计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
46(Suppl.): 78-82, 2009

基于层次结构的元数据动态管理方法的研究

刘 群[1] 冯 丹[2]
[1](华中科技大学网络与计算中心 武汉 430074)
[2](华中科技大学计算机科学与技术学院 武汉 430074)
(liliuqun@mail.hust.edu.cn)

Research on the Method of Metadata Dynamic Management Based on Hierarchical Structure

Liu Qun[1] and Feng Dan[2]
[1](*Center of Network and Computing, Huazhong University of Science and Technology, Wuhan* 430074)
[2](*School of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract In large distributed storage systems, it is critical to research on high performance metadata service and scalability. In the metadata server (MDS), metadata is divided into directory object and file object. The former is locating metadata, providing the file location and access control; the latter is descriptive metadata, depicting the data of file. Each MDS is responsible for the whole directory objects and its own file objects. Meanwhile, the hash value of key including directory object ID and file name is used as an index to the local metadata lookup table (LMLT). Through Bloom Filter algorithm, the LMLT of each MDS can be compressed into a brief. Not only does it make full use of the cache, improving the hit rate of cache, reducing the number of disk I/O, and extending MDS dynamically, but also achieves rapid metadata finding.

Key words metadata management; hierarchical structure; local metadata lookup table

摘 要 在大规模分布式存储系统中,元数据高性能服务和扩展性已成为一个重要的研究热点.在元数据服务器(metadata server,MDS)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.每个 MDS 负责所有目录对象和自身的文件对象,同时,以目录对象 ID 和文件名为关键字的 Hash 值作为局部元数据查找表的索引,通过 Bloom Filter 算法将每个 MDS 的局部元数据查找表压缩成一个摘要,这样既可利用 MDS 中 Cache,提高 Cache 的命中率,减少磁盘 I/O 次数,动态扩展 MDS,又能够实现快速的元数据查找.

关键词 元数据管理;层次结构;局部元数据查找表

中图法分类号 TP333

元数据是描述数据的数据.在传统存储系统中,如直接存储系统(direct access storage, DAS)元数据和数据通常是在同一机器中由同一文件系统管理,并存储于同一设备中.为了提高效率,元数据储存在其描述的数据附近[1].在大规模分布式存储系统中,元数据管理与数据读写分离的趋势日益明显.特别在基于对象存储(object-based storage, OBS)中,将元数据分离为存储数据的逻辑视图(约占负载

收稿日期:2008-09-16
基金项目:国家"九七三"重点基础研究发展计划基金项目(2004CB318201);国家自然科学基金项目(60503059);华中科技大学实验技术研究基金项目

的20%)和物理视图(约占负载的80%),将存储数据的逻辑视图保留在元数据服务器(metadata server, MDS)中,而存储数据的物理视图存放在对象存储设备(object-based storage device, OSD)中[2-3]. 由一个或多个MDS组成的MDS集群管理着整个系统的命名空间、目录层次和文件及目录的许可,并且实现文件到对象的映射. 虽然相对系统的整个存储容量来说元数据很小,但是文件系统中大约50%~80%操作是对元数据进行的[4],因此实现元数据服务的高性能、高可靠、负载均衡以及扩展性至关重要,元数据存储与管理已经成为一个重要的研究热点.

元数据管理分为静态与动态两种方法,静态方法有静态子树分割和静态Hash,静态子树分割是通过人工方法将不同的目录子树分布到各个MDS中,但不同目录下的文件数量和被访问的热度都会不同,导致负载不均衡,同时迁移时需将整个目录子树移走,数据量大. 静态Hash是通过文件的全路径名进行Hash计算直接分配到MDS,之前须遍历前缀目录来确定访问权限,当更名时导致所有与之相关的大量元数据迁移.

动态方法包括动态子树分割、延迟混合、动态Hash以及基于目录路径元数据管理. 动态子树分割与静态子树分配不同的是它能够根据负载情况实现动态的负载均衡,但有大量前缀目录信息在不同MDS缓存中重迭,降低Cache的利用率和命中率[5]. 延迟混合[6]通过文件的全路径名的Hash值作为元数据查找表(metadata look-up table,MTL)的索引,分配元数据到MDS中,避免热点请求,需维护目录层次结构以便提供标准的目录语义和操作. 动态Hash[7]采用与延迟混合类似的方法,借用Lazy方法对元数据进行更新,但同一目录下不同文件可以分布到不同的MDS中,从而导致前缀目录在不同的MDS中大量重叠,降低了MDS的性能. 目录路径元数据管理方法[8]是将目录路径单独存放在目录路径索引服务器中,采用字典表与Hash相结合,建立目录路径索引服务器与MDS的两张映射表,易消耗大量内存,众多客户端并发访问时容易造成的I/O瓶颈.

在大规模PB级对象存储系统(object-based storage system, OBSS)中,元数据可达GB级,甚至到TB级,在这么大的范围中广泛搜索,支持庞大的目录结构,实现不同目录下的不同文件、相同目录下的不同文件甚至是同一文件的成百上千地并行访问,是当前急需解决的问题. 因此,本文提出一种基于层次对象的元数据动态管理,它克服了由于更名所引起的大量元数据的迁移,提高了功能I/O性能,具有良好的扩展性和快速的查找与定位功能.

1 MDS的层次结构

元数据从上到下可分为定位性元数据、描述性元数据和存储性元数据,其中前两种在OBSS中由MDS负责,存储性元数据则由OSD自行管理. 因此,MDS中的元数据可分为目录对象和文件对象,其中目录对象为定位性元数据,提供文件所在位置和访问控制,存储于目录对象管理器中;文件对象为描述性元数据,描述文件的数据,属性包括文件修改时间、存取时间、所有权和文件大小等,由文件管理器负责. 每一个MDS中都含着整个系统的目录对象管理和本服务器的文件对象管理.

1)目录对象

目录对象是指目录路径名,由目录对象ID标识,属性为目录对象的创建时间、记录个数、访问权限等,将目录的层次结构、权限控制与目录文件相分离,每一个目录对象所对应的目录文件对象则保存在文件对象管理器,目录对象由目录对象管理器独立管理,包括内容如下.

DiOID:全局唯一64b的目录对象ID;

DirectoryName:该目录对象的目录路径名;

AC:该目录对象的访问控制.

其中目录对象由*DiOID*标识,当为根目录对象"/"时,*DiOID*为全零. 由于系统能达到2^{64}个目录,即使假设每个目录只有1KB的文件,$1KB * 2^{64} = 2^{24}PB$,这足以满足PB级存储系统的需求. 当创建一个目录时,将该目录路径的访问控制记录在此数据结构中,若此目录创建新的子目录,则不需要遍历所有的目录路径,直接获取当前目录对象的访问控制,构建新子目录对象的访问控制.

这种设计分离了目录路径信息和文件元数据,采用*DiOID*和文件名合并的值进行Hash动态分配到MDS中,因此目录对象信息可以任意修改,如更改目录名、移动目录、修改访问权限等,避免大量元数据的迁移.

假定文件hust.doc所在的目录层数为n,若采用传统系统的元数据管理,则获取目录元数据需要$2n$次,这是因为须遍历文件的所有目录路径获得该文件的元数据和访问权限;而采用HMDM只需2次,具体为:

① 客户端根据 hust. doc 的目录路径名查询该目录对象的 AC,根据 AC 确定客户端是否具备访问 hust. doc 的权限,若没有,则拒绝访问,若有,则返回 *DiOID*;

② 客户端将 *DiOID* 与文件名 hust. doc 的合并进行 Hash,所得的值作为元数据查找表(metadata look-up table,MTL)的入口从中确定该文件所在的 MDS,获取该文件对象信息.

同时,当系统有 N 台 MDS 时就有了 $N-1$ 份冗余目录对象信息,避免仅仅由一台专门目录对象服务器而因众多客户端同时并发访问造成的 I/O 瓶颈,以及专门服务器失效而导致整个系统的无法访问.

2) 文件对象

文件对象是指目录对象中的文件及其属性,同一目录对象中的所有文件对象称为目录文件对象. 在 Linux 文件系统中,目录和文件分开存储,当读取某一个文件时至少需要 2 次磁盘 I/O 访问. 在 HMDM 中,采用与嵌入式 inode 相似的方法,将文件对象嵌入到目录文件对象中,因此目录文件对象入口项包含了该目录对象中所有文件对象,当读取目录文件对象时能一次获得该目录下的所有文件对象.

目录文件对象采用与其对应的目录对象 ID 标识,即 *DiOID*,*CountFile* 表示该目录对象下的文件对象数量(不含下属子目录中文件对象);*Entry* 则表示该目录对象下文件对象信息,包括文件名、属性和分配 SO 等信息. 元数据定位方法将在第 2 节详细介绍.

目录对象与文件对象均存储到轻量级目录访问协议(lightweight directory access protocol,LDAP),通过目录对象和文件对象到 LDAP 信息映射模块,将新建的对象信息转换成 LDAP 标准格式,并提交到后台数据库中. 例如根目录对象"/"存放为 *dn*:*cn*=00000000h,*dc*=bsomss,*dc*=sys,假设/test/usr/src 这个目录对象 ID 为 00000007h,/test/usr 目录对象 ID 为 00000004h,则/test/usr/src//hust. doc 对应为目录对象/test/usr/src/和文件对象 hust. doc,并会分别存放到相应的记录中:*dn*:*cn*=00000007h,*cn*=00000004h,*dc*=bsomss,*dc*=sys 与 *dn*:*cn*=hust. doc,*cn*=00000007h,*dc*=bsomss,*dc*=sys. 具体对象的数据结构也会被存放在此条记录下的属性项中,还可自定义属性等,再通过如 *ldap_open*(),*ldap_bind*(),*ldap_add*()等 API 函数向后台数据库提交结果. 而对于 LDAP 信息到文件对象和目录对象的映射模块中,为文件对象和目录对象到 LDAP 信息映射模块的逆过程.

2 元数据动态管理

目前元数据定位为目录子树和 Hash 两种方法. 目录子树是根据子目录确定所在的 MDS,它保持了文件系统的层次结构,由于不同目录中的文件数量和被访问的热度存在着很大的差异,虽然动态子树分割根据委托权限将子树分割到不同的 MDS 中,但仍然存在系统负载的不平衡.

Hash 是一种基于 Hash 函数的构造方法,分直接定位和间接定位两种. 直接定位是根据文件全路径名将文件的元数据平均分布到不同的 MDS 中,但文件全路径名更改时导致了大量元数据迁移;间接定位则是通过文件全路径名的 Hash 值与 MDS 建立一张 MLT,通过 MLT 确定 MDS,更换文件名需更新 MTL,但增加额外的开销,特别在 PB 级存储系统中,当更名或者 MLT 结构发生改变时,MTL 未能及时更新,Hash 值所映射的 MDS 与实际不相符,则需逐个遍历所有 MDS 中元数据进行查找定位,导致性能急剧下降;同时由于 MTL 的查询复杂度为 $O(\log n)$,n 为系统中文件个数,若系统有并行访问大量不同文件时,极大地影响元数据访问效率.

因此,本文采用局部元数据查找表(local metadata look-up table,LMTL)提供一种间接定位查询,MDS 只管理和存储自己的 LMTL,将 *DiOID* 和文件名合并的 Hash 值作为索引,入口项包括 *MDS_ID* 和版本号,并储存到自己的磁盘介质中,同时广播给其他 MDS,使每一个 MDS 都有其他 MDS 的 LMTL,可构成一张完整的 MTL(如图 1 所示):

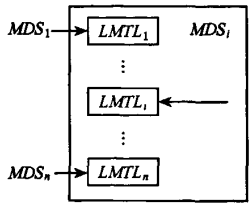

图 1 局部元数据查找表

同时,MDS 为自身的 LMTL 计算对应的 Bloom Filter 值(Bloom Ffilter value,BFV),Bloom

Filter[9-10]算法是一种以增加计算时间来节省存储空间进行快速查找的方法,每个 MDS 的 LMTL 通过 Bloom Filter 算法生成一个摘要串,概括该节点上有哪些文件元数据,没有哪些文件元数据.将这些摘要串传给客户端,节省客户端大量的存储空间,提高检索速度,并减少信息传输,降低了访问延迟.如果摘要串认为该节点没有某个文件,则该节点必然没有这个文件,这样过滤了该文件;如该摘要认为该节点可能有某个文件,则该节点以很大的可能性存有这个文件,能够极大地提高查询时间.

由于采用 Bloom Filter 算法不会错误地否定,但查询元素在集合 S 中存在一定的错误率(false positive rate,FPR).使用 k 个 Hash 函数的 m 位长的 Bloom Filter 中装入 n 个元素后向量中某一位仍然为 0 的概率为

$$f=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}, \tag{1}$$

因为

$$(1-1/m)^{kn}\approx e^{-kn/m}, \tag{2}$$

$$f\approx(1-e^{-kn/m})^{k}, \tag{3}$$

对式(3)进行等价转化,得

$$f=e^{k\ln(1-e^{-kn/m})}, \tag{4}$$

设 $g=k\ln(1-e^{-kn/m})$,并对式子两边求导,令 $\frac{dg}{dk}=0$,得到 Hash 函数个数的最小值:

$$k_{\min}=(\ln 2)\left(\frac{m}{n}\right), \tag{5}$$

存在的错误率为

$$f(k_{\min})=(1/2)^{k}=(0.6185)^{m/n}. \tag{6}$$

Bloom Filter 算法的性能主要取决于错误率,其越小算法性能越好.因此,可以通过选择合适的 Hash 函数、Hash 函数的个数及适合的 m/n 值使错误率更小.图 2 表示 k 与 m/m 之间成正比例关系,错误率随着 m/n 或者 k 的增加而降低(如图 3 所示).

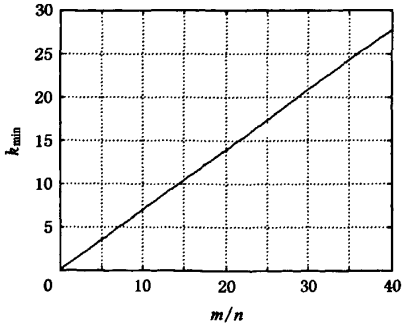

图 2　m/n 与 k 之间的关系

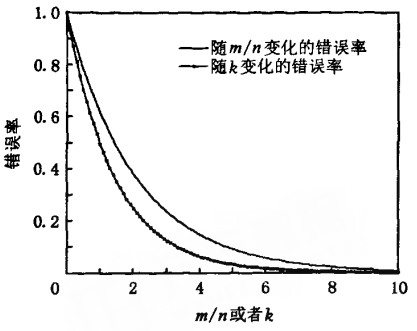

图 3　k,m/n 与错误率之间的关系

具体实现如算法 1 所示,其中假设 $m=1024$,A 是用 m 个比特单元构成的数组,初始时所有比特为 0;$key=DiOID+$文件名,n 为 MDS 个数,k 由式(5)计算;$count$ 为计数器,记载表示 BFV_i 存在该文件的个数,str 记录着 BFV_i 存在该文件的 MDS 的 id,初始值均为零或者空.

算法 1.

BF_lookup(key)

1) $HV=Hash(key)$;

2) $A[h_i(HV)]=1,i=1,\cdots,k$; /* 使用 k 个相互独立的 Hash 函数 $h_1,h_2,\cdots,h_k$ 将数组 A 对应位置设为 1 */

3) for($i=1;i<=n;i++$)

{if $A\in BFV_i$ /* 即检查 A 中所有 1 的位数在 $LMTL_i$ 对应 BFV_i 的相应位数是否为 1 */

$count=count+1$,$str=str+i+$" ";

}

4) if $count=1$,str 中对应的 MDS 的 id 就是所找的,返回成功的 MDS_id;

5) if $count>1$ and $count\neq n$,从 str 对应的 MDS 的 id 的 LMTL 中查找,返回成功的 MDS_id;

6) if $count=0$,该文件不在所有 MDS 中,返回无此文件.

因此,当错误率低时采用 Bloom Filter 算法进行过滤能够一次找到文件所在地 MDS,查询复杂度为 $O(1)$;而最坏情况下,即发生错误率为 100%,则需查找由所有的 LMTB 所组成的 MTB,查询复杂度为 $O(\log n)$.因此,在系统不断地扩大时,需要合适地采用式(5)选择 Hash 函数、Hash 函数的个数及 m/n 值,以最大程度降低错误率.虽然重新选择时增加了系统的一定开销,但能够满足系统请求时元数据快速的查找和定位.

3 结束语

在PB级大规模存储系统中,需要由数十台元数据服务器满足成千上万客户进行并发访问的要求.因此,将元数据按层次结构分为目录对象和文件对象,每一个MDS负责所有目录对象和自己的文件对象,同时采用Bloom Filter算法将每一个MDS的LMLT压缩成一个摘要,实现快速的元数据查找和定位.这样不仅动态扩展元数据,而且一旦改变目录权限、改变目录名、移动目录都不会造成元数据的迁移.

参 考 文 献

[1] McKusick M K, Joy W N, Leffler S J, et al. A fast file system for Unix. ACM Trans on Computer Systems, 1984, 2(3): 181-197

[2] Mesnier M, Ganger G R, Riedel E. Object-based storage. Communications Magazine, 2003, 41(8): 84-90

[3] Liu Qun, Feng Dan. An approximate analytic performance model of object-based storage //LNCS 3980: Proc of the Int Conf on Computational Science and Its Applications. Berlin: Springer, 2006: 1045-1052

[4] Ousterhout J K, Costa H D, Harrison D, et al. A trace-driven analysis of the Unix 4.2 BSD file system //Proc of the 10th ACM Symp on Operating Systems Principles. New York: ACM, 1985: 15-24

[5] Weil S A, Pollack K T, Brandt S A, et al. Dynamic metadata management for petabyte-scale file systems //Proc of the 2004 ACM/IEEE Conf on Supercomputing. Washington: IEEE Computer Society, 2004: 1-4

[6] Brandt S A, Xue L, Miller E L, et al. Efficient metadata management in large distributed file systems //Proc of the 20th IEEE/11th NASA Goddard Conf on Mass Storage Systems and Technologies. Los Alamitos: IEEE Computer Society, 2003: 290-298

[7] Li Weijia, Xue Wei, Shu Jiwu, et al. Dynamic hashing: adaptive metadata management for petabyte-scale file systems //Proc of the 23rd IEEE/14th NASA Goddard Conf on Mass Storage Systems and Technologies. Los Alamitos: IEEE Computer Society, 2006: 15-18

[8] 王召福, 章文嵩, 刘仲. 大规模集群文件系统LCFS的元数据管理与访问机制. 计算机工程与科学, 2005, 27(18): 103-105

[9] Burton Bloom. Space time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7): 422-426

[10] Zhu Yifeng, Jiang Hong, Wang Jun. Hierarchical bloom filter arrays (HBA): A novel, scalable metadata management system for large cluster-based storage //Proc of the 2004 Int Conf on Cluster Computing. Los Alamitos: IEEE Computer Society, 2004: 165-174

刘 群 女,1969年生,博士,工程师,主要研究方向为高性能网络存储、计算机网络,先后在国内外发表论文10余篇.

冯 丹 女,1970年生,教授,博士生导师,主要研究方向为计算机系统结构、磁盘阵列技术、高性能网络存储.

基于层次结构的元数据动态管理方法的研究

作者： 刘群， 冯丹， Liu Qun， Feng Dan
作者单位： 刘群,Liu Qun(华中科技大学网络与计算中心,武汉,430074)， 冯丹,Feng Dan(华中科技大学计算机科学与技术学院,武汉,430074)
刊名： 计算机研究与发展 ISTIC EI PKU
英文刊名： JOURNAL OF COMPUTER RESEARCH AND DEVELOPMENT
年，卷(期)： 2009，46(z2)
被引用次数： 0次

参考文献(10条)

1. McKusick M K. Joy W N. Leffler S J A fast file system for Unix 1984(3)

2. Mesnier M. Ganger G R. Riedel E Object-based storage 2003(8)

3. Liu Qun. Feng Dan An approximate analytic performance model of object-based storage 2006

4. Ousterhout J K. Costa H D. Harrison D A tracedriven analysis of the Unix 4.2 BSD file system 1985

5. Weil S A. Pollack K T. Brandt S A Dynamic metadata management for petabyte-scale file systems 2004

6. Brandt S A. Xue L. Miller E L Efficient metadata management in large distributed file systems 2003

7. Li Weijia. Xue Wei. Shu Jiwu Dynamic hashing:adaptive metadata management for petabyte-scale file systems 2006

8. 王召福. 章文嵩. 刘仲 大规模集群文件系统LCFS的元数据管理与访问机制[期刊论文]-计算机工程与科学 2005(8)

9. Burton Bloom Space time trade-offs in hash coding with allowable errors 1970(7)

10. Zhu Yifeng. Jiang Hong. Wang Jun Hierarchical bloom filter arrays(HBA):A novd,sealable metadata management system for large cluster-based storage 2004

相似文献(10条)

1. 学位论文 殷武峰 元数据管理技术在商业智能系统中的研究与应用 2007

随着科学技术和数据仓库技术的发展，管理和访问大型数据集的复杂性已成为数据生产者和用户共同面临的突出问题，数据生产者需要有效的办法来组织、管理和维护海量数据。元数据作为描述数据的内容、质量、状况和其它特性的信息的作用已变得越来越重要，成为信息资源的有效管理和应用的重要手段。

针对商业智能系统的数据异构，用户层次多，以及数据范围广的特点，做了一些工作。主要包括：

1. 设计了商业智能系统的元数据管理系统。该系统采用JAVA语言开发，实现了平台无关性，采用了当前流行的XML，XSLT等技术，同时使用了很多目前成熟的开源软件项目。元数据管理系统充分发挥了XML的技术优势，特别是XML的分离性。

2. 对元数据层次结构建模做出了研究。具体研究是非齐整非平衡及其混合层次结构在元数据和OLAP中的实现。

创新之处：

1. 主要创新之处是结合目前的一些元数据标准规范，开发实现了自己的元数据管理系统。

2. 在元数据和OLAP中实现了非齐整非平衡及其混合层次结构。

2. 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）技术逢时崛起，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，既有了“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，BSO-MSS）吸取了OBS 的优点，在“对象”现有的含义基础上扩充，使它不仅只包括用户数据，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BSO-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BSO-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BSO-MSS动力演变规律。构建了一个通用框架的BSO-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BSO-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，而且能够动态扩展MDS。同时，以目录对象ID 和文件名为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BSO-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一维空间中命名，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BSO-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BSO-MSS 具有更高的性能、可扩展性和自适应负载均衡能力。

3.学位论文 吴婷 海量存储系统中元数据管理机制的研究 2010

海量存储系统中需要保存Terabyte、Petabyte级别甚至更大规模的数据。数据的元数据如文件的名字、属性、保存地址和访问授权等信息一般由元数据服务器进行管理。在访问海量存储系统的数据前，需要首先查找和获得元数据。因此元数据管理机制将直接关系到海量存储系统的I/O性能。

现有的海量存储系统一般采用目录层次结构和哈希算法管理元数据，存在修改元数据和查询目录等操作所需时间和空间开销大等问题，也没有针对海量存储系统中元数据访问特性的优化机制，严重制约了海量存储系统的I/O性能。

本文在分析海量存储系统中元数据管理特性的基础上，引入DBMS技术以及数据分级的方法，提高管理元数据的效率。论文的具体工作包括：

首先引入二维表保存系统中的元数据信息，提出了基于DBMS的新型元数据管理策略，给出了各类元数据操作的流程；分析了在海量存储系统中用于管理元数据信息时所需的时间和空间开销以及适应不同运行环境的能力；实现了基于DBMS元数据管理策略的原型系统，采集实际文件系统中的元数据，构建多种测试环境进行测试与分析，结果表明基于DBMS的元数据管理策略能有效地减少管理元数据所需的时间和空间开销，提高管理元数据的灵活性，增强适应能力。

在分析海量存储系统中元数据时间特性的基础上，依据元数据的生命周期，设计了元数据分级算法，将元数据分为活跃元数据和非活跃元数据；设计了分区索引算法，提高查询活跃元数据的性能；改进了基于哈希函数的索引方法，设计了非活跃元数据的索引算法，减少了管理非活跃元数据所需的时间与空间开销；从查找元数据与更新索引所需的时间与空间开销两方面进行了分析，验证了其能有效地减少了查询元数据和更新索引所需的时间和空间开销；实现了元数据分级索引算法的原型系统，采集实际文件系统中的元数据，构建多种测试环境进行测试与分析，结果表明元数据分级索引算法能有效地提高查询元数据的性能。

关键词：海量存储系统，元数据管理，数据库管理技术，数据生命周期，索引算法

4.学位论文 李伟 协同GIS理论模型与技术研究 2005

本文的研究重点是协同GIS理论模型与技术。作者在分析第二代互联网条件下分布式GIS软件有关新技术特征的基础上，探讨了以分布式对象技术支持下的GIS协同工作机制，构建了面向GIS的协同模型，实现了对面向GIS的协同工作环境的模拟，开发了支持同步/异步协同地图编辑行为的分布式GIS原型系统。

论文具体研究工作主要包括如下几个方面：

1、分针对GIS的特点，参考当前主流的分布式对象技术，着重研究和讨论了分布式GIS的软件设计方法、体系结构、对象设计方法以及关键技术的实现策略等内容。通过对分布式GIS相关理论及技术的研究，确定了论文所要实现的分布式协同GIS的软件框架，解决了实现分布式对象及构建原型系统的部分技术难题。

2、协同工作特征是网络新时代下新一代GIS的重要特征。对于协同GIS的研究工作，论文研究工作主要集中在面向GIS的协同模型上，其主要包括基于任务组模式的协同工作模型、面向协同GIS特征的空间数据模型和基于Multi-Agent的混合协同交流模型。(1)协同工作模型是对客观世界中协同交互对象及行为相互关系的逻辑抽象，是协同工作系统建立的基础。论文给出了基于任务组模式的协同工作模型，并通过形式化语言和面向对象的模型设计方法对其进行了定义和描述。(2)论文首次提出了面向协同GIS特征的空间数据模型—OCFSDM。OCFSDM是协同工作环境下具有协同特征的地理空间数据模型。对其的讨论，论文从认知过程、框架结构和层次结构三个方面展开，结合UML等建模语言对其进行定义和描述，并总结了OCFSDM的特征。(3)协同交流模型是对协同工作环境中信息主体之间信息交互行为的模拟，是协同工作系统的重要研究内容。基于Multi-Agent技术，论文提出了基于Multi-Agent的混合协同交流模型—MAMCIM，并从纵向和横向两个角度对其进行分析和讨论；并对其实现机制进行了讨论和分析。

3、GIS数据互操作是数据层面上的协同工作。论文在分布式框架下，结合现有实现GIS数据互操作的策略，给出分布式环境下，多源异构空间数据互操作引擎的设计思路。基于面向对象及开放性思想，本部分对引擎的体系结构以及引擎的各个组成部分分别进行讨论，其中主要包括开放多源空间数据库连接(OMSDBC)模块、异构空间数据转换(DSSDT)模块、空间元数据管理模块等。

4、基于分布式对象等技术，论文构建了分布式环境下支持GIS协同工作机制的原型系统，并针对协同GIS同步/异步协同地图编辑行为，给出了解决协同工作环境下，协同用户间行为的协同感知和协同冲突处理等技术难题的实现技术方法。

研究及测试结果表明，分布式对象技术是解决当前GIS软件所面临诸多问题的有力手段；论文所设计的协同模型能较好地对面向GIS的协同环境进行模拟，支持同步/异步环境下对地图编辑行为的协同处理：面向分布式的开放多源异构空间数据互操作引擎为实现GIS数据互操作提供了崭新的解决思路。

5.学位论文 裴军峰 基于P2P技术的渔业空间数据共享的研究 2007

地理信息系统(GIS)能进行有效的空间数据管理和决策分析，已经在资源环境调查、数字农业、数字海洋和数字地球等多个领域中得到了广泛应用，并已形成海量的地理空间数据。进一步共享这些海量空间数据，对提高人们的工作效率、降低生产成本，特别是对构建国家空间信息基础设施和实施数字地球战略具有重要意义。P2P(Peer-to-Peer，简称P2P)最近几年来迅速成为计算机界关注的热门话题之一，财富杂志更将P2P列为影响Internet未来的四项科技之一。在P2P体系结构中，每个对等节点既扮演服务器的角色又扮演客户端的角色，实现了对等节点之间资源的传输和共享。这种节点之间完全对等的方式使得P2P技术在数据共享方面有着广阔的发展前景。

本文首先介绍了P2P技术、渔业GIS和空间数据共享的现状，对国内外相关研究现状进行了介绍。然后分析了P2P的四种不同的网络模型，包括集中目

录式模型、非结构化网络模型、结构化网络模型以及混合式模型的优缺点，讨论了他们和GIS结合的可能性。随后介绍了SUN公司推出的开源的P2P开发平台JXTA，分析JXTA在开发P2P应用平台上的优势，以及层次结构和核心协议。

在研究现有空间数据共享理论和技术的基础上，结合P2P技术负载均衡、可扩展性和健壮性等优点，提出了基于P2P技术的空间数据共享体系。该体系中的资源和服务分散在所有节点上，信息的传输和服务的实现都直接在节点之间进行，除目录查询外的服务无需中间环节和服务器的介入。体系包括七个部分，客户端：代表用户与PMS服务器交互，负责发送地图请求以及可视地图的显示；PMS服务器：其主要作用是为客户端提供透明的访问地图服务。负责和中心服务器的交互(注册、获得目标用户列表等)、地图数据的获取；中心服务器：包括目录服务器DS和PMS种子服务器。目录服务器DS：提供注册登记服务、查询服务和初始路由服务。PMS种子服务器：提供下载服务，是一种特殊的PMS服务器(长期在线提供种子)。元数据P2P服务器：负责各个全局性(所有已经注册的空间数据服务器)的空间元数据管理、提供空间元数据的注册与变更服务。数据P2P服务器：提供空间数据，向元数据：P2P服务器进行元数据的注册与变更。然后对PMS服务器的路由查找策略进行了设计和描述，分析了节点加入和退出时候应该注意的问题。

最后借助SUN公司推出的开源P2P开发平台JXTA，采用某地区部分渔业相关空间数据，对基于P2P的空间数据共享系统进行了模拟，实验验证了其正确性和可行性，为空间数据共享提供了一种新的思路。

6.学位论文 张颖 数据仓库与数据挖掘技术在移动行业中的应用研究 2006

当前电信运营企业之间的竞争日益加剧，由于硬件设施的差距在减小，竞争最终体现在对客户的服务上。为了能够提供高质量的服务来吸引和留住客户、扩大市场份额、降低运营成本、提高收益，国内各大运营企业都启动了经营分析系统的建设工作，经营分析系统主要通过数据仓库、联机分析处理和数据挖掘技术完成各类分析功能。

本文主要研究了移动经营分析系统中的数据仓库和数据挖掘技术。本文首先介绍了数据仓库和数据挖掘的基本概念，接着论述了移动经营分析系统的体系结构和功能框架，在此基础上重点论述了数据仓库的建模和ETL(抽取、转换和加载)过程，并给出了数据挖掘的实例。本文以客户主题为例，设计了数据仓库的数据模型，并对数据仓库设计的关键环节ETL过程进行了设计，给出了ETL子系统的模块层次结构，深入研究了文件传输模块、数据调度模块、数据ETL模块及ETL元数据管理模块的功能及处理流程。然后本文以高价值用户离网预测为例，详细阐述了数据挖掘的步骤，给出了数据理解、数据准备、建立模型、模型评估和模型发布的过程，该例分别采用C5．0、分类和回归树(C&RTree)及神经网络节点建立预测模型，使用神经网络创建了打分模型，并对预测模型和打分模型进行了评估。本文的最后部分论述了建设经营分析系统的几个关键点，并提出了后续研究的发展方向。

7.学位论文 熊劲 大规模机群文件系统的关键技术研究 2006

机群结构已成为高性能计算机的主流结构。随着CPU处理能力和通信速度的迅速提高，I／O成为制约机群应用实际性能的瓶颈。机群文件系统作为解决机群I／O瓶颈的核心技术，其研究具有重要的意义。

机群文件系统的发展趋势为：第一，元数据处理与文件I／O分离；第二，利用大规模网络存储系统来提供多条数据I／O通路；第三，利用一组元数据服务器来提供多条元数据I／O通路。

针对这种结构的机群文件系统，我们研究了其中的几个关键问题，包括元数据的分布问题，元数据的一致性和快速故障恢复问题，以及PB级机群文件系统的相关问题。本文的主要贡献在于：

(1)提出一种高效的大存储空间的管理策略——Bitmap-Extent混合策略。针对PB级机群文件系统，打破了传统文件系统基于一个块设备的限制，提出将机群文件系统与物理存储分离的一种逻辑空间策略，从而解决了文件系统容量受限问题和存储扩展问题等；而且针对PB级存储空间管理，提出一种基于位图与extent链表相结合的大规模存储空间管理机制，以提高存储空间的管理效率。

(2)提出一种基于粒度的动态元数据分布策略。元数据分布问题是决定非集中式元数据处理性能的关键问题。我们提出的基于粒度的动态元数据分布策略以提高元数据处理整体性能为目标，综合考虑元数据分布均衡度和文件系统层次结构关系两个因素对元数据处理整体性能的影响，按照一定粒度来划分名字空间和分布元数据，实验结果表明在模拟真实环境的负载下它的性能高于动态随机分布策略和动态根子树分布策略。

(3)提出一种基于简化的两阶段提交协议的、故障后可快速恢复元数据一致性的分布式元数据处理协议。元数据一致性问题是影响分布式元数据管理的可靠性和高可用性的关键问题。为了解决元数据服务器之间的元数据一致性问题，我们将两阶段提交协议与元数据的处理协议结合起来，提出一种基于简化的两阶段提交协议的分布式元数据处理协议，在元数据服务器失效或客户节点失效时，能够快速恢复文件系统的元数据一致性，保证文件系统的可用性。

(4)设计和实现了面向多用户多任务环境的、支持大规模机群系统的、面向海量数据存储的机群文件系统。DCFS2。在机群文件系统性能评价方面，提出从峰值性能、稳定性能、系统规模扩展性、元数据服务器扩展性、存储设备扩展性和存储I／O带宽利用率六个性能评价指标。并用这六个指标对DCFS2的性能进行全面评价。我们的结果表明，DCFS2能够获得比GFS等文件系统更高的聚合I／O带宽和聚合元数据处理性能。

8.学位论文 江恭和 BI中即席查询及分析技术的研究实现 2008

商业智能(BI)是一种运用了数据仓库、联机分析处理和数据挖掘等技术来处理和分析数据的软件系统，目的是为企业决策者提供决策支持。商业智能在电信、保险、金融、商业等领域应用广泛。商业智能在政府中也得到了应用，分析型电子政务应用如领导查询、经济分析、决策支持，它们很大程度上依赖于商业智能的技术和方法。

本文根据在建设多个分析型电子政务应用中即席查询与分析的需求基础上，对BI中的即席查询与分析技术进行研究和实现。即席查询是BI中的核心功能，它能让用户根据需要从数据仓库中获取所关心的数据，进行分析和处理。

文中分析了维的正常、不齐整、不平衡以及不平衡和不齐整层次结构的特点、并进一步分析了在数据聚焦与不聚集情况下，层级结构建模的选择与实现，并分别分析了数据仓库星型模型和雪花模型中事实表与维表数据存储的特点，从而为实现相应算法如检索、聚集、钻取打下基础。

研究的即席查询作为一种数据仓库分析工具，需要支持对多维数据进行操作，也即OLAP的一些操作，文中分析了OLAP中的一些概念、数据立方体计算的一些实现方法，从而为实现数据OLAP操作的算法提供基础。

针对即席查询的不同需求，提出了二种不同即席查询，一种为面向一个主题的查询，另外一种为面向多个主题的综合性查询，以支持对不同指标的综合分析和处理。为了提高对数据的分析能力，系统中引入了模型库系统，利用数据库的BLOB字段存储模型的Java执行文件，通过数据库CRUD操作实现对模型的维护，并通过Java反射机制实现模型的调用，实现模型的动态增加和动态部署。此外，设计了元数据管理工具，以支持图形化形式对Cube进行建模，CubeSchema以XML形式发布。

在软件实现方面系统基于SOA架构，系统建设过程遵循了RUP统一过程进行研发。首先分析了系统的功能，然后进行了软件架构设计、系统的详细设计，以及系统的实现。最后对论文进行了总结，并对系统的进一步完善提出了展望。

9.期刊论文 吴婷.鞠时光.蔡涛.WU Ting.JU Shi-guang.CAI Tao 基于DBMS的元数据管理策略 -计算机应用研究 2010,27(4)

海量存储系统的元数据一般采用层次结构或哈希法来管理,存在元数据修改和查询目录等操作所需时间和空间开销大等问题,严重影响了系统的性能.通过引入二维表保存元数据信息,提出了一种基于DBMS的新型元数据管理策略.分析了将基于DBMS元数据管理策略用于管理海量存储系统中的元数据信息时,所需的时空开销以及管理元数据的灵活性.验证了基于DBMS元数据管理策略能有效地减少查询和更新所需的时空开销,实现高效、灵活的元数据管理功能,从而有效地提高海量存储系统的性能.

10.学位论文 李嘉佑 Web信息智能获取系统GHunt 2005

因特网(Internet)为人们开辟了一个共同的、全新的天地。人们在这个虚拟的世界里，以一种全新的方式进行交流。任何人在任何时间、任何地点都可以通过网络发布任何信息，这使得网络成为最重要的信息来源。但是面对潮水般涌来的电子文献，人们变得无所适从。所以研究Web信息智能获取技术以帮助用户快速、准确地定位到自己需要的信息，具有广泛的应用背景和实用价值，已经成为近年来的研究热点。本文包括以下重点内容：

(1)通过分析Web数据存在的特点，实现Web页面解析和清洗方法：基于URL标记信息的噪音判别、基于噪音数据冗余特点的判别、基于URL标记文本信息熵的噪音数据判别，并且基于这三种判别方法实现了Web噪音数据去除算法。实验结果验证了三种噪音去除方法的可行性和有效性。Web噪音去除算法可以有效解决数据质量问题，从而可以提高后续文本分类、聚类、查询等文本挖掘算法的性能。

(2)本文先按照已有概念空间的层次结构对相关文档进行共现分析，然后，用基于直接聚类的概念空间生成方式实现文本语义检索机制。

科学研究的最终目的是研究成果能够在实际中得到应用。基于此目的，笔者将本文的研究结果和实验室其他同学的研究成果有机地结合起来，实现

了一个完整的网络信息智能获取和处理系统GHunt。本文给出了系统的功能框架和体系结构，提出了GHunt对信息获取过程和获取知识的元数据管理模型，并用实验结果验证了系统的性能。GHunt是笔者多年来研究成果的结晶，是研究成果走向产品化的重要环节，同时也为后续研究开发提供了一个很好的平台和检验环境。如果想详细了解GHunt系统，可访问地址http://www.intsci.ac.cn/GHuntWeb/。