Computer Science and Application 计算机科学与应用, 2014, 4, 333-343
Published Online December 2014 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2014.412045

Hans汉斯

The Study of Cloud Storage System Based on Object-Based Storage

Shouxin Cao, Liutao Zhao, Yi Jin

Beijing Computing Center, Beijing
Email: caosx@bcc.ac.cn, zhaolt@bcc.ac.cn, jinyi@bcc.ac.cn

Received: Oct. 28th, 2014; revised: Nov. 29th, 2014; accepted: Dec. 7th, 2014

Abstract

This paper mainly introduces the key technology of cloud storage system which is based on an open source technology. The system is based on object-based storage technology, and uses consistent hashing algorithm, eventual consistency and other technologies. In addition, it is compatible with Amazon S3 through its API, and it is usually highly available, simple operation, easy expansion and high fault tolerance.

Keywords

Cloud Storage, Object-Based Storage, Consistent Hashing, Amazon S3

基于对象存储的云存储系统研究

曹守欣，赵琉涛，金　翊

北京市计算中心，北京
Email: caosx@bcc.ac.cn, zhaolt@bcc.ac.cn, jinyi@bcc.ac.cn

收稿日期：2014年10月28日；修回日期：2014年11月29日；录用日期：2014年12月7日

摘　要

本文主要介绍基于开源技术的云存储系统关键技术，系统基于对象存储技术，采用一致性哈希算法、最

终一致性等技术，并提供兼容Amazon S3的API，其具有高可用、操作简单、易扩展，容错性高等特点。

关键词

云存储，对象存储，一致性哈希，Amazon S3

1. 引言

在将数据存储到物理设备上时，前端数据库首先需要对数据进行持久化操作。在过去几年，关系型数据库一直是数据持久化的唯一选择，数据工作者考虑的也只是在这些传统数据库中做筛选，比如 SQL Server、Oracle 或者是 MySQL。甚至是做一些默认的选择，比如使用.NET 的一般会选择 SQL Server；使用 Java 的可能会偏向 Oracle，Ruby 是 MySQL，Python 则是 PostgreSQL 或 MySQL 等等。原因很简单：过去很长一段时间内，关系数据库的健壮性已经在多数应用程序中得到证实。我们可以使用这些传统数据库良好的控制并发操作、事务等等。

然而，随着信息系统的复杂以及用户数量的剧增，人们每天在各种社交网站上上传海量的视频、照片、音乐，每天发送数千亿封电子邮件，数据规模日益扩大。据 IDC 统计，到 2020 年全球数据将增加到 35 ZB，其中 80%是非结构化数据。随着数据规模的不断扩大与数据结构的日益复杂，传统关系型数据库技术在某些方面的性能表现出弱势：

1) 存储结构与数据库结构不匹配现象

我们使用 Python、Ruby、Java、.Net 等语言编写应用程序，这些语言有一个共同的特性——面向对象。但是我们使用 MySQL、PostgreSQL、Oracle 以及 SQL Server，这些数据库同样有一个共同的特性——关系型数据库。存储结构是面向对象的，但是数据库却是关系的，所以这种不匹配现象导致，在每次存储或者查询数据时，我们都需要做转换。类似 Hibernate、Entity Framework 这样的 ORM 框架可以简化这个过程，但是在对查询有高性能需求时，这些 ORM 框架就捉襟见肘了。

2) 应用程序规模的变大

网络应用程序的规模日渐变大，我们需要储存更多的数据、服务更多的用户、需要更多的计算能力。为了应对这种情形，我们需要不停的扩展。扩展分为两类：一种是纵向扩展，即购买更好的机器，更多的磁盘、更多的内存等等；另一种是横向扩展，即购买更多的机器组成集群。在巨大的规模下，纵向扩展发挥的作用并不是很大。首先单机器性能提升需要巨额的开销并且有着性能的上限，在 Google 和 Facebook 这种规模下，永远不可能使用一台机器支撑所有的负载，必须搭建机器集群。而关系数据库集群往往使用共享存储，这并不是我们想要的类型。鉴于这种情况，我们需要新的数据库技术。于是就有了以 Google、Facebook、Amazon 这些试图处理更多传输所引领的 NoSQL 纪元。而对象存储技术便是 NoSQL 纪元的产物。

2006 年 Amazon 发布 AWS，简单存储服务(Simple Storage Service，简称 S3) [1]及其使用的 REST、SOAP 访问接口已成为对象存储的事实标准，Amazon S3 还成功为对象存储注入云服务的基因，使其成为一个基于对象存储的云存储[2]系统。

亚马逊，作为全球最大的云提供商，也是对象存储的开创者，为各种规模的企业和政府提供公有云和私有云服务。这样成熟的品牌所使用的对象存储全部为 S3。据统计，如图 1 所示，2013 年，AWS 的市场占有率超过 60%，其 38 亿美金的收入中，包含了包括服务器、存储、软件、负载均衡等产品。S3 的增长最为疯狂。如图 2 所示，从 2010 年第四季度开始，客户存储的“对象”数目呈指数增长，到

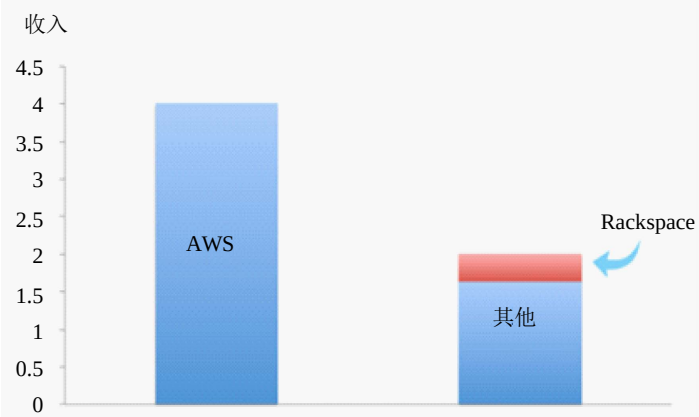

Figure 1. Amazon IaaS income in 2013 (A US$ 1 billion for the unit)
图 1. 2013 年亚马逊 IaaS 云收入(以十亿美金记)

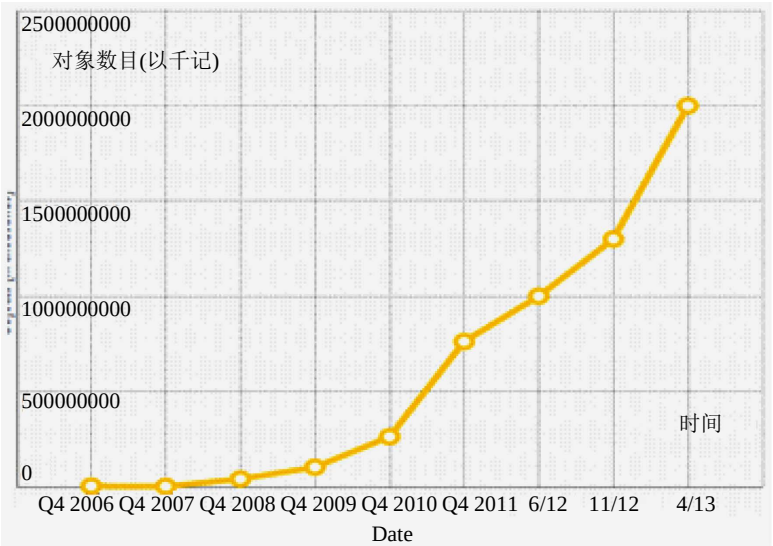

Figure 2. The growth of Amazon S3
图 2. 亚马逊 S3 存储对象增长情况

2013 年上半年，已经有 2 万亿个“对象”。事实证明，对象存储，势必会成为未来主流数据库存储技术。

然而，Amazon 公司不会开放其存储技术。为打破 Amazon 对对象存储技术的封锁，同时满足国内众多企业用户对对象存储和云存储的需求，我们基于一种开源的技术，搭建了基于对象存储技术的云存储系统，为用户提供公有云或私有云服务。该系统可以完全兼容 Amazon S3，使企业可以享受类似 Amazon S3 的存储能力，帮助国内云供应商与亚马逊的存储产品进行竞争。

本文旨在介绍该存储系统的关键技术，让更多的人可以了解对象存储及云存储相关技术。

2. 云存储与对象存储简介

2.1. 云存储

“云计算”是指由一堆廉价的服务器所组成的网络，在远程位置(云)为用户提供所需要的计算机服务，它将一个或多个数据中心的物理计算资源虚拟化之后，以资源租用的方式向用户提供服务。

云存储是云计算的核心组成部分，它通过将网络中大量各种不同类型的磁盘阵列、存储设备通过应

用软件集合起来协同工作，并整合虚拟化、数据保护、数据管理等技术共同构建云存储系统对外提供服务。目前，云存储分为公共云存储、私有云存储和混合云存储[3]。

2.2. 对象存储

传统的存储技术包括 DAS、SAN 和 NAS，它们是基于块存储或文件存储。对象存储是为了满足用户大数据量及非结构化数据的存储而出现的[4]。

DAS 和 SAN 属于块存储，NAS 属于文件存储，

1) 直连式存储 DAS (Direct Attach Storage)：是一种将存储介质直接安装在服务器上或者安装在服务器外的存储方式，每一台主机服务器有独立的储存设备，每台主机服务器的储存设备无法互通。通常用在单一网络环境下且数据交换量不大，性能要求不高的环境下，可以说是一种应用较为早的技术实现。

2) 存储区域网络 SAN (Storage Area Network)：采用光纤通道技术、交换机连接存储阵列和服务器主机，建立专用于数据存储的区域网络，一般而言，SAN 应用在对网络速度、数据的可靠性和安全性要求较高的应用环境中，它具有高带宽、低延迟的优势，目前广泛应用在如电信、银行等关键应用中。

3) 网络附加存储 NAS (Network Attached Storage)：将存储设备连接到现有的网络上，提供数据和文件服务，NAS 是一个专有文件服务器或一个智能文件访问设备，它有独立的文件操作和管理系统，可以提供文件级的数据访问。

而对象存储(Object-Based Storage, OBS)综合了 NAS 和 SAN 的优点，同时具有 SAN 的高速直接访问和 NAS 的数据共享等优势，提供了具有高性能、高可靠性、跨平台以及安全的数据共享的存储体系结构

如图 3 所示，DAS、SAN 和 NAS 以固定大小的数据块作为存储的基本单元，而对象存储以对象为基本单位，采用扁平化存储结构管理所有数据，特别适合大数据量和非结构化数据的存储。对象存储系统包含两种数据描述：容器(Bucket)、对象(Object)，容器和对象都有一个全局唯一的 ID，只需要根据 ID 就可以访问容器/对象及相关的数据(Data)、元数据(metadata)和对象属性(Attribute)。

3. 系统关键技术介绍

在该云存储存储系统中，主要有两大组件互相配合工作：

1) 分布式数据库系统——作为后端存储的数据库系统

2) 云存储 Portal——建立在分布式数据库系统之上，提供了存储 API，可完全兼容 Amazon S3，并支持大对象和对象的分块上传。

此外，还有一个中间支持层来提供一个用来管理和保证用户账号和 Bucket Names 唯一性的应用程序。

3.1. 分布式数据库系统

该存储系统后端采用一个类 Dynamo [5]的，NoSQL 型分布式数据库系统。相较于传统关系型数据库，NoSQL 数据库系统具有关系型数据库所不具备的高可用性、高扩展性、简单的数据模型等特性，适合大量数据与非结构化数据的写入处理。

NoSQL 数据库具有多种类型，如键值存储(key/value store)、文档存储(Document store)、列存储、图存储(Graph store)等，在该系统中采用 key/value 数据模型进行对象存储。如图 4 所示，数据库将数据组织成 Buckets、Keys、Object 三个层次，Object 由一对 key 和 value 组成。Bucket 相当于数据库的一个命名空间，用于存放 Object。Object 有唯一的 key，Bucket 允许相同的 key 存在于多个桶中，Buckets 和 Keys 是数据库中组织数据的唯一方式，这些数据通过 bucket + key 来存储和引用。

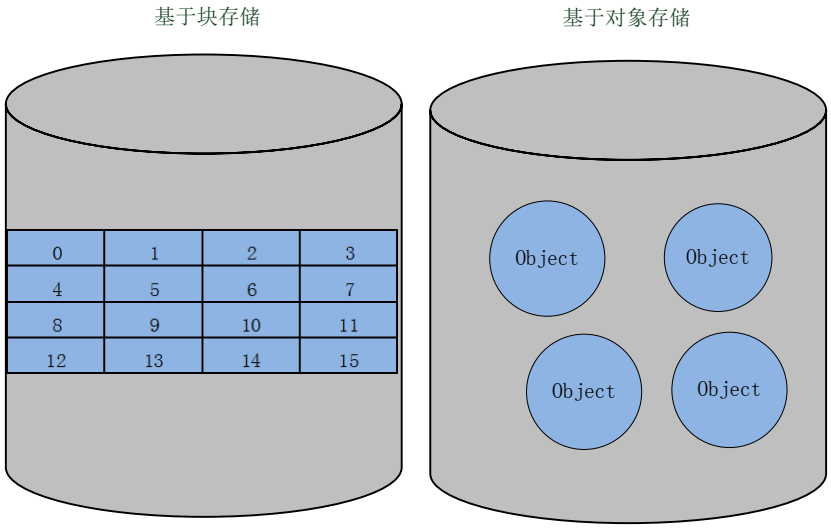

Figure 3. Block storage and object storage
图 3. 块存储与对象存储

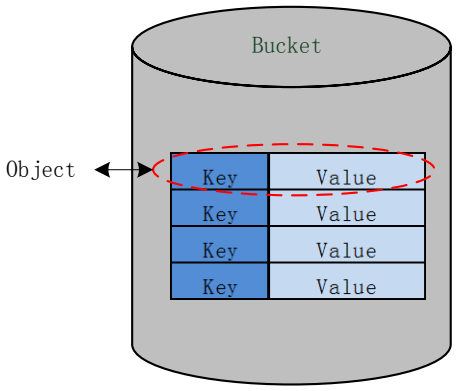

Figure 4. Database storage structure
图 4. 数据库存储结构

在数据库中，Object 最后在磁盘上存储都是以二进制字节形式，因此数据库对 Object 类型没有限制，可以存储任意类型的数据。

3.1.1. 负载均衡与自动扩展

在该分布式数据库集群中，物理服务器在集群中被称为节点(node)，在节点上运行着一定数目的虚拟节点(vnode)。

传统的多服务器系统中，为了实现数据的分布式，通常利用一种叫做“sharding”的技术，将一个数据库切分成多个部分放到不同的数据库(server)上，从而缓解单一数据库的性能问题。这种方式存在大量弊端，首先 sharding 会增加数据库维护的开销，其次，在数据大量增长的情况下，需要不断重新手工对数据进行划分，并且 sharding 容易导致热点的产生。而该数据库通过一致性哈希[6]算法将数据自动均匀的分配到集群的各个 node 中，并且在增加/减少 node 时，数据库会自动重新平衡系统中的数据的分布，避免了热点产生，并且只会影响新增 node 相邻 node 的数据分布[7]。

图 5 所示，数据库集群将一个物理 node 虚拟成多个 vnode，并形成一个环状均匀分布。任何一个集群都是一个 160 位的整型空间。数据库将该整型空间划分成大小相等的分区(partition，默认划分成 64 个)，

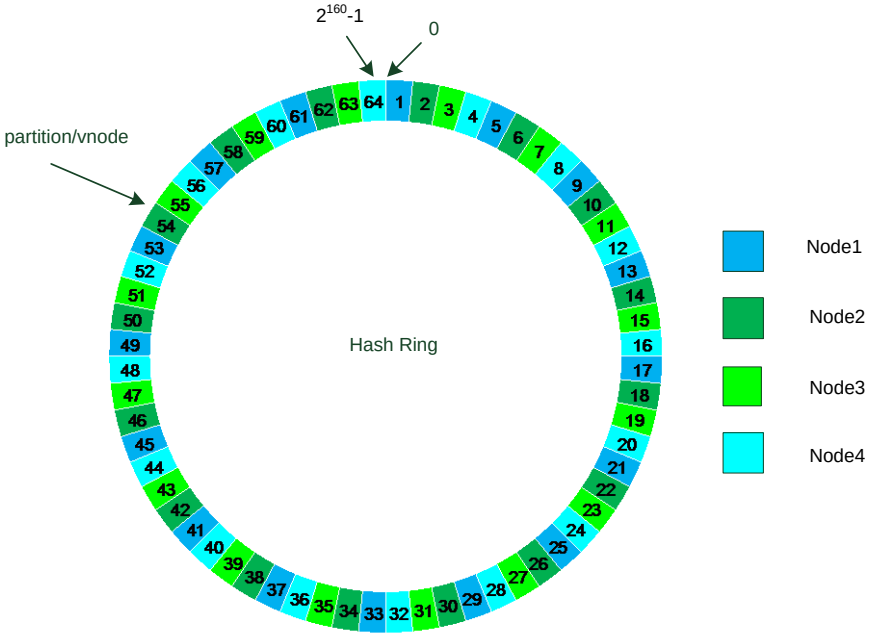

Figure 5. Consistent hashing
图 5. 一致性哈希

每个 vnode 都会落在一个分区中。当向数据库集群中写入数据(Object)时，会根据该 Object 的 bucket + key 生成一个哈希值，哈希到该整型空间中。该哈希值落入哪个分区，就表示该 Object 存放到该分区对应的 node 中。

3.1.2. 高可用性保障

1) 数据多重备份

CAP (Consistency, Availability, Partition tolerance)，一致性、可用性、分区容忍性是分布式数据库的基石，但这三个要素只能实现其中的两点，不可能三者兼顾，分区容忍性是基本要求，因此在分布式系统中，不能同时保证数据一致性和可用性，所以，需要在一致性和可用性之间做出平衡。

数据库系统把 CAP 的选择权交给用户，提供可调节的 CAP 机制。用户可以在 bucket 级别设置 N/R/W 参数，灵活的调节分布式数据库系统的可用性与一致性。

a) N：代表数据在分布式数据库系统中存多少副本；

b) R：读数据的最小节点数；

c) W：写成功的最小节点数。

R + W > N，保证读操作至少能读取到一个最新数据，读写速度受性能最差的节点影响，R 和 W 越小，系统的响应越快，W 越大，数据的一致性，可用性越强，通常：N = 3，R = W = 2。

数据库系统的数据请求处理过程如下：设定数据冗余存储 N 份，每次读取其中的 R 份，写操作需要写 W 份。通过计算得出请求的 key 所在的 N 个节点，向这 N 个节点依次发起请求，等待这 N 个节点中的 W 个(如果是写操作的)或 R 个(如果是读操作)返回成功，返回相应的数据给客户端。

假设(N,R,W)分别设为(3,2,2)，数据的读写流程如图 6 所示。

每个 Object 在数据库系统中都会有一个数据存储的位置的 prefer list，该 list 记录了这 N 份数据应该存储在哪个 partition/Vnode 中。

如图 7 所示，假设 N 值设为 3，3 号 vnode 存储了这个 Object，数据库系统会将该 Object 复制到 prefer

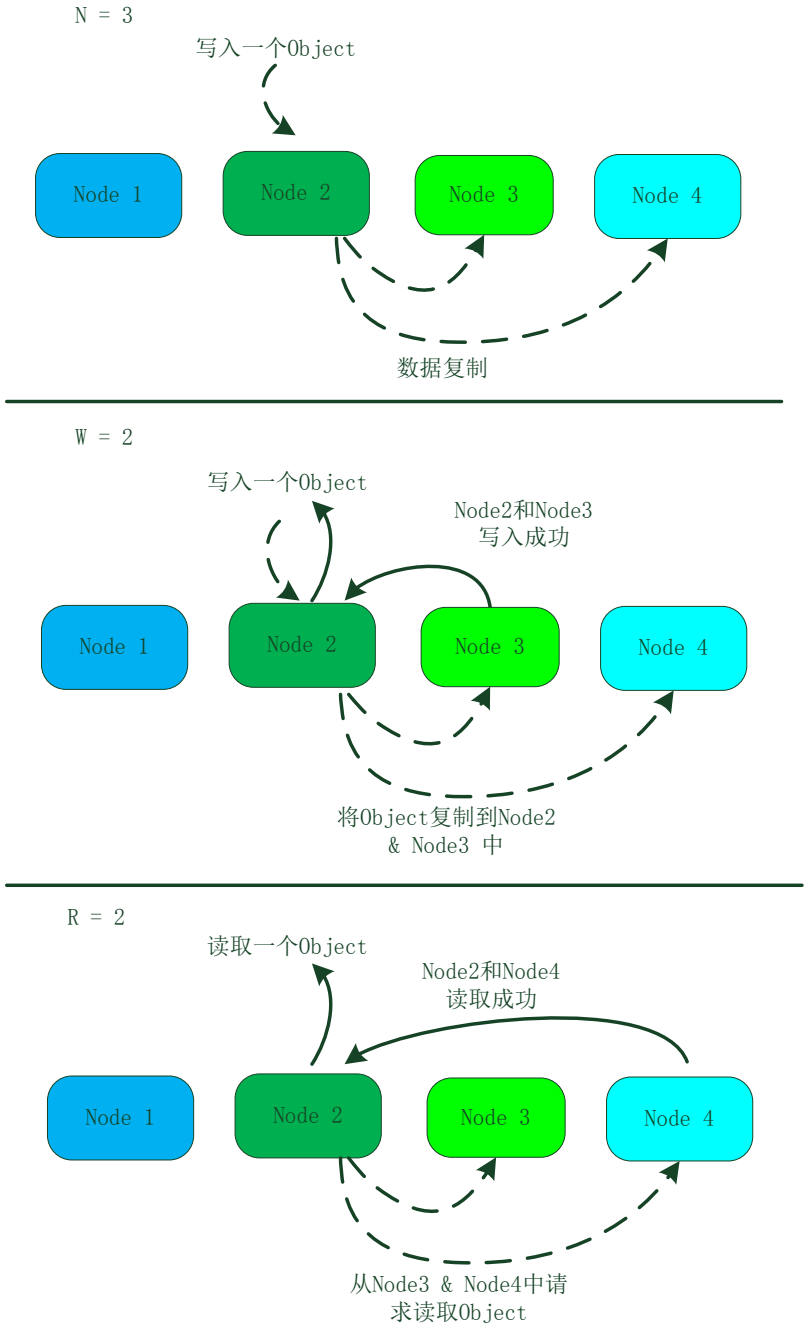

Figure 6. Data read/write process
图 6. 数据读写流程

list 中的另外的 2 个 vnode 中，就像图 7 中的 4、5。3、4、5 vnode 正好对应三个不同的物理节点 Node 3、Node 4 和 Node 1，这样任何一个或两个物理节点 down 掉也不会影响数据库的可用性。这样就保证了高可用。

当集群中的某个节点发生故障时，数据库系统会采用“hinted handoff”来处理对故障节点的数据写入操作，即当节点发生故障时，相邻节点会接管该故障节点的写请求操作；当故障节点恢复正常时，相邻节点向故障节点发送线索，归还接收到的更新。

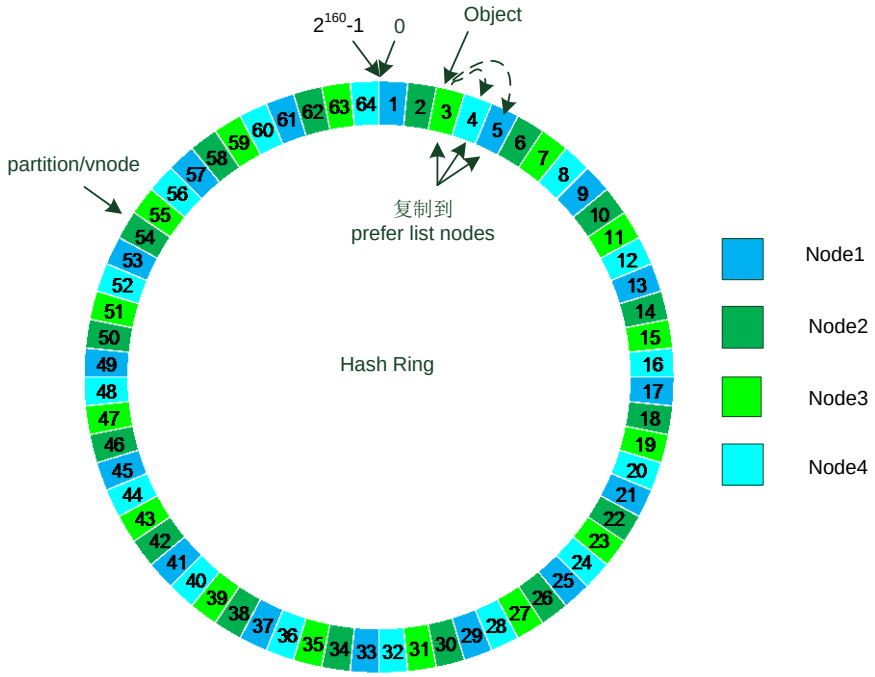

Figure 7. Hinted handoff failure handling
图 7. Hinted handoff 故障处理

2) 最终一致性

在大型分布式系统中，一些平常很少遇到的意外情况往往成为常态，如两个用户同时更新同一对象的数据，或进行数据复制时，由于某些服务器的故障或网络的中断，导致多个节点存储的数据不一致。所以数据的不一致性是必须被容忍的。

在该数据库系统中，采用最终一致性保证系统高可用，并采用向量时钟处理数据冲突[8]。

最终一致性是指，存储系统保证如果对象没有新的更新，最终(在不一致窗口关闭之后)所有访问都将返回最后更新的值。

如图 8 所示，数据在 Node 1、Node 2、Node 3 三个机器中存储了三份。

a) t0 时刻：三个节点中数据均为 A；

b) t1 时刻：Node 1 将数据更新为 B；

c) t2 时刻：此时，Node 2 中也接到更新请求，将数据更新为 B，但 Node 3 中依旧维持旧值 A；

d) t3 时刻：此时 Node 1 出现故障，Node 3 中接收到更新请求，将数据更新为 B；

e) t4 时刻：Node 3 将数据更新为 C；

f) t5 时刻：Node 2 接收到更新请求，将数据更新为 C；

g) t6 时刻，Node 1 中故障已恢复正常，数据也更新为 C，此时，Node 1，Node 2，Node 3 中的数据都已保持一致。

由此可见，最终一致性数据的不一致情况只是暂时的，数据最终会维持一致性状态。并且，个别节点的故障也不会影响整个系统的正常工作，保证系统高可用。

3.1.3. 多数据中心

该数据库系统还可实现多数据中心的主从复制。在多数据中心中，一个数据中心作为主数据中心，其它数据中心为从数据中心。主中心负责处理来自一个或多个从集群的复制请求，当主中心宕机时，一

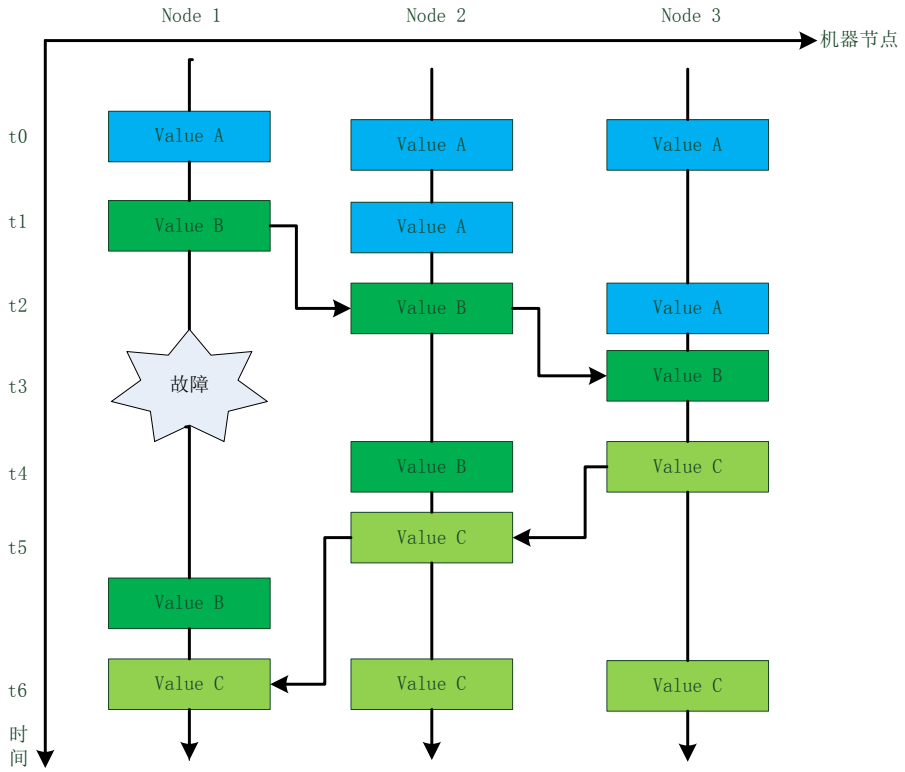

Figure 8. Data processing flow of eventually consistent
图 8. 最终一致性数据处理流程

个从中心会接管它的工作。

数据复制有两种方式：实时(REAL-TIME)或完全同步(FULL-SYNC)。数据库系统中使用了 source 和 sink 的概念，Source 表示用来发起复制的数据中心，Sink 表示接受复制数据的数据中心。

实时同步会在 source 发生更新时，以一种增量的方式复制到 sink 中，而完全同步在第一次建立连接时进行一次完全备份，之后会在固定的时间间隔内定期向 sink 进行完全备份。

在建立多数据中心时，source 站点会有一个或多个 node 作为 listener node，它可以监控 sink 站点的复制请求和它的连通性等。Sink 站点会有一个 site node，它和 listener node 相连。可以设置多个 node 作为 listener node 或 site node，但是只能有一个处于 active 状态的。

数据在多个数据中心间进行复制时，为保证复制速率，会建立多个并发的 TCP 连接。

3.2. 云存储 Portal

云存储 Portal 建立在分布式数据库系统之上，它可以用来构建公共或私有云，或作为可靠的存储应用程序和服务。

Portal 中，存储一个对象的过程如图 9 所示：通过 Portal 提供的存储 API 上传一个对象，Portal 会将该对象划分成一个个流式的小块，并将其存储到底层分布式数据库中，每个块都提供 metadata 以供以后检索使用。

3.2.1. 支持大对象和 Multipart 上传

云存储 Portal 可以支持大对象，并且可以将对象分成多个小块上传，用户可以将大对象划分成多个小块，每个小块作为一个单独的对象进行上传。划分后的小块可以并行进行上传，以加快上传速度。每

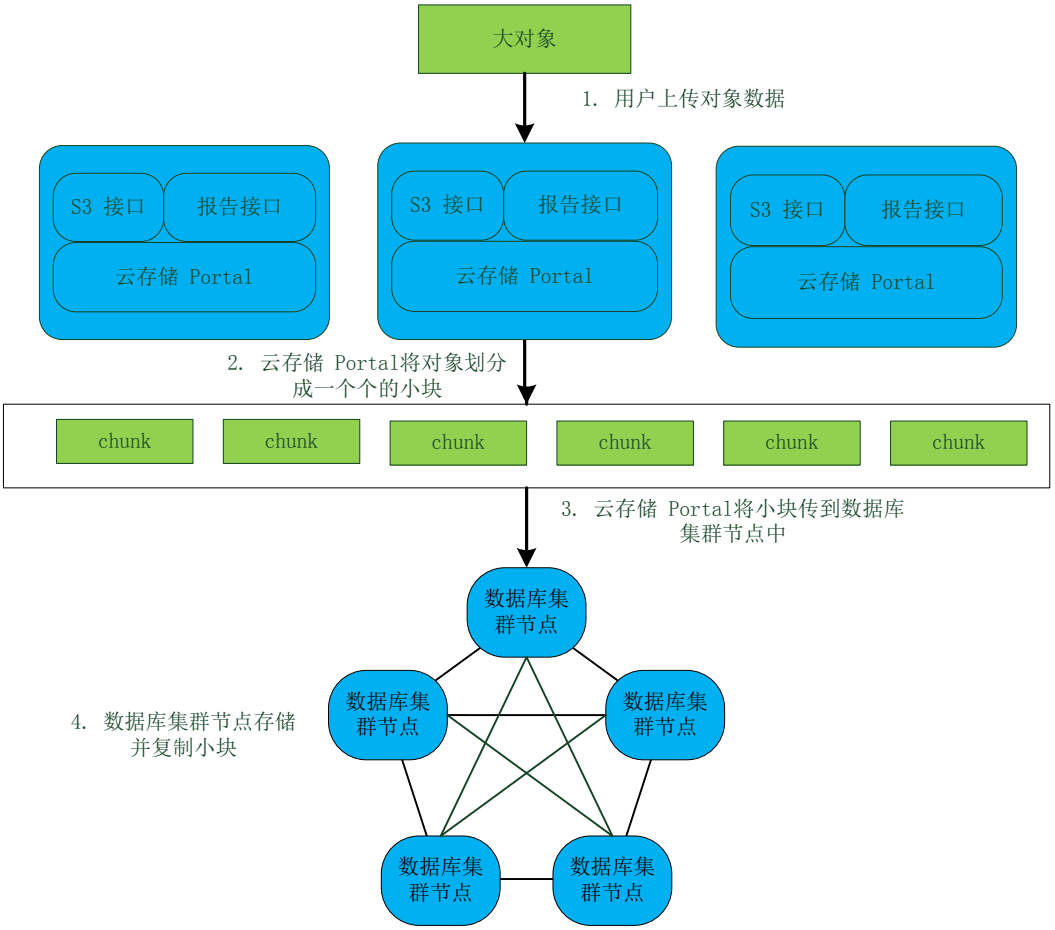

Figure 9. Data storage process in Portal
图 9. Portal 中的数据存储流程

个对象支持 5 MB 到 5 GB 的大小。

在上传时，会记录一个唯一的 upload ID，每个划分后的小块会有一个 part number，用来标记它在整个对象中的位置。当原始对象的所有小块都上传完毕后，Portal 会根据其上传 ID、所有的 part number 和其它信息，重新将对象组合还原。

3.2.2. 与 Amazon S3 兼容的 API

云存储 Portal 与 Amazon S3 应用程序界面相互兼容，提供易于使用的界面，允许企业使用现有 S3 工具与框架，或者导入和提取亚马逊数据。系统同时支持利用 Rest 对 bucket 和 object 进行 Get、Put、Delete 等操作。此外，云存储 Portal 还采用 Amazon S3 的 ACL 对 bucket 和 object 的操作权限进行控制，每个 bucket 和 object 都有一个 ACL。当一个请求发过来的时候，会检查 ACL 列表来确定当前请求是否具有相应的权限。当 bucket 或者 object 创建的时候，会生成一个默认的 ACL，会赋予资源所有者对资源的所有操作权限。

支持 Amazon S3 的身份认证机制。云存储 Portal 使用和 S3 一样的方式进行用户身份认证。在请求报文中，用户除发送 access_key 之外，还会根据请求内容和 secret_key 计算出一个签名一起添加在请求报文中。服务器端收到请求后，会根据 access_key 得到 secret_key，并使用同样的算法计算出签名，与请求报文中的签名进行比较，只有两者一致，访问才是合法的。

4. 总结

综上所述，该云存储系统通过多种技术，具备了以下特点：

1) 高可用性：对多个服务器进行数据读写操作，即使个别服务器或网络故障，也能保证数据的可用性。

2) 操作简单：集群需要扩容时，只需要简单的向集群中增加机器即可，不会给运营工作带来负担。

3) 易扩展：系统扩容，集群自动实现数据的负载均衡。

4) 容错性：不会因为网络分区或硬件故障而导致数据丢失。

对象存储技术虽然出现较晚，但是随着数据量的集聚增长，对象存储，尤其是赋予云计算特性的对象存储，必然会成为未来主流存储技术。

参考文献 (References)

[1] (2013) Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/cn/s3/

[2] Fox, A. and Griffith, R. (2009) Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

[3] 高东升 (2012) 大数据时代的云存储技术. *网络与信息*, **9**, 58-58.

[4] 朱平, 朱建涛, 高剑刚等 (2011) 高性能计算存储关键技术研究. *计算机研究与发展*, **48**, 354-364.

[5] DeCandia G., Hastorun D., Jampani M., Kakulapati G., Lakshman A., Pilchin A., Sivasubramanian S., Vosshall P. and Vogels W. (2007) Dynamo: Amazon's highly available key-value store. *Proceedings of* 21*st ACM SIGOPS Symposium on Operating Systems Principles*, Stevenson, 14-17 October 2007, 205-220.

[6] Lewin, D.M. (1998) Consistent hashing and random trees: Algorithms for caching in distributed networks. Massachusetts Institute of Technology, Massachusetts Institute of Technology, Cambridge.

[7] 覃灵军, 冯丹, 曾令仿等 (2006) 基于对象存储系统的动态负载均衡算法. *计算机科学*, **5**, 88-91.

[8] 罗军, 王宏, 李文生等 (2013) 基于向量时钟模型的NoSQL最终一致性的研究. *计算机工程与应用*, **23**, 100-102, 131.