1原理：

Spark Application的运行架构由两部分组成：driver program（SparkContext）和executor。Spark Application一般都是在集群中运行，比如Spark Standalone、YARN、mesos，这些集群给Spark Applicaiton提供了计算资源和这些资源管理，这些资源既可以给executor运行，也可以给driver program 运行。根据Spark Application的driver program是否在资源集群中运行，Spark Application的运行方式又可以分为Cluster模式和Client模式

Spark的Application在运行时，首先在Driver程序中会创建SparkContext，将其作为调度的总入口，在其初始化的过程中会分别创建DAGSchedule（进行Stage调度）和TaskSchedule（进行Task调度）两个模块。DAGSchedule模块是基于Stage的调度模块，它为每个Spark job计算具有依赖关系的多个Stage任务阶段，然后将每个Stage划分为一组具体的任务（通常会考虑数据的本地性）以TaskSet的形式提交给底层的TaskSchedule模块来具体执行。TaskSchedule负责具体启动任务，监控和汇报任务的运行情况。而任务运行所需的资源需要向Cluster Manager申请。

  
Spark基本工作流程

    每个Spark Application都有自己的Executor进程，此进程的生命周期和整个Application的生命周期相同，此进程内部维持着多个线程来并行的执行分配给他的Task。这种运行形式有利于进行不同Application之间的资源、调度隔离，但也意味着不同的Application之间难以做到相互通信和信息交换。

2场景：

图计算，迭代计算 交互式查询计算

机器学习

3运行环境：spark安装 jdk hadoop

4开发环境搭建：

5 Eclipse环境搭建:

Jar包引入

切换到: /usr/lib/discover/lib/

spark streaming 消费:

spark-submit --class com.kit.spark\_demo.MsgFromkfkToHbase --executor-memory 20G --total-executor-cores 100 /usr/lib/discover/lib/kfkspkhbase-0.0.1-SNAPSHOT.jar 100

代码讲解：

日志查看:

Spark常用操作命令： 执行程序后打印日志到指定文本

run-example JavaWordCount ./wordcountdata.txt > Sparkpilog.txt

run-example SparkPi 10 > Sparkpilog.txt

run-example SparkPi

spark-submit --class org.apache.spark.examples.SparkPi ../lib/spark-examples-1.6.1-hadoop2.4.0.jar 100

spark-submit --class org.apache.spark.examples.SparkPi --master spark://172.16.19.151:7077 ../lib./ spark-examples-1.6.1-hadoop2.4.0.jar 100

spark-submit --class org.apache.spark.examples.JavaSparkPi --master local /usr/lib/spark-1.6.1/lib/spark-examples-1.6.1-hadoop2.4.0.jar 100

spark-submit --class org.apache.spark.examples.JavaSparkPi --master spark://172.16.19.151:7077 --executor-memory 20G --total-executor-cores 100 /usr/lib/spark-1.6.1/lib/spark-examples-1.6.1-hadoop2.4.0.jar 100

# For Python examples, use spark-submit directly:

./bin/spark-submit examples/src/main/python/pi.py

# For R examples, use spark-submit directly:

./bin/spark-submit examples/src/main/r/dataframe.R

1. 运行./spark-shell.sh

2. scala> val data = Array(1, 2, 3, 4, 5) //产生data

data: Array[Int] = Array(1, 2, 3, 4, 5)

3. scala> val distData = sc.parallelize(data) //将data处理成RDD

distData: spark.RDD[Int] = spark.ParallelCollection@7a0ec850 （显示出的类型为RDD）

4. scala> distData.reduce(\_+\_) //在RDD上进行运算，对data里面元素进行加和

12/05/10 09:36:20 INFO spark.SparkContext: Starting job...

5. 最后运行得到

12/05/10 09:36:20 INFO spark.SparkContext: Job finished in 0.076729174 s

res2: Int = 15

spark local模式 ：linux命令执行：

spark-submit --class com.kit.spark\_demo.JavaSparkPiTest --master local[8] ./spark\_demo-0.0.1-SNAPSHOT.jar 100



Yarn client模式提交

spark-submit --class com.kit.spark\_demo.JavaSparkPiTest --master yarn-client ./spark\_demo-0.0.1-SNAPSHOT.jar 100

读写指定目录文件夹下面的文件:

val textFile = sc.textFile("wordcountdata.txt")

spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client --executor-memory 20G --total-executor-cores 100 /usr/lib/discover/lib/spark-examples-1.5.1-hadoop2.5.2-transwarp.jar 100

spark-submit --class com.kit.spark\_demo.JavaDirectKafkaWordCount --master yarn-client --executor-memory 20G --total-executor-cores 100 /usr/lib/discover/lib/ sparkdemo-0.0.1-SNAPSHOT.jar 100





