文章编号:1000-5641(2014)05-0164-09

OceanBase 分布式存储引擎

黄 贵, 庄明强
(阿里巴巴集团公司,杭州 310000)

摘要: OceanBase 是一个分布式关系型数据库,其目的是存储海量的高速增长的结构化数据,以廉价的服务器集群实现高可用、高可扩展、高性价比的服务. OceanBase 采用内外存混合存储的模式,使用内存存储增量(新写入)数据,而使用外存存储基线(只读)数据,并将基线数据划分成大致等量的数据分片并采用分布式 B^+ tree 的形式将分片存放在很多的数据服务器上,利用定时合并机制不断将增量数据与基线数据融合. 本文介绍 OceanBase 基线数据存储的基本结构和分布方式、定时合并机制,以及基线数据在 OceanBase 中的具体存储格式的设计和实现.

关键词: 存储引擎; 分布式系统; 每日合并; 分块存储; 基线数据; 增量数据
中图分类号: TP31 **文献标识码**: A **DOI**:10.3969/j.issn.1000-5641.2014.05.014

Scalable distributed storage of OceanBase

HUANG Gui, ZHUANG Ming-qiang
(*Alibaba Group*, *Hangzhou* 310000, *China*)

Abstract: OceanBase is a distributed relational database, its purpose is to store vast amounts of structured data in high-growth, low-cost servers to achieve high availability, high scalability and cost-effective services. OceanBase using memory and external store hybrid storage mode, stores the incremental (update) data in memory, and the baseline (read-only) data in external storage (usually disk), baseline data is divided into slices we called tablet roughly the same amount of data and the use of distributed B^+ tree stored on many data servers, using the daily merge mechanism to keep the combined incremental data into baseline. This article describes the basic structure and distribution methods of OceanBase baseline data storage, as well as the daily merge mechanism, in addition, we will introduce in OceanBase baseline data storage format of the specific design and implementation.

Key words: storage engine; distributed system; daily merge; block stable store; base data; increment data

0 引 言

随着大数据时代的到临,越来越多的应用需要处理数百 TB 乃至 PB 级的数据. 当前,学术界与工业界已经出现了多种数据管理技术,但是这些技术在带来一定优势的同时,也具有

收稿日期:2014-06
第一作者:黄贵,男,阿里巴巴集团技术专家,研究方向为分布式数据库. E-mail:qushan@alipay.com.

一定的局限性. 首先,分布式文件系统 GFS[1]、分布式表格系统 BigTable[2] 和全球分布式数据库 Spanner[3] 使用分布式系统解决了 PB 级数据的实时存储和查询,但 BigTable 不支持完整的关系数据模型,使用有一定的限制;传统关系数据库(例如 Oracle[4]、DB2[5] 和 MySql[6])功能完善,但可扩展性欠佳;内存数据库(VoltDB[7]、MemSQL[8])已具备高性能,但其受内存大小和成本限制. 因此,亟需设计与开发新型数据管理技术,以实现高性、低成本、可扩展和高可用性等目标.

为了满足阿里巴巴迅速增长的数据处理需求,我们设计、实现并部署了分布式关系数据库 OceanBase,以可靠地处理数百 TB 级别的数据. OceanBase 能部署在数百台机器上,具有关系型数据库的功能,并已经实现了上述四个目标. OceanBase 将分布式和关系数据库巧妙结合,充分发挥分布式系统可扩展强和高可用性优点,在相对廉价的普通硬件上提供可靠的数据服务. OceanBase 使用了很多关系型数据库的设计策略,支持完整的关系数据模型. 分布式存储引擎是 OceanBase 的核心模块之一,主要分为两大部分:内存事务存储引擎和分布式基线数据存储引擎. 本文主要介绍分布式基线数据存储引擎的设计和实现.

本文第 1 节描述 OceanBase 基线数据、增量数据的数据模型;第 2 节描述每日合并如何将基线数据与增量数据进行融合;第 3 节描述数据分片的容错与负载均衡;第 4 节描述基线数据分裂与合并;第 5 节描述基线数据分块存储格式;第 6 节总结全文.

1　数据模型

1.1　读写分离结构

大量数据库应用每天增量修改的数据量相对较小,而保持不变的数据量很大,这两部分数据需要根据不同特点采用不同的存储管理方式. 因此 OceanBase 使用读写分离的结构,将数据分为基线数据和修改增量,增量删改数据保存在单台服务器内存中(redo log 保存在磁盘),称为 MemTable,对应的基线数据(即数据库在 MemTable 开始时刻的快照),按照行主键排序后,动态分成多个数据分区,称为 Tablet,每个 Tablet 的多个副本均匀分布到多台数据节点(我们称之为 ChunkServer)的多块磁盘(通常是 SSD)上,Tablet 是数据分布和负载均衡的最小单位. 图 1 描述了读写分离结构.

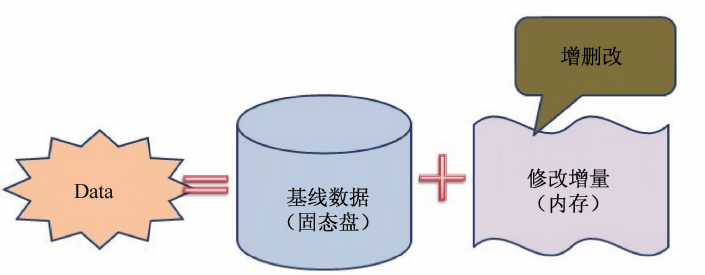

图 1　OceanBase 的读写分离结构

Fig. 1　The read/write separate architecture

单台服务器内存有限,无法存储持续所有修改增量,因此 OceanBase 通常每天在某个时刻(例如后半夜业务低谷期)冻结当前 MemTable 并开启新的 MemTable,此后新的增删改写入新的 MemTable,然后系统在后台将冻结的 MemTable 与当前基线数据融合,生成新的基线数据,这个过程称为每日合并. 每日合并完成后,冻结的 MemTable 以及旧的 Tablet 即

可释放,其占用的内存也被回收.

1.2 分布式 B+tree 结构

一个 OceanBase 集群存储了很多表,每个表包含了一个 Tablet 集合,每个 Tablet 包含某个范围内的所有行数据. OceanBase 使用一个三层、类似 B+树的结构存储 Tablet 的位置信息,如图 2 所示:

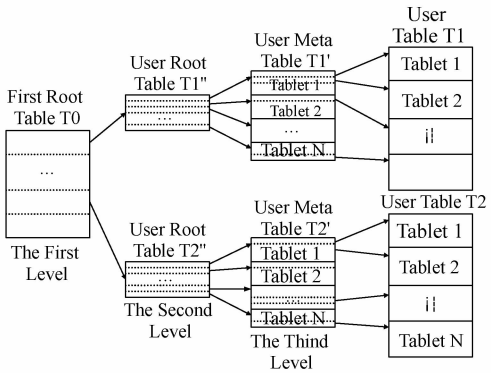

图 2 Tablet 位置层级图

Fig. 2 Tablet hierarchy chart

图 2 最右边为 User Table,每个 User Table 的基线数据包含多个 User Tablet,在每日合并中 User Tablet 按照配置的大小分裂成多个 User Tablet,共分为三层. 第三层为 User Table 的 Meta Table,称为 User Meta Table. 每个 User Table 对应一个 User Meta Table,每行存储一个 User Tablet 的位置信息,User Meta Table 的基线数据包含多个 User Meta Tablet. 第二层为 User Meta Table 的 Meta Table,称为 User Root Table,每行存储一个 User Meta Tablet 的位置信息,User Root Table 的基线数据只包含一个 Tablet,不允许分裂. 第一层为 User Root Table 的 Meta Table,称为 First Root Table,每行存储一个 User Root Tablet 的位置信息,First Root Table 的基线数据只包含一个 Tablet,不允许分裂.

OceanBase 采用分块 SSTable 格式来存储基线数据的文件. 分块 SSTable 是持久化、按行主键有序、不可更改的 map 结构. 存储基线数据的数据节点包含一个 Tablet 集合,OceanBase 使用类似 B^+ 树的结构来定位行数据在数据节点磁盘上的位置,如图 3 所示.

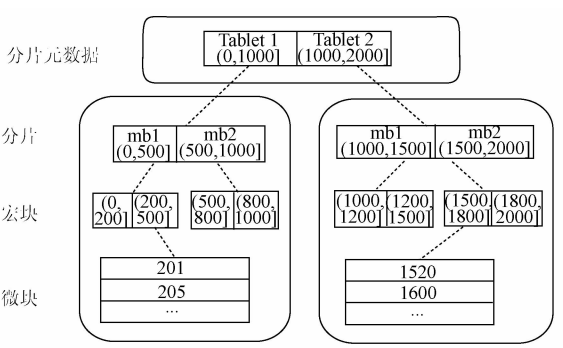

图 3 数据存储逻辑图

Fig. 3 Data storage logic diagram

每个 Tablet 使用一个 SSTable 存储，分块 SSTable 包含一系列宏块（通常每个宏块的大小是 2M，这个大小可以配置），分块 SSTable 使用宏块索引定位宏块位置，在打开 SSTable 时，SSTable 的宏块索引被加载到内存中. 宏块内部包含一系列微块（通常大小为 16K，这个大小可以配置），宏块使用微块索引定位微块位置，在打开宏块时，宏块的微块索引被加载到内存中.

在数据节点查询数据时，通过二分查找在 Tablet 索引中找到 Tablet，然后从 Tablet 对应的 SSTable 的宏块索引中使用二分查找找到宏块位置，然后从宏块的微块索引中使用二分查找找到微块位置，然后从硬盘读取相应的微块，最后从微块中查询到需要的数据.

2　每日合并

2.1　每日合并过程

随着写操作的执行，MemTable 的大小不断增加，当 MemTable 尺寸达到某个阈值或者系统达到预设冻结时间的时候，OceanBase 冻结当前 MemTable 并创建新的 MemTable，每日合并将冻结的 MemTable 与当前基线数据在后台进行融合，生成新的基线数据. 每日合并完成后，冻结的 MemTable 以及旧的 Tablet 即可释放，其占用的内存也被回收. 每日合并可以收缩服务器使用的内存，以及在服务器灾难恢复过程中，减少必须从提交日志里读取的数据量. 在每日合并过程中，正在进行的读写操作仍能继续. 每日合并过程如图 4 所示.

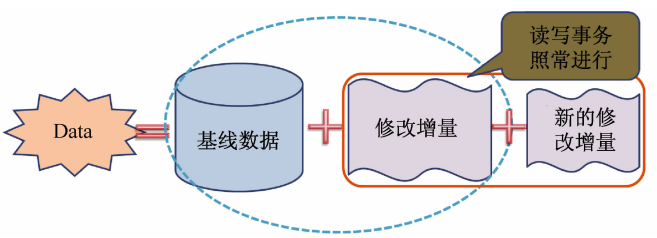

图 4　每日合并

Fig. 4　Daily merge

MemTable 冻结后，存储基线数据的数据节点启动多个合并线程，每个合并线程以 Tablet 为单位，向存储修改增量的服务器请求 Tablet 的修改增量数据，并与该 Tablet 的本地基线数据进行融合. 在每日合并过程中，Tablet 的不同副本按照相同的规则进行分裂，当所有数据节点的 Tablet 都合并完成后，回收冻结的 MemTable 和旧 Tablet，释放内存和磁盘空间.

2.2　每日合并版本更新

OceanBase 使用版本号管理基线数据和增量数据，增量数据一般比基线数据高一个版本. 每日合并时，将冻结的高版本增量数据与低版本的基线数据合并成新的高版本基线数据. 例如基线数据版本为 1，增量数据版本为 2，MemTable 冻结时创建一个新的 MemTable 提供写服务，冻结的 MemTable 版本为 2，新创建的 Memtable 版本为 3，然后每日合并将版本为 1 的基线数据与版本为 2 的冻结 MemTable 合并成版本 2 的基线数据.

为了减少版本管理的负担，存储基线数据的数据节点保存两个版本的基线数据，冻结的高版本增量数据在该版本每日合并完成后立即删除. 用户可以指定保存最后 n 个版本（一般

是两个,可以修改)的基线数据.

由于基线数据是多个副本分布在多个 ChunkServer 上,而每个 ChunkServer 的合并进度并不一致,在合并过程中的读事务会因为调度到不同的服务器上,查询到的 Tablet 有可能是不同版本的基线数据,而这并不会造成数据不一致的问题,如上所述的合并过程,假设 Tablet1 在 ChunkServer1 上合并到了版本 2,而在 ChunkServer2 仍然是版本 1,两个并发的请求分别发送到两个 ChunkServer 上,ChunkServer1 向 UpdateServer 请求 MemTable 版本 3 的数据,与本地版本为 2 的基线数据合并,返回最终结果,ChunkServer2 向 UpdateServer 请求 MemTable 版本 2 的冻结增量和 MemTable 版本 3 的数据,与本地版本为 1 的基线数据合并,返回最终结果,这两种合并的结果都是完全一致的.

3 自动容错与负载均衡

组成 OceanBase 数据库的集群中所使用的服务器均为廉价的 PC 服务器,其可靠性远不如高端服务器和存储.需要指出的是,存储系统中频繁使用的存储介质(机械磁盘)属于损坏概率最高的一类硬件.

OceanBase 采用基线数据多个副本的机制,以避免由于存储或是服务器损坏而导致的数据损失或无法提供服务.在通常情况下,OceanBase 集群中的基线数据会保留三个副本,分别存放在不同服务器上;不管某一个服务器因何原因下线,其持有的数据都会在其他服务器上有相同的副本,从而提供完整、不间断的服务.

3.1 自动重试机制

查询服务器在读事务中查询某个数据分片的数据.如果发现一个数据服务器执行查询失败,会根据这个数据分片的位置信息,重新选取一台数据服务器进行查询,这个过程会检查整个查询过程的超时时间,如果用户给定的超时时间未到,则可以一直重试可用的副本所在的数据服务器一直到成功为止.对于用户来说,该重试机制是透明的.

对于查询失败的 ChunkServer,MergeServer 会在失败一定次数之后将其加入黑名单,加入黑名单中的 ChunkServer 在之后一段时间的查询计划当中,都会被排除在外;直到指定时间过去之后,才重新从黑名单中洗白,开始重新加入查询列表,因为 ChunkServer 可能重新加入集群而提供正常的服务.MergeServer 会在必要时更新 Tablet 的位置信息,以免位置信息因为数据复制和迁移而过时.

当服务器因维护或硬件损坏等因素下线时,无法恢复其保存的数据.换言之,这些下线的服务器中所有存储的副本都会永久失效.RootServer 会侦测到这种情况,并适时发起数据副本的复制操作,以补足默认个数的副本,保证系统正常运行.

得益于 OceanBase 的读写分离设计,只读部分的基线数据复制非常容易,因为数据复制的副本在一个合并周期内是不进行任何修改的,因此数据复制就是对这个只读数据副本的拷贝.在其他分布式系统中[引用],由于是读写并发的,数据副本是可变的,在复制的时候往往会产生新的数据,复制的过程是一个循环追赶的过程.

OceanBase 的副本复制系统,为了提高系统的鲁棒性,可以在集群之间,甚至是不同版本的异构集群之间进行.在进行副本复制时,可以根据复制源和目的数据格式的版本来确定使用逻辑复制或是物理复制.

3.2 负载均衡

OceanBase 良好的扩展性保证集群在运行服务的过程中可以随意地增减服务器,为了在集群伸缩的过程中不影响服务的质量,随时保证系统的各个服务器的负载均衡是非常必要的,OceanBase 的控制节点 RootServer 负责整个集群的负载均衡控制,每个数据节点 ChunkServer 会定期汇报负载信息,比如 Tablet 个数、大小、系统容量等;RootServer 汇总这些信息以后进行计算,得出每个 ChunkServer 应该承载的 Tablet 个数或大小,再根据目标值进行目标间的迁移任务调度,直到整个集群的每个服务器负载基本相当.

4　分裂与合并

OceanBase 的数据分片划分是按照主键(PK)排序以后,根据数据的物理大小进行范围划分为大致相当的 Tablet,一般单个 Tablet 大小为 256MB;随着数据库的不断修改,Tablet 基线数据与增量数据每日合并的过程当中,某些 Tablet 会不断地插入新的数据,变得越来越大,某些 Tablet 可能会不断地删除数据,变得越来越小;一段时间以后,系统中的 Tablet 大小不一,在各个服务器中分布也不均等,会导致热点集中,影响查询性能和负载均衡.

为了使 Tablet 在运行的过程中一直保持大小均等,系统会在每日合并的过程中发起 Tablet 的分裂和合并过程,如果发现每日合并的过程中 Tablet 过大,则自动将 Tablet 分裂为多个标准大小的 Tablet,如果发现每日合并过程中 Tablet 过小,则会在每日合并结束后将相邻的 Tablet 合并.

4.1　数据分片的分裂方法

在其他类似采用主键排序进行数据分片的分布式系统,比如 HBase[引用]当中,进行数据分片的自动分裂往往是一个非常复杂的过程,因为每个数据分片在不断地写入,而且要同步到其他副本;要在多个副本上保持一致性,也就是找到一个一致的分裂点,从而使分裂以后的各个子范围保持一致,是一个很困难的过程. HBase 采用的是一个使用 Master 节点作为仲裁来保证一致的方法,这个分裂的过程是一个很长的分布式事务,一旦涉及到的某个节点出现问题,难于回滚,很可能需要人工介入,因此在实践当中,数据分裂会被人为禁止甚至采用手工的方法进行.

OceanBase 的基线数据是只读的,而每日合并的过程,如前所述,是将某个版本的只读数据与 UpdateServer 中的相邻的冻结版本的数据进行合并,生成一份新版本的只读基线数据的过程,因为每日合并的源(只读的基线数据)在每个 ChunkServer 上的副本完全一致,加上合并的增量(冻结版本的增量数据)在每个 UpdateServer 上也完全一致,而每日合并的算法完全一致,所以只需在每日合并过程中采用完全相同的分裂规则,则可以保证分裂的结果是完全一致的.

OceanBase 不需要仲裁来决定每个需要分裂的 Tablet 的分裂点,也不需要分布式事务来保证分裂的过程,而是完全采用每个副本独立计算分裂点的方法. 为了保证分裂的大小均等并且不会分裂出过小的数据分片,我们设定了一个分裂溢出比例,假设 Tablet 标准大小为 S,分裂溢出比例为 p, 那么在每日合并过程当中,如果生成的数据大小超过了 S,我们记录一个分裂点,并且如果余下生成的数据大小超过了($S*p$),将前一个分裂点确定为一个实际的分裂点,如果余下生成的数据大小没有超过($S*p$),那么不进行分裂,将整个数据作为一个 Tablet,如果生成数据需要多次分裂,以此类推,每超过 S 的数据分片,都计算余下的数据是否构成一次真正的分裂. 因此,这个分裂的结果保证生成的 Tablet 最大不会超过

$S*(1+p)$,而最小不会小于 $S*p$;实际过程,我们一般选用 p 为 0.5.

4.2 数据分片的合并方法

如果每日合并的过程中,某些 Tablet 因为数据删除变得过小,就会触发 Tablet 合并过程,合并过程是由控制节点 RootServer 发起的,在每日合并完成之后,RootServer 会扫描数据分片的元数据集合 RootTable,如果发现范围上连续的 Tablet 都小于标准大小,满足合并条件,比如大小累加以后接近标准大小,则触发合并,在扫描整个 RootTable 后,RootServer 会建立一个合并计划,首先将要合并的相邻的 Tablets 中的每一个小的 Tablet 的副本都迁移到相同的 ChunkServer 上,然后对每个 ChunkServer 推送合并命令,ChunkServer 接收到命令之后,在本地将小的 Tablet(这个时候它们的实际分片数据都应该已经迁移到了本地)合并为一个大的 Tablet,最后将合并后的结果汇报,RootServer 会对比推送的命令,如果结果一致,则接收汇报结果并修改元数据.

5 分块存储格式

5.1 为何引入分块存储格式

如前所述,数据的每日合并过程是将某个版本只读基线数据与冻结的增量数据进行合并的过程,而这个合并的过程需要将两份数据全部读取到内存当中,逐行的进行比对,进行归并排序,并将主键相同的行进行数据合并的过程,这个过程至少需要读取旧版本的基线数据,加上冻结的增量数据,并且写入新的版本基线数据.这个过程必然对系统的磁盘读写 IO 产生巨大的挑战.

OceanBase 面向的应用通常都是数据总量(只读基线数据)非常大,但一段时间(例如一天)内增删改的数据总量(增量数据)通常不大,通过对线上已经运行的数据库实例的数据分析,我们发现,将 Tablet 的尺寸划分为不同的大小,其被修改的几率也不同,具体分析数据结果如表 1 所示.

表 1 Tablet 的尺寸划分

Tab. 1 The size of Tablet

块大小	1	2	4	8
修改比率(百分比)	5.4	7.3	11.7	16.3

为了避免每日合并的时候将所有的基线数据全部读一遍,并且合并以后重复写入磁盘,将 Tablet 划分的更小,并通过判断是否有增量的数据来进行合并的方式,绝大部分没有修改的基线数据不参与到合并过程当中,只是对必要的被修改过的 Tablet 进行实际的合并操作,将原来的每日合并的实际运算量减少到合理水平.

5.2 分块存储格式结构

如果直接将 Tablet 标准大小设置为 2 MB,其个数会变为原来的 128 倍.在某些应用当中,由于 Tablet 个数已经很多,这种膨胀比率会使得 RootServer 在管理元数据 RootTable 的时候付出极大的代价,而在一个庞大的 RootTable 中查询定位到具体的 Tablet 也会耗费更多的时间,因此 OceanBase 的存储结构并没有直接将 Tablet 大小缩减,而是引入了我们称为分块存储结构的全新格式.

保持 Tablet 的标准大小不变,而 Tablet 内部划分为固定大小的数据块,我们称这种数据块为“宏块”,宏块的大小一般是 2 MB,这也是每日合并的基本单位.而宏块又划分为微

块，宏块中存放微块的索引用于查询，由于引入了宏块这个层次，整个系统的存储结构是 Tablet，宏块，以及微块三层组成. 引入宏块层次以后，由于宏块本身需要元数据，而这种元数据本身是存储在 ChunkServer 上，如果以每个磁盘 2 TB 的数据来算，用 2 MB 作为宏块大小，一共是 1 M 个宏块，按每个宏块需要约 100 byte 大小的元数据，总共需要耗费 100 MB 大小的内存，这在 ChunkServer 上可以接受的.

ChunkServer 上的每个磁盘中，都会初始化一个数据文件，数据文件会被划分为固定大小的宏块，宏块是存放数据的基本单元，不仅 Tablet 的实际数据，包括 Tablet 元数据，以及宏块元数据本身也会存放在宏块当中（如图 5 所示）. Tablet 元数据除了 Tablet 本身所在的范围外，最主要的部分就是包含组成其自身的宏块的索引.

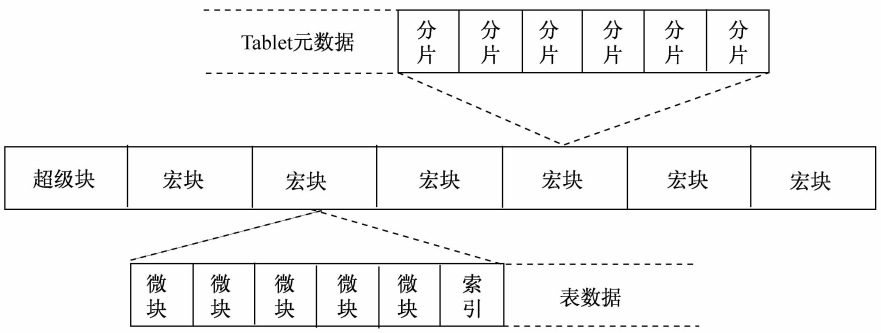

图 5　分块 SSTable 格式

Fig. 5　Block SSTable format

5.3　分块存储的每日合并过程

ChunkServer 在进行对每个 Tablet 进行每日合并操作时，会遍历这个 Tablet 元数据中的宏块索引，根据宏块索引编号引用的宏块，计算其主键范围，通过查询 UpdateServer 判断宏块对应的范围是否有增量数据，如果有增量，则对该宏块进行每日合并，在数据文件中分配一个新的宏块位置，将合并生成的数据写入新分配的宏块当中，并将新宏块的索引信息加入新版本的 Tablet 元信息；如果没有增量数据，则直接将宏块的索引信息加入新的 Tablet 元信息.

如图 6 所示，旧版本的 Tablet 索引了 1—6 这 6 个宏块，合并过程中，宏块 2 有增量数据，其他的宏块均没有变化，宏块 2 进行合并以后，新生成的数据写入了宏块 7，新版本的 Tablet 索引了 1，3—7 这几个宏块，完成了这个 Tablet 的合并过程.

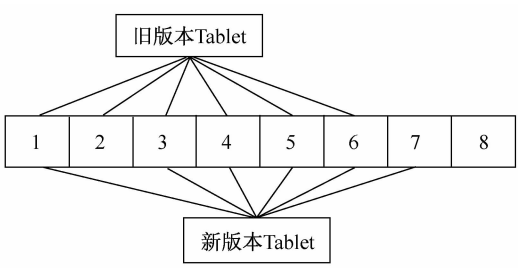

图 6　分块 SStable 每日合并

Fig. 6　Daily merge on block SStable

6 小 结

OceanBase 作为一种分布式关系型数据库,区别于传统关系型数据库,在存储结构上采用分布式 B^+ tree 结构.为了实现可扩展和高可用的系统架构,OceanBase 还设计了多副本冗余、自动容错以及负载均衡等方法来保证系统在各种情形下仍能持续提供高水准的服务.本文描述了 OceanBase 的主要存储结构,数据格式以及实现每日合并的技术细节.

[参 考 文 献]

[1] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system[C]//Proc of the 19th ACM SOSP, 2003: 29-43.

[2] CHANG F, DEAN J, GHEMAWAT S, et al. Bigtable: A distributed storage system for structured data[C]//OSDI, 2006.

[3] CORBETT J C, DEAN J, EPSTEIN M, et al. Spanner: Google's globally-distributed database[C]//OSDI, 2012.

[4] Oracle Database[EB/OL]. http://www.oracle.com/index.html, 2014.

[5] DB2 Database[EB/OL]. http://www-01.ibm.com/software/data/db2/, 2014.

[6] MySQL[EB/OL]. http://www.mysql.com, 2014.

[7] VoltDB[EB/OL]. http://voltdb.com/, 2014.

[8] MemSQL[EB/OL]. http://www.memsql.com/, 2014.

[9] Apache Hbase Region Splitting and Merging[EB/OL]. http://zh.hortonworks.com/blog/apache-hbase-region-splitting-and-merging/.

(责任编辑 林 磊)