[资料介绍 2](#_Toc17854)

[环境预准备 4](#_Toc31877)

[hadoop配置和操作 5](#_Toc22361)

[HDFS的基本操作 7](#_Toc22293)

[配置YARN 8](#_Toc15911)

[hive 9](#_Toc20819)

[spark 11](#_Toc3356)

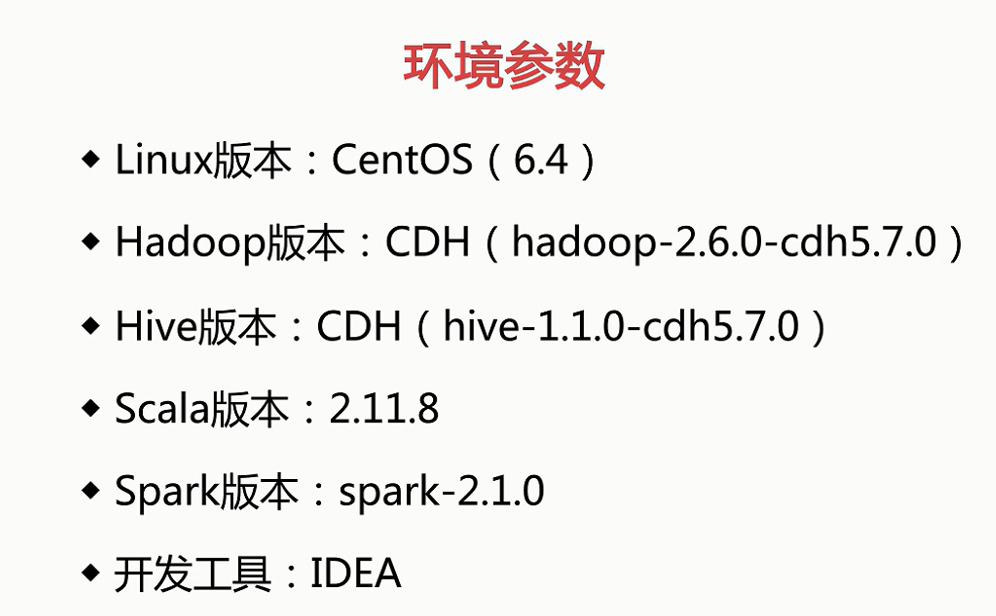
[分析日志 19](#_Toc20105)

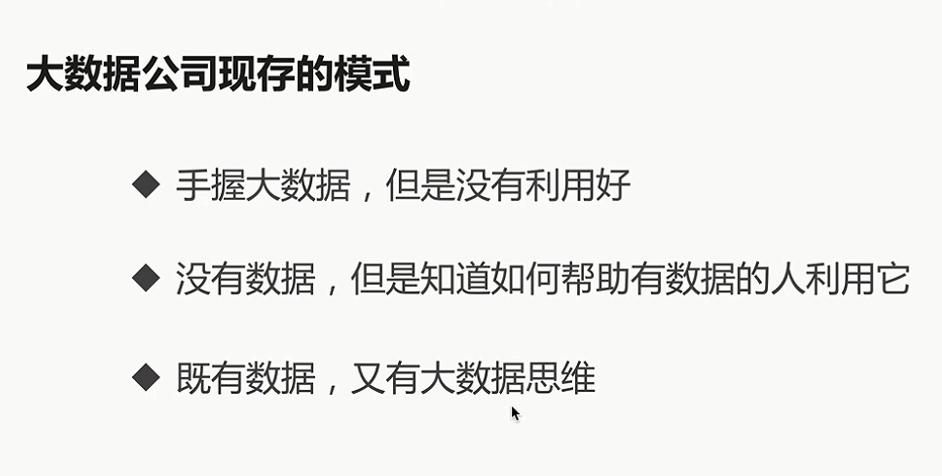
[windows下使用idea跑spark 21](#_Toc28121)

[一个hadoop的集群 22](#_Toc24399)

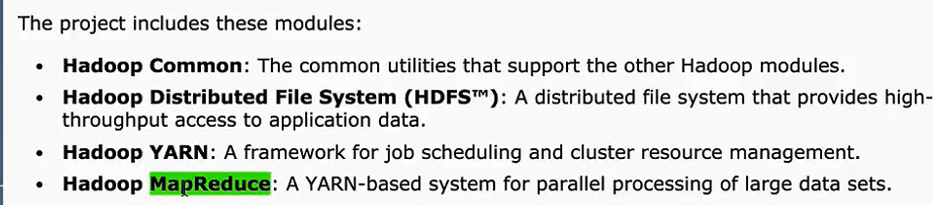
[常见问题总结 29](#_Toc28205)

# 资料介绍

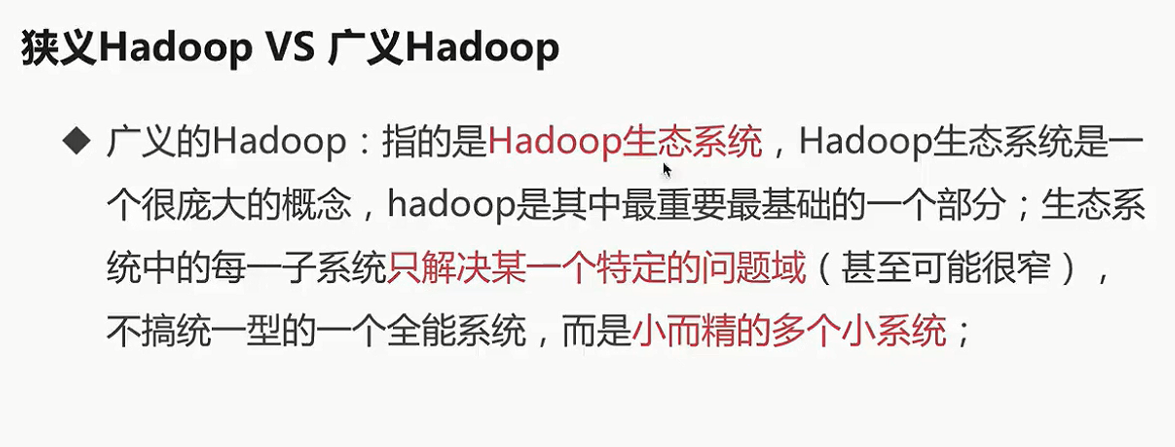












# 环境预准备

1. 首先新建一台虚拟机，设置的名字为hadoop001，用户名为hadoop001，密码为root
2. yum install lrzsz （安装上传插件）
3. 预先建好一些目录，再做解压

mkdir software

mkdir app

mkdir source

mkdir data

tar -zxf ~/software/hadoop-2.6.0-cdh5.7.0.tar.gz -C ~/app/

tar -zxf ~/software/hive-1.1.0-cdh5.7.0.tar.gz -C ~/app/

tar -zxf ~/software/jdk-8u45-linux-x64.tar.gz -C ~/app/

unzip ~/software/apache-maven-3.5.2-bin.zip -d ~/app/

tar -zxf ~/software/spark-2.1.0.tgz -C ~/app/

tar -zxf ~/software/spark-2.1.0-bin-2.6.0-cdh5.7.0.tgz -C ~/app/

4）在.bash\_profile环境中预先写入一些配置

vim ~/.bash\_profile

export JAVA\_HOME=/home/hadoop001/app/jdk1.8.0\_45

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

export HADOOP\_HOME=/home/hadoop001/app/hadoop-2.6.0-cdh5.7.0

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

export HIVE\_HOME=/home/hadoop001/app/hive-1.1.0-cdh5.7.0

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

export MAVEN\_HOME=/home/hadoop001/app/apache-maven-3.5.2

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

export SPARK\_HOME=/home/hadoop001/app/spark-2.1.0-bin-2.6.0-cdh5.7.0

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

应用生效

source ~/.bash\_profile

hadoop全局命令的方式

4）修改主机名

vim /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=hadoop001

如果出现问题那就这么解决

hadoop001 is not in the sudoers file. This incident will be reported.

<https://jingyan.baidu.com/article/2a1383284bb3e8074a134f2d.html>

5）设置ip和hostname的映射关系.

vim /etc/hosts

192.168.199.200 hadoop001

127.0.0.1 localhost

6）ssh免密码登陆

(本步骤可以省略，但是后面你重启hadoop进程时是需要手工输入密码才行)

ssh-keygen -t rsa

cp ~/.ssh/id\_rsa.pub ~/.ssh/authorized\_keys

7）修改你本机windows的映射

C:\Windows\System32\drivers\etc

8）关闭防火墙

sudo service iptables stop

chkconfig iptables off

# hadoop配置和操作

1）hadoop-env.sh

export JAVA\_HOME=/home/hadoop001/app/jdk1.8.0\_45

2）core-site.xml

<property>

<name>fs.defaultFS</name>

<value>hdfs://hadoop001:8020</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/home/hadoop001/app/tmp</value>

</property>

3）hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

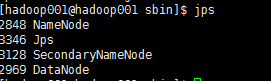
4）格式化HDFS

注意：这一步操作，只是在第一次时执行，每次如果都格式化的话，那么HDFS上的数据就会被清空

bin/hdfs namenode -format

5）启动HDFS

sbin/start-dfs.sh



如果出现以下问题

The authenticity of host 'hadoop001 (192.168.100.15)' can't be established.

<https://blog.csdn.net/w3045872817/article/details/70820949/>

ssh -o StrictHostKeyChecking=no hadoop001

网站访问

<http://hadoop001:50070/>

<http://hadoop001:50070/explorer.html#/>

停止HDFS

sbin/stop-dfs.sh

# HDFS的基本操作

1）浏览查看

hadoop fs -ls /

2）递归查看

hadoop fs -ls -R /

3）创建目录

hadoop fs -mkdir /test/

4）创建多层目录要加p

hadoop fs -mkdir -p /a/b

5）将一个文件上传上去

hadoop fs -put hdfs.cmd /test/

6）查看文件内容

hadoop fs -text /test/hdfs.cmd

7）下载文件

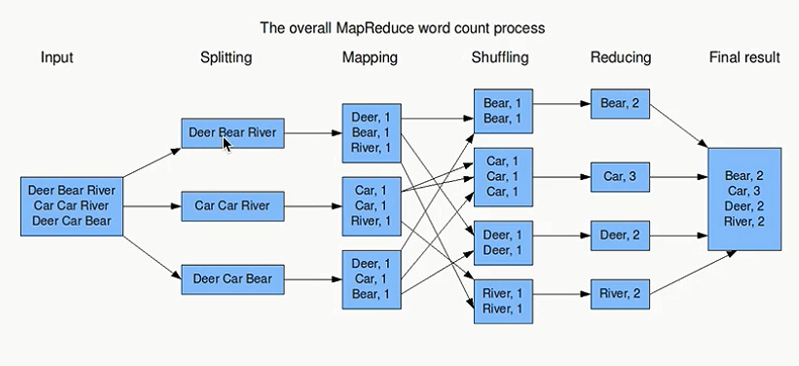
hadoop fs -get /test/hdfs.cmd a\_tmp

8）删除文件

hadoop fs -rm /test/hafs.cmd

9）删除文件夹

hadoop fs -rmr /test/hafs.cmd



# 配置YARN

1）mapred-site.xml

cp mapred-site.xml.template mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

2）yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

1. 启动yarn

sbin/start-yarn.sh

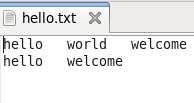
网站访问

<http://hadoop001:8088>

停止yarn： sbin/stop-yarn.sh

5）提交一个作业

先将需要统计的文件上传到hdfs上



创建一个目录

hadoop fs -mkdir -p /input/wc

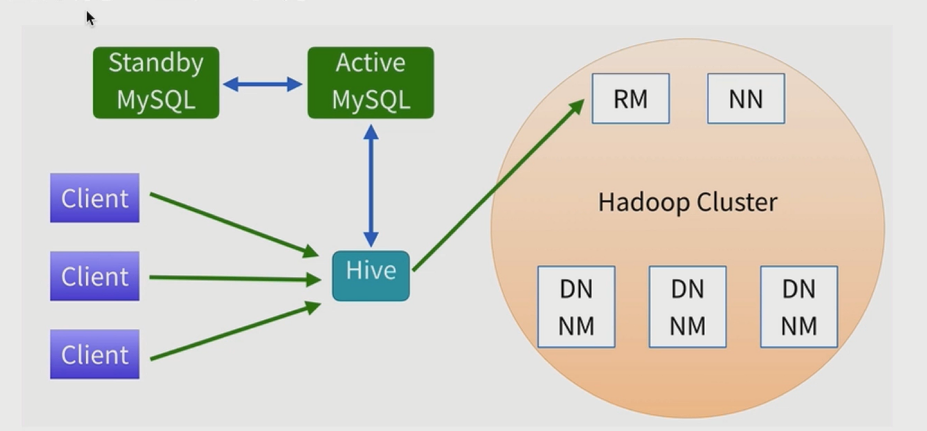
上传数据文件

hadoop fs -put hello.txt /input/wc/

执行mp任务

hadoop jar /home/hadoop001/app/hadoop-2.6.0-cdh5.7.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.7.0.jar wordcount /input/wc/hello.txt /output/wc/

# hive



1）先安装一个mysql

2）hive-env.sh

拷贝一份配置文件然后编辑

cp hive-env.sh.template hive-env.sh

HADOOP\_HOME=/home/hadoop001/app/hadoop-2.6.0-cdh5.7.0

(pwd命令可输出当前目录)

3）hive-site.xml

新建一个文件hive-site.xml

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost:3306/sparksql?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>root</value>

</property>

</configuration>

4）将mysql的jar包拷贝到hive的lib目录下

cp /home/hadoop001/software/mysql-connector-java-5.1.24-bin.jar /home/hadoop001/app/hive-1.1.0-cdh5.7.0/lib/

5）启动hive

./hive

6）第一个例子

create table hive\_wordcount(context string);

将数据加载到表里

load data local inpath '/home/hadoop001/data/hello.txt' into table hive\_wordcount;

select word, count(1) from hive\_wordcount lateral view explode(split(context,'\t')) wc as word group by word;

lateral view explode(): 是把每行记录按照指定分隔符进行拆解

7）再来一个例子吧

create table emp(

empno int,

ename string,

job string,

mgr int,

hiredate string,

sal double,

comm double,

deptno int

) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

create table dept(

deptno int,

dname string,

location string

) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

load data local inpath '/home/hadoop001/data/emp.txt' into table emp;

load data local inpath '/home/hadoop001/data/dept.txt' into table dept;

求每个部门的人数（要启动yarn哦）

select deptno, count(1) from emp group by deptno;

# spark

1）先安装maven

export MAVEN\_OPTS="-Xmx2g -XX:ReservedCodeCacheSize=512m"

如果遇到如下问题

Failed to execute goal on project spark-launcher\_2.11: Could not resolve dependencies for project org.apache.spark:spark-launcher\_2.11:jar:2.1.0: Could not find artifact org.apache.hadoop:hadoop-client:jar:2.6.0-cdh5.7.0 in central (https://repo1.maven.org/maven2) -> [Help 1]

就在pom文件里加个仓库链接

<repository>  
<id>cloudera</id>  
<url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>  
</repository>

2）编译完成后解压

tar -zxvf spark-2.1.0-bin-2.6.0-cdh5.7.0.tgz -C ~/app/

1. 启动spark local模式

spark-shell --master local[2]

4）standalone模式

到conf目录下，拷贝一份文件

cp spark-env.sh.template spark-env.sh

SPARK\_MASTER\_HOST=hadoop001

SPARK\_WORKER\_CORES=2

SPARK\_WORKER\_MEMORY=2g

SPARK\_WORKER\_INSTANCES=1

修改实例数量也可

SPARK\_WORKER\_INSTANCES=2

在sbin目录下，直接启动

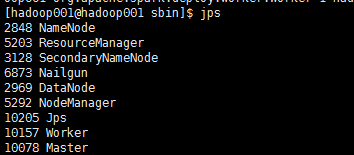
./start-all.sh

出现java\_home is not set的问题

然后在sbin目录下的spark-config.sh中配置

export JAVA\_HOME=/home/hadoop001/app/jdk1.8.0\_45

然后重启就会有两个worker



在conf下拷贝配置文件

cp slaves.template slaves

hadoop1 : master

hadoop2 : worker

hadoop3 : worker

hadoop4 : worker

...

hadoop10 : worker

slaves:

hadoop2

hadoop3

hadoop4

....

hadoop10

==> start-all.sh 会在 hadoop1机器上启动master进程，在slaves文件配置的所有hostname的机器上启动worker进程

还没试过。。。

5）再启动

spark-shell --master spark://hadoop001:7077

6）简单使用

val file = spark.sparkContext.textFile("file:///home/hadoop001/data/wc.txt")

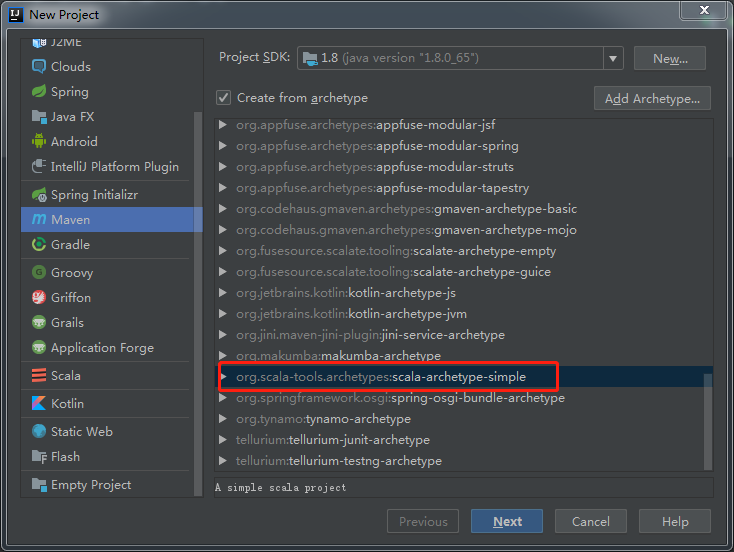
val wordCounts = file.flatMap(line => line.split(",")).map((word => (word, 1))).reduceByKey(\_ + \_)

wordCounts.collect

然后就能跑了

SQLContext的使用

使用idea创建一个工程



源码啥的就不贴出来了

打包成jar包上传到服务器环境中

提交Spark Application到环境中运行

spark-submit \

--name SQLContextApp \

--class com.SQLContextApp \

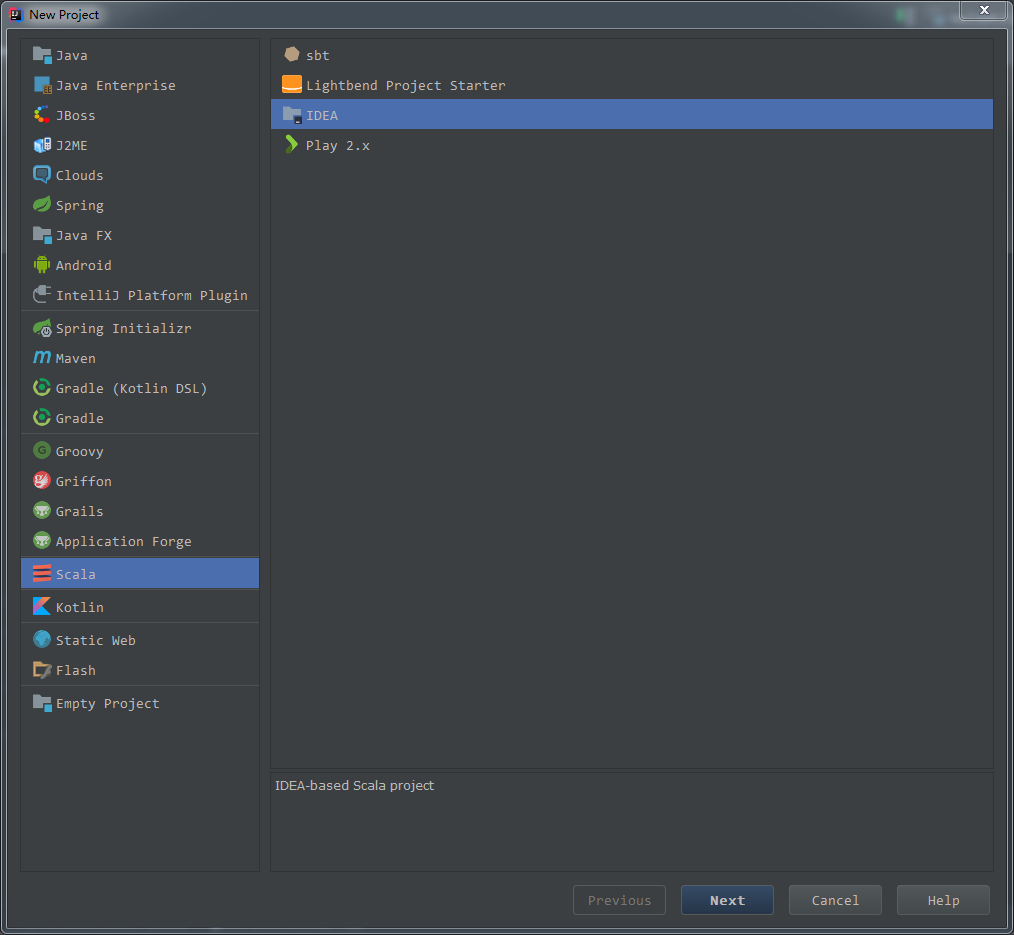
--master local[2] \

/home/hadoop001/lib/sql-1.0.jar \

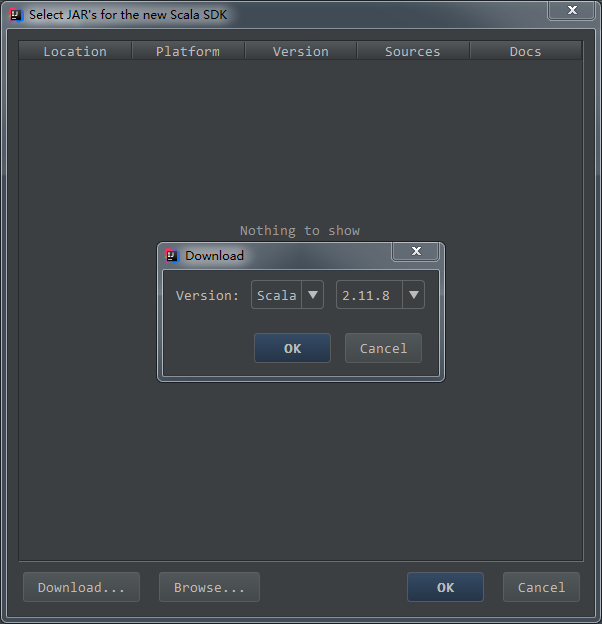
/home/hadoop001/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/examples/src/main/resources/people.json

安装idea的scala插件

新建工程的话这么建



顺便下载scala2.11.8



HiveContext的使用

省略

SparkSession的使用

在spark中使用hive

先启动hadoop

先拷贝hive-site.xml到spark的conf下

cp hive-site.xml /home/hadoop001/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/conf/

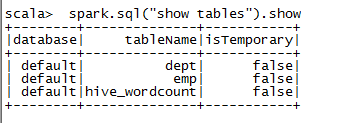
再启动spark

spark-shell --master local[2] --jars ~/software/mysql-connector-java-5.1.27.jar

执行下面语句

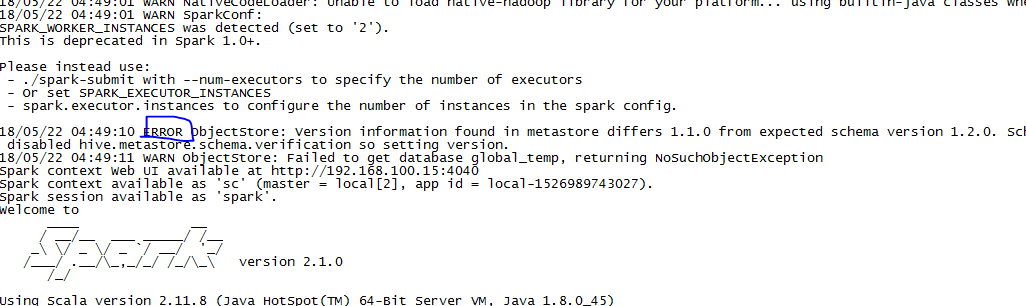
spark.sql("show tables").show

即可查出hive中的表



跟用原来的hive比较，速度是快的。

在启动spark过程中出现的错误

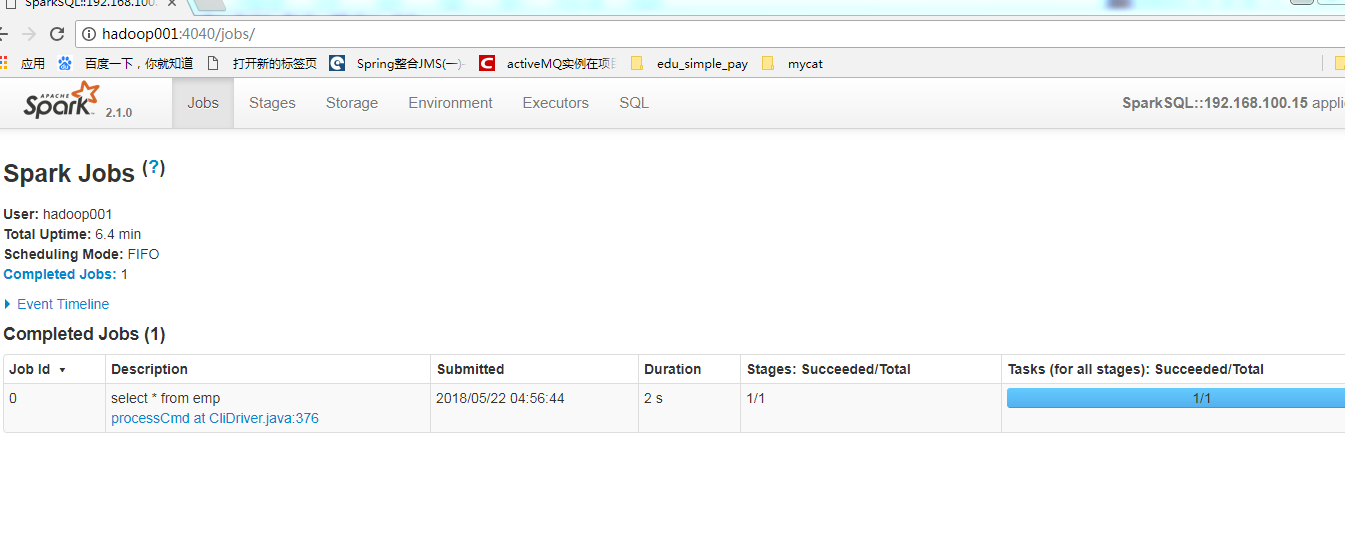


这个很简单去掉的，自己上网找。

spark-sql --master local[2] --jars ~/software/mysql-connector-java-5.1.27.jar

然后输入sql语句即可马上查询

访问网页，也能看到相关任务信息



接下来做一个事情

创建一个表

create table t(key string, value string);

explain extended select a.key\*(2+3), b.value from t a join t b on a.key = b.key and a.key > 3;

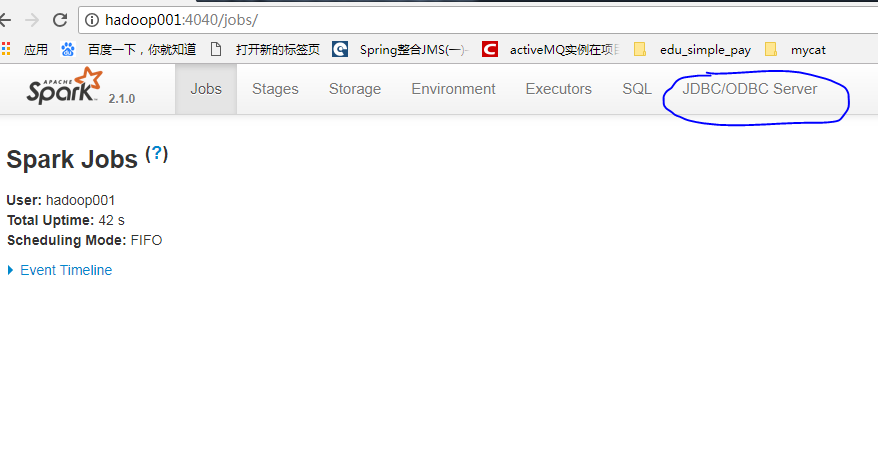
可以看到一些没啥用的东西

thriftserver&beeline的使用

在sbin目录下

./start-thriftserver.sh --master local[2] --jars ~/software/mysql-connector-java-5.1.27.jar

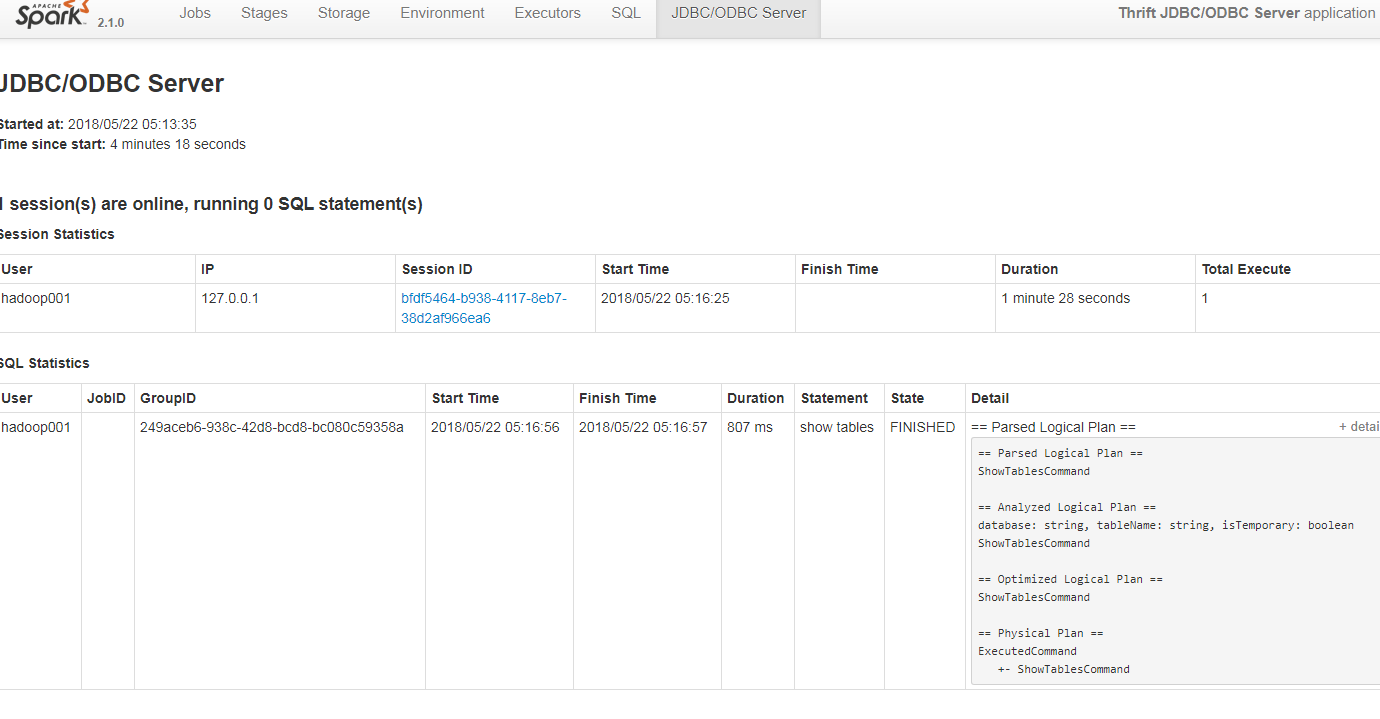
查看页面，你会发现多了一个



启动beeline（在bin目录下，不是sbin）

beeline -u jdbc:hive2://localhost:10000 -n hadoop001

然后也能查sql了



修改thriftserver启动占用的默认端口号：

./start-thriftserver.sh \

--master local[2] \

--jars ~/software/mysql-connector-java-5.1.27-bin.jar \

--hiveconf hive.server2.thrift.port=14000

beeline -u jdbc:hive2://localhost:14000 -n hadoop

jdbc方式编程访问

又是idea编程，自己看代码

dataframe

使用api对dataframe进行操作

又是idea编程，自己看吧

记得完成一个学生信息的统计例子

# 分析日志

1. 配置环境

想要将任务跑在yarn上，就要配置一些东西

export HADOOP\_CONF\_DIR=/home/hadoop/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop

或者

cp spark-env.sh.template spark-env.sh

spark-env.sh

配置进去也可

2）清洗数据

我们先将数据放到服务器硬盘上

再把数据上传到hdfs中

先创建目录

hadoop fs -mkdir -p /demo/input

hadoop fs -mkdir -p /demo/clean

上传文件

hadoop fs -put access.log /demo/input

3）提交任务

spark-submit \  
--class com.imooc.log.my.SparkStatFormatJobYARN \  
--name SparkStatFormatJobYARN \  
--master yarn \  
--executor-memory 1G \  
--num-executors 1 \  
/home/hadoop001/lib/spark-sql-demo-1.0-jar-with-dependencies.jar \

hdfs://hadoop001:8020/demo/input/\* hdfs://hadoop001:8020/demo/clean

spark-submit \  
--class com.imooc.log.my.SparkStatCleanJobYARN \  
--name SparkStatCleanJobYARN \  
--master yarn \  
--executor-memory 1G \  
--num-executors 1 \  
/home/hadoop001/lib/spark-sql-demo-1.0-jar-with-dependencies.jar \

hdfs://hadoop001:8020/demo/input/\* hdfs://hadoop001:8020/demo/clean2

分析出了一个parquet文件，这种文件不能正常打开的

1. 创建一个mysql数据库

建立一个新的数据库，名字就叫spark\_demo

create table day\_video\_access\_topn\_stat (  
day varchar(8) not null,  
cms\_id bigint(10) not null,  
times bigint(10) not null,  
primary key (day, cms\_id)  
);  
  
  
create table day\_video\_city\_access\_topn\_stat (  
day varchar(8) not null,  
cms\_id bigint(10) not null,  
city varchar(20) not null,  
times bigint(10) not null,  
times\_rank int not null,  
primary key (day, cms\_id, city)  
);  
  
create table day\_video\_traffics\_topn\_stat (  
day varchar(8) not null,  
cms\_id bigint(10) not null,  
traffics bigint(20) not null,  
primary key (day, cms\_id)  
);

1. 提交最终的统计任务

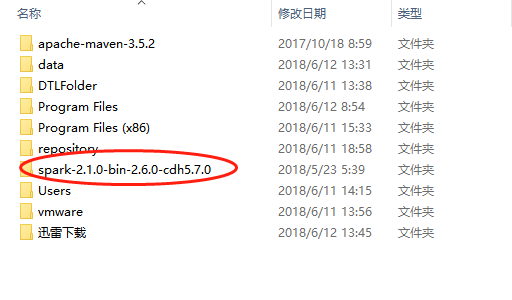
spark-submit \  
--class com.imooc.log.my.TopNStatJobYARN \  
--name TopNStatJobYARN \  
--master yarn \  
--executor-memory 1G \  
--num-executors 1 \  
/home/hadoop001/lib/spark-sql-demo-1.0-jar-with-dependencies.jar \

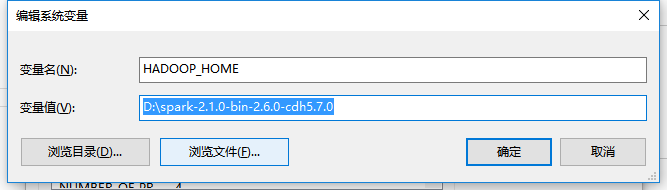
hdfs://hadoop001:8020/demo/clean2 20170511

创建一个hdfs的目录，传点东西上去然后跑任务就可以了

# windows下使用idea跑spark

解压spark，并且配置环境变量





Path下添加;%HADOOP\_HOME%\bin

将winutils.exe放入bin目录下，重启电脑

# 一个hadoop的集群

1）准备3台机器

192.168.31.101 hadoop001 200g（注意挂载点，把DataNode的挂载点人为调整大，我这里是root的挂载点，而linux系统默认比较大的挂在点是home）

192.168.31.102 hadoop002 200g

192.168.31.103 hadoop003 200g

2）修改windows下的映射

3）修改3台主机名字

hostnamectl set-hostname centos77.magedu.com

vim /etc/sysconfig/network

4）修改映射（hadoop001）

vim /etc/hosts

各节点分配

hadoop002：NameNode/DataNode ResourceManager/NodeManager

hadoop003：DataNode NodeManager

hadoop004：DataNode NodeManager

5）配置ssh免密码登录

在每台机器上运行

ssh-keygen -t rsa

然后在hadoop001上运行

ssh-copy-id -i ~/.ssh/id\_rsa.pub hadoop001

ssh-copy-id -i ~/.ssh/id\_rsa.pub hadoop002

ssh-copy-id -i ~/.ssh/id\_rsa.pub hadoop003

验证是否通

ssh hadoop001

ssh hadoop002

ssh hadoop003

6）创建一些目录

在hadoop001创建目录

mkdir software

mkdir app

7）配置jdk和hadoop

在hadoop001配即可

先安装yum install lrzsz

上传jdk和jadoop然后解压到app目录下

tar -zxf ~/software/hadoop-2.6.0-cdh5.7.0.tar.gz -C ~/app/

tar -zxf ~/software/jdk-8u45-linux-x64.tar.gz -C ~/app/

8）配置hadoop-env.sh

export JAVA\_HOME=/home/hadoop/app/jdk1.8.0\_45

9）配置core-site.xml

<property>

<name>fs.default.name</name>

<value>hdfs://hadoop002:8020</value>

</property>

10）配置hdfs-site.xml

<property>

<name>dfs.namanode.name.dir</name>

<value>/home/hadoop/app/tmp/dfs/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/home/hadoop/app/tmp/dfs/data</value>

</property>

10）配置yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>hadoop002</value>

</property>

11）配置mapred-site.xml

cp mapred-site.xml.template mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

12）配置slaves

hadoop001

hadoop002

hadoop003

1. 分发

scp -r ~/app hadoop@hadoop002:~/

scp -r ~/app hadoop@hadoop003:~/

scp ~/.bash\_profile hadoop@hadoop002:~/

scp ~/.bash\_profile hadoop@hadoop003:~/

13）格式化hdfs

在hadoop001上进行hdfs格式化

hdfs namenode -format

14）记得关闭防火墙

（3台机器）

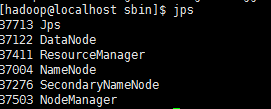
service iptables stop

开机关闭防火墙

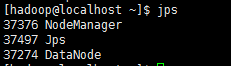
chkconfig iptables off

15）启动

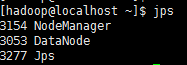
在hadoop001上启动即可



hadoop002



hadoop003



查看web

[http://hadoop001:50070/](http://hadoop002:50070/)

[http://hadoop001:8088/](http://hadoop002:8088/)

1. 配置spark集群

cp spark-env.sh.template spark-env.sh

JAVA\_HOME=/home/hadoop/app/jdk1.8.0\_45

SPARK\_MASTER\_HOST=hadoop001

SPARK\_WORKER\_CORES=2

SPARK\_WORKER\_MEMORY=2g

SPARK\_WORKER\_INSTANCES=2

export HADOOP\_CONF\_DIR=/home/hadoop/app/hadoop-2.6.0-cdh5.7.0/etc/hadoop

cp slaves.template slaves

hadoop002

hadoop003

scp -r ~/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/ hadoop@hadoop002:~/app/

scp -r ~/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/ hadoop@hadoop003:~/app/

启动spark

sbin/start-all.sh

测试集群

yarn模式

spark-submit \

--class org.apache.spark.examples.SparkPi \

--master yarn \

--deploy-mode cluster \

--driver-memory 1G \

--executor-memory 1G \

--num-executors 3 \

--executor-cores 1 \

/home/hadoop/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/examples/jars/spark-examples\_2.11-2.1.0.jar 40

Standalone 集群模式运行

spark-submit \

--class org.apache.spark.examples.SparkPi \

--master spark://hadoop001:7077 \

/home/hadoop/app/spark-2.1.0-bin-2.6.0-cdh5.7.0/examples/jars/spark-examples\_2.11-2.1.0.jar 40

第一步清洗

spark-submit \  
--class com.imooc.log.my.SparkStatFormatJobYARN \  
--name SparkStatFormatJobYARN \  
--master yarn \

--deploy-mode cluster \  
--executor-memory 1G \

--num-executors 1 \

--executor-cores 1 \  
/home/hadoop/libs/spark-sql-demo-1.0-jar-with-dependencies.jar \

file:/home/hadoop/data/input/\* file:/home/hadoop/data/output/

spark-submit \  
--class com.imooc.log.my.SparkStatFormatJobYARN \  
--name SparkStatFormatJobYARN \  
--master yarn \

--deploy-mode cluster \  
--executor-memory 1G \

--num-executors 3 \

--executor-cores 2 \  
/home/hadoop/libs/spark-sql-demo-1.0-jar-with-dependencies.jar \

hdfs://hadoop001:8020/demo/input/\* hdfs://hadoop001:8020/demo/clean

hadoop fs -rmr /demo/clean

第二步清洗

spark-submit \  
--class com.imooc.log.my.SparkStatCleanJobYARN \  
--name SparkStatCleanJobYARN \  
--master yarn \

--deploy-mode cluster \  
--executor-memory 1G \  
--num-executors 3 \

--executor-cores 2 \  
/home/hadoop/libs/spark-sql-demo-1.0-jar-with-dependencies.jar \

hdfs://hadoop001:8020/demo/input/\* hdfs://hadoop001:8020/demo/clean3

spark-submit \  
--class com.imooc.log.my.SparkStatCleanJobYARN \  
--name SparkStatCleanJobYARN \  
--master spark://hadoop001:7077 \  
/home/hadoop/libs/spark-sql-demo-1.0-jar-with-dependencies.jar \

hdfs://hadoop001:8020/demo/input/\* hdfs://hadoop001:8020/demo/clean3

spark-submit \  
--class com.zjicm.spark.Launcher \  
--name SparkStatCleanJobYARN \  
--master yarn \

--deploy-mode client \  
--executor-memory 1G \  
--num-executors 3 \

--executor-cores 2 \  
/home/hadoop/libs/springboot-spark-0.0.1-SNAPSHOT-jar-with-dependencies.jar \

2 hdfs://hadoop001:8020/demo/input/\* hdfs://hadoop001:8020/demo/clean3

hadoop fs -rmr /demo/clean2

集群测试

使用该模式可便于调试

spark-shell --master local[2]



# Flume的使用

tar -zxf ~/software/apache-flume-1.8.0-bin.tar.gz -C ~/app/

在conf下新建一个叫a4.conf的配置文件写入下面

#定义agent名， source、channel、sink的名称

a4.sources = r1

a4.channels = c1

a4.sinks = k1

#具体定义source

a4.sources.r1.type = spooldir

a4.sources.r1.spoolDir = /home/hadoop/logs

#a4.sources.r1.ignorePattern = ^(.)\*\\.log$

#具体定义channel

a4.channels.c1.type = memory

a4.channels.c1.capacity = 10000

a4.channels.c1.transactionCapacity = 100

#定义拦截器，为消息添加时间戳

a4.sources.r1.interceptors = i1

a4.sources.r1.interceptors.i1.type = org.apache.flume.interceptor.TimestampInterceptor$Builder

#具体定义sink

a4.sinks.k1.type = hdfs

a4.sinks.k1.hdfs.path = hdfs://hadoop001/flume/%Y%m%d

a4.sinks.k1.hdfs.filePrefix = events-

a4.sinks.k1.hdfs.fileType = DataStream

#不按照条数生成文件

a4.sinks.k1.hdfs.rollCount = 0

#HDFS上的文件达到128M时生成一个文件

a4.sinks.k1.hdfs.rollSize = 134217728

#HDFS上的文件达到60秒生成一个文件

a4.sinks.k1.hdfs.rollInterval = 60

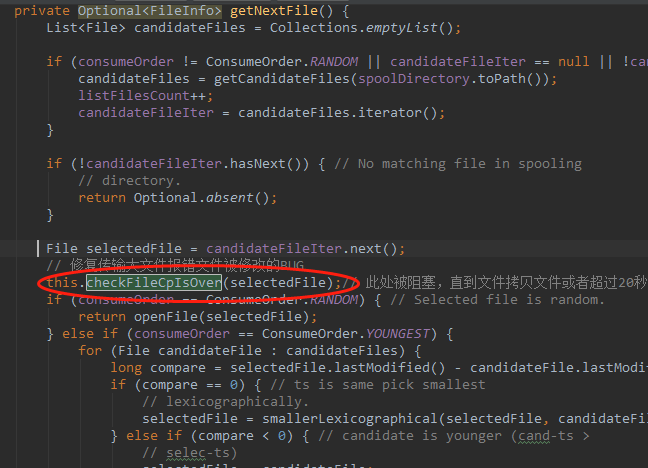
#组装source、channel、sink

a4.sources.r1.channels = c1

a4.sinks.k1.channel = c1

bin/flume-ng agent -n a4 -c conf -f conf/a4.conf -Dflume.root.logger=INFO,console

private void checkFileCpIsOver(File file) {  
 long modified = file.lastModified();// 目前文件的修改时间  
 long length = file.length();// 目前文件的大小  
 try {  
 Thread.*sleep*(1000);// 等待1秒钟  
 } catch (InterruptedException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 File currentFile = new File(file.getAbsolutePath());  
 int count = 0;// 记录循环次数，超过20次，也就是10秒后抛出异常  
 while (currentFile.lastModified() != modified || currentFile.length() != length) {  
 if (count > 20) {  
 String message = "File Copy time too long. please check copy whether exception!" + "\n" + "File at :"  
 + file.getAbsolutePath() + "\n" + "File current length is：" + currentFile.lastModified();  
 new IllegalStateException(message);  
 }  
 count++;  
 modified = currentFile.lastModified();  
 length = currentFile.length();  
 try {  
 Thread.*sleep*(500);// 等待500毫秒  
 } catch (InterruptedException e) {  
 // *TODO Auto-generated catch block* e.printStackTrace();  
 }  
 currentFile = new File(file.getAbsolutePath());  
  
 }  
 // 一直到文件传输完成就可以退出  
}



set MAVEN\_OPTS= -Xms512m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m

mvn clean install -DskipTests -Dcheckstyle.skip

# 常见问题总结



有关datanode启动不起来的问题

删掉tmp，然后重新格式化hdfs