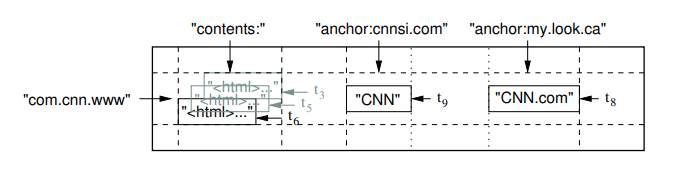
**BigTable读后感**

**18301103 刘镠**

**一、BigTable数据模型**

BigTable是一个稀疏的、分布式的、持久化存储的多维度排序Map。在BigTable中，最小的存储单元是数据项，通过（行，列，时间戳）三元组可以定位一个数据项。每一个数据项都是一个未经解析的byte数组。其中行关键字是任意的字符串，而BigTable通过行关键字的字典顺序组织数据。行可以动态分区，每个分区叫做一个”Tablet”，Tablet是数据分布和负载均衡调整的最小单位。对同一个行关键字的读和写都是原子的。

许多个列组成一个“列族“，列族是方坤控制的基本单位。每一个列族下可以有很多个列，通常存放在同一列族下的所有数据都是一个数据类型的，每一个列用“列族：限定词”的方式来命名。一张表中的列族不能太多，最多几百个，列族在运行期间很少改变。但是一张表可以有无限多个列。访问控制、磁盘和内存使用统计都是在列族的层面上进行的。

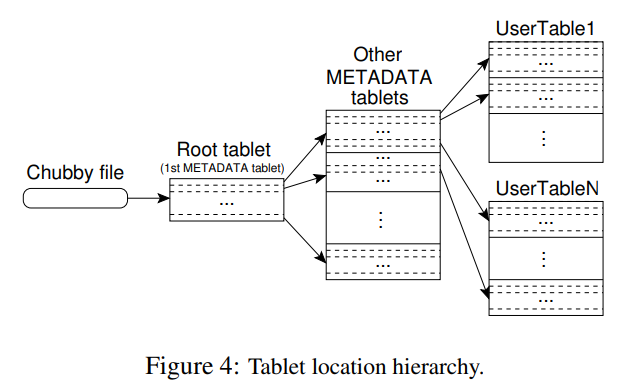
表的每一个数据项可以包含一个数据的不同版本，这就需要使用时间戳进行区分。在数据项中，不同版本的数据按照时间戳进行排序，最新的数据排在最前面。

以上表为例，我们使用一张表WebTable存储海量的网页及相关信息，其中行关键字为反向索引的URL，如“com.cnn.www”。每一个列族代表了网页的一种属性，如anchor列族存放了引用该网页的锚链接文本，其中有两列“cnnsi.com”，“my.look.ca”。在“com.cnn.www”这一行和“contents”这一列中，存放了三个版本的html文件，分别用不同的时间戳t3，t5，t6表示，最新的版本放在最上面。

**二、BigTable构件**

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。BigTable可以动态的向集群中添加或删除Tablet服务器以均衡系统工作的负载状况。

Master服务器主要负责为Tablet服务器分配Tablets、控制Tablets服务器的状态即监控有没有新加入或者失效的Table服务器、对Tablet服务器进行负载均衡、管理表和列族。每个Tablet服务器都管理一个Tablet的集合，负责Tablet的读写操作。一个Tablet只能分配给一个Tablet服务器。客户端读取数据是直接和Tablet服务器进行交互，而不经过Master服务器。

**三、存储结构**

BigTable使用三层类似B+树的结构存储Tablet的位置信息。第一层是存储在Chubby File中的Root Tablet的位置，Root Tablet中一个特殊的METADATA存储了所有Tablet的位置信息，每个Tablet位置信息都存放在一个行关键字下。客户端使用的库会缓存Tablet的位置信息，当客户端需要访问某一个Tablet时发现地址信息为空或者不正确，需要在存储结构中进行搜索。为了减少搜索次数，每次读取Tablet的数据时，都会多读取几个Tablet。

**四、Tablet分配机制**

Master服务器为了跟踪所有Tablet服务器的状态，采取了一种机制。当一个Tablet服务器启动时，在Chubby的一个目录下建立一个有唯一性名字的文件并获取独占锁。当Tablet服务器丢失了独占锁，会尝试重新获取，如果文件不存在了，Tablet服务器会自行退出，并释放他持有的锁。Master服务器监控这个目录，通过检查Tablet服务器文件锁的状态来判断Tablet服务器的状态。如果Tablet服务器丢失了文件锁或Master服务器与其联系没有响应，Master服务器就知道Tablet服务器发生了故障，会获取该Tablet服务器上的独占锁，并将分配给这个Tablet服务器的Tablet放入未分配的Tablet集合中。

为保证Tablet的持久性，所有的更新操作都提交到REDO日志中，其中较早的更新操作存放在一系列SSTable中，最近的更新操作存放在memtable缓存中。随着写操作的执行，memtable的大小不断增加，当到达一定大小之后会被冻结，然后创建一个新的memtable，这个memtable会转化为一个SSTable。当需要恢复一个Tablet的时候，从之日中读取该Tablet的所有记录重新构建Tablet。

**五、BigTable优化**

以上就是BigTable的基本实现，在应用过程中用户可以进行优化。如将多个列族组合成一个局部性群族，将通常不会一起访问的列族分割成不同的局部性群组可以提高读取操作的效率；客户端可以控制每一个局部性群组的SSTable是否需要压缩以及用什么格式压缩，这样分块压缩的方式使得只读取SSTable一小部分数据时不需要解压整个文件；使用Bloom过滤器查询一个SSTable是否包含指定行和列的数据，减少磁盘访问次数；为每个Tablet服务器设置一个Commit日志文件，而不是为每一个Tablet设置一个日志文件，并将日志文件按照关键字排序，这样就可以减少磁盘的Seek操作；当Master服务器将Tablet从一个服务器移动到另一个服务器时，会进行两次Compaction以消除归并记录，这样Tablet被装载到新的服务器上之后，就不需要从日志文件中进行恢复。

**六、BigTable性能分析**

单个Tablet服务器下，随机读的性能比其他操作慢一个数量级或以上，因为每个随机读的操作都要通过网络从GFS传输64KB的SSTable到Tablet服务器上，而只是用其中一个值。相比较之下，内存的随机读就快很多，因为直接从本地内存中读数据，不需要从GFS中读取。

和随机读相比较，随机写的性能要好很多，因为每次服务器只是把内容追加到日志末尾，然后再批量提交。

序列读的性能要好于随机读，但是序列写的性能和随机写没有太大的区别。

当Tablet服务器的数量从1增加到500时，系统整体的吞吐量有了大幅度的增长，但是如果仅将服务器的数量从1增加到50时，会发现每台服务器的吞吐量都有明显的下降，这是由于每台服务器的负载不均衡导致的。