

引用格式:范中磊,李小亮,惠小红. 对象存储设备中对象的均匀分布方法[J]. 微电子学与计算机, 2019, 36(6): 70-73. [FAN Z L, LI X L, HUI X H. Uniform placement method for objects in object storage device [J]. Microeletronics & Computer, 2019, 36(6): 70-73.]

对象存储设备中对象的均匀分布方法

范中磊, 李小亮, 惠小红

(长安大学 信息工程学院, 陕西 西安 710064)

摘 要: 针对目前对象存储设备上的对象分布策略可能导致的单一目录下对象过多、分布不均匀等问题, 本文提出了一种基于对象 ID 分段的均匀分布方法, 有效地提升了对象存储设备上对象分布的均匀性, 从而达到提升对象访问性能的目的. 该方法在我们自主研发的 CSCloud 对象存储系统中得以应用和验证.

关键词: 对象存储; 对象存储设备; 对象; 均匀分布; 云存储

中图分类号: TP333

文献标识码: A

文章编号: 1000—7180(2019)06—0070—04

Uniform placement method for objects in object storage device

FAN Zhong-lei, LI Xiao-liang, HUI Xiao-hong

(School of Information Engineering, Chang'an University, Xi'an 710064, China)

Abstract: Aiming at the problem that the current object distribution strategy on the object storage device may cause too many objects in a single directory and uneven distribution, this paper proposes a uniform distribution method based on object ID segmentation, which effectively improves the uniformity of object distribution on an object storage device, and then achieve the purpose of improving object access performance. This method is applied and verified in our self-developed CSCloud object storage system.

Key words: object storage; object storage device; object; uniform placement; cloud storage

1 引言

数据的爆炸式增长和云存储应用的快速发展对数据存储系统提出了严峻的挑战. 对象存储系统以其在可伸缩性、易用性、可靠性和低成本等方面的优势而成为应对这些挑战的很有前途的解决方案, 因而得到日益广泛的应用^[1-3].

对象存储系统中主要涉及到两个概念, 数据对象(Object, 以下简称为对象)和数据对象 ID(Object ID, 简称为 OID). 对象包含需要存储的文本、图片、音频、视频等数据, 是对象存储系统的基本存储单元. OID 是对象的唯一标识符, 每个对象都有一个唯一的 OID. 对象通过 OID 在对象存储设备上进行

存取, 对象存储设备承担着对象的访问任务. 因此, 如何将对象存储设备上的海量对象进行有效的分布将直接关系到对象存储设备的访问性能乃至整个对象存储系统的性能.

亚马逊 AWS(Amazon Web Services)的简单存储服务 S3(Simple Storage Services)是目前广泛应用的商业化对象存储系统^[4-5].

相比 AWS 的 S3 对象存储服务, OpenStack 的对象存储服务(SWIFT)采用了一种静态的目录分布方法. SWIFT 这种两层、静态对象分布模型, 在一定程度上缓解了单一目录下数据分布过多的问题. 但是, suffix_path 取三个 16 进制数会使得 suffix_path 层目录容量确定, 当对象数量很多(如海

量对象的云存储应用)时,将导致 suffix_path 层目录下子目录过多,影响对象访问性能乃至系统的整体性能。

针对上述问题,本文提出了一种基于 OID 分段的对象均匀分布方法—UPMO(Uniform Placement Method for Objects)。UPMO 可以根据对象数量动态地调整目录层次和每层中的目录容量,从而满足不同数量级下对象的均匀分布,是对已有对象存储系统中对象分布方法的扩展和补充。

2 UPMO 算法设计

UPMO 是在对象与存储地址之间建立起高效映射关系的方法,换言之,是将对象合理地分配到对象存储设备中,实现对象的均匀分布,从而最大限度地简化寻址过程,达到提升对象访问性能的目的。对象存储设备中对象的存储地址是以目录的形式存在的,因此,在算法设计中主要考虑目录的命名形式、目录层次和每层目录的最大容量三个方面的问题。

为了实现对象的高效存取,UPMO 对目录的命名形式采用伪随机的方法,即目录名是通过给定的算法动态生成的,并且生成的目录名长度是固定的。目录层次和每层目录的最大容量决定了对象分布的均匀性,考虑到每个对象都有一个唯一的 OID,UPMO 就从 OID 入手,通过划分 OID 并结合目录名生成算法,动态地构建目录层次和每层目录的最大容量,一定程度上实现了目录结构的随机化,从而使得当对象数量不断增长时,对象被均匀地分布在动态构建的目录结构中,避免单一目录下对象过多的问题。图 1 给出了 UPMO 的算法流程图,其中: $A = L(P[i])$ 表示通过算法 L 生成目录名 A。

3 UPMO 实现示例

在对象存储系统中,当用户上传一个对象时,系统会首先为其生成全局唯一的 OID,UPMO 就是基于对 OID 的划分来工作的。下面的实现示例假定 OID 是 32 位 16 进制数组成的字符串,目录层次是两层目录加对象层(第三层)构成的。

假设用户之前未上传任何对象,现在开始上传对象 A 和 B,且:A 的 OID 为 a5c12521a79bb79c22debca347dbbe00,B 的 OID 为 a879091c5a880faf6fb5e6087eb1b2dc,对象存储的根目录为 /rootDir,则上传对象 A 的流程如下:

(1)OID 划分为两部分:a5c12521a79bb79c 和

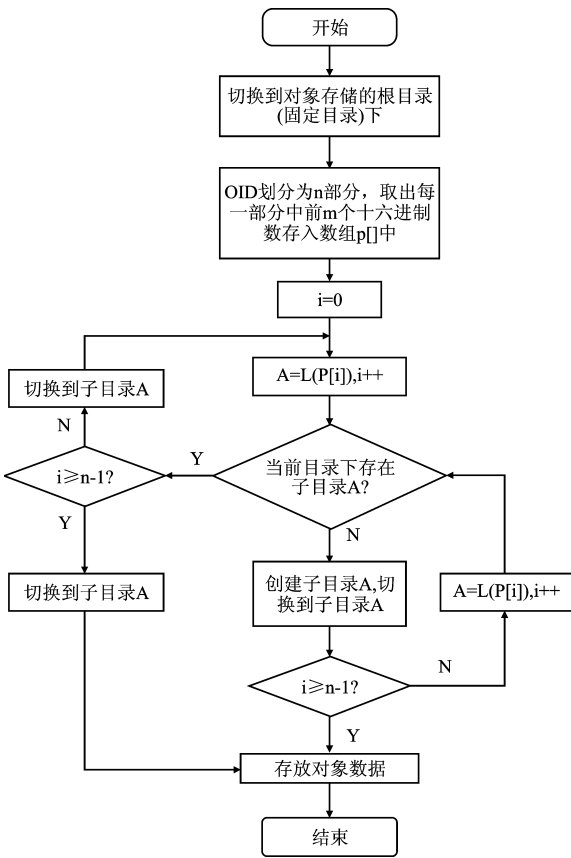


图 1 UPMO 算法流程

- 22debca347dbbe00;
- (2)取每一部分的前一位 16 进制数分别得到 a 和 2,存入数组 P[];
- (3)从 P[]中取出 a,通过 md5(a)生成目录名 0cc175b9c0f1b6a831c399e269772661;
- (4)由于之前没上传文件,所以(3)中生成的目录肯定不存在,创建该目录并切换到该目录下;
- (5)从 P[]中取出 2,通过 md5(2)生成目录名 c81e728d9d4c2f636f067f89cc14862c;
- (6)由于之前没上传文件,所以(5)中生成的目录肯定不存在,创建该目录并切换到该目录下;
- (7)上传对象 A,最终的对象存储路径为:/rootDir/0cc175b9c0f1b6a831c399e269772661/c81e728d9d4c2f636f067f89cc14862c/a5c12521a79bb79c22debca347dbbe00.
- 上传对象 B 和上传对象 A 流程大致相同,只是第一层目录 md5(a)已创建,所以只需创建第二层由 md5(6)生成的目录 1679091c5a880faf6fb5e6087eb1b2dc,最终的对象存储路径为:/rootDir/0cc175b9c0f1b6a831c399e269772661/1679091c5a880faf6fb5e6087eb1b2dc /a879091c5a880faf6fb5e6087eb1b2dc

4 实验结果比较与分析

本文基于 Linux 操作系统,编写程序分别对 S3, SWIFT 和 UPMO 三种对象分布方法进行模拟,并从目录分布和操作性能两方面进行比较和分析.

4.1 目录分布比较与分析

编写脚本程序模拟 S3, SWIFT 和 UPMO 上传对象,实验中观察对象数量分别为[500, 1 000, 2 000, 5 000, 10 000, 20 000, 100 000, 1 000 000]时各层目录分布情况. 为便于比较,取出每层目录的最大目录数进行比较,如表 1 所示.

表 1 S3,SWIFT,UPMO 对象分布对比

对象数量	目录层次	最大目录数		
		S3	SWIFT	UPMO
500	第一层	500	469	16
	第二层	0	2	11
	第三层	0	1	3
1000	第一层	1 000	886	16
	第二层	0	2	16
	第三层	0	1	4
2000	第一层	2 000	1 584	16
	第二层	0	2	16
	第三层	0	1	8
5000	第一层	5 000	2 910	16
	第二层	0	2	16
	第三层	0	1	20
10 000	第一层	10 000	3 730	16
	第二层	0	3	16
	第三层	0	1	40
20000	第一层	20 000	4 096	16
	第二层	0	5	16
	第三层	0	1	79
100 000	第一层	100 000	4 096	16
	第二层	0	26	16
	第三层	0	1	398
1000000	第一层	1 000 000	4 096	256(取 OID 前两位)
	第二层	0	257	16
	第三层	0	1	251

由表 1 可以看出:S3 策略下,第一层目录数随着对象数量增加成正比增长,用户所有对象存储在单一目录下;SWIFT 策略下,刚开始随着对象数量的增长,第一层目录数几乎直线增长,而第二层只有

1~2 个目录. 当对象数量超过 10000 个时,第一层目录数增长逐渐缓慢,直至该层目录数达到饱和. 之后对象数量的增加将只增加第二层的目录数;UPMO 策略下,当对象数量到达 500 时,第一层目录数稳定到 16(实际上在 500 之前就已经稳定到 16),第二层目录数随着对象数量增加而增长,最后稳定到 16. 之后第一、第二层目录数不变,第三层对象数量随着整体对象数量的增长而增长.

需要说明的是,表 1 中当对象数量达到 10^6 时(最后一行),UPMO 取 OID 的前两位数通过目录名生成算法构成第一层目录,避免了第三层目录下对象数量过多的情况. 可以预测. 当系统中用户的对象数量为 10^9 级别时,S3 单一目录下子目录数量也为 10^9 级别,SWIFT 单一目录下目录数量可达 10^6 级别,而 UPMO 通过对 OID 分段和选择不同的位数构成目录名的方法动态配置目录层次和目录容量,将单一目录下目录数量(或对象数量)控制在期望的范围内(如 10^3 级别),实现对象的均匀化存储.

4.2 操作性能比较与分析

分别以 S3, SWIFT 和 UPMO 策略上传对象,当对象数量分别为[1 000, 10 000, 100 000, 1 000 000]时(如表 1 所示),模拟 List, Read, Write, Add 和 Delete 五种常见对象操作过程,比较 S3, SWIFT 和 UPMO 完成上述各项操作所花费的时间. 实验中采用多组不同数量级的对象,避免由于选取一组数据或者由于数据量较少带来的实验结果的偏差. 实验结果如表 2 所示,分析如下:

(1)针对 List 操作,UPMO 响应时间最短. 当对象数量达到 10^6 时,S3 和 SWIFT 策略下 List 操作完成所用时间分别长达 22 分钟和 15 分钟以上(这对于用户来说是很难接受的),而 UPMO 完成时间只有 34 秒左右. 可以预测,当对象数量继续增大时,S3 和 SWIFT 策略下 List 操作完成所用时间会更长,而 UPMO 可以将其控制在一个可接受的范围;

(2)针对 Read 和 Write 操作,UPMO 响应时间最短. 在实验过程中我们发现,当对象数量不断增大时,S3 和 SWIFT 策略下读写操作完成时间随之增大,而 UPMO 通过调整目录结构可以使得当对象数量不断增加时,读写响应时间几乎不随之变化;

(3)针对 Add 和 Delete 操作,UPMO 响应时间最短. 可以预测,当对象数量继续增长时,S3 和 SWIFT 策略下操作完成时间随之增大,而 UPMO 可以通过调整目录层次和每层目录容量,维持响应

时间的相对稳定.

5 结束语

UPMO 已经在我们自主研发的 CSCloud 对象存储系统中得以实现和应用,用户可以根据对象的数量,通过配置文件动态地调整目录层次和每层目录的最大容量.由实验结果可以看出,UPMO 实现了对象存储设备上对象的均匀化分布,解决了单一目录下目录数量或对象数量过多,从而降低系统访问性能的问题.

参考文献:

[1] EMC Corporation. Information Storage and Management [M](V3). USA: EMC Education Services, 2015.

[2] 范中磊. 面向云存储的归档方法[J]. 微电子学与计算机, 2011, 28(7): 136-138.

[3] 范中磊. 非对称集群存储系统的块虚拟化方法[J]. 微电子学与计算机, 2011, 28(6): 5-7.

[4] Amazon. Amazon S3 [EB/OL]. [2018-08-01]. <https://aws.amazon.com/cn/documentation/s3/>.

(上接第 69 页)

[6] YANG G, TAN CH, HUANG Q, et al. Probabilistic Public Key Encryption with Equality Test[C]// International Conference on Topics in Cryptology. San Francisco: Springer-Verlag, 2010:119-131.

[7] ZHU H, WANG L, AHMAD H, et al. Key-Policy Attribute-Based Encryption with Equality Test in Cloud Computing [J]. IEEE Access, 2017 (5): 20428-20439.

[8] LIAO Y, CHEN H, LI F, et al. Insecurity of A Key-policy Attribute Based Encryption Scheme with Equality Test[J]. IEEE Access, 2018(6): 10189-10196.

[9] WANG Q, PENG L, XIONG H, et al. Ciphertext-Policy Attribute-based Encryption with Delegated Equality Test in Cloud Computing[J]. IEEE Access, 2017(6): 760-771.

[10] ROUSELAKIS Y, WATERS B. Practical construc-

[5] 刘鹏. 云计算[M](3 版). 北京: 电子工业出版社, 2015.

[6] 阿里云. 阿里云存储服务 OSS 基本概念 [EB/OL]. [2018-03-01]. https://help.aliyun.com/knowledge_detail/5974194.html.

[7] 腾讯. 对象存储 [EB/OL]. [2018-04-01]. <https://cloud.tencent.com/document/product/436>.

[8] 新浪云存储. 开发指南 [EB/OL]. [2018-02-01]. <http://open.sinastorage.com/doc/scs/guide>.

[9] 余兴超. 深入云存储系统 SWIFT 存储节点 [EB/OL]. [2012-07-04]. <http://www.cnblogs.com/yuxc/archive/2012/07/04/2575536.html>.

作者简介:

范中磊 男,(1970-),博士,教授.研究方向为分布式计算与存储. E-mail:zlfan@chd.edu.cn.

李小亮 男,(1995-),硕士研究生.研究方向为对象存储系统.

惠小红 女,(1991-),硕士研究生.研究方向为存储监控技术.

tions and new proof methods for large universe attribute-based encryption[C]// ACM SIGSAC Conference on Computer & Communications Security. Berlin, Germany: ACM, 2013: 463-474.

[11] CUI H, DENG R H, WU G, et al. An Efficient and Expressive Ciphertext-Policy Attribute-Based Encryption Scheme with Partially Hidden Access Structures [C]// International Conference on Provable Security. Nanjing, China: Springer-Verlag, 2016: 19-38.

作者简介:

姚莉沙 女,(1993-),硕士研究生.研究方向为密码理论与网络安全. E-mail:yaolishahq@163.com.

王尚平 男,(1963-),教授,博士生导师.研究方向为数字签名与网络身份认证、密码理论与网络安全等.