Apache Spark相比Hadoop的优势

以下的话是由Apache Spark committer的Reynold Xin阐述。

从很多方面来讲，Spark都是MapReduce 模式的最好实现。比如从程序抽象的角度来看：

1、他抽象出Map/Reduce两个阶段来支持tasks的任意DAG。大多数计算通过依赖将maps和reduces映射到一起(Most computation maps (no pun intended) into many maps and reduces with dependencies among them.)。而在Spark的RDD编程模型中，将这些依赖弄成DAG。通过这种方法，更自然地表达出计算逻辑。

2、通过更好的语言来集成到模型中的数据流，他抛弃了Hadoop MapReduce中要求的大量样板代码。通常情况下，当你看一个的Hadoop MapReduce的程序，你很难抽取出这个程序需要做的事情，因为 the huge amount of boiler plates，而你阅读Spark程序的时候你会感觉到很自然。（这段翻译起来很别扭，请参见下面原文）

> Through better language integration to model data flow, it does away with the huge amount of boilerplate code required in Hadoop MapReduce. Typically when you look at a Hadoop MapReduce program, it is difficult to extract what it attempts to do because of the huge amount of boilerplates, whereas it is much more natural to read a Spark program.

3. 由于Spark的灵活编程模型，Hadoop MapReduce中必须和嵌入的操作现在直接在应用程序的环境中。也就是应用程序可以重写shuffle或者aggregation 函数的实现方式。而这在MapReduce是不可能的！虽然不是绝大部分的应用程序会重写这些方法，但是这种机制可以使得某些人基于特定的场景来重写相关的函数，从而使得计算得到最优。

4. 最后，应用程序可以将数据集缓存到集群的内存中。这种内置的机制其实是很多应用程序的基础，这些应用程序在短时间内需要多次方法访问这些数据集，比如在机器学习算法中。

从系统的高层次来看：

1、Spark通过快速的RPCs 方式来调度作业

2、Spark在线程池中来运行task，而不是一系列的JVM进程。上面两个计算结合起来，使得Spark可以在毫秒级别的时间内调度task。然而在MP调度模型中，需要花费数秒甚至是数分钟（繁忙的集群）来调度task。

3、Spark不仅支持基于checkpointing(checkpointing-based)的容错(这种方式也是Hadoop MP采用的)，也支持基于血统(lineage-based)的容错机制。错误是很常见的，基于血统(lineage-based)的容错机制可以快速地从失败者恢复！

4、部分也是由于学术方面的原因，Spark社区常常有新的思维，其中一个例子就是，在Spark中采用BT协议来广播数据。