**spark应用执行机制分析**

前段时间一直在编写指标代码，一直采用的是--deploy-mode client方式开发测试，因此执行没遇到什么问题，但是放到生产上采用--master yarn-cluster方式运行，那问题就开始陆续暴露出来了。因此写一篇文章分析并记录一下spark的几种运行方式。

**1.spark应用的基本概念**

spark运行模式分为：Local(本地idea上运行)，Standalone,yarn,mesos等，这里主要是讨论一下在yarn上的运行方式，因为这也是最常见的生产方式。

根据spark Application的Driver Program是否在集群中运行，spark应用的运行方式又可以分为Cluster模式和Client模式。

**spark应用涉及的一些基本概念：**

1.mater:主要是控制、管理和监督整个spark集群

2.client：客户端，将用应用程序提交，记录着要业务运行逻辑和master通讯。

3.sparkContext：spark应用程序的入口，负责调度各个运算资源，协调各个work node上的Executor。主要是一些记录信息，记录谁运行的，运行的情况如何等。这也是为什么编程的时候必须要创建一个sparkContext的原因了。

4.Driver Program：每个应用的主要管理者，每个应用的老大，有人可能问不是有master么怎么还来一个？因为master是集群的老大，每个应用都归老大管，那老大疯了。因此driver负责具体事务运行并跟踪，运行Application的main()函数并创建sparkContext。

5.RDD：spark的核心数据结构，可以通过一系列算子进行操作，当Rdd遇到Action算子时，将之前的所有的算子形成一个有向无环图(DAG)。再在spark中转化成为job，提交到集群执行。一个app可以包含多个job

6.worker Node:集群的工作节点，可以运行Application代码的节点，接收mater的命令并且领取运行任务，同时汇报执行的进度和结果给master，节点上运行一个或者多个Executor进程。

7.exector：为application运行在workerNode上的一个进程，该进程负责运行Task，并且负责将数据存在内存或者磁盘上。每个application都会申请各自的Executor来处理任务。

**spark应用(Application)执行过程中各个组件的概念：**

1.Task(任务)：RDD中的一个分区对应一个task，task是单个分区上最小的处理流程单元。

2.TaskSet(任务集)：一组关联的，但相互之间没有Shuffle依赖关系的Task集合。

3.Stage(调度阶段)：一个taskSet对应的调度阶段，每个job会根据RDD的宽依赖关系被切分很多Stage，每个stage都包含 一个TaskSet。

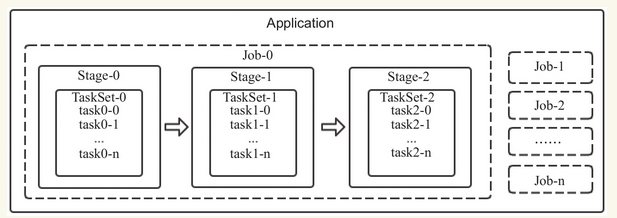
4.job(作业)：由Action算子触发生成的由一个或者多个stage组成的计算作业。

5.application：用户编写的spark应用程序，由一个或者多个job组成，提交到spark之后，spark为application分派资源，将程序转换并执行。

6.DAGScheduler：根据job构建基于stage的DAG，并提交stage给TaskScheduler。

7.TaskScheduler:将Taskset提交给Worker Node集群运行并返回结果。

**spark基本概念之间的关系**



一个Application可以由一个或者多个job组成，一个job可以由一个或者多个stage组成，其中stage是根据宽窄依赖进行划分的，一个stage由一个taskset组成，一个TaskSET可以由一个到多个task组成。

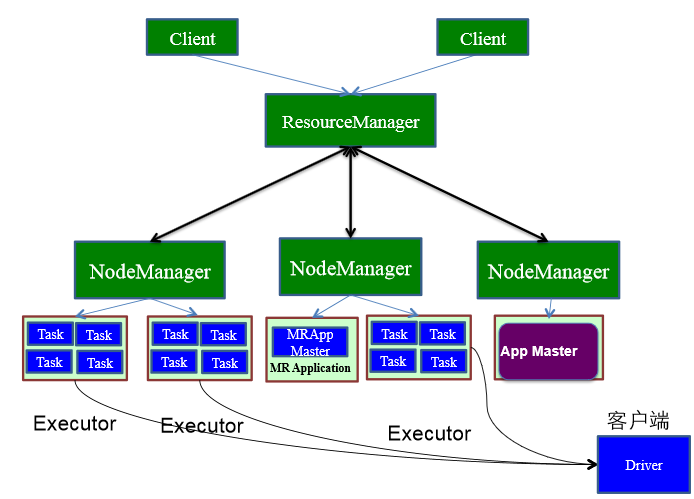
**应用提交与执行**

spark使用driver进程负责应用的解析，切分Stage并且调度task到Executor执行,包含DAGscheduler等重要对象。Driver进程的运行地点有如下两种：

1.driver进程运行在client端，对应用进行管理监控。

2.Master节点指定某个Worker节点启动Driver进程，负责监控整个应用的执行。

**driver运行在client**



用户启动Client端，在client端启动Driver进程。在Driver中启动或实例化DAGScheduler等组件。

1.driver在client启动，做好准备工作，计划好任务的策略和方式（DAGScheduler)后向Master注册并申请运行Executor资源。

2.Worker向Master注册，Master通过指令让worker启动Executor。

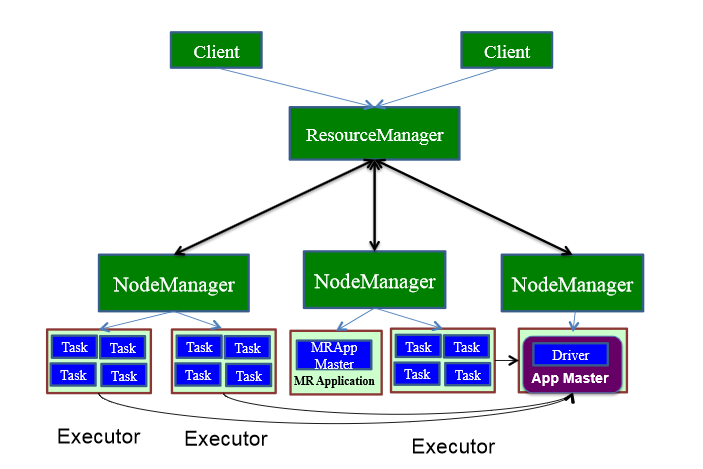
3.worker收到指令后创建ExecutorRunner线程，进而ExecutorRunner线程启动executorBackend进程。

4.ExecutorBackend启动后，向client端driver进程内的SchedulerBackend注册,这样dirver进程就可以发现计算资源了。

5.Driver的DAGScheduler解析应用中的RDD DAG并生成相应的Stage，每个Stage包含的TaskSet通过TaskScheduler分配给Executor，在Exectutor内部启动线程池并行化执行Task，同事driver会密切注视，如果发现哪个execuctor执行效率低，会分配其他exeuctor顶替执行，观察谁的效率更高（推测执行）。

6.计划中的所有stage被执行完了之后，各个worker汇报给driver，同事释放资源，driver确定都做完了，就向master汇报。同时driver在client上，应用的执行进度clinet也知道了。

**Driver运行在Worker节点**



用户启动客户端，客户端提交应用程序给Master

1.Master调度应用，指定一个worker节点启动driver，即Scheduler-Backend。

2.worker接收到Master命令后创建driverRunner线程，在DriverRunner线程内创建SchedulerBackend进程，Dirver充当整个作业的主控进程。

3.Master指定其他Worker节点启动Exeuctor，此处流程和上面相似，worker创建ExecutorRunner线程，启动ExecutorBackend进程。

4.ExecutorBackend启动后，向client端driver进程内的SchedulerBackend注册,这样dirver进程就可以发现计算资源了。

5.Driver的DAGScheduler解析应用中的RDD DAG并生成相应的Stage，每个Stage包含的TaskSet通过TaskScheduler分配给Executor，在Exectutor内部启动线程池并行化执行Task，同事driver会密切注视，如果发现哪个execuctor执行效率低，会分配其他exeuctor顶替执行，观察谁的效率更高（推测执行）。

6.计划中的所有stage被执行完了之后，各个worker汇报给driver，同事释放资源，driver确定都做完了，就向master汇报。客户也会跳过master直接和drive通讯了解任务的执行进度。