# Discover 操作手册

## 1 原理

<http://mt.sohu.com/20160522/n450849016.shtml>

<https://www.zybuluo.com/xiaop1987/note/106718#executors>

弹性分布式数据集(RDD，从Spark 1.3版本开始已被DataFrame替代)是Apache Spark的核心理念。它是由数据组成的不可变分布式集合，其主要进行两个操作：transformation和action。Transformation是类似在RDD上做 filter()、map()或union() 以生成另一个RDD的操作，而action则是count()、first()、take(n)、collect() 等促发一个计算并返回值到Master或者稳定存储系统的操作。Transformations一般都是lazy的，直到action执行后才会被执行。Spark Master/Driver会保存RDD上的Transformations。这样一来，如果某个RDD丢失(也就是salves宕掉)，它可以快速和便捷地转换到集群中存活的主机上。这也就是RDD的弹性所在。

尽管MapReduce适用大多数批处理工作，并且在大数据时代成为企业大数据处理的首选技术，但由于以下几个限制，它对一些场景并不是最优选择：

缺少对迭代计算以及DAG运算的支持

Shuffle过程多次排序和落地，MR之间的数据需要落Hdfs文件系统

Spark在很多方面都弥补了MapReduce的不足，比MapReduce的通用性更好，迭代运算效率更高，作业延迟更低，它的主要优势包括：

提供了一套支持DAG图的分布式并行计算的编程框架，减少多次计算之间中间结果写到Hdfs的开销

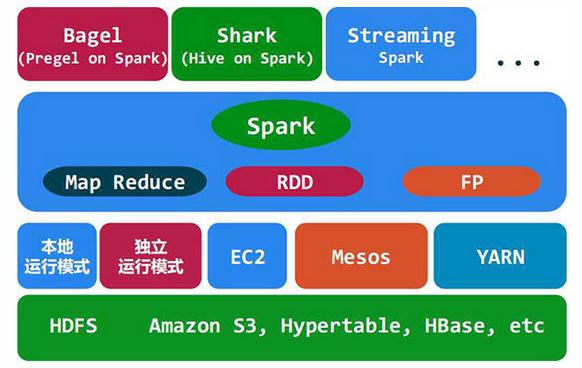
提供Cache机制来支持需要反复迭代计算或者多次数据共享，减少数据读取的IO开销

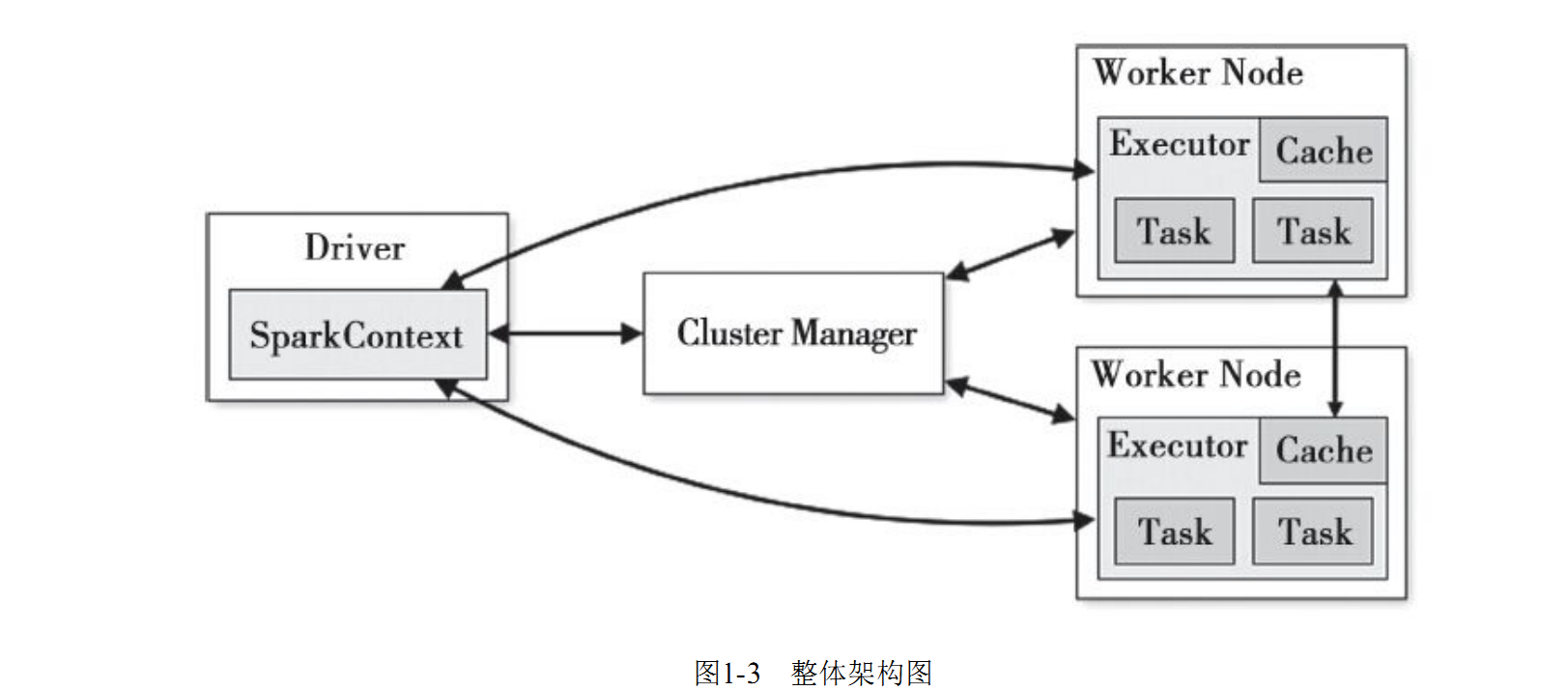
使用多线程池模型来减少task启动开稍，shuffle过程中避免不必要的sort操作以及减少磁盘IO操作

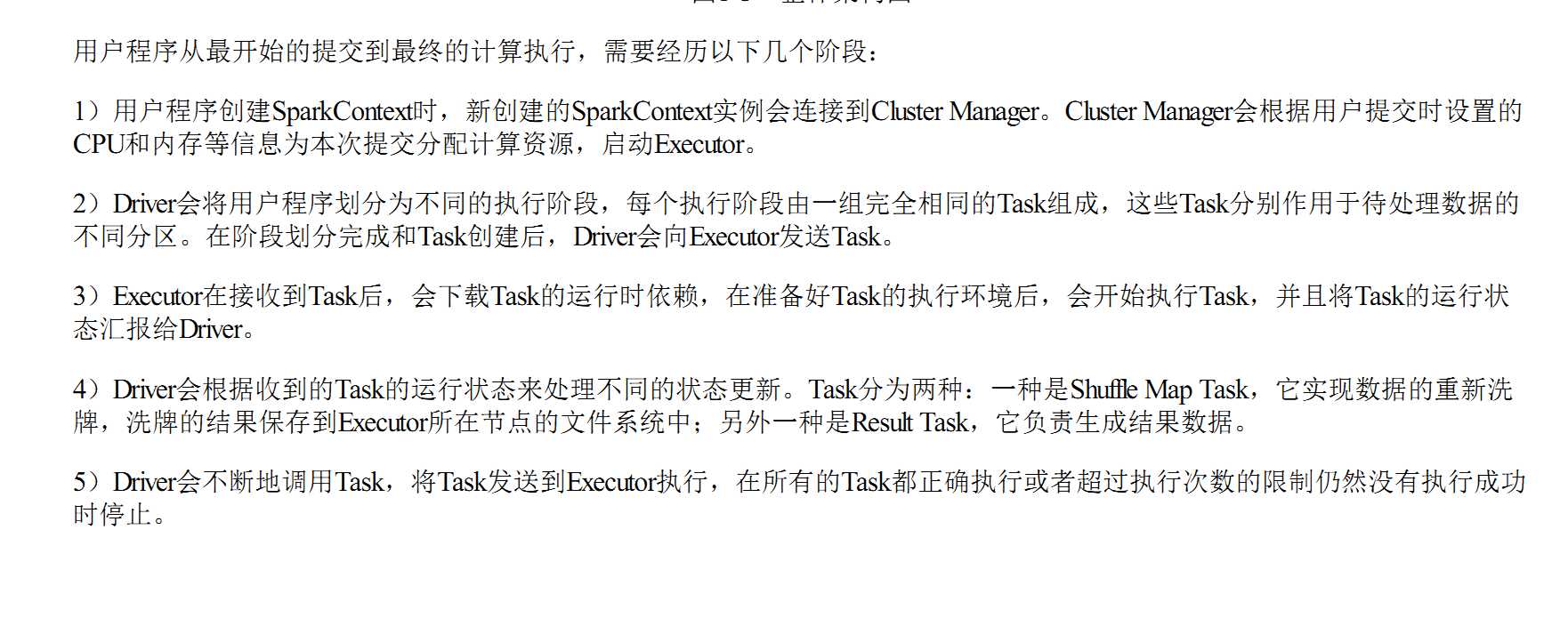
广泛的数据集操作类型

MapReduce由于其设计上的约束只适合处理离线计算，在实时查询和迭代计算上仍有较大的不足，而随着业务的发展，业界对实时查询和迭代分析有更多的需求，单纯依靠MapReduce框架已经不能满足业务的需求了。Spark由于其可伸缩、基于内存计算等特点，且可以直接读写**[Hadoop](http://lib.csdn.net/base/hadoop" \t "_blank" \o "Hadoop知识库)**上任何格式的数据，成为满足业务需求的最佳候选者。

Spark运行框架如下图所示，首先有集群资源管理服务(Cluster Manager)和运行作业任务的结点(Worker Node)，然后就是每个应用的任务控制结点Driver和每个机器节点上有具体任务的执行进程(Executor)。



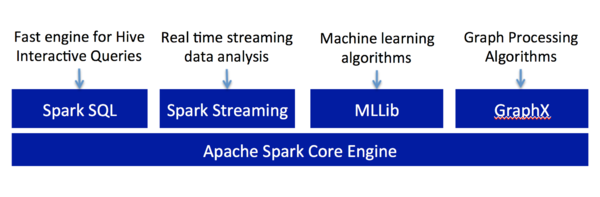




与MR计算框架相比，Executor有二个优点：一个是多线程来执行具体的任务，而不是像MR那样采用进程模型，减少了任务的启动开稍。二个是Executor上会有一个BlockManager存储模块，类似于KV系统(内存和磁盘共同作为存储设备)，当需要迭代多轮时，可以将中间过程的数据先放到这个存储系统上，下次需要时直接读该存储上数据，而不需要读写到hdfs等相关的文件系统里，或者在交互式查询场景下，事先将表Cache到该存储系统上，提高读写IO性能。另外Spark在做Shuffle时，在Groupby，Join等场景下去掉了不必要的Sort操作，相比于MapReduce只有Map和Reduce二种模式，Spark还提供了更加丰富全面的运算操作如filter,groupby,join等。

Spark的Application在运行时，首先在Driver程序中创建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 Streaming：构建在Spark上处理Stream数据的框架，基本的原理是将Stream数据分成小的时间片断（几秒），以类似batch批量处理的方式来处理这小部分数据。Spark Streaming构建在Spark上，一方面是因为Spark的低延迟执行引擎（100ms+），虽然比不上专门的流式数据处理软件，也可以用于实时计算，另一方面相比基于Record的其它处理框架（如[Storm](http://baike.baidu.com/view/1012011.htm" \t "_blank)），一部分窄依赖的RDD数据集可以从源数据重新计算达到容错处理目的。此外小批量处理的方式使得它可以同时兼容批量和实时数据处理的逻辑和算法。方便了一些需要历史数据和实时数据联合分析的特定应用场合。

## 2 业务场景使用

目前大数据在互联网公司主要应用在广告、报表、推荐系统等业务上。在广告业务方面需要大数据做应用分析、效果分析、定向优化等，在推荐系统方面则需要大数据优化相关排名、个性化推荐以及热点点击分析等。

这些应用场景的普遍特点是计算量大、效率要求高。Spark恰恰满足了这些要求，该项目一经推出便受到开源社区的广泛关注和好评。并在近两年内发展成为大数据处理领域最炙手可热的开源项目。

## 3 运行环境与开发环境搭建

配置修改:

============================ SetUp Spark=============================

Configuration

spark-env.sh

HADOOP\_CONF\_DIR=/opt/data02/hadoop-2.6.0-cdh5.4.0/etc/hadoop

JAVA\_HOME=/opt/modules/jdk

#######################################################

SPARK\_MASTER\_IP=192.168.56.101

SPARK\_MASTER\_PORT=7077

SPARK\_MASTER\_WEBUI\_PORT=8080

SPARK\_WORKER\_CORES=1

SPARK\_WORKER\_MEMORY=1000m

SPARK\_WORKER\_PORT=7078

SPARK\_WORKER\_WEBUI\_PORT=8081

SPARK\_WORKER\_INSTANCES=1

slaves

hadoop-spark.dragon.org

spark-defaults.conf

spark.master spark:// 192.168.56.101:7077

Start Spark

Start Master

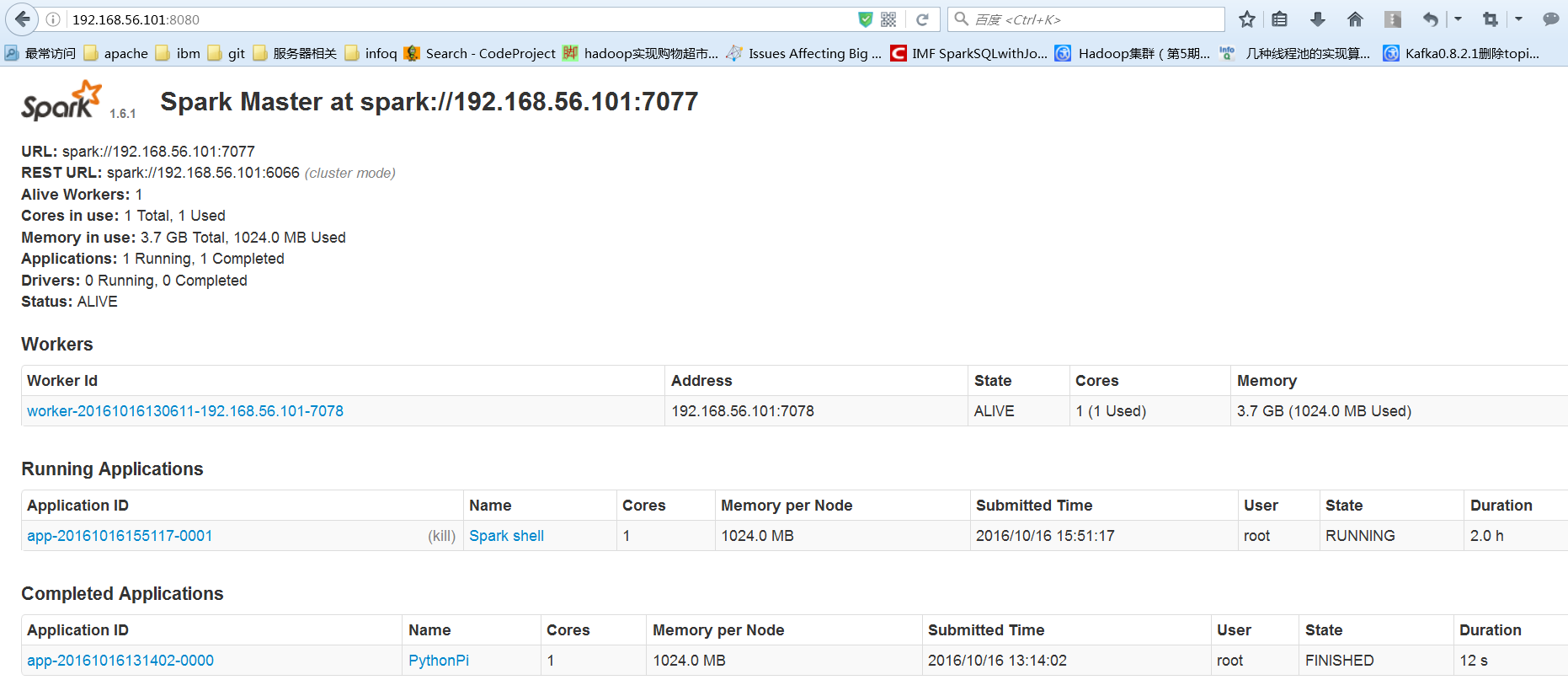
sbin/start-master.sh

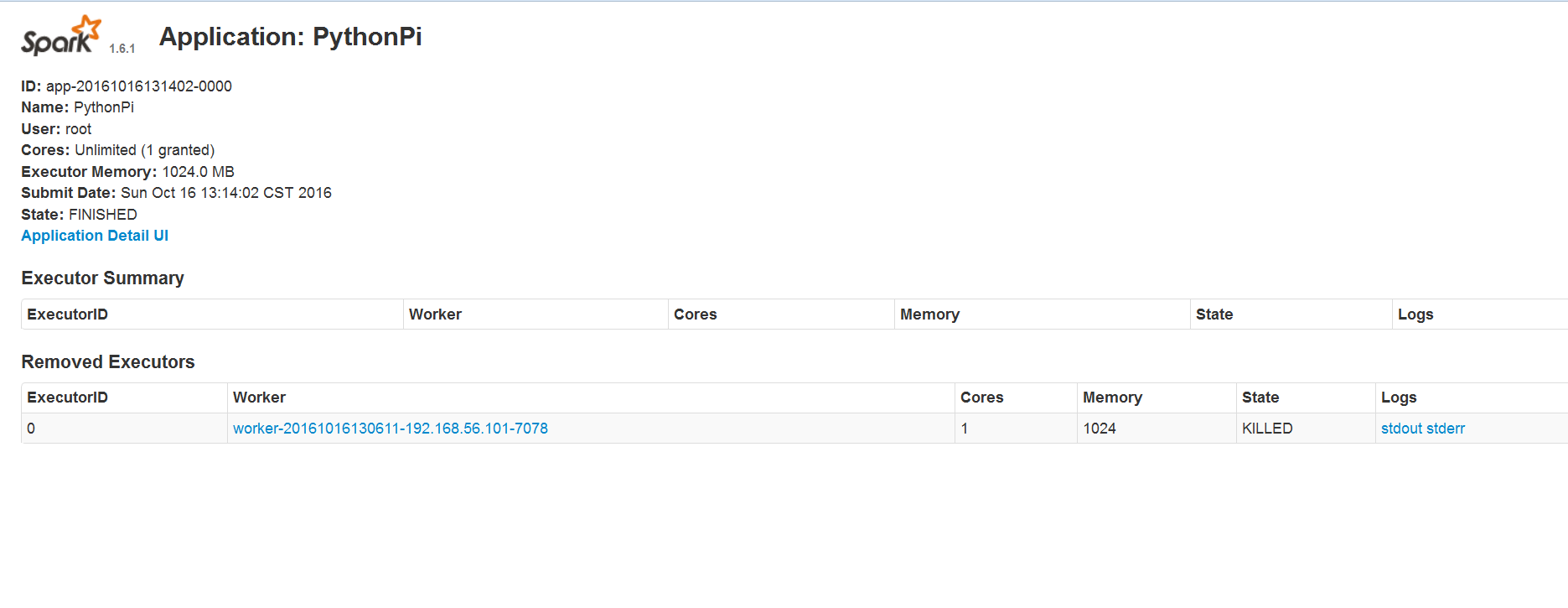
Start Slaves

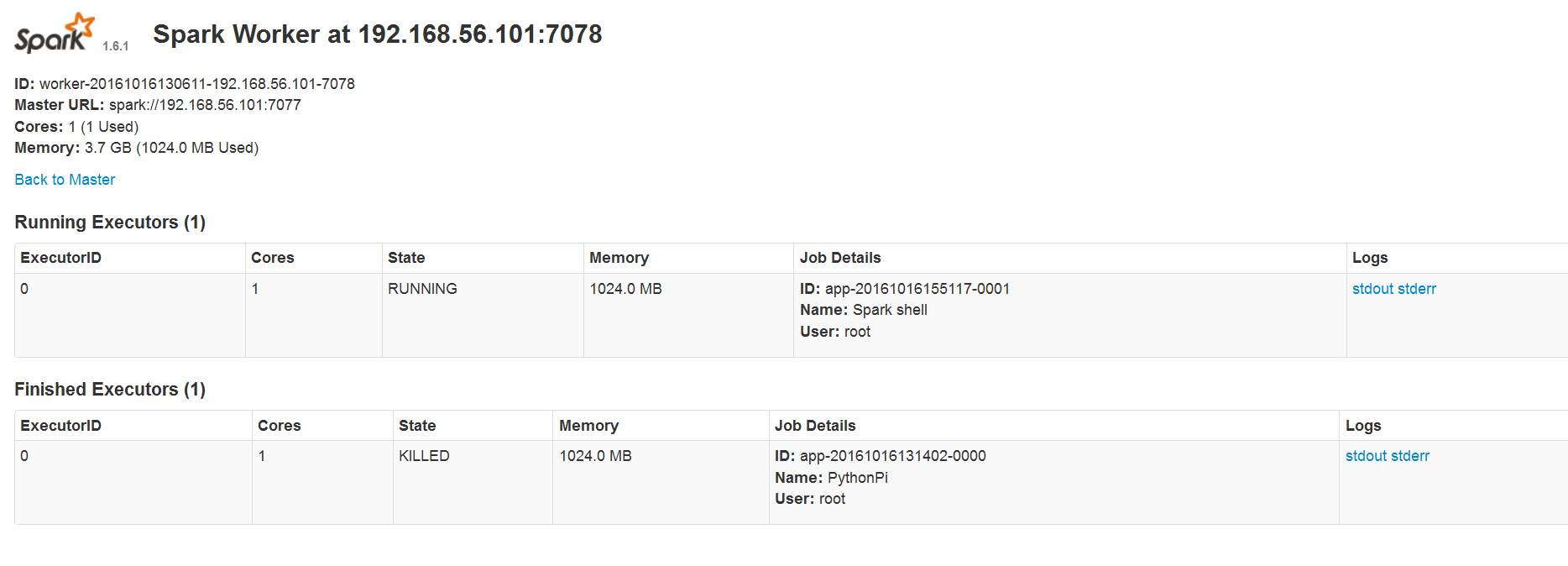
sbin/start-slaves.sh

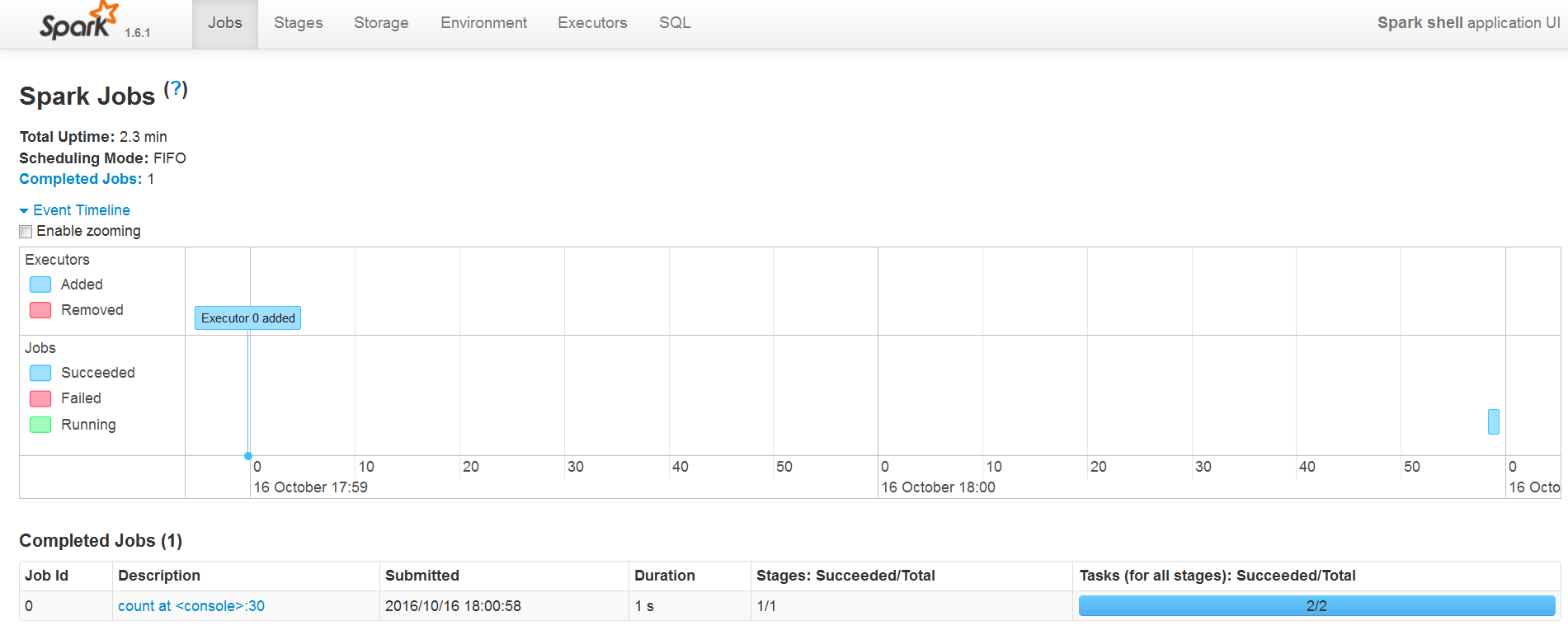
星环访问web查看：

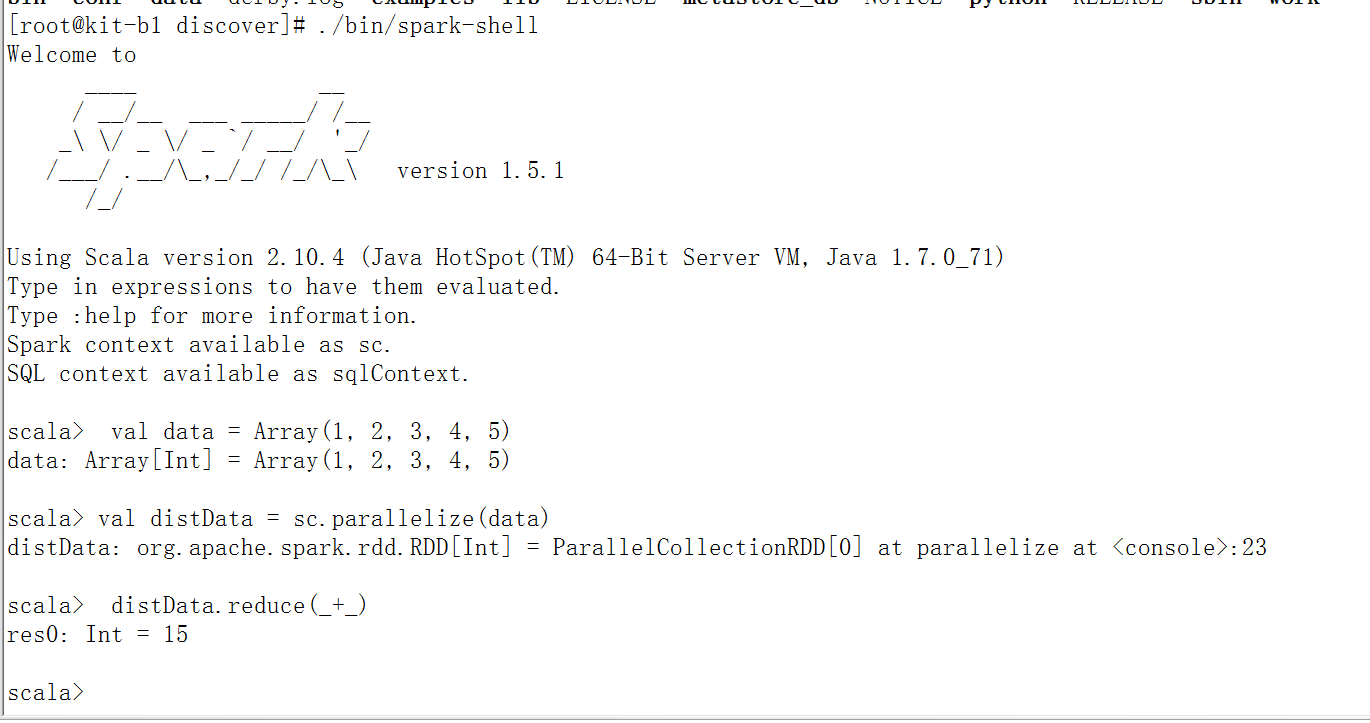
http://kit-b1:18080/











### 1 sparkshell 命令示列

#### 1.1 Spark shell 示列一：执行rdd

在终端执行spark：

##### 1.1.1 运行./spark-shell.sh

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

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

##### 1.1.3. 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）

##### 1.1.4. 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...

##### 1.1.5. 最后运行得到

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

res2: Int = 15

#### 1.2 spark shell示列二：执行程序后打印结果到指定文本

##### 1.2.1 run-example JavaWordCount ./wordcountdata.txt > Sparkpilog.txt //执行自带的计数程序 把计算结果打印到指定文件

##### 1.2.2 run-example SparkPi 1 > Sparkpilog.txt //启动一个线程 执行spark自带程序把pi值打印到指定文件

#### 1.3 spark shell 示列三：

##### 1.3.1 run-example SparkPi //执行spark自带程序 打印pi值

### 2 spark-submit 命令示列

### 服务器端部署与操作演示

172.16.19.151

提交任务：

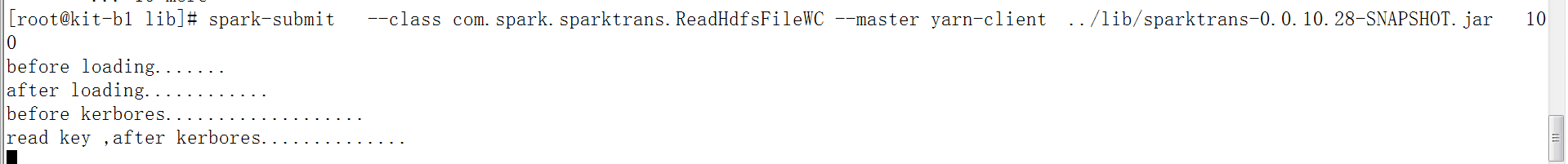
切换到/usr/lib/discover/lib/

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.spark.sparktrans.JavaSparkPiTest ../lib/sparktrans-0.0.10.28-SNAPSHOT.jar 100

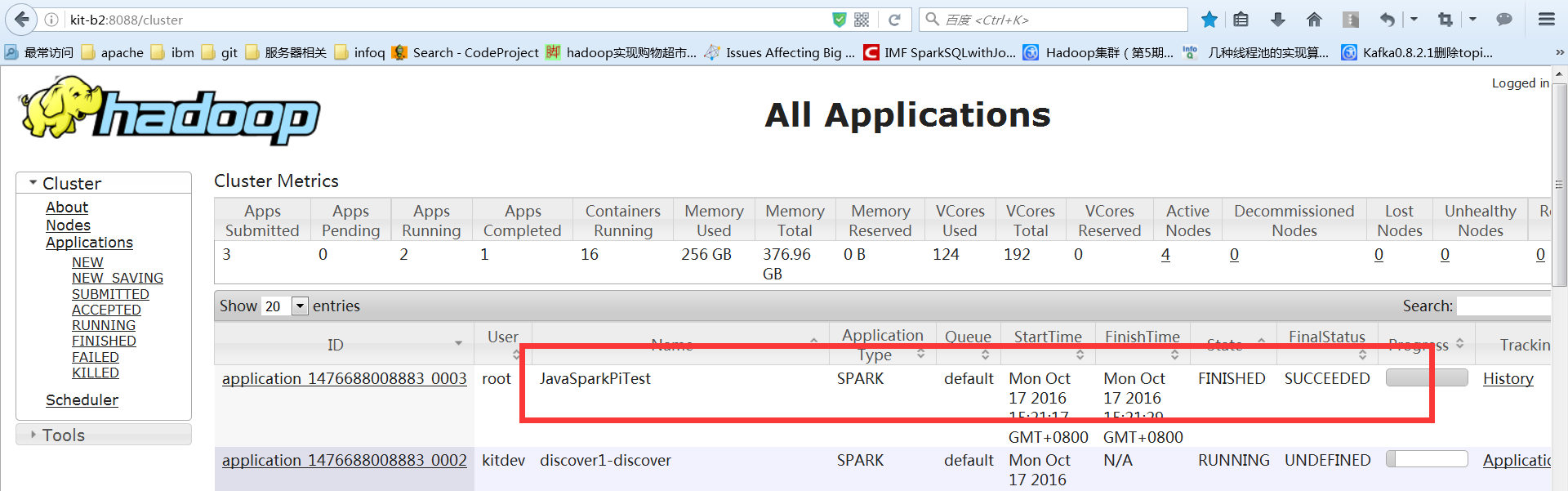
spark-submit --class com.spark.sparktrans.JavaSparkPiTest --master spark://kit-b1:7077 ../lib/sparktrans-0.0.10.28-SNAPSHOT.jar 100

spark-submit --class com.spark.sparktrans.JavaSparkPiTest --master yarn-client ../lib/sparktrans-0.0.10.28-SNAPSHOT.jar 100



spark-submit --class com.spark.sparktrans.ReadHdfsFileWC --master yarn-client ../lib/sparktrans-0.0.10.28-SNAPSHOT.jar 100

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

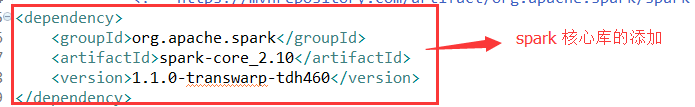


### 3 构建spark maven工程

#### 3.1 新建java工程



#### 3.2 引用相关jar包



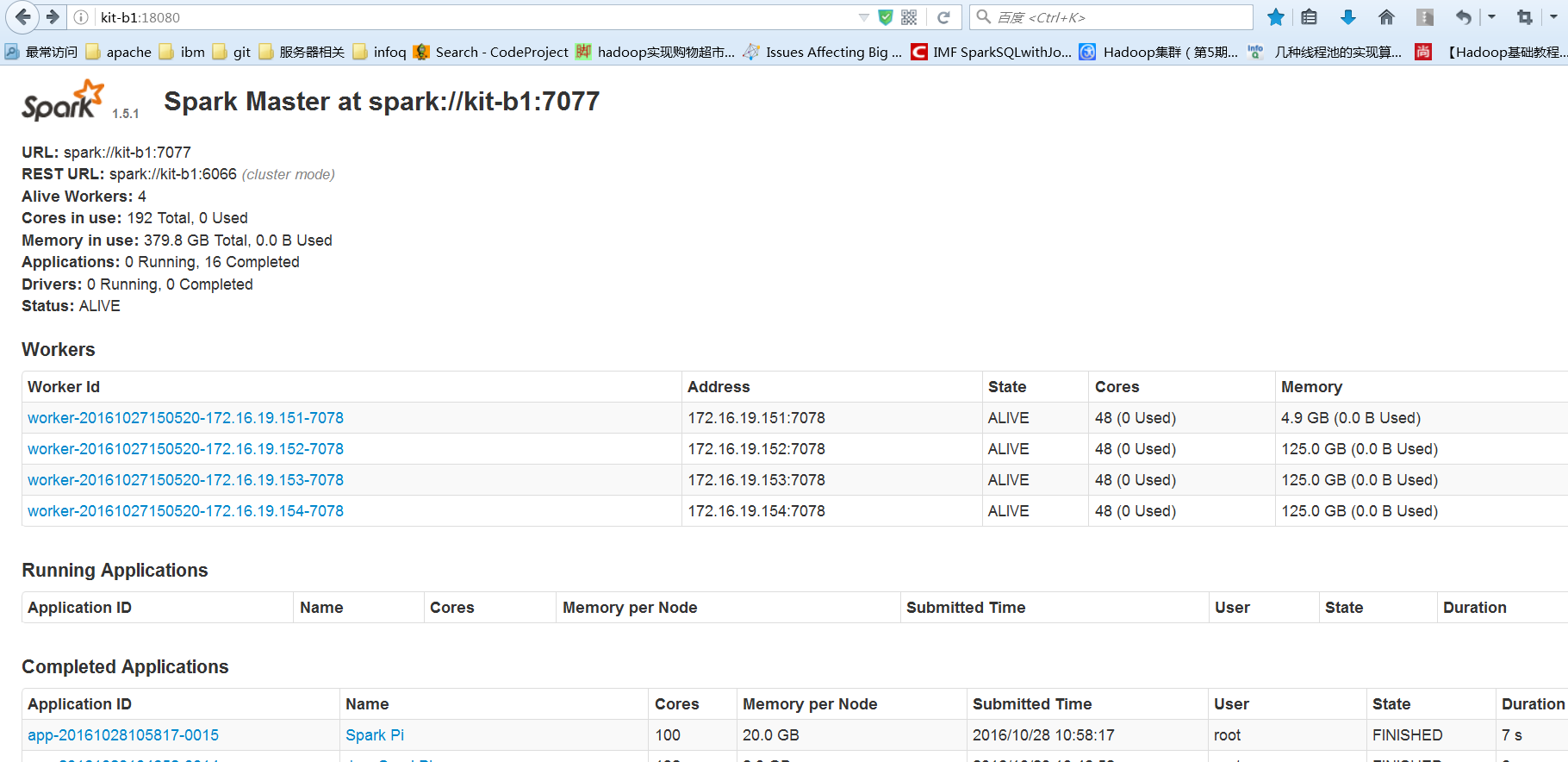
#### 3.3 编写spark程序

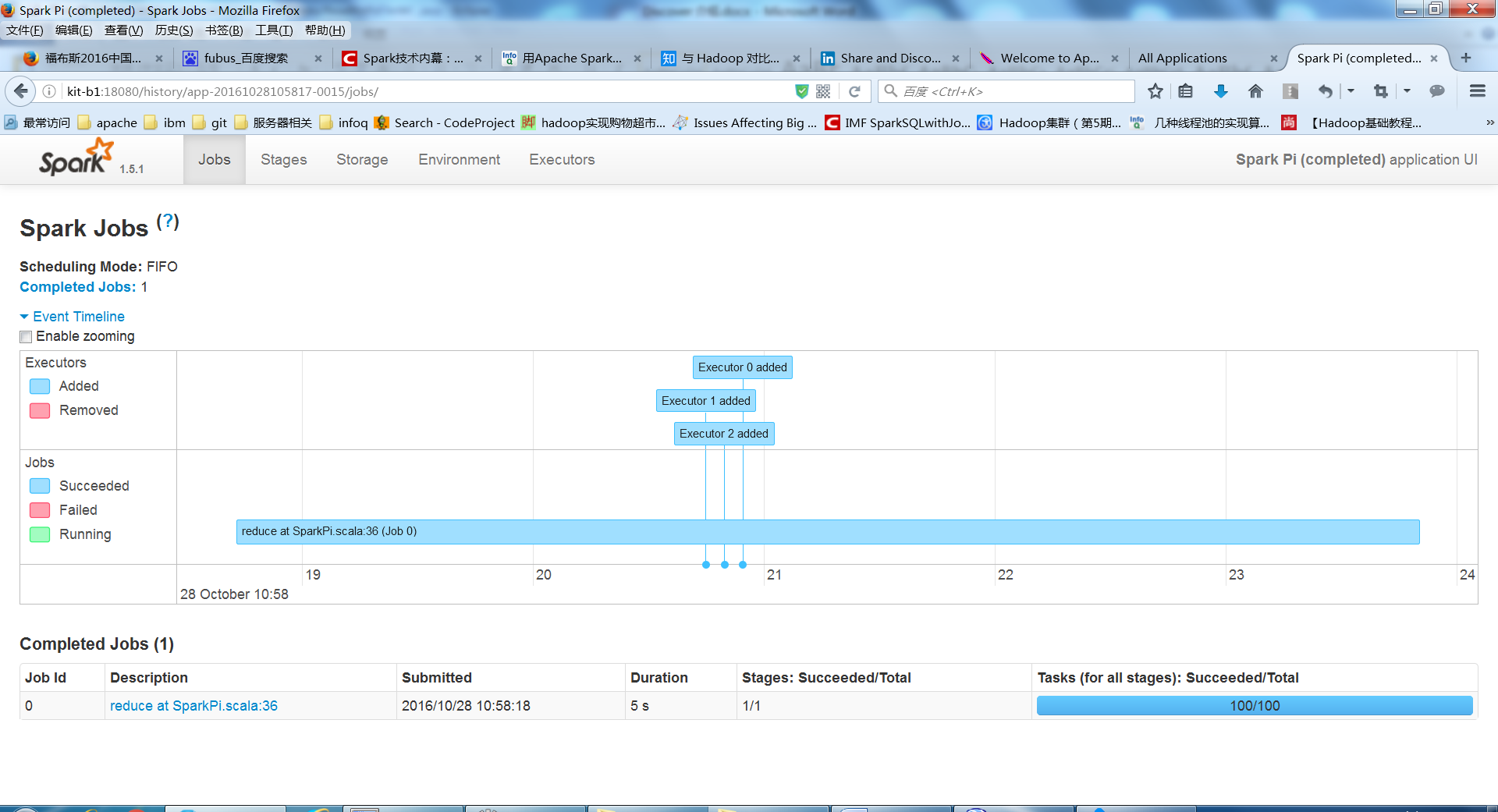
##### 3.3.1 打印pi值代码

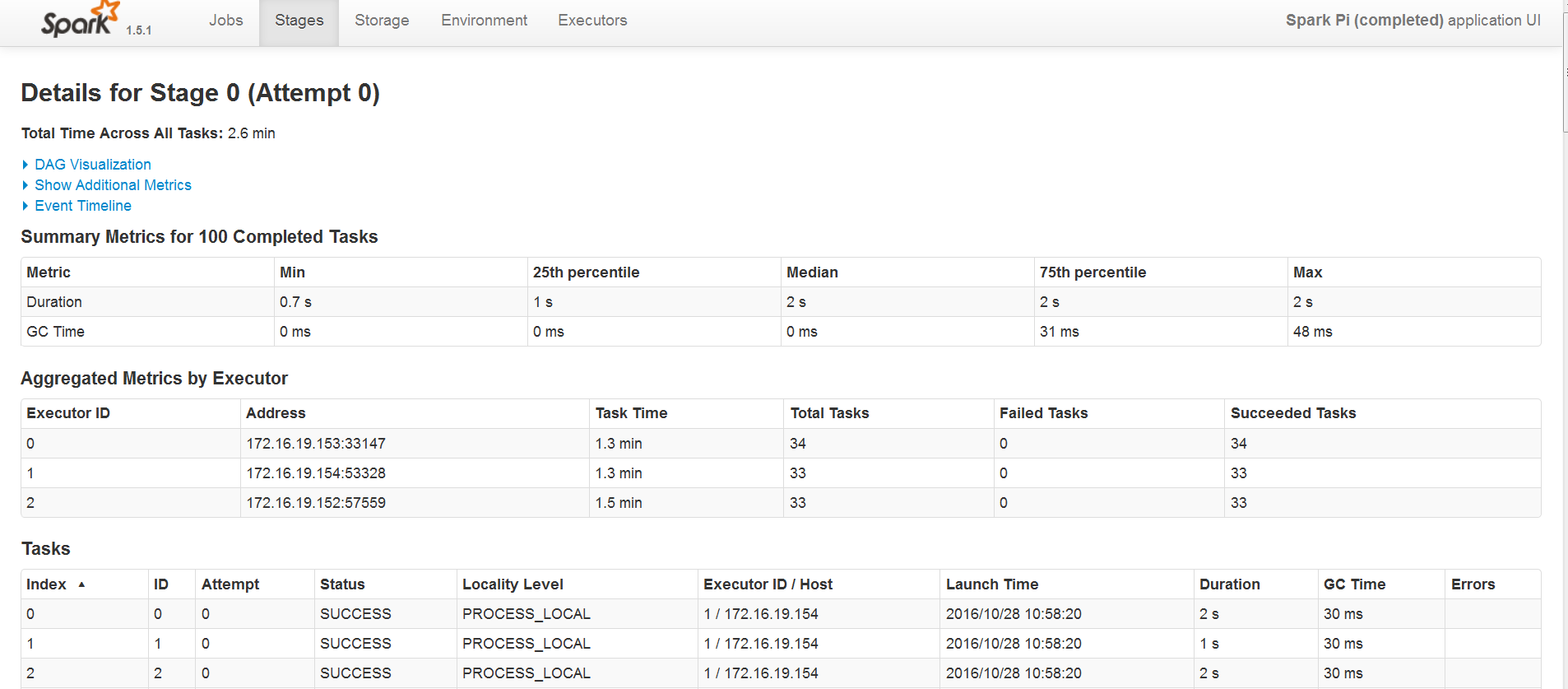


## 4 代码演示

## 5结果查看









tdh：

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.5.2-transwarp-tdh460.jar pi 10 100