**谷歌三驾马车读后感**

**18301118 张羽扬**

 Google的伟大之处，不仅因为它建立了一个很好很强大的搜索引擎，而且还在于它创造了3项革命性技术：GFS、MapReduce和BigTable，即所谓的Google三驾马车，他们是许多基础服务的基石。

GFS于2003年提出，是一个分布式的文件系统，与此前的很多分布式系统的前提假设存在很大的不同，适用于以下场景

1）认为组件失效是一种常态，提供了容错机制，自动负载均衡，使得分布式文件系统可以在廉价机器上运行

2）面向大文件存储，系统主要的工作负载是大规模的流式读取，写操作主要是追加方式写，很少有随机写

3）一次写入，多次读取，例如互联网上的网页存储。

Google文件系统GFS是一个可扩展的分布式文件系统，用于大型的、分布式的、对大量数据进行访问的应用。它运行于廉价的普通硬件上，但可以提供容错功能。它可以给大量的用户提供总体性能较高的服务。  
 GFS与过去的 分布式文件系统有很多相同的目标，但GFS的设计受到了当前及预期的应用方面的工作量及技术环境的驱动，这反映了 它与早期的文件系统明显不同的设想。这就需要对传统的选择进行重新检验并进行完全不同的设计观点的探索。  
一个GFS集群文件系统可能是多层分布的。一般情况下是成千上万个文件块服务器分布于不同的机架上，而这些文件块服务器又被分布于不同机架上的客户来访问。因此，不同机架上的两台机器之间的通信可能通过一个或多个交换机。 数据块 冗余配置策略要达到连个目的：最大的数据可靠性和可用性，最大的网络带宽利用率。因此，如果仅仅把数据的拷贝置于不同的机器上很难满足这两个要求，必须在不同的机架上进行数据备份。这样即使整个机架被毁或是掉线，也能确保数据的正常使用。这也使数据传输，尤其是读数据，可以充分利用带宽，访问到多个机架，而写操作，则不得不涉及到更多的机架。  
 在一个文件被删除之后，GFS并不立即收回磁盘空间，而是等到垃圾收集程序在文件和 数据块级的的检查中收回。当一个文件被应用程序删除之后，MASTER会立即记录下这些变化，但文件所占用的资源却不会被立即收回，而是重新给文件命了一个隐藏的名字，并附上了删除的 时间戳。在MASTER定期检查名字空间时，它删除超过三天（可以设定）的隐藏的文件。在此之前，可以以一个新的名字来读文件，还可以以前的名字恢复。当隐藏的文件在名字空间中被删除以后，它在内存中的元数据即被擦除，这就有效地切断了他和所有 数据块的联系。在一个相似的定期的名字空间检查中，MASTER确认孤儿 数据块（不属于任何文件）并擦除他的元数据，在和MASTER的心跳信 息交换中，每个服务器报告他所拥有的数据块，MASTER返回元数据不在内存的数据块，服务器即可以删除这些数据块。

而BigTable是谷歌设计的数据存储系统，它是全球化的、分布式的、持久化存储的、多维度排序的(数个层级)、可以被部署在几千台计算机上用来处理海量数据的一种非关系型的数据库。  
Bigtable特点：适用性广泛（全谷歌平台）、可扩展（大量数据）、高效处理性能和高可靠性。Bigtable广泛使用于谷歌的产品，适应谷歌不同平台不同的要求，可以适应大量数据快速处理，可以把数据快速输送给大量用户，可以在一台或者几台大型服务器使用，也可以在几千台小计算机配置。Bigtable在很多方面和数据库很类似，很多数据库功能它都可以实现，但是它以不同于数据库的方式在连接。  
Bigtable为客户提供了简单的数据模型，利用这个模型，用户可以自己随意控制数据的尺寸读取，它是动态设置而不是静态。用户可以自己决定文件储存在什么地方，以什么样的方式储存，数据会标明储存位置，名字则可以由用户任意决定。Bigtable将存储的数据都视为字符串来进行处理，但是Bigtable本身不去解读或者修改这些东西，而交由客户程序处理。客户程序会把储存的数据用一定方式与这些字符串相关联以供使用，而这些都可以由用户自己去控制如何操作设置。最后， BigTable也可以设置数据在哪里储存，比如储存在你的硬盘还是内存之中。  
特点:  
1、全球大规模海量数据；  
2、各地的几千台计算机同时进行都可以，效率极高；  
3、没有太大限制，扩展空间高；  
4、微机都可以使用；  
5、适合存取，不适合编辑；  
6、不适用于传统关系型数据库；  
 Bigtable的功能实现依靠三个主要部分：一个可以连接到每个用户的库文件，一个主要服务器控制其他，许多目录服务器来分配文件的储存。主服务器负责把目录分配到目录服务器上，检测目录服务器的改变，平衡目录服务器负载，对谷歌文件系统进行垃圾收集，还有控制不同列族等内容的改变。  
 每一个目录服务器都管理一组目录（一个目录服务器负责很多目录的管理）。目录服务器处理对该服务器上目录的读写请求，也就是不同目录对应的不同文件，也会将过大超过储存极限的目录分裂为多个目录来储存数据。一个Bigtable集群存储了大量的表。每一个表都由一组目录构成，每一个目录包含一定的储存数据。初始情况下，每个表仅包含一个目录。随着表内的数据的增加，它会自动分裂成多个目录来储存，默认每个表可以达到1GB。  
目录以树状形式储存，读取也是有主干具体到某个文件所在的叶，一层层读取下去，每一层目录都有个记录位置关键词下面，主服务器会缓存这个关键词以供查询和备份。每个目录只能分配给一个目录服务器，主服务器会自动检测目录服务器并实现合理分配利用。

MapReduce处理数据过程主要分成Map和Reduce两个阶段。首先执行Map阶段，再执行Reduce阶段。Map和Reduce的处理逻辑由用户自定义实现，但要符合MapReduce框架的约定。MapReuce处理数据的完整流程如下：

输入数据：对文本进行分片，将每片内的数据作为单个Map Worker的输入。分片完毕后，多个Map Worker便可以同时工作。

在正式执行Map前，需要将输入数据进行分片。所谓分片，就是将输入数据切分为大小相等的数据块，每一块作为单个Map Worker的输入被处理，以便于多个Map Worker同时工作。

Map阶段：每个Map Worker在读入各自的数据后，进行计算处理，最终输出给Reduce。Map Worker在输出数据时，需要为每一条输出数据指定一个Key，这个Key值决定了这条数据将会被发送给哪一个Reduce Worker。Key值和Reduce Worker是多对一的关系，具有相同Key的数据会被发送给同一个Reduce Worker，单个Reduce Worker有可能会接收到多个Key值的数据。在进入Reduce阶段之前，MapReduce框架会对数据按照Key值排序，使得具有相同Key的数据彼此相邻。如果您指定了合并操作（Combiner），框架会调用Combiner，将具有相同Key的数据进行聚合。Combiner的逻辑可以由您自定义实现。与经典的MapReduce框架协议不同，在MaxCompute中，Combiner的输入、输出的参数必须与Reduce保持一致，这部分的处理通常也叫做洗牌（Shuffle）。Reduce阶段：进入Reduce阶段，相同Key的数据会传送至同一个Reduce Worker。同一个Reduce Worker会接收来自多个Map Worker的数据。每个Reduce Worker会对Key相同的多个数据进行Reduce操作。最后，一个Key的多条数据经过Reduce的作用后，将变成一个值。输出结果数据。