文章编号:1004-1478(2011)01-0087-04

基于在线存储的对象放置算法的设计与实现

甘勇[1], 韩姗姗[1], 陈国栋[2], 何振[1]

(1.郑州轻工业学院 计算机与通信工程学院, 河南 郑州 450002;
2.河南有线电视网络集团有限公司 郑州分公司, 河南 郑州 450001)

摘要:提出了一种对象放置算法C-SIEVE,它使用离散的数据算法,可以有效地定位数据对象,还解决了对象副本的放置问题.实验表明,该算法能根据存储设备的负载高效且均衡地完成对象映射,对于失效节点的对象恢复也具有很好的性能,是行之有效的.

关键词:云存储;在线存储;对象放置
中图分类号:TP311 **文献标志码**:A

Design and implementation of object placement algorithm based on online storage

GAN Yong[1], HAN Shan-shan[1], CHEN Guo-dong[2], HE Zhen[1]

(1. *College of Comp. and Com. Eng., Zhengzhou Univ. of Light Ind., Zhengzhou* 450002, *China*;
2. *Zhengzhou Branch, He'nan Cable TV Network Group Limited Co., Zhengzhou* 450001, *China*)

Abstract: A new object placement algorithm C-SIEVE was put forward. It can not only locate data object effectively, but also solves problems of object copies placement by discrete data algorithm. Experiments indicated that this algorithm was effective. It can accomplish efficient and balanced object mapping according to the load of storage devices, but also have a good performance for object recovery once there are invalid storage nodes.

Key words: cloud storage; online storage; object placement

0 引言

随着云计算[1]的出现,一种面向Internet的存储服务——云存储[2]应运而生,它的核心是将应用软件与存储设备相结合,通过应用软件实现存储设备向存储服务的转变.与此同时,云存储的出现为在线存储[3]提供了广阔的发展空间,通过为用户提供面向Internet的存储服务,完美地解决了用户数据的存储和保护问题.

目前已存在的各种网络存储系统结构,不论是直连方式存储(direct attached storage, DAS),网络附属存储[4](network attached storage, NAS),还是存储域网络[5](storage area network, SAN),它们都无法满足云存储中以服务为核心的应用,而基于对象的存储[6-7](object-based storage, OBS)则为云存储和在线存储的发展提供了契机.OBS以对象为接口,既有"块"接口的快速,又有"文件"接口的共享,其核心思想是将存储系统中的数据交互分为数据通路和

收稿日期:2010-08-25
基金项目:河南省科技攻关计划项目(102102310030)
作者简介:甘勇(1965—),男,湖南省株洲市人,郑州轻工业学院教授,主要研究方向为计算机网络.

控制通路2部分,它是一种带外传输方式,由客户端(Client)、元数据服务器(metadata server,MDS)和对象存储节点(object storage node,OSN)组成.

目前国内外对于云存储和在线存储的研究如火如荼,但很多都停留在概念层次,由于云存储和在线存储的目标存储系统由成千上万个OSN组成,如果用一个集中的目录来实现对象到OSN的映射,将会造成严重的性能瓶颈.鉴于此,本文将基于SIEVE[8]算法,提出一种离散的对象放置算法C-SIEVE,以期在保证SIEVE算法原有的公平性、高效性的基础上,解决对象及其副本的放置和节点失效时对象的恢复问题.

1 相关研究

对象到OSN的映射即对象的放置方法有静态和动态2种:静态放置方法有轮循法等,动态放置方法有剩余空间权重法[9]、写时创建法[9]和SIEVE法等.

1.1 静态对象放置算法

静态对象放置算法是指对象最终的存储位置按照预先确定的规则或算法进行映射,一旦映射规则或算法确定,地址映射的结果也就确定,不可变更.这种固定的策略减少了系统运行的开销,但它对于不断变化的I/O流是不合适的.例如轮循法(round-robin)虽然可以自动避免失效的OSN来放置对象,但没有考虑各个OSN的动态变化.如果某个OSN存储空间已满,而新创建的对象仍然选择该OSN,就会因为存储空间不足而报错,新创建的对象也不能使用该OSN的网络带宽、磁盘带宽、CPU和内存资源,更不能以该OSN为目标进行负载迁移.

1.2 动态对象放置算法

在对象存储系统中,随着文件和对象的创建、删除和修改,各个OSN的存储对象数目以及剩余可用容量以不可预知的方式动态地变化着,此时必须采用动态的对象放置算法才能更好地利用各个OSN的负载,合理地分布对象.

1.2.1 剩余空间权重法 在放置对象之前,各个OSN将其配额空间、可用存储空间比例等信息发送给MDS,MDS根据这些信息智能地将存储对象放置在合适的OSN上.该方法虽然较静态的对象放置方法有所改进,但它需要MDS的配合才能使用,无形中加大了MDS的负载.

1.2.2 写时创建法 该方法采用延迟的思想来分配存储对象,直到需要对该存储对象进行写入操作时,才分配该存储对象.虽然在第1次写入操作时有些延迟,但这种策略可以根据各个OSN存储空间以及负载情况更加合理地选择分配目标,通过该策略可以大大减少系统存储对象的数量,而且简化文件删除操作.但这种策略只适用于创建时已经指定了文件的大小且该文件为大文件的情形,对于大多数小文件的对象分配来说代价过高.

1.2.3 SIEVE法 以上几种算法都是相对集中的数据算法,对于云存储的分布式集群环境来说它们是致命的性能瓶颈.A. Brinkmann等[8]提出了一种数据放置策略SIEVE,它是一个离散的数据算法,通过使用单位区间的分割,随机地将数据分布到多个磁盘上.该算法思想如下:把数据对象看作一个球,把存储设备看作一个容器,通过把球放入容器而实现对象的放置.SIEVE算法的步骤如下:1)将系统中n个容器的权重用$(d_1,\cdots,d_n)$来表示,并把这些权重值规格化到[0,1]区间,且$\sum_{i=1}^{n} d_i = 1$;2)从n个容器中选择一个具有最大权重值的作为"fall back"容器;3)取$n' = 2^{\lceil \log n \rceil + 1}$,再将[0,1]区间重新分割成$n'$个大小为$1/n'$的区间,然后根据每个容器的权重值分配区间;4)当一个球被放置时,使用一系列随机散列函数$(h_1,\cdots,h_L)$计算出一个落在[0,1]之间的值;5)如果该值在某个容器对应的区间内,则这个球分配给该容器,否则转步骤6);6)该球被分配给"fall back"容器;7)结束.

算法中的权重指存储设备的容量、带宽、计算能力等或这些参数的混合,在此用权重值来统一表示.在区间分割时,有下列几个规则:1)一旦某个区间x被分配给某个容器i,则该容器可以选择完全占有该区间对应的全部间隔,也可以占有该区间的一部分间隔(但必须从区间开始处开始);2)1个容器可以拥有多个区间,但部分占有的至多只能有1个;3)1个区间只能被1个容器拥有.

对于非"fall back"容器,它所分配到的所有间隔之和为$d_i/(1-1/2^L)$,对于"fall back"容器,它所分配到的所有间隔之和为$(d_i - 1/2^L)/(1-1/2^L)$,其中L为算法中所使用的散列函数个数.

SIEVE算法在理论上可以使用$O(n)$个字(不考虑散列函数)在希望的时间$O(1)$内确定一个球

的位置.

SIEVE 算法的缺点是一旦存储节点 OSN 失效,它不具有错误容忍的能力,因为它没有提供任何数据冗余支持.

2　C-SIEVE 算法

2.1　算法描述

C-SIEVE 目前只针对复制级别为 2 的情况,即每个对象只有 1 个副本,因为在实际中,复制级别为 2 已经完全可以满足需求了. 在该算法中将副本也看作对象来放置,且对象和其副本的关键字相同,这一过程由对象副本生成算法完成. C-SIEVE 算法的步骤如下:

1)将系统中 n 个容器的权重值用$(d_1,\cdots,d_n)$来表示,并把这些权重值规格化到[0,1] 区间,且 $\sum_{i=1}^{n} d_i = 1$; 2)将已规格好的 OSN 按照权重大小排序; 3)将已排好序的 OSN 分为 2 个组,要求每个组的权重和均衡; 4)放置对象时,先查找左组中是否已经存在该对象,若不存在,则直接使用 SIEVE 算法放置该对象;若存在,则转步骤 5); 5)使用 SIEVE 算法将该对象放置于右组; 6)结束.

在对象的查找中,需要用到一个对象查找表 OLT 来记录对象的放置位置,由于本算法的需要,只列出该表中核心的表项,如表 1 所示.

表 1　一个 OLT 表的实例

对象关键字	OSNID	所在组号
a	0	0
b	1	1
c	3	0
a	2	1

如果某个 OSN 失效,C-SIEVE 算法中失效节点的恢复流程如下: 1)通过 OLT 查找该失效 OSN 上存放的所有对象并删除该 OSN; 2)使用对象副本生成算法的逆向算法得到失效 OSN 上所存放对象的副本信息; 3)通过 OLT 查找失效 OSN 上对象的副本所在的组号和 OSNID; 4)将副本按照 SIEVE 算法直接恢复至新 OSN 上; 5)结束.

2.2　实验

为了验证 C-SIEVE 算法的特性,本文在一个模拟的存储系统中对它进行实现,并测量它在正常运行和节点失效时对象的映射情况.

2.2.1　对象的映射　模拟一个存储系统,将系统中的 OSN 按权重分为 3 个等级(假设权重值分别为 w,$2w$,$3w$)且每个等级中有 20 个 OSN,初始有 600 000 个复制级别为 2 的对象,然后每次增加 600 000 个对象. 实验结果如图 1 所示.

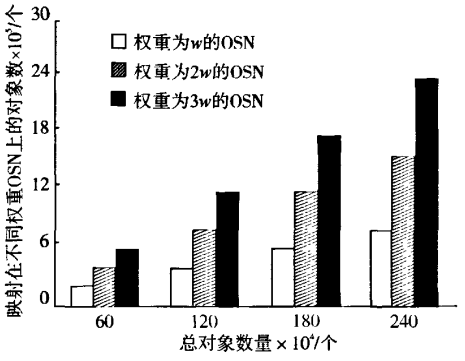

图 1　对象数不同时在不同权重 OSN 上的映射

上面的实验是通过改变对象的数量来验证其映射情况,还可以通过改变 OSN 的数量来验证算法的特性. 系统中初始有 20 个 OSN,权重值记作 w,然后每次加入 20 个 OSN,且新加入的节点权重是上次加入的 2 倍;有 1 000 000 个复制级别为 2 的对象. 实验结果如图 2 所示.

从上述 2 个实验结果可以看出,无论改变对象数量还是 OSN 的权重,C-SIEVE 算法都能确保系统中的所有对象映射在合适的 OSN 上,并且映射在每个 OSN 上的对象数与其权重成正比.

2.2.2　对象副本的映射　一旦某个 OSN 失效,具有该节点上所存放对象副本的 OSN 必须立即承担该失败节点的工作. 一般情况下,在某个指定存储节点上的对象副本会被简单地分布在临近的节点上,与失效节点临近的节点在失败期间为了接替失效节点的工作而经历严重的负载高峰,从而导致性能的下降.

C-SIEVE 算法根据各个 OSN 的权重将对象及其副本分布在整个系统的所有存储节点上,一旦某个 OSN 失效,它的工作负载也会分布在系统中其他所有存活的 OSN 上.

为了验证 C-SIEVE 算法将某个指定 OSN 上的对象和副本分布在整个系统中,现模拟一个存储系统,初始有 10 个 OSN,权重值为 w,放置复制级别为 2 的 100 000 个对象,然后每次加入 15 个(权重值为 w)和 10 个 OSN(权重值为 $2w$). 这样做的目的是验证算法中公平的对象分布并不依赖于每次加入系统中的 OSN 个数,实验结果如图 3 所示.

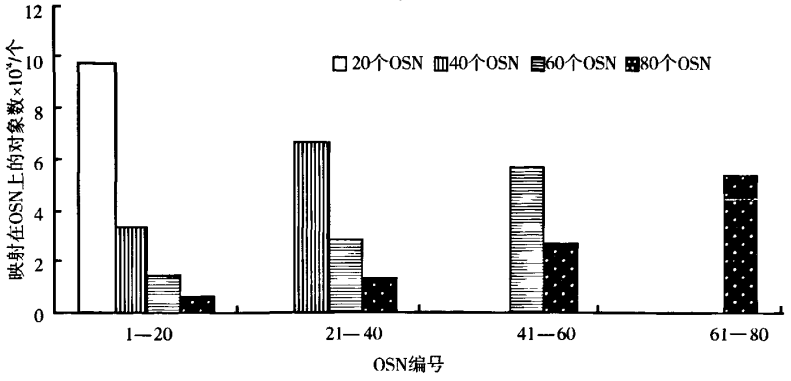

图2 不同权重不同数量OSN上对象的映射

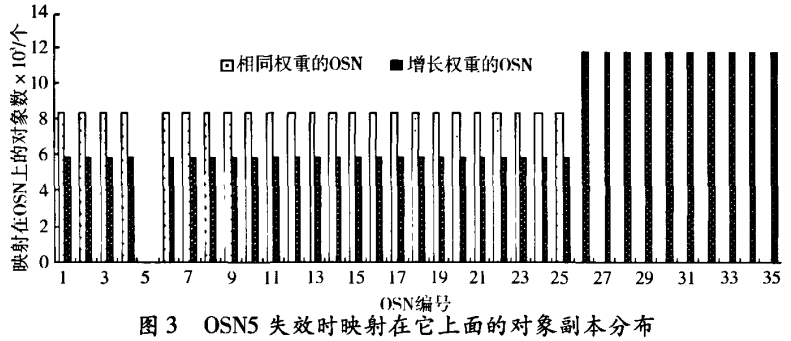

图3 OSN5失效时映射在它上面的对象副本分布

由图3可以看出,一旦系统中某个存储节点失效,存储在它上面的对象及其副本能够快速地被放置到系统中其他的存储节点上,并且该过程仍然按照各个OSN的权重来合理承担对象及其副本的放置任务.

3 结语

本文在分析现有的对象放置算法优缺点的基础上,结合云存储自身的特点提出了一种离散的对象放置算法C-SIEVE.这种方法不仅可以根据OSN的负载情况均衡地选择存储设备,而且有效地解决了对象副本的放置问题,同时还考虑了失效节点的对象恢复.结果表明该算法是高效可行的.

参考文献:

[1] 殷康.云计算概念、模型和关键技术[J].中兴通讯技术,2010(4):24.

[2] 冯华中.你了解云存储吗[J].电脑商报,2009(7):35.

[3] 吴勇毅.SaaS厂商:拿什么为中小企业服务[J].软件工程师,2010(1):45.

[4] Garth A Gibson, Rodney Van Meter. Network attached storage architecture[J]. Com of the ACM, 2000, 43(11):37.

[5] Tom Clark. Designing Storage Area Networks: A Practical Reference for Implementing Fiber Channel and IP SANs [M]. [s. l.]: Addison-Wesley Professional, 2003: 12-34.

[6] Mesnier M, Ganger G R, Riedel E. Object-based storage [J]. Com Magazine IEEE, 2003, 41(8):84.

[7] Mesnier M, Ganger G R, Riedel E. Object-based storage: pushing more functionality into storage [J]. Potentials IEEE, 2005, 24(2):31.

[8] Brinkmann A, Salzwedel K, Scheideler C. Compact adaptive placement schemes for non-uniform capacities[C]// Proc of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA), Winnipeg: ACM Press, 2002: 53-62.

[9] 钱迎进.基于对象存储的高可用技术的研究与实现[D].长沙:国防科学技术大学,2005:47-49.