**用Apache Spark进行大数据处理**

## Hadoop和Spark

Hadoop这项大数据处理技术大概已有十年历史，而且被看做是首选的大数据集合处理的解决方案。MapReduce是一路计算的优秀解决方案，不过对于需要多路计算和算法的用例来说，并非十分高效。数据处理流程中的每一步都需要一个Map阶段和一个Reduce阶段，而且如果要利用这一解决方案，需要将所有用例都转换成MapReduce模式。

在下一步开始之前，上一步的作业输出数据必须要存储到分布式文件系统中。因此，复制和磁盘存储会导致这种方式速度变慢。另外Hadoop解决方案中通常会包含难以安装和管理的集群。而且为了处理不同的大数据用例，还需要集成多种不同的工具（如用于机器学习的Mahout和流数据处理的Storm）。

如果想要完成比较复杂的工作，就必须将一系列的MapReduce作业串联起来然后顺序执行这些作业。每一个作业都是高时延的，而且只有在前一个作业完成之后下一个作业才能开始启动。

而Spark则允许程序开发者使用有向无环图（[DAG](http://en.wikipedia.org/wiki/Directed_acyclic_graph)）开发复杂的多步数据管道。而且还支持跨有向无环图的内存数据共享，以便不同的作业可以共同处理同一个数据。

Spark运行在现有的Hadoop分布式文件系统基础之上（[HDFS](http://wiki.apache.org/hadoop/HDFS)）提供额外的增强功能。它支持[将Spark应用部署到](http://databricks.com/blog/2014/01/21/Spark-and-Hadoop.html)现存的Hadoop v1集群（with SIMR – Spark-Inside-MapReduce）或Hadoop v2 YARN集群甚至是[Apache Mesos](http://mesos.apache.org/)之中。

我们应该将Spark看作是Hadoop MapReduce的一个替代品而不是Hadoop的替代品。其意图并非是替代Hadoop，而是为了提供一个管理不同的大数据用例和需求的全面且统一的解决方案。

## Spark特性

Spark通过在数据处理过程中成本更低的洗牌（Shuffle）方式，将MapReduce提升到一个更高的层次。利用内存数据存储和接近实时的处理能力，Spark比其他的大数据处理技术的性能要快很多倍。

Spark还支持大数据查询的延迟计算，这可以帮助优化大数据处理流程中的处理步骤。Spark还提供高级的API以提升开发者的生产力，除此之外还为大数据解决方案提供一致的体系架构模型。

Spark将中间结果保存在内存中而不是将其写入磁盘，当需要多次处理同一数据集时，这一点特别实用。Spark的设计初衷就是既可以在内存中又可以在磁盘上工作的执行引擎。当内存中的数据不适用时，Spark操作符就会执行外部操作。Spark可以用于处理大于集群内存容量总和的数据集。

Spark会尝试在内存中存储尽可能多的数据然后将其写入磁盘。它可以将某个数据集的一部分存入内存而剩余部分存入磁盘。开发者需要根据数据和用例评估对内存的需求。Spark的性能优势得益于这种内存中的数据存储。

Spark的其他特性包括：

* 支持比Map和Reduce更多的函数。
* 优化任意操作算子图（operator graphs）。
* 可以帮助优化整体数据处理流程的大数据查询的延迟计算。
* 提供简明、一致的Scala，Java和Python API。
* 提供交互式Scala和Python Shell。目前暂不支持Java。

## Spark生态系统

除了Spark核心API之外，Spark生态系统中还包括其他附加库，可以在大数据分析和机器学习领域提供更多的能力。

这些库包括：

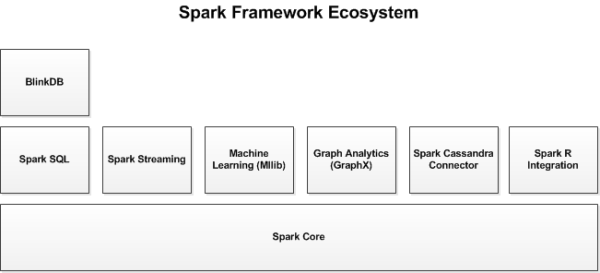
* **Spark Streaming:**
  + [Spark Streaming](https://spark.apache.org/streaming/)基于微批量方式的计算和处理，可以用于处理实时的流数据。它使用DStream，简单来说就是一个弹性分布式数据集（RDD）系列，处理实时数据。
* **Spark SQL:**
  + [Spark SQL](https://spark.apache.org/sql/)可以通过JDBC API将Spark数据集暴露出去，而且还可以用传统的BI和可视化工具在Spark数据上执行类似SQL的查询。用户还可以用Spark SQL对不同格式的数据（如JSON，Parquet以及数据库等）执行ETL，将其转化，然后暴露给特定的查询。
* **Spark MLlib:**
  + [MLlib](https://spark.apache.org/mllib/)是一个可扩展的Spark机器学习库，由通用的学习算法和工具组成，包括二元分类、线性回归、聚类、协同过滤、梯度下降以及底层优化原语。
* **Spark GraphX:**
  + [GraphX](https://spark.apache.org/graphx/)是用于图计算和并行图计算的新的（alpha）Spark API。通过引入弹性分布式属性图（Resilient Distributed Property Graph），一种顶点和边都带有属性的有向多重图，扩展了Spark RDD。为了支持图计算，GraphX暴露了一个基础操作符集合（如subgraph，joinVertices和aggregateMessages）和一个经过优化的Pregel API变体。此外，GraphX还包括一个持续增长的用于简化图分析任务的图算法和构建器集合。

除了这些库以外，还有一些其他的库，如BlinkDB和Tachyon。

[BlinkDB](http://blinkdb.org/)是一个近似查询引擎，用于在海量数据上执行交互式SQL查询。BlinkDB可以通过牺牲数据精度来提升查询响应时间。通过在数据样本上执行查询并展示包含有意义的错误线注解的结果，操作大数据集合。

[Tachyon](http://tachyon-project.org/index.html)是一个以内存为中心的分布式文件系统，能够提供内存级别速度的跨集群框架（如Spark和MapReduce）的可信文件共享。它将工作集文件缓存在内存中，从而避免到磁盘中加载需要经常读取的数据集。通过这一机制，不同的作业/查询和框架可以以内存级的速度访问缓存的文件。  
此外，还有一些用于与其他产品集成的适配器，如Cassandra（[Spark Cassandra 连接器](http://www.datastax.com/dev/blog/accessing-cassandra-from-spark-in-java)）和R（SparkR）。Cassandra Connector可用于访问存储在Cassandra数据库中的数据并在这些数据上执行数据分析。

下图展示了在Spark生态系统中，这些不同的库之间的相互关联。



**图1. Spark框架中的库**

我们将在这一系列文章中逐步探索这些Spark库

## 弹性分布式数据集

[弹性分布式数据集](https://spark.apache.org/docs/latest/programming-guide.html#resilient-distributed-datasets-rdds)（基于Matei的[研究论文](https://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf)）或RDD是Spark框架中的核心概念。可以将RDD视作数据库中的一张表。其中可以保存任何类型的数据。Spark将数据存储在不同分区上的RDD之中。

RDD可以帮助重新安排计算并优化数据处理过程。

此外，它还具有容错性，因为RDD知道如何重新创建和重新计算数据集。

RDD是不可变的。你可以用变换（Transformation）修改RDD，但是这个变换所返回的是一个全新的RDD，而原有的RDD仍然保持不变。

RDD支持两种类型的操作：

* 变换（Transformation）
* 行动（Action）

**变换(Transformations)：**[变换](https://spark.apache.org/docs/latest/programming-guide.html" \l "transformations)的返回值是一个新的RDD集合，而不是单个值。调用一个变换方法，不会有任何求值计算，它只获取一个RDD作为参数，然后返回一个新的RDD。

变换函数包括：map，filter，flatMap，groupByKey，reduceByKey，aggregateByKey，pipe和coalesce。

**行动(Actions)：**[行动](https://spark.apache.org/docs/latest/programming-guide.html" \l "actions)操作计算并返回一个新的值。当在一个RDD对象上调用行动函数时，会在这一时刻计算全部的数据处理查询并返回结果值。

行动操作包括：reduce，collect，count，first，take，countByKey以及foreach。