一种应用于网络硬盘存储系统的一致性协议

刘青昆,石彦博,梁 莹
(辽宁师范大学 计算机与信息技术学院,辽宁 大连 116081)

摘 要:通常网络硬盘系统会向用户提供高可用性和低延迟的网络服务,并且通过对不同程度上的一致性进行选取,从而寻找可用性、低延迟与一致性这三者的平衡。高一致性意味着系统将会牺牲掉一定的可用性,但低一致性会增加程序设计的难度。为一款网络硬盘系统设计一种一致性协议,使得该网盘可以高效地处理用户对共享目录或文件的移动、删除和重命名等操作,与此同时还能确保共享目录及文件在各个客户端中不会有混乱无序的情况出现。

关键词:网络硬盘存储系统;一致性协议;可用性;可扩展性
中图分类号:TP302.7 **文献标识码**:A **文章编号**:1674-7720(2013)14-0072-03

A consistency protocol applied to online storage system

Liu Qingkun, Shi Yanbo, Liang Ying
(College of Computer and Information Technology, Liaoning Normal University, Dalian 116081, China)

Abstract: Usually, the online systems promise high availability and low latency, and they provide different degrees of consistency to find the trade-off between availability, low latency and consistency. High consistency usually means the systems will sacrifice the availability, but low consistency can bring difficulty to the programmer. In this paper, we present a new consistency protocol for a online storage system, which can let it deal with the openration of move delete and rename on the sharing catalogue and file efficiently. At the same time, this protocol can guarantee the catalogue and file on the client will not end in chaos.

Key words: online storage system; consistency protocol; availability; scalability

在信息技术飞速发展的今天,企业和个人所拥有的数据总量在不断地增加,而且为了能够更好地协同工作,人们对数据共享的需求也日益旺盛。相比将大量数据放置在本地,越来越多的人们开始选择使用网络硬盘系统来保存数据。网络硬盘系统所提供的高可靠性及可用性使得用户所上传的数据不易丢失,而且随时可以获取[1]。

近些年,针对一致性协议的研究与设计逐渐成为分布式系统领域的一个热点问题[2-3]。而新兴的网络硬盘系统与其他的分布式存储系统一样面临着同样的问题,即它们都需要在一致性和可用性之间寻找一个较为合理的平衡点。本文在一款由清华大学所研制的网盘系统——Corsbox 的基础之上,为其设计一种用于客户端之间进行共享操作的一致性协议,使得该网盘系统在此种情况下具有高可用性和低延迟的特点。

1 背景知识

系统中保存的数据被某一客户端更新时,可以根据其余客户端何时能够看到此更新的数据来对一致性进行分类。强一致性是指当系统中的数据更新完成以后,任何后续的客户端访问此数据时都能够返回更新过的值;弱一致性是指系统不能保证当数据更新完成以后,客户端的后续访问能够返回更新过的值;最终一致性是指系统保证如果用户数据在没有新的更新操作发生的情况下,最终所有对此数据的访问都将返回最后更新过的值。

分布式共享存储系统在理想情况下应该同时具有强一致性、系统可用性和网络分区容忍性。但不幸的是,Eric Brewer 的 CAP[4]理论证明了这三者不可能同时实现,并指出任何系统在任一时刻只能满足其中两条性质。在网络分区已经成为一种基本要求的情况下,分布式系统的设计者几乎一边倒地选择了可用性而放弃了强一致性。之所以如此是因为对于大多数应用来说,并不一定需要强一致性;而且选择可用性不仅可以降低系统响应客户端请求时的延迟,还能增加系统的可扩展性。Wyatt Lloyd[5]等人将具有可用性(Availability)、低延迟(Low latency)、

分区可容忍性(Partition-tolerance)和可扩展性(Scalability)这4种特性的系统称之为ALPS系统。现如今有很多系统同时具有ALPS这4种特性,如Dynamo[6]、Voldemort[7]和Memcached[8]等一些具有最终一致性特点的键值对存储系统。Facebook的Cassandra[9]是一种可配置的系统,可以根据需要设置其具有上述4种特性,也可以视具体情况使其具有线性一致性。本文将为一款网络硬盘系统——Corsbox设计一种一致性协议,使得该网盘在移动、删除和重命名共享目录或文件的过程中具有高可用性和低延迟的特点,加之Corsbox网盘在网络分区的情况下本身就有很强的扩展能力,所以使其满足了成为ALPS系统所需要的所有条件。

2 网络硬盘与底层云存储系统

大部分网络硬盘系统通常由客户端、服务器端和底层的云存储端3个部分组成。云存储是云计算技术的发展和延伸,云端将用户数据分布在由大量计算机节点构成的资源池中,由于云存储系统使用了特殊的容错机制,所以可以采购一些价格相对低廉的节点构成云,而且还可以将各种不同类型的存储设备集合起来协同工作。为了降低运营成本,大部分网络硬盘都不会自己搭建底层的云存储系统,而是转而与大型的云存储供应商合作。

Corsbox网络硬盘采用的底层云存储系统是开源的OpenStack Swift[10],它具有极高的数据持久性、数据冗余和高扩展性等特点。OpenStack并不是传统文件系统,它为每一个使用者创建一个账户(Account),用户可以在其账户下建立多个容器(Container),并将数据文件以对象(Object)的形式存放在其中,这一点与Amazon S3[11]很相似。

Corsbox网盘在设计上规定了以目录为最小的共享单位,并且允许目录的所有者与被共享者都有权移动、重命名和删除此目录及其中的文件。由于OpenStack Swift[10]在处理删除、移动和重命名等操作时只能确保数据的最终一致性,也就是说某一用户对数据的修改不会立刻得到更新,所以当多个用户在共享同一数据时,网盘系统不可能满足他们之间的强一致性需求。并且即便Swift底层云存储系统能够确保强一致性,由于共享用户不可能同时在线的原因,也无法做到在任何时刻,所有用户所共享的数据都是一致的。因此需要为网络硬盘系统设计一种一致性协议,来保证共享用户在目录或文件不会再发生任何改变时,他们所看到的视图都是相同的。

3 一致性协议设计

本文在描述为Corsbox网络硬盘系统所设计的一致性协议时,以用户操作共享文件为例进行说明。假设网盘系统中有5名用户A、B、C、D、E,他们共享了同一目录,其中用户A是此共享目录的拥有者,目录的路径为:Corsbox/album,其余用户都是被共享者。在共享目录中存有两个子目录sarah和enya,sarah目录下有一音频文件living.mp3。按照如下顺序对共享目录中的文件进行操作:

(1)B、C、D 3名用户首先上线,并一直保持在线状态,且此时不对共享文件做任何操作;

(2)用户E和用户A登录,并在其后的任一时刻退出系统;

(3)用户B、C在用户E和用户A登录网盘之后分别对共享文件进行操作,B用户将living.mp3改名为journay.mp3,C用户将living.mp3移动至enya目录下;

(4)用户E再次登录,并在其后任意时间下线;

(5)用户D在用户E再次登录后将共享文件living.mp3改名为crazy.mp3;

(6)用户A再次登录网盘系统。

在如上所述的操作系列下,本文所设计的协议会按照时间先后顺序记录下每个用户的登录情况和对共享文件的操作。当用户E在步骤(4)中再次登录时,网盘系统会根据记录向上搜索,直至找到其上一次的登录记录为止。在搜索过程中,系统获取了在用户E的两次登录之间,B、C分别对共享文件进行了操作。当搜索工作结束之后,本协议会对由用户A在云端所保存的这一共享文件做统一处理,即分别向云端发出B、C两者对此文件的操作请求。之后网盘系统会根据本文所设计的协议将B、C的操作信息发送至用户E,E的客户端会根据这一信息,按照B、C的操作顺序依次修改本地的共享文件。在进入到步骤(6)时,用户A再次上线,网盘系统同样会向前搜索至其上一次登录的时间点,此时系统会检测到在这一时间段内B、C、D都对共享文件进行了操作,而且用户B、C的请求已经在云端完成,所以网盘系统现在只需要处理D用户的操作请求即可。在完成了云端同步之后,B、C、D 3者的操作信息会被发送给用户A,由于用户A的客户端之前没有处理过B、C的操作,所以此时会连同D的操作按次序一并处理。需要注意的是,当这5位用户中的任意一个对共享文件做了删除操作时,其余用户在此后对这一文件的操作都将被忽略。具体操作流程如图1所示。

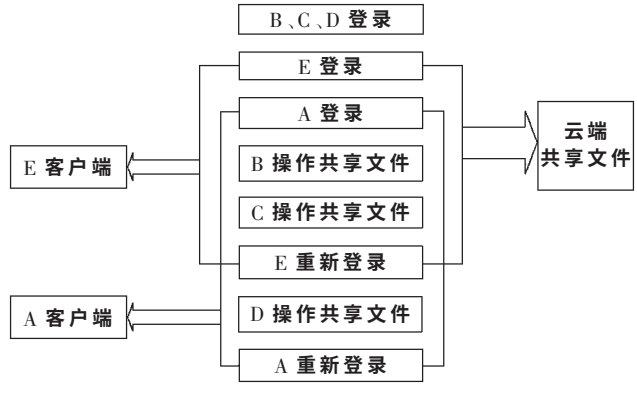

图1 用户操作共享文件的流程

使用本文为网盘系统所设计的上述协议可以带来很多优点：

(1)此协议可以很好地适应 OpenStack Swift 底层云存储系统的最终一致性特点。

(2)用户对共享目录及文件的移动、删除和重命名操作会得到网盘系统的迅速响应，因为网盘系统并没有立刻向底层云系统转发用户的操作请求，而只是将其操作记录下来而已。具体的底层操作会推迟到其后任一用户登录时进行，即网盘系统会在各个用户上线时对他们进行同步操作，这也是现行大多数网盘的通行做法。

(3) 网盘系统推迟了对用户的操作请求，不会对共享此目录或文件的用户和其余被共享者造成任何使用上的影响。

(4)由于协议严格地按照时间先后顺序执行各个用户的请求，所以最终共享用户所看到的目录及文件视图不会有混乱不一致的情况发生。

(5)本协议在处理由各共享用户的操作所引起的冲突时，使用了类似 last-writer-wins[12]的原则，即最后修改共享目录或文件的操作将覆盖掉之前其余所有用户的操作(除删除操作)。它不同于 Coda[13]和 Dynamo[6]等一些系统需要客户端直接人为地介入来解决冲突问题。

4 一致性协议的实现

为了实现所设计的协议，在网盘系统中引入了 MySQL 数据库，每当用户共享一个新的目录给其他用户时，系统都会为其生成一张数据表与之对应，此后所有共享此目录的用户的登录操作和对这一共享目录及其所包含的子目录和文件的删除、移动、重命名操作都会被记录到这张表上。当有用户登录进行数据同步时，网盘系统根据协议只需对该表做降序搜索即可，并将搜索到的各个用户的操作信息保存在特定的数据结构中。此数据结构的具体成员如表 1 所示。

表 1 数据结构表

字段	描述
userName	表明此共享目录或文件的拥有者
type	指明用户对共享目录或文件做了何种操作
path	指明用户所操作的目录或文件的路径
hostName	指明用户所使用的机器名
flag	指明用户所操作的对象是文件还是目录

用户 E 在登录网盘以后，记录在数据结构中的具体信息如表 2 所示，其中 path 项中的“&”符号用于分隔文件的新旧路径。网盘系统在执行第一条操作之后会对表中的第二条记录做预处理，即 B 用户重命名后的新路径替换第二条记录中 path 项的原有路径，替换后的结果为 Corsbox/album/sarah/journay.mp3&Corsbox/album/enya/，这样网盘在其后操作第二条记录时就能够作出正确处理。

表 2 E 登录后数据结构中的具体信息

userName	type	path	hostName	flag
A	Rename	Corsbox/album/sarah/living.mp3&Corsbox/album/sarah/journay.mp3	B_PC	file
A	Move	Corsbox/album/sarah/living.mp3&Corsbox/album/enya/	C_PC	file

当用户 A 再次登录网盘时，数据结构中的信息如表 3 所示。由于用户 B、C 的操作已经在底层云端处理妥当，此时网盘只需在云端执行第三条操作记录即可。但是在这之前，还是需要对第二、三条记录中的 path 项做两次循环替换，替换后的最终结果如表 4 所示。

表 3 A 再次登录后数据结构中的具体信息

userName	type	path	hostName	flag
A	Rename	Corsbox/album/sarah/living.mp3&Corsbox/album/sarah/journay.mp3	B_PC	file
A	Move	Corsbox/album/sarah/living.mp3&Corsbox/album/enya/	C_PC	file
A	Rename	Corsbox/album/sarah/living.mp3&Corsbox/album/sarah/crazy.mp3	D_PC	file

表 4 处理后的信息

userName	type	path	hostName	flag
A	Rename	Corsbox/album/sarah/living.mp3&Corsbox/album/sarah/journay.mp3	B_PC	file
A	Move	Corsbox/album/sarah/journay.mp3&Corsbox/album/enya/	C_PC	file
A	Rename	Corsbox/album/enya/journay.mp3&Corsbox/album/enya/crazy.mp3	D_PC	file

Corsbox 网络硬盘系统在经过了这一系列处理之后，A、B、C、D、E 这 5 位用户在云端所对应的共享文件达到最终一致。共享用户在客户端的一致性操作与在云端的大致相同，不再累述。

本文为一款网络硬盘系统——Corsbox 设计了一种一致性协议，从而使得该系统具备了高可用性和低延迟等特点。这意味着该网络硬盘系统拥有了成为 ALPS 系统所需要的所有条件。当该网络硬盘系统使用此协议时，能够高效、快速地处理用户对共享目录及文件的操作请求，并达到最终一致。

参考文献

[1] Zeng Wenying, Zhao Yuelong, Ou Kairi, et al. Research on cloud storage architecture and key technologies[C]. Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, Seoul: 2009: 1044-1048.

[2] COOPER B F, RAMAKRISHNAN R, SRIVASTAVA U, et al. PNUTS: Yahoo! 's hosted data serving platform[J]. Proceedings of the VLDB Endowment, 2008, 1(2): 1277-1288.

[3] SOVRAN Y, POWER R, AGUILERA M K, et al. Transactional storage for geo-replicated systems[C]. Proceedings of the 23rd ACM Symposium on Operating Systems Principles, Cascais: 2011: 385-400.

[4] BREWER E. Towards robust distributed systems[C]. Proceedings of the 19th annual ACM symposium on Principles of distributed computing, ACM, 2000: 7.

[5] LLOYD W, FREEDMAN M J, KAMINSKY M, et al. Don't settle for eventual: scalable causal consistency for wide-area storage with COPS[C]. Proceedings of the 23rd ACM Symposium on Operating Systems Principles, Cascais: 2011: 401-416.

[6] CANDIA G D, HASTORUN D, JAMPANJ M, et al. Dynamo: Amazon's highly available key-value store[J]. Operating systems review, 2007, 41(6): 205-220.

[7] SUMBALY R, KREPS J, Gao Lei et al. Serving large-scale batch computed data with project Voldemort[C]. Proceedings of the 10th USENIX conference on File and Storage Technologies, 2012: 18.

[8] FITZPATRICK B. Distributed caching with memcached[J]. Linux Journal, 2004(124): 5.

[9] LAKSHMAN A, MALIK P. Cassandra-a decentralized structured storage system[J]. ACM SIGOPS Operating Systems Review, 2009, 44(2): 35-40.

[10] TAHERIMONFARED A, JAATUN M G. As strong as the weakest link: handling compromised components in openStack[C]. Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Athens: 2011: 189-196.

[11] YOON H, GAVRILOVSKA A, SCHWAN K, et al. Interactive use of cloud services: Amazon SQS and S3[C]. Proceedings of the 2012 twelfth IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ottawa: 2012: 523-530.

[12] THOMAS R H, BERANEK B, NEWMAN. A majority consensus approach to concurrency control for multiple copy databases[J]. ACM Transactions on Database Systems, 1979, 4(2): 180-209.

[13] KISTLER J, SATYANARAYANAN M. Disconnected operation in the Coda file system[J]. ACM Transactions on Computer Systems, 1992, 10(1): 3-25.

(收稿日期:2013-03-29)

作者简介:

刘青昆,男,1971年生,副教授,硕士生导师,主要研究方向:嵌入式操作系统,并行计算等。

石彦博,男,1984年生,硕士研究生,主要研究方向:大规模安全存储系统。

梁莹,女,1988年生,硕士研究生,主要研究方向:存储系统。