谷歌三大马车读后感

谷歌的三大论文分别是Google File System、Google Bigtable以及Google MapReduce。在2003年到2006年短短三年的时间内，连续发表了轰动计算及学术界的三大论文，为二十一世纪计算机的发展起到了历史性的促进作用。

2003年发表的Google File System论文，是一个可拓展的分布式文件系统，用来储存服务产生和处理的数据以及大规模数据集的研究和开发。考虑到GFS由几百甚至上千台普通廉价的设备组装而成，并且会同时被大量的客户机访问，设计者将组件失效定义为常态事件，并将持续的监控、错误侦测、灾难冗余以及自动恢复的机制集成在GFS中来解决问题。

一个GFS集群以单独的master节点为主，辅以大量的chunk服务器，同时能被多个客户端访问。每一个chunk都会被分配一个不变的、全球唯一的64位的chunk标识。客户端向master服务端请求操作时，会传递chunk的标识和字节范围，master节点保存信息作为元数据储存在内存中，并返回相应的chunk标识和副本信息发还给客户端，客户端利用文件名和chunk索引作为key缓存这些信息，客户端可以在一次请求中查询多个chunk信息，避免和master节点未来可能发生的通讯。

为了能够让元数据储存在内存中，GFS设计了大空间的chunk，其尺寸为64MB，远远超出一般文件的Block size，大容量往往会造成内部碎片导致空间浪费，GFS设计者通过惰性空间的分配策略来避免浪费。大空间的设计减少了客户端与master节点通讯的需求，降低了工作负载，客户端也可以在更少的块上进行操作，与chunk服务器保持长时间的tcp连接，减少了网络负载。减少了需要保存的元数据的数量，有效提升了master服务器的操作速度，这使得master服务器可以在后台简单高效的周期性扫描保存的全部状态信息，保证了持续的监控。并且元数据保存在内存仅受限于master服务器所拥有的内存的大小，增加内存的开销少，但元数据存在内存里增强了系统的简洁性、可靠性，提供了高性能和灵活性。

GFS系统的工作负载主要由两种读操作组成：大规模的流式读取以及小规模的随机读取两种。大规模的流式读取通常一次读取数百KB的数据，常见的是一次读取1MB甚至更多的数据。来自同一个客户机的连续操作通常是读取同一个文件中连续的一个区域。小规模的随机读取通常是在文件某个随机的位置读取几个KB数据。如果应用程序对性能非常关注，通常的做法是把小规模的随机读取操作合并并排序，之后按顺序批量读取，这样就避免了在文件中前后来回的移动读取位置。

GFS系统的工作负载还包括大规模的、顺序的、数据追加方式的写操作。绝大部分文件的修改是采用在文件尾部追加数据，而不是覆盖原有数据的方式。对文件的随机写入操作在十几种几乎不存在。一旦写完之后，对文件的操作就只有读，而且是按顺序读，这些特性符合大量的数据。追加操作优化了性能，保证了原子性。

2004年公布的MapReduce论文描述了大数据的分布式计算方式，主要思想是将任务分解然后在多台处理能力较弱的计算节点中同时处理，然后将结果合并从而完成大数据处理。

MapReduce编程模型是利用一个输入key/value pair集合来产生一个输出的key/value pair集合。MapReduce库的用户用Map、Reduce函数表达这个计算。Map函数输出文档中的每个词、以及这个词的出现次数。Reduce函数负责把Map函数产生的每一个特定的词的计数累加起来。

MapReduce库设计的初衷是使用由成百上千的机器组成的集群来处理超大规模的数据，所以这个库必须要处理好机器故障。调用MapReduce函数时，会有一个特殊的程序master来分配Map任务和Reduce任务，master将一个Map任务或Reduce任务分配给一个空闲的worker。Master会周期性的ping每个worker，如果在约定的时间范围内没有收到worker返回的信息，那么这个worker就会被master标记为失效。所有由这个失效的worker完成的Map任务都会被重新设计为空闲状态，并将这些任务安排给其他的worker。同样的，失效worker重在运行的Map或Reduce任务也会被重新置为空闲状态等待重新调度。为了预防master失败，系统会周期性的把master数据结构写入磁盘，作为checkpoint，如果master失效了就从最后一个checkpoint开始启动另一个master进程。

谷歌最后发布的论文是2006年的Bigtable，为后续无数NoSQL数据库的出现提供了理论基础，目前Bigtable已经实现了适用性广泛、可拓展、高性能和高可用性的目标。

Bigtable是一个稀疏的、分布式的、持久化存储的多维排序Map。Map的索引是行关键字、列关键字以及时间戳；Map中的每个vaulue都是一个未经解析的byte数组。表中的行关键字可以是任意的字符串。对行关键字的读写操作都是原子的。列关键字组成列族，列族是访问控制的基本单位，同意列族下的所有数据通常属于同一类型。

Bigtable使用GFS存储日志文件和数据文件。BigTable内部存储数据的文件是Google SSTable格式的。SSTable是一个持久化的、排序的、不可更改的Map<key,value>结构，其值都是任意的byte串，因此使用key查询速度很快。Big Table还依赖一个高可用的、序列化的分布式锁服务组件——Chubby。BigTable使用Chubby完成以下几个任务：

1. 1. 确保在任何时间内最多只有一个活动的Master副本；
2. 确保在任何时间内最多只有一个活动的Master副本；
3. 存储BigTable数据的自引导指令的位置
4. 查找Tablet服务器，以及在Tablet服务器失效时进行善后；
5. 存储BigTable模式信息
6. 存储访问控制列表。

Bigtable包括了三个主要组件：连接到客户程序的库、一个master服务器和多个tablet服务器。针对系统工作负载的变化情况，BigTable可以动态的向集群中添加Tablet服务器。Tablet的位置信息储存在一个三层的、类似B+树的结构，第一层包含了Root Tablet的位置信息，存储在Chubby中。Root Tablet包含了一个特殊的METADATA表里所有的Tablet位置信息，每个Tablet包含了一个用户Tablet的集合。Root Tablet永远不会被分割，这就保证了Tablet的位置信息存储结构不会超过三层。客户端读取的数据都不经过Master服务器；客户程序直接和Tablet服务器通信进行读写操作。在任何一个时刻，一个Tablet只能分配给一个Tablet服务器。Master服务器记录了当前有那些活跃的Tablet服务器、那些Tablet分配给了那些Tablet服务器、那些Tablet还没有被分配。

整个BigTable设计符合大部分大数据程序的需求，打破了关系型数据库的结构化存储，能够部署在成千上万台服务器上，可以存储PB级数据，对整个互联网行业的快速发展提供了坚实的理论基础与成功案例。