Apache Spark Learning

01课时 Spark简介

Spark是类Hadoop MapReduce的通用并行框架

中间输出结果保存在内存中

适合于数据挖掘和机器学习需要迭代的MapReduce的算法

启用了内存分布数据集，优化迭代工作负载

Spark是在Scala语言实现的

Spark和Scala能够紧密继承，像操作本地集合对象一样操作分部署数据集

Spark 是对Hadoop的补充，可以在Hadoop文件系统中并行运行

Spark目标：

使用在分布式数据上使用“变换”，轻松操纵大规模数据

传统的分布式计算平台扩展性好，但是受限于API(MapReduce)

Spark突破了单机的限制

有快速Data API，编写大规模数据处理程序很轻松

Spark 、Scala、Python

Spark最初使用的是Scala语言编写的，支持简介语法和交互式使用

添加了JavaAPI是为了支持独立的应用程序

Python的添加时为了交互式的Shell

02课时 Spark 预编译版本下安装与启动

1.下载安装

有两种安装模式：预编译模式和源代码安装模式，我们这里使用预编译模式的安装。

1）解压文件夹到 /usr/cloud 创建软链接 到spark

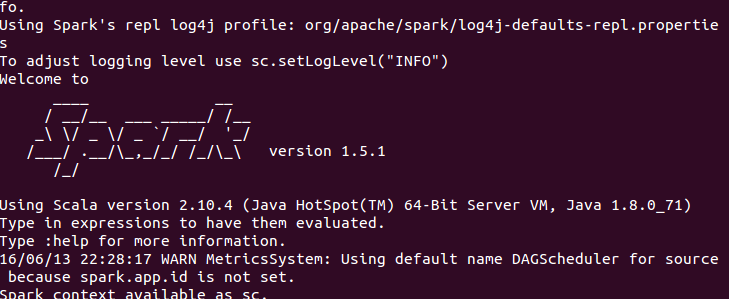
2）配置环境变量到/etc/profile,重新启动，或者执行source

export SPARK\_HOME=/usr/cloud/spark

export PATH=$PATH:$SPARK\_HOME/bin

2.本地启动模式

spark-shell --master local[num] # num是线程的数量，根据实际的CPU内核情况和实际的业务需求进行启动



03课时 Spark体验Shell 操作与WC操作，RDD介绍

RDD 弹性数据集

Spark内置对象 sc （SparkContext）

val file = sc.textFile(“path”)

file.count

file.first

file.filter(l=>l.contains(“info”))

file.filter(l=>l.contains("INFO")).count

file.map(l=>l.split(" ").size).reduce((a,b)=>if(a>b) a else b)

file.map(l=>l.split(" ").size).reduce((a,b)=>Math.max(a,b))

Spark RDD(Resilient Distributed DataSet ):

弹性分布式数据集，Spark的核心抽象。只读的对象集合，按照集群主机进行分区。

多个RDD作为input进行加载并进行一些列变换转变成新的RDD。

弹性是指Spark可以根据数据计算的来源方式，通过重新计算后进行自动重构，恢复丢失的分区。

加载数据或者执行变换并不会触发数据的处理，只是生成执行计划，直到action动作执行的时候，才进行数据处理，比如foreach。

04课时 Spark通过Maven进行源码编译安装

我们需要使用特定版本的Hadoop与Spark进行整合，因此需要通过源码编译的方式进行安装Spark。

安装maven

1）下载maven

2）加压到指定的文件夹

tar –zxvf apache-maven\*.tar.gz –C /usr/develop

3）配置环境变量PATH /etc/profile M2\_HOME PATH

maven -version

4）更改Maven仓库地址为国内OSChina M2\_HOME/conf/settings.xml

<mirror>

<id>repo2</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo2.maven.org/maven2/</url>

</mirror>

源代码安装Spark

1. 下载源代码

http://www.apache.org/dyn/closer.lua/spark/spark-1.6.1/spark-1.6.1.tgz

2. 加压文件

tar -zxvf spark.tgz

3. 使用maven编译源代码

mvn -Dhadoop.version=2.6.4 -Phadoop-2.6 -DskipTests clean package

出现内存不够的情况

export MAVEN\_OPTS=”-Xms256m –Xmx1024m –XX:MaxPermSize=128m”

05课时 Spark搭建Maven本地仓库服务器

maven 从服务器中下载依赖关系包，到本地的~/.m2/repository，在进行编译，但是如果我们在局域网中需要在多个主机上进行编译项目，为了加快速度，可以从采用搭建本地的Maven仓库服务器，使用已经下载的局域网资源，减少网络带宽。

局域网Maven服务器可以自己进行选择国内和国外服务器。

1)需要的软件

Nexus.war

JDK

Tomcat6+

2)install

复制到tomcat/webapps

修改tomcat的端口${tomcat\_home}/conf/

启动tomcat ${tomcat\_home}/bin/startup.sh JPS bootstrap进程

3） 主目录下有sonatype-work/nexus

这样修改maven的配置文件

06课时 Spark通过本地Maven仓库服务器进行编译以及进行描述文件的设置

略

07课时 Spark官方DOC的宏观描述和优势说明

Spark lightning-fast cluster computing

Apache Spark is a fast and general engine for large-scale data processing.

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk

Write application quickly in java Scala Python: Spark offers over 80 high-leveloperators that make it easy to build parallel apps. And you can use it interactively from the scala, python and R shell.

Generality: Combine SQL, Streaming and Complex analytics. Spark powrers a stack of libraries including SQL and DataFrames, Mllib for machine learning, Graphx and Spark Streaming. You can combine these libraries seamlessly in the same application.

Run Every: Spark runs on hadoop, Mesos, standalone,or in the cloud. It can access diverse data sources including HDFS, Cassandra, Hbase, and S3.

08课时 Spark独立集群模式部署和启动

在每一台机器上面安装好Spark

可以使用我们自己编译的spark，也可以使用下载的已经编译好的spark，我们自己编译的核心包是spark-core\*.jar,我们自己编译完成的jar包在为一个文件夹下面的target包下。

1. Standalone mode

我们之前安装仅仅解压到本地，我们使用spark-shell --master local[4]

使用jsp查看java进程：

Spark-shell ===> SparkSubmit ===> 4040 Web Werver

我们可以查看WebUI查看Spark 任务

2.独立集群模式

独立集群模式实际上就是手动通过脚本一一启动所需要的进程，如Master进程、Worker进程等等。在${SPARK\_HOME}/sbin目录中提供每一个进程启动的脚本：

start-master.sh stop-master.sh

start-slave.sh stop-slave.sh

start-slaves.sh stop-slaves.sh

start-all.sh stop-all.sh 启动所有的进程，在Master主机上执行,此脚本因为在hadoop中存在一个start-all.sh脚本，因此我们需要注意使用的时候，使用的是这个脚本。最好是加上文件的路径执行脚本。

1）配置conf/slaves

hadoop-yarn

hadoop-yarn-170

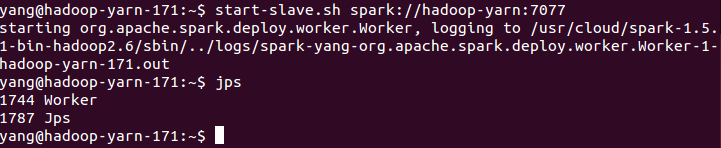
hadoop-yarn-171

2)）同步三台主机的配置文件

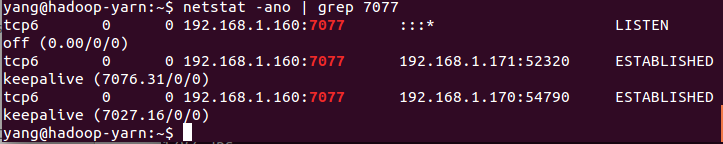
3）在master节点启动master进程 ./sbin/start-master.sh ,在那一台服务器上启动master，那一台就是master。

4）分别到slaves节点启动start-slave

start-slave.sh spark://hadoop-yarn:7077# 启动work进程



7077端口通信



一步启动可以在主节点使用start-slaves.sh

可能在启动slaves的时候，master节点上启动Worker进程就会出现的JAVA\_HOME没有设置的错误，从而导致无法启动的错误，我们可以在spark\_home/conf/spark-env.sh中设置该环境变量.

关于Spark的Web 端口,在master上面使用的是8080,在客户端使用的是8081端口,可以通过Web查看Spark集群的状况.

09课时 Spark Master和Woker的webui查看以及FIFO作业调度讲解

Spark Master Web 8080

Spark Worker Web 8081

命令行进入Spark： spark-shell --master spark://<hostname>:7077

默认情况下，对于Spark集群，我们从不同的命令行进入的话，会按照FIFO的顺序进行执行，如果第一个命令行不退出的话，第二个程序端口进入的任务永远不会被执行。

10课时 Spark 独立集群模式端口修改

在修改spark-env.sh文件中，修改配置文件，

SPARK\_MASTER\_PORT #7077

SPARK\_MASTER\_WEBUI\_PORT #8080

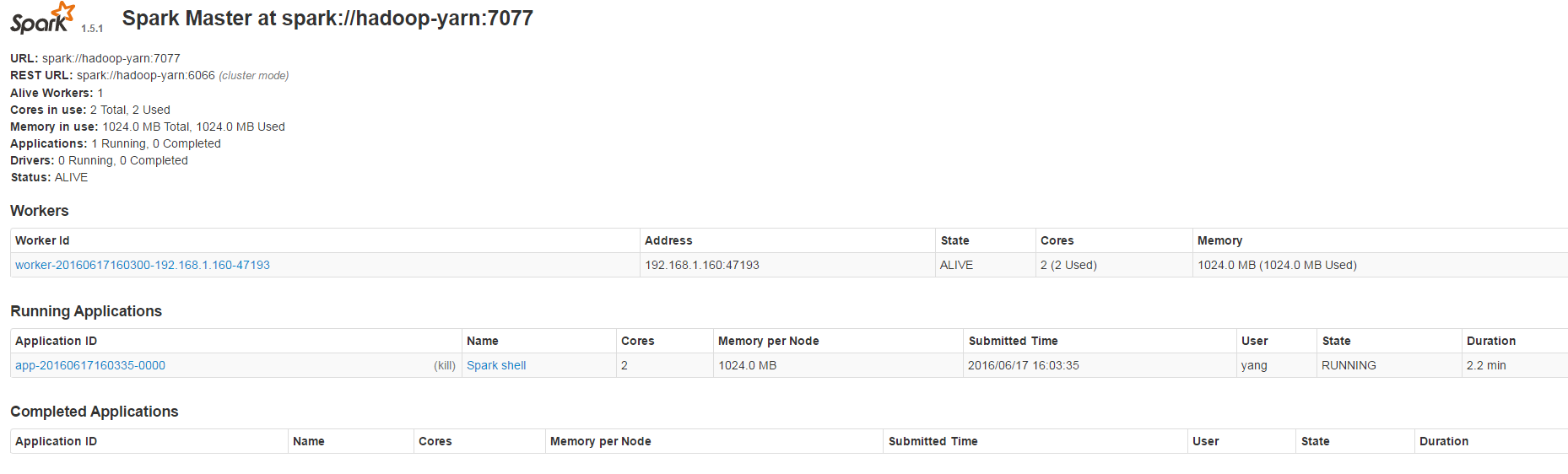
SPARK\_WORKER\_PORT

SPARK\_WORKER\_WEBUI\_PORT #to use non-default ports for the worker 8081

11课时 Spark Shell应用于addFile

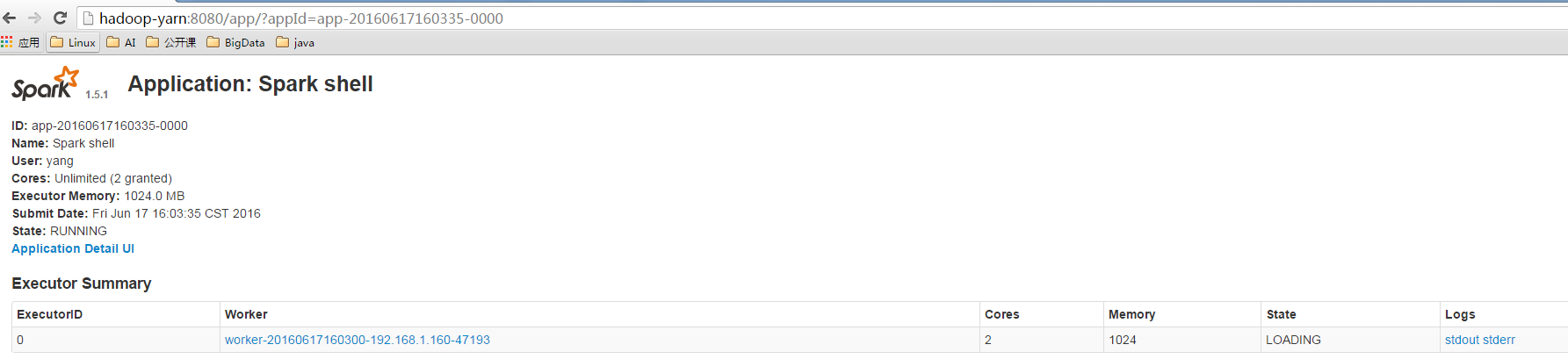
spark-shell --master spark://hadoop-yarn:7077

spark cluster : master(1) + worker(n)



Application(1) ====> executor(n)

Executor(1) =======> worker(1)



App(1) ------job(n): 每一个action的执行对应一个job。

Job(1)-------task(n): 每一个job有多个task。

val file = sc.textFile(“stringPath”)# 所有节点的数据是一起加载的（如果是本地文件系统），如果内容不一样的话，但是文件名一样，这样就会出现加载数据的时候不一致。因此最好使用分布式文件系统，使用统一的名字空间，这样加载才能够按照我们的要求进行。

1）该命令将每行文本作为RDD元素加载到Spark集群中（执行计划，并没有触发）

2）连接到spark cluster,如果集群不是分布式文件系统，spark会在每一个worker上加载数据

3）通常会选择分布式文件系统，如HDFS文件系统

4）local模式下，从本地进行加载

12课时 Spark 程序build的宏观介绍

Spark编译的过程：

Scala：可以使用sbt（simple build tool）编译（不推荐使用该方式）

Java：maven编译比较流行

Scala：maven编译也支持

Eclipse：maven编译支持

13课时 通过交互模式初始化Maven项目

使用maven方式编译java编写的Spark应用

1）创建目录，存放项目:mkdir myspark

2）使用mvn初始化命令，初始化项目

mvn archetype:generate

-DarchetypeGroupId=org.apache.maven.archetypes

-DgroupId= com.yang.spark

-DartifactId=JavaWordCount

-Dfilter=org.apache.maven.archetypes:maven-archetype-quickstart 可以省略，后面提示

14课时 通过参数指定放置直接初始化Maven项目

mvn archetype:generate -DarchetypeDroupId=org.apache.maven.archetypes -DgroupId=com.yang.spark -DartifactId=MySpark -Dfilter=org.apache.maven.archetypes:maven-archetype-quickstart

15课时 准备JAVA源文件到自己的源码包中

在spark安装包中的example中有一个JavaWordCount文件复制到src中的目录中，虽然该源文件的package生命和实际的目录地址不一致，但是maven会根据源代码中的包生命生成target中对应目录下的字节码文件。

16课时 设置pom.xml文件以来插件以及与自由仓库目录树的对应关系

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.apache.spark</groupId>

<artifactId>spark-core\_2.10</artifactId>

<version>1.5.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

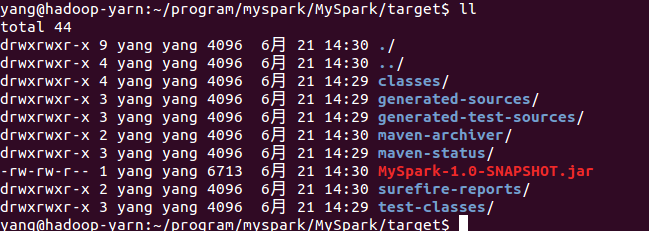
</build>

17课时 使用mvn package 指令进行编译并进行打包与自由仓库目录树的对应关系

cd ${pom.xml}

mvn package

就会生成target文件夹，下面就是各种字节码文件，最后在target中生成一个jar文件



18课时 使用java指令添加spark类库运行spark程序

java -cp MySpark\*.jar:/$SPARK\_HOME/lib/xxx.jar com.yang.spark.JavaWordCount ~/file

19课时 使用maven的exec执行插件运行java程序

<plugin>

**<groupId>org.codehaus.mojo</groupId>**

**<artifactId>exec-maven-plugin</artifactId>**

<version>1.2.1</version>

<configuration>

<mainClass>**com.yang.spark.JavaWordCount**</source>

<arguments>

<argument>**/home/yang/zookeeper.out**</argument>

</arguments>

</configuration>

</plugin>

</plugins>

</build>

mvn exec:java –Dexec:mainClass=”com.yang/spark.JavaWordCount” –Dexec:args=””

20课时 配置Eclipse的maven插件使用本地的maven仓库服务器

Eclipse Windows Maven user setting 修改setting.xml的目录，使用大陆的maven仓库

21课时 eclipse下maven项目的配置编译运行

1）创建maven项目（项目结构）

2）创建Java类

22课时 eclipse下Scala插件的安装卸载

Eclipse下安装Scala插件 marketplace 只安装第一项即可

23课时 eclipse下Scala程序运行

创建Scala项目

添加Scala Object

运行程序

**object** MyScala {

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

println("hello scala")

}

}

24课时 通过Scala命令手动编译Scala程序并且执行

scalac -d target/classes -classpath target/classes:/usr/cloud/spark/lib/xxx.jar src/main/java/com/yang/scala/MyScala.scala

生成对应的class文件

运行程序 java -classpath target/classes:/usr/cloud/spark/lib/xxx.jar MyScala

25课时 通过Maven命令编译运行Scala程序

Scala-maven编译插件: scala-maven-plugin

运行插件：exec-maven-plugin, 解决classpath问题

依赖关系scala-library version=2.10.4

Maven的使用需要网络依赖，暂时不学习

mvn scala:compile

mvn exec:java –Dexec.mainClass=”MyScala” –Dexec.args=”/home/yang/zookeeper.out”

Maven的使用需要网络依赖，暂时不学习

26课时 Eclipse下通过Maven-scala插件进行Spark开发

Maven 需要依赖网络，这里的pom.xml文件内容如下:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.yang</groupId>

<artifactId>mysaprk</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>mysaprk</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<build>

<plugins>

<plugin>

<groupId>net.alchim31.maven</groupId>

<artifactId>scala-maven-plugin</artifactId>

<version>3.2.1</version>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>1.2.1</version>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.apache.spark</groupId>

<artifactId>spark-core\_2.10</artifactId>

<version>1.5.1</version>

</dependency>

<dependency>

<groupId>org.scala-lang</groupId>

<artifactId>scala-library</artifactId>

<version>2.10.4</version>

</dependency>

</dependencies>

</project>

27课时 创建SparkContext简介与环境准备

1）新建maven项目

2）修改pom.xml 添加对于spark的依赖 spark-core\_2.10

3）添加scala的库依赖，调整scala的版本

28课时 使用Scala创建SparkContext对象以及conf配置和调试

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

**val** master = Properties.envOrElse("PATH", "spark://hadoop-yarn:7077")

println(master)

**val** conf = **new** SparkConf();

conf.setMaster("spark://hadoop-yarn:7077")

conf.setAppName("Spark test")

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

println("create spark context is over")

}

不要使用最新的Eclipse Mar，使用keper，同时Scala2.11与Spark1.5.1不兼容，所以需要使用Scala2.10版本

<http://download.scala-ide.org/sdk/helium/e38/scala210/stable/site>

不要直接使用MarketPlace中搜索出的Scala4.2，这个不能够正确的安装，需要在手动添加上面的网址。

29课时 使用java创建SparkContext对象以及元数据介绍

SparkConf conf = **new** SparkConf();

conf.setMaster("spark://hadoop-yarn:7077");

conf.setAppName("Zhangsan Java Spark");

JavaSparkContext sc = **new** JavaSparkContext(conf);

System.*out*.println("finished over");

30课时 使用Java创建RDD对象并在Eclipse中运行

数据的load与save

从集合中转化：

val rdd = sc.parallelize(List(1,2,3))

rdd.take(3)

JavaRDD<Integer> rdd = sc.parallilize(Arrays.asList());

31课时 使用Scala创建RDD对象并在Eclipse中调试运行

RDD:Resilient Distributed Dataset

数据来源：

Hadoop

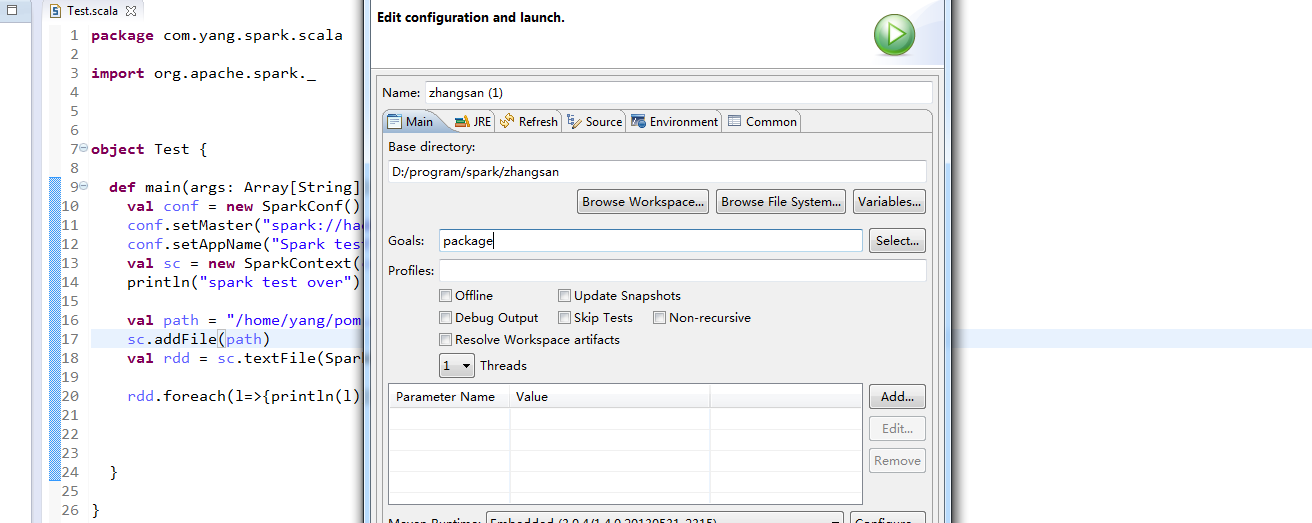
Scala Python Java等集合

文件系统

RDD数据计算时机：仅当数据被访问的时候，持久化persist()的时候才会去访问

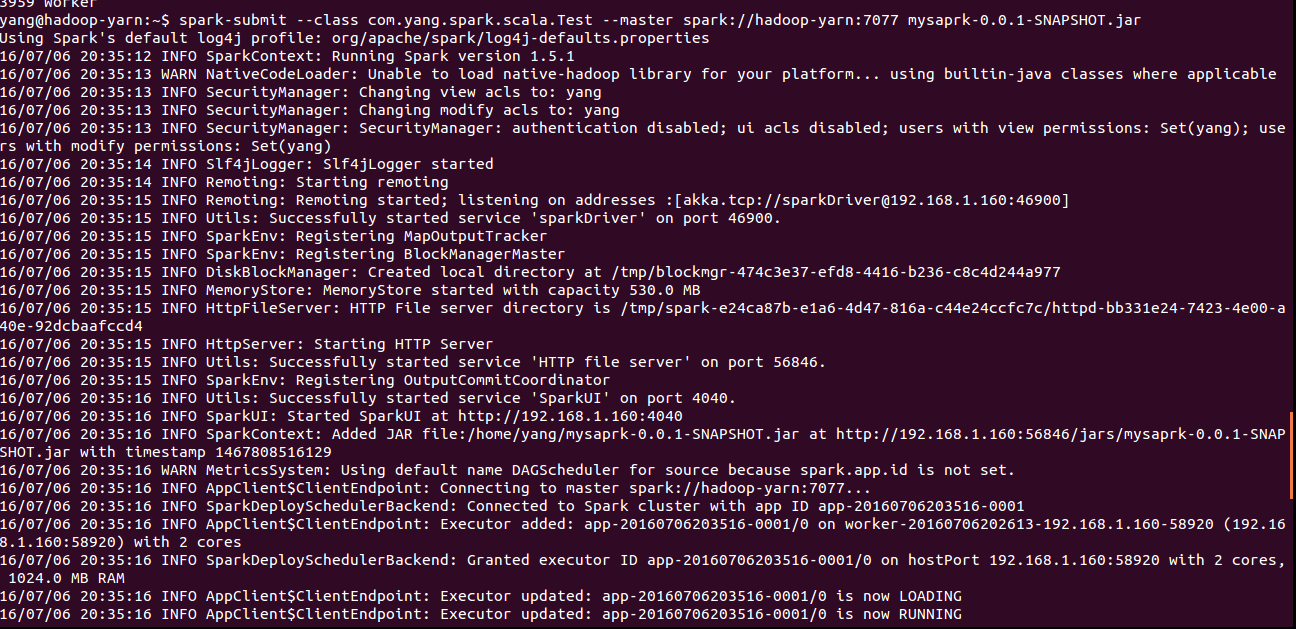
数据集重用： Cache策略

当我们编写程序的时候，涉及文件的操作，因为我们使用的是Spark集群，因此不能够直接使用Eclipse执行程序，不然就会出现乱七八糟的错误，比如ClassNotFound等等，找不出错误的原因，因为Spark集群是需要将所有的文件副本复制到每一个Worker几点，但是我们使用的方式直接将local本地文件传入到其中，在加载文件的时候就会出现无法将文件复制到所有的节点上面。而且我们使用的Eclipse是在Windows操作系统上面，通过交换机连接的Spark集群服务器。如果需要使用Spark集群进行处理的话，则需要将Maven项目打包成为一个package，



这样在target中就会生成一个jar 包，然后我们使用spark-submit提交任务到Spark集群中进行执行。

spark-submit --class com.yang.spark.scala.Test --master spark://hadoop-yarn:7077 mysaprk-0.0.1-SNAPSHOT.jar



32课时 addfile方法的使用和常见错误

addFile就是把文件保存在集群中的每一个节点上面（一个副本），这样便于分布式计算。

支持的文件类型。

**val** conf = **new** SparkConf()

conf.setMaster("spark://hadoop-yarn:7077")

conf.setAppName("Spark test")

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

println("spark test over")

**val** path = "/home/yang/pom.xml"

sc.addFile(path)

**val** rdd = sc.textFile(SparkFiles.get("pom.xml"))

rdd.foreach(l=>{println(l)})

33课时 flatMap函数处理cvs文件进行扁平化RDD处理

略

Reference：

1. 徐培成Apache Spark视频资源