Google三大论文读后感

--18221261 林美辰

身处大数据时代，面对纷繁的数据，我们经常会疑惑，数据是如何存储，存储之后又是怎样原封不动的读取出来，读了Google的三大论文，对于这个问题，Google给出了答案，那么也让我交流一下读完这三篇论文之后我的感受。

三篇论文分别介绍了GFS、MapReduce、BigTable三大理论，这三大理论通常被人称为“三驾马车”，这也足以见得这三大理论的重要价值和地位，我认为要想完全理解这三篇论文需要长期的、丰富的知识积累，作为一名大三的学生，对于这三篇论文我还不能做到完全的理解，但是我也可以从中了解到大数据的相关知识和处理这些问题的使用的方法，并且通过学习先辈们解决这个问题的方法，丰富自己的知识积累，以便更轻松的解决自己遇到的问题，并且可以开拓自己的视野，从而更加热爱软件这个行业。

马云在卸任时讲到：“很多人还没搞清楚什么是PC互联网，移动互联网来了，我们还没搞清楚移动互联的时候，大数据时代又来了。”可见，正处于大数据时代的我们很有必要了解一下大数据方面的知识。

下面我会分别介绍每一篇论文并发表我自己的见解：

1. Google File System文件系统

为了满足谷歌迅速增长的数据处理需求，谷歌设计并实现了GFS分布式文件系统。GFS包括几百甚至几千台普通的廉价设备组装的存储机器，同时被相当数量的客户机访问，这也导致任何给定时间内都有可能发生某些组件无法工作，所以常常出现如应用程序错误、操作系统的错误、人为失误，甚至还有硬盘、内存、连接器、网络以及电源失效等问题。系统必须持续监控自身的状态，它必须将组件失效作为一种常态，能够迅速地侦测、冗余并恢复失效的组件。

GFS分布式文件系统具有容错能力强、支持的数据量大的设计特点，并且采用追加的方式，不支持随机写入，除此之外还采取跨平台的操作。因为GFS构建在廉价的PC机上，可以在出现错误时常规地进行错误恢复，主要支持适当数量的大文件，应当支持的小文件，但是不会对其进行优化，文件读取一般一次读取较大的比特数，随机读写的问题可以次要考虑，并且支持多个客户端对一个文件读写的同步。

关于GFS的构建，一个GFS集群包含一个单独的master节点，GFS的设计原则之一就是尽量减少master和client之间的交互。多台chunk服务器，并且同时被多个客户端访问。在GFS中储存的文件都会被分割成固定大小的chunk，并且为了方便识别储存的数据，会在创建每一个chunk时，分配一个不变的，全球唯一的64位chunk标识，并且以LINUXS文件的形式保存在本地硬盘上。单一的master节点会大大简化GFS的设计，因为客户端通过master找到需要询问的chunk服务器，客户端将这些元数据缓存一段时间后，后续会直接与chunk服务器进行数据读取。对于元数据都储存在master的内存中，有三种主要类型的元数据，文件和chunk的命名空间，文件和chunk的对应关系，每个chunk副本的存放地点。关于我自己的理解是这些类型的元数据会方便实现读取，保持master服务器与chunk同步状态，不会导致master服务器崩溃。

而后，关于文件的读取和写入，对于读取和写入效率，当读取和写入的客户机增加时，多个客户机同时读取和写入的几率也增加，导致整体的读取和写入效率降低。感觉就像我们平时用网线，当多个人用同一条网线时，你的网速就会变得很慢，放视频打游戏动不动就卡机。GFS系的读取有两种操作，一种是大规模的流式读取，大规模的流式读取通常一次读取数百KB的数据，甚至更多。而另外一种是小规模的随机读取，并且在GFS中对于记录追加通过chunk和chunk的副本进行操作，保证至少有一次原子的写入操作成功执行。

最后，master服务器的稳定和数据恢复备份。在GFS中为了保持master服务器的可靠性，master服务器的状态也要复制，master服务器的所有操作日志和checkpoint文件都被复制到多台机器上，其中操作日志是元数据的唯一持久化存储记录，在master服务器在灾难恢复时，通过重演操作日志把文件系统恢复到最近的状态，而checkpoint是对数据库做一次快照的行为，读取checkpoint文件以及重演checkpoint之后的有限个日志文件就能恢复系统。而且在论文中可以了解到快照是对一个文件或者目录树做一个拷贝，可以瞬间完成，并且不会对同时进行的其他操作造成干扰，所以通过快照，用户可以创建一个巨大的数据集的分支拷贝，就如同备份一样，可以直接提交或者回到拷贝当前状态。

二、Bigtable 分布式结构化数据存储系统

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具有适用性广泛、可扩展、高效处理效能和高可靠行性的特点。关于适应性广，BigTable适用于全谷歌平台，并且广泛适用于谷歌的产品中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。

关于Bigtable的功能实现，主要依靠三个主要部分：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。

而且对于BigTable的目录来说，每一个目录服务器都管理一组目录，即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。

BigTable的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。

Bigtable中的行关键字可以是任意的字符串，并且每行的读写操作都是原子的，Bigtable中的行关键字是按照字典顺序排序存储的，表中的行都可以进行动态分区，每个分区叫tablet，而tablet是数据分布和负载均衡的最小单位。由于行键是按照字典序存储的，所以查询时以行关键字作为条件查询速度是毫秒级的。

Bigtable使用谷歌的分布式文件系统GFS存储日志文件和数据文件。BigTable内部存储数据的问文件时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可以完成的任务，包括确保在任何时间内最多只有一个活动的master副本，存储BigTable数据的自引导指令的位置，查找Tablet服务器，以及在Tablet服务器失效时进行善后，存储BigTable模式信息和存储访问控制列表。

三、MapReduce编程模型

MapReduce 是一个编程模型，也是一个处理和生成超大数据集的算法模型的相关实现。它有一个很大的优势就是大量的减少了时间，使用MapReduce模型，再结合用户实现的Map函数和Reduce函数，我们就可以非常容易的实现大规模并行化计算。

下面我想要介绍一下MapReduce设计上具有的主要的技术特征：

首先MapReduce是向“外”横向扩展，而非向“上”纵向扩展 即MapReduce集群的构建完全选用价格便宜、易于扩展的低端商用服务器，而非价格昂贵、不易扩展的高端服务器。

1. 失效被认为是常态 MapReduce集群中使用大量的低端服务器，因此，节点硬件失效和软件出错是常态，因而一个良好设计、具有高容错性的并行计算系统不能因为节点失效而影响计算服务的质量，任何节点失效都不应当导致结果的不一致或不确定性；任何一个节点失效时，其他节点要能够无缝接管失效节点的计算任务；当失效节点恢复后应能自动无缝加入集群，而不需要管理员人工进行系统配置。 MapReduce并行计算软件框架使用了多种有效的错误检测和恢复机制，如节点自动重启技术，使集群和计算框架具有对付节点失效的健壮性，能有效处理失效节点的检测和恢复。
2. MapReduce把处理向数据迁移 传统高性能计算系统通常有很多处理器节点与一些外存储器节点相连，如用存储区域网络（Storage Area，SAN Network）连接的磁盘阵列，因此，大规模数据处理时外存文件数据I/O访问会成为一个制约系统性能的瓶颈。 为了减少大规模数据并行计算系统中的数据通信开销，代之以把数据传送到处理节点（数据向处理器或代码迁移），应当考虑将处理向数据靠拢和迁移。MapReduce采用了数据/代码互定位的技术方法，计算节点将首先尽量负责计算其本地存储的数据，以发挥数据本地化特点，仅当节点无法处理本地数据时，再采用就近原则寻找其他可用计算节点，并把数据传送到该可用计算节点。
3. MapReduce顺序处理数据、避免随机访问数据 大规模数据处理的特点决定了大量的数据记录难以全部存放在内存，而通常只能放在外存中进行处理。由于磁盘的顺序访问要远比随机访问快得多，因此MapReduce主要设计为面向顺序式大规模数据的磁盘访问处理。 为了实现面向大数据集批处理的高吞吐量的并行处理，MapReduce可以利用集群中的大量数据存储节点同时访问数据，以此利用分布集群中大量节点上的磁盘集合提供高带宽的数据访问和传输。
4. MapReduce为应用开发者隐藏系统层细节 软件工程实践指南中，专业程序员认为之所以写程序困难，是因为程序员需要记住太多的编程细节（从变量名到复杂算法的边界情况处理），这对大脑记忆是一个巨大的认知负担，需要高度集中注意力；而并行程序编写有更多困难，如需要考虑多线程中诸如同步等复杂繁琐的细节。由于并发执行中的不可预测性，程序的调试查错也十分困难；而且，大规模数据处理时程序员需要考虑诸如数据分布存储管理、数据分发、数据通信和同步、计算结果收集等诸多细节问题。 MapReduce提供了一种抽象机制将程序员与系统层细节隔离开来，程序员仅需描述需要计算什么（What to compute），而具体怎么去计算（How to compute）就交由系统的执行框架处理，这样程序员可从系统层细节中解放出来，而致力于其应用本身计算问题的算法设计。

最后，MapReduce平滑无缝的可扩展性 这里指出的可扩展性主要包括两层意义上的扩展性：数据扩展和系统规模扩展性。 理想的软件算法应当能随着数据规模的扩大而表现出持续的有效性，性能上的下降程度应与数据规模扩大的倍数相当；在集群规模上，要求算法的计算性能应能随着节点数的增加保持接近线性程度的增长。绝大多数现有的单机算法都达不到以上理想的要求；把中间结果数据维护在内存中的单机算法在大规模数据处理时很快失效；从单机到基于大规模集群的并行计算从根本上需要完全不同的算法设计。奇妙的是，MapReduce在很多情形下能实现以上理想的扩展性特征。 多项研究发现，对于很多计算问题，基于MapReduce的计算性能可随节点数目增长保持近似于线性的增长。

具有了这些功能的MapReduce具有哪些功能呢？

首先，MapReduce具有数据划分和计算任务调度功能。MapReduce系统自动将一个作业（Job）待处理的大数据划分为很多个数据块，每个数据块对应于一个计算任务（Task），并自动调度计算节点来处理相应的数据块。作业和任务调度功能主要负责分配和调度计算节点（Map节点或Reduce节点），同时负责监控这些节点的执行状态，并负责Map节点执行的同步控制。

1. MapReduce具有数据/代码互定位功能。为了减少数据通信，一个基本原则是本地化数据处理，即一个计算节点尽可能处理其本地磁盘上所分布存储的数据，这实现了代码向数据的迁移；当无法进行这种本地化数据处理时，再寻找其他可用节点并将数据从网络上传送给该节点（数据向代码迁移），但将尽可能从数据所在的本地机架上寻找可用节点以减少通信延迟。
2. MapReduce可以进行系统优化。为了减少数据通信开销，中间结果数据进入Reduce节点前会进行一定的合并处理；一个Reduce节点所处理的数据可能会来自多个Map节点，为了避免Reduce计算阶段发生数据相关性，Map节点输出的中间结果需使用一定的策略进行适当的划分处理，保证相关性数据发送到同一个Reduce节点；此外，系统还进行一些计算性能优化处理，如对最慢的计算任务采用多备份执行、选最快完成者作为结果。
3. MapReduce具有出错检测和恢复的功能。以低端商用服务器构成的大规模MapReduce计算集群中，节点硬件（主机、磁盘、内存等）出错和软件出错是常态，因此MapReduce需要能检测并隔离出错节点，并调度分配新的节点接管出错节点的计算任务。同时，系统还将维护数据存储的可靠性，用多备份冗余存储机制提高数据存储的可靠性，并能及时检测和恢复出错的数据。

简单来说，MapReduce能将海量的数据浓缩成人们想要的数据。Hadoop的两大核心是HDFS和MapReduce，Hadoop的体系结构主要通过HDFS的分布式存储作为底层数据支持的。并且通过MapReduce来进行计算。

三大论文中的Google File System文件系统，是一个面向大规模数据密集型应用的、可伸缩的分布式文件系统。Bigtable 是一个分布式的结构化数据存储系统，它被设计用来处理海量数据：通常是分布在数千台普通服务器上的 PB 级的数据。而MapReduce 是一个编程模型，也是一个处理和生成超大数据集的算法模型的相关实现。

总的来说，三大论文涉及到的内容十分新颖，并且开拓了我的视野，让我对于数据的存储有了新的认识。不得不说，谷歌为大数据的存储提供了一种新思路，让我感到十分新奇，但是要完全理解确实还有一定的难度，当然这需要我在日后的学习过程中多领悟，甚至对论文的内容多读几次才能完全掌握，但是对于现阶段来说，谷歌的三大论文开拓了我的视野，让我不在局限于传统的数据存储形式，这也为我的学习生活大有裨益！