Google Bigtable读后感悟

18301054徐紫程

BigTable是Google设计的分布式数据存储系统，用来处理海量的数据的一种非关系型的数据库。BigTable是非关系型数据库，是一个稀疏的、分布式的、持久化存储的多维度排序Map。Bigtable的设计目的是快速且可靠地处理PB级别的数据，并且能够部署到上千台机器上。

BigTable已经在超过60个Google的产品和项目上得到了应用，包括 Google Analytics、GoogleFinance、Orkut、Personalized Search、Writely和GoogleEarth。这些产品对Bigtable提出了迥异的需求，有的需要高吞吐量的批处理，有的则需要上千台服务器，存储几百TB的数据。因此，针对 Google 的这些产品，BigTable已经实现了目标适用性广泛、可扩展、高性能和高可用性，还是成功的提供了一个灵活的、高性能的解决方案。

众所周知，对于一个大型的具有海量PV和海量数据的系统来说，其分布式服务是可以通过简单的增加节点进行扩展，但是底层的海量数据则因其单点和需要保持一致性等特点，成为一些大型系统的瓶颈。Google的BigTable就是底层海量数据的一个很好解决方案。从CAP理论来看的话，BigTable的思想是通过在一定程度上放弃底层数据的高可用性，主要加强的数据一致性和可扩展性。

存在即合理，我认为BigTable存在的原因大概是它能够对海量的数据进行快速存取。而BigTable的优势就在于它拥有更高效的存取性能而不需要复杂的SQL逻辑。为了实现一个高性能的存取目标：BigTable采用B+树的数据结构、key-value的形式来进行存储。BigTable用于海量数据，就比如google搜索引擎，google地图这样的操作相对简单的海量数据。而对于一些经常要改变的非海量数据，则使用传统的SQL数据比较合适。

关于BigTable的实现方式，BigTable将数据存在一个三维有序的表中，这个表除了传统二维表的row，column以外还增加了第三维TimeStamp，用来表示版本。这样rowid，colume family和timestamp就构成了一个三维有序的大表，数据就存储在这张大表中（当然这里的3维存储的格式和传统的数据库有一些不同）。

从上层来看的话，一个数据表就是一个三维有序的表的样子，而在底层来说，这个大表的实现方式则比较巧妙。每个大表被切分成若干个部分称为tablet，各个tablet分布在各个不同的tablet服务器上，这些tablet服务器都包含了缓存，日志和持久存储（这里的持久存储是将数据存储到GFS(Google File System)上去）。在每个服务器上缓存，日志和持久存储相互协作，最大程度的保证了数据的存取性能和安全性能。tablet服务器之间的负载均衡是通过合并与切分tablet来动态实现的，保证了服务器的高效利用。

为了提高性能和提高安全性，BigTable有一些其他机制。除了Tablet服务器以外还有Root服务器和Meta服务器，从Root服务器到Meta服务器再到Tablet服务器使用了B+树的数据结构，使得PV性能提高。在Tablet服务器之间存在一个Mater服务器用于统筹管理所有服务器的状态。BigTable还与Chubby紧密联系，添加了BigTable的安全性能。

BigTable与其他“两大马车”之间的关系大概是，BigTable是建立在 GFS 和 MapReduce 之上的。它的底层的架构是：GFS。每个Table都是一个多维的稀疏图为了管理巨大的Table，把Table根据行分割，这些分割后的数据统称为：Tablets。每个Tablets大概有100-200 MB，每个机器存储100个左右Tablets。BigTable包括三个主要的组件：链接到客户程序的lib、一个Master服务器和多个Tablet服务器，Master服务器负责；为Tablet服务器分配Tablet ；检测新加入或者过期失效的Tablet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 ；处理模式信息的修改（新建表、列族及访问控制信息）。由于GFS是一种分布式的文件系统，采用Tablets的机制后，可以获得很好的负载均衡。比如：可以把经常响应的表移动到其他空闲机器上，然后快速重建。现在Bigtable已经实现了下面的几个目标： 适用性广泛、可扩展、高性能和高可用性。但是Bigtable不支持完整的关系数据模式。且Bigtable将存储的数据都视为字符串。Bigtable为客户提供了简单的数据模型，利用这个模型，客户可以动态控制数据的分布和格式，用户也可以自己推测底层存储数据的位置相关性。另外，可以通过Bigtable的模式参数来控制数据是存放在内存中还是硬盘上。