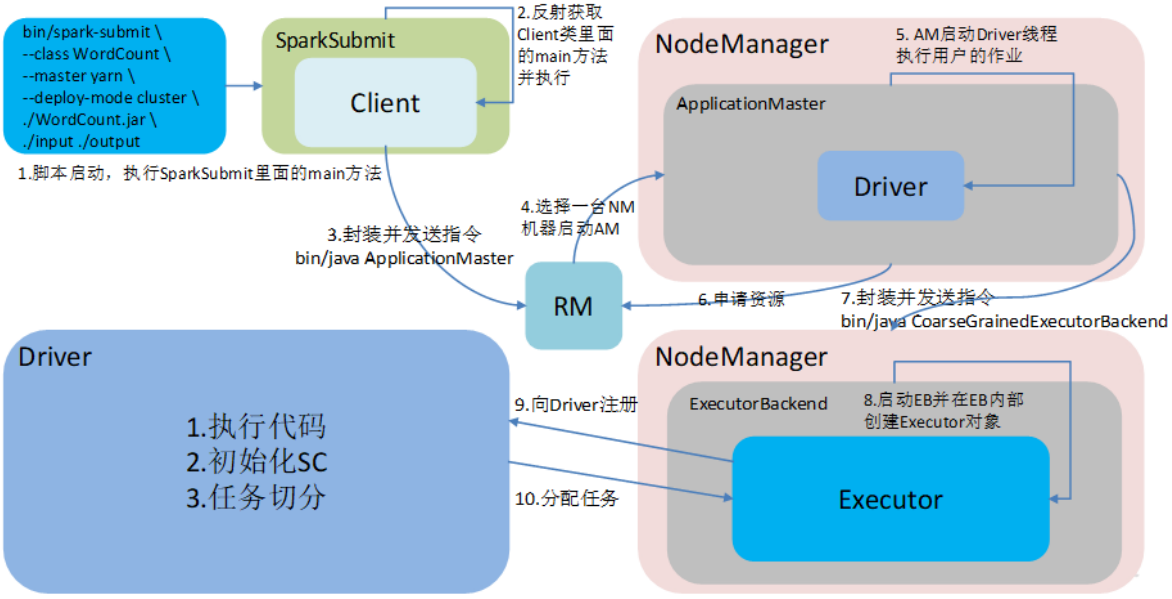
> 乘法思维！！！

Spark模式运行机制

Yarn模式运行机制

Yarn-Cluster模式



（1）执行脚本提交任务，实际是启动一个SparkSubmit的JVM进程；

（2）SparkSubmit类中的main方法反射调用Client的main方法；

（3）Client创建Yarn客户端，然后向Yarn发送执行指令：bin/java ApplicationMaster；

（4）Yarn框架收到指令后会在指定的NM中启动ApplicationMaster；

（5）ApplicationMast启动Driverer线程，执行用户的作业；

（6）AM向RM注册，申请资源；

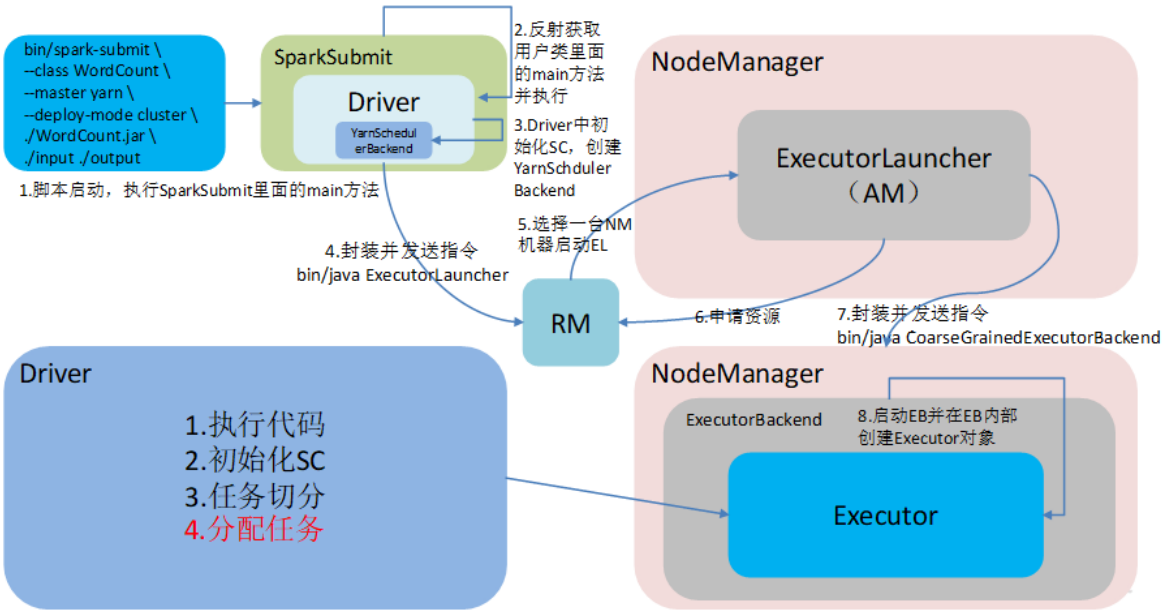
（7）获取资源后AM向NM发送指令：bin/java CoarseGrainedExecutorBacken；

（8）ExecutorBackend进程会接收消息，启动计算对象Executor并跟Driver通信，注册已经启动的Executor；

（9）Driver分配任务并监控任务的执行。

注意：SparkSubmit、ApplicationMaster和CoarseGrainedExecutorBacken是独立的进程；Client和Driver是独立的线程；Executor是一个对象。

Yarn-Client模式



（1）执行脚本提交任务，实际是启动一个SparkSubmit的JVM进程；

（2）SparkSubmit类中的main方法反射调用用户代码的main方法；

（3）启动Driver线程，执行用户的作业，并创建ScheduleBackend；

（4）YarnClientSchedulerBackend向RM发送指令：bin/java ExecutorLauncher；

（5）Yarn框架收到指令后会在指定的NM中启动ExecutorLauncher（实际上还是调用ApplicationMaster的main方法）

```
object ExecutorLauncher {
 def main(args: Array[String]): Unit = {
  ApplicationMaster.main(args)
 }
}
```

（6）AM向RM注册，申请资源；

（7）获取资源后AM向NM发送指令：bin/java CoarseGrainedExecutorBacken；

（8）ExecutorBackend进程会接收消息，启动计算对象Executor并跟Driver通信，注册已经启动的Executor；

（9）Driver分配任务并监控任务的执行。

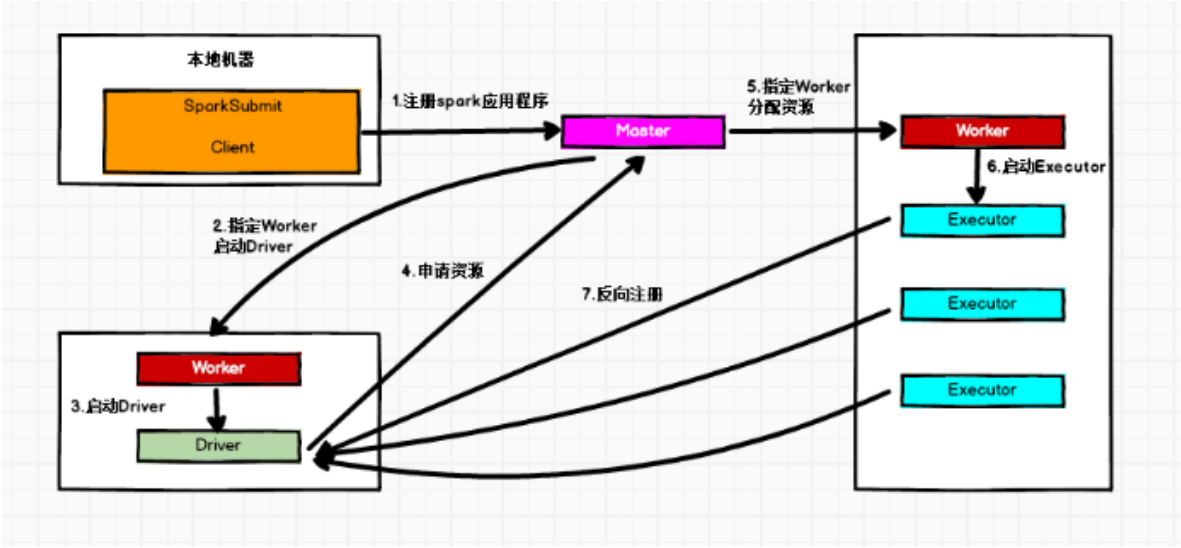
注意：SparkSubmit、ExecutorLauncher和CoarseGrainedExecutorBacken是独立的进程；Client和Driver是独立的线程；Executor是一个对象。【有问题都可以私聊我WX：focusbigdata，或者关注我的公众号：FocusBigData，注意大小写】

Standalone模式运行机制

Standalone集群有2个重要组成部分，分别是：

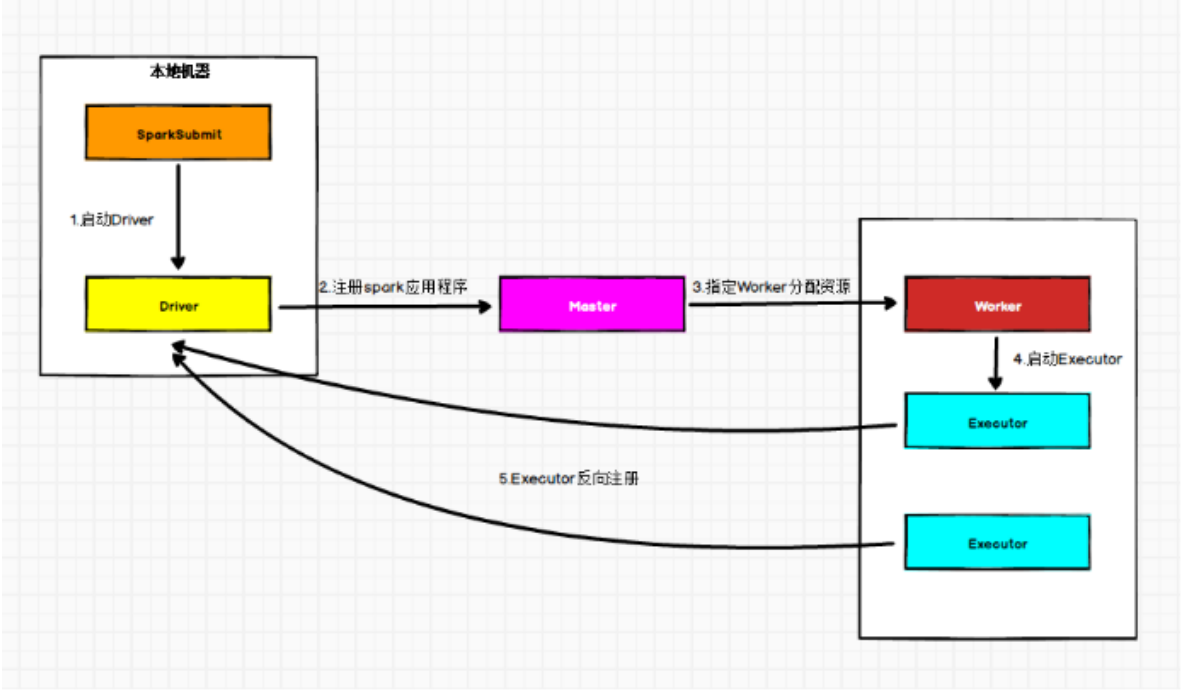
> （1）Master(RM)：是一个进程，主要负责资源的调度和分配，并进行集群的监控等职责；
>
> （2）Worker(NM)：是一个进程，一个Worker运行在集群中的一台服务器上，主要负责两个职责，一个是用自己的内存存储RDD的某个或某些partition；另一个是启动其他进程和线程（Executor），对RDD上的partition进行并行的处理和计算。

Standalone-Cluster模式



> 在Standalone Cluster模式下，任务提交后，Master会找到一个Worker启动Driver。Driver启动后向Master注册应用程序，Master根据submit脚本的资源需求找到内部资源至少可以启动一个Executor的所有Worker，然后在这些Worker之间分配Executor，Worker上的Executor启动后会向Driver反向注册，所有的Executor注册完成后，Driver开始执行main函数，之后执行到Action算子时，开始划分Stage，每个Stage生成对应的taskSet，之后将Task分发到各个Executor上执行。

Standalone-Client模式



> 在Standalone Client模式下，Driver在任务提交的本地机器上运行。Driver启动后向Master注册应用程序，Master根据submit脚本的资源需求找到内部资源至少可以启动一个Executor的所有Worker，然后在这些Worker之间分配Executor，Worker上的Executor启动后会向Driver反向注册，所有的Executor注册完成后，Driver开始执行main函数，之后执行到Action算子时，开始划分Stage，每个Stage生成对应的TaskSet，之后将Task分发到各个Executor上执行。