili、HOU Zifeng 一种基于对象存储中的元数据组织管理方](#)

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.

9. 学位论文 [王涌 面向PB级存储系统的元数据集群管理容错方法研究与实现](#) 2007

随着计算机技术, 信息技术和互联网络的发展, 高性能计算、商业计算、大规模数据处理、信息处理等技术得到广泛的应用. 集群系统因其较高的性能价格比和较好的可扩展性而受到越来越多的青睐. 与此同时, 这些应用对分布式数据存储提出了更大容量, 更高性能, 更高可用性的要求.

新兴的对象存储结构能够利用现有的处理技术、网络技术和存储组件提供空前的可伸缩性和聚合吞吐量, 为构建新一代的大规模并行存储系统提供了基础.

本文在全面深入了解对象存储体系结构与现有对象存储系统的基础上, 对基于对象存储体系结构的大规模集群存储系统所涉及的元数据集群管理的容错问题进行了深入的研究, 提出了新颖的思想和解决方法. 主要的贡献如下:

(1) 提出一种面向PB级存储系统的基于目录对象副本的高可用元数据管理模型. 通过采用高效的、并发的目录对象副本放置、更新、迁移策略以及管理机制, 既保证实现了目录对象数据的可靠性和可用性, 又增加了读取数据时的聚合网络带宽, 提高读操作的性能. 同时采用马尔可夫激励模型进行了定量可用性分析. 实际的功能测试与性能测试表明该模型方法能够有效保证存储系统的高可用性, 提高元数据服务器集群的整体访问性能.

(2) 提出了一种对面向基于日志的元数据管理方法进行检查点操作的思想方法. 通过对基于日志的元数据管理方法实施检查点操作, 既保证了系统存储空间的有效利用, 同时实现了系统的快速恢复, 保证了系统的高可用性. 实际的测试表明该方法能够充分地利用磁盘空间, 提高系统的恢复时间, 保证元数据的高可用性, 提高元数据的访问效率.

引证文献(1条)

1. [赵曦, 陈建阳 一种基于目录和关系的混合数据模型](#)[期刊论文]-[计算机科学](#) 2008 (2)

本文链接: http://d.g.wanfangdata.com.cn/Periodical_jsjgcykx200703037.aspx

授权使用: 中科院计算所(zkyjsc), 授权号: cbe4e651-eb6a-479f-90b5-9e400128d36a

下载时间: 2010年12月2日