这一节我们将和大家一起学习Spark运行架构基本概念和架构设计

如图6.3.1所示：这是Spark1.3.0官方给出的图片，我们可以看一下，Spark的架构总共就是三大组件：最左边Driver节点，中间是集群资源管理器，最右边是工作节点

另一个概念是Spark Application：就是指Spark应用程序

指的是用户编写的Spark应用程序，比如说我们后面会介绍一个餐饮平台菜品智能推荐系统，那么这个系统程序就是用户编写的Spark 应用程序application，那么一个application包含了Driver功能代码和分布在集群中多个节点上运行的Executor代码。

并且会被分解成一个或多个作业JOB组成，如图6.3.2所示:

好，让我们先来看一下Spark的运行架构：第一个组件Driver：任务控制节点，它相当于一个应用程序的“指挥所”，整个Driver就相当于一个企业，它运行Application的main函数（）并创建SparkContext，相当于CEO的地位。Application，刚才我们说了指的是用户编写的Spark应用程序，

Spark中的Driver即运行上述Application的Main()函数并且创建SparkContext，其中创建SparkContext的目的是为了准备Spark应用程序的运行环境。什么意思呢。就是说我们在开始工作之前，肯定要先布置好工作环境，这个SParkContext有点像给我们配备的工作秘书，在Spark中由SparkContext这个秘书负责和ClusterManager通信，进行资源的申请、任务的分配和监控等；当Executor部分运行完毕后，Driver负责将SparkContext关闭。通常SparkContext代表用上下文，控制整个生命周期。

如图6.3.3所示

Cluster Manager：集群资源管理器，他是帮你对整个集群资源进行调度和管理的，什么是集群资源呢，CPU、内存、带宽、这些都是集群资源，而这些集群资源的管理就是由我们的集群资源管理器Cluster manager来负责的，如图6.3.4所示

我们有很多种不同的集群资源管理器可以选择目前主要有Standalone和Hadoop Yarn, Standalone是Spark原自带的资源管理器，若是使用Yarn模式，就是Hadoop的资源管理器，则是由ResourceManage负责资源的分配，也可以理解为使用Standalone时Cluster manager是Master主节点。若是使用Yarn模式，则是由ResourceManager负责资源的分配。当然还有Mesos和EC2可以选择。

Spark Worker：集群中任何可以运行Application代码的节点，类似于Yarn中的NodeManager节点。在Standalone模式中指的就是通过Slave文件配置的Worker节点，在Spark on Yarn模式中指的就是NodeManager节点，在Spark on Messos模式中指的就是Messos Slave节点，如图6.3.5所示:

Executor：执行器，运行在Worker的Task执行器。Executor启动线程池运行Task，并且负责将数据存在内存或磁盘上，每个Application都会申请各自的Executor来处理任务。如图6.3.6所示

RDD：这是一个非常重要的概念，你会发现整个Spark的学习过程就是不断的和这个RDD打交道，因为RDD就是我们Spark中最核心的数据抽象，我们的数据到了Spark以后，就会被它包装成RDD，就像在数据库中，数据是一条条记录一样，RDD（是Resillient Distributed Dataset的缩写）弹性分布式数据集，是分布式内存的抽象概念，通常RDD很大，会被分成很多个分区，分别保存在不同的节点上，如图6.3.7所示一旦RDD生成以后就是只读的，不能改了，所以我们说RDD是高度受限的。

，它是Spark的基本计算单元，可以通过一系列算子进行操作（主要有Transformation和Action操作），为什么会有 Spark?因为传统的并行计算模型无法有效的解决迭代计算(iterative)和交互式计算(interactive);而Spark的使命便是解决这两个问题，这也是他存在的价值和理由。Spark如何解决迭代计算?其主要实现思想就是RDD，把所有计算的数据保存在分布式的内存中。迭代计算通常情况下都是对同一个数据集做反复的迭代计算，数据在内存中将大大提升IO操作。这也是Spark涉及的核心：内存计算

父RDD每一个分区最多被一个子RDD的分区所用；表现为一个父RDD的分区对应于一个子RDD的分区，或两个父RDD的分区对应于一个子RDD 的分区。如图6.3.8所示:

Task：任务

被送到某个Executor上的工作任务；单个分区数据集上的最小处理流程单元。如图所示:

TaskSet：任务集由一组关联的，但相互之间没有Shuffle依赖关系的任务所组成的任务集。如图6.3.9所示:

为了更加深入的了解Spark的核心原理，有必要先了解几个概念，在图6.3.10中，父RDD每一个分区最多被一个子RDD的分区所用；表现为一个父RDD的分区对应于一个子RDD的分区，或两个父RDD的分区对应于一个子RDD 的分区。

这种叫做窄依赖，父RDD的每个分区都可能被多个子RDD分区所使用，子RDD分区通常对应所有的父RDD分区。这种叫做宽依赖，如图6.3.11所示:

Spark中还有一个重要的概念，就是Stage，一个任务集对应的调度阶段；每个Job会被拆分很多组Task，每组任务被称为Stage，也可称TaskSet，一个作业分为多个阶段；Stage分成两种类型ShuffleMapStage、ResultStage。如图6.3.12所示:1）一个Stage创建一个TaskSet；

2）为Stage的每个Rdd分区创建一个Task,多个Task封装成TaskSet

总体如图6.3.14所示：

从这张图里我们可以看出来，Spark各种概念之间的相互关系：整个用户应用程序在执行的时候呢需要一个Driver，任务控制节点也就是管家节点，，然后一个应用提交以后，会根据代码生成若干个作业（也就是JOB，具体生成多少个作业是取决于里面相关的代码，每个作业又会被切分成很多阶段（Stage），每个阶段包含了很多任务（Task），而这些任务会被派发到不同的工作节点上（也就是Worknode），那么应用程序在执行的时候就由Driver向资源管理器去需要申请资源（因为计算需要CPU，内存）,然后申请到资源以后呢就会启动对应节点的Executor进程，然后会向对应节点的EXEcutor进程去发送应用程序代码和文件，就在它上面派发出线程去执行，具体的任务，然后再把结果返回给Driver，Driver再存储在数据库中。

Spark 运行架构有以下四个特点：1、Executor进程专属

2、支持多种资源管理器

3、Job提交就近原则

4、移动程序而非移动数据的原则执行

1、Executor进程专属

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tasks。Spark Application不能跨应用程序共享数据，除非将数据写入到外部存储系统。如图6.3.15所示:

2、支持多种资源管理器

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，说白了就是这个资源管家由你自己决定。Spark支持资源管理器包含： Standalone、On Mesos、On YARN、Or On EC2。目前使用比较多的是Standalone模式和YARN模式，如图6.3.16所示:

3、Job提交就近原则

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（机架）里，因为Spark Application运行过程中SparkContext和Executor之间有大量的信息交换；如果想在远程集群中运行，最好使用RPC将SparkContext提交给集群，不要远离Worker运行SparkContext。如图6.3.17所示:

4、移动程序而非移动数据的原则执行，数据量巨大的时候，移动数据需要消耗大量时间，所以移动执行程序比移动数据要高效

Task采用了数据本地性和推测执行的优化机制。关键方法：taskIdToLocations、getPreferedLocations。如图6.3.18所示:

6.4 Spark运行基本流程

接下来，我们来一起了解一下spark运行的基本流程：前面我们给大家讲过，整个应用程序提交以后，它有一个指挥所就是Driver，它相当于一个管家，就是应用程序的指挥所，而指挥所我们知道需要一个指挥官，后面我们写代码的时候会经常生成一个叫做Sparkcontext的对象，而这个对象就是指挥官。我们给大家讲过，整个应用程序提交以后，它有一个指挥所就是Driver，它相当于一个管家，就是应用程序的指挥所，而指挥所我们知道需要一个指挥官，后面我们写代码的时候会经常生成一个叫做Sparkcontext的对象，而这个对象就是指挥官。由它负责对整个作业的调度，作业怎么分解，分解之后交给不同的节点去执行，执行完之后怎么把不同节点上执行的结果汇总提交给用户，这些都由指挥官说了算，这个指挥官就是SparkContext，这个指挥官就呆在指挥所里，指挥所就是Driver。如图6.4.1，当应用程序提交了以后，究竟发生了什么呢？

首先需要构建集群，这是最基本的运行环境，整个环境要搭建起来，怎么构建集群呢？集群有了，应用程序提交了，所以应用提交的时候，要先给它运行应用的主节点，就是Driver节点，一旦确定好，指挥所建好以后，就给他配备了指挥官，就是SparkContext对象，如图6.4.2

由这个对象向资源管理器去申请资源，然后负责把整个作业分成不同的阶段，并且把每个阶段的任务调度到不同的工作节点去完成，这都是它的工作。执行过程中，它还要负责监控，一旦失败了，它要负责恢复过来，如图6.4.3

第二步就是资源管理器，如图6.4.4上节我们讲过，资源管理器有几种，可能是YARN，可能是Mesos，资源管理器收到来自SparkContext资源请求之后，资源管理器会为Executor的进程去分配资源去执行，分配什么资源呢，即CPU和内存资源，分配好资源后就可以去启动Executor的进程了，就是由进程再派生出线程，Executor进程是驻留在不同的worknode节点上的，可能有几百甚至上千台机器。这几百上千台机器Worknode上面都是各个Executor进程，如图6.4.5

然后第三步，这个Sparkcontext对象要根据RDD的依赖关系，构建一张DAG图，怎么知道RDD的依赖关系呢，写的代码就是针对RDD的一次又一次操作，每一行语句写下去，就是对他的一次操作，这个操作就会被转换成一个有向无环图就是DAG图，如图6.4.6

生成这个图做什么呢？这个图会被提交到DAG Scheduler的模块，去进行解析，解析它做什么呢，解析它是要把DAG图分解成不同的Stage，如图6.4.7

每个Stage都包含若干个Task，所以Stage是一个Task集合，得到一个又一个任务阶段之后呢，就提交给下一个Taskschedular，如图6.4.8

这个Taskschedular就是具体负责把任务到底分发到哪个节点上面去，，由它来负责，它怎么分发呢，我们整个分发过程了解一下，Taskschedular不会拿到任务就往外扔，而是各个WorkNode上面的那些Executor会主动向TaskTracker去申请运行任务，然后TaskTracker就会把我们的任务根据它申请的情况，扔到相对应的Worknode上，让Executor进程去派送线程去执行，任务怎么分配呢，这里有一个基本原则，什么原则呢，就是这么多的数据要计算到底应该把它扔到哪个节点上面呢，

有一个原则就是要保证计算向数据靠拢。数据在机器A，就优先把数据扔到所在节点，完成本地化处理，而不是把数据远程传输到计算所在的机器B上。TaskSchedular就本着计算向数据靠拢的原则，如果数据在机器A上，就把程序给机器A，而不是把它扔给机器B，这是它的基本原则，TaskScheduler就会向Task运行的地方发送相关的代码，代码就是执行计算的，把它给到机器A。（需要做一个小动画）

第四步，这些任务在Excutor上面运行以后呢，它会得到结果，这个结果又会反馈给TaskSchedule，再由TaskSchedule往上传递给DAGSchedular。然后再由Sparkcontext对象做一个最后的处理，或者返回给用户看，或者写入到HDFS，或者写入到其他的文件当中去。这就是Spark执行的整个流程。

然后就知道SparkContext对象就是整个运行过程的指挥官，而它也代表了应用程序连接集群的通道，应用程序怎么和整个集群里几千台机器发生关联呢，是由这个SparkContext建立的一个通道，（制作一个小动画）

它把应用程序拿过来进行分发，扔到不同的机器上去，所以Sparkcointext就相当于在底层集群之间建立起了这样的一个连接，你是通过这个对象才和底层的几千台机器发生交互的。这正好体现了我们古人“大事化小，小事化了”

处事思想。