云计算之——谷歌的三驾马车

指导老师：张帆

作者：牟钰潇

学号：2017221105020

1. **云计算与Google云计算体系结构**

**1.1云计算基本原理**

云计算是对分布式处理、并行处理和网格计算及分布式数据库的改进处理，其前身是利用并行计算解决大型问题的网格计算和将计算资源作为可计量的服务提供的公用计算，在互联网宽带技术和虚拟化技术高速发展后萌生出云计算。

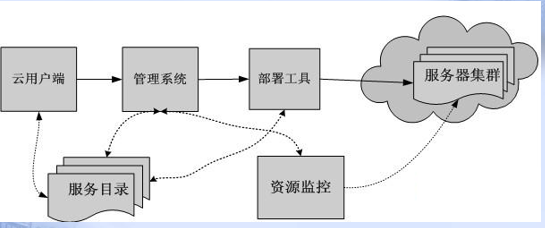
而基于云计算的发展和我们对于云计算的接触愈渐上涨，云计算又可以概括为：

利用非本地或远程服务器（集群）的分布式计算机为互联网用户提供服务（计算、存储、软硬件等服务）。这使得用户可以将资源切换到需要的应用上，根据需求访问计算机和存储系统。云计算可以把普通的服务器或者PC连接起来以获得超级计算机计算机的计算和存储等功能，但是成本更低。

云计算真正地实现按需计算，从而有效地提高了我们对于软硬件资源地利用效率。它的出现使得高效并行计算不再仅仅局限于科学家和专业人士使用，普通用户也开始渐渐享受它所带来的便利，使得人人都有机会使用并行机。人们不需要了解内部如何运转，不用知道服务器在哪，只要用互联网就可以透明地使用各种资源。可以说云计算的出现改变了人类，是一次巨大的革新。

**1.2 云计算体系结构**

云计算平台作为一个强大的云网络，连接了大量的并发网络计算和服务，它的通用体系结构大致如下：



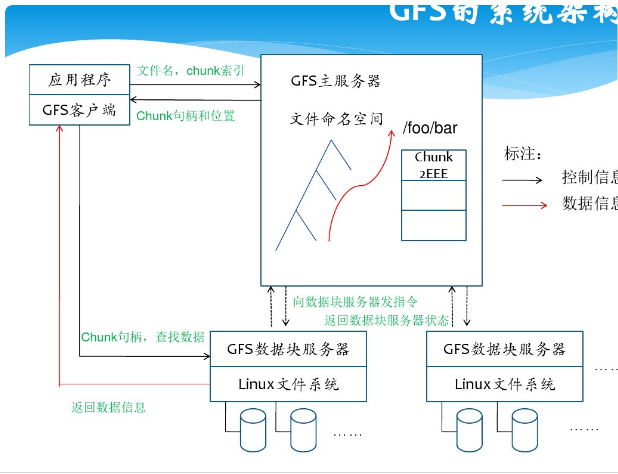
而Google云计算地基础架构（最主要的3大技术构件）=GFS+MapReduce+BigTable：



**二、Google云计算的三驾马车**

**2.1核心技术之GFS**

网页搜索业务需要海量的数据存储，同时还需要满足高可用性、高可靠性和经济性等要求。为此，Google开发了分布式文件系统(Google file system)。GFS支持海量数据处理，高并发访问、硬件故障自动恢复，并实现了一次写入、多次读取的数据处理模式。因此用一句话来概括GFS（Google文件系统）的功能，那就是：GFS提供了[海量数据](https://www.baidu.com/s?wd=%E6%B5%B7%E9%87%8F%E6%95%B0%E6%8D%AE&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)存储和访问的能力。



GFS技术基于大量安装有Linux操作系统的普通PC构成的集群系统。整个集群系统由一台Master(通常有几台备份)和若干台TrunkServer构成。GFS中文件备份成固定大小的Trunk分别存储在不同的TrunkServer上，每个Trunk有多份(比如3份或其他，一般是3份)拷贝，也存储在不同的TrunkServer上。Master负责维护GFS中的Metadata，即文件名及其Trunk信息。客户端先从Master上得到文件Metadata，根据要读取的数据在文件中的位置与相应的 TrunkServer通信，获取文件数据。

高可靠性是GFS最重要的特点。GFS使用的是可靠性较差的普通PC，节点失效属于正常现象，起中很大一部分设计就是要解决单节点甚至双节点同时实效的问题。

鉴于Google Inc. 的特殊应用环境，需要处理海量数据，经常有大文件(几十G)的操作，而且常是多台机器同时数据输出到一个大文件中，供后面的流程使用。GFS也对这种特殊的应用需求做了很多优化，保证往大文件并发追写数据时的可靠和高效。

目前Google拥有超过200个的GFS集群，其中有些集群的计算机数量超过5000台。Google现在拥有数以万计的连接池从GFS集群中获取数据，集群的数据存储规模可以达到5个PB，并且集群中的数据读写吞吐量可达到每秒40G。

而从设计上来讲：GFS专门为Google的核心数据即页面搜索的存储进行了优化。数据使用大到若干G字节的大文件持续存储，而这些文件极少被删除、覆盖或者减小；通常只是进行添加或读取操作。它也是针对Google的计算机集群进行的设计和优化，这些节点是由廉价的“常用”计算机组成，这就意味着必须防止单个节点的高损害率和随之带来的数据丢失。其它设计理念包括高数据吞吐率，甚至这带来了存取反应期变差。

节点分为两类：*主*节点和*Chunkservers*。Chunkservers存储数据文件，这些单个的文件象常见的文件系统中的簇或者扇区那样被分成固定大小的数据块（这也是名字的由来）。每个数据块有一个唯一的64位标签，维护从文件到组成的数据块的逻辑映射。每个数据块在网络上复制一个固定数量的次数，缺省次数是3次，对于常用文件如可执行文件的次数要更多。

主服务器通常并不存储实际的大块数据，而是存储与大块数据相关的元数据，这样的数据如映射表格将64位标签映射到大块数据位置及其组成的文件、大块数据副本位置、哪个进程正在读写特定的大数据块或者追踪复制大块数据的“快照”（通常在主服务器的激发下，当由于节点失败的时候，一个大数据块的副本数目降到了设定的数目下）。所有这些元数据通过主服务器周期性地接收从每个数据块服务器来的更新（“心跳消息”）保持最新状态。

操作的允许授权是通过限时的、倒计时“租期”系统来处理的，主服务器授权一个进程在有限的时间段内访问数据块，在这段时间内主服务器不会授权其它任何进程访问数据块。被更改的chunkserver——总是主要的数据块存储器，然后将更改复制到其它的chunkserver上。这些变化直到所有的chunkserver确认才存储起来，这样就保证了操作的完整性和自动性。

访问大数据块的程序首先查询主服务器得到所要数据块的位置，如果大数据块没有进行操作（没有重要的租约），主服务器回答大数据块的位置，然后程序就可以直接与chunkserver进行联系接收数据（类似于[Kazaa](https://wiki.huihoo.com/wiki/Kazaa" \o "Kazaa)和它的超级节点）。

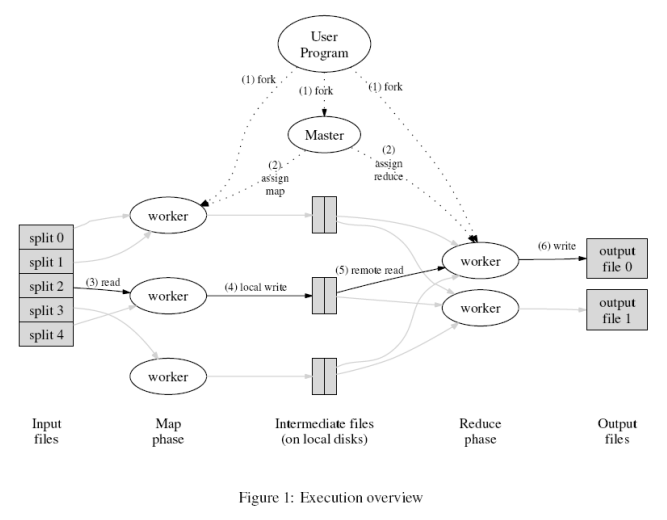
**2.2核心技术之MapReduce**

  MapReduce是一个编程模式，它是与处理/产生海量数据集的实现相关。用户指定一个map函数，通过这个map函数处理key/value（键/值）对，并且产生一系列的中间key/value对，并且使用reduce函数来合并所有的具有相同key值的中间键值对中的值部分。MapReduce的主要贡献在于提供了一个简单强大的接口，通过这个接口，可以把大尺度的计算自动的并发和分布执行。

这个定义可能比较抽象，用一个例子来说明比如：在很大的文档集合中通机每一个单词出现的次数。map函数检查每一个单词，并且对每一个单词增加1到其对应的计数器;reduce函数把特定单词的所有出现的次数进行合并。

使用这样的函数形式实现的程序可以自动分布到一个由普通机器组成的超大几群上并发执行。run-time系统会解决输入数据的分布细节，跨越机器集群的程序执行调度，处理机器的失效，并且管理机器之间的通讯请求。这样的模式允许程序员可以不需要有什么并发处理或者分布式系统的经验，就可以处理超大的分布式系统得资源。

我们的MapReduce系统的实现运行在一个由普通机器组成的大型集群上，并且有着很高的扩展性：一个典型的MapReduce计算处理通常分布到上千台机器上来处理上TB的数据。程序员会发现这样的系统很容易使用：已经开发出来了上百个MapReduce程序，并且每天在Google的集群上有上千个MapReduce job正在执行。



上图是我们实现的MapReduce操作的整体数据流。当用户程序调用MapReduce函数，就会引起如下的操作（图一中的数字标示和下表的数字标示相同）。

1. 用户程序中的MapReduce函数库首先把输入文件分成M块，每块大概16M到64M（可以通过参数决定）。接着在cluster的机器上执行处理程序。
2. 这些分排的执行程序中有一个程序比较特别，它是主控程序master。剩下的执行程序都是作为master分排工作的worker。总共有M个map任务和R个reduce任务需要分排。master选择空闲的worker并且分配这些map任务或者reduce任务
3. 一个分配了map任务的worker读取并处理相关的输入小块。他处理输入的数据，并且将分析出的key/value对传递给用户定义的map函数。map函数产生的中间结果key/value对暂时缓冲到内存。
4. 这些缓冲到内存的中间结果将被定时刷写到本地硬盘，这些数据通过分区函数分成R个区。这些中间结果在本地硬盘的位置信息将被发送回master，然后这个master负责把这些位置信息传送给reduce的worker。
5. 当master通知reduce的worker关于中间key/value对的位置时，他调用remote procedure来从map worker的本地硬盘上读取缓冲的中间数据。当reduce的worker读到了所有的中间数据，他就使用中间key进行排序，这样可以使得相同key的值都在一起。因为有许多不同key的map都对应相同的reduce任务，所以，排序是必须的。如果中间结果集太大了，那么就需要使用外排序。
6. reduce worker根据每一个唯一中间key来遍历所有的排序后的中间数据，并且把key和相关的中间结果值集合传递给用户定义的reduce函数。reduce函数的对于本reduce区块的输出到一个最终的输出文件。
7. 当所有的map任务和reduce任务都已经完成了的时候，master激活用户程序。在这时候MapReduce返回用户程序的调用点。

当这些成功结束以后，mapreduce的执行数据存放在总计R个输出文件中（每个都是由reduce任务产生的，这些文件名是用户指定的）。通常，用户不需要合并这R个输出文件到一个文件，他们通常把这些文件作为输入传递到另一个MapReduce调用，或者用另一个分布式应用来处理这些文件，并且这些分布式应用把这些文件看成为输入文件由于分区（partition）成为的多个块文件。

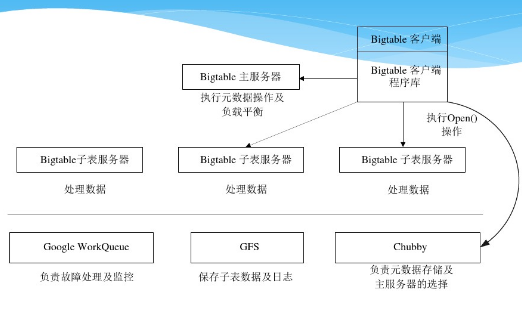
**2.3 核心技术之BigTable**

有了前两个核心技术，现在谷歌需要的是数据存储的可靠性，高速的检索，读取，海量数据的记录，还有多版本的记录保存。

而为了实现这些需求，Google使用了BigTable。

BigTable是一种用于管理超大规模结构化数据的分布式存储系统，可以管理分布在数以千计服务器上的以PB计的数据。它的存在使得海量数据的管理和组织变得方便BigTable API将包括用于创建、编辑表和列，改变群集、表、列元数据的函数。BigTable不支持完全的关系数据模型，而是为客户提供了简单的数据模型，让客户来动态控制数据的分布和格式。值得一提的是：BigTable能支持大部分SQL。

BigTable中的数据项按照行关键字的字典序排列，每行动态地划分到Tablet中，每个服务器节点Tablet Server负责管理大约100个记录板。时间戳是一个64位的整数，表示数据的不同版本。列簇是若干列的集合，BigTable中的存取权限控制在列簇的粒度进行。BigTable系统依赖于集群系统的底层结构，一个是分布式的集群任务调度器，一个是GFS文件系统，另一个是分布式锁服务Chubby，如图1-10所示。Chubby是一个非常健壮的粗粒度锁，BigTable使用Chubby来保存Root Tablet的指针，并使用一台服务器作为主服务器，用来保存和操作元数据。当客户端读取数据时，用户首先从Chubby Server中获得Root Tablet的位置信息，并从中读取相应的元数据表Metadata Tablet的位置信息，接着从Metadata Tablet中读取包含目标数据位置信息的User Table的位置信息，然后从该User Table中读取目标数据的位置信息项。BigTable的主服务器除了管理元数据之外，还负责对Tablet Server进行远程管理与负载调配。客户端通过编程接口与主服务器进行控制通信以获得元数据，与Tablet Server进行数据通信，而具体的读写请求则由Tablet Server负责处理。BigTable是客户端和服务器端的联合设计，使得性能能够最大程度地符合应用的需求。



最后，要说为什么Google的三大核心技术奠定了大数据的基础，主要在于：大数据的特点：海量、多样，高速，易变。三大核心技术无论是在海量，多变，高速还是多样方面都有其独特的解决思路，这三大核心技术改变，颠覆了人类，使得我们的生活进步，使得我们进入了Cloud时代。