Spark on YARN客户端模式作业运行全过程分析

《Spark on YARN集群模式作业运行全过程分析》
《Spark on YARN客户端模式作业运行全过程分析》
《Spark:Yarn-cluster和Yarn-client区别与联系》
《Spark和Hadoop作业之间的区别》
《Spark Standalone模式作业运行全过程分析》（未发布）

在前篇文章中我介绍了Spark on YARN集群模式（yarn-cluster）作业从提交到运行整个过程的情况（详情见《Spark on YARN集群模式作业运行全过程分析》），我们知道Spark on yarn有两种模式：yarn-cluster和yarn-client。这两种模式作业虽然都是在yarn上面运行，但是其中的运行方式很不一样，今天我就来谈谈Spark on YARN yarn-client模式作业从提交到运行的过程剖析。

和yarn-cluster模式一样，整个程序也是通过spark-submit脚本提交的。但是yarn-client作业程序的运行不需要通过Client类来封装启动，而是直接通过反射机制调用作业的main函数。下面就来分析：

1、通过SparkSubmit类的launch的函数直接调用作业的main函数（通过反射机制实现），如果是集群模式就会调用Client的main函数。

2、而应用程序的main函数一定都有个SparkContent，并对其进行初始化；

3、在SparkContent初始化中将会依次做如下的事情：设置相关的配置、注册MapOutputTracker、BlockManagerMaster、BlockManager，创建taskScheduler和dagScheduler；其中比较重要的是创建taskScheduler和dagScheduler。在创建taskScheduler的时候会根据我们传进来的master来选择Scheduler和SchedulerBackend。由于我们选择的是yarn-client模式，程序会选择YarnClientClusterScheduler和YarnClientSchedulerBackend，并将YarnClientSchedulerBackend的实例初始化YarnClientClusterScheduler，上面两个实例的获取都是通过反射机制实现的，YarnClientSchedulerBackend类是CoarseGrainedSchedulerBackend类的子类，YarnClientClusterScheduler是TaskSchedulerImpl的子类，仅仅重写了TaskSchedulerImpl中的getRackForHost方法。

4、初始化完taskScheduler后，将创建dagScheduler，然后通过taskScheduler.start()启动taskScheduler，而在taskScheduler启动的过程中也会调用SchedulerBackend的start方法。在SchedulerBackend启动的过程中将会初始化一些参数，封装在ClientArguments中，并将封装好的ClientArguments传进Client类中，并client.runApp()方法获取Application ID。

5、client.runApp里面的做是和前面客户端进行操作那节类似，不同的是在里面启动是ExecutorLauncher（yarn-cluster模式启动的是ApplicationMaster）。

6、在ExecutorLauncher里面会初始化并启动amClient，然后向ApplicationMaster注册该Application。注册完之后将会等待driver的启动，当driver启动完之后，会创建一个MonitorActor对象用于和CoarseGrainedSchedulerBackend进行通信（只有事件AddWebUIFilter他们之间才通信，Task的运行状况不是通过它和CoarseGrainedSchedulerBackend通信的）。然后就是设置addAmIpFilter，当作业完成的时候，ExecutorLauncher将通过amClient设置Application的状态为FinalApplicationStatus.SUCCEEDED。

7、分配Executors，这里面的分配逻辑和yarn-cluster里面类似，就不再说了。

8、最后，Task将在CoarseGrainedExecutorBackend里面运行，然后运行状况会通过Akka通知CoarseGrainedScheduler，直到作业运行完成。

9、在作业运行的时候，YarnClientSchedulerBackend会每隔1秒通过client获取到作业的运行状况，并打印出相应的运行信息，当Application的状态是FINISHED、FAILED和KILLED中的一种，那么程序将退出等待。

10、最后有个线程会再次确认Application的状态，当Application的状态是FINISHED、FAILED和KILLED中的一种，程序就运行完成，并停止SparkContext。整个过程就结束了。