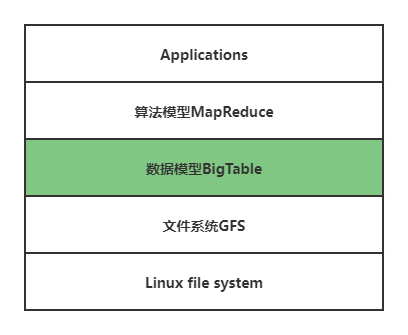
BigTable读后感

1. **BigTable介绍**

Google搜索引擎的支撑架构图如下，本文将思考并分析其中的BigTable部分：



BigTable是一个分布式存储系统，它用于处理海量数据。对于不同的大数据相关应用，BitTable提供了一个灵活性、高适用性、高容错性以及高可扩展性的解决方案。此外，用户能够通过此系统动态地控制数据的分布和格式。

1. **设计原理**
   1. **BigTable结构**

BigTable是一个稀疏、分布、永久的多维排序图，通过行列以及时间戳进行索引。

表中的行可以是任意字符串，在BigTable中对行的读写为原子操作，用户能够对同一行进行并发更新操作。此外，行关键字以字典序进行维护，且能够动态分区，每个区为一个Table。Table是数据分布和负载均衡调整的最小单位。

表中的列所组成的集合叫做“列族”，是访问控制的基本单位。存在同一个族下的数据通常为同一类型。一张表中的列族数量有限，但可以加载无限个列。

时间戳用于进行版本控制和索引。BigTable中，表的每一个数据项都包含该数据的不同版本，不同版本的数据能够按照时间戳倒叙排序。此外，用户能通过设置参数来对废弃版本的数据进行自动的垃圾收集，这样能够减轻多版本数据的管理负担。

* 1. **如何保存很大的表**

系统保存表的方法为key值索引，当用户需要查找时，使用key进行排序，然后通过二分查找找当相应文件。而一个大表则需要将其拆分成许多小表，并让每一个小表都是用key值存储。此外，再通过一个Metadata表来表示每个小表的存放位置。

那么，一个超大表则也使用类似的思路，通过二级分割的方式。分为多个大表，再进行存储。而多个大表也分别分为了多个小表。

* 1. **如何读写数据**

首先是读数据，系统通过再小表创建时同时写入一个内存索引的方式来加快查找小表位置的速度。此外，系统使用bloom filter过滤器，此过滤器能够以一个很小的花费来告诉系统需要查找的小表是否在此集合中。因此，只需要首先使用过滤器，若有可能，再对当前集合进行遍历查找。

那么，系统是如何写数据的呢？此时将GFS看作底层的硬盘，然后将系统架构分为内存和硬盘两部分。在内存中，系统通过存储内存表的方式，加速写效率。每次插入时，只需要对内存表进行排序，然后再写入硬盘中的表。因此，每个表将看作内存表加小表的集合。

当内存表达到存储上限时，系统将其作为一个小表写入硬盘。

* 1. **如何避免丢失内存数据**

在内存表写入硬盘前，是有可能丢失数据的。系统通过日志的方式解决此问题。系统使用WAL的形式写入日志，即往内存表写入数据前，首先往硬盘的日志中顺序写入记录。

* 1. **如何将BigTable与GFS连接**

基于前文的概括，实际上就是将小表和日志放置在GFS的ChunkServer中，并保存相应的副本。

1. **应用举例**

在论文中，Google给出了实际应用的例子：个性化查询。

个性化查询为一个双向服务，Google方和用户方都能够主动进行使用。如用户能够浏览查询历史，而Google能够通过查询历史为用户定制个性化结果。

在BigTable的使用上，个性化查询存储每个用户的数据。每个用户拥有唯一的id和列名绑定。而一个每个列族被用于存储用户的不同行为。而每个行为产生的数据项用时间戳进行记录。