**读BigTable论文报告**

18301143余福

一、什么是BigTable

Bigtable是一个分布式的结构化数据存储系统，它被设计用来处理海量数据：通常是分布在数千台普通服 务器上的PB级的数据。Google的很多项目使用Bigtable存储数据，包括Web索引、Google Earth、 Google Finance。这些应用对Bigtable提出的要求差异非常大，无论是在数据量上（从URL到网页到卫 星图像）还是在响应速度上（从后端的批量处理到实时数据服务）。尽管应用需求差异很大，但是，针对 Google的这些产品，Bigtable还是成功的提供了一个灵活的、高性能的解决方案。

在很多方面，Bigtable很像一个数据库：它实现了很多数据库的策略。并行数据库和内存数据库已经实现了可扩展和高性能，但是Bigtable与这些系统相比提供了不同的接口。Bigtable不支持全关系型的数据模型，BigTable为客户端提供了一种简单的数据模型，客户端可以动态地控制数据的布局和格式，并且利用底层数据存储的局部性特征。Bigtable将数据统统看成无意义的字节串，客户端需要将结构化和非结构化数据串行化再存入Bigtable。

二、数据模型

论文中对BigTable是这样描述的：Bigtable是一个稀疏的、分布式的、持久化存储的多维度排序Map 。接下来对其关键字进行解释：

1、Map

这个东东想必大家都会了解，无非就是一个key-value对，Bigtable的键有三维，分别是行键（row key）、列键（column key）和时间戳（timestamp），行键和列键都是字节串，时间戳是64位整型；而值是一个字节串。可以用 (row:string, column:string, time:int64)→string 来表示一条键值对记录。

2. 持久化

Persistence（持久性）只是表示当你的程序结束或数据入口关闭后，你保存在map中的数据会被持久化，这个和其他的持久化存储方式没区别。

3. 分布式

BigTable构建在分布式文件系统上，以便底层文件存储可以在独立机器阵列之间传播。BigTable可以运行在Google File System上。数据以类似的方式在多个参与节点中复制，以便数据在基于独立冗余磁盘阵列的系统中的光盘之间进行条带化。

4. 有序

与大多数Map实现不同，在BigTable中，键值对以严格的字母顺序保存。 也就是说，键“aaaaa”的存储行应该在键“aaaab”的旁边，并且距离具有键“zzzzz”的存储行非常远。回到我们的map示例，排序后的版本如下：

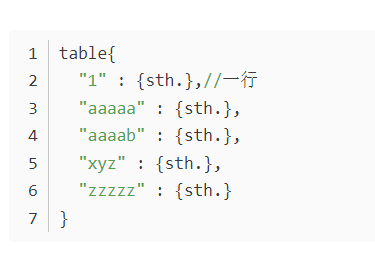


因为这些系统往往是特别庞大的，并且数据是分布式存储，所以这种排序功能其实非常重要。具有类似键的行的空间优势确保了当一个操作必须扫描表格时，查询最感兴趣的内容将彼此靠近。需要注意的是，在HBase和BigTable中的value是不被排序的，只有key被排序。除了这个，其他的和普通的map实现是一样的。

5.多维度

Bigtable不是关系型数据库，但是却沿用了很多关系型数据库的术语，像table（表）、row（行）、column（列）等。这容易让读者误入歧途，将其与关系型数据库的概念对应起来，从而难以理解论文。但如果跳脱出关系型数据库的思想，而把BigTable只是想成一个简单的三维表，那么就可以解释得通了。

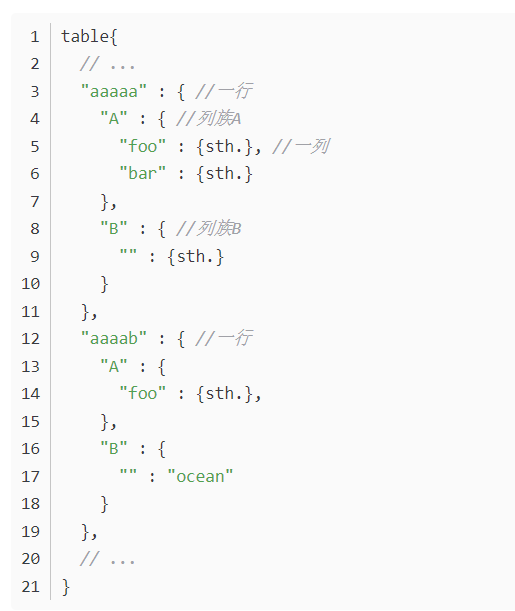
行键可以是任意字节串，通常有10-100字节。行的读写都是原子性的。Bigtable按照行键的字典序存储数据。Bigtable的表会根据行键自动划分为片（tablet），片是负载均衡的单元。最初表都只有一个片，但随着表不断增大，片会自动分裂，片的大小控制在100-200MB。行是表的第一级索引，我们可以把该行的列、时间和值看成一个整体，简化为一维键值映射，类似于：



列是第二级索引，每行拥有的列是不受限制的，可以随时增加减少。为了方便管理，列被分为多个列族（column family，是访问控制的单元），一个列族里的列一般存储相同类型的数据。一行的列族很少变化，但是列族里的列可以随意添加删除。列键按照family:qualifier格式命名的。这次我们将列拿出来，将时间和值看成一个整体，简化为二维键值映射，类似于：



或者可以将列族当作一层新的索引，类似于：



时间戳是第三级索引。Bigtable允许保存数据的多个版本，版本区分的依据就是时间戳。时间戳可以由Bigtable赋值，代表数据进入Bigtable的准确时间，也可以由客户端赋值。数据的不同版本按照时间戳降序存储，因此先读到的是最新版本的数据。我们加入时间戳后，就得到了Bigtable的完整数据模型，类似于：



查询时，如果只给出行列，那么返回的是最新版本的数据；如果给出了行列时间戳，那么返回的是时间小于或等于时间戳的数据。比如，我们查询"aaaaa"/"A:foo"，返回的值是"y"；查询"aaaaa"/"A:foo"/10，返回的结果就是"m"；查询"aaaaa"/"A:foo"/2，返回的结果是空。

三.API

Bigtable提供了建立和删除表以及列族的API函数。Bigtable还提供了修改集群、表和列族的元数据的 API，比如修改访问权限。：写入或者删除Bigtable中的值、从每个行中查找值、或者遍历 表中的一个数据子集。

Bigtable还支持一些其它的特性，利用这些特性，用户可以对数据进行更复杂的处理。首先，Bigtable支 持单行上的事务处理，利用这个功能，用户可以对存储在一个行关键字下的数据进行原子性的读-更新-写 操作。虽然Bigtable提供了一个允许用户跨行批量写入数据的接口，但是，Bigtable目前还不支持通用的 跨行事务处理。其次，Bigtable允许把数据项用做整数计数器。最后，Bigtable允许用户在服务器的地址 空间内执行脚本程序。脚本程序使用Google开发的Sawzall数据处理语言。

四.BigTable构件

Bigtable是建立在其它的几个Google基础构件上的。BigTable使用Google的分布式文件系统(GFS) 存储日志文件和数据文件。BigTable集群通常运行在一个共享的机器池中，池中的机器还会运行其 它的各种各样的分布式应用程序，BigTable的进程经常要和其它应用的进程共享机器。BigTable依赖集群 管理系统来调度任务、管理共享的机器上的资源、处理机器的故障、以及监视机器的状态。

Google的SSTable文件格式用于内部存储Bigtable数据。SSTable提供了一个持久的、排序不变的key-value映射，其中key和value都可以是任意字符串。提供了根据一个指定的key查找value的操作，以及遍历指定key的一个范围内所有的key-value对。内部地，每个SSTable包含了一系列的块（一般情况下，每个块的大小为64KB，但是这是可配置的）。一个块索引（存储在SSTable的末尾）用来定位块，这个索引会在SSTable打开时载入到内存中。一个查询可以执行一次独立的磁盘查询：我们首先通过在内存索引中的二叉查找找到适当的块，然后从磁盘中将适当的块读取出来。同样地，一个SSTable能够完整的被映射到内存中，这样可以让我们在不接触磁盘的情况下来执行查询和浏览。

Bigtable还依赖一个高可用、序列化的分布式锁服务组件，叫做Chubby。一个Chubby服务包括五个活动的副本，其中一个副本被选为master，并且处理请求。只有在大多数副本都是正常运行，并且彼此之间能够互相通信的情况下，Chubby服务才是可用的。当有副本失效的时候，Chubby使用Paxos算法来保证副本的一致性。Chubby提供了一个名字空间，包含目录和小文件。每个目录和小文件都可以当成一个锁，读写文件的操作都是原子的。Chubby客户程序提供对Chubby文件的一致性缓存。每个Chubby客户程序都维护一个与Chubby服务的会话。如果客户程序不能在租约到期的时间内重新签订会话的租约，这个会话就过期失效了。当一个会话失效时，它所拥有的锁和打开的文件句柄都失效了。Chubby客户程序可以在文件和目录上注册回调函数，当文件或目录改变、或者会话过期时，回调函数会通知客户程序。

Bigtable使用Chubby函数完成以下的几个任务：确保在任何给定的时间内最多只有一个活动的Master副本；存储Bigtable数据的自引导指令的位置；查找Tablet服务器，以及在Tablet服务器失效时进行善后；存储Bigtable的模式信息（每张表的列族信息）；以及存储访问控制列表。如果Chubby长时间无法访问，Bigtable就会失效。

五.工作原理

1.组成

Bigtable包括了三个主要的组件：链接到客户程序中的库、一个Master服务器和多个Tablet服务器。针对系统工作负载的变化情况，BigTable可以动态的向集群中添加（或者删除）Tablet服务器。

Master服务器主要负责以下工作：为Tablet服务器分配Tablets、检测新加入的或者过期失效的Table服务 器、对Tablet服务器进行负载均衡、以及对保存在GFS上的文件进行垃圾收集。除此之外，它还处理对模 式的相关修改操作，例如建立表和列族。

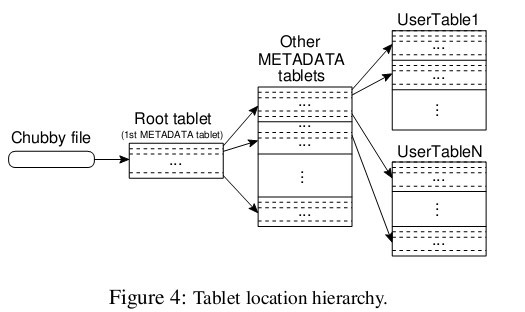
每个Tablet服务器都管理一个Tablet的集合（通常每个服务器有大约数十个至上千个Tablet）。每个 Tablet服务器负责处理它所加载的Tablet的读写操作，以及在Tablets过大时，对其进行分割。

和很多Single-Master类型的分布式存储系统类似，客户端读取的数据都不经过Master服务 器：客户程序直接和Tablet服务器通信进行读写操作。由于BigTable的客户程序不必通过Master服务器来 获取Tablet的位置信息，因此，大多数客户程序甚至完全不需要和Master服务器通信。在实际应用 中，Master服务器的负载是很轻的。

一个BigTable集群存储了很多表，每个表包含了一个Tablet的集合，而每个Tablet包含了某个范围内的行 的所有相关数据。初始状态下，一个表只有一个Tablet。随着表中数据的增长，它被自动分割成多个 Tablet，缺省情况下，每个Tablet的尺寸大约是100MB到200MB。

2.Tablet的存储结构

Bigtable使用一个类似B+树的数据结构存储片的位置信息。



首先是第一层，Chubby file。这一层是一个Chubby文件，它保存着root tablet的位置。这个Chubby文件属于Chubby服务的一部分，一旦Chubby不可用，就意味着丢失了root tablet的位置，整个Bigtable也就不可用了。

第二层是root tablet。root tablet其实是元数据表（METADATA table）的第一个分片，它保存着元数据表其它片的位置。root tablet很特别，为了保证树的深度不变，root tablet从不分裂。

第三层是其它的元数据片，它们和root tablet一起组成完整的元数据表。每个元数据片都包含了许多用户片的位置信息。

可以看出整个定位系统其实只是两部分，一个Chubby文件，一个元数据表。注意元数据表虽然特殊，但也仍然服从前文的数据模型，每个分片也都是由专门的片服务器负责，这就是不需要主服务器提供位置信息的原因。客户端会缓存片的位置信息，如果在缓存里找不到一个片的位置信息，就需要查找这个三层结构了，包括访问一次Chubby服务，访问两次片服务器。

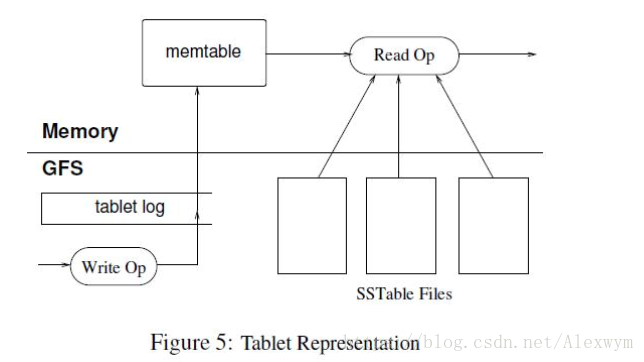
3.Tablet的分配方式

在任何一个时刻，一个Tablet只能分配给一个Tablet服务器。Master服务器记录了当前有哪些活跃的 Tablet服务器、哪些Tablet分配给了哪些Tablet服务器、哪些Tablet还没有被分配。当一个Tablet还没有被 分配、并且刚好有一个Tablet服务器有足够的空闲空间装载该Tablet时，Master服务器会给这个Tablet服 务器发送一个装载请求，把Tablet分配给这个服务器。 BigTable使用Chubby跟踪记录Tablet服务器的状态。当一个Tablet服务器启动时，它在Chubby的一个 指定目录下建立一个有唯一性名字的文件，并且获取该文件的独占锁。Master服务器实时监控着这个目录 （服务器目录），因此Master服务器能够知道有新的Tablet服务器加入了。如果Tablet服务器丢失了 Chubby上的独占锁 — 比如由于网络断开导致Tablet服务器和Chubby的会话丢失 — 它就停止对Tablet 提供服务。（Chubby提供了一种高效的机制，利用这种机制，Tablet服务器能够在不增加网络负担的情 况下知道它是否还持有锁）。只要文件还存在，Tablet服务器就会试图重新获得对该文件的独占锁；如果 文件不存在了，那么Tablet服务器就不能再提供服务了，它会自行退出 。

当 Tablet服务器终止时（比如，集群的管理系统将运行该Tablet服务器的主机从集群中移除），它会尝试释放它持有的文件锁，这样一来，Master服务器就能尽快把Tablet分配到其它的Tablet服务器。 Master服务器负责检查一个Tablet服务器是否已经不再为它的Tablet提供服务了，并且要尽快重新分配它 加载的Tablet。Master服务器通过轮询Tablet服务器文件锁的状态来检测何时Tablet服务器不再为Tablet 提供服务。如果一个Tablet服务器报告它丢失了文件锁，或者Master服务器最近几次尝试和它通信都没有 得到响应，Master服务器就会尝试获取该Tablet服务器文件的独占锁；如果Master服务器成功获取了独占锁，那么就说明Chubby是正常运行的，而Tablet服务器要么是宕机了、要么是不能和Chubby通信了， 因此，Master服务器就删除该Tablet服务器在Chubby上的服务器文件以确保它不再给Tablet提供服务。 一旦Tablet服务器在Chubby上的服务器文件被删除了，Master服务器就把之前分配给它的所有的Tablet 放入未分配的Tablet集合中。为了确保Bigtable集群在Master服务器和Chubby之间网络出现故障的时候 仍然可以使用，Master服务器在它的Chubby会话过期后主动退出。但是不管怎样，如同我们前面所描述 的，Master服务器的故障不会改变现有Tablet在Tablet服务器上的分配状态。 当集群管理系统启动了一个Master服务器之后，Master服务器首先要了解当前Tablet的分配状态，之后才能够修改分配状态。

Master服务器在启动的时候执行以下步骤：（1）Master服务器从Chubby获取一个 唯一的Master锁，用来阻止创建其它的Master服务器实例；（2）Master服务器扫描Chubby的服务器文 件锁存储目录，获取当前正在运行的服务器列表；（3）Master服务器和所有的正在运行的Tablet表服务 器通信，获取每个Tablet服务器上Tablet的分配信息；（4）Master服务器扫描METADATA表获取所有的 Tablet的集合。在扫描的过程中，当Master服务器发现了一个还没有分配的Tablet，Master服务器就将这 个Tablet加入未分配的Tablet集合等待合适的时机分配。 可能会遇到一种复杂的情况：在METADATA表的Tablet还没有被分配之前是不能够扫描它的。因此，在开 始扫描之前（步骤4），如果在第三步的扫描过程中发现Root Tablet还没有分配，Master服务器就把Root Tablet加入到未分配的Tablet集合。这个附加操作确保了Root Tablet会被分配。由于Root Tablet包括了 所有METADATA的Tablet的名字，因此Master服务器扫描完Root Tablet以后，就得到了所有的METADATA表的Tablet的名字了。 保存现有Tablet的集合只有在以下事件发生时才会改变：建立了一个新表或者删除了一个旧表、两个 Tablet被合并了、或者一个Tablet被分割成两个小的Tablet。Master服务器可以跟踪记录所有这些事件， 因为除了最后一个事件外的两个事件都是由它启动的。Tablet分割事件需要特殊处理，因为它是由Tablet 服务器启动。在分割操作完成之后，Tablet服务器通过在METADATA表中记录新的Tablet的信息来提交这 个操作；当分割操作提交之后，Tablet服务器会通知Master服务器。如果分割操作已提交的信息没有通知 到Master服务器（可能两个服务器中有一个宕机了），Master服务器在要求Tablet服务器装载已经被分割 的子表的时候会发现一个新的Tablet。通过对比METADATA表中Tablet的信息，Tablet服务器会发现 Master服务器要求其装载的Tablet并不完整，因此，Tablet服务器会重新向Master服务器发送通知信息。

4.Tablet服务



如图，Tablet的持久化状态信息保存在GFS上。更新操作提交到REDO日志中。在这些更新操作中，最近提交的那些存放在一个排序的缓存中，我们称这个缓存为memtable；较早的更新存放在一系列SSTable中。为了恢复一个Tablet，Tablet服务器首先从METADATA表中读取它的元数据。Tablet的元数据包含了组成这个Tablet的SSTable的列表，以及一系列的Redo Point，这些Redo Point指向可能含有该Tablet数据的已提交的日志记录。Tablet服务器把SSTable的索引读进内存，之后通过重复Redo point之后提交的更新来重建memtable。

当对Tablet服务器进行写操作时，Tablet服务器首先要检查这个操作格式是否正确、操作发起者是否有执行这个操作的权限。权限验证的方法是通过一个从Chubby文件里读取出来的具有写权限的操作者列表来进行验证（这个文件几乎一定会存放在Chubby客户缓存里）。成功的修改操作会记录在提交日志里。可以采用批量提交的方式来提高大量小的修改操作的应用程序的吞吐量。当一个写操作提交后，写的内容插入到memtable里面。

当对Tablet服务器进行读操作时，Tablet服务器会作类似的完整性和权限检查。一个有效的读操作在一个由一系列SSTable和memtable合并的视图里执行。由于SSTable和memtable是按字典排序的数据结构，因此可以高效生成合并视图。

当进行Tablet的合并和分割时，正在进行的读写操作能够继续进行。

5. 相关应用编辑

HBASE

HBase是Apache下Hadoop的存储系统，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统 [1] 。HBase在Hadoop之上提供了Bigtable的功能，HBase已经成为Apache开源项目的一个顶级项目 [1] 。

Hypertable

Hypertable是Bigtable的开源实现，由Zvents公司使用Boost C++进行开发，并于2008年3月公布其源码 。Hypertable是一个稀疏、分布式、高性能的多维有序映射表，用于处理海量结构化和半结构化数据。Hypertable可以让用户通过主键来组织海量数据，并为用户提供了数据查询语言HQL以实现高效查询。