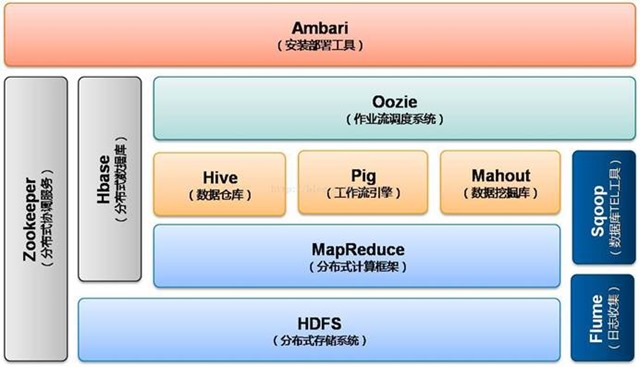
1. **大数据平台架构（hadoop）**

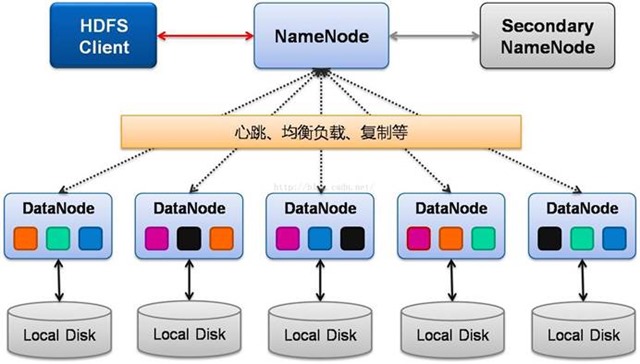
# **1、 hadoop1.0时期架构**



# **2、 hadoop2.0时期架构**

# https://images2017.cnblogs.com/blog/789140/201709/789140-20170926023530964-865715017.jpg

# **3、 hdfs架构**



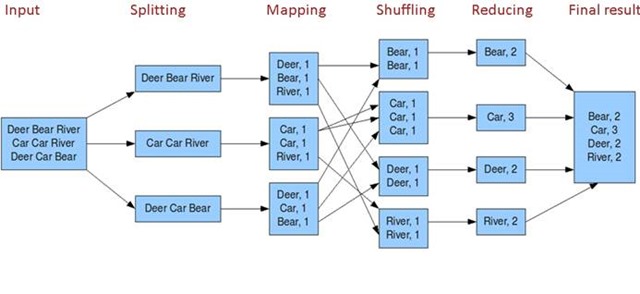
【Active Namenode】：主 Master（只有一个），管理 HDFS 的名称空间，管理数据块映射信息；配置副本策略；处理客户端读写请求

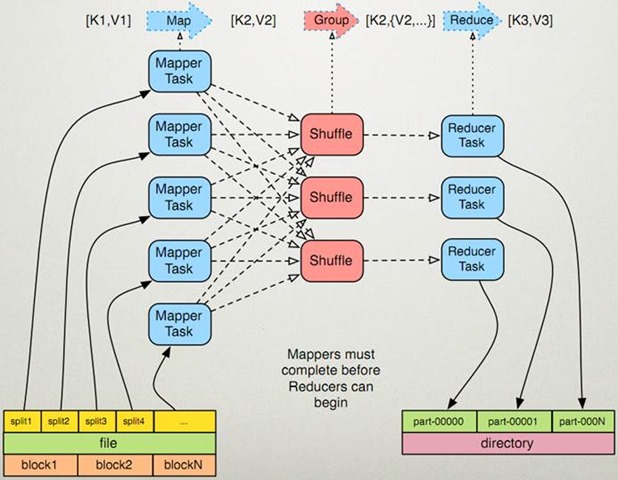
【Secondary NameNode】：NameNode 的热备；定期合并 fsimage 和 fsedits，推送给 NameNode；当 Active NameNode 出现故障时，快速切换为新的 Active NameNode。

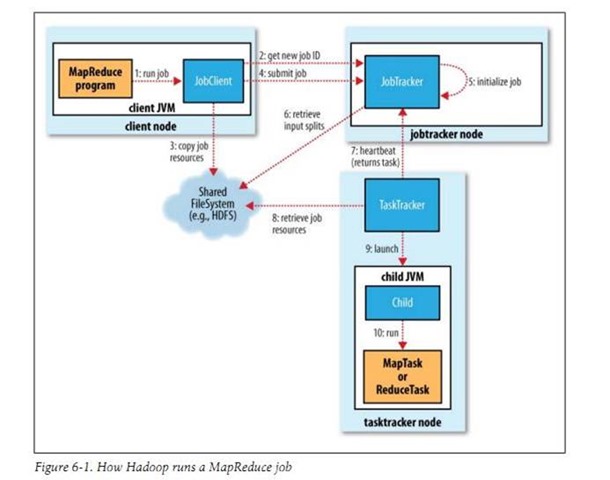
【Datanode】：Slave（有多个）；存储实际的数据块；执行数据块读 / 写

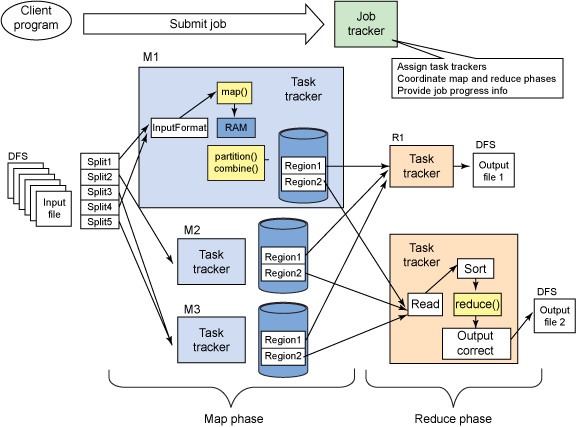
【Client】：与 NameNode 交互，获取文件位置信息；与 DataNode 交互，读取或者写入数据；管理 HDFS、访问 HDFS。

# ****4、**MapReduce**架构****

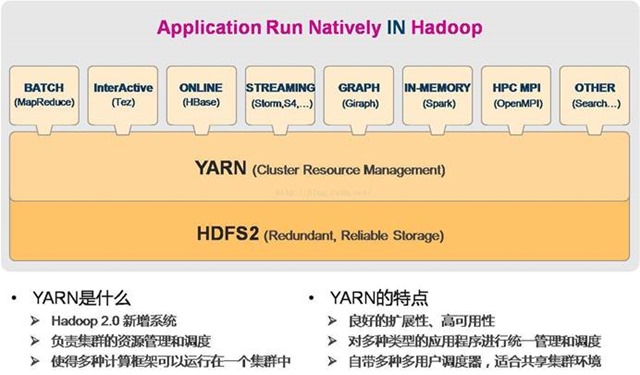


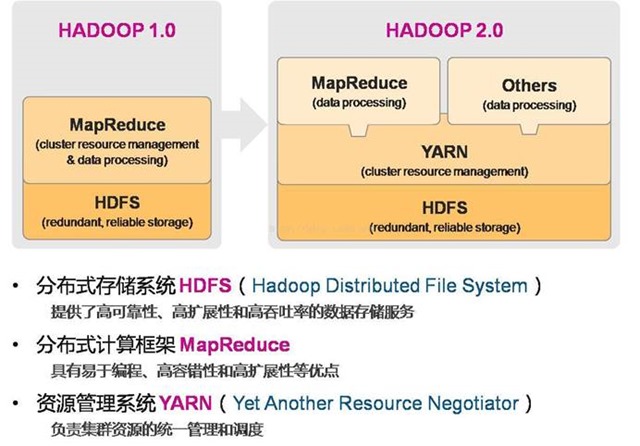






# **5、 yarn架构**





**二、大数据平台搭建（案例一）**

## （Hadoop生态+SparkStreaming）

## 1、平台需求

## 随着公司业务发展，对大数据的获取和实时处理的要求就会越来越高，日志处理、用户行为分析、场景业务分析等等，传统的写日志方式根本满足不了业务的实时处理需求，所以本人准备开始着手改造原系统中的数据处理方式，重新搭建一个实时流处理平台，主要是基于Hadoop生态，利用Kafka作为中转，SparkStreaming框架实时获取数据并清洗，将结果多维度的存储进HBase数据库

## 整个平台大致的框架如下：

## C:\Users\jenyou\AppData\Local\Temp\WeChat Files\ecb8afd8c1673b04fde76aaf6f778075.png

## 2、组件选型

* 操作系统：Centos7  
  　1. Flume1.8.0  
  　2. Hadoop2.9.0  
  　3. kafka2.11-1.0.0  
  　4. Spark2.2.1  
  　5. HBase1.2.6  
  　6. ZooKeeper3.4.11  
  　7. maven3.5.2

## 3、平台搭建

## 整体的开发环境是基于JDK1.8以上以及Scala，所以得提前把java和Scala的环境给准备好，接下来就开始着手搭建基础平台：

## 3.1 ****配置开发环境****

下载并解压JDK1.8,、下载并解压Scala，配置profile文件：

vim /etc/profile

|  |
| --- |
|  |

export JAVA\_HOME=/usr/java/jdk1.8.0\_144  
export PATH=$JAVA\_HOME/bin:$PATH  
export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar  
export SCALA\_HOME=/usr/local/scala-2.11.12  
export PATH=$PATH:$SCALA\_HOME/bin

source /etc/profile

## 3.2 ****配置zookeeper、maven环境****

## 下载并解压zookeeper以及maven并配置profile文件

wget http://mirrors.hust.edu.cn/apache/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz  
tar -zxvf apache-maven-3.5.2-bin.tar.gz -C /usr/local  
wget http://mirrors.hust.edu.cn/apache/zookeeper/zookeeper-3.4.11/zookeeper-3.4.11.tar.gz  
tar -zxvf zookeeper-3.4.11.tar.gz -C /usr/local  
vim /etc/profile

export MAVEN\_HOME=/usr/local/apache-maven-3.5.2  
export PATH=$PATH:$MAVEN\_HOME/bin

source /etc/profile

zookeeper的配置文件配置一下：

cp/usr/local/zookeeper-3.4.11/conf/zoo\_sample.cfg /usr/local/zookeeper-3.4.11/conf/zoo.cfg

然后配置一下zoo.cfg里面的相关配置，指定一下dataDir目录等等

启动zookeeper：

/usr/local/zookeeper-3.4.11/bin/zkServer.sh start

如果不报错，jps看一下是否启动成功

## 3.3 ****安装配置Hadoop****

Hadoop的安装配置在之前文章中有说过（传送门），为了下面的步骤方便理解，这里只做一个单机版的简单配置说明：

下载hadoop解压并配置环境：

wget http://mirrors.hust.edu.cn/apache/hadoop/common/hadoop-2.9.0/hadoop-2.9.0.tar.gz  
tar -zxvf hadoop-2.9.0.tar.gz -C /usr/local  
vim /etc/profile

export HADOOP\_HOME=/usr/local/hadoop-2.9.0  
export PATH=$PATH:$HADOOP\_HOME/bin

source /etc/profile

配置hadoop 进入/usr/local/hadoop-2.9.0/etc/hadoop目录

cd /usr/local/hadoop-2.9.0/etc/hadoop

首先配置hadoop-env.sh、yarn-env.sh，修改JAVA\_HOME到指定的JDK安装目录/usr/local/java/jdk1.8.0\_144

创建hadoop的工作目录

mkdir /opt/data/hadoop

编辑core-site.xml、hdfs-site.xml、yarn-site.xml等相关配置文件，具体配置不再阐述请看前面的文章，配置完成之后记得执行hadoop namenode -format，否则hdfs启动会报错，启动完成后不出问题浏览器访问50070端口会看到hadoop的页面。

## 3.4 ****安装配置kafka****

还是一样，先下载kafka，然后配置：

wget http://mirrors.hust.edu.cn/apache/kafka/1.0.0/kafka\_2.11-1.0.0.tgz  
tar -zxvf kafka\_2.11-1.0.0.tgz -C /usr/local  
vim /etc/profile

export KAFKA\_HOME=/usr/local/kafka\_2.11-1.0.0  
export PATH=$KAFKA\_HOME/bin:$PATH

source /etc/profile

进入kafka的config目录，配置server.properties,指定log.dirs和zookeeper.connect参数；配置zookeeper.properties文件中zookeeper的dataDir，配置完成后启动kafka

kafka-server-start.sh -daemon $KAFKA\_HOME/config/server.properties

可以用jps查看有没有kafka进程，然后测试一下kafka是否能够正常收发消息，开两个终端，一个用来做producer发消息一个用来做consumer收消息，首先，先创建一个topic

kafka-topics.sh --create --zookeeper 127.0.0.1:2181

--replication-factor 1 --partitions 1 --topic testTopic  
kafka-topics.sh --describe --zookeeper localhost:2181 --topic testTopic

如果不出错，会看到如下输出：

Topic:testTopic    PartitionCount:1    ReplicationFactor:1 Configs:  
Topic: testTopic    Partition: 0    Leader: 0   Replicas: 0 Isr: 0

然后在第一个终端中输入命令：

kafka-console-producer.sh –broker-list localhost:9092 –topic testTopic  
在第二个终端中输入命令：

kafka-console-consumer.sh –zookeeper 127.0.0.1:2181 –topic testTopic  
如果启动都正常，那么这两个终端将进入阻塞监听状态，在第一个终端中输入任何消息第二个终端都将会接收到。

## 3.5 ****安装配置HBase****

下载并解压HBase：

wget http://mirrors.hust.edu.cn/apache/hbase/1.2.6/hbase-1.2.6-bin.tar.gz  
tar -zxvf hbase-1.2.6-bin.tar.gz -C /usr/local/  
vim /etc/profile

|  |
| --- |
|  |

export HBASE\_HOME=/usr/local/hbase-1.2.6  
export PATH=$PATH:$HBASE\_HOME/bin

source /etc/profile

修改hbase下的配置文件,首先修改hbase-env.sh，主要修改JAVA\_HOME以及相关参数，这里要说明一下HBASE\_MANAGES\_ZK这个参数，因为采用了自己的zookeeper，所以这里设置为false，否则hbase会自己启动一个zookeeper

cd /usr/local/hbase-1.2.6/conf  
vim hbase-env.sh

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_144/  
HBASE\_CLASSPATH=/usr/local/hbase-1.2.6/conf  
export HBASE\_MASTER\_OPTS="$HBASE\_MASTER\_OPTS -XX:PermSize=256m -XX:MaxPermSize=1024m"  
export HBASE\_REGIONSERVER\_OPTS="$HBASE\_REGIONSERVER\_OPTS -XX:PermSize=256m -XX:MaxPermSize=1024m"  
export HBASE\_PID\_DIR=/opt/data/hbase  
export HBASE\_MANAGES\_ZK=false

然后修改hbase-site.xml，我们设置hbase的文件放在hdfs中，所以要设置hdfs地址，其中tsk1是我安装hadoop的机器的hostname，hbase.zookeeper.quorum参数是安装zookeeper的地址，这里的各种地址最好用机器名

vim hbase-site.xml

<configuration>  
    <property>  
        <name>hbase.rootdir</name>  
        <value>hdfs://tsk1:9000/hbase</value>  
    </property>  
    <property>  
        <name>hbase.master</name>  
        <value>tsk1:60000</value>  
    </property>  
    <property>  
        <name>hbase.master.port</name>  
        <value>60000</value>  
    </property>  
    <property>  
        <name>hbase.cluster.distributed</name>  
        <value>true</value>  
    </property>  
    <property>  
        <name>hbase.zookeeper.quorum</name>  
        <value>192.168.70.135</value>  
    </property>  
    <property>  
        <name>zookeeper.znode.parent</name>  
        <value>/hbase</value>  
    </property>  
    <property>  
        <name>hbase.zookeeper.property.dataDir</name>  
        <value>/opt/data/zookeeper</value>  
    </property>  
    <property>  
        <name>hbase.master.info.bindAddress</name>  
        <value>tsk1</value>  
    </property>  
</configuration>

配置完成后启动hbase，输入命令：

start-hbase.sh  
完成后查看日志没有报错的话测试一下hbase，用hbase shell进行测试：

hbase shell  
hbase(main):001:0>create 'myTestTable','info'  
0 row(s) in 2.2460 seconds  
=> Hbase::Table - myTestTable  
hbase(main):003:0>list  
TABLE                                                                                                                      
testTable                                                                                                                  
1 row(s) in 0.1530 seconds  
=> ["myTestTable"]

至此，hbase搭建成功，访问以下hadoop的页面，查看file system(菜单栏Utilities->Browse the file system)，这时可以看见base的相关文件已经载hadoop的文件系统中。

## C:\Users\jenyou\AppData\Local\Temp\WeChat Files\b7c1d182b1cb98f99073ce1a4ce57015.jpg

## 3.6 ****安装spark****

下载spark并解压

wget http://mirrors.hust.edu.cn/apache/spark/spark-2.2.1/spark-2.2.1-bin-hadoop2.7.tgz  
tar -zxvf spark-2.2.1-bin-hadoop2.7.tgz -C /usr/local  
vim /etc/profile

|  |
| --- |
|  |

export SPARK\_HOME=/usr/local/spark-2.2.1-bin-hadoop2.7  
export PATH=$PATH:$SPARK\_HOME/bin

## source /etc/profile

**3.7 测试**

至此，环境基本搭建完成，以上搭建的环境仅是服务器生产环境的一部分，涉及服务器信息、具体调优信息以及集群的搭建就不写在这里了，下面我们写一段代码整体测试一下从kafka生产消息到spark streaming接收到，然后处理消息并写入HBase。先写一个HBase的连接类HBaseHelper：

|  |
| --- |
|  |

public class HBaseHelper {  
    private static HBaseHelper ME;  
    private static Configuration config;  
    private static Connection conn;  
    private static HBaseAdmin admin;  
    public static HBaseHelper getInstances() {  
        if (null == ME) {  
            ME = new HBaseHelper();  
            config = HBaseConfiguration.create();  
            config.set("hbase.rootdir", "hdfs://tsk1:9000/hbase");  
            config.set("hbase.zookeeper.quorum", "tsk1");  
            config.set("hbase.zookeeper.property.clientPort", "2181");  
            config.set("hbase.defaults.for.version.skip", "true");  
        }  
        if (null == conn) {  
            try {  
                conn = ConnectionFactory.createConnection(config);  
                admin = new HBaseAdmin(config);  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
        return ME;  
    }  
    public Table getTable(String tableName) {  
        Table table = null;  
        try {  
            table = conn.getTable(TableName.valueOf(tableName));  
        } catch (Exception ex) {  
            ex.printStackTrace();  
        }  
        return table;  
    }  
    public void putAdd(String tableName, String rowKey, String cf, String column, Long value) {  
        Table table = this.getTable(tableName);  
        try {  
            table.incrementColumnValue(rowKey.getBytes(), cf.getBytes(), column.getBytes(), value);  
            System.out.println("OK！");  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
 //......以下省略  
}

再写一个测试类KafkaRecHbase用来做spark-submit提交

package com.test.spark.spark\_test;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.regex.Pattern;  
import org.apache.log4j.Level;  
import org.apache.log4j.Logger;  
import org.apache.spark.SparkConf;  
import org.apache.spark.streaming.Durations;  
import org.apache.spark.streaming.api.java.JavaDStream;  
import org.apache.spark.streaming.api.java.JavaPairReceiverInputDStream;  
import org.apache.spark.streaming.api.java.JavaStreamingContext;  
import org.apache.spark.streaming.kafka.KafkaUtils;  
import scala.Tuple2;  
public class KafkaRecHbase {  
    private static final Pattern SPACE = Pattern.compile(" ");  
    public static void main(String[] args) throws Exception {  
        Logger.getLogger("org").setLevel(Level.ERROR);  
        SparkConf sparkConf = new SparkConf();  
        sparkConf.setAppName("kafkaRecHbase");  
        sparkConf.setMaster("local[2]");  
        JavaStreamingContext ssc = new JavaStreamingContext(sparkConf, Durations.seconds(5));  
        int numThreads = Integer.parseInt(args[3]);  
        Map<String, Integer> topicMap = new HashMap<>();  
        String[] topics = args[2].split(",");  
        for (String topic : topics) {  
            topicMap.put(topic, numThreads);  
        }  
        JavaPairReceiverInputDStream<String, String> kafkaStream =  
                KafkaUtils.createStream(ssc, args[0], args[1], topicMap);  
        JavaDStream<String> lines = kafkaStream.map(Tuple2::\_2);  
        JavaDStream<String> lineStr = lines.map(line -> {  
            if (null == line || line.equals("")) {  
                return "";  
            }  
            String[] strs = SPACE.split(line);  
            if (strs.length < 1) {  
                return "";  
            }  
            try {  
                for (String str : strs) {  
                    HBaseHelper.getInstances().putAdd("myTestTable", str, "info", "wordCunts", 1l);  
                }  
                return "strs:" + line;  
            } catch (Exception ex) {  
                System.out.println(line);  
                return "报错了：" + ex.getMessage();  
            }  
        });  
        lineStr.print();  
        ssc.start();  
        System.out.println("spark 启动！！！");  
        ssc.awaitTermination();  
    }  
}

编译提交到服务器，执行命令：

spark-submit --jars $(echo /usr/local/hbase-1.2.6/lib/\*.jar | tr ' ' ',') --class com.test.spark.spark\_test.KafkaRecHbase --packages org.apache.spark:spark-streaming-kafka-0-8\_2.11:2.2.1 /opt/FileTemp/streaming/spark-test-0.1.1.jar tsk1:2181 test testTopic 1

没报错的话执行kafka的producer，输入几行数据在HBase内就能看到结果了！

## 3.8 ****装一个Flume实时采集Nginx日志写入Kafka****

Flume是一个用来日志采集的框架，安装和配置都比较简单，可以支持多个数据源和输出，具体可以参考Flume的文档，写的比较全传送门

下载Flume并配置环境

wget http://mirrors.hust.edu.cn/apache/flume/1.8.0/apache-flume-1.8.0-bin.tar.gz  
tar -zxvf apache-flume-1.8.0-bin.tar.gz -C /usr/local  
vim /etc/profile

export FLUME\_HOME=/usr/local/apache-flume-1.8.0-bin/  
export PATH=$FLUME\_HOME/bin:$PATH

source /etc/profile

写一个Flume的配置文件在flume的conf目录下：

vim nginxStreamingKafka.conf

|  |
| --- |
|  |

agent1.sources=r1  
agent1.channels=logger-channel  
agent1.sinks=kafka-sink  
agent1.sources.r1.type=exec  
agent1.sources.r1.deserializer.outputCharset= UTF-8  
agent1.sources.r1.command=tail -F /opt/data/nginxLog/nginxLog.log  
agent1.channels.logger-channel.type=memory  
agent1.sinks.kafka-sink.type=org.apache.flume.sink.kafka.KafkaSink  
agent1.sinks.kafka-sink.topic = flumeKafka  
agent1.sinks.kafka-sink.brokerList = tsk1:9092  
agent1.sinks.kafka-sink.requiredAcks = 1  
agent1.sinks.kafka-sink.batchSize = 20  
agent1.sources.r1.channels=logger-channel  
agent1.sinks.kafka-sink.channel=logger-channel

kafka创建一个名为flumeKafka的topic用来接收，然后启动flume：

flume-ng agent --name agent1 --conf $FLUME\_HOME/conf --conf-file $FLUME\_HOME/conf/nginxStreamingKafka.conf -Dflume.root.logger=INFO,console

如果没有报错，Flume将开始采集opt/data/nginxLog/nginxLog.log中产生的日志并实时推送给kafka，再按照上面方法写一个spark streaming的处理类进行相应的处理就好。真好了？ NO，这只是开始。。。。后面的路长着呢

**三、大数据平台搭建（案例二）**

**1. 服务器基本信息**

| **主机名** | **ip地址** | **安装服务** |
| --- | --- | --- |
| spark-master | 172.16.200.81 | jdk、hadoop、spark、scala |
| spark-slave01 | 172.16.200.82 | jdk、hadoop、spark |
| spark-slave02 | 172.16.200.83 | jdk、hadoop、spark |
| spark-slave03 | 172.16.200.84 | jdk、hadoop、spark |

**2. 软件基本信息**

| **软件名** | **版本** | **安装路径** |
| --- | --- | --- |
| oracle jdk | 1.8.0\_111 | /usr/local/jdk1.8.0\_111 |
| hadoop | 2.7.1 | /usr/local/hadoop-2.7.3 |
| spark | 2.0.2 | /usr/local/spark-2.0.2 |
| scala | 2.12.1 | /usr/local/scala2.12.1 |

**3.环境变量汇总**

############# java ############

export JAVA\_HOME=/usr/local/jdk1.8.0\_111

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

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

########### hadoop ##########

export HADOOP\_HOME=/usr/local/hadoop-2.7.3

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

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

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

######### spark ############

export SPARK\_HOME=/usr/local/spark-2.0.2

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

######### scala ##########

export SCALA\_HOME=/usr/local/scala-2.12.1

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

**4. 基本环境配置（master、slave相同操作）**

**4.1 配置jdk**

cd /usr/loca/src/

tar -C /usr/local/ -xzf /usr/local/src/jdk-8u111-linux-x64.tar.gz

**4.2 配置java环境变量**

vim /etc/profile

添加如下信息

######### jdk ############

export JAVA\_HOME=/usr/local/jdk1.8.0\_111

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

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

**4.3 刷新配置文件：**

source /etc/profile

**4.4 配置hosts**

vim /etc/hosts

172.16.200.81 spark-master

172.16.200.82 spark-slave1

172.16.200.83 spark-slave2

**4.5 配置免密码**

生成密钥对

ssh-keygen

如果密钥不设置密码，则连按几下回车

先配置本机免密码登录

cd /root/.ssh

cat id\_rsa.pub > authorized\_keys

chmod 600 authorized\_keys

再将其它主机id\_rsa.pub 内容追加到 authorized\_keys中，三台配置完成后即可实现免密码登录

**二.大数据平台搭建**

**1. 搭建Hadoop（master、slave相同操作）**

**1.1 安装hadoop**

cd /usr/loca/src/

tar -C /usr/local/ -xzf hadoop-2.7.3.tar.gz

**1.2 配置hadoop环境变量**

vim /etc/profile

添加如下信息

######### hadoop ############

export HADOOP\_HOME=/usr/local/hadoop-2.7.3

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

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

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

**1.3 刷新配置文件：**

source /etc/profile

**1.4 修改hadoop配置文件**

cd /usr/local/hadoop-2.7.3/etc/hadoop

查看

root@spark-master hadoop]# ll

总用量 152

-rw-r--r--. 1 root root 4436 8月 18 09:49 capacity-scheduler.xml

-rw-r--r--. 1 root root 1335 8月 18 09:49 configuration.xsl

-rw-r--r--. 1 root root 318 8月 18 09:49 container-executor.cfg

-rw-r--r--. 1 root root 1037 12月 21 14:58 core-site.xml

-rw-r--r--. 1 root root 3589 8月 18 09:49 hadoop-env.cmd

-rw-r--r--. 1 root root 4235 12月 21 11:17 hadoop-env.sh

-rw-r--r--. 1 root root 2598 8月 18 09:49 hadoop-metrics2.properties

-rw-r--r--. 1 root root 2490 8月 18 09:49 hadoop-metrics.properties

-rw-r--r--. 1 root root 9683 8月 18 09:49 hadoop-policy.xml

-rw-r--r--. 1 root root 1826 12月 21 14:11 hdfs-site.xml

-rw-r--r--. 1 root root 1449 8月 18 09:49 httpfs-env.sh

-rw-r--r--. 1 root root 1657 8月 18 09:49 httpfs-log4j.properties

-rw-r--r--. 1 root root 21 8月 18 09:49 httpfs-signature.secret

-rw-r--r--. 1 root root 620 8月 18 09:49 httpfs-site.xml

-rw-r--r--. 1 root root 3518 8月 18 09:49 kms-acls.xml

-rw-r--r--. 1 root root 1527 8月 18 09:49 kms-env.sh

-rw-r--r--. 1 root root 1631 8月 18 09:49 kms-log4j.properties

-rw-r--r--. 1 root root 5511 8月 18 09:49 kms-site.xml

-rw-r--r--. 1 root root 11237 8月 18 09:49 log4j.properties

-rw-r--r--. 1 root root 931 8月 18 09:49 mapred-env.cmd

-rw-r--r--. 1 root root 1383 8月 18 09:49 mapred-env.sh

-rw-r--r--. 1 root root 4113 8月 18 09:49 mapred-queues.xml.template

-rw-r--r--. 1 root root 1612 12月 21 12:03 mapred-site.xml

-rw-r--r--. 1 root root 56 12月 21 16:30 slaves

-rw-r--r--. 1 root root 2316 8月 18 09:49 ssl-client.xml.example

-rw-r--r--. 1 root root 2268 8月 18 09:49 ssl-server.xml.example

-rw-r--r--. 1 root root 2191 8月 18 09:49 yarn-env.cmd

-rw-r--r--. 1 root root 4564 12月 21 11:19 yarn-env.sh

-rw-r--r--. 1 root root 1195 12月 21 14:24 yarn-site.xml

**1.4.1 修改hadoop全局配置文件**

vim core-site.xml

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

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

<!--配置namenode的地址-->

<property>

<name>fs.defaultFS</name>

<value>hdfs://172.16.200.81:9000</value>

</property>

<!-- 指定hadoop运行时产生文件的存储目录 -->

<property>

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

<value>file:///data/hadoop/data/tmp</value>

</property>

</configuration>

**1.4.2 配置hadoop关联jdk**

vim hadoop-env.sh

# Licensed to the Apache Software Foundation (ASF) under one

# or more contributor license agreements. See the NOTICE file

# distributed with this work for additional information

# regarding copyright ownership. The ASF licenses this file

# to you under the Apache License, Version 2.0 (the

# "License"); you may not use this file except in compliance

# with the License. You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA\_HOME. All others are

# optional. When running a distributed configuration it is best to

# set JAVA\_HOME in this file, so that it is correctly defined on

# remote nodes.

# The java implementation to use.

#配置jdk的环境

export JAVA\_HOME=/usr/local/jdk1.8.0\_111

# The jsvc implementation to use. Jsvc is required to run secure datanodes

# that bind to privileged ports to provide authentication of data transfer

# protocol. Jsvc is not required if SASL is configured for authentication of

# data transfer protocol using non-privileged ports.

#export JSVC\_HOME=${JSVC\_HOME}

export HADOOP\_CONF\_DIR=${HADOOP\_CONF\_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.

for f in $HADOOP\_HOME/contrib/capacity-scheduler/\*.jar; do

if [ "$HADOOP\_CLASSPATH" ]; then

export HADOOP\_CLASSPATH=$HADOOP\_CLASSPATH:$f

else

export HADOOP\_CLASSPATH=$f

fi

done

# The maximum amount of heap to use, in MB. Default is 1000.

#export HADOOP\_HEAPSIZE=

#export HADOOP\_NAMENODE\_INIT\_HEAPSIZE=""

# Extra Java runtime options. Empty by default.

export HADOOP\_OPTS="$HADOOP\_OPTS -Djava.net.preferIPv4Stack=true"

# Command specific options appended to HADOOP\_OPTS when specified

export HADOOP\_NAMENODE\_OPTS="-Dhadoop.security.logger=${HADOOP\_SECURITY\_LOGGER:-INFO,RFAS} -Dhdfs.audit.logger=${HDFS\_AUDIT\_LOGGER:-INFO,NullAppender} $HADOOP\_NAMENODE\_OPTS"

export HADOOP\_DATANODE\_OPTS="-Dhadoop.security.logger=ERROR,RFAS $HADOOP\_DATANODE\_OPTS"

export HADOOP\_SECONDARYNAMENODE\_OPTS="-Dhadoop.security.logger=${HADOOP\_SECURITY\_LOGGER:-INFO,RFAS} -Dhdfs.audit.logger=${HDFS\_AUDIT\_LOGGER:-INFO,NullAppender} $HADOOP\_SECONDARYNAMENODE\_OPTS"

export HADOOP\_NFS3\_OPTS="$HADOOP\_NFS3\_OPTS"

export HADOOP\_PORTMAP\_OPTS="-Xmx512m $HADOOP\_PORTMAP\_OPTS"

# The following applies to multiple commands (fs, dfs, fsck, distcp etc)

export HADOOP\_CLIENT\_OPTS="-Xmx512m $HADOOP\_CLIENT\_OPTS"

#HADOOP\_JAVA\_PLATFORM\_OPTS="-XX:-UsePerfData $HADOOP\_JAVA\_PLATFORM\_OPTS"

# On secure datanodes, user to run the datanode as after dropping privileges.

# This \*\*MUST\*\* be uncommented to enable secure HDFS if using privileged ports

# to provide authentication of data transfer protocol. This \*\*MUST NOT\*\* be

# defined if SASL is configured for authentication of data transfer protocol

# using non-privileged ports.

export HADOOP\_SECURE\_DN\_USER=${HADOOP\_SECURE\_DN\_USER}

# Where log files are stored. $HADOOP\_HOME/logs by default.

#export HADOOP\_LOG\_DIR=${HADOOP\_LOG\_DIR}/$USER

# Where log files are stored in the secure data environment.

export HADOOP\_SECURE\_DN\_LOG\_DIR=${HADOOP\_LOG\_DIR}/${HADOOP\_HDFS\_USER}

###

# HDFS Mover specific parameters

###

# Specify the JVM options to be used when starting the HDFS Mover.

# These options will be appended to the options specified as HADOOP\_OPTS

# and therefore may override any similar flags set in HADOOP\_OPTS

#

# export HADOOP\_MOVER\_OPTS=""

###

# Advanced Users Only!

###

# The directory where pid files are stored. /tmp by default.

# NOTE: this should be set to a directory that can only be written to by

# the user that will run the hadoop daemons. Otherwise there is the

# potential for a symlink attack.

export HADOOP\_PID\_DIR=${HADOOP\_PID\_DIR}

export HADOOP\_SECURE\_DN\_PID\_DIR=${HADOOP\_PID\_DIR}

# A string representing this instance of hadoop. $USER by default.

export HADOOP\_IDENT\_STRING=$USER

**1.4.3 配置hdfs**

vim hdfs-site.xml

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

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<!--指定hdfs的副本数-->

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<!--设置hdfs的权限-->

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

<!-- secondary name node web 监听端口 -->

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>172.16.200.81:50090</value>

</property>

<!-- name node web 监听端口 -->

<property>

<name>dfs.namenode.http-address</name>

<value>172.16.200.81:50070</value>

</property>

<!-- 真正的datanode数据保存路径 -->

<property>

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

<value>file:///data/hadoop/data/dfs/dn</value>

</property>

<!-- NN所使用的元数据保存-->

<property>

<name>dfs.namenode.name.dir</name>

<value>file:///data/hadoop/data/dfs/nn/name</value>

</property>

<!--存放 edit 文件-->

<property>

<name>dfs.namenode.edits.dir</name>

<value>file:///data/hadoop/data/dfs/nn/edits</value>

</property>

<!-- secondary namenode 节点存储 checkpoint 文件目录-->

<property>

<name>dfs.namenode.checkpoint.dir</name>

<value>file:///data/hadoop/data/dfs/snn/name</value>

</property>

<!-- secondary namenode 节点存储 edits 文件目录-->

<property>

<name>dfs.namenode.checkpoint.edits.dir</name>

<value>file:///data/hadoop/data/dfs/snn/edits</value>

</property>

</configuration>

**1.4.4 配置mapred**

vim mapred-site.xml

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<!-- 指定mr运行在yarn上 -->

<property>

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

<value>yarn</value>

</property>

<!--历史服务的web端口地址 -->

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>172.16.200.81:19888</value>

</property>

<!--历史服务的端口地址-->

<property>

<name>mapreduce.jobhistory.address</name>

<value>172.16.200.81:10020</value>

</property>

<!--Uber运行模式-->

<property>

<name>mapreduce.job.ubertask.enable</name>

<value>false</value>

</property>

<!--MapReduce作业产生的日志存放位置。-->

<property>

<name>mapreduce.jobhistory.intermediate-done-dir</name>

<value>${yarn.app.mapreduce.am.staging-dir}/history/done\_intermediate</value>

</property>

<!--MR JobHistory Server管理的日志的存放位置-->

<property>

<name>mapreduce.jobhistory.done-dir</name>

<value>${yarn.app.mapreduce.am.staging-dir}/history/done</value>

</property>

<!--是job运行时的临时文件夹-->

<property>

<name>yarn.app.mapreduce.am.staging-dir</name>

<value>/data/hadoop/hadoop-yarn/staging</value>

</property>

</configuration>

**1.4.5 配置slaves**

vim slaves

172.16.200.81

172.16.200.82

172.16.200.83

172.16.200.84

**1.4.6 配置yarn**

vim yarn-site.xml

<?xml version="1.0"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<configuration>

<!-- 指定nodeManager组件在哪个机子上跑 -->

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<!-- 指定resourcemanager组件在哪个机子上跑 -->

<property>

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

<value>172.16.200.81</value>

</property>

<!--resourcemanager web地址-->

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>172.16.200.81:8088</value>

</property>

<!--启用日志聚集功能-->

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

</property>

<!--在HDFS上聚集的日志最多保存多长时间-->

<property>

<name>yarn.log-aggregation.retain-seconds</name>

<value>86400</value>

</property>

</configuration>

**2. 搭建Spark（master、slave相同操作）**

**2.1 安装spark**

cd /usr/loca/src/

tar zxvf spark-2.0.2-bin-hadoop2.7.tgz

mv spark-2.0.2-bin-hadoop2.7 /usr/local/spark-2.0.2

**2.2 配置spark环境变量**

vim /etc/profile

添加如下信息

######### spark ############

export SPARK\_HOME=/usr/local/spark-2.0.2

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

**2.3 刷新配置文件：**

source /etc/profile

**2.4 修改spark配置文件**

cd /usr/local/spark-2.0.2/conf

mv spark-env.sh.template spark-env.sh

[root@spark-master conf]# ll

总用量 36

-rw-r--r--. 1 500 500 987 11月 8 09:58 docker.properties.template

-rw-r--r--. 1 500 500 1105 11月 8 09:58 fairscheduler.xml.template

-rw-r--r--. 1 500 500 2025 11月 8 09:58 log4j.properties.template

-rw-r--r--. 1 500 500 7239 11月 8 09:58 metrics.properties.template

-rw-r--r--. 1 500 500 912 12月 21 16:55 slaves

-rw-r--r--. 1 500 500 1292 11月 8 09:58 spark-defaults.conf.template

-rwxr-xr-x. 1 root root 3969 12月 21 15:50 spark-env.sh

-rwxr-xr-x. 1 500 500 3861 11月 8 09:58 spark-env.sh.template

**2.4.1 spark关联jdk**

vim spark-env.sh

#!/usr/bin/env bash

#

# Licensed to the Apache Software Foundation (ASF) under one or more

# contributor license agreements. See the NOTICE file distributed with

# this work for additional information regarding copyright ownership.

# The ASF licenses this file to You under the Apache License, Version 2.0

# (the "License"); you may not use this file except in compliance with

# the License. You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

#

# This file is sourced when running various Spark programs.

# Copy it as spark-env.sh and edit that to configure Spark for your site.

# Options read when launching programs locally with

# ./bin/run-example or ./bin/spark-submit

# - HADOOP\_CONF\_DIR, to point Spark towards Hadoop configuration files

# - SPARK\_LOCAL\_IP, to set the IP address Spark binds to on this node

# - SPARK\_PUBLIC\_DNS, to set the public dns name of the driver program

# - SPARK\_CLASSPATH, default classpath entries to append

# Options read by executors and drivers running inside the cluster

# - SPARK\_LOCAL\_IP, to set the IP address Spark binds to on this node

# - SPARK\_PUBLIC\_DNS, to set the public DNS name of the driver program

# - SPARK\_CLASSPATH, default classpath entries to append

# - SPARK\_LOCAL\_DIRS, storage directories to use on this node for shuffle and RDD data

# - MESOS\_NATIVE\_JAVA\_LIBRARY, to point to your libmesos.so if you use Mesos

# Options read in YARN client mode

# - HADOOP\_CONF\_DIR, to point Spark towards Hadoop configuration files

# - SPARK\_EXECUTOR\_INSTANCES, Number of executors to start (Default: 2)

# - SPARK\_EXECUTOR\_CORES, Number of cores for the executors (Default: 1).

# - SPARK\_EXECUTOR\_MEMORY, Memory per Executor (e.g. 1000M, 2G) (Default: 1G)

# - SPARK\_DRIVER\_MEMORY, Memory for Driver (e.g. 1000M, 2G) (Default: 1G)

# Options for the daemons used in the standalone deploy mode

# - SPARK\_MASTER\_HOST, to bind the master to a different IP address or hostname

# - SPARK\_MASTER\_PORT / SPARK\_MASTER\_WEBUI\_PORT, to use non-default ports for the master

# - SPARK\_MASTER\_OPTS, to set config properties only for the master (e.g. "-Dx=y")

# - SPARK\_WORKER\_CORES, to set the number of cores to use on this machine

# - SPARK\_WORKER\_MEMORY, to set how much total memory workers have to give executors (e.g. 1000m, 2g)

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

# - SPARK\_WORKER\_INSTANCES, to set the number of worker processes per node

# - SPARK\_WORKER\_DIR, to set the working directory of worker processes

# - SPARK\_WORKER\_OPTS, to set config properties only for the worker (e.g. "-Dx=y")

# - SPARK\_DAEMON\_MEMORY, to allocate to the master, worker and history server themselves (default: 1g).

# - SPARK\_HISTORY\_OPTS, to set config properties only for the history server (e.g. "-Dx=y")

# - SPARK\_SHUFFLE\_OPTS, to set config properties only for the external shuffle service (e.g. "-Dx=y")

# - SPARK\_DAEMON\_JAVA\_OPTS, to set config properties for all daemons (e.g. "-Dx=y")

# - SPARK\_PUBLIC\_DNS, to set the public dns name of the master or workers

# Generic options for the daemons used in the standalone deploy mode

# - SPARK\_CONF\_DIR Alternate conf dir. (Default: ${SPARK\_HOME}/conf)

# - SPARK\_LOG\_DIR Where log files are stored. (Default: ${SPARK\_HOME}/logs)

# - SPARK\_PID\_DIR Where the pid file is stored. (Default: /tmp)

# - SPARK\_IDENT\_STRING A string representing this instance of spark. (Default: $USER)

# - SPARK\_NICENESS The scheduling priority for daemons. (Default: 0)

#java的环境变量

export JAVA\_HOME=/usr/local/jdk1.8.0\_111

#spark主节点的ip

export SPARK\_MASTER\_IP=172.16.200.81

#spark主节点的端口号

export SPARK\_MASTER\_PORT=7077

**2.4.2 配置slaves**

vim slaves

#

# Licensed to the Apache Software Foundation (ASF) under one or more

# contributor license agreements. See the NOTICE file distributed with

# this work for additional information regarding copyright ownership.

# The ASF licenses this file to You under the Apache License, Version 2.0

# (the "License"); you may not use this file except in compliance with

# the License. You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

#

# A Spark Worker will be started on each of the machines listed below.

172.16.200.81

172.16.200.82

172.16.200.83

172.16.200.84

**3. 安装scala**

cd /usr/loca/src/

tar zxvf scala-2.12.1.tgz

mv scala-2.12.1 /usr/local

**3.1 配置scala环境变量（只master安装）**

vim /etc/profile

添加如下信息

######### scala ##########

export SCALA\_HOME=/usr/local/scala-2.12.1

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

**3.2 刷新配置文件：**

source /etc/profile

**4. 启动程序**

**4.1 启动hadoop**

**4.1.1 格式化namenode**

hadoop namenode -format

**4.1.2 master启动hadoop**

cd /usr/local/hadoop-2.7.3/sbin

./start-all.sh

提示

start-all.sh //启动master和slaves

stop-all.sh //停止master和slaves

查看进程 （master）

[root@spark-master sbin]# jps

8961 NodeManager

8327 DataNode

8503 SecondaryNameNode

8187 NameNode

8670 ResourceManager

9102 Jps

[root@spark-master sbin]#

查看进程 （slave）

[root@spark-slave01 ~]# jps

4289 NodeManager

4439 Jps

4175 DataNode

[root@spark-slave01 ~]#

slave01、slve02、slave03显示相同

**4.2 启动spark**

**4.1.2 master启动hadoop**

cd /usr/local/spark-2.0.2/sbin

./start-all.sh

提示

start-all.sh //启动master和slaves

stop-all.sh //停止master和slaves