大数据分析进阶

（资源整理）带你入门 **Spark**

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

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

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

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

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

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

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

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

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

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

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

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

目录

- 一、官网介绍
  - 1、什么是 Spark
- 二、Spark 的四大特性
  - 1、高效性
  - 2、易用性
  - 3、通用性
  - 4、兼容性
- 三、Spark 的组成

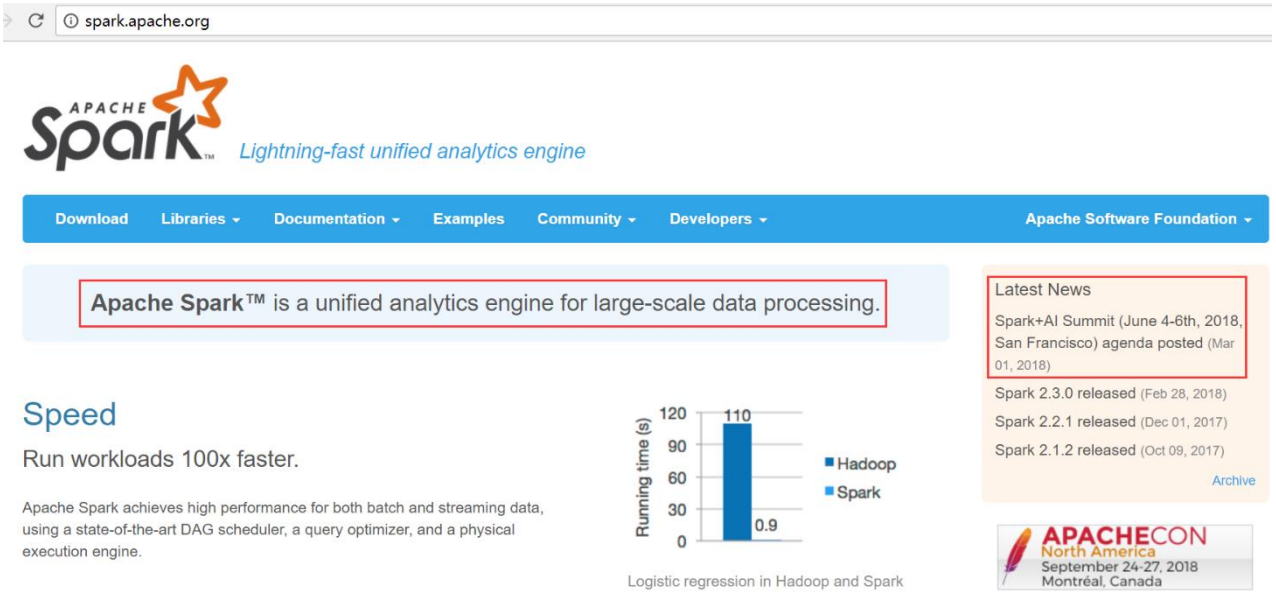
- 四、应用场景

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

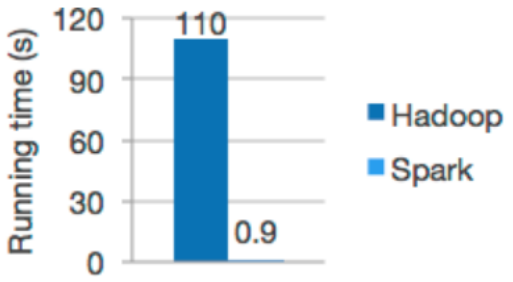
1、高效性

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。

Speed

Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.



Logistic regression in Hadoop and Spark

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。

Ease of Use

Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells.

```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

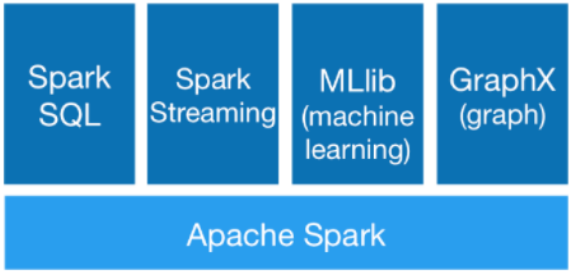
3、通用性

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。

Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.



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。

Runs Everywhere

Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.

You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes. Access data in HDFS, Apache Cassandra, Apache HBase, Apache Hive, and hundreds of other data sources.

Mesos：Spark 可以运行在 Mesos 里面（Mesos 类似于 yarn 的一个资源调度框架）
standalone：Spark 自己可以给自己分配资源（master，worker）
YARN：Spark 可以运行在 yarn 上面
Kubernetes：Spark 接收 Kubernetes 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

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：

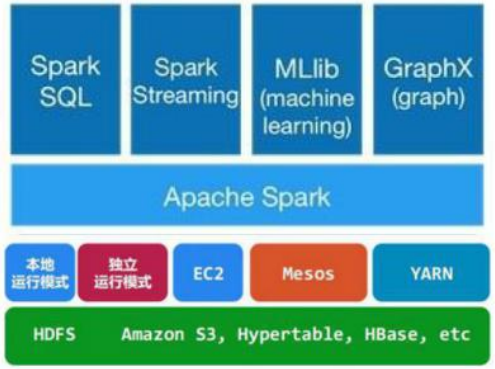
- Spark 提供了一个全面、统一的框架用于管理各种有着不同性质（文本数据、图表数据等）的数据集和数据源（批量数据或实时的流数据）的大数据处理的需求
- 官方资料介绍 Spark 可以将 Hadoop 集群中的应用在内存中的运行速度提升 100 倍，甚至能够将应用在磁盘上的运行速度提升 10 倍

目标：

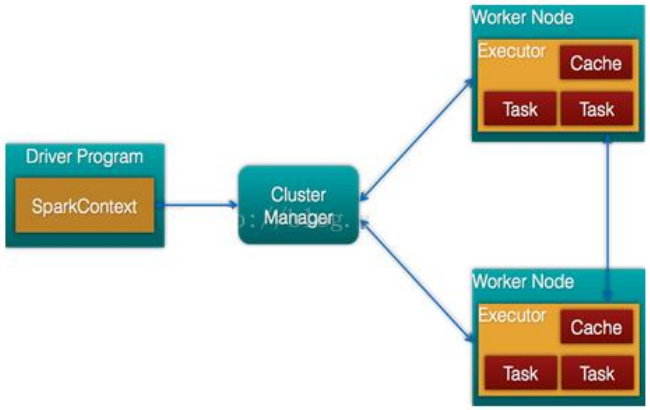
- 架构及生态
- spark 与 hadoop
- 运行流程及特点
- 常用术语
- standalone 模式
- yarn 集群
- RDD 运行流程

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：



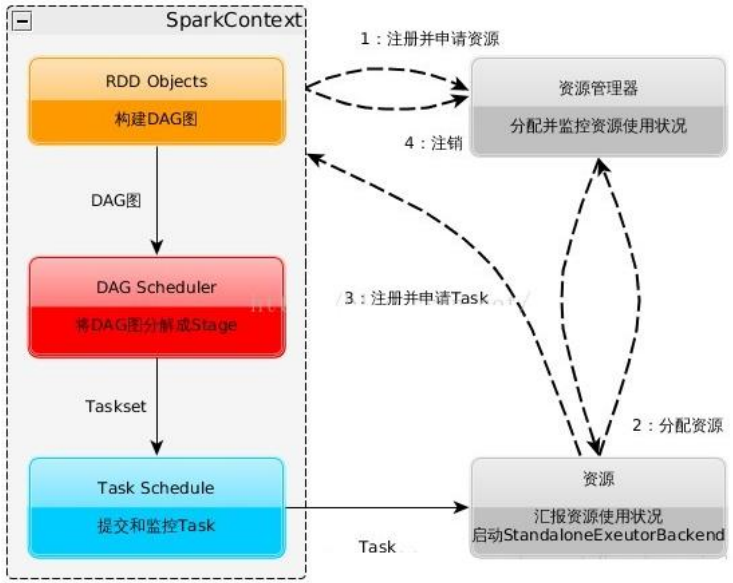
- Cluster Manager：在 standalone 模式中即为 Master 主节点，控制整个集群，监控 worker。在 YARN 模式中为资源管理器
- Worker 节点：从节点，负责控制计算节点，启动 Executor 或者 Driver。
- Driver： 运行 Application 的 main()函数
- Executor：执行器，是为某个 Application 运行在 worker node 上的一个进程

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：

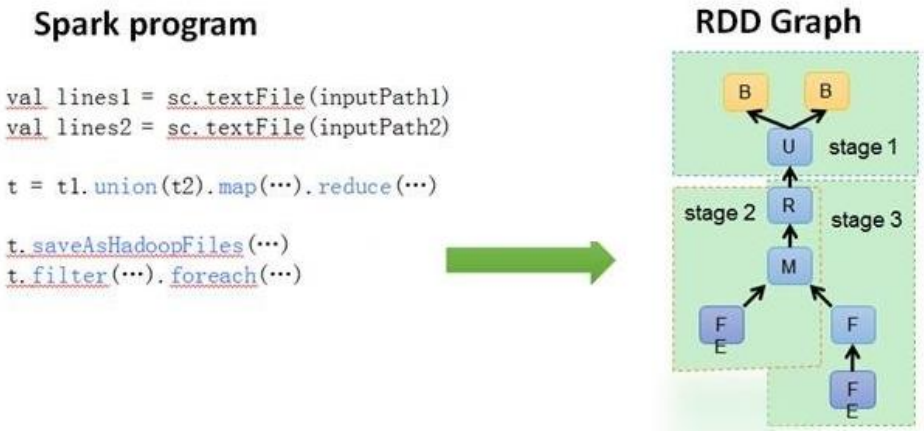
1. 每个 Application 获取专属的 executor 进程，该进程在 Application 期间一直驻留，并以多线程方式运行 Task。这种 Application 隔离机制是有优势的，无论是从调度角度看（每个 Driver 调度他自己的任务），还是从运行角度看（来自不同 Application 的 Task 运行在不同 JVM 中），当然这样意味着 Spark Application 不能跨应用程序共享数据，除非将数据写入外部存储系统
2. Spark 与资源管理器无关，只要能够获取 executor 进程，并能保持相互通信就可以了
3. 提交 SparkContext 的 Client 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:

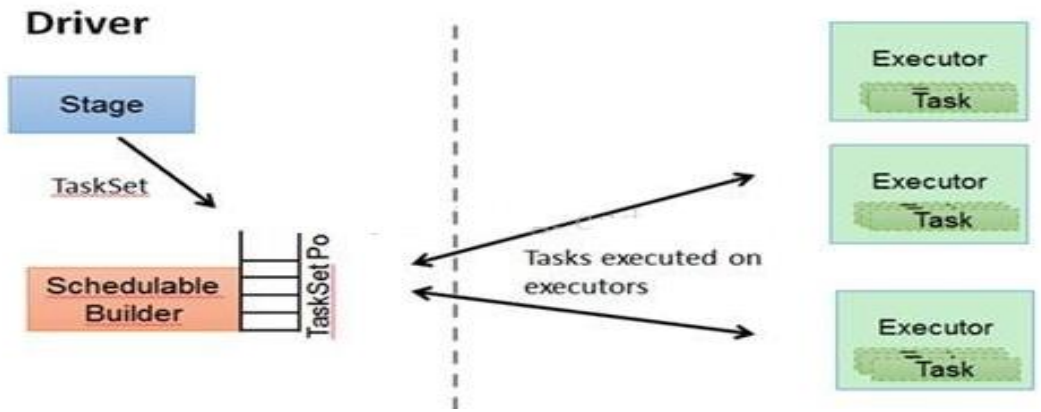
- Application: Appliction 都是指用户编写的 Spark 应用程序，其中包括一个 Driver 功能的代码和分布在集群中多个节点上运行的 Executor 代码
- Driver: Spark 中的 Driver 即运行上述 Application 的 main 函数并创建 SparkContext，创建 SparkContext 的目的是为了准备 Spark 应用程序的运行环境，在 Spark 中有 SparkContext 负责与 ClusterManager 通信，*进行资源申请*、*任务的分配和监控*等，当 Executor 部分运行完毕后，Driver 同时负责将 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



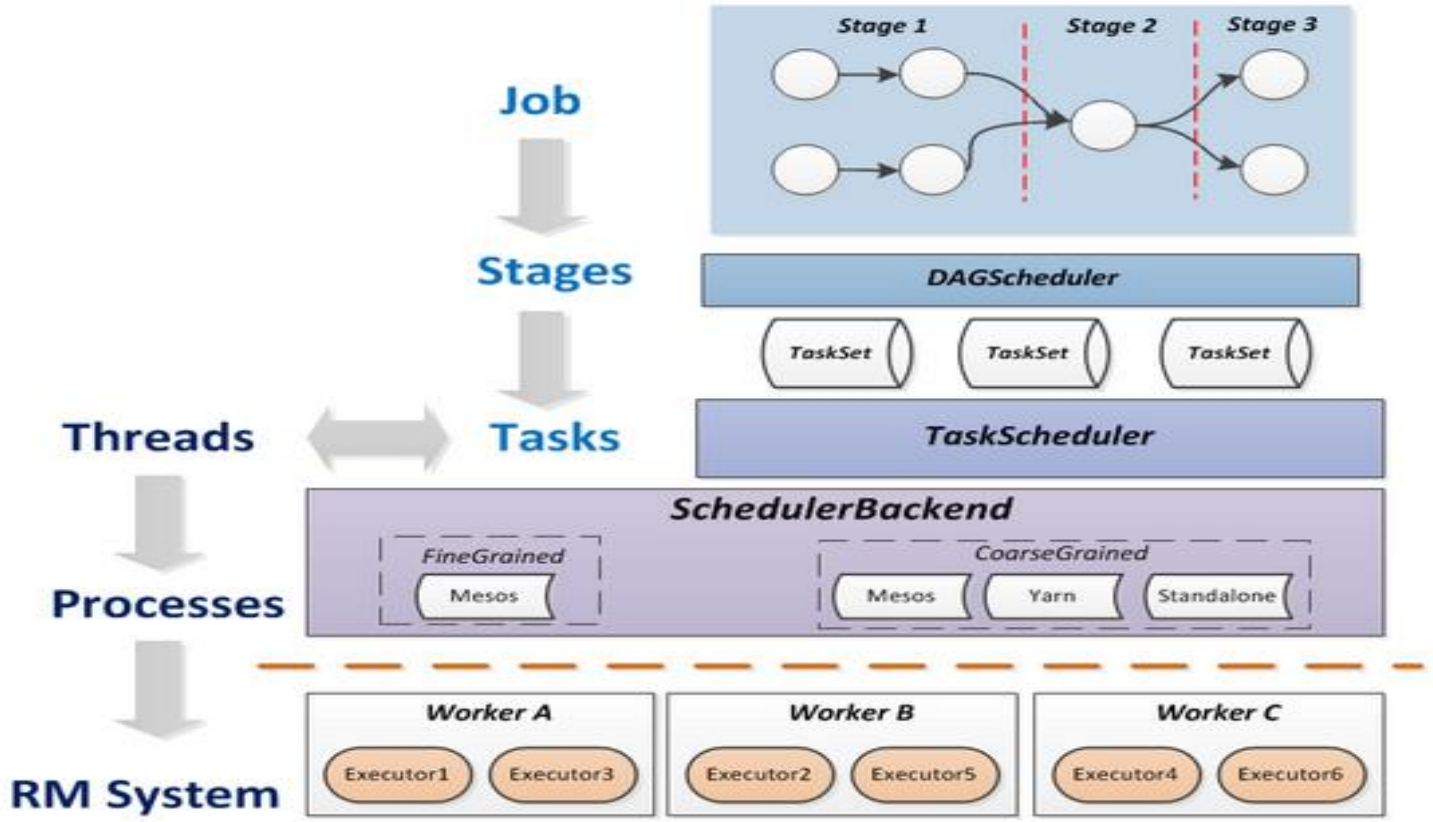
- TASKSedulter: 将 TaskSET 提交给 worker 运行，每个 Executor 运行什么 Task 就是在此处分配的. TaskScheduler 维护所有 TaskSet，当 Executor 向 Driver 发生心跳时，TaskScheduler 会根据资源剩余情况分配相应的 Task。另外 TaskScheduler 还维护着所有 Task 的运行标签，重试失败的 Task。下图展示了 TaskScheduler 的作用



- 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：



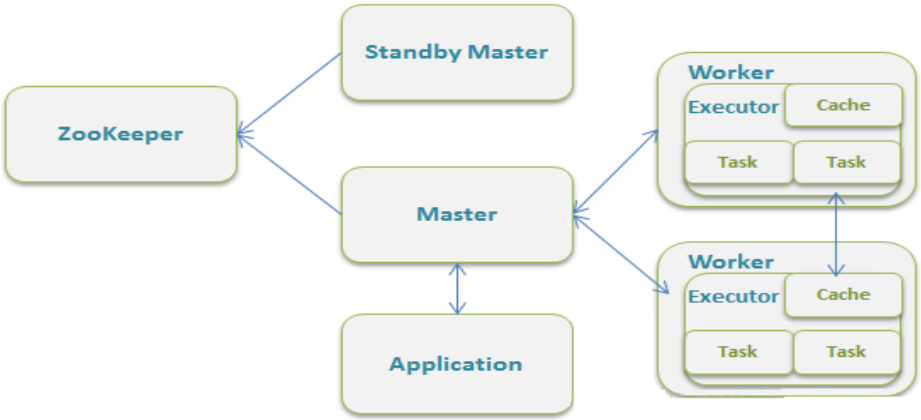
- Job=多个 stage，Stage=多个同种 task, Task 分为 ShuffleMapTask 和 ResultTask，Dependency 分为 ShuffleDependency 和 NarrowDependency

Spark 运行模式：

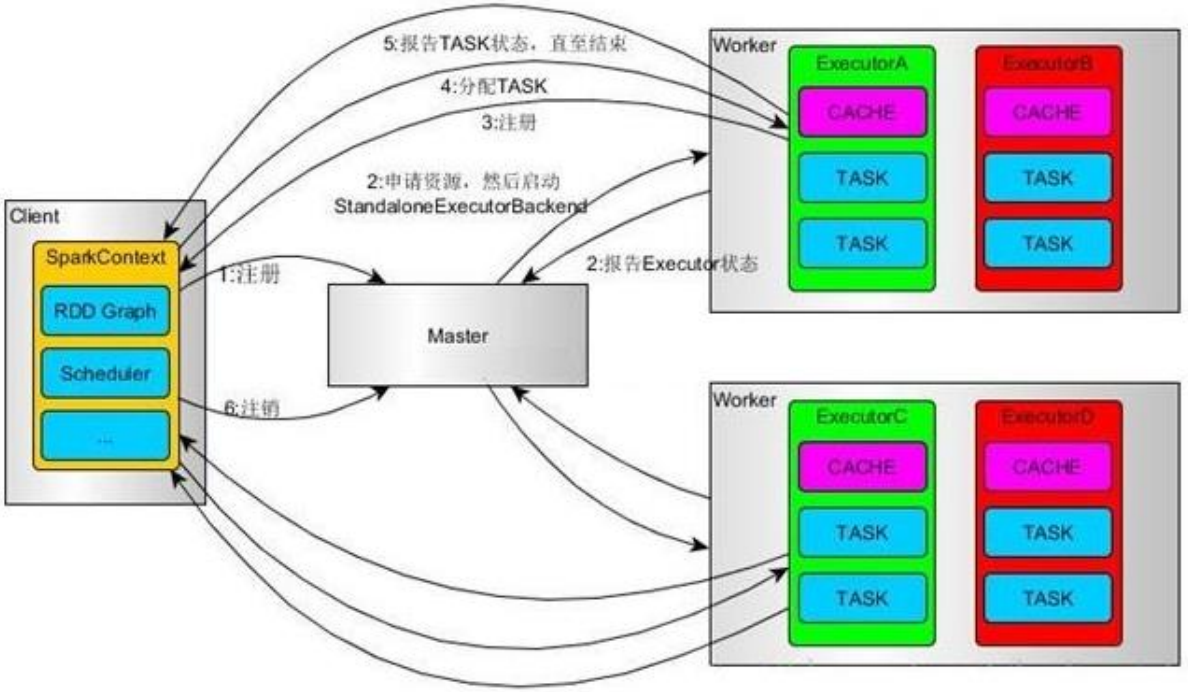
- Spark 的运行模式多种多样，灵活多变，部署在单机上时，既可以用本地模式运行，也可以用伪分布模式运行，而当以分布式集群的方式部署时，也有众多的运行模式可供选择，这取决于集群的实际情况，底层的资源调度即可以依赖外部资源调度框架，也可以使用 Spark 内建的 Standalone 模式。
- 对于外部资源调度框架的支持，目前的实现包括相对稳定的 Mesos 模式，以及 hadoop YARN 模式
- 本地模式：常用于本地开发测试，本地还分别 local 和 local cluster

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:



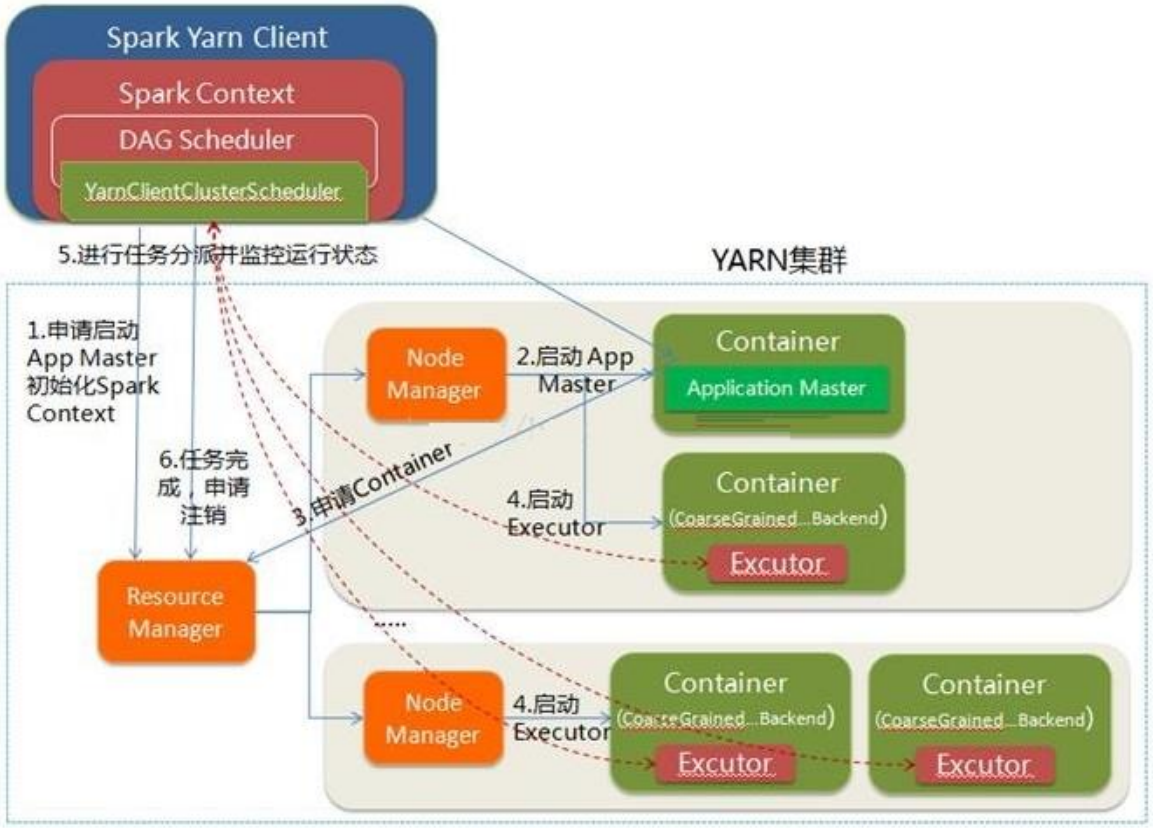
- 该模式主要的节点有 Client 节点、Master 节点和 Worker 节点。其中 Driver 既可以运行在 Master 节点上中，也可以运行在本地 Client 端。当用 spark-shell 交互式工具提交 Spark 的 Job 时，Driver 在 Master 节点上运行；当使用 spark-submit 工具提交 Job 或者在 Eclips、IDEA 等开发平台上使用”new SparkConf.setManager(“spark://master:7077”)”方式运行 Spark 任务时，Driver 是运行在本地 Client 端上的
- 运行过程如下图：（*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：

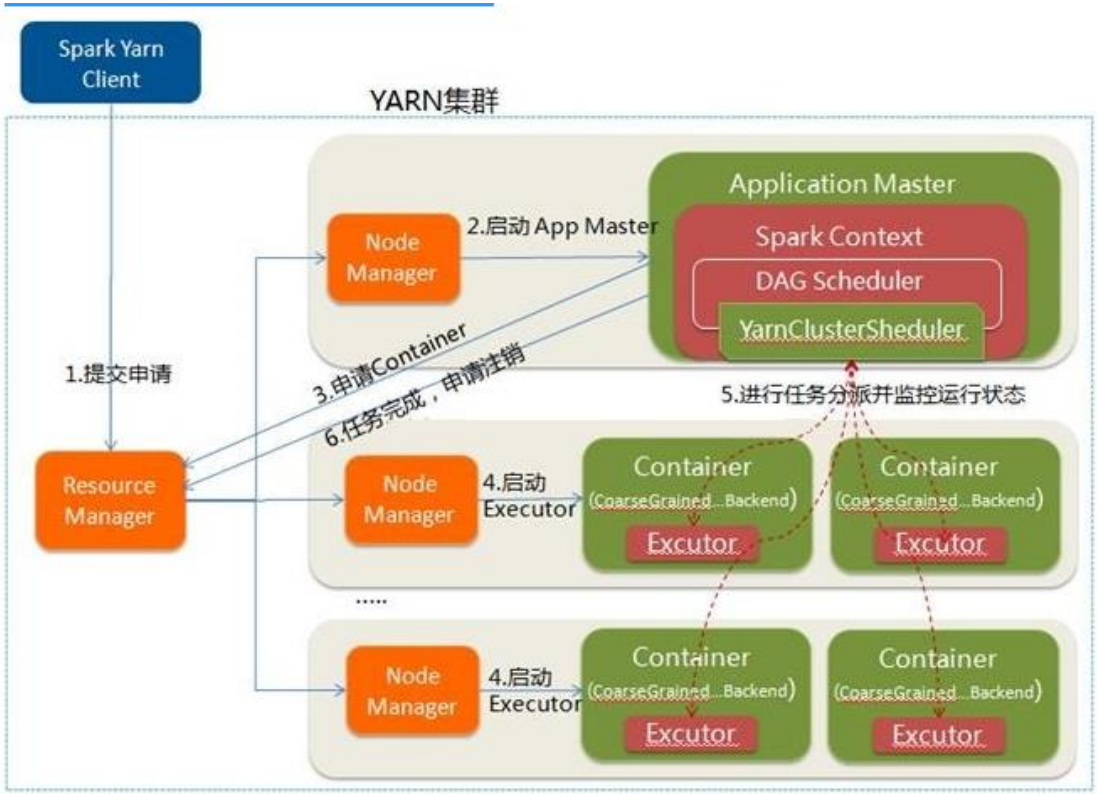


- Spark Yarn Client 向 YARN 的 ResourceManager 申请启动 Application Master。同时在 SparkContent 初始化中将创建 DAGScheduler 和 TASKScheduler 等，由于我们选择的是 Yarn-Client 模式，程序会选择 YarnClientClusterScheduler 和 YarnClientSchedulerBackend
- ResourceManager 收到请求后，在集群中选择一个 NodeManager，为该应用程序分配第一个 Container，要求它在这个 Container 中启动应用程序的 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：

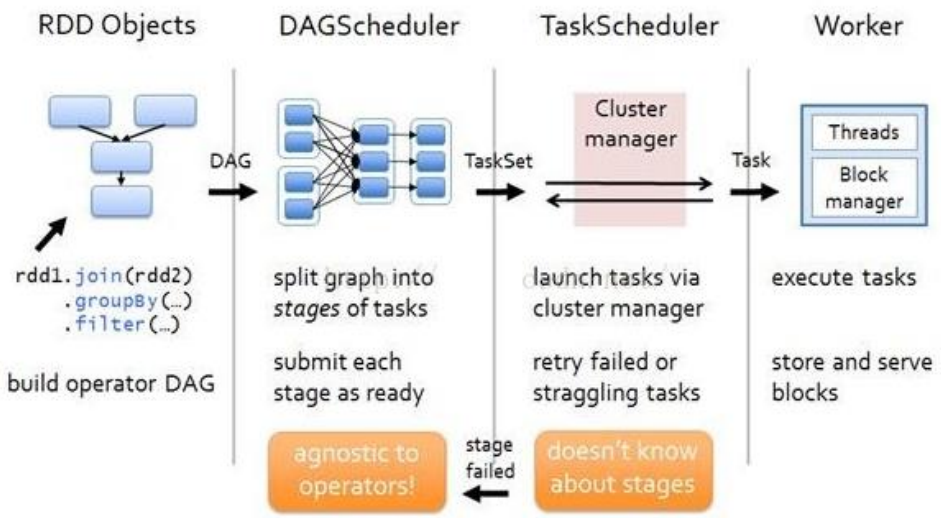
- 理解 YARN-Client 和 YARN-Cluster 深层次的区别之前先清楚一个概念：Application Master。在 YARN 中，每个 Application 实例都有一个 ApplicationMaster 进程，它是 Application 启动的第一个容器。它负责和 ResourceManager 打交道并请求资源，获取资源之后告诉 NodeManager 为其启动 Container。从深层次的含义讲 YARN-Cluster 和 YARN-Client 模式的区别其实就是 ApplicationMaster 进程的区别
- YARN-Cluster 模式下，Driver 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？

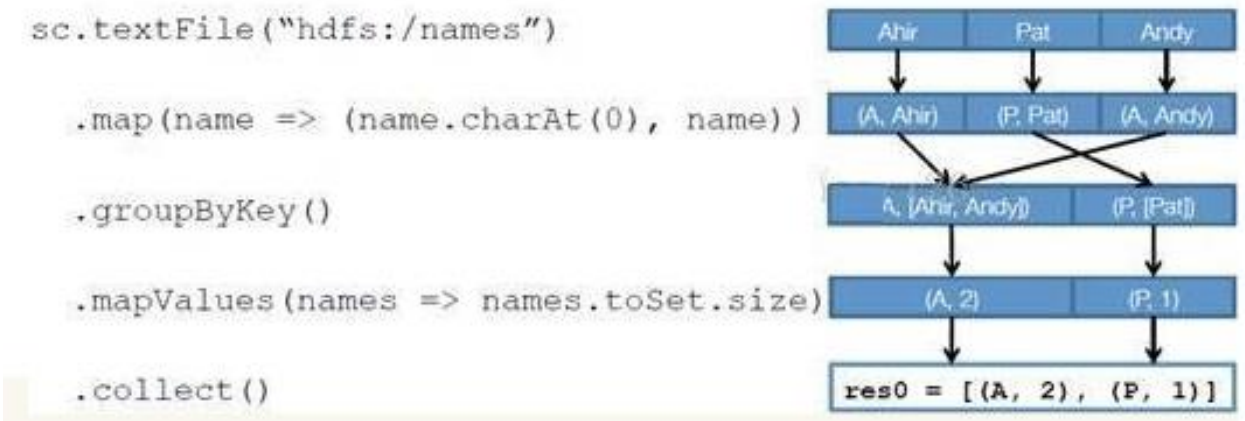
RDD 运行流程：

- RDD 在 Spark 中运行大概分为以下三步：

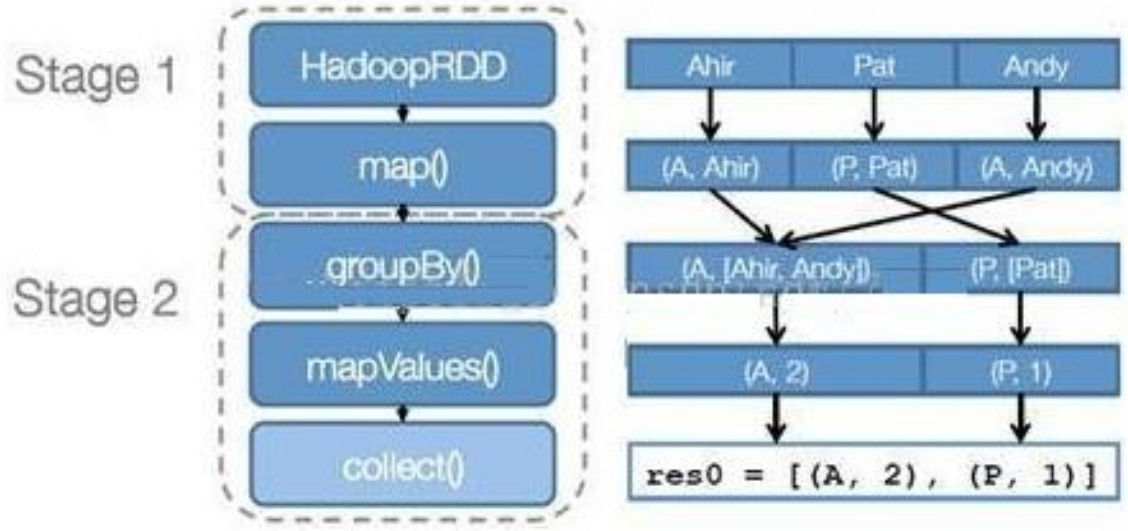
1. 创建 RDD 对象
2. DAGScheduler 模块介入运算，计算 RDD 之间的依赖关系，RDD 之间的依赖关系就形成了 DAG
3. 每一个 Job 被分为多个 Stage。划分 Stage 的一个主要依据是当前计算因子的输入是否是确定的，如果是则将其分在同一个 Stage，避免多个 Stage 之间的消息传递开销
    - 示例图如下：



- 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

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；

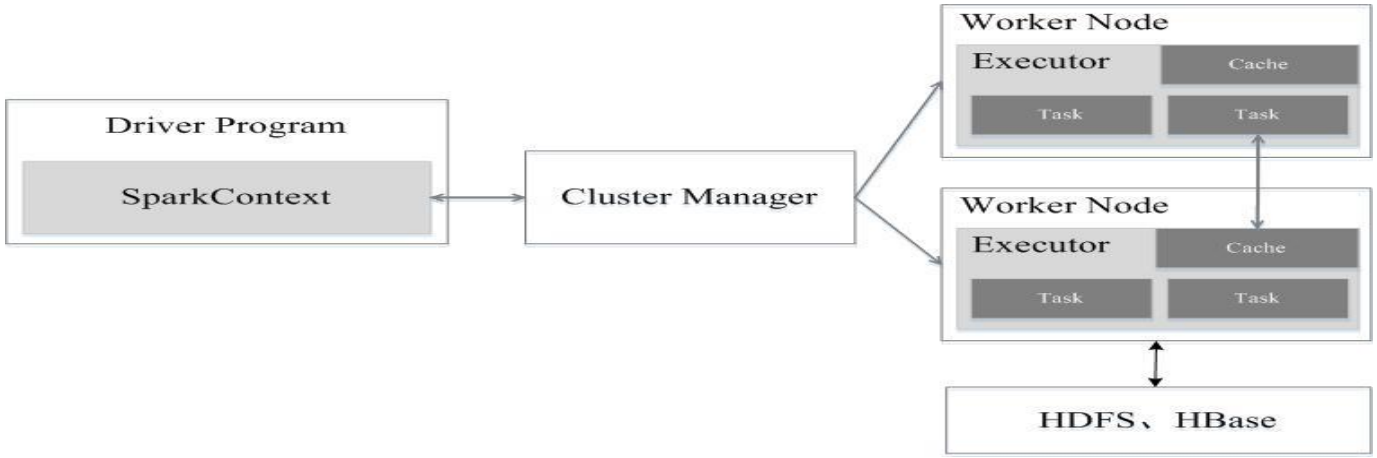
5、任务：运行在 Executor 上的工作单元；

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”。

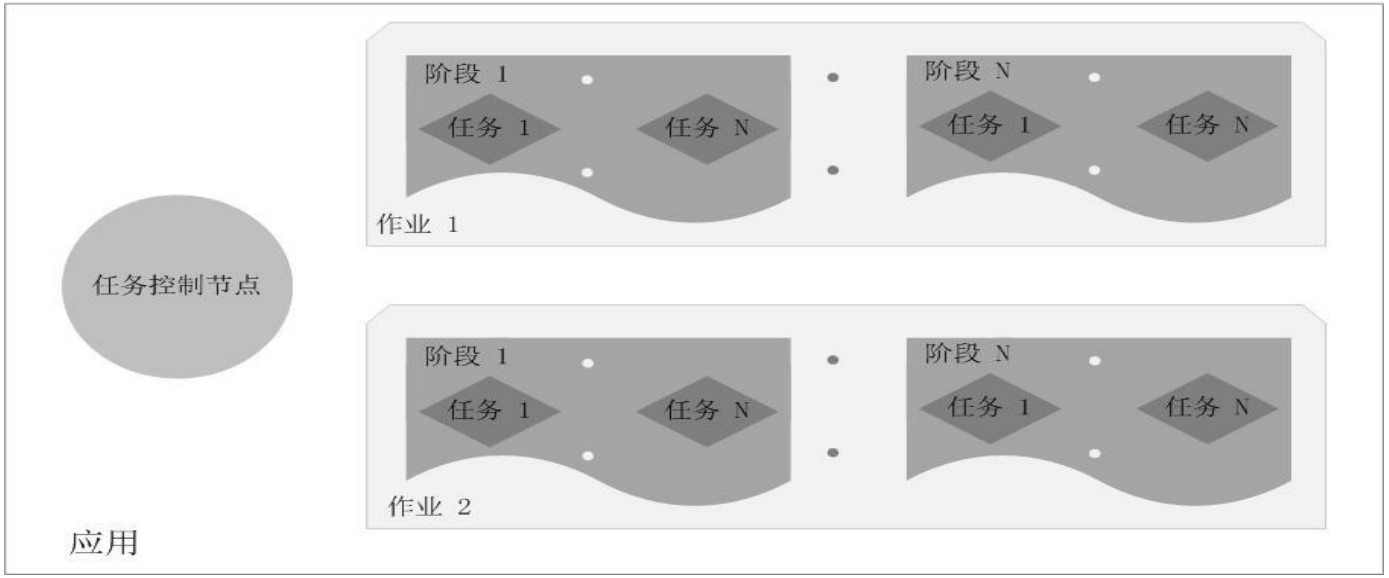
Spark 结构设计

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

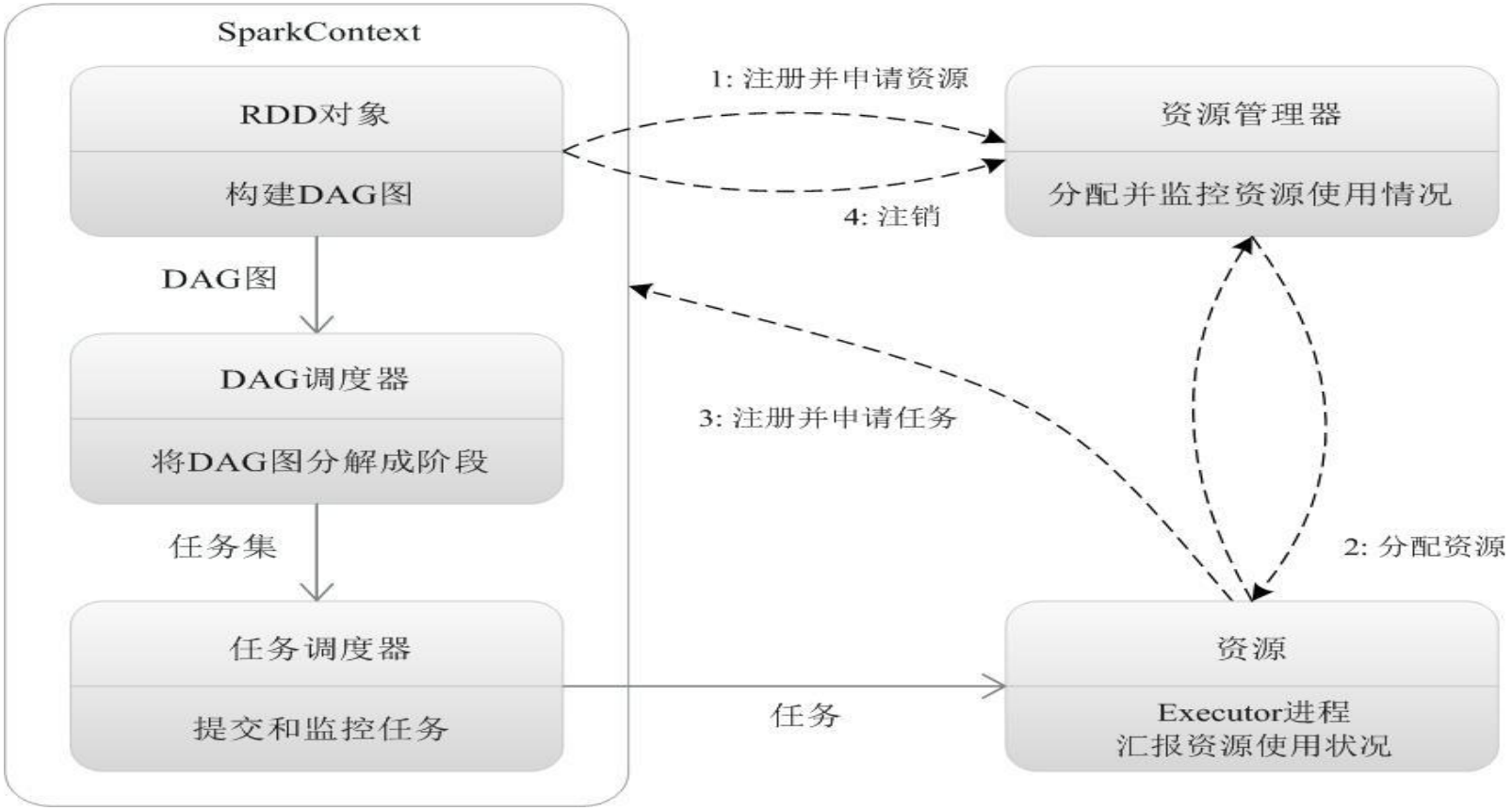
Spark 的基本运行流程如下：

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

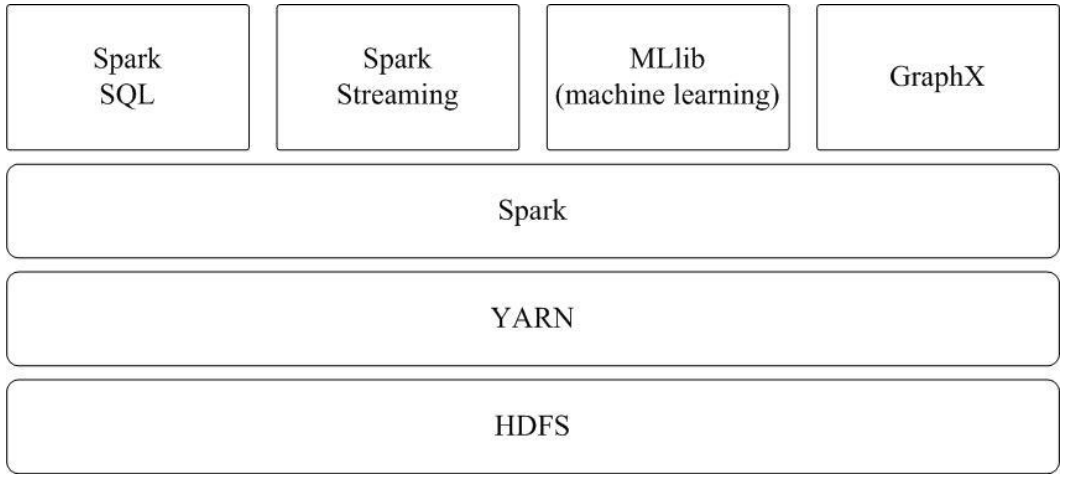
与 MapReduce1.0 框架类似，Spark 框架本身也自带了完整的资源调度管理服务，可以独立部署到一个集群中，而不需要依赖其他系统来为其提供资源管理调度服务。在架构的设计上，Spark 与 MapReduce1.0 完全一致，都是由一个 Master 和若干个 Slave 构成，并且以槽（slot）作为资源分配单位。不同的是，Spark 中的槽不再像 MapReduce1.0 那样分为 Map 槽和 Reduce 槽，而是只设计了统一的一种槽提供给各种任务来使用。

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。