**“三驾马车”读后感**

18301104 牛亚翔

在读完关于这三篇重要论文后，我收获颇丰，对大数据和分布式系统有了初步的了解。由于三篇文稿是英文的，所以读起来有一定的难度。所以我从网上查找了很多相关的资料尽心辅助阅读。并根据网上的前辈总结，有了一些自己对三驾马车的认识。

**GFS**

首先是GFS。它是Google面向大规模数据密集型应用的、可伸缩的分布式文件系统。GFS是一个可扩展的大型数据密集型应用的分布式文件系统，该文件系统可在廉价的硬件上运行，并具有可靠的容错能力，该文件系统可为用户提供极高的计算性能，而同时具备最小的硬件投资和运营成本。

任何技术都在有了需求后才会诞生的。对于GFS来说，谷歌进行了四种假设：一、系统由许多廉价的普通组件组成，组件失效是一种常态。二、系统的工作负载主要由两种读操作组成：大规模的流式读取和小规模的随机读取。三、系统的工作负载还包括许多大规模的、顺序的、数据追加方式的写操作。四、高性能的稳定网络带宽远比低延迟重要。

GFS采用主/从模式，一个GFS包括一个master服务器r和多个chunk服务器。对于超大数据存放，比如几百万的文件的话。我们可以分成好几个chunk server. 每个chunk server可以简单的理为一个廉价linux电脑。然后由一个Master server， 一个老大来做索引。首先能够知道文件在哪个chunk server上，先去那个server，然后再让chunk server从它自己的index里找东西。

**MR**

其次是MR。MR全称为MapReduce。MR是针对分布式并行计算的一套编程模型。基于MapReduce编写的程序是在成千上万的普通PC机上被并行分布式自动执行的。它将所有服务器中的处理器有效地利用起来计算保存在谷歌文件系统中的海量数据并得到想要的结果。

对于MR来说，它是由Map和reduce组成,来自于Lisp，Map是影射，把指令分发到多个worker上去，Reduce是规约，把Map的worker计算出来的结果合并。通过将Map调用的输入数据自动分割为M个数据片段的集合，Map调用被分布到多台机器上执行。输入的数据片段能够在不同的机器上并行处理。使用分区函数将Map调用产生的中间key值分成R个不同分区，Reduce调用也被分布到多台机器上执行。分区数量（R）和分区函数由用户来指定。对于MR的调用过程，主要有以下七步。

1.用户程序调用的MapReduce库 首先将输入文件分成M个数据片度，每个数据片段的大小一般从 16MB到64MB，程序可以通过可选的参数来控制每个数据片段的大小。 然后master在机群中创建大量的用户程序副本，并把map/reduce函数给不同的机器执行。

2.这些程序副本中的有一个特殊的程序–master。副本中其它的程序都是worker程序，由master分配任务。有M个Map任务和R个Reduce任务将被分配，master将一个Map任务或Reduce任务分配给一个空闲的worker。

3.被分配了map任务的worker程序读取相关的输入数据片段，从输入的数据片段中解析出key/value pair，然后把key/value pair传递给用户自定义的Map函数，由Map函数生成并输出的中间key/value pair，并缓存在本机内存中。

4.缓存中的key/value pair 通过分区函数分成R个区域，之后周期性的写入到本地磁盘上。缓存的key/value pair在 本地磁盘上的存储位置将被回传给master，由master负责把这些存储位置再传送给Reduce worker。

5.当Reduce worker程序接收到master程序发来的数据存储位置信息后，Reduce worker使用remote procedure calls从Map worker所在主机的磁盘上读取这些缓存数据。当Reduce worker读取了所需的所有中间数据后，通过对key进行排序后使得具有相同key值的数据聚合在一起。由于许多不同的key值会映射到相同的Reduce任务上，因此必须进行排序，从而保证相同的key的序列依次出现。如果中间数据太大无法在内存中完成排序，那么就要在外部进行排序。

6.Reduce worker程序遍历排序后的中间数据，对于每一个唯一的中间key值， Reduce worker程序将这个key值和它相关的中间value值的集合传递给用户自定义的Reduce函数并计算出最终输出。Reduce函数的输出被追加到所属分区的输出文件。

7.当所有的Map和Reduce任务都完成之后，master唤醒用户程序。在这个时候，在用户程序里的对MapReduce调用才返回。

在成功完成任务之后，MapReduce的输出存放在R个输出文件中。这R个文件对应每个Reduce任务产生的一个输出文件，文件名由用户指定。一般情况下，用户不需要将这R个输出文件合并成一个文件–他们经常把这些文件作为另外一个MapReduce的输入，或者在另外一个可以处理多个分割文件的分布式应用中使用。

**Bigtable**

最后是Bigtable。Bigtable是一个为管理大规模结构化数据而设计的分布式存储系统，可以扩展到PB级数据和上千台服务器。很多google的项目使用Bigtable存储数据，这些应用对Bigtable提出了不同的挑战，比如数据规模的要求、延迟的要求。Bigtable能满足这些多变的要求，为这些产品成功地提供了灵活、高性能的存储解决方案。

就像文件系统需要数据库来存储结构化数据一样，GFS也需要Bigtable来存储结构化数据。BigTable 是建立在 GFS ，Scheduler ，Lock Service 和 MapReduce 之上的。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个左右的 Tablets。底层的架构是：GFS。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。

Bigtable使用GFS来存储数据文件和日志，数据文件采用SSTable格式，它提供了关键字到值的映射关系。Bigtable使用分布式的锁服务Chubby来保证集群中主服务器的唯一性、保存Bigtable数据的引导区位置、发现Tablet服务器并处理Tablet服务器的失效、保存Bigtable的数据模式信息Bigtable看起来像一个数据库，采用了很多数据库的实现策略。但是Bigtable并不支持完整的关系型数据模型；而是为客户端提供了一种简单的数据模型，客户端可以动态地控制数据的布局和格式，并且利用底层数据存储的局部性特征。Bigtable将数据统统看成无意义的字节串，客户端需要将结构化和非结构化数据串行化再存入Bigtable。