转 Spark(一): 基本架构及原理

2017年03月08日 11:26:45 阅读数：79259

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：

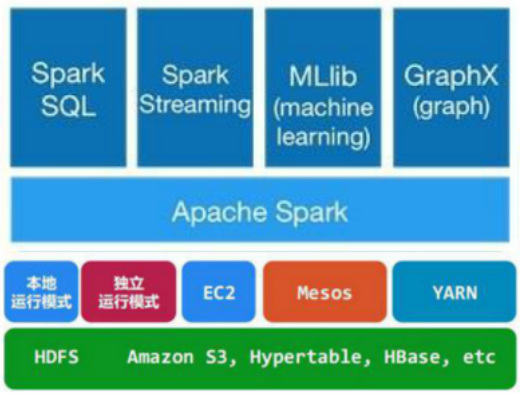
- Spark提供了一个全面、统一的框架用于管理各种有着不同性质（文本数据、图表数据等）的数据集和数据源（批量数据或实时的流数据）的大数据处理的需求
- 官方资料介绍Spark可以将Hadoop集群中的应用在内存中的运行速度提升100倍，甚至能够将应用在磁盘上的运行速度提升10倍

目标：

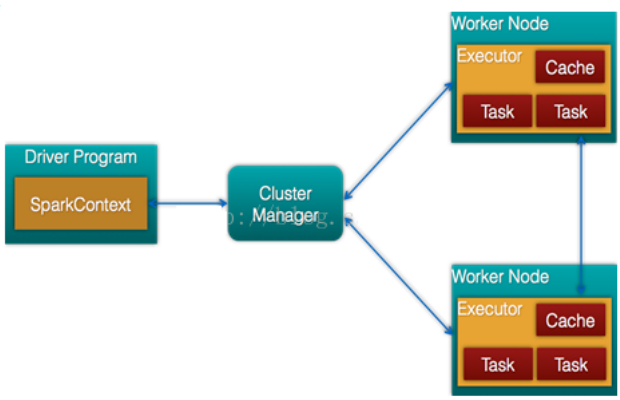
- 架构及生态
- spark 与 hadoop
- 运行流程及特点
- 常用术语
- standalone模式
- yarn集群
- RDD运行流程

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

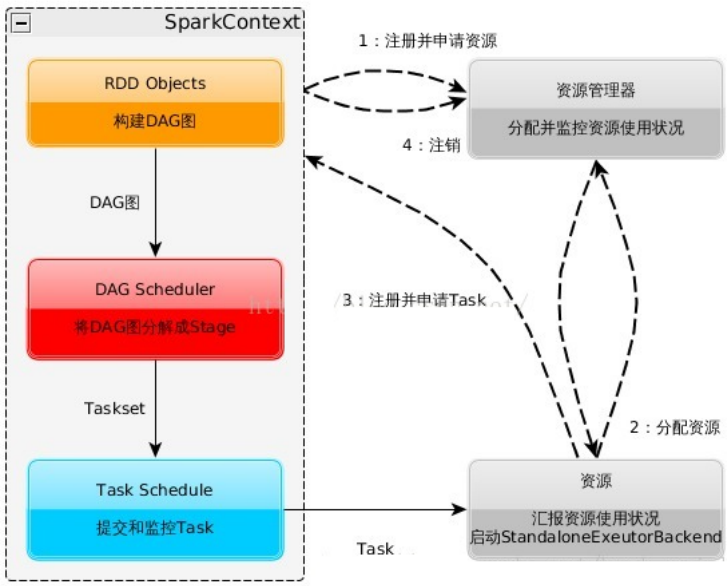
- Executor：执行器，是为某个Application运行在worker node上的一个进程

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：

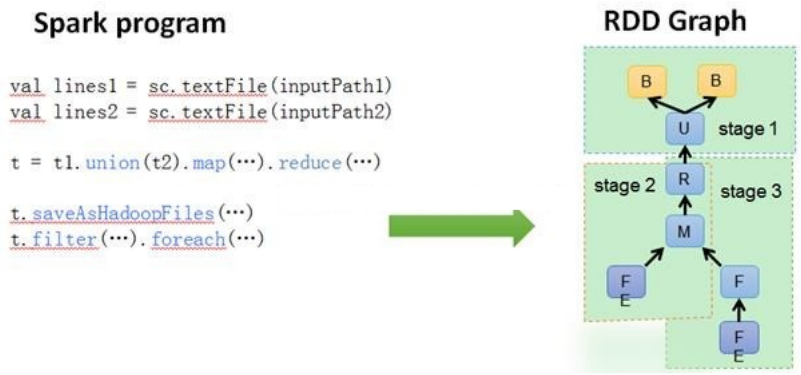
1. 每个Application获取专属的executor进程，该进程在Application期间一直驻留，并以多线程方式运行Task。这种Application隔离机制是有优势的，无论是从调度角度看（每个Driver调度他自己的任务），还是从运行角度看（来自不同Application的Task运行在不同JVM中），当然这样意味着Spark Application不能跨应用程序共享数据，除非将数据写入外部存储系统
2. Spark与资源管理器无关，只要能够获取executor进程，并能保持相互通信就可以了
3. 提交SparkContext的Client应该靠近Worker节点（运行Executor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

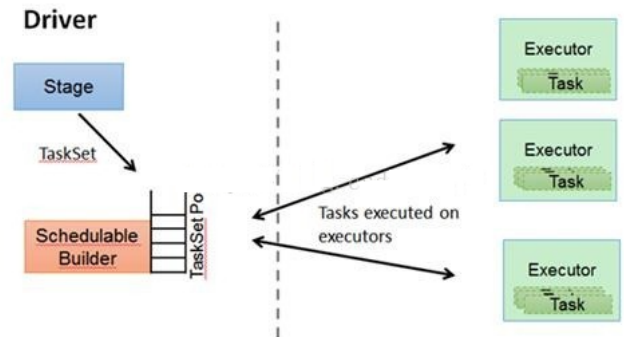
- **Application:** Appliction都是指用户编写的Spark应用程序，其中包括一个Driver功能的代码和分布在集群中多个节点上运行的Executor代码
- **Driver:** Spark中的Driver即运行上述Application的main函数并创建SparkContext，创建SparkContext的目的是为了准备Spark应用程序的运行环境，在Spark中有SparkContext负责与ClusterManager通信，*进行资源申请、任务的分配和监控*等，当Executor部分运行完毕后，Driver同时负责将SparkContext关闭，通常用SparkContext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



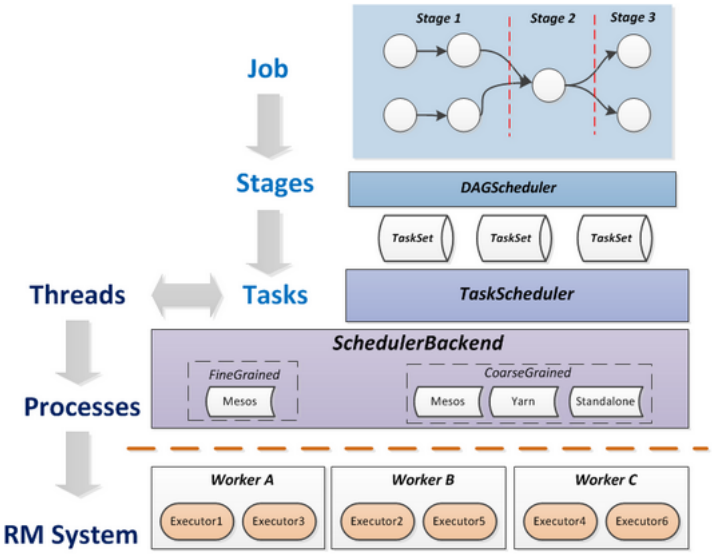
- **TASKSedulter:** 将TaskSET提交给worker运行，每个Executor运行什么Task就是在此处分配的. TaskScheduler维护所有TaskSet，当Executor向Driver发生心跳时，TaskScheduler会根据资源剩余情况分配相应的Task。另外TaskScheduler还维护着所有Task的运行标签，重试失败的Task。下图展示了TaskScheduler的作用



- 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：



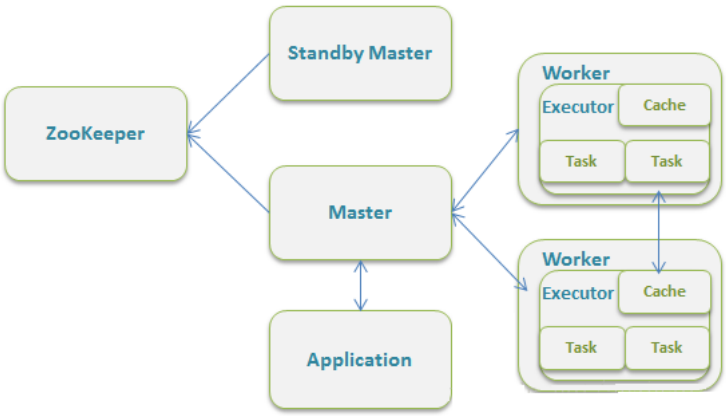
- Job=多个stage，Stage=多个同种task, Task分为ShuffleMapTask和ResultTask，Dependency分为ShuffleDependency和NarrowDependency

Spark运行模式：

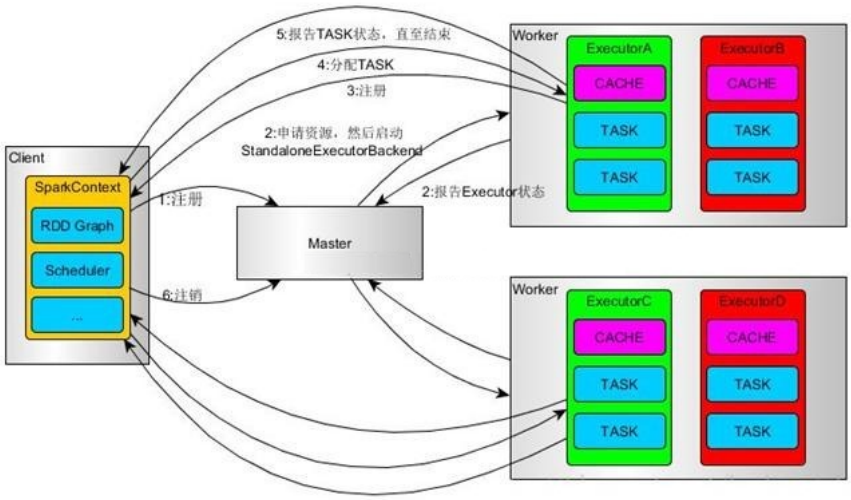
- Spark的运行模式多种多样，灵活多变，部署在单机上时，既可以用本地模式运行，也可以用伪分布模式运行，而当以分布式集群的方式部署时，也有众多的运行模式可供选择，这取决于集群的实际情况，底层的资源调度即可以依赖外部资源调度框架，也可以使用Spark内建的Standalone模式。
- 对于外部资源调度框架的支持，目前的实现包括相对稳定的Mesos模式，以及hadoop YARN模式
- **本地模式：**常用于本地开发测试，本地还分别 local 和 local cluster

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:



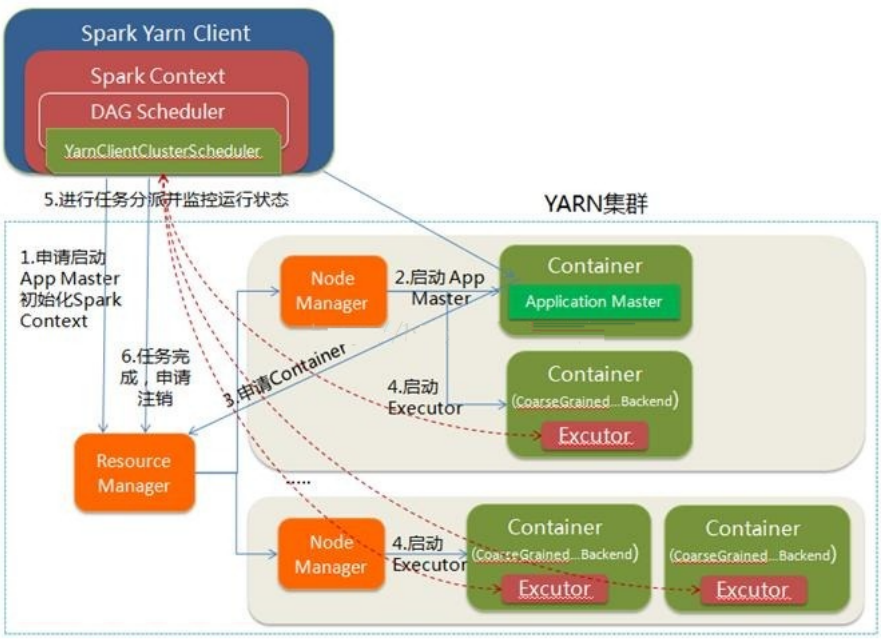
- 该模式主要的节点有Client节点、Master节点和Worker节点。其中Driver既可以运行在Master节点上中，也可以运行在本地Client端。当用spark-shell交互式工具提交Spark的Job时，Driver在Master节点上运行；当使用spark-submit工具提交Job或者在Eclips、IDEA等开发平台上使用”new SparkConf.setManager(“spark://master:7077”)”方式运行Spark任务时，Driver是运行在本地Client端上的
- 运行过程如下图：（*参考至：http://blog.csdn.net/gamer_gyt/article/details/51833681*）



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：



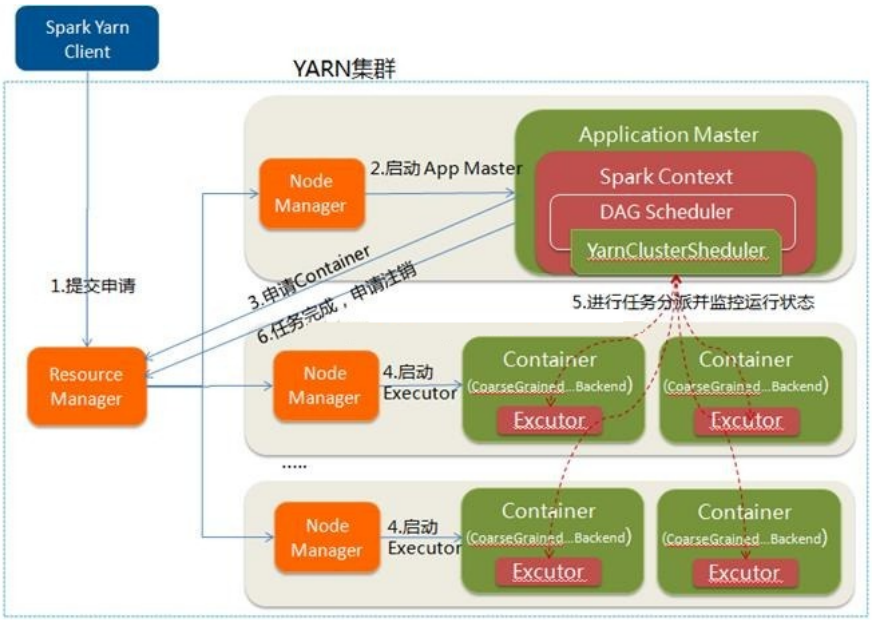
- Spark Yarn Client向YARN的ResourceManager申请启动Application Master。同时在SparkContent初始化中将创建DAGScheduler和TASKScheduler等，由于我们选择的是Yarn-Client模式，程序会选择YarnClientClusterScheduler和YarnClientSchedulerBackend
- ResourceManager收到请求后，在集群中选择一个NodeManager，为该应用程序分配第一个Container，要求它在这个Container中启动应用程序的ApplicationMaster，与YARN-Cluster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

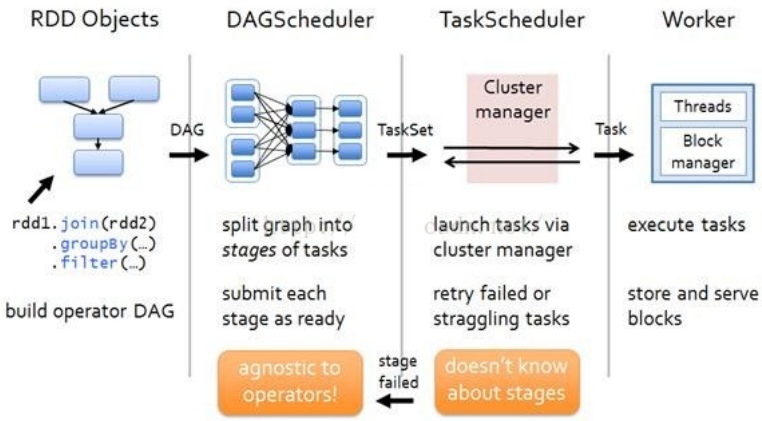
思考： 我们在使用Spark提交job时使用的哪种模式？

RDD运行流程：

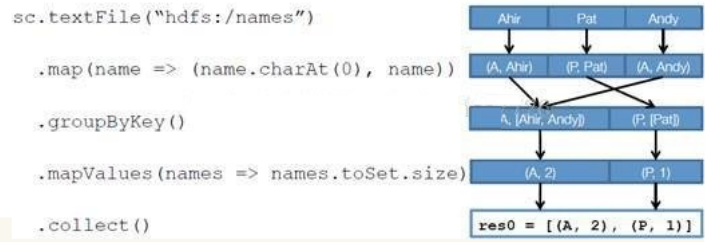
- RDD在Spark中运行大概分为以下三步：

1. 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

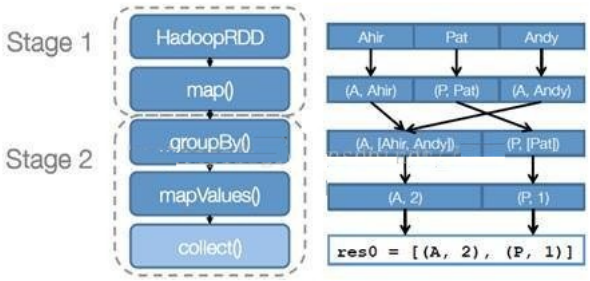
- 示例图如下：



- 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

转自：http://www.cnblogs.com/tgzhu/p/5818374.html

个人分类： spark ▼查看关于本篇文章更多信息

上一篇 centos7 能联通内网，但是不能访问外网网页问题

下一篇 Spark(二): 内存管理