一：通过Spark-shell运行程序来观察TaskScheduler内幕

1. 当我们启动Spark-shell本身的时候命令终端反馈回来的主要是ClientEndpoint和SparkDeploySchedulerBackend，这是因为此时还没有任何Job的触发，这是启动Application本身而已，所以主要就是实例化SparkContext并注册当前的应用程序给Master且从集群中获得ExecutorBackend计算资源。
2. DAGScheduler划分好Stage后会通过TaskSchedulerImpl中的TaskSetManager来管理当前要运行的Stage中的所有任务TaskSet，TaskSetManager会根绝locality aware来为Task分配计算资源，监控Task的执行状态（例如重试，满任务进行推测或执行）；

二：TaskScheduler与SchedulerBackend

1. 总体的底层任务调度的过程如下：
   1. TaskSchedulerImpl。submitTasks：主要的作用就是讲TaskSet加入到TaskSetManager中进行管理；
   2. SchedulerBuilder.addTaskSetManager：Scheduler会确定TaskSetManager的运行顺序，然后按照TaskSetManager的locality aware来确定每个Task具体在哪个ExecutorBackend中；
   3. CoarseGrainedSchedulerBackend。reviveOffers（）给DiverEndpoint发送ReviveOffers，ReviveOffers本身是一个空的case object对象，只是起到触发底层资源调度的作用，在有Task提交或者计算资源变动的时候会发送ReviveOffers这个消息作为触发器；
   4. 在DirverEndpoint接受ReviveOffers消息并路由到makeOffers具体的方法中，在makeOffers方法中首先准备好所有可以用于计算的workOffers（代表了所有可用ExecutorBackend中可以使用的cores等信息）；
   5. TaskSchedulerImpl。resourceOffers：为每一个Task具体分配计算资源，输入时ExecutorBackend及其上可用的cores，输出是TaskDescription的二维数组，在其中确定了每个Task具体运行在哪个ExecutorBackend，resourceOffers到底是如何确定Task具体运行在哪个ExecutorBackend上呢？算法的实现具体如下：
      1. 通过Random.shuffle方法重新洗牌所有的计算资源以寻求计算的负载均衡；
      2. 根据每个ExecutorBackend的cores的个数声明类型为TaskDescription的ArrayBuffer数组；
      3. 如果有新的ExecutorBackend分配给我们的Job，此时会调用executorAdded来获得最新的完整的可用计算资源；
      4. 通过下述代码追求最高级别的优先级本地性：

**for** (taskSet <- sortedTaskSets; maxLocality <- taskSet.*myLocalityLevels*) {  
 **do** {  
 launchedTask = resourceOfferSingleTaskSet(  
 taskSet, maxLocality, shuffledOffers, availableCpus, tasks)  
 } **while** (launchedTask)  
}

* + 1. 通过调用TaskSetManager的ResourceOffer最终确定每个Task具体运行在哪个ExecutorBackend的具体的Locality level；
  1. 通过launchTasks把任务发送给ExecutorBackend去执行；

补充：

1. Task默认的最大重试次数是4次；
2. Spark默认程序目前支持两种调度器：FIFO，FAIR，可以通过spark-env。Sh中spark。Scheduler。mode进行具体的设置，默认情况下是FIFO；
3. TaskScheduler中要负责为Task分配计算资源：此时程序已经具备集群中的计算资源了，根据计算本地性原则来确定Task要具体运行在哪个ExecutorBackend中，此时程序已经具备集群中的计算资源了；
4. TaskDescription中已经确定好了Task具体要运行在哪个ExecutorBackend中，而确定Task具体运行在哪个ExecutorBackend上的算法是由taskSetManager的resourceOffer方法确定的。
5. 数据本地性优先级从高到低依次为：PROCESS\_LOCAL,NODE\_LOCAL,NO\_PREF,RACK\_LOCAL,ANY,其中NO\_PREF是指机器本地性；
6. 每个Task默认是采用一个线程进行计算的；
7. DAGScheduler是从数据层面考虑preferedLocation的，而TaskScheduler是从具体计算Task角度考虑计算的本地性；