## 介绍

Bigtable用于可靠的处理PB级别的数据。Bigtable和数据库很类似，但Bigtable不支持完整的关系数据模型；与之相反，Bigtable为客户提供了简单的数据模型，即Bigtable是需要用户自己定义表结构的。

## 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。Map的索引是行关键字、列关键字以及时间戳；Map中的每个value都是一个未经解析的byte数组。

## 行

行是任意的字符串，同一个行关键字的读或者写操作都是原子的。Bigtable通过行关键字的字典顺序来组织数据。表中的每个行都可以动态分区。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。这提高了读取较少行数据的效率。

## 列族

列关键字组成的集合叫做“列族“，列可以有很多，列族不能有太多。列族是访问控制的基本单位。必须先创建再使用，而且在运行期间很少更改。命名语法为**列族：限定词***。*列族的名字必须是可打印的字符串，而限定词的名字可以是任意的字符串。

## 时间戳

Bigtable中，表的每一个数据项都可以包含同一份数据的不同版本。不同版本的数据通过时间戳来索引。Bigtable时间戳的类型是64位整型。Bigtable和用户程序都可以给时间戳赋值。数据项中，数据按时间戳倒序排序，最新的数据排在最前面。

## API函数

Bigtable提供了建立和删除表和列族，修改集群、表和列族的元数据的API函数。客户程序可以对Bigtable：写入或者删除Bigtable中的值、从每个行中查找值、遍历表中的一个数据子集。

Bigtable其他支持的功能：支持单行上的事务处理，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；允许把数据项用做整数计数器；允许用户在服务器的地址空间内执行脚本程序。但是Bigtable目前还不支持通用的跨行事务处理。

## 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内部存储数据的文件是Google SSTable格式的。Bigtable还依赖高可用的、序列化的分布式锁服务组件Chubby，Chubby中的一个副本被选为Master，用于处理请求，可以完成查找Tablet服务器、存储Bigtable的每张表的列族信息等功能。

## Bigtable实现

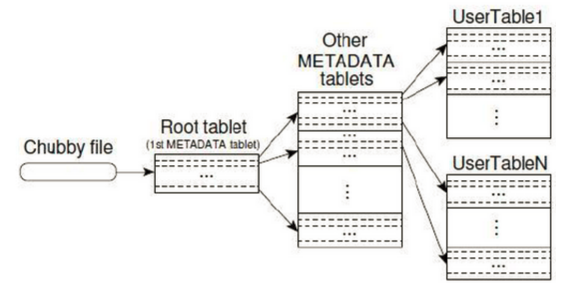
Bigtable包括了三个主要的组件：链接到客户程序中的库、一个Master服务器和多个Tablet服务器。针对系统工作负载的变化情况，Bigtable可以动态的向集群中增删Tablet服务器。

Master服务器主要负责：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、对保存在GFS上的文件进行垃圾收集、处理对模式的相关修改操作。每个Tablet服务器都管理一个Tablet的集合。

一个BigTable集群存储了很多表，每个表包含了一个Tablet的集合。初始状态下，一个表只有一个Tablet。随着表中数据的增长，会自动分割成多个Tablet。

## Tablet的位置

如图，使用一个三层的、类似B+树的结构存储Tablet的位置信息。第一层是一个存储在Chubby中的文件，它包含了Root Tablet的位置信息。第二层代表每个Root Tablet包含一个METADATA表里所有的Tablet的位置信息。第三层表示METADATA表的每个Tablet包含了一个用户Tablet的集合。



## Tablet的分配

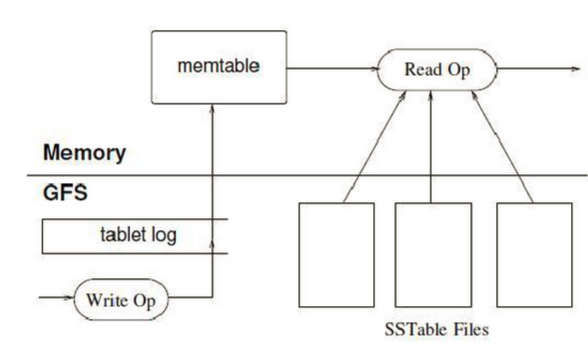
一个Tablet只能分配给一个Tablet服务器。Master服务器记录了当前有哪些已分配的和未被分配的Tablet服务器。当一个Table未被分配、并且刚好有一个Tablet服务器有足够的空闲空间装载该Tablet时，Master服务器会给这个Tablet服务器发送一个装载请求，并分配Tablet给该服务器。

BigTable使用Chubby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。如果独占锁丢失，则停止为这个Tablet提供服务。

Master服务器通过检查一个Tablet服务器的独占锁是否丢失来判断是否停止为Tablet提供服务了，如果已经不提供服务，就重新分配它加载的Tablet。

## Tablet的服务

如下图，Tablet的持久化状态信息保存在GFS上。更新操作会提交到Redo日志。memtable是最近提交的更新操作的缓存。更早的更新操作存放在SSTable里。恢复一个Tablet需要metadata表中它的元数据，包括SSTable和Redo point。Tablet进行读写操作是，要检查格式，并通过Chubby中的操作者列表来验证权限。写的内容会插入memtable，当memtable的尺寸到达限定值的时候，这个memtable就会被冻结，然后创建一个新的memtable。读的操作在一个由一系列SSTable和memtable合并的视图里执行。



## Bigtable优化

**局部性群组**：用户程序可以将多个列族组合成一个局部性群族。对Tablet中的每个局部性群组都会生成一个单独的SSTable。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。

**压缩**：客户程序可以控制一个局部性群组的SSTable是否需要压缩和压缩的格式。常使用“两遍”压缩格式。

**缓存**：为了提高读操作的性能，Tablet服务器使用二级缓存的策略。扫描缓存是第一级缓存，主要缓存Tablet服务器通过SSTable接口获取的Key-Value对；Block缓存是二级缓存，缓存的是从GFS读取的SSTable的Block。

**Bloom过滤器**：通过允许用户程序对特定局部性群组的SSTable指定Bloom过滤器，来减少硬盘访问的次数。比如可以使用Bloom过滤器查询一个SSTable是否包含了特定行和列的数据。

**Commit日志**：设置每个Tablet服务器一个Commit日志文件，把修改操作的日志以追加方式写入同一个日志文件。一个实际的日志文件中混合了对多个Tablet修改的日志记录。

**恢复提速**：当Master服务器将一个Tablet从一个Tablet服务器移到另外一个Tablet服务器时，源Tablet服务器会对这个Tablet做一次Minor Compaction。

**利用不变性**：在使用Bigtable时，除了SSTable缓存之外的其它部分产生的SSTable都是不变的。例如，当从SSTable读取数据的时候，我们不必对文件系统访问操作进行同步。而且，还可以把永久删除被标记为“删除”的数据的问题，转换成对废弃的SSTable进行垃圾收集的问题。