1. DAGScheduler面向Job划分出了不同的Stage,划分Stage之后,Stage是从后往前划分的,从前往后执行的,每一个Stage内部有一系列的任务,Stage里面的任务是并行计算的,这些并行计算的任务的逻辑是完全相同的,只不过是处理的数据不同而已.DAGScheduler会以TaskSet的方式把我们一个DAGScheduler构造的每个Stage中的所有任务 提交给底层的调度器TaskScheduler,TaskScheduler是一个接口,他作为接口的好处是跟具体的调度器解耦合,这样Spark就可以运行在不同的资源调度的模式上,包括Standalone,Yarn,Mesos之上.交给TaskScheduler，TaskScheduler中会通过TaskSetManager来管理我们具体的任务，TaskScheduler的核心任务是提交TaskSet到集群运算并汇报结果。

一：TaskScheduler原理解密

1. DAGScheduler在提交TaskSet给底层调度器的时候是面向接口TaskScheduler的，这符合面向对象中依赖抽象而不依赖具体的原则，带来了底层资源调度器的可插拔性，导致了Spark可以运行在众多的资源调度模式上，例如Standalone，Yarn、Mesos，Loca，EC2，其他自定义的资源调度器；在Standalone的模式下我们聚焦于TaskSchedulerImpl。
2. TaskScheduler的核心任务是提交TaskSet到集群运算并汇报结果。
   1. 为TaskSet创建和维护一个TaskSetManager并追踪任务的本地性以及错误信息；
   2. 遇到staggle任务会放到其他的节点进行重试；
   3. 想DAGScheduler汇报执行情况，包括在shuffle输出lost的时候报告fetchfailed错误等信息；
3. TaskScheduler内部会握有SchedulerBackend，从Standalone的模式来讲具体实现是SparkDeploySchedulerBackend。
4. SparkDeploySchedulerBackend在启动功能的时候构造了AppClient实例并在该实例start的时候启动了ClientEndpoint这个消息循环体，ClientEndpoint在启动的时候会向Master注册当前程序，而SparkDeploySchedulerBackend的父类CoarseGrainedSchedulerBackend在start的时候会实例化类型为DriverEndpoint（这就是我们程序运行时候的经典对象Driver）的消息循环体，SparkDeploySchedulerBackend专门负责收集Worker上的资源信息，当ExecutorBackend启动的时候会发送RegisteredExecutor信息向DriverEndpoint注册，此时SparkDeploySchedulerBackend就掌握了当前应用程序拥有的计算资源，TaskScheduler就是通过SparkDeploySchedulerBackend拥有的计算资源来具体运行Task。
5. SparkContext，DAGScheduler，TaskSchedulerImpl，SparkDeploySchedulerBackend在应用程序启动的时候只实例化一次，应用程序存在期间始终存在这些对象。

大总结：在SparkContext实例化的时候调用createTaskScheduler来创建TaskSchedulerImpl和SparkDeploySchedulerBackend，同时在SparkContext实例化的时候会调用TaskSchedulerImpl的start，在start方法中会调用SparkDeploySchedulerBackend的start，在该start方法中会创建AppClient对象并调用AppClient对象的start方法，在start方法中会创建ClentEndpoint，在创建ClientEndpoint会传入Command来指定具体为当前应用程序启动的Executor进程的入口类的名称为CoarseGrainedExecutorBackend，然后ClientPoint启动并通过tryRegisterMaster来注册当前的应用程序到Master中，Master接受到注册信息后如果可以运行程序，则会为该程序生成JobID并通过Schedule来分配计算资源，具体计算资源的分配是通过应用程序的运行方法、Memory、Cores等配置信息来决定的，最后Master会发送指令给Worker，Worker中为当前应用程序分配计算资源时会首先分配ExecutorRunner，ExecutorRunner内部会通过Thread的方式构建ProcessBuilder来启动另外一个JVM进程，这个jVM进程启动时候加载的main方法所在的类的名称就是在创建ClientEndpoint时候传入的command指定具体名称为CoarseGrainedExecutorBackend的类，此时JVM在通过ProcessBuilder启动的时候获得了CoarseGrainedExecutorBackend后加载并调用其中的main方法，在main方法中会实例化CoarseGrainedExecutorBackend本身这个消息循环体，而CoarseGrainedExecutorBackend在实例化的时候会通过回调Onstart向DriverEndpoint发送RegisterExecutor来注册当前的CoarseGrainedExecutorBackend，此时DirverEndpoint收到该注册信息并保存在了SparkDeploySchedulerBackend实例的内存数据结构中，这样Driver就获得了计算资源！