**Impala-Kudu安装文档**

目录

[1. 准备工作：系统环境搭建 2](#_Toc490766745)

[1.1. 设置ssh无密码登陆（server端到所有节点） 2](#_Toc490766746)

[1.2. 关闭防火墙和SELinux 2](#_Toc490766747)

[1.4 所有节点配置NTP服务 3](#_Toc490766748)

[2. 安装jdk 3](#_Toc490766749)

[3. 安装原生hadoop 2.6.0 4](#_Toc490766750)

[4. server节点安装mysql5.6 7](#_Toc490766751)

[4.1. 首先需要卸载centos自带的mysql 7](#_Toc490766752)

[4.2. 安装mysql5.6 7](#_Toc490766753)

[5. 安装hive 9](#_Toc490766754)

[5.1. 下载tar包： 9](#_Toc490766755)

[5.2. 解压tar包到指定位置， 9](#_Toc490766756)

[5.3. 配置环境变量 9](#_Toc490766757)

[5.4. 修改配置文件 9](#_Toc490766758)

[5.4.1. 修改hive-site.xml配置文件 9](#_Toc490766759)

[5.4.2. 修改hive-log4j.properties配置文件 10](#_Toc490766760)

[5.4.3. 修改hive-env.sh配置文件 11](#_Toc490766761)

[5.5. 将mysql的驱动放置到hive的lib下 11](#_Toc490766762)

[5.6. 在mysql中创建hive用户及数据库 11](#_Toc490766763)

[5.7. 替换jline的jar包 11](#_Toc490766764)

[5.8. 启动hive 12](#_Toc490766765)

[6. 安装和配置kudu、impala 12](#_Toc490766766)

[6.1. 安装kudu、impala 12](#_Toc490766767)

[6.1.1. 在线安装kudu、impala 12](#_Toc490766768)

[6.1.2. yum离线安装kudu、impala 13](#_Toc490766769)

[6.1.3. rpm离线安装kudu、impala 17](#_Toc490766770)

[6.1.4. 简洁版rpm离线安装impala 19](#_Toc490766771)

[6.2. 配置并启动kudu、impala 21](#_Toc490766772)

[6.2.1. 配置并启动kudu 21](#_Toc490766773)

[6.2.2. 配置并启动impala 22](#_Toc490766774)

# 准备工作：系统环境搭建

这里以3台机器（centos 6.5）为示例安装，

172.16.22.53

172.16.22.54

172.16.22.55

其中172.16.22.53节点安装kudu-master。

以下操作均用root用户操作。

## 配置hosts文件

执行命令 vi /etc/hosts 打开hosts文件，在此文件中追加如下内容

172.16.22.53 iZn7m989dgmfgm3ttb83y1Z

172.16.22.54 iZn7mgishthhlt5sdw1oelZ

172.16.22.55 iZn7m110rd41tfy2sdrnk9Z

说明：以上蓝色部分，每行的内容中前半段为ip地址，后半段为主机名(hostname)

## 设置ssh无密码登陆（server端到所有节点）

在主节点上执行ssh-keygen -t rsa一路回车，生成无密码的密钥对。

用ssh-copy-id把公钥复制到远程主机上（注意本机也要复制）

例如cdr-yl-01主机到其它3节点设置无密码登录

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@172.16.22.53

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@172.16.22.54

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@172.16.22.55

## 关闭防火墙和SELinux

关闭防火墙：

service iptables stop （临时关闭）

chkconfig iptables off （重启后生效）

关闭SELINUX：

setenforce 0 （临时生效）

修改 /etc/selinux/config 下的 SELINUX=disabled （重启后永久生效）

vi /etc/selinux/config

将SELINUX 的值改为 disabled

## 1.4 所有节点配置NTP服务

集群中所有主机必须保持时间同步，如果时间相差较大会引起各种问题。 具体思路如下：

master(172.16.22.53)节点作为ntp服务器与外界对时中心同步时间。

这里选择172.16.22.53节点为时钟同步服务器，其他节点为客户端同步时间到该节点。

1. 安装ntp:

yum -y install ntp

1. 修改172.16.22.53上的配置文件:

执行 vi /etc/ntp.conf  ，加入如下内容（绿色注释部分不要加入）

restrict default ignore    #默认不允许修改或者查询ntp,并且不接收特殊封包

restrict 127.0.0.1        #给于本机所有权限

restrict 172.16.22.0 mask 255.255.255.0 notrap nomodify  #给于局域网机的机器有同步时间的权限

server  172.16.22.53   prefer  # local clock prefer代表优先使用此ip做同步

driftfile /var/lib/ntp/drift

server 127.127.1.0 #当所有服务器都不能使用时，使用本机作为同步服务器

fudge   127.127.1.0 stratum 10

1. 在172.16.22.54和172.16.22.55节点上修改配置文件，

执行 vi /etc/ntp.conf  加入如下内容

server 172.16.22.53 prefer

driftfile /var/lib/ntp/drift

fudge 127.127.1.0 stratum 10

1. 启动ntp, 所有节点执行一下命令

service ntpd start

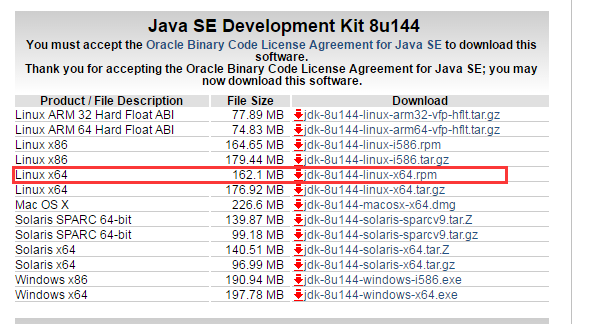
chkconfig ntpd on

# 安装jdk

下载rpm包

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

下载对应的版本



将rpm包上传到各个节点上，然后rpm -ivh 包名 进行安装

# 安装原生hadoop 2.6.0

1. 下载tar包，链接如下 ：<http://archive.apache.org/dist/hadoop/core/hadoop-2.6.0/hadoop-2.6.0.tar.gz>
2. 在tar包所在的目录，执行

tar -xzvf hadoop-2.6.0.tar.gz -C /usr/lib/

将tar包解压到指定位置

1. 进入解压到配置文件的目录，修改配置文件

cd /usr/lib/hadoop-2.6.0/etc/hadoop

1. 修改hadoop-env.sh

在此文件的末尾加入

export JAVA\_HOME=/usr/java/jdk1.8.0\_144

注意：jdk所在路径以及SSH端口号需视自己的环境而定，

如果ssh端口号不为22，则需要加入以下参数（端口号视自己环境而定）：

export HADOOP\_SSH\_OPTS="-p 49622"

1. 修改yarn-env.sh

同hadoop-env.sh，在文件的末尾加入 export JAVA\_HOME=/usr/java/jdk1.8.0\_144

注意：jdk所在路径需视自己的环境而定

1. 修改core-site.xml，在<configuration>标签间加入如下内容

注意：红色字部分为主节点ip，需要根据具体情况而定

<property>

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

<!--路径视环境而定，一定要配置 系统默认的缓存文件夹在每次重启后会被删除-->

<value>/data/hadoop/tmp</value>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://172.16.22.53:9000</value>

</property>

1. 修改hdfs-site.xml，在<configuration>标签间加入如下内容

注意：红色字部分为主节点ip，需要根据具体情况而定

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property>

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

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

</property>

<property>

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

<value>file:/data/hadoop/dfs/data</value> <!--数据路径，尽量不要用根目录-->

</property>

<property> <!--这个属性节点是为了防止后面eclipse存在拒绝读写设置的-->

<name>dfs.permissions</name>

<value>false</value>

</property>

1. 修改mapred-site.xml, 在<configuration>标签间加入如下内容

注意：红色字部分为主节点ip，需要根据具体情况而定

<property>

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

<value>yarn</value>

</property>

<property>

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

<value>172.16.22.53:10020</value>

</property>

<property>

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

<value>172.16.22.53:19888</value>

</property>

<property>

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

<value>/history</value>

</property>

<property>

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

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

</property>

<property>

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

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

</property>

<property>

<name>mapreduce.map.log.level</name>

<value>DEBUG</value>

</property>

<property>

<name>mapreduce.reduce.log.level</name>

<value>DEBUG</value>

</property>

1. 修改yarn-site.xml, 在<configuration>标签间加入如下内容

注意：红色字部分为主节点ip，需要根据具体情况而定

<property>

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

<value>yarn</value>

</property>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

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

<value>172.16.22.53</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8032</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>${yarn.resourcemanager.hostname}:8031</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8033</value>

</property>

<property>

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

<value>${yarn.resourcemanager.hostname}:8088</value>

</property>

<property>

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

<value>true</value>

</property>

<property>

<name>yarn.log.server.url</name>

<value>http://${yarn.resourcemanager.hostname}:19888/jobhistory/logs</value>

</property>

1. 修改slaves文件，删掉里面的所有内容，将各个从节点的ip或hosts映射加入，这里设置3个从节点，将3个节点的ip加入到slaves文件中：

172.16.22.53

172.16.22.54

172.16.22.55

1. 修改环境变量，在/etc/profile 文件或~/.bash\_profile文件的末尾加入以下内容，注意HADOOP\_HOME的值需根据环境而定：

export HADOOP\_HOME=/usr/lib/hadoop-2.6.0

export PATH=.:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:${PATH}

export HADOOP\_MAPRED\_HOME=${HADOOP\_HOME}

export HADOOP\_COMMON\_HOME=${HADOOP\_HOME}

export HADOOP\_HDFS\_HOME=${HADOOP\_HOME}

export YARN\_HOME=${HADOOP\_HOME}

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=${HADOOP\_HOME}/lib/natvie

加入完成后，执行 source /etc/profile 或者 source ~/.bash\_profile

1. 将配置好的整个hadoop-2.6.0目录复制到各个从节点上

scp -r $HADOOP\_HOME 172.16.22.54:/usr/lib

scp -r $HADOOP\_HOME 172.16.22.55:/usr/lib

1. 格式化hdfs

执行命令 hdfs namenode -format

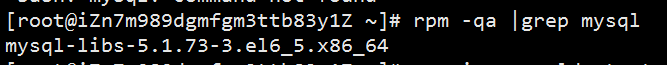
1. 启动hadoop

执行命令 $HADOOP\_HOME/sbin/start-all.sh

# server节点安装mysql5.6

## 首先需要卸载centos自带的mysql

1. 执行 rpm -qa |grep mysql



1. 然后执行：rpm -e --nodeps mysql-libs-5.1.73-3.el6\_5.x86\_64 # 注意 –nodeps后面的名字为上一步查询到的名字

注意：如果是centos7，需要卸载maridb，卸载方法类似以上步骤，先执行

rpm -qa |grep mariadb，centos7中可能不止一个包， 然后执行 rpm -e --nodeps + 包名 将其全部卸载

## 安装mysql5.6

在server(172.16.22.53)节点安装mysql

1. 首先在172.16.22.53上新建目录 /usr/rpms

mkdir /usr/rpms

1. 由于server节点不能上网，所以需要先登录可以上网的202.98.194.89，执行以下命令以获取安装包

wget <https://dev.mysql.com/get/Downloads/MySQL-5.6/MySQL-5.6.37-1.el6.x86_64.rpm-bundle.tar>

1. 在202.98.194.89 上通过scp命令将安装包复制到server(172.16.22.53)节点

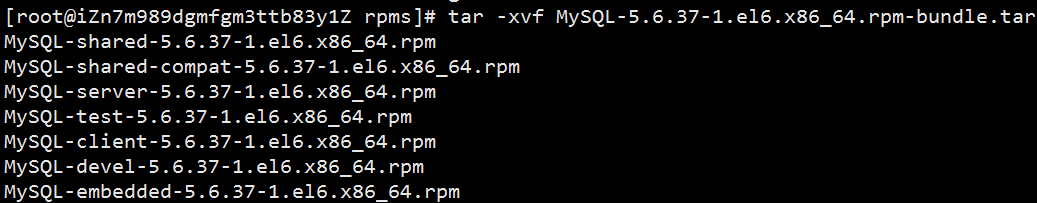
scp MySQL-5.6.37-1.el6.x86\_64.rpm-bundle.tar 172.16.22.53:/ usr/rpms

1. 进入server(172.16.22.53)节点的/usr/rpms目录

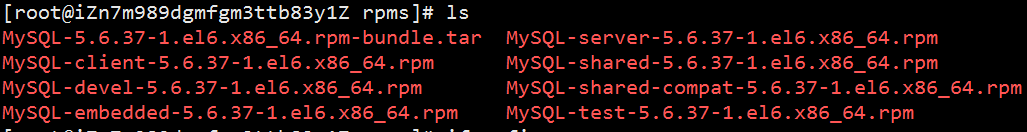
cd mkdir /usr/rpms

1. 执行如下命令解压安装包

tar -xvf MySQL-5.6.37-1.el6.x86\_64.rpm-bundle.tar



1. 解压后通过ls命令查看解压出的rpm包



1. 我们只需安装MySQL-client-5.6.15-1.el6.x86\_64.rpm、

MySQL-devel-5MySQL-devel-5.6.15-1.el6.x86\_64.rpm、MySQL-server-5.6.15-1.el6.x86\_64.rpm这三个rpm包，依次执行如下命令进行安装：

rpm -ivh MySQL-server-5.6.37-1.el6.x86\_64.rpm

rpm -ivh MySQL-devel-5.6.37-1.el6.x86\_64.rpm

rpm -ivh MySQL-client-5.6.37-1.el6.x86\_64.rpm

1. 修改配置文件位置，执行如下命令：

cp /usr/share/mysql/my-default.cnf /etc/my.cnf

1. 初始化MySQL及设置密码，启动mysql服务，执行如下2条命令：

/usr/bin/mysql\_install\_db

service mysql start

1. 查看root账号密码，执行以下命令：

cat /root/.mysql\_secret



1. 登录mysql, 输入 mysql -uroot -pRW2RvZ9dmKetxg5E 登录之后，执行

SET PASSWORD = PASSWORD('root');命令将密码设置为root, 如下图所示：



**如果登录时报出以下错误：**



1. 执行 mysql -h172.16.22.53 -uroot -pRW2RvZ9dmKetxg5E进行登录其中172.16.22.53为本地ip
2. 执行  grant all privileges on \*.\* to [root@localhost](mailto:joe@localhost) identified by 'root';
3. 最后 执行  flush privileges; 既可解决问题

# 安装hive

impala使用的是hive的元数据，所以hive也应进行安装，只需在启动某一个节点安装既可

## 下载tar包：

<http://mirror.bit.edu.cn/apache/hive/hive-1.2.2/apache-hive-1.2.2-bin.tar.gz>

## 解压tar包到指定位置，

执行 tar -xzvf apache-hive-1.2.2-bin.tar.gz -C /usr/lib/

## 配置环境变量

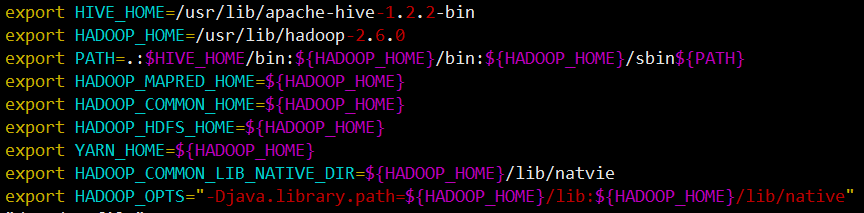
修改/etc/profile文件或者~/.bash\_profile文件，加入以下内容：

export HIVE\_HOME=/usr/lib/apache-hive-1.2.2-bin

PATH中加入

:$HIVE\_HOME/bin

如下图所示：



加入完成后，执行 source /etc/profile 或者 source ~/.bash\_profile

## 修改配置文件

### 修改hive-site.xml配置文件

执行命令：

cd $HIVE\_HOME/conf

cp hive-default.xml.template hive-site.xml

vi hive-site.xml # 红色字体部分需要根据自己的环境进行配置

删除<configuration>标签下的所有内容，并加入以下内容

<!--增加这两行，在使用Mysql时 -->

<property>

<name>system:java.io.tmpdir</name>

<value>/usr/lib/apache-hive-1.2.2-bin/iotmp</value>

<description/>

</property>

<property>

<name>system:user.name</name>

<value>username</value>

<description />

</property>

<!--增加这两行，在使用Mysql时 -->

<property>

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

<value>jdbc:mysql://172.22.16.53:3306/hive?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

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

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

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

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

<value>hive</value>

<description>username to use against metastore database</description>

</property>

<property>

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

<value>hive</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.server2.thrift.port</name>

<value>10000</value>

</property>

<property>

<name>hive.server2.thrift.bind.host</name>

<value>172.22.16.53</value>

</property>

### 修改hive-log4j.properties配置文件

执行

cp hive-log4j.properties.template hive-log4j.properties

vi hive-log4j.properties

将hive.log.dir的值修改为/var/log/hive

### 修改hive-env.sh配置文件

执行

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

vi hive-env.sh

配置Hive配置文件的路径和Hadoop的安装路径

export HIVE\_CONF\_DIR=/usr/lib/apache-hive-1.2.2-bin/conf

HADOOP\_HOME=/usr/lib/hadoop-2.6.0

## 将mysql的驱动放置到hive的lib下

下载<https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.43.tar.gz>

解压到hive的lib目录下，执行如下命令

tar -xzvf mysql-connector-java-5.1.43.tar.gz

mv mysql-connector-java-5.1.43/mysql-connector-java-5.1.43-bin.jar $HIVE\_HOME/lib

## 在mysql中创建hive用户及数据库

mysql -uroot –proot 登录mysql

mysql> create user 'hive' identified by 'hive';

mysql> grant all privileges on \*.\* to 'hive' with grant option;

mysql> grant all privileges on \*.\* to 'hive' @'localhost' identified by 'hive' with grant option;

mysql> flush privileges;

## 替换jline的jar包

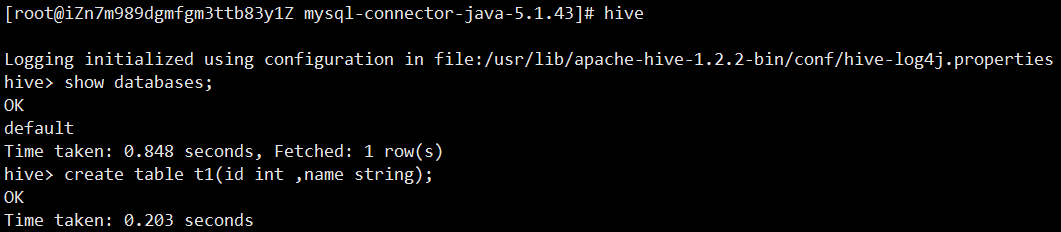
将/usr/lib/hadoop-2.6.0/share/hadoop/yarn/lib 下的jline-0.9.94.jar 移除，并将/usr/lib/apache-hive-1.2.1/lib 下的jline-2.12.jar 复制到该文件夹下，执行如下命令

rm -f $HADOOP\_HOME/share/hadoop/yarn/lib/jline-0.9.94.jar

cp $HIVE\_HOME/lib/jline-2.12.jar $HADOOP\_HOME/share/hadoop/yarn/lib/

## 启动hive

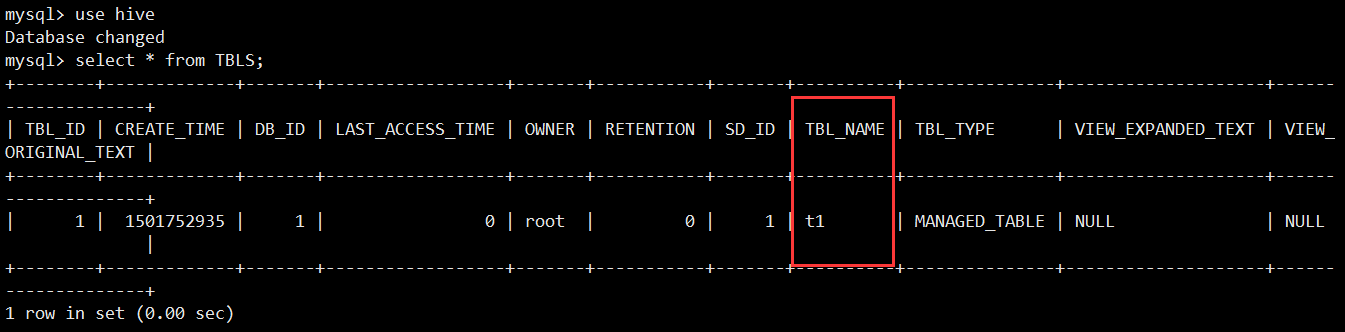
1. 命令行输入hive 即可进入hive的命令行
2. 创建表，以验证hive配置正确



1. 登录mysql 查看创建的表

执行 use hive;

select \* from TBLS; 查看创建的table



# 安装和配置kudu、impala

## 安装kudu、impala

kudu和impala的安装分为在线安装和离线安装两种。在线安装要求所有节点都可以联网且网络状况较好，否则容易出现不可预知的问题。

### 在线安装kudu、impala

1. 首先配置kudu和impala的yum源，执行如下命令：

cd /etc/yum.repos.d/

如果系统为CentOS6，执行如下语句

wget https://archive.cloudera.com/cdh5/redhat/6/x86\_64/cdh/cloudera-cdh5.repo

如果系统为CentOS7,则执行如下语句

wget http://archive.cloudera.com/cdh5/redhat/7/x86\_64/cdh/cloudera-cdh5.repo

1. 执行如下命令

yum install kudu kudu-master kudu-tserver kudu-client kudu-client-devel -y

yum install impala impala-catalog impala-state-store impala-server impala-shell -y

yum install impala-debuginfo impala-udf-devel -y

### rpm离线安装kudu、impala

#### rpm离线安装kudu

1. 下载kudu的rpm安装包

a．用于centos6的连接地址：

<https://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/RPMS/x86_64/>

b．用于centos7的连接地址

<https://archive.cloudera.com/cdh5/redhat/7/x86_64/cdh/5/RPMS/x86_64/>

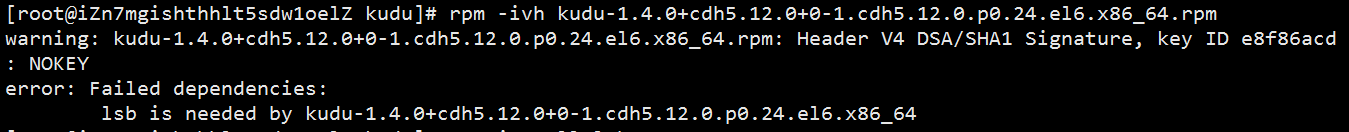
注： 共有6个kudu的rpm包，都需要下载，如下图所示



1. 使用sftp 或者winSCP 等工具将下载的rpm包上传到linux
2. cd到rpm包所在的位置，安装rpm包

rpm -ivh kudu\* 进行安装

如果报以下错误



先执行 yum install lsb 安装依赖

然后执行rpm -ivh kudu\*

#### rpm离线安装impala

注意：由于cdh和centos的版本更迭，浅蓝色部分的版本号可能会与文档中有所不同

1. 下载impala的rpm安装包

a．用于centos6的连接地址：

<http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/RPMS/x86_64/>

b．用于centos7的连接地址

<http://archive.cloudera.com/cdh5/redhat/7/x86_64/cdh/5/RPMS/x86_64/>

下载的如下的rpm包：

impala-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-catalog-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-debuginfo-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-server-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-shell-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-state-store-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-udf-devel-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

bigtop-jsvc-0.6.0+cdh5.12.0+869-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

hadoop-0.20-mapreduce-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hadoop-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hadoop-client-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hadoop-hdfs-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hadoop-libhdfs-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hadoop-mapreduce-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hadoop-yarn-2.6.0+cdh5.12.0+2512-1.cdh5.12.0.p0.38.el6.x86\_64.rpm

hbase-1.2.0+cdh5.12.0+358-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

zookeeper-3.4.5+cdh5.12.0+112-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

还需要依赖包

下载链接 <http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/RPMS/noarch/>

parquet-1.5.0+cdh5.12.0+186-1.cdh5.12.0.p0.37.el7.noarch.rpm

parquet-format-2.1.0+cdh5.12.0+17-1.cdh5.12.0.p0.37.el6.noarch.rpm

solr-4.10.3+cdh5.12.0+513-1.cdh5.12.0.p0.37.el6.noarch.rpm

hive-1.1.0+cdh5.12.0+1164-1.cdh5.12.0.p0.37.el6.noarch.rpm

hive-jdbc-1.1.0+cdh5.12.0+1164-1.cdh5.12.0.p0.37.el6.noarch.rpm

bigtop-tomcat-0.7.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.noarch.rpm

bigtop-utils-0.7.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.noarch.rpm

avro-libs-1.7.6+cdh5.12.0+132-1.cdh5.12.0.p0.37.el6.noarch.rpm

以及其他版本的依赖包

地址：http://archive.cloudera.com/cdh5/redhat/6/x86\_64/cdh/5.12/RPMS/noarch/下载

sentry-1.5.1+cdh5.12.1+329-1.cdh5.12.1.p0.3.el6.noarch.rpm

1. 下载centos所需的rpm包

如果配置有yum源或者服务器可以连接外网，则可以直接执行yum install nc lsb python-setuptools -y

否则需要到<https://pkgs.org/>网站上查找、下载并通过rpm命令进行安装（centos7系统，可能没有相应的RPM，这时需要使用yum安装）

nc-1.84-24.el6.x86\_64.rpm

redhat-lsb-4.0-7.el6.centos.x86\_64.rpm

redhat-lsb-compat-4.0-7.el6.centos.x86\_64.rpm

python-setuptools-0.6.10-3.el6.noarch.rpm

1. 使用sftp 或者winSCP 等工具将下载的rpm包上传到linux
2. cd到rpm包所在的位置，安装rpm包(每个节点都需要安装)

按照以下顺序进行安装：

rpm -ivh bigtop-utils-\*.noarch.rpm

rpm -ivh bigtop-jsvc-\*.rpm

rpm -ivh bigtop-tomcat-\*.rpm

rpm -ivh avro-libs-\*.noarch.rpm

rpm -ivh zookeeper-\*.rpm

rpm -ivh parquet-\*.noarch.rpm --nodeps --force

rpm -ivh hadoop-2\*.rpm

rpm -ivh parquet-format-\*.rpm

rpm -ivh hadoop-hdfs-\*.rpm

rpm -ivh hadoop-libhdfs-\*.rpm

rpm -ivh hadoop-yarn-\*.rpm

rpm -ivh hadoop-mapreduce-\*.rpm

rpm -ivh hadoop-0.20-mapreduce-\*.rpm

rpm -ivh hadoop-client-\*.rpm

rpm -ivh hbase-\*.rpm

rpm -ivh hive-jdbc-\*.rpm

rpm -ivh sentry-\*.rpm --nodeps --force

rpm -ivh solr-\*.rpm

rpm -ivh hive-\*.rpm

rpm -ivh parquet-\*.rpm

rpm -ivh impala-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

rpm -ivh impala-\*.rpm

rpm -ivh impala-debuginfo-\*.rpm

rpm -ivh impala-server-\*.rpm

rpm -ivh impala-shell-\*.rpm

rpm -ivh impala-state-store-\*.rpm

rpm -ivh impala-udf-devel-\*.rpm

### 简洁版rpm离线安装impala

kudu的安装方式同6.1.3.1中的安装方法，这里不多赘述，只阐述impala的安装

#### 安装步骤

1) 下载impala的rpm安装包

a．用于centos6的连接地址：

<http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/RPMS/x86_64/>

b．用于centos7的连接地址

<http://archive.cloudera.com/cdh5/redhat/7/x86_64/cdh/5/RPMS/x86_64/>

下载的如下的rpm包(centos 6 为例)：

impala-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-catalog-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-debuginfo-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-server-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-shell-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-state-store-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

impala-udf-devel-2.9.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.x86\_64.rpm

还需要一个依赖包

下载链接 <http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/RPMS/noarch/>

bigtop-utils-0.7.0+cdh5.12.0+0-1.cdh5.12.0.p0.37.el6.noarch.rpm

2) 按照如下顺序安装上一步中下载的rpm包（每个节点都需要安装）：

rpm -ivh bigtop-utils-\*.rpm

rpm -ivh impala-2.\* --nodeps --force

rpm -ivh impala-catalog\*.rpm

rpm -ivh impala-debuginfo-\*.rpm

rpm -ivh impala-server-\*.rpm

rpm -ivh impala-shell-\*.rpm

rpm -ivh impala-state-store-\*.rpm

rpm -ivh impala-udf-devel-\*.rpm

3) 获取impala依赖包impala2.9\_depend\_jars.tar

文件位置：

云上贵州 172.16.22.53

/data/1/yum\_repo/impala/impala2.9dependjars/ impala2.9\_depend\_jars.tar

4) 将impala依赖包impala2.9\_depend\_jars.tar 上传到某个节点，执行

tar -xzvf impala2.9\_depend\_jars.tar -C /usr/lib/impala/lib

5) 通过scp命令，将完整的impala依赖复制到其他节点，我们这里总共3个节点，需要复制到其他两个节点上：

scp -r /usr/lib/impala/lib 172.16.22.54:/usr/lib/impala/lib

scp -r /usr/lib/impala/lib 172.16.22.55:/usr/lib/impala/lib

#### 简洁安装的原理及jar包提取方法:

此小节说明一下简洁安装的原理及jar包提取方法，仅作为记录，不属于安装步骤，此小节不属于安装步骤，可跳过不看。

1. **原理阐述：**

以上所述的三种安装方法都需要安装很多的依赖，包含了很多的CDH自带组件，如hadoop、hive、hbase、zookeeper等等，其实，impala只是需要这些框架提供的jar包及其他.so等文件。

我们只需将这些jar包、.so文件提取出来，再在其他机器上安装impala时，就可以强制安装impala的rpm包，再将提取出来的jar包、.so文件放到impala的lib目录下，这样就可以正常启动impala，而无需再安装那么多的依赖，使系统看起来更干净、整洁

1. **jar包、.so文件提取方法**

impala的lib目录下为其所需文件库，大部分为jar包，也有.so文件等，其中使用软文件为其所依赖的其他组件的jar包，这里可以选择将整个目录内的文件全部提取，可以只提取缺失的文件。

在已经安装impala并可以正常运行的机器上执行一下操作：

* 创建一个目录，用于存储提取出来的jar包、.so文件：mkdir lib
* 进入此目录：cd lib
* **全部提取**：

执行 cp /usr/lib/impala/lib/\* . 就可以将全部的依赖文件提取到此目录。

* **只提取缺失部分**

新装impala的机器上，通过xshell软件，ssh进入/usr/lib/impala/lib目录下

执行ll查看，出现闪烁的软连接即为缺失的文件，将这些闪烁的文件名记录下来

一个文件一个文件的执行cp命令复制到当前目录既可。这里提供一个提取的脚本，可以将缺失的文件提取到当前目录



## 配置并启动kudu、impala

### 配置并启动kudu

安装完成之后，在/etc/kudu/conf目录下有两个文件：master.gflagfile和tserver.gflagfile两个文件中的目录可以根据自己的习惯来制定，但在启动kudu之前一定要事先创建好，否则无法正常启动。

这里我们使用172.16.22.53节点作为kudu-master，172.16.22.53，172.16.22.54，172.16.22.55作为kudu-tserver，所以172.16.22.53节点需要修改master.gflagfile和tserver.gflagfile，而172.16.22.54，172.16.22.55 只需修改tserver.gflagfile

1. master.gflagfile的配置修改:

--fromenv=rpc\_bind\_addresses

--fromenv=log\_dir

--fs\_wal\_dir=/data/lib/kudu/master

--fs\_data\_dirs=/data/lib/kudu/master

--default\_num\_replicas=1 # 此参数可以调整备份数量，默认为3

1. tserver.gflagfile 的配置修改:

--fromenv=rpc\_bind\_addresses

--fromenv=log\_dir

--fs\_wal\_dir=/data/lib/kudu/tserver

--fs\_data\_dirs=/data/lib/kudu/tserver

--tserver\_master\_addrs=172.16.22.53:7051 # 此参数指定master

注意，这里的--tserver\_master\_addrs指明了集群中master的地址，指向同一个master的tserver形成了一个kudu集群

1. 创建master.gflagfile和tserver.gflagfile文件中指定的目录，并将所有者更改为kudu，执行如下命令：

mkdir -p /data/lib/kudu/master /data/lib/kudu/tserver

chown -R kudu:kudu /data/lib/kudu/

1. 修改 /etc/security/limits.d/90-nproc.conf 文件，解除kudu用户的线程限制

vi /etc/security/limits.d/90-nproc.conf

加入如下两行内容：

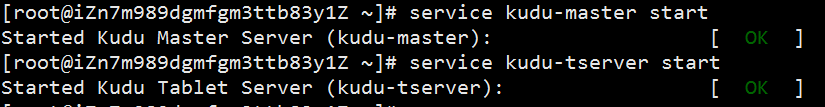
kudu soft nproc unlimited

impala soft nproc unlimited

1. 启动kudu

master节点（172.16.22.53） 执行 service kudu-master start

所有tserver节点执行 service kudu-tserver start



**可能遇到的问题：**

启动失败，查看日志发现kudu的启动有一个前提，NTP服务必须启动处以同步状态

启动NTP服务的命令为：service ntpd start

服务启动后，需要5-10钟来完成同步。判断是否已经完成同步可以使用命令：ntpstat，如果输出为：

synchronised to NTP server (120.25.108.11) at stratum 3

   time correct to within 114 ms

   polling server every 1024 s

表示同步已经完成。

### 配置并启动impala

在安装好之后、就可以发现在 /etc/default  和  /etc/impala 生成配置文件

       /etc/default 路径下会新生成两个文件 bigtop-utils  impala

1. 在bigtop-utils 文件中配置 Java路径即可：

vi /etc/default/bigtop-utils

export JAVA\_HOME=/usr/java/jdk1.8.0\_144

1. 修改impala的配置文件，文件位置： /etc/default/impala
2. 打开文件

vi /etc/default/impala

1. 修改以下两项，将ip改为主机节点IP

IMPALA\_CATALOG\_SERVICE\_HOST=172.16.22.53

IMPALA\_STATE\_STORE\_HOST=172.16.22.53

1. 在IMPALA\_SERVER\_ARGS 中加入kudu-master的ip，多个master用逗号分隔

IMPALA\_SERVER\_ARGS=" \

-log\_dir=${IMPALA\_LOG\_DIR} \

-catalog\_service\_host=${IMPALA\_CATALOG\_SERVICE\_HOST} \

-state\_store\_port=${IMPALA\_STATE\_STORE\_PORT} \

-use\_statestore \

-kudu\_master\_hosts=172.16.22.53:7051 \

-state\_store\_host=${IMPALA\_STATE\_STORE\_HOST} \

-be\_port=${IMPALA\_BACKEND\_PORT}"

1. 文件末尾加入mysql驱动jar包的地址：

export MYSQL\_CONNECTOR\_JAR=/usr/lib/apache-hive-1.2.2-bin/lib/mysql-connector-java-5.1.43-bin.jar

1. 拷贝并配置[Hadoop](http://lib.csdn.net/base/hadoop)的配置文件core-site.xml 和 hdfs-site.xml
2. 将core-site.xml 和 hdfs-site.xml文件拷贝到 /etc/impala/conf.dist 下面，

执行以下语句

cp $HADOOP\_HOME/etc/hadoop/core-site.xml /etc/impala/conf.dist

cp $HADOOP\_HOME/etc/hadoop/hdfs-site.xml /etc/impala/conf.dist

1. 修改/etc/impala/conf.dist下的core-site.xml 和 hdfs-site.xml 文件，做如下操作：

* 在core-site.xml文件中增加如下配置

<!-- impala -->

<**property**>

<**name**>dfs.client.read.shortcircuit</**name**>

<**value**>true</**value**>

</**property**>

<**property**>

<**name**>dfs.client.read.shortcircuit.skip.checksum</**name**>

<**value**>false</**value**>

</**property**>

<**property**>

<**name**>dfs.datanode.hdfs-blocks-metadata.enabled</name>

<**value**>true</**value**>

</**property**>

* 在hdfs-site.xml 文件中增加如下配置（在configuration中）

<!--impala-->

<**property**>

<**name**>dfs.datanode.hdfs-blocks-metadata.enabled</**name**>

<**value**>true</**value**>

</**property**>

<**property**>

<**name**>dfs.block.local-path-access.user</**name**>

<**value**>impala</**value**>

</**property**>

<**property**>

<**name**>dfs.client.file-block-storage-locations.timeout.millis</name>

<**value**>60000</**value**>

</**property**>

1. 将Hive的配置文件hive-site.xml拷贝到 /etc/impala/conf.dist

cp $HIVE\_HOME/conf/hive-site.xml /etc/impala/conf.dist

1. 将配置文件复制到其他节点，执行如下语句

scp /etc/default/bigtop-utils 172.16.22.54:/etc/default/

scp /etc/default/bigtop-utils 172.16.22.55:/etc/default/

scp /etc/default/impala 172.16.22.54:/etc/default/

scp /etc/default/impala 172.16.22.55:/etc/default/

scp /etc/impala/conf.dist/\* 172.16.22.54:/etc/impala/conf.dist/

scp /etc/impala/conf.dist/\* 172.16.22.55:/etc/impala/conf.dist/

1. 接下来就可以启动服务了，主节点执行以下3行命令

service impala-state-store start

service impala-catalog start

service impala-server start

从节点执行

service impala-server start

执行impala-shell，看是否可以进入impala的shell命令行，如果可以进入，查看是否已经连接到主节点

# Zookeeper配置

**说明：**

HBase依赖zookeeper，Hadoop、Spark等高可用配置也依赖zookeeper，HBase自身的安装包中就包含有zookeeper，在单机上部署HBase时，可以直接使用HBase自带的zookeeper，而部署集群时，推荐使用外部的zookeeper

**zookeeper安装步骤如下：**

Zookeeper服务集群规模一般不小于三个节点，要求各服务之间系统时间要保持一致。

1. 下载安装包

下载地址：

<http://archive.apache.org/dist/zookeeper/zookeeper-3.4.5/zookeeper-3.4.5.tar.gz>

2.  在172.16.22.53的/usr/lib目录下，解压缩zookeeper

执行命令 tar -zvxf zookeeper-3.4.5.tar.gz -C /usr/lib

3.  设置环境变量

打开/etc/profile文件(vi /etc/profile)，内容如下：

export ZOOKEEPER\_HOME=/usr/lib/zookeeper-3.4.5

export PATH=.:$HADOOP\_HOME/bin:$ZOOKEEPER\_HOME/bin:$JAVA\_HOME/bin:$PATH

**注：修改完后profile记得执行source /etc/profile**

4. 在解压后的zookeeper的目录下进入conf目录修改配置文件

更名操作：mv zoo\_sample.cfg zoo.cfg

5. 编辑zoo.cfg (vi zoo.cfg)

(1)文件如下：(红色字体部分为修改内容，内容需根据自己的环境而定)

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/data/zookeeper/data

# the port at which the clients will connect

clientPort=2181

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

server.0=172.16.22.53:2888:3888

server.1=172.16.22.54:2888:3888

server.2=172.16.22.55:2888:3888

 (2)说明：(仅作为理解内容，不属于安装步骤)

server.0=172.16.22.53:2888:3888

server.1=172.16.22.54:2888:3888

server.2=172.16.22.55:2888:3888

这三行为配置zookeeper集群的机器（172.16.22.53、172.16.22.54、172.16.22.55）分别用server.0和server.1、server.2标识，2888和3888为端口号（zookeeper集群包含一个leader(领导)和多个fllower(随从)，启动zookeeper集群时会随机分配端口号，分配的端口号为2888的为leader，端口号为3888的是fllower）

6.  创建文件夹，执行命令mkdir -p /data/zookeeper/data

7.  在 /data/zookeeper/data目录下，创建文件myid，值为0  (0用来标识s1这台机器的zookeeper )

**到此为止 172.16.22.53上的配置就已经完成；接下来配置172.16.22.54和172.16.22.55.**

1. 在172.16.22.54和172.16.22.55上创建文件夹：

执行命令 mkdir -p /data/zookeeper/data

9.   把zookeeper目录复制到172.16.22.54和172.16.22.55中:

scp -r /data/zookeeper/data 172.16.22.54:/data/zookeeper/data

scp -r /data/zookeeper/data 172.16.22.55:/data/zookeeper/data

10.    设置环境变量

打开/etc/profile文件(vi /etc/profile)，内容如下：

export ZOOKEEPER\_HOME=/usr/lib/zookeeper-3.4.5

export PATH=.:$HADOOP\_HOME/bin:$ZOOKEEPER\_HOME/bin:$JAVA\_HOME/bin:$PATH

**修改完成后记得执行** source /etc/profile

11. 把172.16.22.54中/data/zookeeper/data目录下myid中的值改为1，172.16.22.55中相应的myid中的值改为2

12. 启动，在三个节点上分别执行命令zkServer.sh  start

13. 检验，在三个节点上分别执行命令zkServer.sh  status

启动zookeeper：zkServer.sh  start

进入zookeeper：zkCli.sh

# HBase配置

## 下载安装包

下载地址：<http://archive.apache.org/dist/hbase/1.2.6/hbase-1.2.6-bin.tar.gz>

## 解压缩安装包

执行如下命令，将压缩包解压缩到/usr/lib目录下

tar -xzvf hbase-1.2.6-bin.tar.gz -C /usr/lib

## 设置环境变量

vi /etc/profile

内容如下：

export JAVA\_HOME=/usr/java/jdk1.8.0\_144

export HADOOP\_HOME=/usr/lib/hadoop-2.6.0

export ZOOKEEPER\_HOME=/usr/lib/zookeeper-3.4.5

export HBASE\_HOME=/usr/lib/hbase-1.2.6-bin

export PATH=.:$HADOOP\_HOME/bin:$ZOOKEEPER\_HOME/bin:$HBASE\_HOME/bin:$JAVA\_HOME/bin:$PATH

**注：修改完成后记得执行** source /etc/profile

## 修改配置文件

1. 修改文件$HBASE\_HOME/conf/hbase-env.sh

执行 vi $HBASE\_HOME/conf/hbase-env.sh 打开此文件，加入如下内容

export JAVA\_HOME=/usr/java/jdk1.8.0\_144

export HBASE\_MANAGES\_ZK=false

1. 修改hbase-site.xml文件

执行 vi $HBASE\_HOME/conf/hbase-site.xml 打开文件，在 <configuration> 标签中间加入如下内容

注：红色字体部分需根据自己的环境进行调整

<property>

<name>hbase.rootdir</name>

<value>hdfs://172.16.22.53:9000/hbase</value>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

</property>

<property>

<!-- 指定master的地址 -->

<name>hbase.master</name>

<value>172.16.22.53:60000</value>

</property>

<property>

<name>hbase.zookeeper.quorum</name>

<value> 172.16.22.53, 172.16.22.54, 172.16.22.55</value>

</property>

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

## 修改regionservers文件

在regionservers文件中，加入region server的hostname或ip

172.16.22.53

172.16.22.54

172.16.22.55

## 复制s1中的hbase文件夹到s2、s3上

scp -r /usr/lib/hbase-1.2.6-bin 172.16.22.54:/usr/lib

scp -r /usr/lib/hbase-1.2.6-bin 172.16.22.55:/usr/lib

## 启动HBase

执行命令: $HBASE\_HOME/bin/start-hbase.sh

注意：启动hbase之前要先启动zookeeper 和 hadoop，确保hadoop可写入数据。

# Spark配置

## 下载安装包

下载地址：

<http://mirrors.hust.edu.cn/apache/spark/spark-2.2.1/spark-2.2.1-bin-hadoop2.6.tgz>

## 解压缩安装包

执行命令 tar -xzvf spark-2.2.1-bin-hadoop2.6.tgz -C /usr/lib

## 配置环境变量

/etc/profile文件中加入SPARK\_HOME变量，同时PATH中加入对应内容

注： 红色字部分需要根据自己的环境而定

export HADOOP\_HOME=/usr/lib/hadoop-2.6.0

export SPARK\_HOME=/usr/lib/spark-2.2.1

export PATH=.:${HADOOP\_HOME}/bin:${SPARK\_HOME}/bin:${HADOOP\_HOME}/sbin:${PATH}

export HADOOP\_MAPRED\_HOME=${HADOOP\_HOME}

export HADOOP\_COMMON\_HOME=${HADOOP\_HOME}

export HADOOP\_HDFS\_HOME=${HADOOP\_HOME}

export YARN\_HOME=${HADOOP\_HOME}

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=${HADOOP\_HOME}/lib/natvie

加入完成之后，执行 source /etc/profile

## 修改配置文件

1. 修改spark-env.sh

依次执行如下命令

cd $SPARK\_HOME/conf

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

vi spark-env.sh

在文件末尾加入如下内容

# master的ip，也可填写主机名

export SPARK\_MASTER\_IP=172.16.22.53

# 指明spark的home目录，spark以此寻找对应的配置文件

export SPARK\_HOME=/usr/lib/spark2.2.1

# 每个work的核心数，最好不要高于当前节点的cpu总核心数

export SPARK\_WORKER\_CORES=32

# 每个work的内存大小，千万不能高于当前节点的总内存

export SPARK\_WORKER\_MEMORY=64G

# 每个节点上的work实例数目

export SPARK\_WORKER\_INSTANCES=1

# 如果ssh的端口不是22的时候需要配置此项，如果ssh的端口是22，此项可以忽略

export SPARK\_SSH\_OPTS="-p 49622"

# 指定JAVA\_HOME

export JAVA\_HOME=/usr/java/jdk1.8.0\_141

# 指定web端口号，默认为8080

export SPARK\_MASTER\_WEBUI\_PORT=8080

1. 修改slaves文件
2. 首先执行如下命令

mv slaves.template slaves

1. 在文件中加入所有从节点的ip或者主机名

172.16.22.53

172.16.22.54

172.16.22.55

1. 修改log4j.properties

首先执行 mv log4j.properties.template log4j.properties

将 log4j.rootCategory=INFO, console 修改为 log4j.rootCategory=WARN, console

1. 修改spark-defaults.conf 文件

注： 此文件用于集群性能优化，不配置此文件spark仍可以正常启动及执行任务，只是在提交任务的时候需要在SparkConf中进行参数设置以进行调优

首先执行 mv spark-defaults.conf.template spark-defaults.conf

加入如下参数(红色字体部分需要根据自己环境进行调整)：

spark.master=spark://172.16.22.53:7077

spark.eventLog.dir=hdfs://172.16.22.53:9000/user/spark/applicationHistory

spark.eventLog.enabled=true

spark.yarn.historyServer.address=172.16.22.53:18082

spark.logConf=false

spark.executor.memory=1g

spark.python.executor.memory=1g

spark.executor.extraClassPath=/usr/lib/spark-2.2.0/extraJars/\*

spark.driver.extraClassPath=/usr/lib/spark-2.2.0/extraJars/\*

spark.executor.userClassPathFirst=false

spark.shuffle.consolidateFiles=false

spark.shuffle.spill=True

spark.shuffle.spill.compress=True

spark.shuffle.compress=true

spark.shuffle.file.buffer=128k

spark.reducer.maxSizeInFlight=48

spark.broadcast.compress=true

spark.rdd.compress=false

spark.io.compression.codec=snappy

spark.broadcast.blockSize=4096

spark.storage.memoryMapThreshold=8192

spark.executor.heartbeatInterval=10000

spark.task.cpus=1

spark.task.maxFailures=4

spark.scheduler.mode=FIFO

spark.authenticate=false

spark.acls.enable=false

spark.localExecution.enabled=false

spark.mesos.coarse=false

## 启动spark

执行语句 $SPARK\_HOME/sbin/start-all.sh

## 通过SparkUI查看各个节点状态

浏览器访问主节点ip 端口为8080 ( spark-env.sh 文件中配置的此端口)

