

Bigtable:结构化数据的分布存储系统读后感

18301031 周润佳



2020-12-5

目录

**0.总述3**

**1.** **Bigtable的数据模型3**

**2.API3**

**3.构建模块3**

**4.实现4**

**5.Tablet的定位4**

**6.Tablet的存储和访问4**

**7.改进5**

**8.评估5**

**9.应用5**

**10.总结6**

0.总述

通过论文介绍部分以及查阅相关资料我们得知Bigtable 是设计来分布存储大规模结构化数据的，从设计上它可以扩展到上2^50 字节，分布存储在几千个普通服务器上。在面对不同的要求时，Bigtable都成功地提供了灵活高效的服务。总体来说，Bigtable是一个高性能，高可靠性，可扩展的类似于数据库的结构化数据的分布存储系统，。为什么说它类似于数据库呢，经过查阅它内部原理，我认为是因为Bigtable采用了很多数据库的实现策略，但是它并不支持完整的关系型数据模型，而是为客户端提供了一种简单的数据模型。Bigtable的客户端可以动态地控制数据的布局和格式，并且利用底层数据存储的局部性特征，它将数据统统看成无意义的字节串，客户端需要将结构化和非结构化数据串行化再存入Bigtable。接下来我按照论文的顺序分段整理我的读后感。

1. Bigtable的数据模型

Bigtable的核心数据模型是一个稀疏的多维Map数据结构，以 (行-row, 列-column, 时间戳-timestamp)为索引，在每个被索引的单元中存储对Bigtable不透明的用户数据。根据其数据模型，我们不难看出虽然Bigtable不是关系型数据库，但是却沿用了很多关系型数据库的术语，像table（表）、row（行）、column（列）等，我们也可以理解为Bigtable是一个键值（key-value）映射。按作者的说法，Bigtable是一个稀疏的，分布式的，持久化的，多维的排序映射。

下面我们来理解Bigtable中多维中的每一维的含义。Bigtable的键有三维，分别是行键（row key）、列键（column key）和时间戳（timestamp），行键和列键都是字节串，时间戳是64位整型；而值是一个字节串。可以用 (row:string, column:string, time:int64)→string 来表示一条键值对记录。行键可以是任意字节串，通常有10-100字节；列是第二级索引，每行拥有的列是不受限制的，可以随时增加减少；时间戳是第三级索引。

1. API

我们先查看API的功能，论文中描述的很清楚。Bigtable的API提供了删除和创建表和列族的功能。它还提供了改变簇、表和列族的元数据，比如访问控制权限。除去我们在各种数据库中了解操作过的增删改查操作，Bigtable还支持其他一些更复杂的处理数据的功能。比如它支持单行处理，该功能可以用来对存储在一个行关键字下的数据进行原子的读-修改-写操 作；又如Bigtable允许把每个表项用做整数记数器。此外，Bigtable支持在服务器的地址空间内执行客户端提供的脚本程序，其脚本程序的语言是google 开发的 Sawzall[28]数据处理语言。

1. 构建模块

Bigtable是构建在其他几个Google基础设施之上的。它使用了分布式Google文件系统来存储日志和数据文件。Bigtable的一个集群通常在一个共享机器池内进行操作，这个共享机器池会运行其他一些分布式应用。其进程通常和其他应用的进程共享同样的机器。Bigtable依赖一个集群管理系统来调度作业、在共享机器上调度资源、处理机器失败和监督机器状态。Bigtable依赖一个高可用的、持久性的分布式锁服务Chubby。一个Chubby服务包含5个动态副本，其中一个被选做主副本对外提供服务。Bigtable使用Chubby来完成许多任务：（1）保证在每个时间点只有一个主副本是活跃的，（2）来存储Bigtable数据的bootstrap的位置，（3）来发现tablet服务器，（4）宣告tablet服务器死亡，（5）存储Bigtable模式信息（即每个表的列家族信息），以及（6）存储访问控制列表。由此可见Bigtable依赖Chubby实现的功能颇多，由此根据两者之间的依赖关系，我们可以了解到，如果在一段时间以后，Chubby不能用了，Bigtable就不能用了。

1. 实现

根据论文中实现部分的介绍部分我们可以得知，Bigtable实现包括三个主要的功能组件：(1)库函数：链接到每个客户端，（2）一个主服务器，（3）许多Tablet服务器。

Bigtable支持Row为单位的transaction，底层依赖于chubby来支持各种同步协同相关操作，底层使用GFS来储存Tablet的文件内容，由于文件内容的只读形式，任何更新操作都是通过创建新版本的Row来实现的，为了加快响应，减少IO操作，写数据时，首先将数据写入内存的Memtable结构中（按Rowkey排序的）当Memtable大小超过一定阀值后，将这个Memtable作为一个新的文件写出到文件系统中。为了防止故障时内存中的数据丢失，同时在磁盘上维护了一个commit Log，数据的操作记录先写入LOG，然后才会被插入到Memtable中。此外读数据时，由于同一行的不同列或不同版本的数据可能分布在不同的物理文件以及Memtable中，需要同时扫描这些位置提交合并的结果。

1. Tablet的定位

既然Bigtable的主服务器不提供片的位置信息，那么客户端访问的方式就有待研究。通过论文中的图示以及解释，我们可以发现Bigtable使用一个类似B+树的数据结构存储片的位置信息。首先是第一层，Chubby file。这一层是一个Chubby文件，它保存着root tablet的位置。这个Chubby文件属于Chubby服务的一部分，一旦Chubby不可用，就意味着丢失了root tablet的位置，整个Bigtable也就不能用了；第二层是root tablet，它其实是元数据表（METADATA table）的第一个分片，它保存着元数据表其它片的位置。root tablet很特别，为了保证树的深度不变，root tablet从不分裂；第三层是其它的元数据片，它们和root tablet一起组成完整的元数据表。每个元数据片都包含了许多用户片的位置信息。

1. Tablet的存储和访问

如论文中图示，一个tablet的持久化存储是存在GFS当中。当片服务器收到一个写请求，片服务器首先检查请求是否合法。如果合法，先将写请求提交到日志去，然后将数据写入内存中的memtable。memtable相当于SSTable的缓存，当memtable成长到一定规模会被冻结，Bigtable随之创建一个新的memtable，并且将冻结的memtable转换为SSTable格式写入GFS，这个操作称为minor compaction。当片服务器收到一个读请求，同样要检查请求是否合法。如果合法，这个读操作会查看所有SSTable文件和memtable的合并视图，因为SSTable和memtable本身都是已排序的，所以合并相当快。每一次minor compaction都会产生一个新的SSTable文件，SSTable文件太多读操作的效率就降低了，所以Bigtable定期执行merging compaction操作，将几个SSTable和memtable合并为一个新的SSTable。BigTable还有个更厉害的叫major compaction，它将所有SSTable合并为一个新的SSTable。

然而memtable和SStable的详细内容在论文中并未有显示，经查询也并未发现有详细的数据结构，故不做详细描述。

1. 改进

以上所有需要实现之处均需要一系列完善措施从而获得高性能、可用性和可靠性，这些都是当下用户所要求的。根据论文中Google人员大致改进之处为：客户端可以把多个列族一起分组到一个locality group中；用户自定义的压缩格式可以被应用到每个SSTable块中（块的尺寸可以采用与locality group相关的参数来进行控制）；为了改进读性能，tablet服务器使用两个层次的缓存。Scan缓存是一个高层次的缓存，它缓存了“键–值”对，这些“键–值”对是由tablet服务器代码的SSTable接口返回的；允许客户端来确定，为某个特定locality group中的SSTable创建Bloom filter；以键（表，行名称，日志顺序号）的顺序对日志文件的条目（entry）进行排序；除了SSTable缓存，BigTable系统的其他部分也进行简化。

1. 评估

为测试BigTable的性能和可扩展性。论文中详细地介绍了评估过程。Tablet服务器、主服务器、测试客户端和GFS服务器都在同一个机器集合上运行。每个机器都运行一个GFS服务器。同时，一些机器还会另外运行一个tablet服务器，一个客户端进程。根据论文中图表我们可以看出：表格内容显示了每个tablet服务器每秒的操作的数量；图显示了每秒的操作的总数量。在系统中把tablet服务器的数量从1增加到500的过程中，累计吞吐量急剧增加，通常以100倍的规模。载均衡算法努力解决这个问题，但是，无法实现完美的目标，主要原因在于：第一，重新负载均衡有时候会被禁止，这样做可以减少tablet迁移的数量（当一个tablet迁移时，在短时间内是不能用的，通常是一秒）；第二， benchmark的负载是动态变化的。

1. 应用

相当多的google应用使用了Bigtable，比如Google Earth和Google Analytics，因此它和GFS、MapReduce并称为谷歌技术"三宝"。论文中详细介绍了三个应用示例：Google Analytics是一种服务，它帮助网站管理者分析网站流量模式。它提供了汇总分析，比如，每天不同访问者的数量，以及每天每个URL的网页视图的数量，以及网站流量报告，比如浏览了某个网页以后发生购买行为的用户的数量；Google提供很多服务，它支持用户访问高清晰度的卫星图片，或者通过基于浏览器的Google Maps接口，或者通过定制的客户端软件Google Earth。这些产品允许用户在地球表面进行导航。该产品支持在不同清晰度下查看和标注地图信息。该系统采用一个表进行数据预处理，用另一个表位用户提供数据服务；Personalized search是一种服务，它记录了用户查询和点击数据，涵盖了各个方面的Google属性，比如，网页搜索、图片和新闻。用户可以浏览他们自己的访问历史，他们可以要求根据Google使用历史模式来获得个性化的搜索结果。

1. 总结

经过阅读学习，我初步了解了Bigtable的内部结构，经过对比我发现它的结构与数据库较为相似但学习过后发现理解方式又大不相同，经查阅，找到一张对比图，如下：



这样对比之下就十分清晰了。接下来我按照论文的顺序进行学习，首先了解其数据模型，这是其内核所在，再了解API以及block，对应这些功能我了解到其实现原理，然后学习其中细节即Tablet的定位，存储和访问。了解完内核之后，继续了解Bigtable近来的改进之处，之后了解相关人员对其的评估过程与方法，最后了解其在Google其他服务中的应用。

通过这篇论文的学习我对Bigtable有了一定的了解，同时也让我对英语阅读的能力有所提升。在读后感的最后非常感谢李老师一学期以来的辛勤教导与耐心指教。