# BigTable读后感

18301172 谢世浩

花了一天的时间粗略地看了英文版的BigTable的大部分章节之后（除了7、8章）之后，我对BigTable有了一个大致的了解，现在我来阐述我对BigTable的个人认识和理解。

以下是BigTable 的一些数据模型（DATA MODEL）：

BigTable采用分布式多维稀疏映射的方式来查询和存储数据。以下是映射的规则。

（row:string,column:string,time:int64）->string

其中映射的字符串最多可以到64K。相比关系型数据库的储存方式来说，BigTable的每个值都需要用键值来映射，看起来是非常浪费储存空间的，但是由于BigTable储存的数据比较稀疏，表里的一行的数据可能只有一两个字段储存信息，所以实际上储存相同的数据，BigTable所用空间可能比关系型数据库还要少；此外，这种储存方式也比关系型数据库的储存方式灵活，不需要读取完整张表或整行，只需要通过映射就可以抽取其中的值。

在BigTable中，行（row）的定义与关系型数据库（RDBMS）的定义一致，但是有一个新的概念：tablet。tablet在BigTable里的定义是按字母顺序排列的动态变化的行范围，即动态地将一张表分块，其中的一块就是tablet。Tablet是分布式和负载均衡的单元。从Google给的例子来看，tablet大概是一个行的集合，BigTable将有某种联系的行划分在一起，放在tablet里。而tablet让用户定位数据入口更加高效。

在BigTable中，列（column）的定义与RDBMS的定义一致，但是BigTable比RDBMS多了列族（column families）的概念。列族的定义是一些列值的集合，列族里的列的数据类型一致。在我的理解里，列族和列的关系大概与父类和子类相似，举列族和列的一个例子：银行卡号是一个列族，而中行卡号、农行卡号和建行卡号等就是列族里的一个列。列的命名方式是：family:qualifier。多个列族还可以加入本地组（locality group），每一个本地组在每一个tablet中都会生成一个独立的SSTable，在读取时，不会读取除了指定列族以外的其他列族，与RDBMS相比，提高了读取效率。

在BigTable中，时间戳（timestamp）的概念也与RDBMS的定义一致。BigTable可以储存多个版本的数据。用户可以自定义时间戳和保存的版本数。

BigTable使用了几个谷歌的系统：谷歌文件系统（Google File System , GFS）、SSTable和Chubby。以下是关于这些系统的简单的理解。

GFS用来进行储存日志和数据文件操作，是一个分布式文件系统，文中没用详细介绍。

SSTable是一种存储映射键值的、排好序的文件存储形式，在写入磁盘后就无法对信息进行更改。此外，SSTable提供特殊的索引，使系统可以仅读取索引的情况下就判断信息是否在次SSTable文件里。在找到数据所在的SSTable后，就将整个SSTable读入内存中。

Chubby是一个高度可用的、持续的分布式文件锁服务，应该是用来处理数据同步问题。

最后是我关于这个BigTable的实现的理解：

首先是tablet的定位，BigTable用三层的B+树来存储和查询tablet，树的第一层是Chubby文件，Chubby文件储存METADATA文件的位置，METADATA文件的第一个tablet是root tablet，root tablet储存其他在METADATA的tablet的位置，而METADATA的其他tablet储存的是用户tablet的位置，即储存用户tablet的SSTable的地址，这个系统最多可以定位2^34个tablet。由于SSTable有特殊的索引，所以tablet的定位应该非常块且占用内存不多。

其次是tablet的分配。每一个tablet都会且仅会分配给一个tablet server，tablet server由master server创建和分配任务，且在创建之后会从Chubby中获取一个文件夹的文件锁，tablet server将SSTable文件写入该文件夹中，如果tablet server失去文件锁，则会被master server终止。而master server在与Chubby的会话连接过期后会自我终止，然后在master server重启时，master server会从Chabby获取master锁，然后读取锁的文件夹里的文件，重新建立与其他tablet server的连接，最后扫描METADATA表获取所有的tablet并更新tablet 的状态。

然后是tablet的服务。当一条数据写入时，数据会写入日志，插入memtable里，并在一个SSTable文件里写入redo point。在恢复一条tablet时，会读取日志和SSTable文件来更新，然后更新memtable。在读取数据时，BigTable会合并SSTable和memtable后再操作。

最后是压缩tablet。当memtable里的数据到达阈值时，会新建一个新的memtable，写入和读取操作不会受到影响。而旧的memtable会异步地在后台操作，把memtable里的数据和相关的tablet写入新的SSTable文件，写入之后memtable和相关的SSTable都可以删除，这种压缩叫minor compaction。另一种压缩操作叫major compaction，即将所有的SSTable文件写入一个SSTable文件。

以上就是我对BigTable的全部认识和理解，从这篇文章中，我得知了许多关于非关系型数据库的基础概念，并有机会窥探非关系型数据库的原理，实在是收益匪浅，非常感想老师的引导。