BigTable 学习报告

BigTable 是一个分布式的结构化数据存储系统。其设计的目的是为了高效地处理海量数据并且能够在多台及其上运行。接下来将从其设计的数据结构、系统配置以及性能优化方面来介绍 BigTable 如何实现上述功能。

数据结构：

BigTable 是一个稀疏的、分布式的、持久化存储的多维排序 Map。通常它的 key 由行关键字、列关键字以及时间戳组成。其中行列关键字都是任意的字符串，最大 64KB，对于同一个行关键字的读写都是原子性操作。

获取列定位的方式是“列族：关键字”，列族名是可以打印的字符串，而限定词可以是任意的字符串，一个列族下理论上可以有无限个列，但实际上最多只能有数百来个。

时间戳对应着的是每一项数据项的不同版本，类型是 64 位整型，需要精确到毫秒。

系统配置：

BigTable 使用 Google 的分布式文件系统（GFS）储存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，其为一个持久化的、排序的、不可改变的 Map 结构，其 key 和 value 的值都是任意的 Byte 串。SSTable 使用块索引（通常存储在 SSTable 的最后）来定位数据块。

BigTale 还依赖于一个高可用、序列化分布式锁服务组件叫做 Chubby。一个 Chubby 服务包括 5 个活动的副本，其中一个被选为 Master 处理请求。主要完成的任务是确保任意给定时间内最多只有一个活动的 Master 副本；存储 BigTable 数据自引导指令的位置；查找 Tablet 服务器；存储 BigTable 模式信息；以及存储访问控制列表。

BigTable 包括三个主要的组件：链接到客户程序中的库、一个 Master 服务器和多个 Tablet 服务器。其中 Master 服务器主要是为了调控 Tablet 服务器，以及对其上面的 GFS 的文件进行垃圾收集。每个 Tablet 管理折一个 Tablet 的集合，通常每个服务器大约有数十个至上千个 Tablet。Tablet 服务器负责处理它所加载的 Tablet 的读写操作，以及在 Tablets 过大时对它进行分割。客户端直接和 Tablet 服务器通信进行读写操作，所以实际中 Master 服务器的负载很轻。

性能优化：

客户端程序可以将多个列族组合成一个局部性群族，而其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。

客户端程序可以控制一个局部性群组的 SSTable 是否需要压缩。每个 SSTable 的块都使用用户指定的压缩格式来压缩，虽然分块压缩浪费了少量空间，但是我们在只读取 SSTable 的一小部分数据的时候就不必解压整个文件。

Table 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 Key-Value 对；Block 缓存时二级缓存，缓存的时从 GFS 读取的 SSTable 的 block。

允许客户端程序对特定局部性群族的 SSTable 制定的 Bloom 过滤器来减少硬盘访问的次数。

设置每个 Tablet 服务器给一个 Commit 日志文件，把修改操作的日志一追加的方式写入同一个日志文件，因此一个实际的日志文件中混合了对多个 Tablet 修改的日志记录。

当 Master 服务器将一个 Tablet 从一个 Tbalet 服务器移到另一个 Tablet 服务器时，源 Tablet 服务器会对整个 Tablet 做一次 Minor Compaction 操作，减少了 Tablet 服务器的日志文件中没有归并的记录，减少了恢复的时间。

当从 SSTable 读取数据的时候，一位内除此之外的部分产生的 SSTable 都是不变的，，我们不必对文件系统访问操作进行同步，如此一来可以非常高效实现对行的并行操作。

个人感想：

Bigtable 提供了一种高可用性和可扩展性的技术，列族使得它的可以轻松修改整个表的结构，具有可伸缩等特性，同时可以处理 PB 级的海量数据，