
# Google 三驾马车之 Bigtable 读后感

姓名：周子涵

学号：18301060

日期：2020.11

# 目录

# 1. 文章综述

提起 Google，人们最先想到的都是“搜索引擎”这四个字。Google 被公认为全球最大的搜索引擎公司，月均处理数千亿次的查询请求。但是对于计算机行业的人们而言，Google 的伟大之处，并不仅仅是因为它建立了一个很好很强大的搜索引擎，还在于它创造了三项革命性的技术：GFS、MapReduce 和 Bigtable，即所谓的 Google 三驾马车。“三驾马车”的出现，才真正把我们带入了大数据时代，并指明了大数据的发展方向。

三驾马车中，GFS 是一个可扩展的大型数据密集型应用的、可伸缩的分布式文件系统，它的特性总结起来就是：同时具备最小的硬件投资和运营成本以及可靠的容错能力和计算性能。

MapReduce 系统使用超大集群来处理数据。基于 MapReduce 编写的程序是在大量的个人计算机上被并行分布式自动执行的，它的特点就是它有效地利用服务器中的所有处理器来计算保存在 GFS 中的海量数据并得到想要的结果。

Bigtable 则是一种用于处理海量数据的分布式结构化数据存储系统，用于处理实时在线应用的海量结构化数据。

总的来说，GFS 解决了分布式文件的存储，MapReduce 解决了海量数据的计算，而 Bigtable 解决了实时在线应用的海量结构化数据存储。

三驾马车中，Bigtable 将是本文将要详细阐述的内容。除了讲述对 Bigtable 的理解，我还会将 Bigtable 与课上讲到的 HBase 进行对比，通过对比从而更加深入地理解 Bigtable 以及 HBase 的原理与应用。

# 2. Bigtable 背景

了解技术的背景，能够帮助我们更好的了解技术本身。在简单了解了 Bigtable 的基本知识之后，我便开始思考：Google 为什么需要 Bigtable 技术。Google 实际应用的分布式系统环境里运行着 google 成千上万台的服务器，系统要对服务器内存以及海量规模的数据进行管理，同时还需要进行数据读写。在实际应用中，系统可能会出现各种各样的故障，网络、数据也都可能会出现问题。我们知道 Google 要存储海量的网页，同时基于网页的实时性特点，Google 甚至需要存储一个网页的不同时期的多个版本。如何保证这个系统中数据的可用性，并对抗系统中各个模块的失败，是需要着力解决的问题。而这也正催生了 Bigtable 技术。

# 3. Bigtable 实现理解

作为一个正在学习关系型数据库的学生，当我看到 Bigtable 这个词时，给我的第一反应是“table”，没错，就是关系数据库的表结构。但事实上 Bigtable 并不是一个关系型数据库，它并不支持完整的关系数据模型，而是为用户提供了简单的数据模型。在 Google 的 Bigtable 论文中 Data Model 章节的第一句话已经给它下了一个定义：“Bigtable 是一种稀疏的、分布式的、持久化的多维有序字典。Bigtable 由索引行、索引列以及时间戳组成，在字典中的每个值都是无解释的字节数组。”虽然对“稀疏分布式持久化”这些词汇还没有太清楚的认识，但是说到字典这个词，我终于将它和 NoSQL 建立起了联系。

接下来我将阐述一下在阅读完论文后对于 Bigtable 的理解。在阅读完全文

后，我终于对于上文所提到的定义有了一定的认识。这句定义概括了 Bigtable 技术的几个关键点：

首先，Bigtable 是一个字典，也就是包含海量 Key-Value 的映射。

其次，这个字典会按照 Key 值进行排序，这个 Key 并不是一个一维的数据，而是一个有 Row key，Column Key，Timestamp 组成的多维结构，其中行是表的第一级索引，列是表的第二级索引，时间戳是表的第三级索引。

此外，每一行每一列的组成并不是严格的某种固定结构，而是稀疏的，我认为这个概念应当是相对于结构化的关系型数据库。在 RDBMS 中，每一行（Row）对应的属性（Column）都是一样的，比如 Student 表中每一条记录都有{ID, Name, Sex, Class}，但是在 Bigtable 中，行与行可以由不同的列组成，如：Row1 对应的 Column 是{ID, Name, Class}，而 Row2 对应的则可以是{ID, Phone, Email}。

最后，由于需要存储海量数据，基于分布式的思想，这个大的 map 需要支持多个分区，并对其进行管理。

基于以上理解，当我再次通读文章之后，对于它数据模型的几个重要部分也有了更加深刻的认识。

首先是行：从逻辑上，Bigtable 的数据是以行为单位存储的，一行可以存储的单元数不限，Bigtable 通过行关键字的字典序来组织数据，表中每行都可以动态分区。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。这样做的好处是读取行中很少几列数据的效率很高，而且可以有效的利用数据的位置相关性，比如键“aaaaa”的存储行应该在键“aaaab”的旁边，而距离键“zzzzz”的存储行就非常远。

接着是列：Bigtable 的列是以 Family 为单位分组的，也就是列簇，以 Family：

qualifier 的格式呈现。这个概念在关系型数据库中是没有的，分组的目的一方面是可以以 Family 为单位进行权限控制，更重要的是，可以将不同相关但是不同类型的数据在物理上分组存储，便于获得更好的压缩效果。考虑到我们之前说到的数据的稀疏性，具体列的数量是没有限制的。但是 Family 必须预先确定好，并且数量不能太大，大量的组数也会让分组的优势减弱。

然后是 Timestamp：前文中也提到，由于 Google 需要存储大量网页数据，而这些网页数据是瞬息万变的，因此 Bigtable 中引入了 Timestamp 这个概念，中文翻译为时间戳。时间戳用来存储同一数据的不同版本，这些版本通过时间戳作为索引。查询时，如果只给出行列，那么返回的是最新版本的数据；如果给出了行列时间戳，那么返回的是时间小于或等于时间戳的数据。当然，考虑到网页数据更新快，我们无法存储所有网页所有时间的数据，为了减轻各版本数据的管理负担，每个列簇有 2 个设置参数，可通过这 2 个参数可以对废弃版本数据进行自动垃圾收集，用户可以指定只保存最后 n 个版本数据。

Bigtable 在实现方面主要包括 3 个主要组件：链接到用户程序的库，1 个 Master 服务器和多个 Tablet 服务器。关于 Master 和 Tablet 服务器的关系有一个很形象的比喻：如果说 Tablet 服务器是宰相，那么 Master 服务器就是皇帝。Master 服务器的主要作用是为 Tablet 服务器分配 Tablets，对 Tablet 服务器进行负载均衡，检测 Tablet 服务器的增减等。Tablet 服务器用来管理一个 Tablets 集合（十到上千个 Tablet），并负责它们的读写操作。Tablet 服务器可根据工作负载动态增减，用户客户端可以直接和 Tablet 服务器通信并进行读写，故 Master 的负载并不大。初始情况下，每个表只含一个 Tablet，随着表数据的增长，它会被自动分割成多个 Tablet，使得每个 Table 维持一定的大小。在任何时刻，一个

Tablet 只能分配给一个 Tablet 服务器，这个由 Master 来控制分配。

Bigtable 的实现依赖于名叫 Chubby 的高可用、序列化的分布式锁。通过 Chubby 跟踪 Tablet 服务器的状态。当 Tablet 服务器启动时，会在 Chubby 锁注册文件节点并获得其独占锁，当 Tablet 服务器失效或关闭时，会释放这个独占锁。Chubby 可以确保在任何时间最多只有一个活动的 Master，可以查找 Tablet 服务器，在 Tablet 服务器失效时进行善后等。如果 Chubby 长时间无法访问，Bigtable 就会失效。

在存储方面，我们知道 Bigtable 是基于 Google 的 GFS 的，因此它使用 GFS 存储日志文件和数据文件。Big table 的内部储存文件为 Google SStable 格式，SSTable 是一个持久化、排序的、不可更改的 Map 结构，它的内部是一系列数据块，并通过块索引定位，块索引在打开 SSTable 时加载到内存中，用于快速查找到指定的数据块。

# 4. Bigtable 优化

Bigtable 为了提高性能也做出了很多优化，比如局部性群簇，就是对列簇进行的优化。用户可以将多个列簇组合成一个局部性群簇，Tablet 中每个局部性群簇都会生产一个 SSTable。基于此，我们可以将通常不会一起访问的列簇分割成不同局部性群族，可以提高读取操作的效率。

此外，Bigtable 还是用了 Bloom（布隆）过滤器。由于一个读操作必须读取构成 Tablet 状态的所有 SSTable 数据，所以如果这些 SSTable 不在内存便需多次访问磁盘。基于此，Bigtable 引入 Bloom 过滤器，来查询 SStable 是否包含指定的行和列数据，通过付出少量 Bloom 过滤器的内存代价来显著减少访问磁盘次

数。

不仅如此，Bigtable 还进行的优化有：压缩、缓存、Commit 日志实现、Tablet 恢复提速、利用不变性等，但是目前我对于这些优化的理解还不够深入，之后将多次阅读论文进行理解。

# 5. Bigtable 与 HBase

老师在课上说，如果能够读懂 Bigtable 这篇论文，那么 HBase 也一定能理解的差不多了。因为 HBase 最初的实现可以说完全参考了 Bigtable 的论文。HBase 已经基本实现了 Bigtable 所有的功能，而且结构也基本相似。

在基础架构方面，两者在技术栈上的组成是类似的：Bigtable 是基于 Google 的 GFS 文件系统，而 HBase 是基于 Hadoop 的 HDFS 文件系统。此外 Bigtable 的 SSTable 对应了 HBase 的 Hfile，Chubby 锁对应了 HBase 的 Zookeeper，Tablet 在 HBase 中称之为 Region。需要指出的一点是 HBase 近来已经支持多个 Master。多个 Master 是“热”待命模式工作，它们都侦听 ZooKeeper 上的 Master 节点。

虽然 HBase 最初的实现可以说完全参考了 Bigtable 的论文，但是随着 HBase 的不断迭代，在一些方面也进行了优化与更新，同时在近些年也引入了诸如 Namespace 表命名空间等的高级特性。目前对这些优化的理解还不够深入，这些工作也将在之后继续深入开展。

# 6. 总结

Bigtable 从产生至今已经过去了 14 年之久，在 Google 后 Hadoop 时代，三驾马车也早已不是原来的模样，变成了由 Caffeine、Pregel、Dremel 组成的“新三驾马车”，但是他们给大数据技术带来的影响是深远的。

目前为止我对 Bigtable 有了一定的认识，但是还需要对它进行更加深入的理解。一篇论文不是读一次就能读明白的，每一次阅读都会产生新的理解，新的认识。

第一次写 paper 读后感，似乎写成了一篇 blog，此外对于 paper 的理解方面的能力还很欠缺，请老师谅解。当然也很感谢老师让我有这次机会，从各种“管理系统实现”中解脱出来，去真正的阅读和理解业内比较重要的 paper，去写出自己的理解与感悟。在接下来的时间里我将用更多的时间去体会行业内重要的 paper，去关注行业内正在发生什么，正在走向何方。