链滴

【大数据入门】Hadoop、Hive、Spark 之间是什么关系?

作者：seanlee

原文链接：https://ld246.com/article/1496990993345

来源网站：链滴

<blockquote>
该文章摘自于知乎问答
如何用形象的比喻描述大数据的技术生态？Hadoop、Hive、Spark 之间是什么关系？
回答者： Xiayu Ma
</blockquote>

大数据本身是个很宽泛的概念，Hadoop生态圈(或者泛生圈)基本上都是为了处理超过单机尺度的数据处理而诞生的。你可以把它比作一个厨房所以需要的各种具。锅碗瓢盆，各有各的用处，互相之间又有重合。你可以用汤锅直接当碗吃饭喝汤，你可以用小刀者刨子去皮。但是每个工具有自己的特性，虽然奇怪的组合也能工作，但是未必是最佳选择。
<h3>大数据，首先你要能存的下大数据</h3>
传统的文件系统是单机的，不能横跨不同的机器。HDFS(Hdoop Distributed FileSystem)的设计本质上是为了大量的数据能横跨成百上千台机器，但是你看到是一个文件系统而不是很多文件系统。比如你说我要获取/hdfs/tmp/file1的数据，你引用的是一个文路径，但是实际的数据存放在很多不同的机器上。你作为用户，不需要知道这些，就好比在单机上你关心文件分散在什么磁道什么扇区一样。HDFS为你管理这些数据。
存的下数据之后，你就开始考虑怎么处理数据。虽然HDF可以为你整体管理不同机器上的数据，但是这些数据太大了。一台机器读取成T上P的数据(很大的数哦，比如整个东京热有史以来所有高清电影的大小甚至更大)，一台机器慢慢跑也许需要好几天甚至好周。对于很多公司来说，单机处理是不可忍受的，比如微博要更新24小时热博，它必须在24小时之内完这些处理。那么我如果要用很多台机器处理，我就面临了如何分配工作，如果一台机器挂了如何重启动相应的任务，机器之间如何互相通信交换数据以完成复杂的计算等等。这就是MapReduce / Tez Spark的功能。MapReduce是第一代计算引擎，Tez和Spark是第二代。MapReduce的设计，采用很简化的计算模型，只有Map和Reduce两个计算过程(中间用Shuffle串联)，用这个模型，已经可以理大数据领域很大一部分问题了。
<h3>那什么是Map,什么是Reduce?</spn></h3>
考虑如果你要统计一个巨大的文本文件存储在类似HDFS，你想要知道这个文本里各个词的出现频率。你启动了一个MapReduce程序。Map阶段，几百台机同时读取这个文件的各个部分，分别把各自读到的部分分别统计出词频，产生类似(hello, 12100次)，world，15214次)等等这样的Pair(我这里把Map和Combine放在一起说以便简化);这几百台机器各自产生了如上的集合，然后又有几百台机器启动Reduce处理。Reducer机器A将从Mapper机器收到所以A开头的统计结果，机器B将收到B开头的词汇统计结果(当然实际上不会真的以字母开头做依据，而用函数产生Hash值以避免数据串化。因为类似X开头的词肯定比其他要少得多，而你不希望数据处理个机器的工作量相差悬殊)。然后这些Reducer将再次汇总，(hello，12100)+(hello，12311)+(hello345881)= (hello，370292)。每个Reducer都如上处理，你就得到了整个文件的词频结果。
这看似是个很简单的模型，但很多算法都可以用这个模型述了。
Map+Reduce的简单模型很黄很暴力，虽然好用，但是很重。第二代的Tez和Spark除了内存Cache之类的新feature，本质上来说，是让Map/Reduce模型更用，让Map和Reduce之间的界限更模糊，数据交换更灵活，更少的磁盘读写，以便更方便地描述复算法，取得更高的吞吐量。
有了MapReduce，Tez和Spark之后，程序员发现，MapRduce的程序写起来真麻烦。他们希望简化这个过程。这就好比你有了汇编语言，虽然你几乎什么都能了，但是你还是觉得繁琐。你希望有个更高层更抽象的语言层来描述算法和数据处理流程。于是就有Pig和Hive。Pig是接近脚本方式去描述MapReduce，Hive则用的是SQL。它们把脚本和SQL语言翻

成MapReduce程序，丢给计算引擎去计算，而你就从繁琐的MapReduce程序中解脱出来，用更简更直观的语言去写程序了。
有了Hive之后，人们发现SQL对比Java有巨大的优势。一是它太容易写了。刚才词频的东西，用SQL描述就只有一两行，MapReduce写起来大约要几十上百。而更重要的是，非计算机背景的用户终于感受到了爱：我也会写SQL!于是数据分析人员终于从乞求程师帮忙的窘境解脱出来，工程师也从写奇怪的一次性的处理程序中解脱出来。大家都开心了。Hive渐成长成了大数据仓库的核心组件。甚至很多公司的流水线作业集完全是用SQL描述，因为易写易改一看就懂，容易维护。
自从数据分析人员开始用Hive分析数据之后，它们发现，Hive在MapReduce上跑，真鸡巴慢!流水线作业集也许没啥关系，比如24小时更新的推荐，反正24小内跑完就算了。但是数据分析，人们总是希望能跑更快一些。比如我希望看过去一个小时内多少人在气娃娃页面驻足，分别停留了多久，对于一个巨型网站海量数据下，这个处理过程也许要花几十分钟至很多小时。而这个分析也许只是你万里长征的第一步，你还要看多少人浏览了跳蛋多少人看了拉赫尼诺夫的CD，以便跟老板汇报，我们的用户是猥琐男闷骚女更多还是文艺青年/少女更多。你无法忍等待的折磨，只能跟帅帅的工程师蝈蝈说，快，快，再快一点!
于是Impala，Presto，Drill诞生了(当然还有无数非著名交互SQL引擎，就不一一列举了)。三个系统的核心理念是，MapReduce引擎太慢，因为它太通用，强壮，太保守，我们SQL需要更轻量，更激进地获取资源，更专门地对SQL做优化，而且不需要那么容错性保证(因为系统出错了大不了重新启动任务，如果整个处理时间更短的话，比如几分钟之内)。些系统让用户更快速地处理SQL任务，牺牲了通用性稳定性等特性。如果说MapReduce是大砍刀，啥都不怕，那上面三个就是剔骨刀，灵巧锋利，但是不能搞太大太硬的东西。
这些系统，说实话，一直没有达到人们期望的流行度。因这时候又两个异类被造出来了。他们是Hive on Tez / Spark和SparkSQL。它们的设计理念是，MapReduce慢，但是如果我用新一代通用计算引擎Tez或者Spark来跑SQL，那我就能跑的更快。而且用户需要维护两套系统。这就好比如果你厨房小，人又懒，对吃的精细程度要求有限，那你可以买个电饭，能蒸能煲能烧，省了好多厨具。
上面的介绍，基本就是一个数据仓库的构架了。底层HDF，上面跑MapReduce/Tez/Spark，在上面跑Hive，Pig。或者HDFS上直接跑Impala，Drill，Prest。这解决了中低速数据处理的要求。
<h3>那如果我要更高速的处理呢?</spa></h3>
如果我是一个类似微博的公司，我希望显示不是24小时热，我想看一个不断变化的热播榜，更新延迟在一分钟之内，上面的手段都将无法胜任。于是又一种计模型被开发出来，这就是Streaming(流)计算。Storm是最流行的流计算平台。流计算的思路是，如要达到更实时的更新，我何不在数据流进来的时候就处理了?比如还是词频统计的例子，我的数据流一个一个的词，我就让他们一边流过我就一边开始统计了。流计算很牛逼，基本无延迟，但是它的短是，不灵活，你想要统计的东西必须预先知道，毕竟数据流过就没了，你没算的东西就无法补算了。此它是个很好的东西，但是无法替代上面数据仓库和批处理系统。
还有一个有些独立的模块是KV Store，比如Cassandra，Hase，MongoDB以及很多很多很多很多其他的(多到无法想象)。所以KV Store就是说，我有一堆键值我能很快速滴获取与这个Key绑定的数据。比如我用身份证号，能取到你的身份数据。这个动作用MaReduce也能完成，但是很可能要扫描整个数据集。而KV Store专用来处理这个操作，所有存和取都门为此优化了。从几个P的数据中查找一个身份证号，也许只要零点几秒。这让大数据公司的一些专操作被大大优化了。比如我网页上有个根据订单号查找订单内容的页面，而整个网站的订单数量无法机数据库存储，我就会考虑用KV Store来存。KV Store的理念是，基本无法处理复杂的计算，大多没JOIN，也许没法聚合，没有强一致性保证(不同数据分布在不同机器上，你每次读取也许会读到不同结果，也无法处理类似银行转账那样的强一致性要求的操作)。但是丫就是快。极快。
每个不同的KV Store设计都有不同取舍，有些更快，有些量更高，有些可以支持更复杂的操作。必有一款适合你。
除此之外，还有一些更特制的系统/组件，比如Mahout是布式机器学习库，Protobuf是数据交换的编码和库，ZooKeeper是高一致性的分布存取协同系统，等。
有了这么多乱七八糟的工具，都在同一个集群上运转，大需要互相尊重有序工作。所以另外一个重要组件是，调度系统。现在最流行的是Yarn。你可以把他看

中央管理，好比你妈在厨房监工，哎，你妹妹切菜切完了，你可以把刀拿去杀鸡了。只要大家都服从妈分配，那大家都能愉快滴烧菜。
你可以认为，大数据生态圈就是一个厨房工具生态圈。为做不同的菜，中国菜，日本菜，法国菜，你需要各种不同的工具。而且客人的需求正在复杂化，你的具不断被发明，也没有一个万用的厨具可以处理所有情况，因此它会变的越来越复杂。
End.

原文链接：【大数据入门】Hadoop、Hive、Spark 之间是什么关系?