2019 年第二届“全国大学生大数据技能竞赛”

选拔赛

主办单位：中国大数据技术与应用联盟

技术平台：北京红亚华宇科技有限公司

第二届全国大学生大数据竞赛选手使用手册

1、引言

1.1 编写目的

选手操作手册的目的是明确本系统的使用操作，帮助选手理解及操作问系统。

1.2 读者对象

大数据竞赛的参赛者、具备一定的大数据基础。

1.3 环境要求

浏览器版本：建议使用谷歌浏览器达到最佳呈现效果，不建议使用 IE。

2、操作说明

2.1 登陆方式

1.通过比赛模式进入比赛

选手可直接通过网址访问 http://10.10.30.2。

图 1 登陆页

2.2 比赛进行中

比赛开始后，用户可通过上述方式进入比赛操作页，如下图：

图 3 操作页

1.虚拟机信息

比赛提供虚拟机，选手可通过给出的虚拟机信息，在终端通过 ssh 的方式登陆，并在此进行比赛的相关操作。

如图：上述提供了多个虚拟机信息，选手可通过展开操作查看所有的虚拟机。

图 4 虚拟机展开

点击眼睛的按钮，查看密码，然后点击密码后的复制即可将密码复制下来。

虚拟机信息登陆到终端，具体的题目操作步骤详见--“大数据竞赛题目操作手册”。

```
[E:\~]$ ssh 10.143.0.18


Connecting to 10.143.0.18:22...
Connection established.
To escape to local shell, press Ctrl+Alt+].

WARNING! The remote SSH server rejected X11 forwarding request.
Last login: Thu Apr  4 04:58:03 2019 from 172.31.0.1
[root@master--clone--lpya1pfxnndl4rqwmaqz ~]#
```

图 5 ssh 访问虚拟机

2.通知栏

比赛的得分信息及奖惩信息、通知都会再次展示。

3.当前成绩

（1）排名：根据得分多少实时排名；

（2）比赛得分：当前所有完成步骤的总得分；

（3）当前步骤用时：根据用户的答题时间计时，直到题目解锁，计时器清零重新开始。

4.比赛信息

包括用户的队伍、队员及比赛开始时间、时长、规则的简介。

5.排行榜

比赛所有队伍的排行情况，hover 展示每个队伍的得分、惩罚分、奖励分。

6.比赛题目

*注：选手验证题目的时间，会影响得分，在虚拟机做完题目需及时提交。

1）点击问号的图标会展示当前步骤的详细描述；

2）用户在将一个题目步骤后，需在此对应的步骤上点击立即验证，系统会判定当前步骤

是否通过，如果通过可继续进行下一步骤，即状态变为绿色，否则需要继续修改直至完成。完成后每个条件的得分会及时同步，当步骤完成时会将用时总得分一起同步；

图 6 比赛题目

3）当在条件中出现以下图示符号，表明这两个条件可以二选一完成，只要完成一个即可得分，同时可进入下一个条件，如果两个题目同时完成，则会取两个条件的最高分加到总分里。

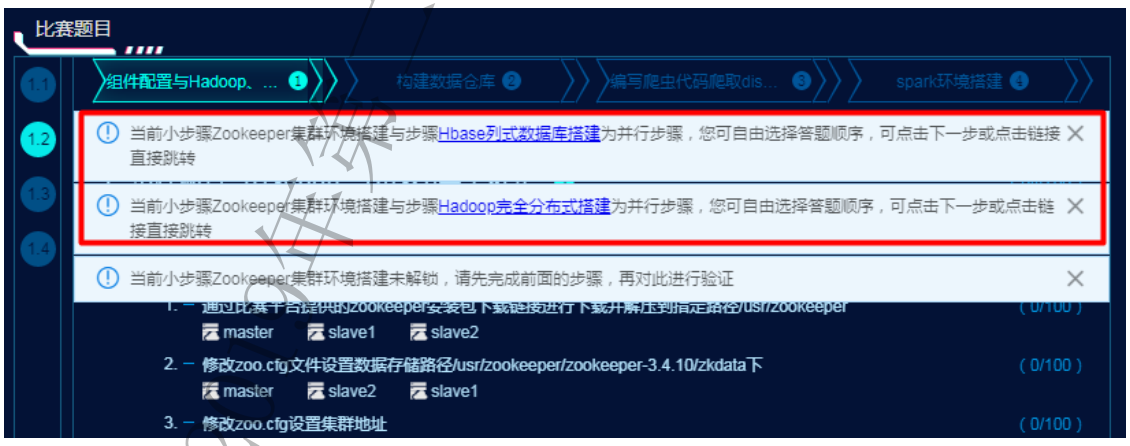


图 7 比赛题目

2.3 比赛已结束

1、比赛结束后，可在比赛页面查看成绩。

图 8 比赛已结束

大数据竞赛本地源介绍

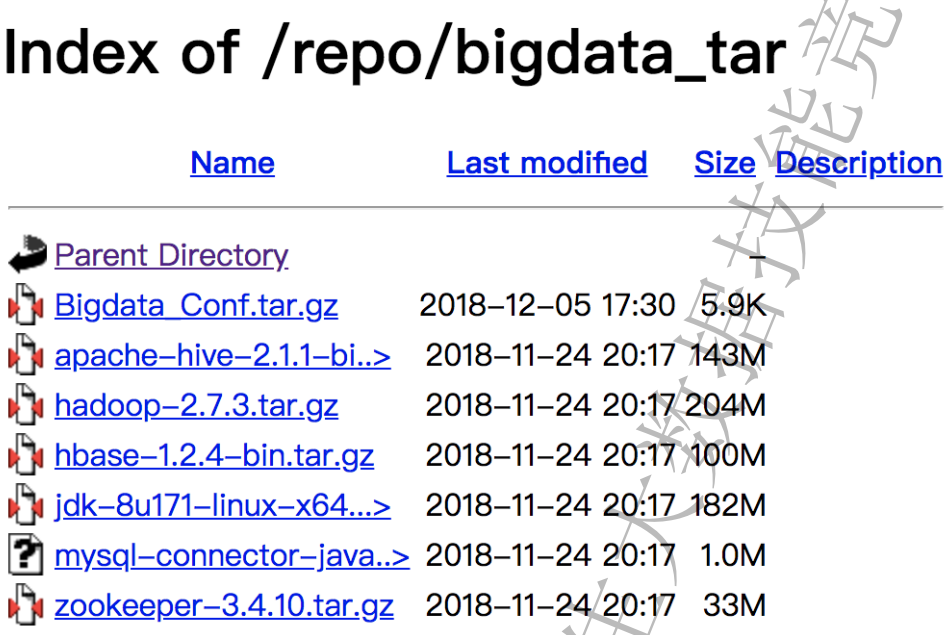
本次比赛提供相应的软件压缩包以及相关的 xml 配置文件、数据集，其地址为 http://10.10.30.2:8000/bigdata/。具体如下：

Index of /repo

Name	Last modified	Size	Description
Parent Directory		-	
Anaconda3-5.2.0-Wind..>	2018-12-05 14:53	631M	
Windows_Python.zip	2018-11-30 14:11	239M	
bigdata_tar/	2018-12-05 17:30	-	
centos7/	2018-11-24 20:07	-	
epel/	2018-11-24 20:10	-	
mysql-5.7/	2018-11-24 20:31	-	
repofile/	2018-12-05 16:57	-	
train_format.csv	2018-12-05 18:00	732M	

相关安装软件、xml 配置文件 Bigdata_Conf.tar.gz 的路径为 http://10.10.30.2:8000/bigdata/bigdata_tar/，如下：

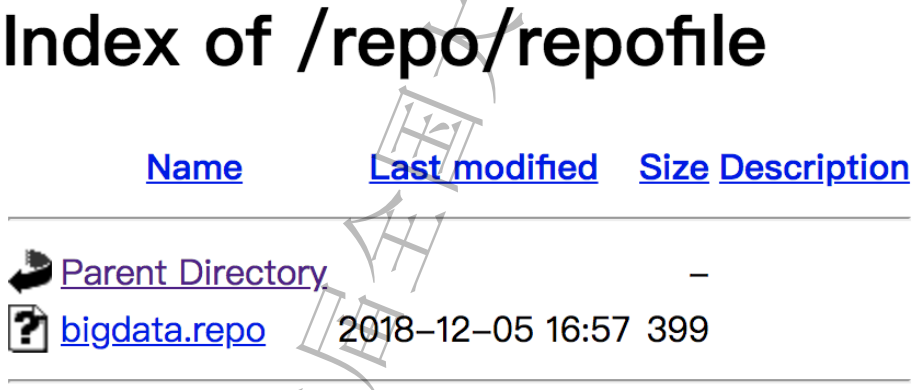
注意：配置文件需要自己修改。



Index of /repo/bigdata_tar

Name	Last modified	Size	Description
Parent Directory		-	
Bigdata_Conf.tar.gz	2018-12-05 17:30	5.9K	
apache-hive-2.1.1-bi..>	2018-11-24 20:17	143M	
hadoop-2.7.3.tar.gz	2018-11-24 20:17	204M	
hbase-1.2.4-bin.tar.gz	2018-11-24 20:17	100M	
jdk-8u171-linux-x64...>	2018-11-24 20:17	182M	
mysql-connector-java..>	2018-11-24 20:17	1.0M	
zookeeper-3.4.10.tar.gz	2018-11-24 20:17	33M	

MySQL Server 的 yum 源路径为 http://10.10.30.2:8000/bigdata/repofile，这里直接提供 repo 文件为 bigdata.repo，直接将其下载至/etc/yum/repos.d/即可，然后就可以下载 MySQL Server。



Index of /repo/repofile

Name	Last modified	Size	Description
Parent Directory		-	
bigdata.repo	2018-12-05 16:57	399	

大数据竞赛题目操作手册

基础部分

1、基础搭建

本次集群搭建共有三个节点，包括一个主节点 master，和两个从节点 slave1 和 slave2。具体操作如下：

1.1 使用连接工具连接比赛节点，更改本地源

1.使用本地 Windows 操作机提供的 Xshell 或 MobaXterm 连接比赛平台所提供的 master，slave1，slave2 三台机器，并按照比赛平台提供的 linux 用户和密码进行登录，登录成功后开始进行接下来的比赛。

注意连接工具没有在桌面上，点击（或者键入）左下角“windows”，即可看到连接工具。

同时可以使用以下命令进行修改主机名：

- hostnamectl set-hostname master（在 master 执行）
- 立即生效：bash

```
[root@10-137-0-97 ~]# hostnamectl set-hostname master
[root@10-137-0-97 ~]# bash
[root@master ~]#
```

主机名已修改

同理修改 slave1 和 slave2 的主机名。

- hostnamectl set-hostname slave1（在 slave1 执行）
- hostnamectl set-hostname slave2（在 slave2 执行）

2.配置本地源。通过比赛平台提供源文件下载路径，将本地源文件下载到/etc/yum.repos.d/目录下（三台机器都执行）。

- 发信号给 yum 进程：pkill -9 yum
- 进入 yum 源配置文件：cd /etc/yum.repos.d
- 删除所有文件：rm -rf *
- 下载 yum 源：wget http://10.10.30.2:8000/bigdata/repofile/bigdata.repo
- 清除 YUM 缓存：yum clean all

1.2 配置 hosts 文件（三台机器都执行）

可以通过 ifconfig 命令进行查看机器的 ip 地址或者直接输入比赛平台提供的 ip 地址。

查看节点地址之后将三个节点的 ip 地址以及其对应的名称写进 hosts 文件。这里我们设置为 master、slave1、slave2。注意保存退出。

- vim /etc/hosts（三台机器都执行）

```
[root@master ~]# vi /etc/hosts

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.15.104 master
192.168.15.127 slave1
192.168.15.124 slave2
~
~
~
~
~
```

输入各节点相应IP

1.3 关闭防火墙（三台机器都执行）

- 关闭防火墙：systemctl stop firewalld
- 查看状态：systemctl status firewalld

```
[root@master ~]# systemctl stop firewalld
[root@master ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Fri 2018-09-28 20:15:05 CST; 25s ago
     Docs: man:firewalld(1)
 Main PID: 652 (code=exited, status=0/SUCCESS)

Sep 28 19:48:28 master systemd[1]: Starting firewalld - dynamic firewall daemon...
Sep 28 19:48:29 master systemd[1]: Started firewalld - dynamic firewall daemon.
Sep 28 20:15:04 master systemd[1]: Stopping firewalld - dynamic firewall daemon...
Sep 28 20:15:05 master systemd[1]: Stopped firewalld - dynamic firewall daemon.
[root@master ~]#
```

关闭防火墙查看状态

防火墙已关闭

注意：当环境重置之后，防火墙会自动开启。

1.4 时间同步

- 选择时区：tzselect（三台执行）

```
[root@master ~]# tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent or ocean.
 1) Africa
 2) Americas
 3) Antarctica
 4) Arctic Ocean
 5) Asia
 6) Atlantic Ocean
 7) Australia
 8) Europe
 9) Indian Ocean
10) Pacific Ocean
11) none - I want to specify the time zone using the Posix TZ format.
#? 5
Please select a country.
 1) Afghanistan           18) Israel                35) Palestine
 2) Armenia               19) Japan                 36) Philippines
 3) Azerbaijan            20) Jordan                37) Qatar
 4) Bahrain               21) Kazakhstan            38) Russia
 5) Bangladesh            22) Korea (North)         39) Saudi Arabia
 6) Bhutan                23) Korea (South)         40) Singapore
 7) Brunei                24) Kuwait                41) Sri Lanka
 8) Cambodia              25) Kyrgyzstan            42) Syria
 9) China                 26) Laos                  43) Taiwan
10) Cyprus                27) Lebanon               44) Tajikistan
11) East Timor            28) Macau                 45) Thailand
12) Georgia               29) Malaysia              46) Turkmenistan
13) Hong Kong             30) Mongolia              47) United Arab Emirates
14) India                 31) Myanmar (Burma)       48) Uzbekistan
15) Indonesia             32) Nepal                 49) Vietnam
16) Iran                  33) Oman                  50) Yemen
17) Iraq                  34) Pakistan
#? 9
Please select one of the following time zone regions.
1) Beijing Time
2) Xinjiang Time
#? 1

The following information has been given:

        China
        Beijing Time

Therefore TZ='Asia/Shanghai' will be used.
Local time is now:      Fri Sep 28 20:33:01 CST 2018.
Universal Time is now:  Fri Sep 28 12:33:01 UTC 2018.
Is the above information OK?
1) Yes
2) No
#? 1
```

选择时区

选择亚洲

中国

北京时间

覆盖时间

下载 ntp（三台机器都执行）

- yum install -y ntp

```
[root@master ~]# yum install -y ntp
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package ntp.x86_64 0:4.2.6p5-28.el7.centos will be installed
--> Processing Dependency: ntpdate = 4.2.6p5-28.el7.centos for package: ntp-4.2.6p5-28.el7.centos.x86
_64
--> Processing Dependency: libcrypto.so.10(OPENSSL_1.0.2)(64bit) for package: ntp-4.2.6p5-28.el7.cent
os.x86_64
--> Processing Dependency: libopts.so.25()(64bit) for package: ntp-4.2.6p5-28.el7.centos.x86_64
--> Running transaction check
---> Package autogen-libopts.x86_64 0:5.18-5.el7 will be installed
---> Package ntpdate.x86_64 0:4.2.6p5-28.el7.centos will be installed
---> Package openssl-libs.x86_64 1:1.0.1e-60.el7 will be updated
--> Processing Dependency: openssl-libs(x86-64) = 1:1.0.1e-60.el7 for package: 1:openssl-1.0.1e-60.el
7.x86_64
---> Package openssl-libs.x86_64 1:1.0.2k-12.el7 will be an update
```

master 作为 ntp 服务器，修改 ntp 配置文件。（**master 上执行，注意空格问题**）

- vim /etc/ntp.conf

```
server 127.127.1.0                # local clock
fudge 127.127.1.0 stratum 10      #stratum 设置为其它值也是可以的，其范围为 0~15
```

修改后注意保存退出。

重启 ntp 服务(master 上执行)

- /bin/systemctl restart ntpd.service

其他机器进行同步（在 slave1，slave2 中执行）

- ntpdate master

```
[root@slave2 ~]# ntpdate  master
28 Sep 20:51:27 ntpdate[2338]: adjust time server 192.168.15.104 offset 0.201592 sec
[root@slave2 ~]#
```

同步master时间

1.5 配置 ssh 免密

1.在 master 上执行如下命令生成公私密钥：**（注意 master 上执行）**

- ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa

2.然后将 master 公钥 id_dsa 复制到 slave1 进行公钥认证。

- ssh-copy-id -i /root/.ssh/id_dsa.pub slave1

```
[root@master ~]# ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
Generating public/private dsa key pair.
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
ba:46:80:6a:0f:66:bf:84:40:5b:e4:4e:12:f1:23:90 root@master
The key's randomart image is:
+--[ DSA 1024]----+
|.+..             |
|E =              |
| + B             |
|. O o            |
|.o . .  S        |
|o=.   ..         |
|+.+. ..          |
|  .o  ..         |
|   ....          |
+-----------------+
[root@master ~]# ssh-copy-id -i /root/.ssh/id_dsa.pub slave1
The authenticity of host 'slave1 (10.137.0.102)' can't be established.
ECDSA key fingerprint is fc:cc:c7:52:4e:58:ba:dd:f6:3e:1e:44:ae:39:fa:56.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
root@slave1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'slave1'"
and check to make sure that only the key(s) you wanted were added.

[root@master ~]# ssh slave1
Last login: Wed Dec  5 19:01:46 2018 from 172.31.0.1
[root@slave1 ~]#
```

输入slave1密码

成功免密登陆slave1

- 退出连接：exit

同理可以将 master 公钥复制到 master、slave2 进行公钥认证。

- ssh-copy-id -i /root/.ssh/id_dsa.pub master
- ssh-copy-id -i /root/.ssh/id_dsa.pub slave2

注意：以上只是 master 到 slave1、slave2 的免密。如果想配置其他免密，其命令类似。

2、安装 JDK

以下操作为先在 master 上操作，然后远程复制到 slave1 和 slave2。参赛选手仅供参考。

首先在根目录下建立工作路径/usr/java。

- mkdir -p /usr/java

进入创建的 java 工作路径。

- cd /usr/java

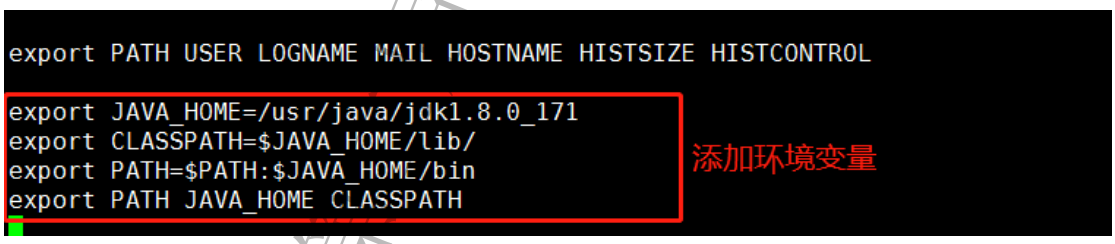
下载 java 安装包(master 上执行)。

- wget http://10.10.30.2:8000/bigdata/bigdata_tar/jdk-8u171-linux-x64.tar.gz（在 master 执行）
- tar -zxvf jdk-8u171-linux-x64.tar.gz -C /usr/java/（在 master 执行）

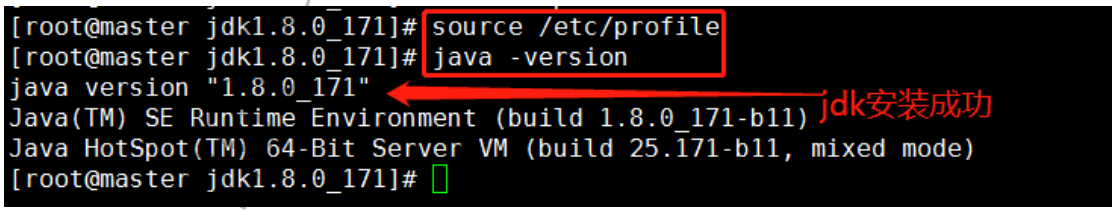
- 修改环境变量：vim /etc/profile（在 master 执行）

添加内容如下：

```
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
```



- 生效环境变量：source /etc/profile（在 master 执行）
- 查看 java 版本：java -version（在 master 执行）



在 master 中将 JDK 复制到 slave1 和 slave2 中。（在 master 执行）

- scp -r /usr/java root@slave1:/usr/
- scp -r /usr/java root@slave2:/usr/

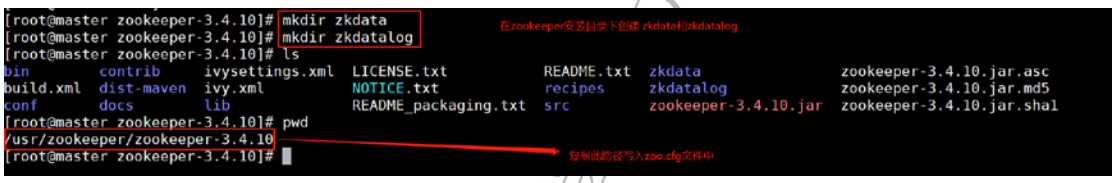
注意：此时需要去 slave1 和 slave2 上配置 java 的环境变量，并使环境变量生效。

3、安装 zookeeper

1.创建工作路径/usr/zookeeper，下载相应软件，解压至工作路径。

在 zookeeper 的目录中，创建配置中所需的 zkdata 和 zkdatalog 两个文件夹。（在 master 执行）

- cd /usr/zookeeper/zookeeper-3.4.10
- mkdir zkdata
- mkdir zkdatalog



2.配置文件 zoo.cfg

进入 zookeeper 配置文件夹 conf，将 zoo_sample.cfg 文件拷贝一份命名为 zoo.cfg，Zookeeper 在启动时会找这个文件作为默认配置文件。

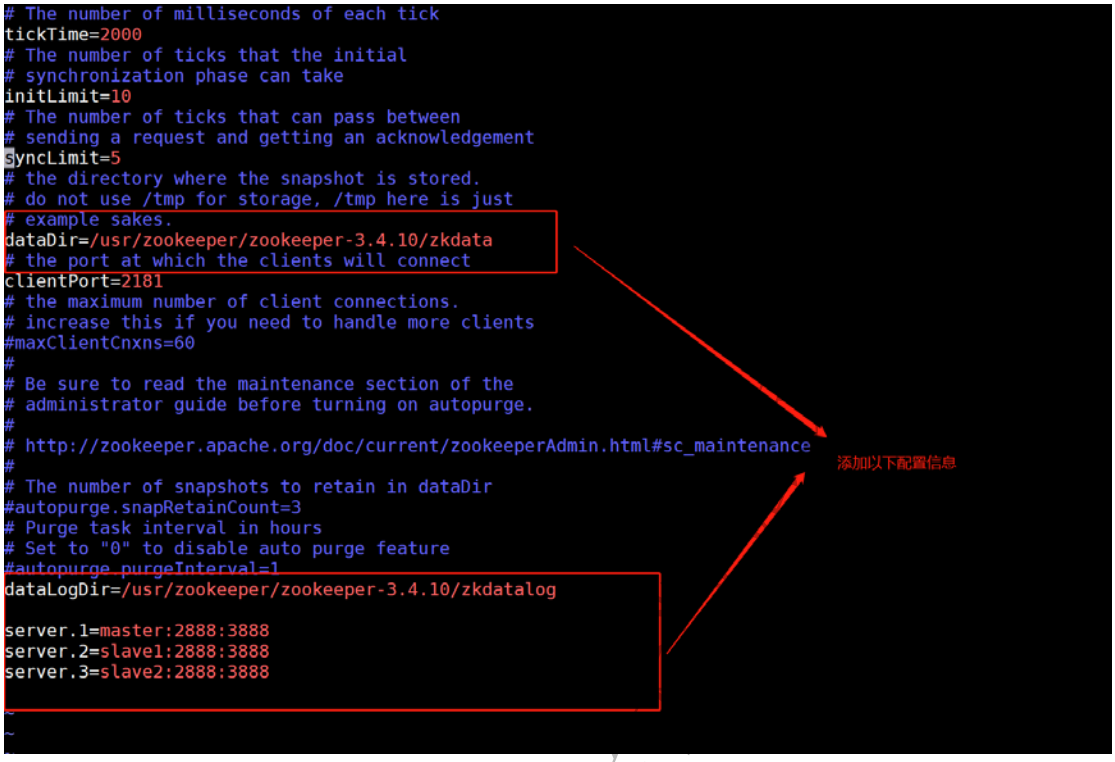
- cd /usr/zookeeper/zookeeper-3.4.10/conf/
- mv zoo_sample.cfg zoo.cfg

对 zoo.cfg 文件配置如下：（在 master 执行）

- vim zoo.cfg

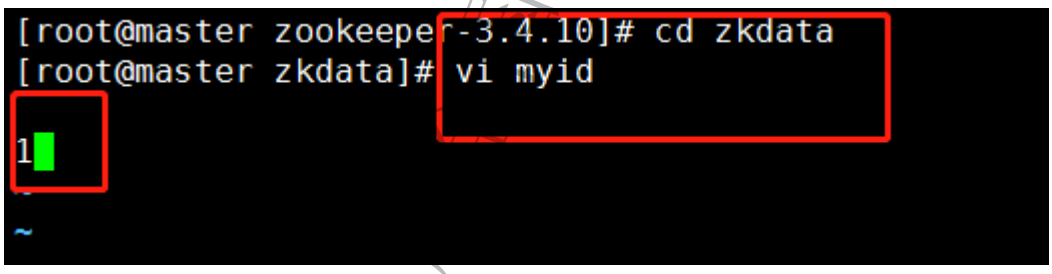
修改如下：

```
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/usr/zookeeper/zookeeper-3.4.10/zkdata
clientPort=2181
dataLogDir=/usr/zookeeper/zookeeper-3.4.10/zkdatalog
server.1=master:2888:3888
server.2=slave1:2888:3888
server.3=slave2:2888:3888
```



3.进入 zkdata 文件夹，创建文件 myid，用于表示是几号服务器。master 主机中，设置服务器 id 为 1。（集群中设置 master 为 1 号服务器，slave1 为 2 号服务器，slave2 为 3 号服务器）

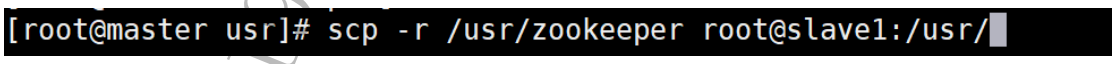
- cd /usr/zookeeper/zookeeper-3.4.10/zkdata
- vim myid



4.远程复制分发安装文件。

以上已经在主节点 master 上配置完成 ZooKeeper，现在可以将该配置好的安装文件远程拷贝到集群中的各个结点对应的目录下：（在 master 执行）

- scp -r /usr/zookeeper root@slave1:/usr/
- scp -r /usr/zookeeper root@slave2:/usr/



5.设置 myid。在我们配置的 dataDir 指定的目录下面，创建一个 myid 文件，里面内容为一个数字，用来标识当前主机，conf/zoo.cfg 文件中配置的 server.X 中 X 为什么数字，则 myid

文件中就输入这个数字。(在 slave1 和 slave2 中执行)

- cd /usr/zookeeper/zookeeper-3.4.10/zkdata
- vim myid

实验中设置 slave1 中为 2；

```
[root@slave1 ~]# cd /usr/zookeeper/zookeeper-3.4.10/zkdata
[root@slave1 zkdata]# ls
myid
[root@slave1 zkdata]# vi myid

2
~
~
```

slave1中为2

slave2 中为 3：

```
[root@slave2 ~]# cd /usr/zookeeper/zookeeper-3.4.10/zkdata
[root@slave2 zkdata]# vi myid

3
~
```

6.修改/etc/profile 文件，配置 zookeeper 环境变量。（三台机器都执行）

- vi /etc/profile

```
#set zookeeper environment
export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10
PATH=$PATH:$ZOOKEEPER_HOME/bin
```

```
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH

export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10
PATH=$PATH:$ZOOKEEPER_HOME/bin

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}
```

ZK环境变量配置

- 生效环境变量：source /etc/profile

7.启动 ZooKeeper 集群。在 ZooKeeper 集群的每个结点上，执行启动 ZooKeeper 服务的脚本。**注意在 zookeeper 目录下：（三台机器都执行）**

- 回到上一层：cd ..
- 开启服务：bin/zkServer.sh start

- 查看状态：bin/zkServer.sh status

通过上面状态查询结果可见，一个节点是 Leader，其余的结点是 Follower。

至此，zookeeper 安装成功。

4、安装 hadoop

以下步骤逻辑仅供选手参考：即 hadoop 安装同样在主节点 master 上进行配置操作，然后将文件复制到到子节点 slave1 和 slave2。

4.1 解压安装包，配置环境变量

1. 创建工作路径/usr/hadoop，下载相应软件，解压至工作路径。

2.添加 hadoop 环境变量（三台机器执行）

- vim /etc/profile

添加如下内容：

```
# # #HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin
```

```
# # # JAVA
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
# # # HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}


if [ -x /usr/bin/id ]; then
    if [ -z "$EUID" ]; then
        # ksh workaround
        EUID=`/usr/bin/id -u`
        UID=`/usr/bin/id -ru`
    fi
    USER="`/usr/bin/id -un`"
    LOGNAME=$USER
    MAIL="/var/spool/mail/$USER"
fi
:wq
```

配置hadoop环境变量

使用以下命令使 profile 生效：

- source /etc/profile

4.2 配置 hadoop 各组件

hadoop 的各个组件的都是使用 XML 进行配置，这些文件存放在 hadoop 的 etc/hadoop 目录下。

Common组件	core-site.xml
HDFS组件	hdfs-site.xml
MapReduce组件	mapred-site.xml
YARN组件	yarn-site.xml

1.编辑 hadoop-env.sh 环境配置文件

- cd $HADOOP_HOME/etc/hadoop
- vim hadoop-env.sh

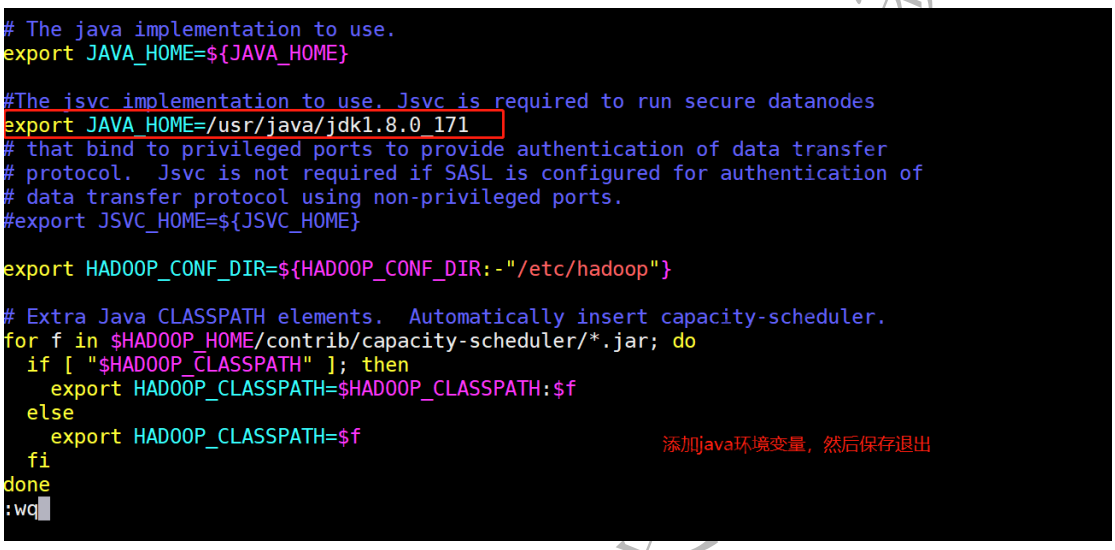
```
[root@master ~]# cd $HADOOP_HOME/etc/hadoop
[root@master hadoop]# ls
capacity-scheduler.xml      hadoop-policy.xml          kms-log4j.properties        ssl-client.xml.example
configuration.xsl           hdfs-site.xml              kms-site.xml                ssl-server.xml.example
container-executor.cfg      httpfs-env.sh              log4j.properties            yarn-env.cmd
core-site.xml               httpfs-log4j.properties    mapred-env.cmd              yarn-env.sh
hadoop-env.cmd              httpfs-signature.secret    mapred-env.sh               yarn-site.xml
hadoop-env.sh               httpfs-site.xml            mapred-queues.xml.template
hadoop-metrics2.properties  kms-acls.xml               mapred-site.xml.template
hadoop-metrics.properties   kms-env.sh                 slaves
[root@master hadoop]# vim hadoop-env.sh
```

进去hadoop配置文件目录

修改文件

输入以下内容，修改 java 环境变量：

```
export JAVA_HOME=/usr/java/jdk1.8.0_171
```



键入“Esc”，退出编辑模式，使用命令“:wq”进行保存退出。

2.编辑 core-site.xml 文件

- vim core-site.xml

```
<property>
  <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
    <value>/usr/hadoop/hadoop-2.7.3/hdfs/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
  <name>io.file.buffer.size</name>
    <value>131072</value>
</property>
<property>
  <name>fs.checkpoint.period</name>
    <value>60</value>
</property>
<property>
  <name>fs.checkpoint.size</name>
    <value>67108864</value>
</property>
```

master：在主节点的 ip 或者映射名。

9000：主节点和从节点配置的端口都是 9000。

```
<configuration>
 <property>
  <name>fs.default.name</name>
  <value>hdfs://master:9000</value>
 </property>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/usr/hadoop/hadoop-2.7.3/hdfs/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>
 <property>
  <name>io.file.buffer.size</name>
  <value>131072</value>
 </property>
 <property>
  <name>fs.checkpoint.period</name>
  <value>60</value>
 </property>
 <property>
  <name>fs.checkpoint.size</name>
  <value>67108864</value>
 </property>
</configuration>
```

注意〈configuration〉

〈/configuration〉

是否重复

同样注意保存退出。

3. 编辑 mapred-site.xml

hadoop 是没有这个文件的，需要将 mapred-site.xml.template 样本文件复制为 mapred-site.xml，对其进行编辑：

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

```
[root@master hadoop]# cp mapred-site.xml.template mapred-site.xml
[root@master hadoop]# vim mapred-site.xml
[root@master hadoop]#
```

在<configuration></configuration>中加入以下代码：

```
<property>
<!--指定 Mapreduce 运行在 yarn 上-->
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
 </property>
```

注意保存退出。

4.修改 yarn-site.xml

在<configuration></configuration>中加入以下代码：

```
<!-- 指定 ResourceManager 的地址-->
<property>
 <name>yarn.resourcemanager.address</name>
    <value>master:18040</value>
 </property>
 <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>master:18030</value>
```

```
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.address</name>
        <value>master:18088</value>
    </property>
    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>master:18025</value>
    </property>
    <property>
      <name>yarn.resourcemanager.admin.address</name>
      <value>master:18141</value>
    </property>
   <!-- 指定 reducer 获取数据的方式-->
    <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
    </property>
    <property>
      <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
      <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
```

```
<?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>
<property>
 <name>yarn.resourcemanager.address</name>
   <value>master:18040</value>
 </property>
 <property>
   <name>yarn.resourcemanager.scheduler.address</name>
   <value>master:18030</value>
 </property>
 <property>
   <name>yarn.resourcemanager.webapp.address</name>
   <value>master:18088</value>
 </property>
 <property>
   <name>yarn.resourcemanager.resource-tracker.address</name>
   <value>master:18025</value>
 </property>
 <property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>master:18141</value>
 </property>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>

<!-- Site specific YARN configuration properties -->

</configuration>
~
~
```

5. 编辑 hdfs-site.xml 配置文件

在<configuration></configuration>中加入以下代码：

```
<property>
  <name>dfs.replication</name>
     <value>2</value>
  </property>
  <property>
     <name>dfs.namenode.name.dir</name>
     <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/name</value>
     <final>true</final>
</property>
  <property>
     <name>dfs.datanode.data.dir</name>
```

```
    <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/data</value>
    <final>true</final>
  </property>
  <property>
   <name>dfs.namenode.secondary.http-address</name>
    <value>master:9001</value>
  </property>
  <property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
```

dfs.replication：因为 hadoop 是具有可靠性的，它会备份多个文本，这里 value 就是指备份的数量（小于等于从节点的数量）。

```
<configuration>
<property>
 <name>dfs.replication</name>
   <value>2</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/name</value>
   <final>true</final>
</property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/data</value>
   <final>true</final>
 </property>
 <property>
  <name>dfs.namenode.secondary.http-address</name>
   <value>master:9001</value>
 </property>
 <property>
   <name>dfs.webhdfs.enabled</name>
   <value>true</value>
 </property>
 <property>
   <name>dfs.permissions</name>
   <value>false</value>
 </property>

</configuration>
```

注意保存退出。

6.编写 slaves 和 master 文件

编写 slaves 文件，添加子节点 slave1 和 slave2；

编写 master 文件，添加主节点 master。

7.分发 hadoop 文件到 slave1、slave2 两个子节点

- scp -r /usr/hadoop root@slave1:/usr/
- scp -r /usr/hadoop root@slave2:/usr/

注意：

slave 各节点上还需要配置环境变量，参考 hadoop 中第一个步骤。

8.格式化 hadoop **(仅在 master 中进行操作)**

- 格式化 namenode：hadoop namenode -format

```
[root@master hadoop]# hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/09/29 16:39:04 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = master/192.168.15.104
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.3
STARTUP_MSG:   classpath = /usr/hadoop/hadoop-2.7.3/etc/hadoop:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-beanutils-core-1.8.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/hadoop-auth-2.7.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/junit-4.11.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/api-util-1.0.0-M20.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-beanutils-1.7.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/protobuf-java-2.5.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jsch-0.1.42.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-net-3.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/hadoop-annotations-2.7.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/mockito-all-1.8.5.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-lang-2.6.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/htrace-core-3.1.0-incubating.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/netty-3.6.2.Final.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-io-2.4.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/curator-client-2.7.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/zookeeper-3.4.6.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/guava-11.0.2.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/curator-framework-2.7.
```

主节点格式化

当出现“Exiting with status 0”的时候，表明格式化成功。

```
18/09/29 16:39:05 INFO namenode.FSImageFormatProtobuf: Saving image file /usr/hadoop/hadoop-2.7.3/hdfs/name/current/fsimage.ckpt_0000000000000000000 using no compression
18/09/29 16:39:05 INFO namenode.FSImageFormatProtobuf: Image file /usr/hadoop/hadoop-2.7.3/hdfs/name/current/fsimage.ckpt_0000000000000000000 of size 351 bytes saved in 0 seconds.
18/09/29 16:39:05 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
18/09/29 16:39:05 INFO util.ExitUtil: Exiting with status 0
18/09/29 16:39:05 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at master/192.168.15.104
************************************************************/
```

格式化成功

9.开启 hadoop 集群

仅在 master 主机上开启操作命令。它会带起从节点的启动。**（仅在 master 中进行操作）**

- 回到 hadoop 目录：cd /usr/hadoop/hadoop-2.7.3
- 主节点开启服务：sbin/start-all.sh

  master 主节点状态如下：

```
[root@master hadoop]# cd ..
[root@master etc]# cd ..
[root@master hadoop-2.7.3]# sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-namenode-master.out
slave2: starting datanode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-datanode-slave2.out
slave1: starting datanode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-datanode-slave1.out
Starting secondary namenodes [master]
master: starting secondarynamenode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-resourcemanager-master.out
slave1: starting nodemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-nodemanager-slave1.out
slave2: starting nodemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-nodemanager-slave2.out
[root@master hadoop-2.7.3]# jps
4722 Jps
4296 SecondaryNameNode
2856 QuorumPeerMain
4456 ResourceManager
4107 NameNode
[root@master hadoop-2.7.3]#
```

回到hadoop路径

master主节点开启hadoop集群

查看进程

slave1 节点状态如下：

```
[root@slave1 hadoop]# jps
3570 DataNode
3782 Jps
2519 QuorumPeerMain
3671 NodeManager
[root@slave1 hadoop]#
```

子节点中进程

slave2 节点状态如下：

```
[root@slave2 hadoop]# jps
2547 QuorumPeerMain
3603 DataNode
3815 Jps
3704 NodeManager
[root@slave2 hadoop]#
```

子节点slave2

11.使用 hadoop 命令“hadoop fs”进行相关操作。

5、安装 hbase

（同样先在主节点 master 下进行操作，然后在复制到子节点 slave1 和 slave2）

1. 创建工作路径/usr/hbase，下载相应软件，解压至工作路径。

2.进入 hbase 配置目录 conf，修改配置文件 hbase-env.sh，添加配置变量。

- cd /usr/hbase/hbase-1.2.4/conf
- vim hbase-env.sh

```
export HBASE_MANAGES_ZK=false
export JAVA_HOME=/usr/java/jdk1.8.0_171
export HBASE_CLASSPATH=/usr/hadoop/hadoop-2.7.3/etc/hadoop
```

```
# Set environment variables here.
export HBASE_MANAGES_ZK=false
# This script sets variables multiple times over the course of starting an hbase process,
# so try to keep things idempotent unless you want to take an even deeper look
# into the startup scripts (bin/hbase, etc.)

# The java implementation to use.  Java 1.7+ required.
# export JAVA_HOME=/usr/java/jdk1.6.0/
export JAVA_HOME=/usr/java/jdk1.8.0_171
# Extra Java CLASSPATH elements.  Optional.
export HBASE_CLASSPATH=/usr/hadoop/hadoop-2.7.3/etc/hadoop

# The maximum amount of heap to use. Default is left to JVM default.
# export HBASE_HEAPSIZE=1G

# Uncomment below if you intend to use off heap cache. For example, to allocate 8G of
# offheap, set the value to "8G".
# export HBASE_OFFHEAPSIZE=1G
```

解释：一个分布式运行的 Hbase 依赖一个 zookeeper 集群。所有的节点和客户端都必须能够访问 zookeeper。默认的情况下 Hbase 会管理一个 zookeep 集群，即 Hbase 默认自带一个 zookeep 集群。这个集群会随着 Hbase 的启动而启动。而在实际的商业项目中通常自己管理一个 zookeeper 集群更便于优化配置提高集群工作效率，但需要配置 Hbase。需要修改

conf/hbase-env.sh 里面的 HBASE_MANAGES_ZK 来切换。这个值默认是 true 的，作用是让 Hbase 启动的时候同时也启动 zookeeper.在本实验中，我们采用独立运行 zookeeper 集群的方式，故将其属性值改为 false。

3.配置 conf/hbase-site.xml 文件

```
<property>
        <name>hbase.rootdir</name>
        <value>hdfs://master:9000/hbase</value>
</property>
<property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
</property>
<property>
        <name>hbase.master</name>
        <value>hdfs://master:6000</value>
</property>
<property>
        <name>hbase.zookeeper.quorum</name>
        <value>master,slave1,slave2</value>
</property>
<property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>/usr/zookeeper/zookeeper-3.4.10</value>
</property>
```

```
<configuration>
<property>
        <name>hbase.rootdir</name>
        <value>hdfs://master:9000/hbase</value>
    </property>
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
    </property>
    <property>
        <name>hbase.master</name>
        <value>hdfs://master:6000</value>
    </property>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>master,slave1,slave2</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>/usr/zookeeper/zookeeper-3.4.10</value>
    </property>
</configuration>
```

解释：要想运行完全分布式模式，加一个属性 hbase.cluster.distributed 设置为 true 然后把 hbase.rootdir 设置为 HDFS 的 NameNode 的位置；

hbase.rootdir：这个目录是 region server 的共享目录，用来持久化 Hbase。URL 需要是’完全正确’的，还要包含文件系统的 scheme；

hbase.cluster.distributed ：Hbase 的运行模式。false 是单机模式，true 是分布式模式。若

为 false,Hbase 和 Zookeeper 会运行在同一个 JVM 里面。在 hbase-site.xml 配置 zookeeper，当 Hbase 管理 zookeeper 的时候，你可以通过修改 zoo.cfg 来配置 zookeeper，对于 zookeepr 的配置，你至少要在 hbase-site.xml 中列出 zookeepr 的 ensemble servers，具体的字段是 hbase.zookeeper.quorum.在这里列出 Zookeeper 集群的地址列表，用逗号分割。

hbase.zookeeper.property.clientPort：ZooKeeper 的 zoo.conf 中的配置.客户端连接的端口。

hbase.zookeeper.property.dataDir：ZooKeeper 的 zoo.conf 中的配置。对于独立的 Zookeeper，要指明 Zookeeper 的 host 和端口。需要在 hbase-site.xml 中设置。

4. 配置 conf/regionservers，添加子节点

在这里列出了希望运行的全部 HRegionServer，一行写一个 host (就 Hadoop 里面的 slaver 一样)。列在这里的 server 会随着集群的启动而启动，集群的停止而停止。

5.hadoop 配置文件拷入 hbase 的目录下(当前目录为/usr/hbase/hbase-1.2.4/conf)

- cp /usr/hadoop/hadoop-2.7.3/etc/hadoop/hdfs-site.xml .
- cp /usr/hadoop/hadoop-2.7.3/etc/hadoop/core-site.xml .

6.分发 hbase 到子节点

- scp -r /usr/hbase root@slave1:/usr/
- scp -r /usr/hbase root@slave2:/usr/

7.配置环境变量（三台机器）

- vim /etc/profile

配置环境变量 Hbase

```
# set hbase environment
export HBASE_HOME=/usr/hbase/hbase-1.2.4
export PATH=$PATH:$HBASE_HOME/bin
```

- 生效环境变量：source /etc/profile

9. 运行和测试，**在 master 上执行(保证 hadoop 和 zookeeper 已开启)**

- jps

```
[root@master hbase-1.2.4]# bin/start-hbase.sh
starting master, logging to /usr/hbase/hbase-1.2.4/logs/hbase-root-master-master.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
slave1: starting regionserver, logging to /usr/hbase/hbase-1.2.4/bin/../logs/hbase-root-regionserver-slave1.out
slave2: starting regionserver, logging to /usr/hbase/hbase-1.2.4/bin/../logs/hbase-root-regionserver-slave2.out
slave1: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
slave1: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
slave2: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
slave2: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
[root@master hbase-1.2.4]# jps
5522 Jps
5299 HMaster
4296 SecondaryNameNode
2856 QuorumPeerMain
4456 ResourceManager
4107 NameNode
[root@master hbase-1.2.4]#
```

开启hbase

查看进程HMaster，说明hbase已经开启成功

6、安装 hive

实验中我们选用 hive 的远程模式，slave2 安装 mysql server 用于存放元数据，slave1 作为 hive server 作为 thrift 服务器，master 作为 client 客户端进行操作。

6.1slave2 上安装 MySQL server

1.配置过本地源了，安装 MySQL Server

- 安装 MySQL：yum -y install mysql-community-server

```
[root@slave2 ~]# yum -y install mysql-community-server
Loaded plugins: fastestmirror
mysql5.7                                                                  | 2.9
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package mysql-community-server.x86_64 0:5.7.23-1.el7 will be installed
--> Processing Dependency: mysql-community-common(x86-64) = 5.7.23-1.el7 for package: mysql-community-server-
4
--> Processing Dependency: mysql-community-client(x86-64) >= 5.7.9 for package: mysql-community-server-5.7.23
--> Running transaction check
---> Package mysql-community-client.x86_64 0:5.7.23-1.el7 will be installed
--> Processing Dependency: mysql-community-libs(x86-64) >= 5.7.9 for package: mysql-community-client-5.7.23-1
---> Package mysql-community-common.x86_64 0:5.7.23-1.el7 will be installed
--> Running transaction check
---> Package mariadb-libs.x86_64 1:5.5.52-1.el7 will be obsoleted
--> Processing Dependency: libmysqlclient.so.18()(64bit) for package: 2:postfix-2.10.1-6.el7.x86_64
--> Processing Dependency: libmysqlclient.so.18(libmysqlclient_18)(64bit) for package: 2:postfix-2.10.1-6.el7
---> Package mysql-community-libs.x86_64 0:5.7.23-1.el7 will be obsoleting
--> Running transaction check
---> Package mysql-community-libs-compat.x86_64 0:5.7.23-1.el7 will be obsoleting
--> Finished Dependency Resolution
```

slave2中安装MySQL Server

2.启动服务

- 重载所有修改过的配置文件：systemctl daemon-reload
- 开启服务：systemctl start mysqld
- 开机自启：systemctl enable mysqld

```
[root@slave2 ~]# systemctl daemon-reload
[root@slave2 ~]# systemctl start mysqld
[root@slave2 ~]# systemctl enable mysqld
```

3.登陆 MySQL

安装完毕后，MySQL 会在/var/log/mysqld.log 这个文件中会自动生成一个随机的密码，获取得这个随机密码，以用于登录 MySQL 数据库：

- 获取初密码： grep "temporary password" /var/log/mysqld.log
- 登陆 MySQL：mysql -uroot -p（注意中英文）

```
[root@slave2 ~]# grep "temporary password" /var/log/mysqld.log
2018-10-09T03:20:29.0654717 1 [Note] A temporary password is generated for root@localhost: A8Q&i!2G,zf<
[root@slave2 ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.23

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

获取初始密码

登陆MySQL

4.MySQL 密码安全策略设置

- 设置密码强度为低级：set global validate_password_policy=0;
- 设置密码长度：set global validate_password_length=4;
- 修改本地密码：alter user 'root'@'localhost' identified by '123456';
- 退出：\q

```
mysql> set global validate_password_policy=0;
Query OK, 0 rows affected (0.00 sec)

mysql> set global validate_password_length=4;
Query OK, 0 rows affected (0.00 sec)

mysql> alter user 'root'@'localhost' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> \q
Bye
[root@slave2 ~]#
```

设置密码强度为low

密码最低长度为4

修改本地root用户密码为123456

退出

密码强度分级如下：

- 0 为 low 级别，只检查长度；
- 1 为 medium 级别（默认），符合长度为 8，且必须含有数字，大小写，特殊字符;
- 2 为 strong 级别，密码难度更大一些，需要包括字典文件。
- 密码长度最低长为 4，当设置长度为 1、2、3 时，其长度依然为 4。

5. 设置远程登录

- 以新密码登陆 MySQL：mysql -uroot -p123456
- 创建用户：create user 'root'@'%' identified by '123456';
- 允许远程连接：grant all privileges on *.* to 'root'@'%' with grant option;
- 刷新权限：flush privileges;

```
mysql> create user 'root'@'%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on *.* to 'root'@'%' with grant option;
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

6.创建数据库 test

6.2 创建工作路径，解压安装包

创建工作路径/usr/hive，下载相应软件，解压至工作路径。

环境中 master 作为客户端，slave1 作为服务器端，因此都需要使用到 hive。因为 hive 相关安装包存放在 master 中，因此我们先在 master 中对 hive 进行解压，然后将其复制到 slave1 中。

6.3slave1 中建立文件

同样 slave1 上建立文件夹/usr/hive，然后 master 中将安装包远程复制到 slave1。

- mkdir -p /usr/hive

master 中将 hive 文件复制到 slave1：

- scp -r /usr/hive/apache-hive-2.1.1-bin root@slave1:/usr/hive/

```
[root@master soft]# scp -r /usr/hive/apache-hive-2.1.1-bin root@slave1:/usr/hive/
srcbucket0.txt                                                                100% 5702
struct2_d.txt                                                                 100%  127
tsformat.json                                                                 100%  108
map_null_schema.avro                                                          100%  187
decimal.txt                                                                   100%   95
map_table.txt                                                                 100%   52
z.txt                                                                         100%    6
small_csv.csv                                                                 100% 2280
smbbucket_1.rc                                                                100%  208
dec.parq                                                                      100%  335
sortdp.txt                                                                    100% 2598
```

master中将hive远程复制到slave1中

注意slave1要提前创建工作路径/usr/hive

修改/etc/profile 文件设置 hive 环境变量。（master 和 slave1 都执行）

- vim /etc/profile

```
#set hive
export HIVE_HOME=/usr/hive/apache-hive-2.1.1-bin
export PATH=$PATH:$HIVE_HOME/bin
```

```
[root@master ~]# vim /etc/profile

# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
#set zookeeper environment
export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10
PATH=$PATH:$ZOOKEEPER_HOME/bin
# # # HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin

# set hbase environment
export HBASE_HOME=/usr/hbase/hbase-1.2.4
export PATH=$PATH:$HBASE_HOME/bin

#set hive
export HIVE_HOME=/usr/hive/apache-hive-2.1.1-bin
export PATH=$PATH:$HIVE_HOME/bin
```

添加hive环境变量

- 生效环境变量：source /etc/profile

```
[root@master ~]# vim /etc/profile
[root@master ~]# source /etc/profile
[root@master ~]#
```

6.4 解决版本冲突和 jar 包依赖问题

由于客户端需要和 hadoop 通信，所以需要更改 Hadoop 中 jline 的版本。即保留一个高版本的 jline jar 包，从 hive 的 lib 包中拷贝到 Hadoop 中 lib 位置为 /usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib。（**master 中执行**）

- cp /usr/hive/apache-hive-2.1.1-bin/lib/jline-2.12.jar /usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/

```
[root@master ~]# cp /usr/hive/apache-hive-2.1.1-bin/lib/jline-2.12.jar /usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/
[root@master ~]#
```

因为服务端需要和 Mysql 通信，所以服务端需要将 Mysql 的依赖包放在 Hive 的 lib 目录下。（**slave1 中进行**）

- cd /usr/hive/apache-hive-2.1.1-bin/lib
- wget http://10.10.30.2:8000/bigdata/bigdata_tar/mysql-connector-java-5.1.47-bin.jar

```
[root@slave1 ~]# cd /usr/hive/apache-hive-2.1.1-bin/lib
[root@slave1 lib]# wget http://172.31.200.100/repo4/bigdata_tar/mysql-connector-java-5.1.47-bin.jar
--2018-12-04 13:50:34--  http://172.31.200.100/repo4/bigdata_tar/mysql-connector-java-5.1.47-bin.jar
Connecting to 172.31.200.100:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1007505 (984K) [application/x-java-archive]
Saving to: 'mysql-connector-java-5.1.47-bin.jar'

100%[======================================================================================================================================>] 1,007,505   --.-K/s   in 0.06s

2018-12-04 13:50:34 (16.1 MB/s) - 'mysql-connector-java-5.1.47-bin.jar' saved [1007505/1007505]

[root@slave1 lib]#
```

6.5Slave1 作为服务器端配置 hive

回到 slave1，修改 hive-env.sh 文件中 HADOOP_HOME 环境变量。进入 hive 配置目录，因为 hive 中已经给出配置文件的范本 hive-env.sh.template，直接将其复制一个进行修改即可：

- cd $HIVE_HOME/conf
- ls
- cp hive-env.sh.template hive-env.sh
- vim hive-env.sh

```
[root@slave1 ~]# cd $HIVE_HOME/conf
[root@slave1 conf]# ls                                          进入配置文件
beeline-log4j2.properties.template  hive-exec-log4j2.properties.template  llap-cli-log4j2.properties.template
hive-default.xml.template           hive-log4j2.properties.template       llap-daemon-log4j2.properties.template
hive-env.sh.template                ivysettings.xml                       parquet-logging.properties
[root@slave1 conf]# cp hive-env.sh.template hive-env.sh         重命名hive-env.sh
[root@slave1 conf]# vim hive-env.sh
```

hive-env.sh 文件中修改 HADOOP_HOME 环境变量。

HADOOP_HOME=/usr/hadoop/hadoop-2.7.3

```
# Set HADOOP_HOME to point to a specific hadoop install directory
HADOOP_HOME=/usr/hadoop/hadoop-2.7.3          修改HADOOP_HOME路径
```

3.修改 hive-site.xml 文件

```
<configuration>
<!-- Hive 产生的元数据存放位置-->
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive_remote/warehouse</value>
</property>
<!-- 数据库连接 JDBC 的 URL 地址-->
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://slave2:3306/hive?createDatabaseIfNotExist=true</value>
</property>
<!-- 数据库连接 driver，即 MySQL 驱动-->
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
</property>
<!-- MySQL 数据库用户名-->
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
</property>
<!-- MySQL 数据库密码-->
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>123456</value>
</property>
<property>
<name>hive.metastore.schema.verification</name>
<value>false</value>
</property>
<property>
<name>datanucleus.schema.autoCreateAll</name>
<value>true</value>
</property>
</configuration>
```

```
[root@slave1 conf]# vim hive-site.xml

<configuration>
  <!-- Hive产生的元数据存放位置-->
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive_remote/warehouse</value>
</property>
    <!-- 数据库连接JDBC的URL地址-->
<property>
    <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://slave2:3306/hive?createDatabaseIfNotExist=true</value>
</property>
    <!-- 数据库连接driver，即MySQL驱动-->
<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
</property>
    <!-- MySQL数据库用户名-->
<property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>root</value>
</property>
    <!-- MySQL数据库密码-->
<property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>123456</value>
 </property>
<property>
    <name>hive.metastore.schema.verification</name>
    <value>false</value>
 </property>
<property>
    <name>datanucleus.schema.autoCreateAll</name>
    <value>true</value>
 </property>
</configuration>
```

连接数据库及其端口
注意这里可以使用的是映射名，
真实环境中，可以使用数据库所
在的IP地址

6.6Master 作为客户端配置 hive

和 slave1 中配置方式类似，直接进入

1.修改 hive-site.xml

```
<configuration>
<!-- Hive 产生的元数据存放位置-->
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive_remote/warehouse</value>
</property>
<!--- 使用本地服务连接 Hive,默认为 true-->
<property>
<name>hive.metastore.local</name>
<value>false</value>
</property>
<!-- 连接服务器-->
<property>
<name>hive.metastore.uris</name>
<value>thrift://slave1:9083</value>
</property>
</configuration>
```

```
[root@master conf]# vim hive-site.xml

<configuration>

<!-- Hive产生的元数据存放位置-->
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive_remote/warehouse</value>
</property>

<!--- 使用本地服务连接Hive,默认为true-->
<property>
    <name>hive.metastore.local</name>
    <value>false</value>
</property>

<!-- 连接服务器-->
<property>
    <name>hive.metastore.uris</name>
    <value>thrift://slave1:9083</value>
</property>
</configuration>
```

远程模式使用的是其他数据库
因此本地数据库修改为false

客户端通过服务器slave1去连接数据库
这里使用slave1是服务器的映射名
可以使用真实ip

2.修改 hive-env.sh 中 HADOOP_HOME 环境变量：

```
HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
```

6.7 成功启动 Hive

1.启动 hive server 服务（slave1 上）

- bin/hive --service metastore（注意空格）

```
[root@slave1 apache-hive-2.1.1-bin]# bin/hive --service metastore
Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hive/apache-hive-2.1.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/St
nder.class]
SLF4J: Found binding in [jar:file:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/sl
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

slave1中开启的hive server 服务
注意在hive目录下进行

开启状态

2.启动 hive client(master 上)

- bin/hive

测试 hive 是否启动成功

- hive>show databases;

```
[root@master apache-hive-2.1.1-bin]# bin/hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hive/apache-hive-2.1.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/im
nder.class]
SLF4J: Found binding in [jar:file:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/o
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/usr/hive/apache-hive-2.1.1-bin/lib/hive-common-2.1.1.jar!
erties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
 (i.e. spark, tez) or using Hive 1.X releases.
hive> show databases;
OK
default
Time taken: 1.954 seconds, Fetched: 1 row(s)
hive>
```

master作为客户端，开启hive

master通过slave1服务器连接MySQL成功
通过show命令进行简单测试

创建数据库 hive_db

- hive>create database hive_db;

提高部分

7、Spark 安装

7.1 安装 scala 环境

任务要求如下：（三台机器）
1.从本地源下载相应安装包，创建对应工作目录/usr/scala，并将工具解压至工作目录。
2.配置 scala 的环境变量并生效：
3.查看 scala 是否安装成功：

7.2 安装 Spark

任务要求如下：
1.从本地源下载相应安装包，创建对应工作目录/usr/spark，将工具解压至工作目录；
2.配置 conf/spark-env.sh 文件，设置要求如下：

设置 master 为 SPARK 主节点 IP（SPARK_MASTER_IP） 设置 SCALA_HOME、JAVA_HOME、HADOOP_HOME 设置 Hadoop 配置目录路径（HADOOP_CONF_DIR） 设置 spark 工作内存为 8G（SPARK_WORKER_MEMORY）

3.配置 spark 从节点，修改 slaves 文件；

提示：注意 slaves 节点中只包含节点信息，其他注释不需要。

4.向所有子节点发送 spark 配置好的安装包；

提示：即要求三台节点 spark 配置一致即可。

5.设置 SPARK_HOME 环境变量，将$SPARK_HOME/bin 加入到 PATH；
6.开启 spark 服务；

提示：主节点执行

8、数据爬取

8.1Python 环境配置

环境中已经安装 Python3.6.8，支持的库为：

beautifulsoup	4 4.8.0
bs4	0.0.1
certifi	2019.6.16
chardet	3.0.4
html5lib	1.0.1
idna	2.8
lxml	4.4.1
pip	18.1
requests	2.22.0
setuptools	40.6.2
six	1.12.0
soupsieve	1.9.3
urllib3	1.25.3
webencodings	0.5.1

8.2 爬虫设计

访问比赛平台提供的 discuzserver（爬虫环境）对应 ip 的网站（比赛平台左上部分，具体参考“大数据竞赛选手使用手册”中 2.2 小节），网站类型为 discuz 论坛，分析网站结构，并针对后续题目要求设计爬虫代码，爬取网站内容。

8.3 数据爬取

要求爬取的数据至少包括帖子 ID，标题，作者 ID。

提示：网站内所有的注册用户，均至少发了一个帖子，可以根据此条件得出网站的全量用户列表。

8.4 设计数据表

根据分析题目要求，表 crawl_discuz(hive_db 库中)需要包含以下字段：

帖子唯一 id ：tid

帖子标题 ：title

帖子作者 id ：uid

爬取后的数据如下图所示：

1	什么发帖软件最好？	2
2	什么发帖软件最好？	3
3	什么发帖软件最好？	4
4	什么发帖软件最好？	2
5	什么发帖软件最好？	3
6	什么发帖软件最好？	4
7	什么发帖软件最好？	5
8	什么发帖软件最好？	6
9	什么发帖软件最好？	7
10	什么发帖软件最好？	8

将爬虫结果输出到文件 discuz_result.txt 中，用于导入数据表。

8.5 数据保存

任务要求如下：

1）创建 crawl_discuz 表，要求字段包括 tid ,title,uid。（分隔符格式自定义）；

2）将 discuz_result.txt 导入 hive_db 数据库的的 crawl_discuz 表中；