关于大数据的五问五答

本文出自本公众号ChinaScala，由陈超所述。

一、Spark能否取代Hadoop？

答： Hadoop包含了Common,HDFS,YARN及MapReduce，Spark从来没说要取代Hadoop，最多也就是取代掉MapReduce。事实上现在Hadoop已经发展成为一个生态系统，并且Hadoop生态系统也接受更多优秀的框架进来，如Spark
(Spark可以和HDFS无缝结合，并且可以很好的跑在YARN上).。 另一方面，Spark除了跟Hadoop生态结合外，也得到了其它一些框架的支持如ElasticSearch及Cassandra等等。
所以Hadoop生态并不是使用Spark的先决条件，尽管Spark非常好的融入了Hadoop生态。

二、谈谈Flink，跟Spark比较下？

答： 首先作为Spark在国内的布道者，我必须承认，我非常早的就关注了Flink :)
目前Flink在欧洲的知名度还是可以的。 Flink跟Spark一样，都是希望做一体化的计算引擎(批处理，流处理，机器学习，图计算等)，并且他们都能很好的融入Hadoop生态(如跟HDFS无缝结合，及支持YARN作为调度框架等)。 咋一看两者略相似，但其实他们走的路子还是不太一样的，在Spark推出之际，就强调了“内存计算”，而Flink走的其实是类似MPP的路子。另一方面，Flink宣称能真正意义上做到实时计算，而Spark只能做micro batch。不过Flink支持的增量迭代还是挺有意思的，它能在迭代过程中只去处理那些变化的数据，事实上迭代到后面的时候，Flink只需要处理一小部分子集而已。不过以上都不是我最初关注Flink的原因，我当初关注Flink的原因是它对内存管理方面有着独到之处。 Flink一开始就决定自己做内存管理，它把heap分为三个部分: Network buffers, Memory Manager pool及Remaing
heap，具体不展开了，有兴趣的可以字节去查看下相关资料。当然Spark也使出了杀手锏: project tungsten，
俗称钨丝计划，除了做自己的内存管理，也会做其它非常强悍的优化。这些优化将在1.4, 1.5, 1.6三个版本中体现。 哦，最后提一下，Spark只支持DAG，而Flink是支持cyclic的。
这个话题就先到这，后期很可能会来篇专门的文章。

三、谈谈你对HBase及Cassandra的看法？

答：首先，一些国内的工程师对Cassandra有着莫名其妙的误解，以为当年Facebook“抛弃”它，它就不行了，对此只能先呵呵。 我个人其实用HBase比较多，但是过去的一段时间，并没有用HBase(最多也就一年多前作为OpenTSDB的后端跑了下)。 HBase和Cassandra在很多地方还是很相似的。一、都是面向列的存储；二、都会先写到Log中，然后进入内存中的存储结构，
最后刷盘，甚至所用数据结构都差不多：LSM。简单来讲，从HBase的角度就是 Data到HLog到Memstor到StoreFile(HFile)；从Cassandra的角度就是Data到CommitLog到memtable到sstable。
三、略，太多了。 那我们还是看看不一样的地方吧，HBase需要ZK支持，Cassandra自给自足；HBase需要有Master，Cassandra需要有seed nodes； HBase要从ZK获取信息，Cassandra使用gossip来通信；HBase底层要有HDFS支持，Cassandra不用；HBase原生不支持二级索引，Cassan

dra支持；HBase老早就有coprocessor，Cassandra木有……不一一列了吧。 最大不同点还是HBase本质上还是一个中心化的组织(peer-to-
peer)，而Cassandra是去中心化的，我可能会更偏向后者。
目死很难说孰优孰劣，你的选型你做主：)

四、分布式消息队列在流处理系统中是十分重要的，你选择的消息队列是哪个？能否简述下原因？

答：毫无疑问 Kafka! 最多前面加个Flume。 任何选型的原因，都源自你的需求是什么。Fast,Scalable,Durable是我的需求，Kafka完美满足。稍微讲些细节，好多想必大家也都知道。Kafka将数据写到磁盘，实际上都会写到OS的page cache里， 而读的时候又用sendfile非常高效的将数据传输到NIC。Kafka的扩展性也非常好，只要增加broker即可。Kafka的逻辑也非常清晰，可以将不同业务逻辑的数据写进不同到topic，而topic又可以切分成若干个partition来并行处理，并且Kafka0.9后，zk只需要被broker所使用，consumer并不再需要使用zk来记录offset，大大降低zk的压力，同时也从侧面降低了scale的压力。Kafka也有比较友好的删除策略。可以直接按照max age或者max size自动删除，也可以按照key进行compact，基本上都能满足需求。另一方面，Kafka的社区非常活跃，并且现在几乎所有流行的(流式)计算框架都支持Kafka，如Spark Streaming，Storm等。对了，有个叫camus的工具定期可以将Kafka里的数据搬到HDFS上，已经推荐一些小伙伴用过5 还是那句话，看需求，且能hold住。

五、 你对Tachyon的看法如何？

答：我是非常早就试用了Tachyon，注意，是试用！ 现在已经有商业公司TachyonNexus为其保驾护航了。哦，对了，先说明下，Tachyon是用Java写的，并不是用Scala写的。我本人对Tachyon的前景非常看好，说了半天好像还没说Tachyon是个什么玩意儿。Tachyon是分布式内存文件系统，随着内存越来越廉价，只要Tachyon本身质量过硬，显然不愁用户。稍微简单谈下Tachyon的原理及特点吧，作为基于内存的文件系统，那显然要非常激进的使用内存了，那这时候就会有人担心了，node挂了内存里数据丢失怎么办，这个其实不用担心，在Tachyon下面一般还会有一层underlying filesystem，大多数情况下是HDFS，当然也支持其它一些文件系统，
Tachyon定期会把数据checkpoint到underlying filesystem里。事实上Tachyon也有Journal(image + edit)的概念，非常有意思的是，Tachyon把Spark里lineage的理念搬了过来，依靠lineage做文件恢复。前面说到，现在出来一个framework，不跟Hadoop生态结合是很难混的，所以Tachyon也非常友好的实现了HDFS的接口，因此MapReduce及Spark，包括Flink都可以在几乎不改动代码的情况下使用Tachyon。Tachyon另一个出色的点是支持table，用户可以把查询密度高的column放进Tachyon，以此提高查询效率。再补充几个Tachyon可以解决的case：Spark的不同job间共享数据；不同框架间共享数据；避免Spark由于blockmanager挂掉后cache全丢失；解决内存重复使用的问题，等。 这个问题就此打住，不展开了。