**论文学习笔记：**

Bigtable——分布式的数据化结构存储系统

部署到数千台服务器上来处理TB级数据

实现了适用性广泛，可扩展，高性能和高可用性的目标

Bigtable不支持完整的关系数据模型

Bigtable将存储的数据都视为字符串

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

表中的行关键字可以是任意的字符串

对同一个行关键字的读或者写操作都是原子操作

Bigtable通过行关键字的字典顺序来组织数据

Tablet是数据分布和负载均衡调整的最小单位

列关键字组成的集合叫做“列族”，列族是访问控制的基本单位

列族在使用之前必须先创建，然后才能在列族中任何的列关键字下存放数据

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

在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提供API函数包括建立、删除表以及列族，修改集群、表和列族的元数据

Bigtable支持单行上的事务处理

Bigtable目前还不支持通用的跨行事务处理

Bigtable允许把数据项用做整数计数器

Bigtable允许用户在服务器的地址空间内执行脚本程序

Bigtable使用Google的分布式文件系统存储日志文件和数据文件

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

Bigtable内部存储数据的文件是Google SSTable格式的

SSTable是一个持久化的、排序的、不可更改的Map结构

Map是一个key-value映射的数据结构

SSTable使用块索引来定位数据块

Chubby是一个高可用的、序列化的分布式锁服务组件

Chubby使用Paxos算法来保证副本的一致性

Bigtable的三个主要组件：链接到客户程序中的库

一个Master服务器

多个Tablet服务器

Master服务器主要工作：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

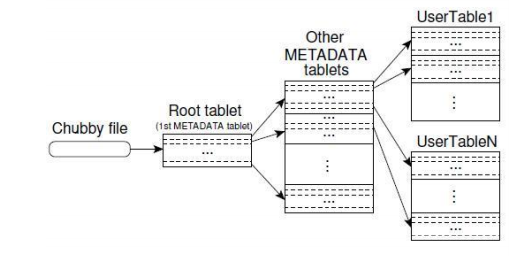
对保存在GFS上的文件进行垃圾收集

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

每个Tablet服务器负责处理它所加载的Tablet的读写操作

Tablet的位置由一个三层的类似B+树的结构存储



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

Root Tablet包含了一个特殊的METADATA表里所有的Tablet的位置信息

METADATA表的每个Tablet包含了一个用户Tablet的集合

一个Tablet只能分配给一个Tablet服务器

Bigtable使用Chubby跟踪记录Tablet服务器的状态

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

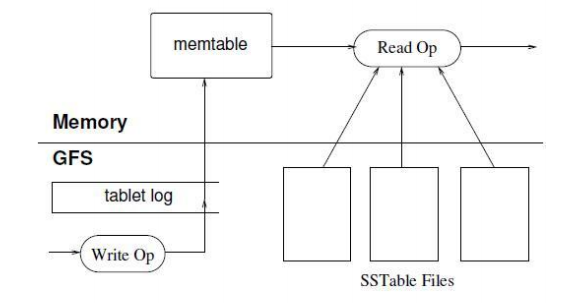
Master服务器在启动的时候执行以下步骤：

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.Masrer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



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

最近提交的更新操作会存放在一个排序的缓存（memtable）中，较早的更新放在SSTable中

恢复Tablet要先从METADATA表中读取元数据

memtable大小随着写操作的执行而增加，当尺寸达到一个门限值时，memtable会被冻结，然后创建一个新的memtable，被冻结的memtable会转换成SSTable，写入GFS

Minor Compaction过程目的：shrink Tablet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

每一次Minor Compaction都会创建一个新的SSTable

合并所有的SSTable并生成一个新的SSTable的Merging Compaction过程叫做Major Compaction

Major Compaction过程生成的SSTable不包含已经删除的信息或数据

Bigtable循环扫描它所有的Tablet，并且定期对它们执行Major Compaction

Major Compaction机制允许Bigtable回收已经删除的数据占有的资源，并且确保Bigtable 能及时清除已经删除的数据

多个列族可以组合成一个局部性群族

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

Tablet服务器依照惰性加载的策略将设定为放入内存的局部性群组的SSTable装载进内存

压缩第一遍采用Bentley and McIlroy’s方式，在一个很大的扫描窗口里对常见的长字符串进行压缩，第二遍采用快速压缩算法

Tablet服务器使用二级缓存的策略来提高读操作的性能

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

一个读操作必须读取构成Tablet状态的所有SSTable的数据

Bloom过滤器可以用来查询一个SSTable是否包含了特定行和列的数据

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可以避免产生大量文件

为了确保在GFS负载高峰时修改操作还能顺利进行，每个Tablet服务器实际上有两个日志写入线程，每个线程都写自己的日志文件，并且在任何时刻，只有一个线程是工作的

Compaction操作减少了Tablet服务器的日志文件中没有归并的记录，从而减少了恢复的时间

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

memtable是唯一一个能被读和写操作同时访问的可变数据结构

对内存表采用COW(Copy-on-write)机制，允许读写操作并行执行，从而减少读操作时的竞争

Master服务器采用“标记-删除”的垃圾回收方式删除SSTable集合中废弃的SSTable

内存中的随机读操作速度快很多

随机和序列写操作的性能比随机读要好些

序列读的性能好于随机读

基准测试的瓶颈是单台Tablet服务器的CPU

性能的提升还不是线性的，Tablet服务器的数量从1台增加到50台时，每台服务器的吞吐量会有一个明显的下降

Google Analytics是用来帮助Web站点的管理员分析他们网站的流量模式的服务，提供整体状况的统计数据和用户使用网站的行为报告

Google Analytics使用的两个表：Row Click表的每一行存放了一个最终用户的会话，Summary表包含了关于每个 Web 站点的、各种类型的预定义汇总信息

系统的整体吞吐量受限于GFS的吞吐量

Google Earth定制的客户端软件可以访问高分辨率的地球表面卫星图像

Imagery表的每一行都代表了一个单独的地理区域

数据预处理流水线高度依赖运行在Bigtable上的MapReduce任务传输数据

个性化查询使用Bigtable存储每个用户的数据

个性化查询的数据会复制到几个Bigtable的集群上，增强了数据可用性，减少由客户端和 Bigtable集群间的“距离”造成的延时