**大数据分析进阶**

[**（资源整理）带你入门Spark**](https://www.cnblogs.com/shiyanlou/p/6183389.html)

**一、Spark简介：**

以下是百度百科对Spark的介绍：

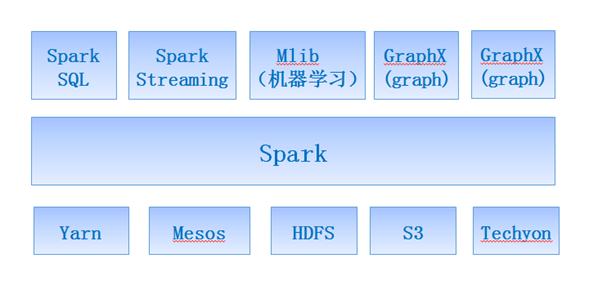
Spark 是一种与 Hadoop 相似的开源集群计算环境，但是两者之间还存在一些不同之处，这些有用的不同之处使 Spark 在某些工作负载方面表现得更加优越，换句话说，Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。

Spark 是在 Scala 语言中实现的，它将 Scala 用作其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。

**二、Spark生态圈介绍**

Spark力图整合机器学习（MLib）、图算法（GraphX）、流式计算（Spark Streaming）和数据仓库（Spark SQL）等领域，通过计算引擎Spark，弹性分布式数据集（RDD），架构出一个新的大数据应用平台。

Spark生态圈以HDFS、S3、Techyon为底层存储引擎，以Yarn、Mesos和Standlone作为资源调度引擎；使用Spark，可以实现MapReduce应用；基于Spark，Spark SQL可以实现即席查询，Spark Streaming可以处理实时应用，MLib可以实现机器学习算法，GraphX可以实现图计算，SparkR可以实现复杂数学计算。



**三、Spark教程**

有很多想要学习Spark的小伙伴都是自学的，但是网上的教程太多太杂太零散，其实并不适合一个Spark小白的人学习，而我们实验楼刚好又有一些系列的教程，因此整理出来，希望对Spark学习者有所帮助~

我们就按照上图的生态圈，从左到右的顺序介绍课程吧；

**1、**[**Spark 讲堂之 SQL 入门**](https://www.shiyanlou.com/courses/586)

Spark SQL 是一个分布式查询引擎，在这个教程里你可以学习到 Spark SQL 的基础知识和常用 API 用法，了解常用的数学和统计函数。最后将通过一个分析股票价格与石油价格关系的实例进一步学习如何利用 Spark SQL 分析数据。

**2、**[**Spark 讲堂之 Streaming 入门**](https://www.shiyanlou.com/courses/571)

Spark Streaming 适用于实时处理流式数据。该教程带你学习 Spark Streaming 的工作机制，了解 Streaming 应用的基本结构，以及如何在 Streaming 应用中附加 SQL 查询。

附带一张Streaming图：



**3、**[**Spark 讲堂之 MLlib 入门**](https://www.shiyanlou.com/courses/600)

这个教程你可以了解到 Spark 的 MLlib 库相关知识，掌握 MLlib 的几个基本数据类型，并且可以动手练习如何通过机器学习中的一些算法来推荐电影。

**4、**[**Spark 讲堂之 GraphX 入门**](https://www.shiyanlou.com/courses/529)

GraphX是Spark用于解决图和并行图计算问题的新组件。GraphX通过RDD的扩展，在其中引入了一个新的图抽象，即顶点和边带有特性的有向多重图，提供了一些基本运算符和优化了的Pregel API，来支持图计算。

**5、**[**Spark 讲堂之 GraphX 图算法**](https://www.shiyanlou.com/courses/534)

GraphX包含了一些用于简化图分析任务的的图计算算法。你可以通过图操作符来直接调用其中的方法。这个教程中讲解这些算法的含义，以及如何实现它们。

**6、**[**Spark 讲堂之 SparkR 入门**](https://www.shiyanlou.com/courses/615)

SparkR是一个提供轻量级前端的R包，集成了Spark的分布式计算和存储等特性。这个教程将以较为轻松的方式带你学习如何在SparkR中创建和操作DataFrame，如何应用SQL查询和机器学习算法等。

**7、**[**Spark 讲堂之 DataFrame 入门**](https://www.shiyanlou.com/courses/536)

DataFrame让Spark具备了处理大规模结构化数据的能力，在比原有的RDD转化方式更加易用、计算性能更好。这个教程通过一个简单的数据集分析任务，讲解DataFrame的由来、构建方式以及一些常用操作。

**8、**[**Spark 讲堂之 DataFrame 详解**](https://www.shiyanlou.com/courses/543)

这个教程通过更加深入的讲解，使用真实的数据集，并结合实际问题分析过程作为引导，旨在让Spark学习者掌握DataFrame的高级操作技巧，如创建DataFrame的两种方式、UDF等。

**9、**[**Sqoop 数据迁移工具**](https://www.shiyanlou.com/courses/575)

Sqoop 是大数据环境中重要的是数据转换工具，这个教程对Sqoop 的安装配置进行了详细的讲解，并列举了Sqoop 在数据迁移过程中基本操作指令。

以上9个教程比较适合有一定的Spark基础的人学习。

**10、**[**Spark 大数据动手实验**](https://www.shiyanlou.com/courses/456)

这个教程是一个系统性的教程，总共15个小节，带你亲身体验Spark大数据分析的魅力，课程中可以实践：  
Spark，Scala，Python，Spark Streaming，SparkSQL，MLlib，GraphX，IndexedRDD，SparkR，Tachyon，KeystoneML，BlinkDB等技术点，无疑是学习Spark最快的上手教程！

这个教程较为系统，非常适合零基础的人进行学习。

**最后**

希望以上10个教程可以帮助想入门Spark的人，入门之后，你自然会知道如何让自己的技术更上一层楼，也自然会有意无意去收集整理Spark学习资源和资料，因此这里就不多介绍了。

[**spark**](https://www.cnblogs.com/fanshudada/p/10958714.html)

**1.spark 开源的分布式cluster(集群)运算框架**

**2.spark相对于Hadoop的优势**

* 数据处理逻辑的代码非常简短
* 提供很多转换和动作,而hadoop只提供Map和Reduce,表达力欠缺
* 一个job可以包含多个转换操作,在调度时可以生成多个stage,多个map操作的RDD分区不变,可以放在同一个task中进行,而hadoop一个job只有map和reduce,复杂计算需要大量job完成
* 逻辑更清晰,可以提供处理逻辑整体视图,而hadoop没有整体逻辑
* spark运算中间结果放在内存中,内存放不下会写入本地磁盘,hadoop中间结果放在hdfs中
* 可以处理流数据和交互式数据,Hadoop时延高,只适用批处理
* 通过内存中缓存数据,迭代式计算能力强

**3.spark生态**

* spark core API:支持python,java,scala,sql等语言,批处理计算
* spark SQL: 支持用sql查询,交互式计算
* streaming: 数据流式处理,相对批量处理速度更快
* MLlib(machine learning): 较少量数据反复迭代运算,机器学习
* graphx: 基于图运算引擎,适用于图运算

**4.spark核心组成部分**

* driver: 写的程序本身
* cluster manager: 分配调度任务
* worker node: 执行计算任务
* executor: worker node中执行任务的角色, 相当与进程, 一个worker node 中可以有多个executor

**5.job：**一个application由多个job组成,与hadoop不同每个action都会产生一个job

**6.stage：**一个job分为多个stage,stage是按照数据计算的边界划分

**7.task：**一个stage可以有多个task,一个task是一个真正的计算任务,是计算任务的最小单位

**8.spark数据来源**

* 本地文件csv,json,txt
* hdfs
* hbase
* MongoDB
* 传统数据库等
* 云端数据S3

**9.RDD-spark核心数据结构**

RDD横跨整个集群,系统自动分隔,partition可以并行计算

RDD是只读的且不可改变

RDD只能从数据源读进来或者从其他RDD转换过来

RDD action计算方法: collect,count,countByValue,reduce,top

重要的RDD可以存档，可以配置保存的位置

**10.大数据最耗时其实是IO操作，spark数据本地化，尽量减少数据移动，提高了处理速度，** 所有开发时尽量考虑数据的位置，尽量减少数据的移动

**11.Lambda架构：**既有批处理又有流式处理

**12.apark stream---mini batch：**比流式处理容易迁移，提供秒级实时计算，对延时要求高的（毫秒级）计算不太适合

[Spark学习之路 （一）Spark初识](https://www.cnblogs.com/qingyunzong/p/8886338.html)

**目录**

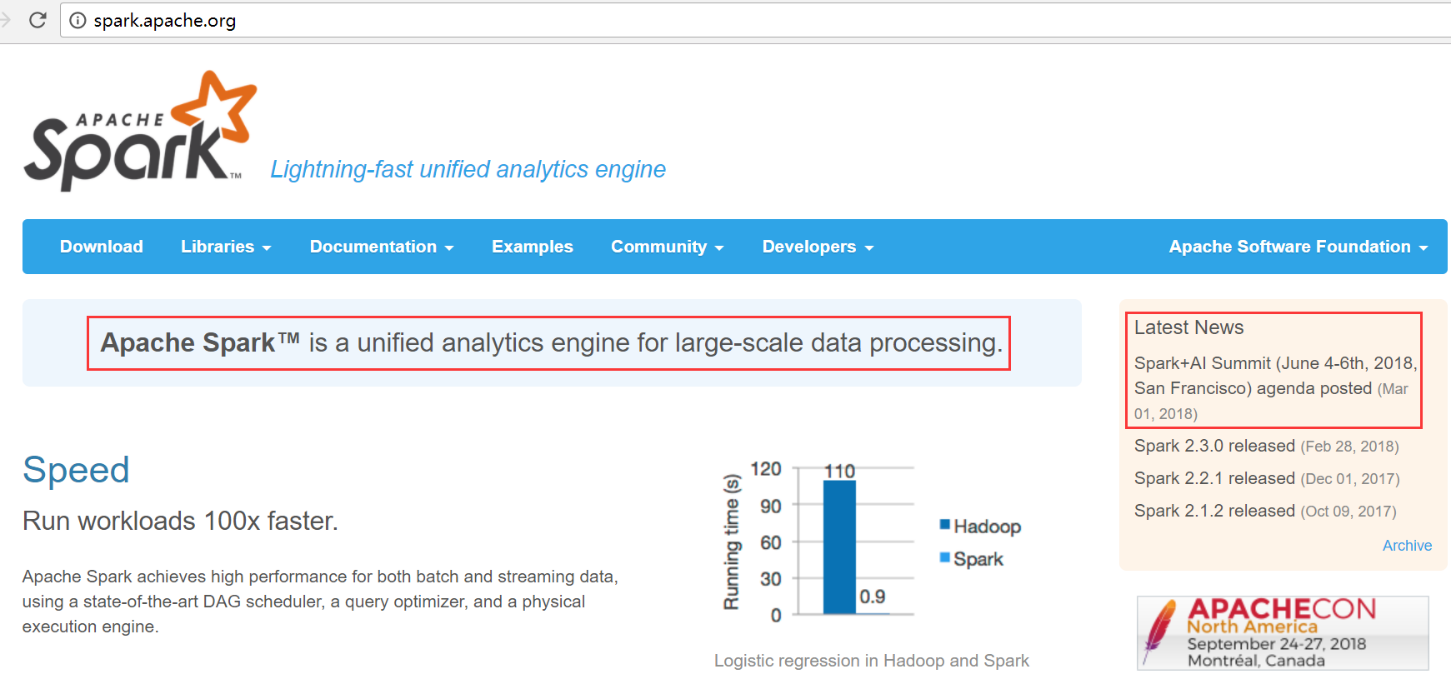
* [一、官网介绍](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label0)
  + [1、什么是Spark](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label0_0)
* [二、Spark的四大特性](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label1)
  + [1、高效性](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label1_0)
  + [2、易用性](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label1_1)
  + [3、通用性](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label1_2)
  + [4、兼容性](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label1_3)
* [三、Spark的组成](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label2)
* [四、应用场景](https://www.cnblogs.com/qingyunzong/p/8886338.html#_label3)

**正文**

**一、官网介绍**

**1、什么是Spark**

官网地址：<http://spark.apache.org/>



**Apache Spark™**是用于大规模数据处理的统一分析引擎。

从右侧最后一条新闻看，Spark也用于AI人工智能

spark是一个实现快速通用的集群计算平台。它是由加州大学伯克利分校AMP实验室 开发的通用内存并行计算框架，用来构建大型的、低延迟的数据分析应用程序。它扩展了广泛使用的MapReduce计算

模型。高效的支撑更多计算模式，包括交互式查询和流处理。spark的一个主要特点是能够在内存中进行计算，及时依赖磁盘进行复杂的运算，Spark依然比MapReduce更加高效。

**2、为什么要学Spark**

**中间结果输出**：基于MapReduce的计算引擎通常会将中间结果输出到磁盘上，进行存储和容错。出于任务管道承接的，考虑，当一些查询翻译到MapReduce任务时，往往会产生多个Stage，而这些串联的Stage又依赖于底层文件系统（如HDFS）来存储每一个Stage的输出结果。

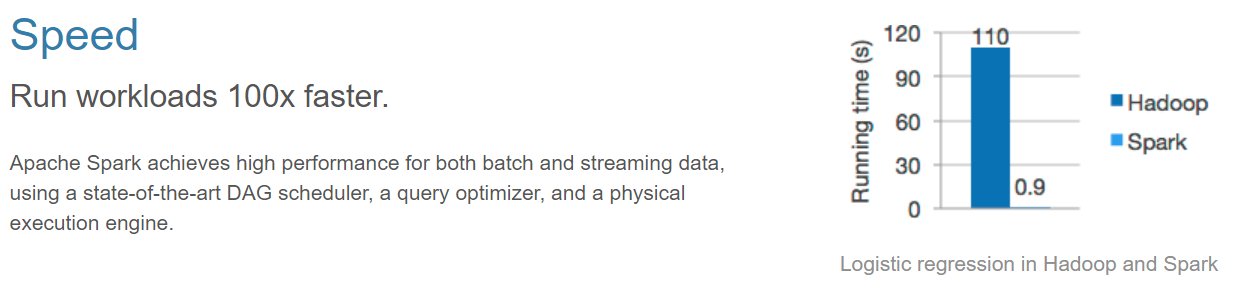
**Spark是MapReduce的替代方案，而且兼容HDFS、Hive，可融入Hadoop的生态系统，以弥补MapReduce的不足。**

**二、Spark的四大特性**

**1、高效性**

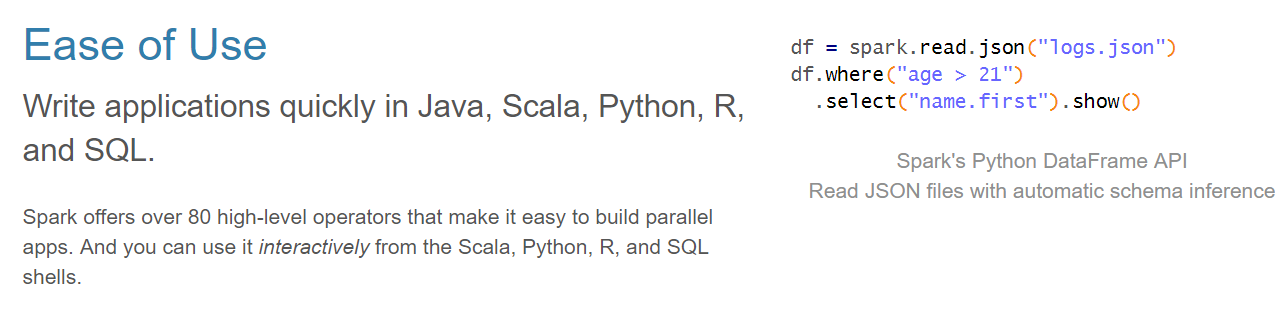
运行速度提高100倍。

Apache Spark使用最先进的DAG调度程序，查询优化程序和物理执行引擎，实现批量和流式数据的高性能。



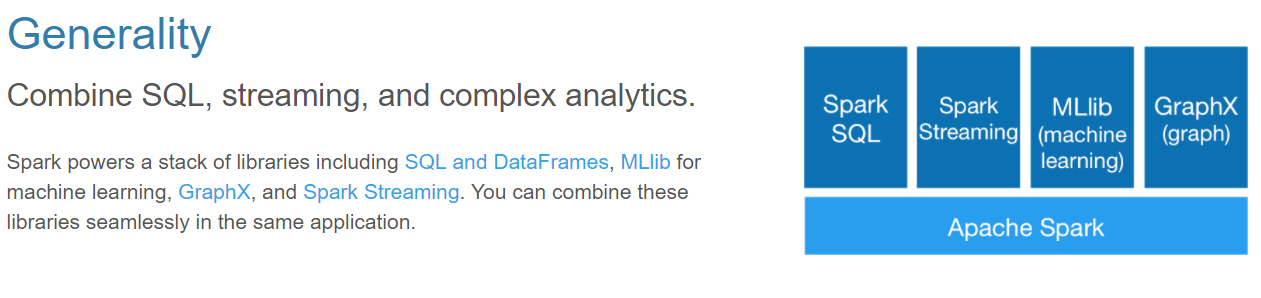
**2、易用性**

Spark支持Java、Python和Scala的API，还支持超过80种高级算法，使用户可以快速构建不同的应用。而且Spark支持交互式的Python和Scala的shell，可以非常方便地在这些shell中使用Spark集群来验证解决问题的方法。



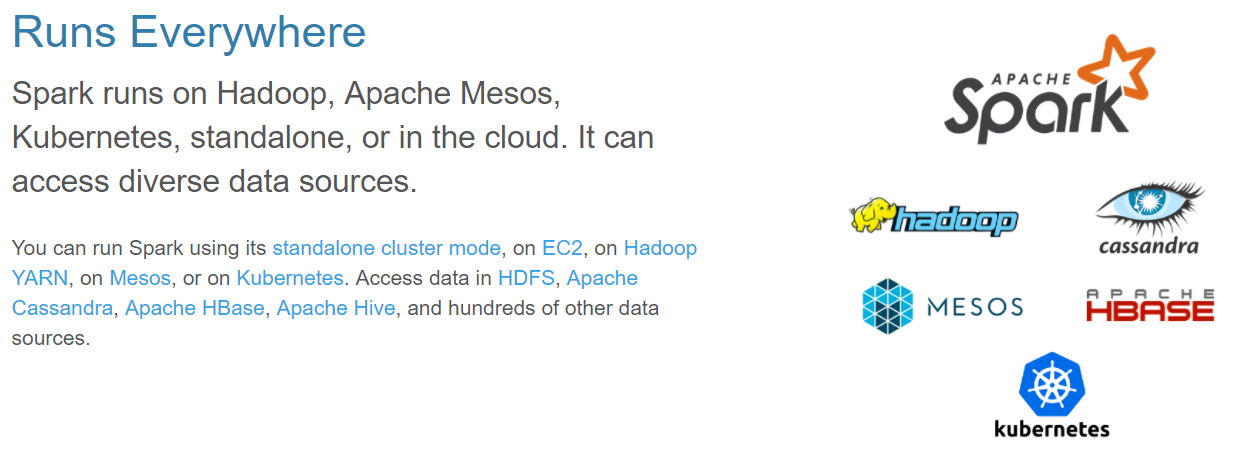
**3、通用性**

Spark提供了统一的解决方案。Spark可以用于批处理、交互式查询（Spark SQL）、实时流处理（Spark Streaming）、机器学习（Spark MLlib）和图计算（GraphX）。这些不同类型的处理都可以在同一个应用中无缝使用。Spark统一的解决方案非常具有吸引力，毕竟任何公司都想用统一的平台去处理遇到的问题，减少开发和维护的人力成本和部署平台的物力成本。



**4、兼容性**

Spark可以非常方便地与其他的开源产品进行融合。比如，Spark可以使用Hadoop的YARN和Apache Mesos作为它的资源管理和调度器，器，并且可以处理所有Hadoop支持的数据，包括HDFS、HBase和Cassandra等。这对于已经部署Hadoop集群的用户特别重要，因为不需要做任何数据迁移就可以使用Spark的强大处理能力。Spark也可以不依赖于第三方的资源管理和调度器，它实现了Standalone作为其内置的资源管理和调度框架，这样进一步降低了Spark的使用门槛，使得所有人都可以非常容易地部署和使用Spark。此外，Spark还提供了在EC2上部署Standalone的Spark集群的工具。



[Mesos](https://mesos.apache.org/)：Spark可以运行在Mesos里面（Mesos 类似于yarn的一个资源调度框架）

[standalone](http://spark.apache.org/docs/latest/spark-standalone.html)：Spark自己可以给自己分配资源（master，worker）

[YARN](https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html)：Spark可以运行在yarn上面

[Kubernetes](https://kubernetes.io/)：Spark接收 [Kubernetes](https://kubernetes.io/)的资源调度

**三、Spark的组成**

Spark组成(BDAS)：全称伯克利数据分析栈，通过大规模集成算法、机器、人之间展现大数据应用的一个平台。也是处理大数据、云计算、通信的技术解决方案。

它的主要组件有：

**SparkCore**：将分布式数据抽象为弹性分布式数据集（RDD），实现了应用任务调度、RPC、序列化和压缩，并为运行在其上的上层组件提供API。

**SparkSQL**：Spark Sql 是Spark来操作结构化数据的程序包，可以让我使用SQL语句的方式来查询数据，Spark支持 多种数据源，包含Hive表，parquest以及JSON等内容。

**SparkStreaming**： 是Spark提供的实时数据进行流式计算的组件。

**MLlib**：提供常用机器学习算法的实现库。

**GraphX**：提供一个分布式图计算框架，能高效进行图计算。

**BlinkDB**：用于在海量数据上进行交互式SQL的近似查询引擎。

**Tachyon**：以内存为中心高容错的的分布式文件系统。

**四、应用场景**

Yahoo将Spark用在Audience Expansion中的应用，进行点击预测和即席查询等

淘宝技术团队使用了Spark来解决多次迭代的机器学习算法、高计算复杂度的算法等。应用于内容推荐、社区发现等  
腾讯大数据精准推荐借助Spark快速迭代的优势，实现了在“数据实时采集、算法实时训练、系统实时预测”的全流程实时并行高维算法，最终成功应用于广点通pCTR投放系统上。  
优酷土豆将Spark应用于视频推荐(图计算)、广告业务，主要实现机器学习、图计算等迭代计算。

问答题：

Q：Spark和Hadoop的架构区别

A：Hadoop:MapRedcue由Map和Reduce两个阶段，并通过shuffle将两个阶段连接起来的。但是套用MapReduce模型解决问题，不得不将问题分解为若干个有依赖关系的子问题，每个子问题对应一个MapReduce作业，最终所有这些作业形成一个DAG。

Spark:是通用的DAG框架，可以将多个有依赖关系的作业转换为一个大的DAG。核心思想是将Map和Reduce两个操作进一步拆分为多个元操作，这些元操作可以灵活组合，产生新的操作，并经过一些控制程序组装后形成一个大的DAG作业。

Q：Spark和Hadoop的中间计算结果处理区别

A：Hadoop:在DAG中，由于有多个MapReduce作业组成，每个作业都会从HDFS上读取一次数据和写一次数据(默认写三份)，即使这些MapReduce作业产生的数据是中间数据也需要写HDFS。这种表达作业依赖关系的方式比较低效，会浪费大量不必要的磁盘和网络IO，根本原因是作业之间产生的数据不是直接流动的，而是借助HDFS作为共享数据存储系统。

Spark：在Spark中，使用内存(内存不够使用本地磁盘)替代了使用HDFS存储中间结果。对于迭代运算效率更高。

Q：Spark和Hadoop的操作模型区别

A：

Hadoop：只提供了Map和Reduce两种操作所有的作业都得转换成Map和Reduce的操作。

Spark：提供很多种的数据集操作类型比如Transformations包括map,filter,flatMap,sample,groupByKey,reduceByKey,union,join,cogroup,mapValues,sort,partionBy等多种操作类型，还提供actions操作包括Count,collect,reduce,lookup,save等多种。这些多种多样的数据集操作类型，给开发上层应用的用户提供了方便。

Q：spark中的RDD是什么，有哪些特性?

A：

Alistofpartitions：一个分区列表，RDD中的数据都存储在一个分区列表中

Afunctionforcomputingeachsplit：作用在每一个分区中的函数

AlistofdependenciesonotherRDDs：一个RDD依赖于其他多个RDD，这个点很重要，RDD的容错机制就是依据这个特性而来的

Optionally,aPartitionerforkey-valueRDDs(eg:tosaythattheRDDishash-partitioned)：可选的，针对于kv类型的RDD才有这个特性，作用是决定了数据的来源以及数据处理后的去向

可选项，数据本地性，数据位置\*\*\*

Q：概述一下spark中的常用算子区别(map,mapPartitions，foreach，foreachPatition)

A：map：用于遍历RDD，将函数应用于每一个元素，返回新的RDD(transformation算子)

foreach：用于遍历RDD，将函数应用于每一个元素，无返回值(action算子)

mapPatitions：用于遍历操作RDD中的每一个分区，返回生成一个新的RDD(transformation算子)

foreachPatition：用于遍历操作RDD中的每一个分区，无返回值(action算子)

总结：一般使用mapPatitions和foreachPatition算子比map和foreach更加高效，推荐使用。

[**Spark(一): 基本架构及原理**](https://www.cnblogs.com/cxxjohnson/p/8909578.html)

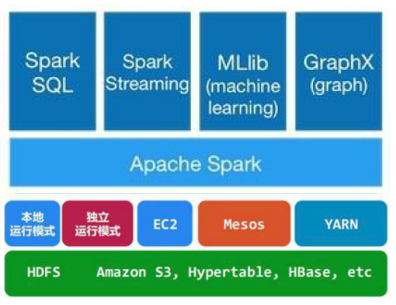
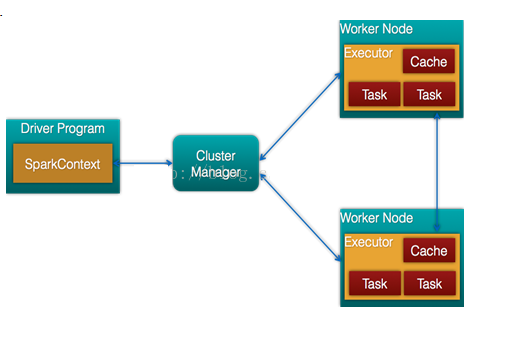
 Apache Spark是一个围绕速度、易用性和复杂分析构建的大数据处理框架，最初在2009年由加州大学伯克利分校的AMPLab开发，并于2010年成为Apache的开源项目之一，与Hadoop和Storm等其他大数据和MapReduce技术相比，Spark有如下优势：

* Spark提供了一个全面、统一的框架用于管理各种有着不同性质（文本数据、图表数据等）的数据集和数据源（批量数据或实时的流数据）的大数据处理的需求
* 官方资料介绍Spark可以将Hadoop集群中的应用在内存中的运行速度提升100倍，甚至能够将应用在磁盘上的运行速度提升10倍

 目标：

* 架构及生态
* spark 与 hadoop
* 运行流程及特点
* 常用术语
* standalone模式
* yarn集群
* RDD运行流程

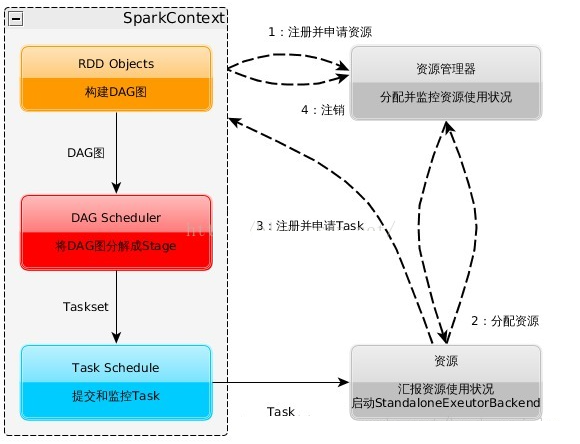
架构及生态：

* 通常当需要处理的数据量超过了单机尺度(比如我们的计算机有4GB的内存，而我们需要处理100GB以上的数据)这时我们可以选择spark集群进行计算，有时我们可能需要处理的数据量并不大，但是计算很复杂，需要大量的时间，这时我们也可以选择利用spark集群强大的计算资源，并行化地计算，其架构示意图如下：
* 
* Spark Core：包含Spark的基本功能；尤其是定义RDD的API、操作以及这两者上的动作。其他Spark的库都是构建在RDD和Spark Core之上的
* Spark SQL：提供通过Apache Hive的SQL变体Hive查询语言（HiveQL）与Spark进行交互的API。每个数据库表被当做一个RDD，Spark SQL查询被转换为Spark操作。
* Spark Streaming：对实时数据流进行处理和控制。Spark Streaming允许程序能够像普通RDD一样处理实时数据
* MLlib：一个常用机器学习算法库，算法被实现为对RDD的Spark操作。这个库包含可扩展的学习算法，比如分类、回归等需要对大量数据集进行迭代的操作。
* GraphX：控制图、并行图操作和计算的一组算法和工具的集合。GraphX扩展了RDD API，包含控制图、创建子图、访问路径上所有顶点的操作
* Spark架构的组成图如下：
* 
* Cluster Manager：在standalone模式中即为Master主节点，控制整个集群，监控worker。在YARN模式中为资源管理器
* Worker节点：从节点，负责控制计算节点，启动Executor或者Driver。
* Driver： 运行Application 的main()函数
* Executor：执行器，是为某个Application运行在worker node上的一个进程

Spark与hadoop:

* Hadoop有两个核心模块，分布式存储模块HDFS和分布式计算模块Mapreduce
* spark本身并没有提供分布式文件系统，因此spark的分析大多依赖于Hadoop的分布式文件系统HDFS
* Hadoop的Mapreduce与spark都可以进行数据计算，而相比于Mapreduce，spark的速度更快并且提供的功能更加丰富
* 关系图如下：
* 

 运行流程及特点：

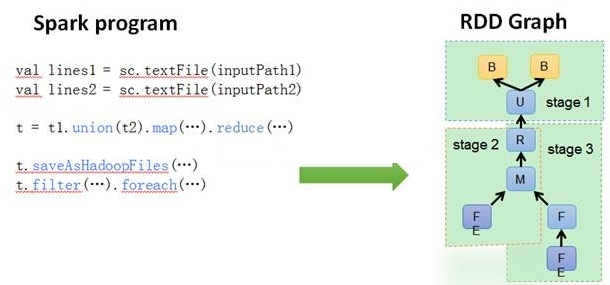
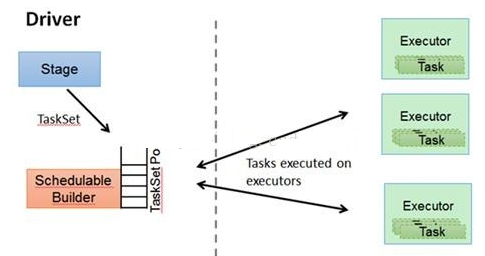
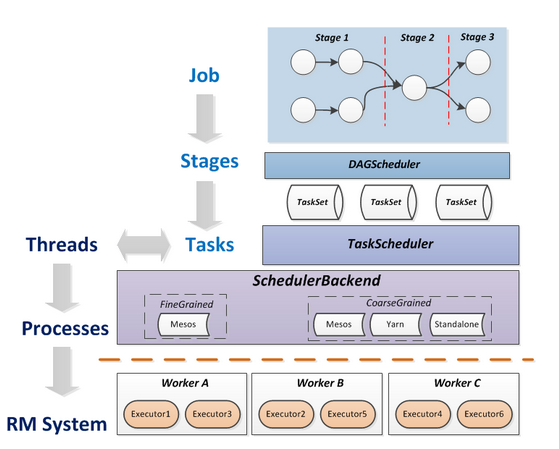
* spark运行流程图如下：
* 

1. 构建Spark Application的运行环境，启动SparkContext
2. SparkContext向资源管理器（可以是Standalone，Mesos，Yarn）申请运行Executor资源，并启动StandaloneExecutorbackend，
3. Executor向SparkContext申请Task
4. SparkContext将应用程序分发给Executor
5. SparkContext构建成DAG图，将DAG图分解成Stage、将Taskset发送给Task Scheduler，最后由Task Scheduler将Task发送给Executor运行
6. Task在Executor上运行，运行完释放所有资源

     Spark运行特点：

1. 每个Application获取专属的executor进程，该进程在Application期间一直驻留，并以多线程方式运行Task。这种Application隔离机制是有优势的，无论是从调度角度看（每个Driver调度他自己的任务），还是从运行角度看（来自不同Application的Task运行在不同JVM中），当然这样意味着Spark Application不能跨应用程序共享数据，除非将数据写入外部存储系统
2. Spark与资源管理器无关，只要能够获取executor进程，并能保持相互通信就可以了
3. 提交SparkContext的Client应该靠近Worker节点（运行Executor的节点），最好是在同一个Rack里，因为Spark Application运行过程中SparkContext和Executor之间有大量的信息交换
4. Task采用了数据本地性和推测执行的优化机制

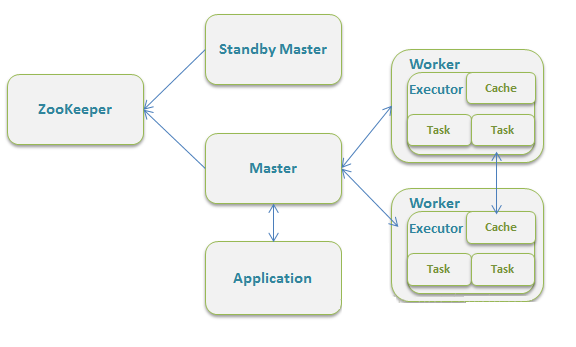
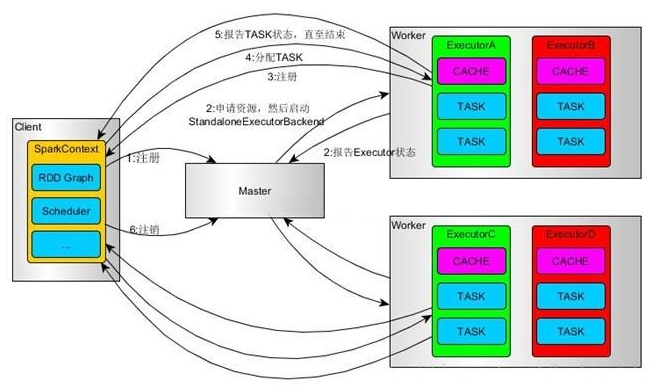
常用术语:

* Application: Appliction都是指用户编写的Spark应用程序，其中包括一个Driver功能的代码和分布在集群中多个节点上运行的Executor代码
* Driver:  Spark中的Driver即运行上述Application的main函数并创建SparkContext，创建SparkContext的目的是为了准备Spark应用程序的运行环境，在Spark中有SparkContext负责与ClusterManager通信，*进行资源申请、任务的分配和监控*等，当Executor部分运行完毕后，Driver同时负责将SparkContext关闭，通常用SparkContext代表Driver
* Executor:  某个Application运行在worker节点上的一个进程，  该进程负责运行某些Task， 并且负责将数据存到内存或磁盘上，每个Application都有各自独立的一批Executor， 在Spark on Yarn模式下，其进程名称为CoarseGrainedExecutor Backend。一个CoarseGrainedExecutor Backend有且仅有一个Executor对象， 负责将Task包装成taskRunner,并从线程池中抽取一个空闲线程运行Task， 这个每一个oarseGrainedExecutor Backend能并行运行Task的数量取决与分配给它的cpu个数
* Cluter Manager：指的是在集群上获取资源的外部服务。目前有三种类型
  1. Standalon : spark原生的资源管理，由Master负责资源的分配
  2. Apache Mesos:与hadoop MR兼容性良好的一种资源调度框架
  3. Hadoop Yarn: 主要是指Yarn中的ResourceManager
* Worker: 集群中任何可以运行Application代码的节点，在Standalone模式中指的是通过slave文件配置的Worker节点，在Spark on Yarn模式下就是NoteManager节点
* Task: 被送到某个Executor上的工作单元，但hadoopMR中的MapTask和ReduceTask概念一样，是运行Application的基本单位，多个Task组成一个Stage，而Task的调度和管理等是由TaskScheduler负责
* Job: 包含多个Task组成的并行计算，往往由Spark Action触发生成， 一个Application中往往会产生多个Job
* Stage: 每个Job会被拆分成多组Task， 作为一个TaskSet， 其名称为Stage，Stage的划分和调度是有DAGScheduler来负责的，Stage有非最终的Stage（Shuffle Map Stage）和最终的Stage（Result Stage）两种，Stage的边界就是发生shuffle的地方
* DAGScheduler: 根据Job构建基于Stage的DAG（Directed Acyclic Graph有向无环图)，并提交Stage给TASkScheduler。 其划分Stage的依据是RDD之间的依赖的关系找出开销最小的调度方法，如下图
* 
* TASKSedulter: 将TaskSET提交给worker运行，每个Executor运行什么Task就是在此处分配的. TaskScheduler维护所有TaskSet，当Executor向Driver发生心跳时，TaskScheduler会根据资源剩余情况分配相应的Task。另外TaskScheduler还维护着所有Task的运行标签，重试失败的Task。下图展示了TaskScheduler的作用
* 
* 在不同运行模式中任务调度器具体为：
  1. Spark on Standalone模式为TaskScheduler
  2. YARN-Client模式为YarnClientClusterScheduler
  3. YARN-Cluster模式为YarnClusterScheduler
* 将这些术语串起来的运行层次图如下：
* 
* Job=多个stage，Stage=多个同种task, Task分为ShuffleMapTask和ResultTask，Dependency分为ShuffleDependency和NarrowDependency

Spark运行模式：

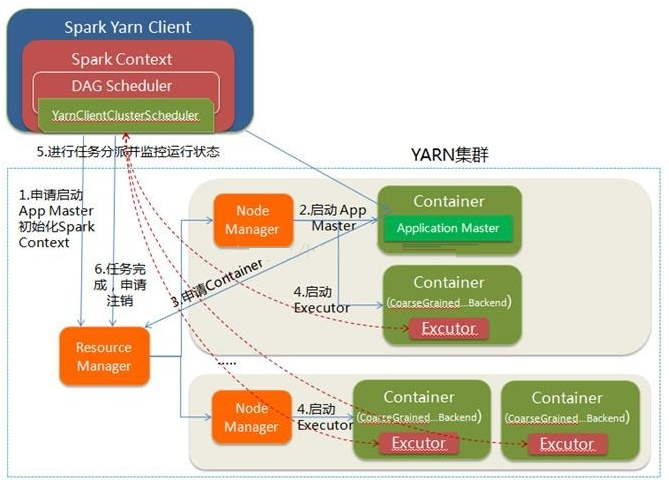
* Spark的运行模式多种多样，灵活多变，部署在单机上时，既可以用本地模式运行，也可以用伪分布模式运行，而当以分布式集群的方式部署时，也有众多的运行模式可供选择，这取决于集群的实际情况，底层的资源调度即可以依赖外部资源调度框架，也可以使用Spark内建的Standalone模式。
* 对于外部资源调度框架的支持，目前的实现包括相对稳定的Mesos模式，以及hadoop YARN模式
* 本地模式：常用于本地开发测试，本地还分别 local 和 local cluster

standalone: 独立集群运行模式

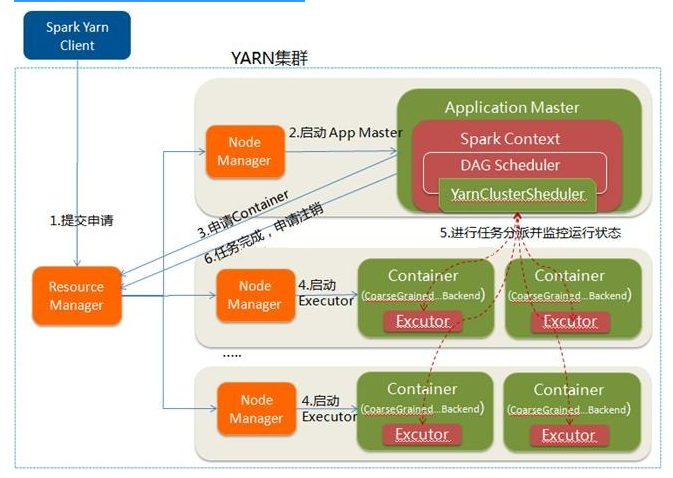
* Standalone模式使用Spark自带的资源调度框架
* 采用Master/Slaves的典型架构，选用ZooKeeper来实现Master的HA
* 框架结构图如下:
* 
* 该模式主要的节点有Client节点、Master节点和Worker节点。其中Driver既可以运行在Master节点上中，也可以运行在本地Client端。当用spark-shell交互式工具提交Spark的Job时，Driver在Master节点上运行；当使用spark-submit工具提交Job或者在Eclips、IDEA等开发平台上使用”new SparkConf.setManager(“spark://master:7077”)”方式运行Spark任务时，Driver是运行在本地Client端上的
* 运行过程如下图：（*参考至：http://blog.csdn.net/gamer\_gyt/article/details/51833681*）
* 

1. SparkContext连接到Master，向Master注册并申请资源（CPU Core 和Memory）
2. Master根据SparkContext的资源申请要求和Worker心跳周期内报告的信息决定在哪个Worker上分配资源，然后在该Worker上获取资源，然后启动StandaloneExecutorBackend；
3. StandaloneExecutorBackend向SparkContext注册；
4. SparkContext将Applicaiton代码发送给StandaloneExecutorBackend；并且SparkContext解析Applicaiton代码，构建DAG图，并提交给DAG Scheduler分解成Stage（当碰到Action操作时，就会催生Job；每个Job中含有1个或多个Stage，Stage一般在获取外部数据和shuffle之前产生），然后以Stage（或者称为TaskSet）提交给Task Scheduler，Task Scheduler负责将Task分配到相应的Worker，最后提交给StandaloneExecutorBackend执行；
5. StandaloneExecutorBackend会建立Executor线程池，开始执行Task，并向SparkContext报告，直至Task完成
6. 所有Task完成后，SparkContext向Master注销，释放资源

yarn:  *（参考：http://blog.csdn.net/gamer\_gyt/article/details/51833681）*

* Spark on YARN模式根据Driver在集群中的位置分为两种模式：一种是YARN-Client模式，另一种是YARN-Cluster（或称为YARN-Standalone模式）
* Yarn-Client模式中，Driver在客户端本地运行，这种模式可以使得Spark Application和客户端进行交互，因为Driver在客户端，所以可以通过webUI访问Driver的状态，默认是http://hadoop1:4040访问，而YARN通过http:// hadoop1:8088访问
* YARN-client的工作流程步骤为：
* 
* Spark Yarn Client向YARN的ResourceManager申请启动Application Master。同时在SparkContent初始化中将创建DAGScheduler和TASKScheduler等，由于我们选择的是Yarn-Client模式，程序会选择YarnClientClusterScheduler和YarnClientSchedulerBackend
* ResourceManager收到请求后，在集群中选择一个NodeManager，为该应用程序分配第一个Container，要求它在这个Container中启动应用程序的ApplicationMaster，与YARN-Cluster区别的是在该ApplicationMaster不运行SparkContext，只与SparkContext进行联系进行资源的分派
* Client中的SparkContext初始化完毕后，与ApplicationMaster建立通讯，向ResourceManager注册，根据任务信息向ResourceManager申请资源（Container）
* 一旦ApplicationMaster申请到资源（也就是Container）后，便与对应的NodeManager通信，要求它在获得的Container中启动CoarseGrainedExecutorBackend，CoarseGrainedExecutorBackend启动后会向Client中的SparkContext注册并申请Task
* client中的SparkContext分配Task给CoarseGrainedExecutorBackend执行，CoarseGrainedExecutorBackend运行Task并向Driver汇报运行的状态和进度，以让Client随时掌握各个任务的运行状态，从而可以在任务失败时重新启动任务
* 应用程序运行完成后，Client的SparkContext向ResourceManager申请注销并关闭自己

Spark Cluster模式:

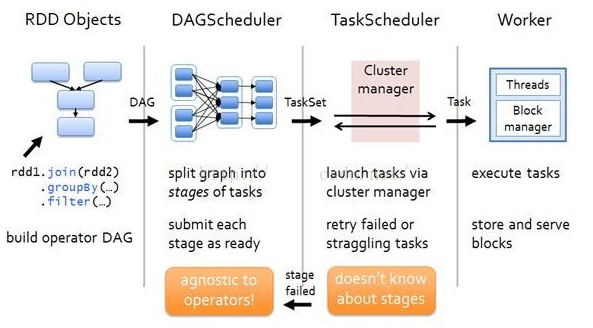
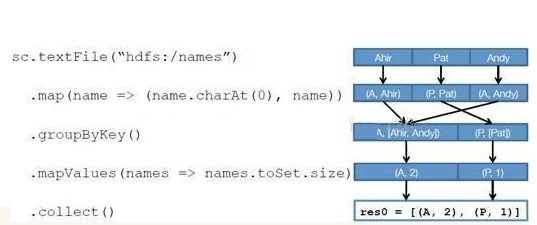
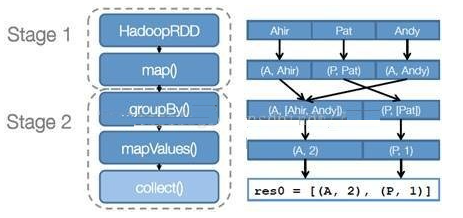
* 在YARN-Cluster模式中，当用户向YARN中提交一个应用程序后，YARN将分两个阶段运行该应用程序：
  1. 第一个阶段是把Spark的Driver作为一个ApplicationMaster在YARN集群中先启动；
  2. 第二个阶段是由ApplicationMaster创建应用程序，然后为它向ResourceManager申请资源，并启动Executor来运行Task，同时监控它的整个运行过程，直到运行完成
* YARN-cluster的工作流程分为以下几个步骤
* 
* Spark Yarn Client向YARN中提交应用程序，包括ApplicationMaster程序、启动ApplicationMaster的命令、需要在Executor中运行的程序等
* ResourceManager收到请求后，在集群中选择一个NodeManager，为该应用程序分配第一个Container，要求它在这个Container中启动应用程序的ApplicationMaster，其中ApplicationMaster进行SparkContext等的初始化
* ApplicationMaster向ResourceManager注册，这样用户可以直接通过ResourceManage查看应用程序的运行状态，然后它将采用轮询的方式通过RPC协议为各个任务申请资源，并监控它们的运行状态直到运行结束
* 一旦ApplicationMaster申请到资源（也就是Container）后，便与对应的NodeManager通信，要求它在获得的Container中启动CoarseGrainedExecutorBackend，CoarseGrainedExecutorBackend启动后会向ApplicationMaster中的SparkContext注册并申请Task。这一点和Standalone模式一样，只不过SparkContext在Spark Application中初始化时，使用CoarseGrainedSchedulerBackend配合YarnClusterScheduler进行任务的调度，其中YarnClusterScheduler只是对TaskSchedulerImpl的一个简单包装，增加了对Executor的等待逻辑等
* ApplicationMaster中的SparkContext分配Task给CoarseGrainedExecutorBackend执行，CoarseGrainedExecutorBackend运行Task并向ApplicationMaster汇报运行的状态和进度，以让ApplicationMaster随时掌握各个任务的运行状态，从而可以在任务失败时重新启动任务
* 应用程序运行完成后，ApplicationMaster向ResourceManager申请注销并关闭自己

Spark Client 和 Spark Cluster的区别:

* 理解YARN-Client和YARN-Cluster深层次的区别之前先清楚一个概念：Application Master。在YARN中，每个Application实例都有一个ApplicationMaster进程，它是Application启动的第一个容器。它负责和ResourceManager打交道并请求资源，获取资源之后告诉NodeManager为其启动Container。从深层次的含义讲YARN-Cluster和YARN-Client模式的区别其实就是ApplicationMaster进程的区别
* YARN-Cluster模式下，Driver运行在AM(Application Master)中，它负责向YARN申请资源，并监督作业的运行状况。当用户提交了作业之后，就可以关掉Client，作业会继续在YARN上运行，因而YARN-Cluster模式不适合运行交互类型的作业
* YARN-Client模式下，Application Master仅仅向YARN请求Executor，Client会和请求的Container通信来调度他们工作，也就是说Client不能离开

思考： 我们在使用Spark提交job时使用的哪种模式？

RDD运行流程：

* RDD在Spark中运行大概分为以下三步：
  1. 创建RDD对象
  2. DAGScheduler模块介入运算，计算RDD之间的依赖关系，RDD之间的依赖关系就形成了DAG
  3. 每一个Job被分为多个Stage。划分Stage的一个主要依据是当前计算因子的输入是否是确定的，如果是则将其分在同一个Stage，避免多个Stage之间的消息传递开销
* 示例图如下：
* 
* 以下面一个按 A-Z 首字母分类，查找相同首字母下不同姓名总个数的例子来看一下 RDD 是如何运行起来的
* 
* 创建 RDD  上面的例子除去最后一个 collect 是个动作，不会创建 RDD 之外，前面四个转换都会创建出新的 RDD 。因此第一步就是创建好所有 RDD( 内部的五项信息 )？
* 创建执行计划 Spark 会尽可能地管道化，并基于是否要重新组织数据来划分 阶段 (stage) ，例如本例中的 groupBy() 转换就会将整个执行计划划分成两阶段执行。最终会产生一个 DAG(directed acyclic graph ，有向无环图 ) 作为逻辑执行计划
* 
* 调度任务  将各阶段划分成不同的 任务 (task) ，每个任务都是数据和计算的合体。在进行下一阶段前，当前阶段的所有任务都要执行完成。因为下一阶段的第一个转换一定是重新组织数据的，所以必须等当前阶段所有结果数据都计算出来了才能继续

分类: [大数据-spark](https://www.cnblogs.com/cxxjohnson/category/1204578.html)

**Spark基础教程**

Spark最初由美国加州伯克利大学的AMP实验室于2009年开发，是基于内存计算的大数据并行计算框架，可用于构建大型的、低延迟的数据分析应用程序。

**Spark特点**

Spark具有如下几个主要特点：

1、 运行速度快：Spark使用先进的DAG（Directed Acyclic Graph，有向无环图）执行引擎，以支持循环数据流与内存计算，基于内存的执行速度可比Hadoop MapReduce快上百倍，基于磁盘的执行速度也能快十倍；

2、 容易使用：Spark支持使用Scala、Java、Python和R语言进行编程，简洁的API设计有助于用户轻松构建并行程序，并且可以通过Spark Shell进行交互式编程；

3、 通用性：Spark提供了完整而强大的技术栈，包括SQL查询、流式计算、机器学习和图算法组件，这些组件可以无缝整合在同一个应用中，足以应对复杂的计算；

4、 运行模式多样：Spark可运行于独立的集群模式中，或者运行于Hadoop中，也可运行于Amazon EC2等云环境中，并且可以访问HDFS、Cassandra、HBase、Hive等多种数据源。

**Spark相对于Hadoop的优势**

Hadoop虽然已成为大数据技术的事实标准，但其本身还存在诸多缺陷，最主要的缺陷是其MapReduce计算模型延迟过高，无法胜任实时、快速计算的需求，因而只适用于离线批处理的应用场景。

回顾Hadoop的工作流程，可以发现Hadoop存在如下一些缺点：

* + 1. 表达能力有限。计算都必须要转化成Map和Reduce两个操作，但这并不适合所有的情况，难以描述复杂的数据处理过程；
    2. 磁盘IO开销大。每次执行时都需要从磁盘读取数据，并且在计算完成后需要将中间结果写入到磁盘中，IO开销较大；
    3. 延迟高。一次计算可能需要分解成一系列按顺序执行的MapReduce任务，任务之间的衔接由于涉及到IO开销，会产生较高延迟。而且，在前一个任务执行完成之前，其他任务无法开始，难以胜任复杂、多阶段的计算任务。

Spark主要具有如下优点：

* 1. Spark的计算模式也属于MapReduce，但不局限于Map和Reduce操作，还提供了多种数据集操作类型，编程模型比MapReduce更灵活；
  2. Spark提供了内存计算，中间结果直接放到内存中，带来了更高的迭代运算效率；
  3. Spark基于DAG的任务调度执行机制，要优于MapReduce的迭代执行机制。
  4. Spark最大的特点就是将计算数据、中间结果都存储在内存中，大大减少了IO开销

Spark提供了多种高层次、简洁的API，通常情况下，对于实现相同功能的应用程序，Spark的代码量要比Hadoop少2-5倍。

但Spark并不能完全替代Hadoop，主要用于替代Hadoop中的MapReduce计算模型。实际上，Spark已经很好地融入了Hadoop生态圈，并成为其中的重要一员，它可以借助于YARN实现资源调度管理，借助于HDFS实现分布式存储。

**Spark生态系统**

Spark的生态系统主要包含了Spark Core、Spark SQL、Spark Streaming、MLLib和GraphX 等组件，各个组件的具体功能如下：

1. Spark Core：Spark Core包含Spark的基本功能，如内存计算、任务调度、部署模式、故障恢复、存储管理等。Spark建立在统一的抽象RDD之上，使其可以以基本一致的方式应对不同的大数据处理场景；通常所说的Apache Spark，就是指Spark Core；
2. Spark SQL：Spark SQL允许开发人员直接处理RDD，同时也可查询Hive、HBase等外部数据源。Spark SQL的一个重要特点是其能够统一处理关系表和RDD，使得开发人员可以轻松地使用SQL命令进行查询，并进行更复杂的数据分析；
3. Spark Streaming：Spark Streaming支持高吞吐量、可容错处理的实时流数据处理，其核心思路是将流式计算分解成一系列短小的批处理作业。Spark Streaming支持多种数据输入源，如Kafka、Flume和TCP套接字等；
4. MLlib（机器学习）：MLlib提供了常用机器学习算法的实现，包括聚类、分类、回归、协同过滤等，降低了机器学习的门槛，开发人员只要具备一定的理论知识就能进行机器学习的工作；
5. GraphX（图计算）：GraphX是Spark中用于图计算的API，可认为是Pregel在Spark上的重写及优化，Graphx性能良好，拥有丰富的功能和运算符，能在海量数据上自如地运行复杂的图算法。

**Spark基本概念**

在具体讲解Spark运行架构之前，需要先了解几个重要的概念：

1. RDD：是弹性分布式数据集（Resilient Distributed Dataset）的简称，是分布式内存的一个抽象概念，提供了一种高度受限的共享内存模型；
2. DAG：是Directed Acyclic Graph（有向无环图）的简称，反映RDD之间的依赖关系；
3. Executor：是运行在工作节点（Worker Node）上的一个进程，负责运行任务，并为应用程序存储数据；
4. 应用：用户编写的Spark应用程序；

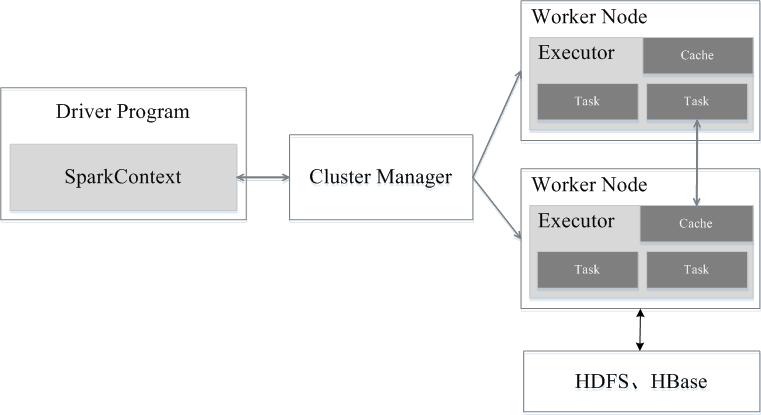
5、任务：运行在Executor上的工作单元；

6、作业：一个作业包含多个RDD及作用于相应RDD上的各种操作；

7、阶段：是作业的基本调度单位，一个作业会分为多组任务，每组任务被称为“阶段”，或者也被称为“任务集”。

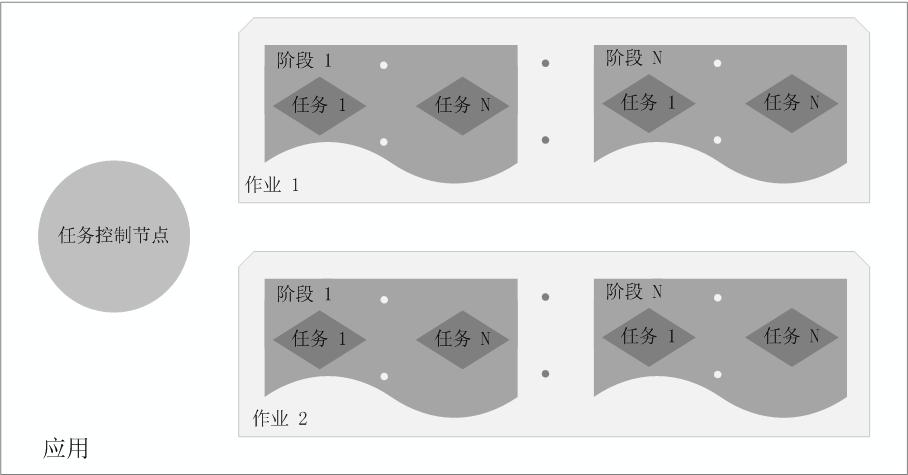
**Spark结构设计**

Spark运行架构包括集群资源管理器（Cluster Manager）、运行作业任务的工作节点（Worker Node）、每个应用的任务控制节点（Driver）和每个工作节点上负责具体任务的执行进程（Executor）。其中，集群资源管理器可以是Spark自带的资源管理器，也可以是YARN或Mesos等资源管理框架。



**Spark各种概念之间的关系**

在Spark中，一个应用（Application）由一个任务控制节点（Driver）和若干个作业（Job）构成，一个作业由多个阶段（Stage）构成，一个阶段由多个任务（Task）组成。当执行一个应用时，任务控制节点会向集群管理器（Cluster Manager）申请资源，启动Executor，并向Executor发送应用程序代码和文件，然后在Executor上执行任务，运行结束后，执行结果会返回给任务控制节点，或者写到HDFS或者其他数据库中。



**Executor的优点**

与Hadoop MapReduce计算框架相比，Spark所采用的Executor有两个优点：

* + - 1. 利用多线程来执行具体的任务（Hadoop MapReduce采用的是进程模型），减少任务的启动开销；
      2. Executor中有一个BlockManager存储模块，会将内存和磁盘共同作为存储设备，当需要多轮迭代计算时，可以将中间结果存储到这个存储模块里，下次需要时，就可以直接读该存储模块里的数据，而不需要读写到HDFS等文件系统里，因而有效减少了IO开销；或者在交互式查询场景下，预先将表缓存到该存储系统上，从而可以提高读写IO性能。

**Spark运行基本流程**

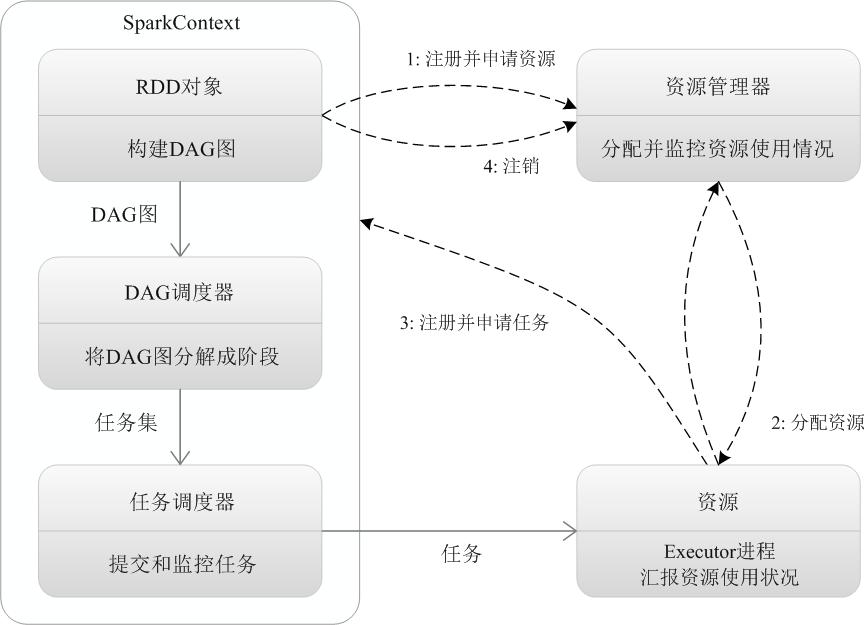
Spark的基本运行流程如下：

1.当一个Spark应用被提交时，首先需要为这个应用构建起基本的运行环境，即由任务控制节点（Driver）创建一个SparkContext，由SparkContext负责和资源管理器（Cluster Manager）的通信以及进行资源的申请、任务的分配和监控等。SparkContext会向资源管理器注册并申请运行Executor的资源；

2.资源管理器为Executor分配资源，并启动Executor进程，Executor运行情况将随着“心跳”发送到资源管理器上；

3.SparkContext根据RDD的依赖关系构建DAG图，DAG图提交给DAG调度器（DAGScheduler）进行解析，将DAG图分解成多个“阶段”（每个阶段都是一个任务集），并且计算出各个阶段之间的依赖关系，然后把一个个“任务集”提交给底层的任务调度器（TaskScheduler）进行处理；Executor向SparkContext申请任务，任务调度器将任务分发给Executor运行，同时，SparkContext将应用程序代码发放给Executor；

4.任务在Executor上运行，把执行结果反馈给任务调度器，然后反馈给DAG调度器，运行完毕后写入数据并释放所有资源。



**Spark运行架构的特点**

Spark运行架构具有以下特点：

1.每个应用都有自己专属的Executor进程，并且该进程在应用运行期间一直驻留。Executor进程以多线程的方式运行任务，减少了多进程任务频繁的启动开销，使得任务执行变得非常高效和可靠；

2.Spark运行过程与资源管理器无关，只要能够获取Executor进程并保持通信即可；

3.Executor上有一个BlockManager存储模块，类似于键值存储系统（把内存和磁盘共同作为存储设备），在处理迭代计算任务时，不需要把中间结果写入到HDFS等文件系统，而是直接放在这个存储系统上，后续有需要时就可以直接读取；在交互式查询场景下，也可以把表提前缓存到这个存储系统上，提高读写IO性能；

4.任务采用了数据本地性和推测执行等优化机制。数据本地性是尽量将计算移到数据所在的节点上进行，即“计算向数据靠拢”，因为移动计算比移动数据所占的网络资源要少得多。而且，Spark采用了延时调度机制，可以在更大的程度上实现执行过程优化。比如，拥有数据的节点当前正被其他的任务占用，那么，在这种情况下是否需要将数据移动到其他的空闲节点呢？答案是不一定。因为，如果经过预测发现当前节点结束当前任务的时间要比移动数据的时间还要少，那么，调度就会等待，直到当前节点可用。

**Spark的部署模式**

Spark支持的三种典型集群部署方式，即standalone、Spark on Mesos和Spark on YARN；然后，介绍在企业中是如何具体部署和应用Spark框架的，在企业实际应用环境中，针对不同的应用场景，可以采用不同的部署应用方式，或者采用Spark完全替代原有的Hadoop架构，或者采用Spark和Hadoop一起部署的方式。

Spark三种部署方式

Spark应用程序在集群上部署运行时，可以由不同的组件为其提供资源管理调度服务（资源包括CPU、内存等）。比如，可以使用自带的独立集群管理器（standalone），或者使用YARN，也可以使用Mesos。因此，Spark包括三种不同类型的集群部署方式，包括standalone、Spark on Mesos和Spark on YARN。

1.standalone模式

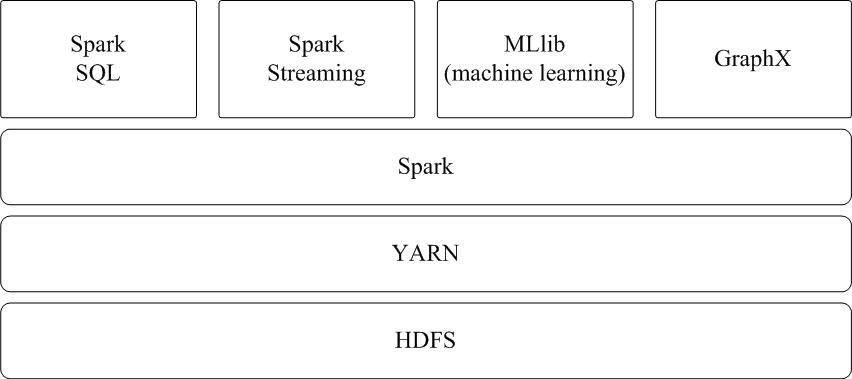
与MapReduce1.0框架类似，Spark框架本身也自带了完整的资源调度管理服务，可以独立部署到一个集群中，而不需要依赖其他系统来为其提供资源管理调度服务。在架构的设计上，Spark与MapReduce1.0完全一致，都是由一个Master和若干个Slave构成，并且以槽（slot）作为资源分配单位。不同的是，Spark中的槽不再像MapReduce1.0那样分为Map 槽和Reduce槽，而是只设计了统一的一种槽提供给各种任务来使用。

2.Spark on Mesos模式

Mesos是一种资源调度管理框架，可以为运行在它上面的Spark提供服务。Spark on Mesos模式中，Spark程序所需要的各种资源，都由Mesos负责调度。由于Mesos和Spark存在一定的血缘关系，因此，Spark这个框架在进行设计开发的时候，就充分考虑到了对Mesos的充分支持，因此，相对而言，Spark运行在Mesos上，要比运行在YARN上更加灵活、自然。目前，Spark官方推荐采用这种模式，所以，许多公司在实际应用中也采用该模式。

3. Spark on YARN模式

Spark可运行于YARN之上，与Hadoop进行统一部署，即“Spark on YARN”，其架构如图9-13所示，资源管理和调度依赖YARN，分布式存储则依赖HDFS。



**Hadoop和Spark的统一部署**

一方面，由于Hadoop生态系统中的一些组件所实现的功能，目前还是无法由Spark取代的，比如，Storm可以实现毫秒级响应的流计算，但是，Spark则无法做到毫秒级响应。另一方面，企业中已经有许多现有的应用，都是基于现有的Hadoop组件开发的，完全转移到Spark上需要一定的成本。因此，在许多企业实际应用中，Hadoop和Spark的统一部署是一种比较现实合理的选择。

由于Hadoop MapReduce、HBase、Storm和Spark等，都可以运行在资源管理框架YARN之上，因此，可以在YARN之上进行统一部署（如图9-16所示）。这些不同的计算框架统一运行在YARN中，可以带来如下好处：

 计算资源按需伸缩；

 不用负载应用混搭，集群利用率高；

 共享底层存储，避免数据跨集群迁移。

