大数据基础存储系统技术研究

王瑞通[1],李炜春[2]
(1. 南京邮电大学 计算机学院,江苏 南京 210003;
2. 福州大学 数学与计算机科学学院,福建 福州 350116)

摘 要:随着大数据技术的发展和对海量数据存储、分析需求的提高,成熟的分布式存储系统越来越多。通过对不同分布式基础存储系统内部的存储策略、管理策略、架构思想等关键技术点的对比和分析,对当前流行的分布式存储系统在设计思想、创新性技术上进行了追根溯源。对比传统数据存储与分布式数据存储的技术发展和应用实例,揭示了数据存储追求更大、更快、更安全的发展潮流,重点研究了大数据基础存储实例中基于文件、键值对和表格这三种分布式存储方式。正如网络技术的发展到 SDN(Software Defined Network)一样,存储方式也在发生深刻变化—软件定义存储。通过对当前大数据主流基础存储系统技术的发展和应用实例所进行的对比研究,为分布式存储系统,特别是基础存储系统的开发,提供了一些在系统设计上的参考,也为在从事大数据方面有业务需求的工作人员在选择底层存储策略时提供了借鉴。

关键词:大数据;存储架构;数据管理;基础数据存储系统;分布式存储系统
中图分类号:TP302 **文献标识码**:A **文章编号**:1673-629X(2017)08-0066-07
doi:10.3969/j.issn.1673-629X.2017.08.014

Research on Technology of Basic Large Data Storage System

WANG Rui-tong[1],LI Wei-chun[2]
(1. College of Computer,Nanjing University of Posts and Telecommunications,Nanjing 210003,China;
2. College of Mathematics and Computer Science,Fuzhou University,Fuzhou 350116,China)

Abstract:With the development of big data technology and the enhanced requirements on storage and analysis of the mass data,more and more mature distributed data storage systems have been provided. According to comparison and analysis on the key technical points like storage strategy,management strategies and architecture in the foundation system of various data distributed storage system,the existing prevailing distributed storage system is gone back to the roots from design philosophy and innovative technologiesl. Comparison on development and application between traditional and distributed data storage,it is revealed that data storage pursues the larger,the faster and the safer. The three distributed storage methods are emphasized including file-based,key-value-based and form-based in instances of basic big data storage. As the development of network technology evolved at SDN (Software Defined Network),the basic storage technology reaches SDS (Software Defined Storage). The elaborating current mainstream development and application instance of big data storage technology has been presented in detail as a reference of design in distributed storage system,especially in basic distributed storage system and some recommendations on the selection of the underlying storage with special business for engineers.

Key words:big data;storage architecture;data management;foundation system of data storage;distributed storage system

0 引 言

自 2013 年大数据元年之后,互联网、物联网、社交网络的数据洪流[1]不断冲击传统数据存储和处理手段。据 IDC(International Data Corporation)报告显示,到 2020 年全球数据总量将达到 40 ZB,全球在 15 年的数据总量为 7.9 ZB,而中国数据总量约占全球数据总量的 13%。面对数据的爆炸性增长,传统的数据存储系统、数据库技术和数据仓库架构越来越不堪重负。数据库也从传统的关系型数据库到 NoSQL 非关系数据库再到现在基于内存的 NewSQL 数据库递进发展。数据存储量和数据处理速度也在不断向前发展。目前基于分布式数据存储和计算的系统,在生产环境下可以很好地处理 PB 级数据,但仍面临海量数据增长的压力。

收稿日期:2016-07-07 **修回日期**:2016-10-19 **网络出版时间**:2017-07-05
基金项目:国家“863”高技术发展计划项目(2006AA01Z201)
作者简介:王瑞通(1990-),男,硕士研究生,研究方向为分布式存储、大数据架构。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20170705.1649.022.html

大数据具有的 4V 特性[2],即 Volume、Velocity、Variety 和 Veracity,也就是海量、快速、多样、低密度。追求快速的从海量且增长快速的低密度数据中进行知识发现是一项艰巨的任务。合格的大数据存储系统必须有效应对大数据时代的 4 大挑战,所以大数据存储系统都具有很高的 IOPS(Input Output Per Second)和大容量,支持动态水平扩展能力。当然易管理维护,多种数据类型兼容性强,高性能,低成本等需求也是刚性的。

根据分布式存储系统对数据存储格式的不同,分别对基于文件的分布式基础存储系统、基于键值对的分布式基础存储系统和基于表格的分布式基础存储系统进行了对比分析与研究。

1　数据存储方式

综合生产环境来看,主流外存设备使用的是机械硬盘(HDD)和固态硬盘(SSD),这两者和网络一起构成了数据存储的硬件基础。这里不讨论关于网络传输的相关问题,而重点讨论数据与硬盘之间的交互。以主流的 HDD 作为基础存储;SSD 为高性能新崛起的存储介质,在存储系统中最多只是作为 HDD 的辅助存储;全 SSD 作为存储基础并不是主流。

传统的底层数据存储,硬盘与服务器之间的连接需要接口协议[3],主要包括 SCSI、FC、SATA、SAS。操作系统扫描发现存储硬件后通过接口驱动程序连接使用硬盘。用户通过操作系统根据文件系统格式对硬盘数据块进行组织划分。Linux 系统下的文件格式多为 ext3 和 ext4。另外,通过 RAID 的方式可以灵活地对硬盘上的数据进行安全备份、错误容灾处理。通常还需要使用 LVM 对硬盘进行逻辑分区。

传统的存储系统无法满足大数据存储的高可靠、高速传输、动态可扩展和低成本等要求。磁盘瓶颈和网络瓶颈的掣肘导致新的存储架构的诞生,分布式存储系统便应运而生。这是一个通过网络连接各离散存储单元的网络存储系统。其本身所具有的高可扩展性、低成本、无接入限制等优点是传统存储系统所无法比拟的。分布式存储系统的核心技术是网络存储技术[4]。它将数据的存储由传统的服务器存储转移到网络设备存储。核心思想是将存储阵列与服务器分开,通过网络拓扑将两者连接起来。主要有以下三种:DAS、NAS 和 SAN。

DAS(直接附加存储)架构中,服务器与磁盘阵列通常采用 SCSI 连接。随着数据量的增大,SCSI 通道会成为瓶颈,而且没法进行动态扩展。NAS(网络附加存储)是通过网络协议(TCP/IP、FTP 等)连接服务器和磁盘阵列,相当于一个 LAN,提供文件级存储服务。由于使用统一的文件系统,每次增加磁盘容量就会对应增加服务器的负载。NAS 通常采用 RAID6 对数据进行容灾,这两者都增加了 NAS 的扩展成本。SAN(存储区域网)是通过交换机等网络连接设备将磁盘阵列与相关服务器连接起来的高速专用子网。服务器可以直接访问 SAN 中的数据,只有和文件信息相关的元数据信息才经过元数据服务器处理,减少了数据传输的中间环节。主流的大数据分布式存储系统基本都使用这一架构思想。目前有一种研究热点是将 NAS 和 SAN 相结合来构造更好的分布式存储系统,以期满足各种应用对存储系统提出的更多需求:大容量、高性能、高可用、动态可扩展、易管理维护等。Alluxio[5] 就是一种最新的尝试。

与传统数据存储系统相比,分布式存储系统有如下几点优势:

(1)成本低。分布式数据存储系统都是使用 x86 日用级硬件作为单节点,所以在硬件成本方面降低很多。通过优秀的架构设计和管理平台,有效降低了用户的维护成本和管理成本。云存储的出现显著降低了企业在数据存储方面的开销。

(2)动态可扩展。通过对服务节点的增删,部分分布式存储系统理论上可以无限制扩展容量。成熟的分布式存储系统可以轻松扩展到几千节点,满足 PB 级数据的存储和管理需求。用户在使用这些系统时可以依据具体业务需要对存储服务进行量身定制。

(3)高吞吐量。由于分布式存储系统采用分而治之的思想,对数据进行并行操作,天然地对大数据量的处理有很高的吞吐量。

在此基础上,根据对分布式系统的分类,对三种分布式基础存储系统进行阐述。

2　基于文件的分布式基础存储系统

2.1　分布式文件系统

文件系统是在操作系统中对存储空间的抽象,向用户提供统一的、对象化的访问接口。分布式文件系统(Distributed File System,DFS)的概念是相对于本地文件系统提出的。DFS 的文件系统管理物理存储资源不一定直接连接在本地,而是通过计算机网络与节点相连。DFS 基于 C/S 模式设计,通常提供多个供用户访问的服务器。由于网络的对等性,节点的客户端和服务端的角色是共存的。

目前的 DFS 主要通过 NAS 虚拟化[6],或者基于 x86 硬件集群和分布式文件系统集成在一起,以实现海量非结构化数据的存储。

为应对海量数据存储的需求,DFS 被设计具有以下特点。

(1)扩展能力强。从 GFS 开始元数据中心化管理,到后来 GlusterFS 采用无中心化管理,理论上系统具有无限的扩展能力。

(2)高可用性。通过中心分布式系统的 master 服务器 HA、网络分区等设计,保障数据的实时可用性。数据完整性则通过本地文件的镜像和文件自动修复、多地备份等手段来解决。

(3)高效弹性存储。可以根据业务需求灵活地增加或减少存储池中的资源,而不需要中断系统运行。弹性存储的最大挑战是减少或增加资源时的数据震荡问题。

(4)其他设计特点:压缩、加密、去重、缓存和存储配额。压缩减少了文件传输的带宽损耗。加密保障了数据传输的安全。值得一提的是,GlusterFS 由于采用 GNU/Hurd 的堆栈式设计,让这些功能模块可以方便地增减到系统。

2.2 分布式文件存储系统实例

DFS 的实现有多种,有 AFS、NFS、GlusterFS 等等。最著名的当属 GFS(Google File System)[7]。2003 年,GFS 的设计思想被 Google 公开,引起了业内轰动。2008 年,GFS 的开源实现 HDFS(Hadoop Distributed File System)[8]集成在 Hadoop 中并成为 Apache 的顶级项目。当今大数据时代,Hadoop 生态圈已成为大数据解决方案的工具箱。作为大数据的基础存储平台,HDFS 为 Hadoop 生态圈提供了最底层的支持。

2.2.1 HDFS 的架构设计

HDFS 借鉴 GFS 的思想把文件默认划分为 64 MB 的数据块(Block),采用主从结构。主控服务器用来实现元数据管理、副本管理、负载均衡管理、日志管理等操作。从属服务器负责数据块存储管理。在分散存储的同时,每个节点也是一个计算资源,进行数据的就近计算,提高了海量数据处理性能。很多其他分布式文件系统都借鉴了 GFS 的思想,如 TFS、Facebook Haystack[9]等。

HDFS 构建在普通商业服务器上的分布式文件系统,将组件失效看作是常态。数据以块(block)为单位进行存储管理。适用于一次写入多次读取的离线大数据处理。整体架构如图 1 所示。

NameNode 作为元数据处理节点,DataNode 作为块数据处理节点。元数据包含文件相关的信息主要有:文件路径,副本数,块大小,块数量,块号,生成时间戳,用户信息,权限信息,等等。NameNode 是集群的大脑,负责整个集群的健康监控,复制策略和负载均衡。为了保证 HDFS 的可用性,系统设计了 SecondaryNamenode。元数据服务器心跳信息缺失会导致集群不可用。块数据服务器心跳信息缺失会导致此 DataNode 不可用,此时 NameNode 会将此失效节点的数据写入其余节点,保证块数据复本数。通过第三方软件 Zookeeper[10]进行公平选举,达到存储集群的 HA。Zookeeper 是著名的 Paxos 一致性算法[11]的开源实现。集群做到 HA 之后,就可以对元数据服务器进行替换而不用中断集群运行。

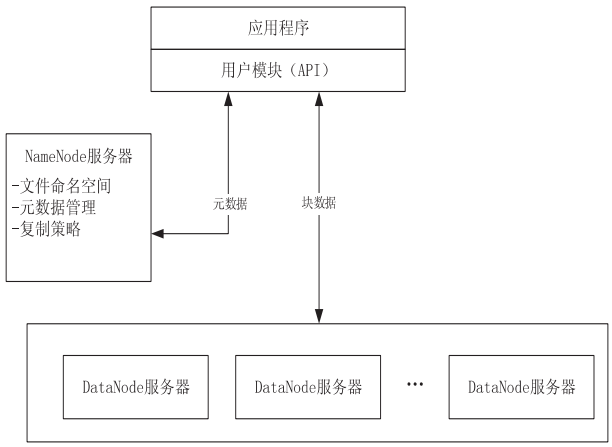

图 1 HDFS 的架构

为了保证数据的可用性,HDFS 将块数据存储在不同节点,默认存放 3 份。节点之间通过心跳机制交流信息。NameNode 只存放文件对应的块 ID,块的具体信息通过心跳机制由块服务器上报。元数据信息在集群运行期间常驻内存,提高了数据访问速度。应用程序首先通过 API 到 NameNode 处获得文件块号等元数据信息,然后直接到 DataNode 处读数据。存数据先要到 NameNode 处申请到块号等元数据信息,然后由 YARN 资源管理器分配资源,最后到具体的 DataNode 服务器进行写入。

2.2.2 HDFS 的高可用和一致性

HDFS 在保证数据可用性的同时会降低数据一致性。HDFS 是数据读写分离的。A 节点负责写入数据,然后将数据同步到 B 节点。B 节点提供数据读取的服务,当 AB 两个节点出现通信问题时,HDFS 保证 B 节点继续提供服务,但是数据不保证一致。即用户处于等待状态,一直等到数据同步完成后系统才继续提供服务。

HDFS 是一种高可用、易扩展、高性能且容错性强的分布式文件存储系统。在实际应用中也存在一些问题:不适用海量小文件处理,实时随机读写等。构建在此存储系统上的数据库系统 HBase(Google 的 BigTable 开源实现),在一定程度上提供了数据的低延迟随机读写能力。海量小文件会增加元数据服务器的压力,通常采用文件压缩、预取和缓存技术改善访问效率。NameNode 节点使用高速带宽(通常万兆带宽),大容量内存,SSD 做中间缓存等手段也是常用的。大数据技术全球领跑者 Google 的大数据管理路线是循着 Big-

Table[12]、Megastore[13]、Spanner[14]发展,而底层的数据存储支持是从 GFS 到 Colossus 技术革新。

3　基于键值对的分布式基础存储系统

3.1　分布式键值存储系统

分布式键值存储系统是一种基于主键构建键值对以实现对半结构化数据的简单 CRUD 操作的大数据基础存储方案。可以看作是一种分布式表格系统的简化实现。最典型的分布式键值系统是 Amazon 的 Dynamo 系统[15]。其他基于同一思想构建的系统还有 Facebook 的 Cassandra[16]和淘宝的 TaoBao Tair。自 07 年 Dynamo 的论文公开后,作为一种与 GFS 截然不同的经典大数据基础存储解决方案一直在发展进步。当前云存储的关注热点在分布式对象存储系统。如 Amazon 的 S3[17]和开源的 OpenStack 的 Swift[18],都可视为分布式键值存储系统。Dynamo 构建主键是基于文件数据,而 S3 和 Swift 构建主键是基于对象。所谓对象其实就是文件数据及其属性信息的集合。分布式键值存储系统的核心技术要点有:P2P 非中心化系统设计,一致性哈希技术,数据最终一致性实现,数据写的高可用性,系统故障处理和动态水平扩展的实现。

3.2　分布式键值系统实例

Dynamo 是一个高性能、高可用、易扩展的分布式键值存储系统。图 2 是 Amazon PaaS(平台即服务)架构。客户端有服务需求,如数据需求或计算需求,通过网页中的个性化组件向聚合服务器请求服务。服务端和客户端通过 LSA(服务水平协议)控制对用户的服务项目和质量。每个聚合服务器都负责一类服务,由多个服务器实例组成。服务器实例可有多种,如 Dynamo 实例、S3 服务实例、其他数据存储实例等等。

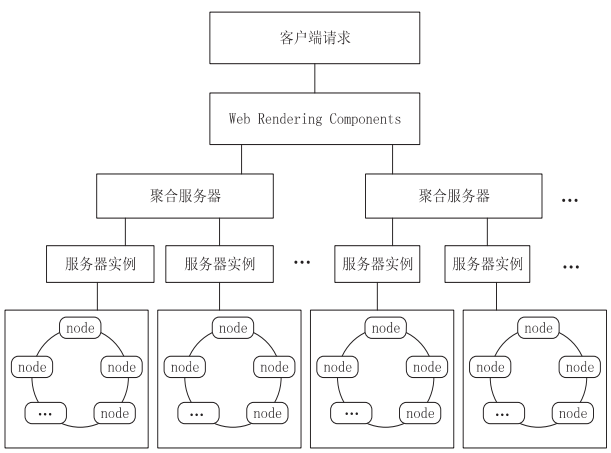

图 2　Dynamo 架构图

3.2.1　Dynamo 的一致性与高可用

亚马逊对 Dynamo 的一个很重要的设计目标是保证写的高可用性。在其他服务器或网络出现故障时,购物车服务器仍提供写服务。一个服务器实例就是一个 P2P 网络架构,去中心化。这是和 HDFS 网络架构的显著区别。当数据依据某个副本策略进行数据备份,出现网络故障时,Dynamo 通过降低数据的可用性来保障数据的强一致性。

Dynamo 被设计成保证数据的最终一致性,即所有的更新都在所有副本中体现。在数据有更新时,利用向量时钟体现版本变化。如图 3 所示,原始的数据 D 被节点 S_x 更新后成为 D_1,但是伴随着数据的有一个向量时钟($[S_x,1]$)。在数据 D_2 分别被 S_y 和 S_z 更新而没有统一体现在所有副本之后,客户端会读取到两个数据 D_3 和 D_4,此时通过向量时钟可以确定数据的共同祖先 S_x,并由 S_x 对数据 D_3 和 D_4 合并更新。这样通过向量时钟获取了数据的上下文变化信息,最终保障了数据的强一致性。同样,通过对数据读取过程的协调,保证了数据写的高可用。

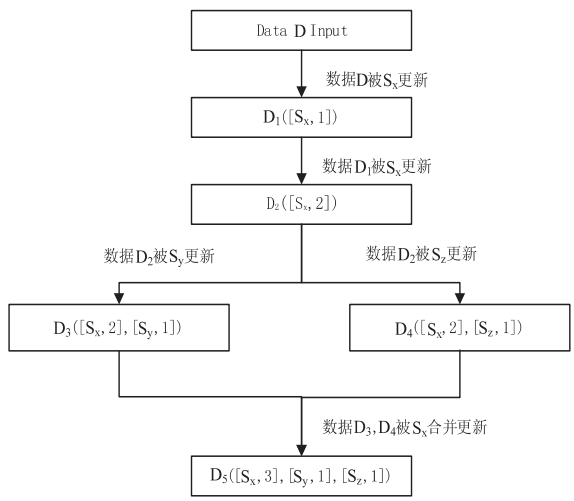

图 3　Dynamo 数据的最终一致性

Dynamo 数据使用一致性哈希环将数据动态分配到存储节点上。每份数据对应于一个唯一主键。主键通过 MD5 哈希生成 128 位的特征码。将所有的特征码首尾相接就形成了图 4 所示的哈希环。每个节点负责根据不同的副本策略存储数据。如图 4 中显示的数据副本数是 3,即每个节点存储当前节点和此节点之前的三个节点之间三段 Key Range 所对应的键值对数据。例如,D 节点会存储(A,B]、(B,C]和(C,D]三个 Key Range 对应的数据。该设计的一大优点是,数据的变动只会影响临近节点而不影响其他节点。

为了消除数据热点的出现,避免节点因热数据负载过高,Dynamo 使用了“虚拟节点”的概念。一个物理节点可以由多个虚拟节点负责。每个物理节点最初只包含本地节点和 token(虚拟节点标识)的信息。系统启动后通过 Gossip 协议[19]与对等节点交换信息。整个系统的分区信息和部署信息也经 Gossip 协议传输到所有物理节点。物理节点的增减也通过 Gossip 协议告知临近节点。每增加一个节点,此节点会被分

配对应的 token 来服务存储集群。删除节点时,对应的虚拟节点出现变化,集群会将被移除节点的数据分配到临近虚拟节点,保证数据副本数。此过程只影响临近的虚拟节点,所以系统内不会出现网络风暴。

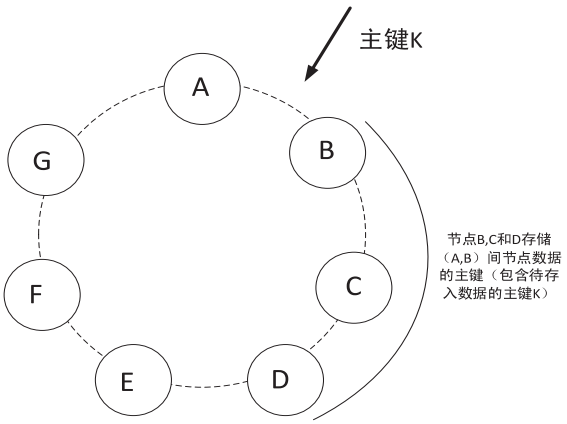

图 4 Dynamo 环上的分区与复制

3.2.2 Dynamo 的灵活读写策略

Dynamo 通过对数据的 get(key)和 put(key)原子操作的动态调整,获取灵活的数据处理性能。在读写操作出现障碍时,使用一个类似于仲裁的一致性协议来保证副本的一致性。协议参数 R 表示判断一个读取操作成功的最少节点数目;协议参数 W 表示判断写操作成功的最少节点数;协议参数 N 表示读写操作涉及的健康节点数,或者是数据目标副本数。当业务在读请求上需求量大时,设置较大的 N 保证数据的健壮性,选择较大的 W 和较小的 N。这样就牺牲了写性能,获得更高的读性能。相反,可以选择较小的 W 和较大的 N 来适应业务大量写操作的需求。以牺牲读性能换取更高的写性能。默认设置 $W+R>N$,认为这是一个可靠仲裁系统,能够保证每次都能读取最新的数据。反之则不然。

此外,Dynamo 采用反熵(anti-entropy)协议[20]来保持副本的同步,采用 MerkleTree[21]来提高副本一致性检验的性能。

以上描述了 Dynamo 这个典型的分布式键值对存储系统的数据存储过程。重点介绍了数据一致性保障、集群水平扩展实现、集群健壮性保障等方面的主要技术要点。

4 基于表格的分布式基础存储系统

4.1 分布式表格存储系统

在国内,大数据生产环境中对 Hadoop 生态工具的运用越来越多。为了适应不同的应用场景,基础存储服务一直由 HDFS 和 HBase 提供。HBase[22]是基于 HDFS 的分布式列簇存储数据库。简单来讲,HBase 就是在分布式文件系统上对数据加了一层索引,它适用于处理半结构化和结构化数据。HBase 实质是对数据索引进行管理的平台。在高吞吐量的批处理场景下,优先选择 HDFS 系统作为存储。在追求低延迟,有随机读写需要的场景下,优先选择 HBase 系统作为存储。市场需要一个既支持高吞吐量又支持低延迟随机读写的存储系统。Kudu 就是以此为目标而设计的基础数据存储解决方案。

4.2 分布式表格存储系统实例

Kudu 是 Cloudera 公司在 2015 年公开的一种纯列式面向结构化数据,支持低延迟随机读写且具有高性能处理数据能力的开源大数据存储引擎。Kudu 一开始的设计目标就是紧密融入 Hadoop 生态圈,对 Yarn、Hive、Mesos 等的支持度很高。Kudu 可以很好地支持 Spark 的内存计算框架和 MapReduce 的键值对计算框架。由于 Kudu 刚刚公布的新型系统,目前仅能获取 beta 版,国内只有小米公司使用了半实验半生产的 Kudu 千节点集群。作为一种新型的分布式基础存储系统有很多创新点值得学习。

4.2.1 Kudu 的架构设计要点

图 5 是 Kudu 的架构图,Kudu 存储系统由三个 master 服务器和若干 tablet 服务器组成。Master 服务器负责管理 metadata,是集群的管理者。每个 tablet 服务器服务多个 tablet 并维护 tablet 的主从状态。每个 tablet 包含多个具体表。Tablet 的主从之间使用 Raft 一致性算法来保证它们间的数据同步。

在使用者看来,Kudu 是一种基于表结构的数据存储系统。一个 Kudu 集群拥有很多数量的表,每一个表都有模式(schema)定义。模式包含一个有限的列。每个列都有一个名字和模式(int32 或 string)是否为空的选项。在这些列中有已经排序好的子集(叫做主键),主键列用户不可添加或删除。行具有主键唯一约束性。主键是删除、更新等高等级操作的唯一依赖。针对行的主键可以进行唯一索引建立,这提高了行的更新删除性能。这种数据模式很像是关系型数据库,而不同于很多分布式数据库存储方式,如 Cassandra、mongoDB、Riak、BigTable 等。

Kudu 对于类型的定义可以具体到某个列。这与其他 NoSQL 对数据类型定义为“everything is bites”不同。这是因为有以下两点考虑:指定类型允许使用特定类型的列编码格式,例如使用整数的位填充;指定类型允许使用类 SQL 的元数据传输到其他系统,例如使用常用的 BI 或数据处理工具。Kudu 自定义列类型,提高了列扫描的性能,实现了随机数据更新的低延迟。

Kudu 在保障数据一致性时提供了两种一致性模型:使用快照,这是单客户节点“读你所写”一致性的可靠实现;使用时间戳令牌。用户自定义在客户端之

间传输时间戳:在写操作后,用户在客户端请求一个令牌时间戳,这个 token 可以通过外部通信传输到另外的客户端,在 APIs 处理数据时能够获得不同客户端对同一数据的写操作的前后关系。注意这里与 Dynamo 使用向量时钟保障一致性的实现方法是不同的。Kudu 使用 spanner(类似控制器组件)提供 token 锁服务,使用 HibridTime 时钟算法[23] 获取时间戳。两者的结合使用基本确保了数据的一致性。

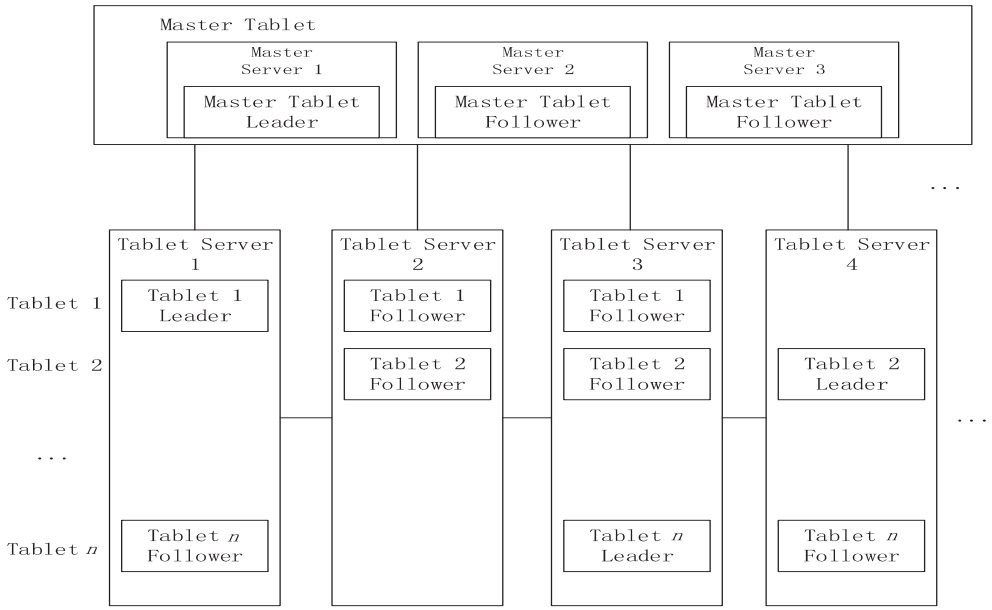

图 5　Kudu 的架构图

数据依据主键在 Kudu 里进行水平分区。每个分区都有分区 key 并对应某个 tablet。主键元组映射到分区 key。这确保了每次对数据的插入更新操作只针对一个 tablet,降低了数据处理的复杂度。Kudu 支持灵活的分区策略,建表时自定义对表设置某种分区模式。主要分区规则包括:hash 分区规则和 range 分区规则。这些灵活的分区策略具备典型的“NoSql”特性,使得通过 MPP(Massive Parallel Processing)分析数据的用户和管理员能够很快入手 Kudu。

4.2.2　Kudu 的存储技术

一个 tablet 被 Kudu 细分为多个行集合(RowSets),这是 Kudu 的最小存储单位。Kudu 没有 HDFS 中块的概念。对行集合的独立定义保证了行的独立性。Kudu 基于行集合进行负载均衡、执行 WAL 日志策略、配置数据读写 IO 阈值和确定数据压缩策略等数据管理操作。行集合又分为内存行集合(MemRowSet)和磁盘行集合(DiskRowSet)。

内存行集合的实现是基于内存的并发 B 树实现的。它使用了乐观锁,借鉴了 MassTree 的设计实现。每个 tablet 只有一个内存行集合保存当前最近插入的行。其他落地的行集合成为磁盘行集合。当一个内存行集合被选择进行刷写处理时,一个新的空内存行集合被创建并接管上一个内存行集合的工作。前一个内存行集合转换为一个或多个磁盘行集合。这个裂变过程类似于 HBase 的 HFile 的裂变过程。刷写过程是并发的。在完成刷写过程之前,任何对这一过程的数据进行更新删除操作都会被追踪,以便回滚时使用。在内存行集合进行刷写的过程中,转变的磁盘行集合默认每 32 MB 大小一个轮回写入磁盘。这确保磁盘行集合不会过大,提高了增量压缩的性能。

数据插入过程:插入前,Kudu 会扫描所有 tablet 的磁盘行集合,选取某个磁盘行集合执行插入操作。此过程中,使用每个磁盘行集合的布隆过滤器确保插入数据 key 的唯一性。构建 key 的 B 树索引并常驻内存以减少磁盘 IO 负载。

数据读取过程:Kudu 通常通过被读取数据的主键上下界或上界进行批量读操作。Kudu 选择了基于内存的批处理方式。内部实现是一种类金字塔的层级关系,使用指针从高层指向底层最终指向更小的被读列块。批处理格式在内存中是列式的,避免了偏移量计算的开销。Kudu 执行按列扫描直到找到目标列并得到行偏移量。

其他技术要点:使用心跳机制和 Raft 算法(一种 Paxos 算法)配合达成选举策略。懒实体化列扫描策略,优先选取满足自定义条件的列。广泛地使用压缩,如对行集合压缩和中间缓存数据的压缩溢写。

5　结束语

海量数据不断冲击当前的存储技术体系,不同的分布式存储技术百花齐放、百家争鸣。除了以上介绍的基础存储方式,还有混合存储方式。如 Ceph 的存储系统,根据对存储对象的定义不同,可以灵活存储不同

类型的大数据资产。异军突起的 Alluxio 是以内存为中心的虚拟分布式存储系统,能够依据对底层任意不同的基础存储系统(如 HDFS 或 AWS 等)存储数据的处理性能需求,自定义或者自动针对特定数据选择存储方式和存储介质。它是构建在计算框架和存储框架之间高效、灵活的管理者。高效的存储架构设计、创新的存储模型思想、灵活的存储手段整合,这三者是应对海量数据存储、应用挑战的重要手段。大数据基础存储系统是构建 MPP 系统、NoSQL 系统、混合系统等的核心。

围绕基于文件、键值对、表格等三大基础大数据存储系统的技术发展脉络进行介绍。梳理出当前大数据基础存储技术的发展现状和发展潮流。对这三种典型的大数据基础分布式存储系统的技术、架构进行了研究和对比。结合具体实例介绍了大数据分布式基础存储系统的基本概念、关键技术和架构设计要点。早期基础存储系统宝刀未老,新型基础存储系统来势汹汹。基础存储技术作为大数据发展的基石,将继续成为工业界和学术界的研究热点。

参考文献:

[1] 孟小峰,慈 祥. 大数据管理:概念、技术与挑战[J]. 计算机研究与发展,2013,50(1):146-169.

[2] Gantz J,Reinsel D. Extracting value from chaos[R]. [s. l.]: IDC IVIEW,2011.

[3] 张 冬. 大话存储[M]. 北京:清华大学出版社,2008:19-58.

[4] 王 月,贾卓生. 网络存储技术的研究与应用[J]. 计算机技术与发展,2006,16(6):107-109.

[5] Li H,Ghodsi A,Zaharia M,et al. Tachyon:reliable,memory speed storage for cluster computing frameworks[C]//Proceedings of the ACM symposium on cloud computing. [s. l.]: ACM,2014:1-15.

[6] 张成峰,谢长生,罗益辉,等. 网络存储的统一与虚拟化[J]. 计算机科学,2006,33(6):11-14.

[7] Ghemawat S,Gobioff H,Leung S T. The Google file system[C]//ACM symposium on operating systems review. [s. l.]: ACM,2003:29-43.

[8] Shvachko K,Kuang H,Radia S,et al. The Hadoop distributed file system[C]//IEEE 26th symposium on mass storage systems and technologies. [s. l.]:IEEE,2010:1-10.

[9] Quan D,Huynh D,Karger D R. Haystack:a platform for authoring end user semantic web applications[C]//Proceeding of ISWC. [s. l.]:[s. n.],2003:738-753.

[10] 刘 芬,王 芳,田 昊. 基于 Zookeeper 的分布式锁服务及性能优化[J]. 计算机研究与发展,2014,51(S):229-234.

[11] 许子灿,吴荣泉. 基于消息传递的 Paxos 算法研究[J]. 计算机工程,2011,37(21):287-290.

[12] Chang F,Dean J,Ghemawat S,et al. Bigtable:a distributed storage system for structured data[J]. ACM Transactions on Computer Systems,2008,26(2):4

[13] Baker J,Bond C,Corbett J,et al. Megastore:providing scalable,highly available storage for interactive services[C]//Fifth biennial conference on innovative data systems research. Asilomar,CA,USA:[s. n.],2011:223-234.

[14] Corbett J C,Dean J,Epstein M,et al. Spanner:Google's globally-distributed database[C]//USENIX conference on operating systems design and implementation. [s. l.]: USENIX Association,2012:251-264.

[15] Decandia G,Hastorun D,Jampani M,et al. Dynamo:amazon's highly available key-value store[C]//ACM SIGOPS operating systems review. [s. l.]:ACM,2007:205-220.

[16] Lakshman A,Malik P. Cassandra:a decentralized structured storage system[J]. ACM SIGOPS Operating Systems Review,2010,44(2):35-40.

[17] Guide G S. Amazon simple storage service[EB/OL]. 2016-03-01. https://s3. cn-north-1. amazonaws. com. cn/aws-dam-prod/china/pdf/s3-ug. pdf.

[18] Arnold J. OpenStack swift:using,administering,and developing for swift object storage[M]. [s. l.]:O'Reilly Media,Inc,2014.

[19] 刘德辉,尹 刚,王怀民,等. 分布环境下的 Gossip 算法综述[J]. 计算机科学,2010,37(11):24-28.

[20] van Renesse R,Dumitriu D,Gough V,et al. Efficient reconciliation and flow control for anti-entropy protocols[C]//Workshop on large-scale distributed systems & middleware. [s. l.]:[s. n.],2008:1-7.

[21] 秦志光,王士雨,赵 洋,等. 云存储服务的动态数据完整性审计方案[J]. 计算机研究与发展,2015,52(10):2192-2199.

[22] Vora M N. Hadoop-HBase for large-scale data[C]//International conference on computer science and network technology. [s. l.]:IEEE,2011:601-605.

[23] Alves D,Garg V. Hybridtime-accessible global consistency with high clock uncertainty[R]. [s. l.]:[s. n.],2014.