一种基于对象存储系统的元数据缓存实现方法*)

周功业　吴伟杰　陈进才

（华中科技大学计算机科学与技术学院　武汉 430074）（武汉光电国家实验室　武汉 430074）

摘　要　对象存储系统中元数据访问速度是影响文件系统性能的关键因素之一。提出了一种在客户端实现元数据缓存的方法，并用元数据操作协议保证缓存一致性，基于 Hash 的 LFU-DA 算法提高缓存查找效率。实验表明该方法减少了系统平均服务响应时间，提高了系统的 I/O 性能。

关键词　对象存储，元数据，元数据缓存，操作协议，LFU-DA

An Implementation of Metadata Cache in Object-based Storage System

ZHOU Gong-Ye　WU Wei-Jie　CHEN Jin-Cai

(College of Computer Science and Technology, Huazhong University of Science andTechnology, Wuhan 430074)

(Wuhan National Laboratory for Optoelectronics, Wuhan 430074)

Abstract　The metadata accessing speed is one of key factors of system performance in object-based storage system. A method for implementation of metadata cache presents at clients. The protocol of metadata operation guarantees the consistency for metadata cache. The hash-based LFU-DA increases the efficiency of searching metadata cache. Experimentation results indicate that the method improves average response time and I/O performance remarkably.

Keywords　Object-based storage, Metadata, Metadata cache, Operation protocol, LFU-DA

1　引言

对象存储正在成为存储领域研究的热点，它结合了 NAS 和 SAN 的优点。基于对象的存储系统已经成为构建大型分布式系统的优选方案，如图 1 所示，此方案实现了元数据与数据对象的分离。基于对象的存储设备(OSD)对外呈现对象访问接口，管理底层的存储任务(如数据的布局和存取)，维持数据对象和存储介质上实际块之间的联系。传统文件系统的数据管理工作被分布到智能的 OSD 上，由其负责管理数据分布和检索，这样传统共享存储系统中约 90%的元数据管理工作被分布到智能的存储设备上。但在海量存储系统中，超过 50%的操作是元数据操作[1,2]，元数据服务器(MDS)仍可能成为系统的瓶颈。目前普遍采用分布式机群来管理元数据的操作和文件到对象的映射，并执行相应安全策略。现已提出目录子树分割法[3]、纯散列分割法[4]、Lazy Hyrid(LH)法[5]、散列分割法(HAP)[6]等元数据服务器管理策略来解决系统连续可用性，提高服务器的性能，但这些策略都在 MDS 端实现，未能充分发挥客户群的处理能力。本文提出了一种在客户端实现元数据缓存的方法，有效降低了元数据操作访问 MDS 的次数，减少了系统服务平均响应时间。

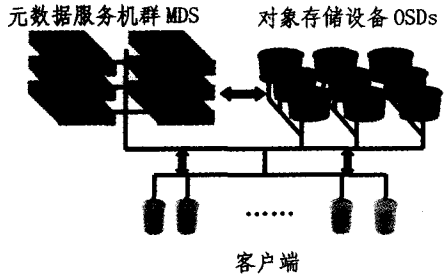

图 1　对象存储系统体系结构

2　元数据缓存方法与实现技术

2.1　客户端元数据缓存的有效性

元数据是文件系统中最为重要的特征信息，实现高速元数据访问是提高文件系统性能的有效途径。分布式并行文件系统中用户对数据对象的任何操作都需要事先访问元数据，对数据对象所做出的任何改动都必须记录在元数据中，可见用户对元数据的访问是非常频繁的。因此，提高元数据访问的速度，对提升文件系统的性能非常重要。客户端上建立元数据缓存可以有效减少访问元数据服务器带来的大量网络通信开销，从而提高元数据访问的速度，提高文件系统性能。

对象存储系统中 MDS 维护的元数据本身占用的空间并不大，通常一个对象的元数据所占用的空间大概只有一百个字节到两百个字节，因此，在客户端建立元数据缓存并不会占用太大的内存空间，在客户端建立缓存是可行的。当应用程序访问一个数据对象时，首先试图从本地元数据缓存中获取被访问对象的元数据，一旦命中，将无需与元数据服务器进行网络通信，即使没有命中，由于访问本地内存的速度远高于网络带宽，对于系统性能也不会造成太大的影响。

2.2　元数据操作协议

在客户端建立元数据缓存后元数据网络访问控制协议如图 2 所示，下面以读文件数据为例，其他操作只是最后几步略有区别，可以看出读文件整个过程主要分如下几个步骤：

1：客户端把要访问文件的路径转化为全局文件系统唯一路径名，应用进程查询元数据缓存，判断缓存是否命中，命中进入步骤 4；

2：元数据缓存不命中则客户端与元数据服务器建立连接，发送元数据查询请求；

3：元数据服务器对客户端访问进行身份验证，合法则查

*)基金项目：国家 973 计划项目(2004C318201)。**周功业**　教授，CCF 会员，主要研究方向为计算机网络，嵌入式系统及应用，计算机外存储技术；**吴伟杰**　硕士研究生，研究方向为网络存储、嵌入式系统；**陈进才**　副教授，CCF 会员，研究方向为计算机存储技术，嵌入式系统，计算机网络。

询元数据，确认所访问的文件是否存在，发送应答信息到客户端，进入步骤5；

4：客户端元数据缓存命中则应答应用进程；

5：客户端应用进程收到应答信息后，访问对象存在则发送读对象请求到元数据服务器，否则报错，过程结束；

6：元数据服务器收到客户读对象请求后，发送对象ID到相关对象存储节点；

7：对象存储节点验证元数据服务器访问请求后，打开对象，发送应答信息到元数据服务器；

8：元数据服务器收到应答后，把对客户端的授权、文件与存储对象ID映射表以及存储对象节点位置信息应答给客户端，同时更新元数据缓存；

9：客户端收到元数据服务器的应答信息后，向所收到元数据对应的对象存储节点发读对象请求命令包，与相应对象存储节点建立连接；

10：对象存储节点验证用户请求，确认后发送数据对象给客户端，直到数据传输结束。

被广泛应用的并行文件系统PVFS等元数据操作需要两次访问MDS，协议实现简单但增加了网络开销。上面所述元数据网络访问控制协议在元数据缓存在命中情况下可以省略一次元数据服务器的访问，同时不失简单性。

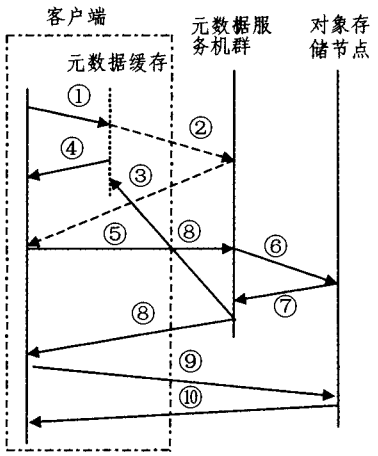

图2　元数据操作协议

2.3　一致性分析

分析上述元数据操作协议，对象元数据一致性是由客户端第二次元数据访问即步骤5～8来保证，因为所有操作的真正元数据都是步骤8从元数据服务器返回的，并没有直接从本地的元数据缓存中读取。客户端打开一个对象或做其它元数据相关操作时，至少需要访问一次元数据服务器，用户对对象的任何操作都必须通知元数据服务器，时刻保证对象元数据的正确性；对象存储节点只接收来自元数据服务器的打开请求。以上两条保证了元数据的一致性。假设某一客户端在元数据缓存中持有一个对象的元数据期间，元数据因其它客户端操作而发生了改变，导致了元数据出现不一致情况。第一次元数据访问请求即步骤1～4的目的只是为了检查数据对象是否存在，第二次元数据访问请求即步骤5～8并没有从发生不一致的元数据缓存中获取元数据，相反，第二次元数据访问所获得的元数据还将更新元数据缓存，由此避免了访问数据时系统中元数据缓存不一致性的发生。

2.4　查找置换算法

一个容量为几兆的元数据缓存可以容纳上万块缓存数据块，一个元数据缓存块对应着一个对象元数据，从而可以缓存上万个对象元数据。对于一般应用来说，如果把元数据缓存容量配置更大一些的话，元数据缓存块几乎没有被置换出去的机会。因此置换算法的好坏不是影响缓存性能的最关键因素，不过缓存的查找算法却显得非常重要。

LFU-DA算法同时考虑了“访问时间”和“访问频率”两个方面，比只考虑访问时间的算法有更高的命中率。然而当元数据缓存块数量很大时，LFU-DA算法有一个弊端：会造成缓存队列过长，导致应用程序在缓存中查找数据耗时太久，降低了元数据缓存的效率。针对以上不足，本文提出了一种改进的LFU-DA算法——基于hash的LFU-DA算法，具体的做法是：将过长的缓存队列根据所选hash函数分成m个小队列，用缓存数据块的块号和数据块所属对象的ID求hash值，然后根据所求hash值把数据块插入到对应的队列中。每一个hash队列都是LFU-DA队列。这样就可以加快缓存的查找速度。证明方法很简单：假设LFU-DA算法的平均查找时间是t，系统缓存队列被分成m个队列，那么改进后算法的平均查找时间约为t/m。

3　实验结果与分析

为了验证在客户端建立元数据缓存的效果，分别测试了添加元数据缓存前后各项对象操作性能。测试硬件环境如表1所示。实验环境中对象存储系统由1个客户端结点，4个元数据服务结点和8个对象存储结点组成。元数据缓存块的大小定为256B，元数据缓存总大小定为5M，也就是总共20000个元数据缓存块。

表1　测试硬件环境

CPU	AMD 2500+
主存	1G
网卡	1000Mb
操作系统	Linux 2.4.20

首先测试没有元数据缓存情况下各元数据相关操作耗费的时间，然后测试添加了元数据缓存后各操作在缓存命中和未命中时的有关操作耗费的时间。测试结果如表2所示。可以看出在添加元数据缓存之后，在缓存没有命中的情况下，各操作耗费的时间比没有添加元数据缓存时的相同操作耗费的时间要稍多一些，这是由添加元数据缓存之后数据块查找开销引起的，但差距是非常微小的，这表明即使要查找的元数据不在缓存中，对系统性能也不会有多少影响。一旦元数据缓存命中，对象操作用时便成倍数地减少。数据对象操作性能提高尤以“CREATE”、“READ”和“WRITE”为甚，提高倍数达14.5倍，其他对象操作平均提高80%。

表2　添加元数据缓存前后各操作耗时(ms)

耗时		CREATE	READ	WRITE	LIST
无缓存		23.42	25.12	24.37	2.13
命中	否	24.03	25.41	24.86	2.25
	是	1.65	1.71	1.68	1.54
耗时		APPEND	GET	FLUSH	SET
无缓存		3.45	1.95	1.56	2.06
命中	否	3.57	2.06	1.73	2.17
	是	1.64	1.02	1.48	1.11

结论　提高元数据的访问速度可以提高服务响应时间，

特别是对于运行元数据密集型应用的集群系统来说至关重要。在客户端设立元数据缓存，经过测试，在缓存命中情况下各操作都有性能提升，有的甚至出现几倍乃至十倍以上的提高。由于元数据缓存块很小决定了几兆容量的内存就能保证极高的命中率，很少有置换情况发生，采用基于 hash 的 LFU-DA 算法改善了元数据缓存查找性能。

参 考 文 献

1 Ousterhout Jk, Costa H D, Harrison D, Kunze J A, Kupfer M, Thompson J G. A tracedriven analysis of the Unix 4. 2 SD file system. In: Proc. eedings of the 10th ACM Symposium on Operating Systems Principles(SOSP'85), Dec. 1985. 15～24

2 Roselli D, Lorch J, Anderson T. A compare-ison of file system workloads. In: Proceedings of the 2000 USENIX Annual Technical Conference, June 2000. 41～54

3 Morris J H, Satyanarayanan M, Conner M H, et al. Andrew: A Distriuted Personal Computing Enviro-nment. Communications of the ACM, 1986, 29(3): 184～201

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

5 Brandt S A, Xue Lan, Miller E L, et al. Efficient Metadata Management in Large Distriuted File System. In: Proceedings of the 20th IEEE 11th NASA Goddard Conference on Mass Storage Systems and Technologies, 2003(4): 290～298

6 Yan Jie, Zhu Yaolong, Xiong Hui. A Design of Meta data Server Cluster in Large Distriuted Oject-based Storage. In: 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies, 2004(4): 13～16

(上接第 111 页)

3.3 方案的安全性证明

定理 1 即使在一个 PPT 敌手勾结了 $t(t<n/2)$ 个签名用户的情况下，所提出的门限签名方案 NTS 也是安全的，即 NTS 方案具有健壮性。

证明：NTS 方案的 TK 算法是采用 GJKR 算法实现的，Gennaro 等人已在文[6]中证明了在敌手勾结用户数 $t<n/2$ 的情况下，GJKR 算法能成功运行，则 TK 算法也能成功运行。对于 TS 算法，由于部分签名的有效性可以通过(2)式进行验证，即使敌手勾结了 $t(t<n/2)$ 个用户，那么从剩余的诚实用户中也能选出 $t+1$ 个用户完成门限签名，也就是说 TS 算法也能成功运行。

定理 2 在随机预言模型下，所提出的门限签名方案 NTS 能够抵抗利用选择消息进行存在性伪造攻击。

证明：在这里采用"模仿敌手观察"的方法[8, 9]实现证明，文[8]指出：若门限签名方案是可模仿的(Simulatable)，而对应的基础签名方案是安全的，则门限签名方案就是安全的(具有不可伪造性)，因此这里我们只需证明 NTS 方案是可模仿的即可。设 A 为 NTS 方案的 PPT 伪造敌手，它勾结了最多 t 个签名用户，并拥有发送方的组公钥 $Y=xP$、消息 m、m 的签名 $\sigma=(U,V)$ 和随机 Oracle H_1，H_2。

对于基于 GJKR 算法的密钥生成算法 TK 的可模仿性，文[6]已证明：对于敌手 A，存在一个模仿器(Simulator，一个 PPT 算法)SIM_1 模拟敌手 A 勾结的恶意用户和诚实用户联合执行 GJKR 算法(实际上是模仿 GJKR 算法)，得到的观察(view)，与实际执行 GJKR 算法得到的观察在敌手 A 看来是计算不可区分的。换一句话说，算法 TK 达到了保密性要求：任何少于 $t+1$ 个诚实的签名用户不能获得关于 x 的任何信息，其中 $t<n/2$。

下面证明 TS 算法的可模仿性。不失一般性，假设敌手 A 勾结的恶意用户为 $\Omega_B=\{P_1,P_2,\cdots,P_t\}$，诚实用户为 $\Omega_G=\{P_{t+1},P_{t+2},\cdots,P_n\}$。现在我们证明能够通过一个模仿器 SIM_2 来模仿敌手 A 执行 TS 算法以生成观察(view)和门限签名。对于恶意用户 Ω_B，其部分签名的生成及验证比较容易，关键是模仿诚实用户 Ω_G 的签名。由于 SIM_2 拥有 m 的签名 $\sigma=(U,V)$，则 SIM_2 可通过以下方法产生模仿敌手 A 执行 TS 算法的观察：

(1)若 $H_1(m,U)$ 和 $H_2(m)$ 未被询问过，则定义 $h=H_1(m,U)$，$Q=H_2(m)$。

(2)计算 $V_i=(r_i+hx_i)Q(1\leqslant i\leqslant t)$。设 $F(x)$ 为一个系数在 G_1^* 上的 t 次插值多项式，并满足 $F(0)=V$ 和 $F(i)=V_i$ $(1\leqslant i\leqslant t)$，计算 $V_i=F(i)(t+1\leqslant i\leqslant n)$ 并广播。

以上说明 TS 算法是可模仿的。换一句话说，对于一个拥有同样公钥 $Y=xP$ 的基础签名 QCXS 方案的伪造敌手 B，若 NTS 方案的敌手 A 能以一个不可忽略的概率伪造一个合法签名 σ，则敌手 B 就能以 σ 作为合法伪造，证毕。

结束语 门限签名实现了由一组签名用户共享对一个消息进行签名的功能，本质上是这组签名用户共享组私钥。由于 Boldyreva 门限签名方案不是概率型的签名体制，从而存在通过对比攻击而导致组私钥暴露的风险。本文基于 GDH 群构造了一个门限签名方案，并给出了安全性证明，该方案通过增加概率签名的特性，克服了 Boldyreva 方案的这一弱点。

参 考 文 献

1 Boneh D, Lynn B, Shacham H. Short Signatures from the Weil Pairing[A]. In: Proceedings of Asiacrypt'01, Lecture Notes in Computer Science 2248[C], Berlin: Springer-Verlag, 2001. 514～532

2 钱海峰，曹珍富，薛庆水. 基于双线性对的新型门限代理签名方案[J]. 中国科学，E 辑，2004，34(6)：711～720

3 Boldyreva A. Efficient Threshold Signature, Multisignature and Blind Signature Schemes Based on the Gap-Diffie-Hellman-Group Signature Scheme[A]. In: Proceedings of PKC 2003, Lecture Notes in Computer Science 2567[C], Berlin: Springer-Verlag, 2003. 31～46

4 Shamir A. How To Share A Secret[J]. Communications of the ACM, 1979, 22(11): 612～613

5 Pedersen T. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing[A]. In: Feigenbum, ed. Advances in Cryptology- Crypto'91, Lecture Notes in Computer Science 576[C], Berlin: Springer-Verlag, 1992. 129～140

6 Gennaro R, Jarecki S, Krawczyk H, et al. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems[A]. In: Proceedings of the Eurocrypt'99, Lecture Notes in Computer Science 1592[C], Berlin: Springer-Verlag, 1999. 295～310

7 Feldman P. A Practical Scheme for Non-Interactive Verifiable Secret Sharing[A]. In: Proceedings of 28th IEEE Symposium on Foundations of Computer Science (FOCS' 87), 1987. 427～437

8 Gennaro R, Jarecki S, Krawczyk H, et al. Robust Threshold DSS Signatures[A]. In: Proceedings of the Eurocrypt'96. Lecture Notes in Computer Science 1070[C], Berlin: Springer-Verlag, 1996. 354～371

9 Micali S, Rogaway P. Secure Computation. In: Feigenbum, ed. Advances in Cryptology-Crypto'91, Lecture Notes in Computer Science 576. Berlin: Springer-Verlag, 1992. 392～404

10 陈伟东，冯登国. 签密方案在分布式协议中的应用[J]. 计算机学报，2005，28(9)：1421～1430

11 Baek J, Zheng Y L. Identity-Based Threshold Signature Scheme from the Bilinear Pairings[A]. In: Proceedings of International Conference on Information Technology: Coding and Computing (ITCC'04), Washington: IEEE Computer Society, 2004. 124～128