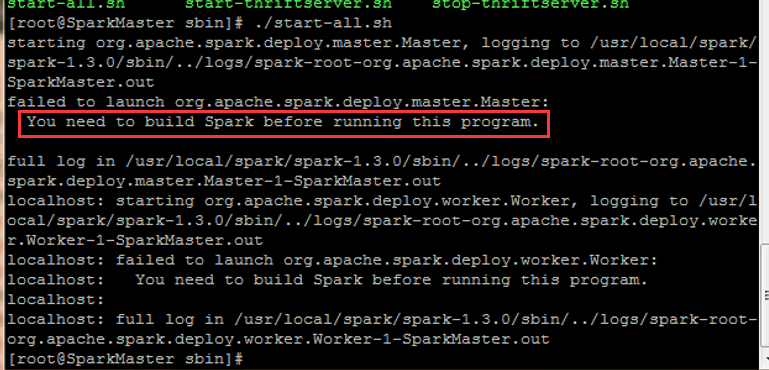
群问题及解决记录

说明：记录部分群中问题，给出可能的分析和解决方法（可能有其他解决方法）

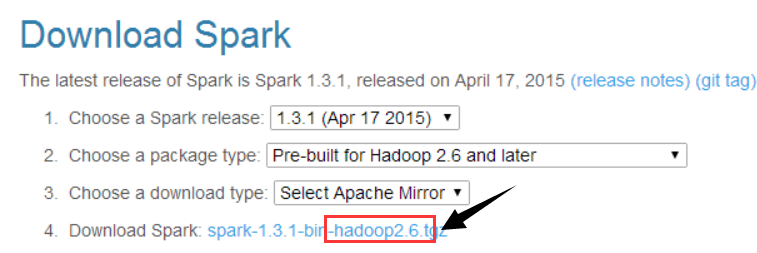
# 问题1，启动Spark失败

1. 错误信息



当出现红色标注的信息时。

1. 使用的不是编译后的版本，需要先build或直接从官网下载。



部署包正常情况下会带Hadoop版本

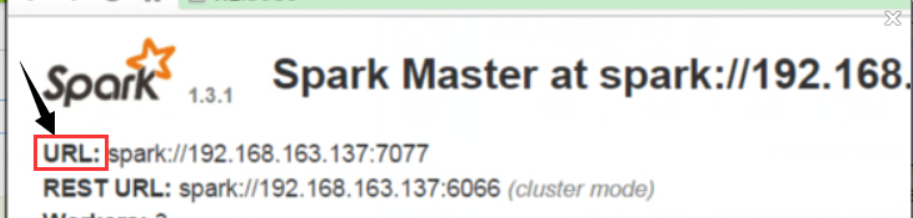
# 界面为什么不显示Applications信息

问题：我起了2个spark-shell 是不是应该有2个Applications？为啥界面一个都没有呢？

情况：直接运行 spark-shell，不加参数

分析：比如，以spark-shell 命令直接运行，此时，在spark的集群界面上是没有对应Applications信息的。

只有在命令中指定--master masterURL，其中masterURL对应集群的Masster URL地址，可以参考8080端口的web ui(默认），如下图，后面的就是当前的master url地址。（直接从界面把URL: 后面的内容拷贝过来）



直接从界面把URL: 后面的内容拷贝过来

# protobuf问题：

报错信息类似：

java.lang.VerifyError: class org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$AbandonBlockRequestProto overrides final method getUnknownFields.()Lcom/google/protobuf/UnknownFieldSet; (state=,code=0)  请问什么原因

原因分析：

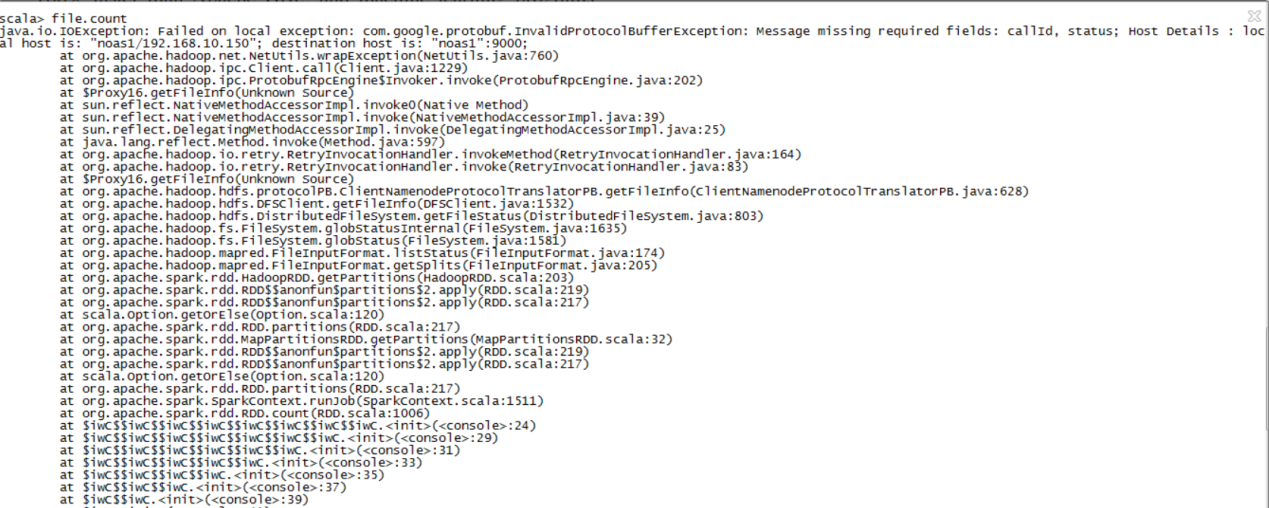
通信使用protobuf，protobuf有版本兼容性要求。

getUnknownFields : 由于通信两端的协议版本不一致，通信内容的字段不同，导致获取了不能识别的Fields而报错。

解决（理论上）：

保证通信两边的protobuf版本兼容。

类似错误，截图如下：



关键的异常信息：

同样应该是两边使用的Protobuf协议不一致造成。

这是spark访问Hadoop文件时的异常，重新用Hadoop client的版本构建spark即可。

# 其他

## reduceByKey 不能使用或编译报错：

1.2及之前版本 --需要倒入隐式转换

import org.apache.spark.SparkContext.\_

# RDD为什么不能嵌套使用

问题描述：为什么不能在其他RDD的任务里用到另一个R照着DD

分析：driver负责调度，action触发job提交，然后dag....；而sc是driver上的，task是在Executor上的，你在Executor上执行task时，不能使用sc。

参考：RDD transformations and actions can only be invoked by the driver, not inside of other transformations; for example, rdd1.map(x => rdd2.values.count() \* x) is invalid because the values transformation and count action cannot be performed inside of the rdd1.map transformation. For more information, see SPARK-5063

## OOM

1. 先确认OOM的位置：driver 侧还是 Executor侧（exe..时，报错处有TaskSet的具体信息）

Driver ：查看是否使用了collect等需要将数据集装载到deriver内存的的操作；增加更多内存；

Executor：增加更多内存；降低用于缓存的内存的比例（spark.storage.memoryFraction从0.6降低到更小值，把分配到缓存的内存用于执行task，会降低速度，但同时增加task的可用内存）。

# 读取textFile时，如何带上行号、偏移量

import org.apache.spark.rdd.\_

import org.apache.spark.SparkContext

import org.apache.hadoop.mapred.{FileInputFormat, InputFormat, JobConf, SequenceFileInputFormat,

TextInputFormat}

import org.apache.hadoop.io.{ArrayWritable, BooleanWritable, BytesWritable, DoubleWritable,

FloatWritable, IntWritable, LongWritable, NullWritable, Text, Writable}

// 自定义方法——参考SparkContext的textFile 实现

// RDD((line\_offset, line\_text))

def textFile(sc: SparkContext, path: String, minPartitions: Int = sc.defaultMinPartitions): RDD[(Long, String)] = {

//assertNotStopped()

sc.hadoopFile(path, classOf[TextInputFormat], classOf[LongWritable], classOf[Text],

minPartitions).map(pair => (pair.\_1.get, pair.\_2.toString)).setName(path)

}

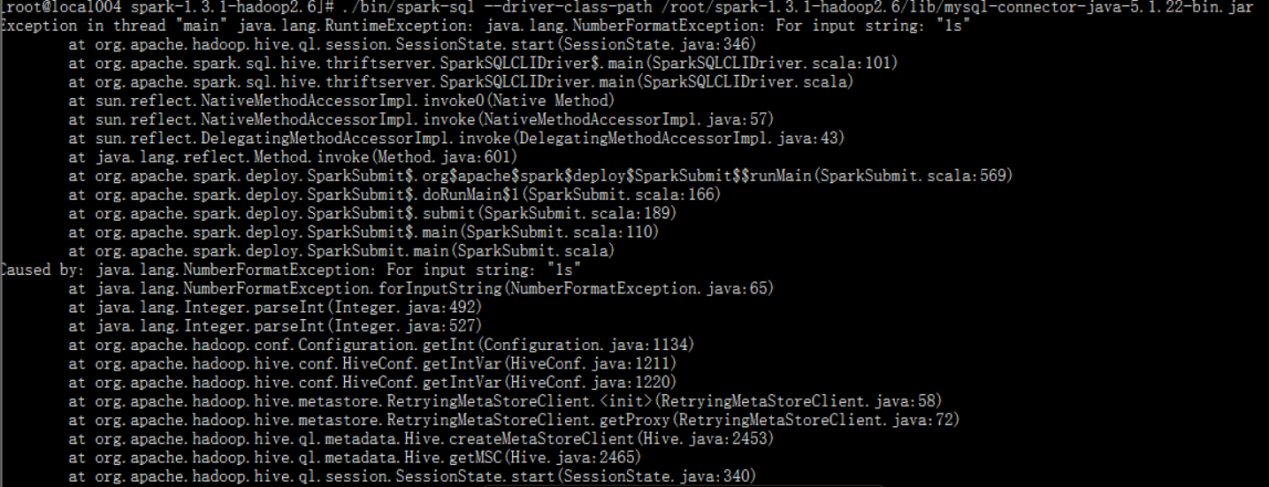
// RDD[(line\_number, line\_text)]

sc.textFile(...).zipWithIndex.map { case (x, y) => (y, x) }

🡺 sc.textFile(...).zipWithIndex.map (\_.swap)

# 访问hive时配置文件读取异常

错误截图：



异常原因：



通过查看异常堆栈，在HiveConf.getIntVar时，配置项的值为1s，导致数值格式化异常。

解决方法：

搜索配置文件中的1s值，改为1000

# 怎么只有本机的服务器被注册，其他节点的worker都没有使用



Standalone模式：Executor分配机制，基于core进行分配，但分配前提为内存够用。

Worker的core分配，根据配置有两种方式，根据Master.scala中的spreadOutApps控制下的两个分支——默认是集中分配cores。

所以在默认情况下，会从worker中取足够的cores，仅在该worker的cores不够时，才会继续从其他worker中获取cores，即在其他worker上启动Executor。

具体情况文件中的注释。

# 如何将RDD保存成一个文件

在RDD数据量比较小时，需要保存成一个文件的话，可以调用rdd的repartition(1)，将该rdd的分区数设置为1，然后saveAs….成文件。

分区数对应task数，一个分区一个task，也就只在一个节点上生成一个文件了。

# Hive:科学计数法表示的字符串的转换

hive导出值太大，自动变成科学计数法；

处理方法：

cast((xxx) as DECIMAL(10,10))  可以通过这种方式直接修改

注意精度等 提供对应的版本 – 0.13 官网…….

参考：https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types

<https://issues.apache.org/jira/browse/HIVE-3976>

# CDH

## HUE任务监控界面

## 时间显示异常问题

1. 查看服务器时区

登录服务器，date命令查看时间，如：

[~]$ date

Wed Aug 26 17:46:16 CST 2015

获取时区为CST

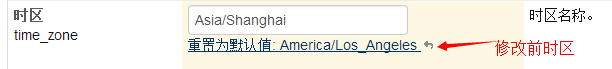
1. 查看hue时区配置

登录hue服务的配置界面





查看时区相关配置，如下：



将默认的时区修改为当前实际的时区

1. 重启hue服务，之后时间就正常了

# Driver修改的值不能传递到Executor的问题

比如以下代码：

|  |
| --- |
| import org.apache.spark.\_ import SparkContext.\_ import rdd.\_  object SparkTest {   def main(args: Array[String]) {     val sc = new SparkContext("spark://master:7077", "SparkTest") //    val sc = new SparkContext("local[2]", "SparkTest")      Temp.temp = 100          sc.parallelize(List(1, 2), 2).foreach(elem => println(Temp.temp))          sc.stop()   } }  object Temp {   var temp = 0L } |

分析：不能传递时，首先考虑序列化的问题

原因：

一、object字段对应java中的静态字段

二、序列化是将对象状态转换为可保持或传输的格式的过程，它也是RMI用来在不同JVM之间传递对象的机制，或者通过方法的参数或者作为一个方法调用的返回值。但有三个例外序列化不需要读或者写入到流：

1. 序列化会忽略静态字段，因为他们不属于对象的任何状态。

2. 基类的字段只有基类本身是序列化的时候才能被序列化。

3. 瞬间字段(transient)

解决方法：使用class（继承序列化接口）的实例