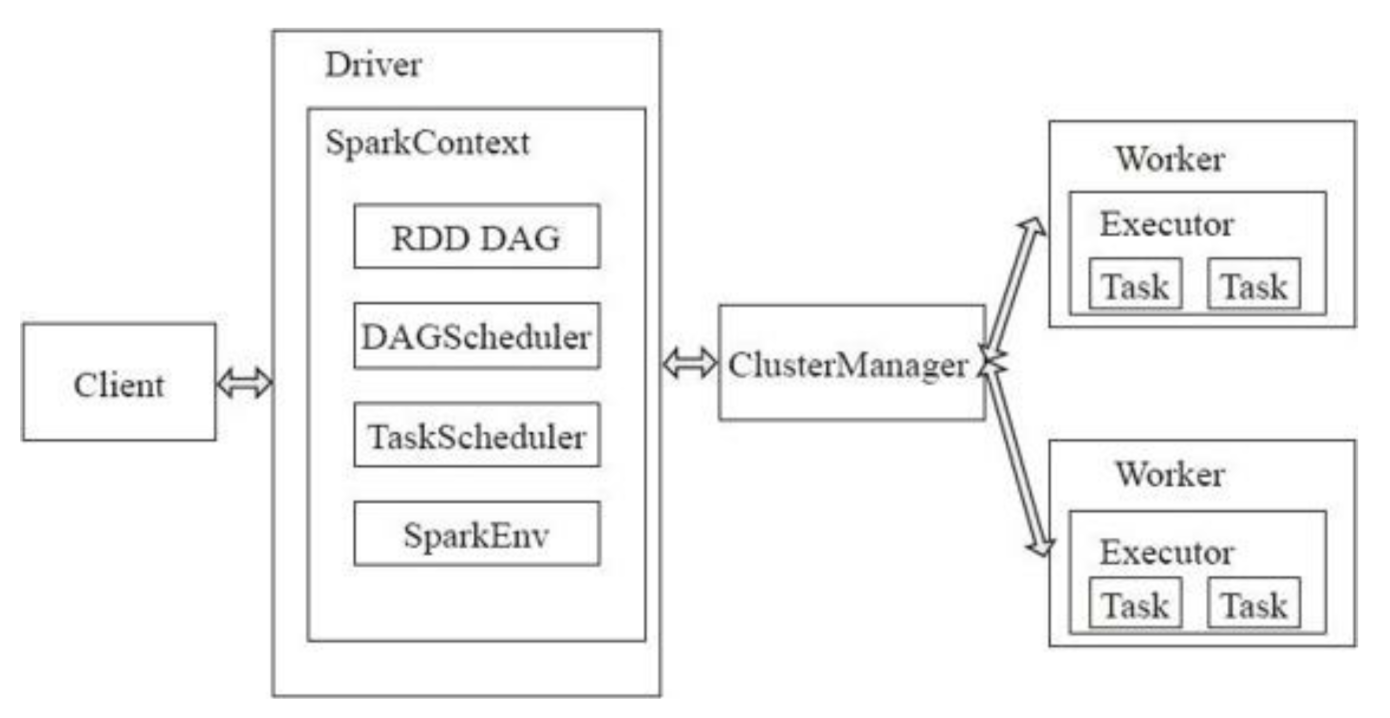
1. 简答题（共5题，每题20分，总100分）



图（1）

1. 根据图（1）解释对应部件在Spark中的作用？

Worker: 相当于计算节点，接收主节点命令与进行状态汇报，启动Executor或Driver

Master: 作为整个集群的控制器，负责整个集群的正常运行

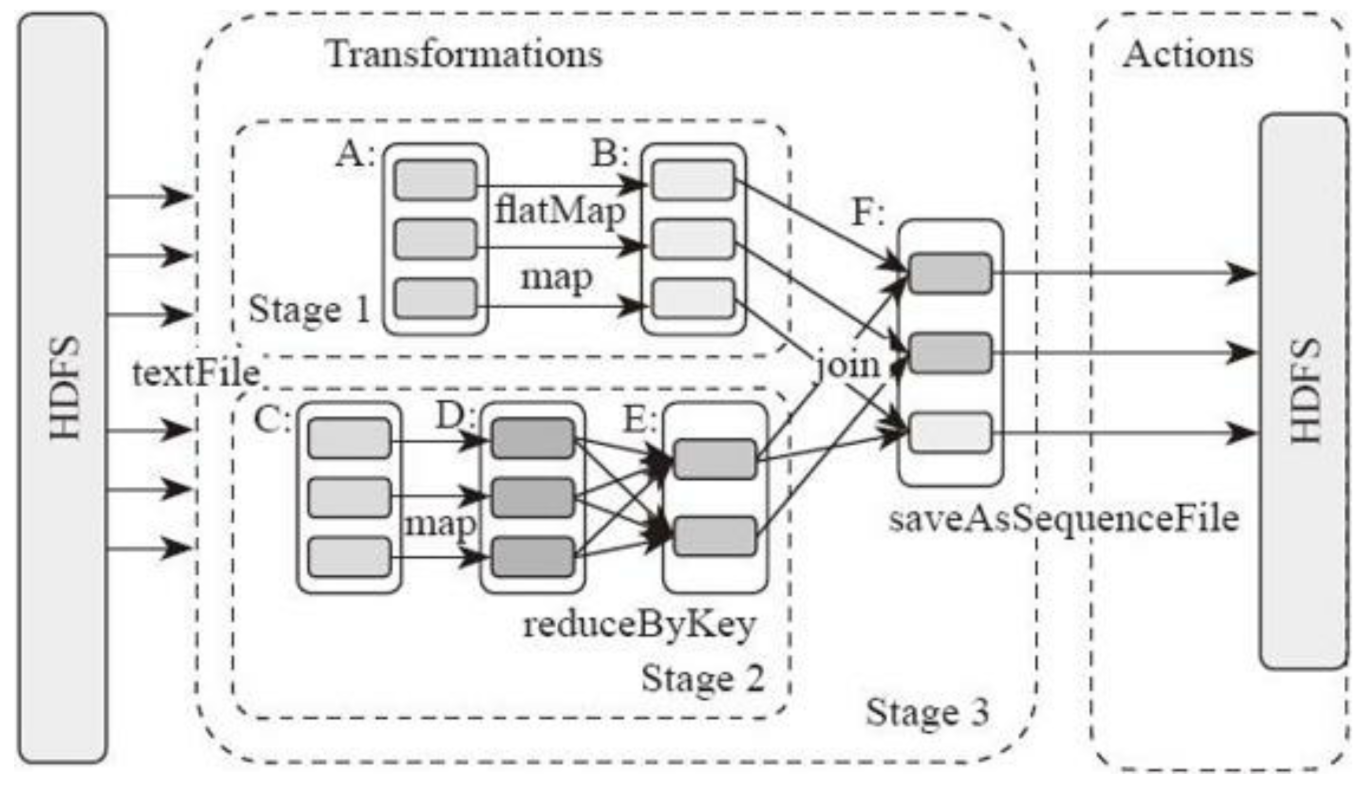
ClusterManager: 在Standalone模式中即为Master，在Yarn模式下为资源管理器

Driver: 负责控制一个应用的执行

Executor: 负责任务的执行

1. 根据图（1）简单描述当client发起命令后，在spark中如何运作？

Client提交应用，Master找到一个Worker启动Driver，Driver向Master或者资源管理器申请资源，之后将应用转化为RDD DAG，再由DAG Schedule将RDD DAG转换为Stage的有向无环图提交给Task Schedule，由TaskSchedule提交任务给Executor执行。在任务执行的过程中，其他组件协同工作，确保整个应用正常执行。



图（2）

1. 根据图（2）简单描述spark的执行逻辑

Action算子触发之后，将所有累积的算子形成有向无环图，然后由调度器调度图上的任务进行运算。Spark根据RDD之间不同的依赖关系切分形成不同的阶段，一个阶段包含一系列执行流水线。

上图A、B、C、D、E、F分别代表不同的RDD,RDD内的方框代表分区。数据从HDFS读取，得到RDD A和RDD C,RDD C上执行map操作，转换为RDD D,RDD B和RDD E进行join操作获得RDD F,在B和E连接转化的过程中会进行Shuffle，最后RDD F通过函数saveAsSequenceFile将数据保存到HDFS中。

1. （1）编写程序并描述创建RDD的方式（15分）

**import** org.apache.spark.{SparkContext, SparkConf}

**object** SparkTest {

**def** main(args: Array[String]):Unit ={

**val** conf = **new** SparkConf().setAppName(**"appName"**).setMaster(**"local[4]"**)

**val** sc = **new** SparkContext(conf)

**val** file = sc.textFile(**this**.getClass.getClassLoader.getResource(**"file"**).getFile)

**val** warn = file.filter(line=> {

line.contains(**"dst\_ip"**)

})

**val** count=warn.count()

*println*(count)

}

}

“file”是RDD元数据，经过算法filter过滤出包含”dst\_ip”的数据行，创建新的RDD。

（2）Spark中RDD算子有哪几种（5分）

两种算子：**transformation**和action

1. 利用Spark map算子和foreach算子将以下的日志中的dst\_ip,src\_ip转成ip类型，并遍历。(提示：这是json类型，利用fastJson对日志进行处理)

{"dst\_as":0,"dst\_ip":1885335600,"sampling\_algorithm":0,"src\_ip":3232243929,"protocol":6,"time\_received":1516410237629}

{"dst\_as":0,"dst\_ip":3395944468,"sampling\_algorithm":0,"src\_ip":3232237789,"protocol":17,"time\_received":1516410237633}

{"dst\_as":0,"dst\_ip":1987799559,"sampling\_algorithm":0,"src\_ip":3232250528,"protocol":6,"time\_received":1516410237629}

**object** SparkMapForeach {

**def** main(args: Array[String]):Unit ={

**val** conf = **new** SparkConf().setAppName(**"appName"**).setMaster(**"local[4]"**)

**val** sc = **new** SparkContext(conf)

**val** file = sc.textFile(**this**.getClass.getClassLoader.getResource(**"file.txt"**).getFile)

**val** dst\_ip = {

file.map(line => (line.split(**","**)(1), 10))

}

**val** src\_ip = {

file.map(line => (line.split(**","**)(3), 10))

}

}

}