## 前言

安装选择的版本为cdh5.8.0，主要组件包括hadoop、zookeeper和hive。具体下载的网址可以参考以下链接：

http://archive.cloudera.com/cdh5/cdh/5/hadoop-2.6.0-cdh5.8.0.tar.gz

http://archive.cloudera.com/cdh5/cdh/5/hive-1.1.0-cdh5.8.0.tar.gz

http://archive.cloudera.com/cdh5/cdh/5/zookeeper-3.4.5-cdh5.8.0.tar.gz

## 规划

1、主机规划

| | bigdata-214/192.168.8.214 | bigdata-215/192.168.8.215 | bigdata-216/192.168.8.216 | bigdata-217/192.168.8.217 | bigdata-218/192.168.8.218 | bigdata-219/192.16 |
|---|---|---|---|---|---|---|
| zookeeper | Y | Y | Y | | | |
| namenode | Y | Y | | | | |
| datanode | | | Y | Y | Y | Y |
| journalnode | Y | Y | Y | | | |
| resourcemanager | Y | Y | | | | |

备注：zookeeper和journalnode保持奇数个，至少为3

2、软件规划

| 软件 | 版本 | 位数 | 说明 |
|---|---|---|---|
| CentOS | 6.5 | 64 | |
| JDK | 1.8.0_102 | 64 | |
| Zookeeper | 3.4.5 | | cloudera发行版 |
| Hadoop | 2.6.0 | | cloudera发行版 |
| Hive | 1.1.0 | | cloudera发行版 |

3、用户规划

| 节点名称 | 用户组 | 用户 | 密码 |
|---|---|---|---|
| bigdata-214 | hadoop | hadoop | hadoop |
| bigdata-215 | hadoop | hadoop | hadoop |
| bigdata-216 | hadoop | hadoop | hadoop |
| bigdata-217 | hadoop | hadoop | hadoop |
| bigdata-218 | hadoop | hadoop | hadoop |
| bigdata-219 | hadoop | hadoop | hadoop |

4、目录规划

| 名称 | 路径 |
|---|---|
| 软件目录 | /home/hadoop |
| 数据及日志目录 | /data/hdfs, /data/logs |

## 环境检查

1、修改主机名

比如将192.168.8.214这台机器主机名修改为bigdata-214，经过如下四步，其他剩余节点以此按这种规则命名：

> vim /etc/sysconfig/network
>
> 修改第二行为 HOSTNAME=bigdata-214，然后保存退出(修改需要以root用户操作)
>
> 执行 hostname -f，输出为biodata-214，说明成功完成

2、绑定主机名

修改所有机器的/etc/hosts,追加如下内容：

```
192.168.8.214 bigdata-214
192.168.8.215 bigdata-215
192.168.8.216 bigdata-216
192.168.8.217 bigdata-217
192.168.8.218 bigdata-218
192.168.8.219 bigdata-219
```

3、禁用防火墙

```
service iptables status  # 查看防火墙状态

servvice iptables stop   # 临时关闭防火墙

chkconfig iptables off   # 永久关闭防火墙，但必须重启后才生效
```

4、时间同步

由于分布式系统对子节点间的时间有要求，如果差异太大，容易造成很多问题。为此，需要每台服务器上开启时间同步后台服务。

```
yum install ntp

service ntpd start

chkconfig ntpd on
```

5、修改打开文件句柄数

Linux对每个用户限制其可以创建的最大进程数，同时限制单个进程可以打开的文件句柄数，默认为1024。由于hadoop集群运行过程中会产生很多句柄，如果不适当调大，很容易超过而报错。方法是修改limits.conf,重启后才能生效

```
vim /etc/security/limits.conf

在文件末尾追加：

 * soft noproc 65535

 * hard noproc 65535

 * soft nofile 65535

 * hard nofile 65535
```

6、安装JDK

Hadoop组件都是以JVM作为运行环境，目前选择的的是JDK8，主要考虑到后面hadoop3.X可能会都切换到JDK8，先将版本调高。

```
/usr/java/jdk1.8.0_102

在/etc/profile中加入：

export JAVA_HOME=/usr/java/jdk1.8.0_102
export CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$PATH:$JAVA_HOME/bin
```

7、创建用户（组）

整个hadoop集群环境，以hadoop用户运行，需要提前创建好对应的用户和组，在每台机器上面。同时创建好对应的目录资源。

```
groupadd hadoop

useradd hadoop -g hadoop

passwd hadoop  # 密码设置为hadoop

su root

mkdir -p /data/hdfs /data/zookeeper /data/logs

chown -R hadoop:hadoop /data/hdfs /data/zookeeper /data/logs
```

8、SSH免密码登录

打通hadoop用户情况下，namenode节点之间、namenode到集群其他节点间的SSH免密码登录，目的是可以在namenode节点，通过命令方便将其他节点部署的服务进程启动，减少每次都询问输入密码的麻烦和痛苦。

```
su hadoop

ssh-keygen -t rsa  # 一路enter即可

chmod 700 ~/.ssh

touch ~/.ssh/authorized_keys
```

> cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys # 把公钥放入认证文件中
>
> chmod 600 ~/.ssh/authorized_keys

这上面列出的需要在所有节点上执行，都成功处理好以后，需要将namenode节点的authorized_key拷贝到其他所有节点的/home/hadoop目录，然后在其他每个节点上执行

> cat ~/authorized_keys ~/.ssh/authorized_keys
>
> delete ~/authorized_keys

然后就可以在namenode节点任意跳转到其他节点了，ssh免密码过去，但第一次有时会提示The authenticity of host "" can't be established ，键入Yes即可，以后不会提示了。

## 安装Zookeeper

将本地下载好的安装包zookeeper-3.4.5-cdh5.8.0.tar.gz上传到bigdata-214节点的/home/hadoop,修改成hadoop用户可以读写，然后解压

> su hadoop
>
> tar xvf zookeeper-3.4.5-cdh5.8.0.tar.gz
>
> cd zookeeper-3.4.5-cdh5.8.0/conf
>
> cp zoo_sample.cfg zoo.cfg
>
> 修改文件内容如下：
>
> tickTime=2000
> initLimit=10
> syncLimit=5
> dataDir=/data/zookeeper
> clientPort=2181
> maxClientCnxns=500
> maxSessionTimeout=600000
> server.1=192.168.8.214:2888:3888
> server.2=192.168.8.215:2888:3888
> server.3=192.168.8.216:2888:3888
> autopurge.snapRetainCount=7
> autopurge.purgeInterval=24

将修改好配置的zookeeper安装包同步到bigdata-215和bigdata-216机器上，然后每台机器上执行以下命名：

> su hadoop
>
> touch /data/zookeeper/myid
>
> echo "1" > /data/zookeeper/myid # 请注意，bigdata-215需要echo"2", biodata-216需要echo "3",其实就是跟配置文件中标识的每台服务器分配的标识号保持一致
>
> vim ~/.bash_profile #将zookeeper配置到环境变量中 ，加入以下内容：
>
> export ZOOKEEPER_HOME=/home/hadoop/zookeeper-3.4.5-cdh5.8.0/
>
> export PATH=$PATH:$ZOOKEEPER_HOME/bin

然后每台机器上执行 zkServer.sh start,如果jps查看到每台机器上都有QuorumPeerMain进程，则应该Zookeeper集群成功部署并启动运行。

## 安装Hadoop

将本地下载好的安装包hadoop-2.6.0-cdh5.8.0.tar.gz上传到bigdata-214节点的/home/hadoop,修改成hadoop用户可以读写，然后解压

> su hadoop
>
> tar xvf hadoop-2.6.0-cdh5.8.0.tar.gz
>
> vim ~/.bash_profile #将hadoop配置到环境变量中 ，加入以下内容：
>
> export HADOOP_HOME=/home/hadoop/hadoop-2.6.0-cdh5.8.0/
>
> export PATH=$PATH:$HADOOP_HOME/bin
>
> 然后切换到以下目录：
>
> cd $HADOOP_HOME/etc/hadoop
>
> 修改hadoop-env.sh,加入JAVA_HOME
>
> export JAVA_HOME=/usr/java/jdk1.8.0_102

配置core-site.xml:

```
<property>
   <name>fs.defaultFS</name>
   <value>hdfs://cluster1</value>
</property>
<property>
   <name>hadoop.tmp.dir</name>
   <value>/data/tmp</value>
 </property>
 <property>
   <name>ha.zookeeper.quorum</name>
   <value>bigdata-214:2181,bigdata-215:2181,bigdata-216:2181</value>
  </property>
```

配置hdfs-site.xml

```
<property>
   <name>fs.replication</name>
   <value>3</value>
</property>
<property>
   <name>dfs.nameservices</name>
   <value>cluster1</value>
 </property>
 <property>
   <name>dfs.ha.namenodes.cluster1</name>
   <value>nameService1,nameService2</value>
  </property>
  <property>
   <name>dfs.namenode.rpc.address.cluster1.nameService1</name>
   <value>bigdata-214:9000</value>
  </property>
    <property>
   <name>dfs.namenode.http.address.cluster1.nameService1</name>
   <value>bigdata-214:50070</value>
  </property>
    <property>
   <name>dfs.namenode.rpc.address.cluster1.nameService2</name>
   <value>bigdata-215:9000</value>
  </property>
    <property>
   <name>dfs.namenode.http.address.cluster1.nameService2</name>
   <value>bigdata-215:50070</value>
  </property>
      <property>
   <name>dfs.ha.automatic-failover.enabled</name>
   <value>true</value>
  </property>
  <property>
   <name>dfs.namenode.shared.edits.dir</name>
   <value>qjournal://bigdata-214:8485;bigdata-215:8485;bigdata-216:8485/cluster1</value>
  </property>
  <property>
   <name>dfs.client.failover.proxy.provider.cluster1</name>
   <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
  </property>
  <property>
   <name>dfs.journalnode.edits.dir</name>
   <value>/data/jn</value>
  </property>
  <property>
   <name>dfs.ha.fencing.methods</name>
   <value>shell(/bin/true)</value>
  </property>
  <property>
   <name>dfs.ha.fencing.ssh.private-key.files</name>
   <value>/home/hadoop/.ssh/id_rsa </value>
  </property>
  <property>
   <name>dfs.ha.fencing.ssh.connect.timeout</name>
   <value>10000</value>
  </property>
    <property>
   <name>dfs.namenode.handler.count</name>
   <value>100</value>
  </property>
</configuration>
```

配置Slaves

```
bigdata-216
bigdata-217
bigdata-218
bigdata-219
```

配置mapred-site.xml

```
mapreduce.framework.name
```

> yarn

配置yarn-site.xml

```
yarn.resourcemanager.connect.retry.interval.ms
2000


yarn.resourcemanager.ha.enabled
true

yarn.resourcemanager.ha.automatic-failover.enabled
true

yarn.resourcemanager.ha.automatic-failover.embedded
true

yarn.resourcemanager.cluster.id
yarn-rm-cluster

yarn.resourcemanager.ha.rm-ids
rm1,rm2

yarn.resourcemanager.hostname.rm1
bigdata-214

yarn.resourcemanager.hostname.rm2
bigdata-215

yarn.resourcemanager.recovery.enabled
true

yarn.resourcemanager.zk.state-store.address
bigdata-214:2181,bigdata-215:2181,bigdata-216:2181

yarn.resourcemanager.zk-address
bigdata-214:2181,bigdata-215:2181,bigdata-216:2181

yarn.resourcemanager.address.rm1
bigdata-214:8032

yarn.resourcemanager.address.rm2
bigdata-215:8032

yarn.resourcemanager.scheduler.address.rm1
bigdata-214:8034
```

```
yarn.resourcemanager.scheduler.address.rm2
  bigdata-215:8034

yarn.resourcemanager.webapp.address.rm1
  bigdata-214:8088

yarn.resourcemanager.webapp.address.rm2
  bigdata-215:8088

yarn.nodemanager.aux-services
  mapreduce_shuffle

yarn.nodemanager.aux-services.mapreduce_shuffle.class
  org.apache.hadoop.mapred.ShuffleHandler
```

然后将安装包拷贝到其他所有节点上，启动bigdata-214，bigdata-215和bigdata-216节点上的journalnode（sbin/hadoop-daemon.sh start journalnode）。接着在bigdata-214节点上，以hadoop用户执行（前提是Zookeeper集群已经成功启动运行了）

```
su hadoop
cd /home/hadoop/hadoop-2.6.0-cdh5.8.0/
bin/hdfs namenode -format  # namenode 格式化
bin/hdfs zkfc -formatZK #格式化高可用
bin/hdfs namenode
然后在备节点上执行数据同步（hadoop用户执行）：
bin/hdfs namenode -bootstrapStandby
```

bigdata-215上数据同步完成后，停止掉biodata-214上面的namenode节点，同时停止掉所有的journalnode节点，最后便可以开始一键启动所有服务：

```
su hadoop
cd /home/hadoop/hadoop-2.6.0-cdh5.8.0/
sbin/start-dfs.sh
sbin/start-dfs.sh
bin/yarn rmadmin -getServiceState rm1 # 查看rm状态
hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /tmp/a.txt /tmp/out/
```

## 安装Hive

前提是已经安装好Hadoop、JDK和MySQL。我们下载好安装包后，将其拷贝解压到/home/hadoop目录下，以hadoop用户操作。

```
su hadoop
cd /home/hadoop/hive-1.1.0-cdh5.8.0
配置环境变量，vim /etc/profile
export HIVE_HOME=/home/hadoop/hive-1.1.0-cdh5.8.0
export PATH=$PATH:$HIVE_HOME/bin
```

准备如下配置文件：

```
su hadoop
cd /home/hadoop/hive-1.1.0-cdh5.8.0/conf
cp hive-env.sh.template hive-env.sh
cp hive-default.xml.template hive-site.xml
```

```
cp hive-log4j2.properties.template hive-log4j2.properties
cp hive-exec-log4j2.properties.template hive-exec-log4j2.properties
```

因为hive使用到了hadoop，所有需要配置hadoop的环境变量：

```
export JAVA_HOME=/usr/java/jdk1.8.0_102                    ##Java路径
export HADOOP_HOME=/home/hadoop/hadoop-2.6.0-cdh5.8.0   ##Hadoop安装路径
export HIVE_HOME=/home/hadoop/hive-1.1.0-cdh5.8.0          ##Hive安装路径
export HIVE_CONF_DIR=/home/hadoop/hive-1.1.0-cdh5.8.0/conf  ##Hive配置文件路径
```

替换hive-site.xml文件中的 ${system:java.io.tmpdir} 和 ${system:user.name}

```
  hive.exec.scratchdir
  /tmp/hive-${user.name}
  HDFS root scratch dir for Hive jobs which gets created with write all (733) permission. For each connecting user, an HDFS scratch dir: ${hive.exec.scratchdir}/<username> is created, with ${hive.scratch.dir.permission}.


  hive.exec.local.scratchdir
  /tmp/${user.name}
  Local scratch space for Hive jobs


  hive.downloaded.resources.dir
  /tmp/hive/resources
  Temporary local directory for added resources in the remote file system.


  hive.querylog.location
  /tmp/${user.name}
  Location of Hive run time structured log file


  hive.server2.logging.operation.log.location
  /tmp/${user.name}/operation_logs
  Top level directory where operation logs are stored if logging functionality is enabled
```

默认情况下, Hive的元数据保存在了内嵌的 derby 数据库里, 但一般情况下生产环境使用 MySQL 来存放 Hive 元数据。首先将 mysql-connector-java-5.1.39.jar 放入 $HIVE_HOME/lib 下。然后hive-site.xml 中配置 MySQL 数据库连接信息：

```
  javax.jdo.option.ConnectionURL
  jdbc:mysql://192.168.8.219:3306/hive?createDatabaseIfNotExist=true&characterEncoding=UTF-8&useSSL=false


  javax.jdo.option.ConnectionDriverName


  javax.jdo.option.ConnectionUserName
  hive


  javax.jdo.option.ConnectionPassword
  hive
```

## 配置文件重命名

在运行 Hive 之前需要使用以下命令修改配置文件:

```
cd /opt/hive/conf
cp hive-env.sh.template hive-env.sh
cp hive-default.xml.template hive-site.xml
cp hive-log4j2.properties.template hive-log4j2.properties
cp hive-exec-log4j2.properties.template hive-exec-log4j2.properties
```

## 修改hive-env.sh

因为 Hive 使用了 Hadoop, 需要在 hive-env.sh 文件中指定 Hadoop 安装路径：

```
export JAVA_HOME=/opt/java    ##Java路径
export HADOOP_HOME=/opt/hadoop    ##Hadoop安装路径
export HIVE_HOME=/opt/hive    ##Hive安装路径
export HIVE_CONF_DIR=/opt/hive/conf    ##Hive配置文件路径
```

## 修改hive-site.xml

替换hive-site.xml文件中的 ${system:java.io.tmpdir} 和 ${system:user.name}

```
<property>
    <name>hive.exec.scratchdir</name>
    <value>/tmp/hive-${user.name}</value>
    <description>HDFS root scratch dir for Hive jobs which gets created with write all (733) permission. For each
connecting user, an HDFS scratch dir: ${hive.exec.scratchdir}/<username> is created, with
${hive.scratch.dir.permission}.</description>
  </property>
  <property>
    <name>hive.exec.local.scratchdir</name>
    <value>/tmp/${user.name}</value>
    <description>Local scratch space for Hive jobs</description>
  </property>
  <property>
    <name>hive.downloaded.resources.dir</name>
    <value>/tmp/hive/resources</value>
    <description>Temporary local directory for added resources in the remote file system.</description>
  </property>
<property>
    <name>hive.querylog.location</name>
    <value>/tmp/${user.name}</value>
    <description>Location of Hive run time structured log file</description>
  </property>
<property>
    <name>hive.server2.logging.operation.log.location</name>
    <value>/tmp/${user.name}/operation_logs</value>
    <description>Top level directory where operation logs are stored if logging functionality is
enabled</description>
  </property>
```

## 配置Hive Metastore

默认情况下, Hive的元数据保存在了内嵌的 derby 数据库里, 但一般情况下生产环境使用 MySQL 来存放 Hive 元数据。

1. 将 mysql-connector-java-5.1.39.jar 放入 $HIVE_HOME/lib 下。
2. hive-site.xml 中配置 MySQL 数据库连接信息

```
<property>
 <name>javax.jdo.option.ConnectionURL</name>
 <value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true&characterEncoding=UTF-
8&useSSL=false</value>
</property>
<property>
 <name>javax.jdo.option.ConnectionDriverName</name>
 <value>com.mysql.jdbc.Driver</value>
</property>
<property>
 <name>javax.jdo.option.ConnectionUserName</name>
 <value>hive</value>
</property>
<property>
 <name>javax.jdo.option.ConnectionPassword</name>
 <value>hive</value>
</property>
```

在 Hive 中创建表之前需要使用以下 HDFS 命令创建 /tmp 和 /user/hive/warehouse (hive-site.xml 配置文件中属性项 hive.metastore.warehouse.dir 的默认值) 目录并给它们赋写权限。

```
hdfs dfs -mkdir /tmp
hdfs dfs -mkdir /usr/hive/warehouse
hdfs dfs -chmod g+w /tmp
hdfs dfs -chmod g+w /usr/hive/warehouse
```

在命令行运行 hive 命令时必须保证 HDFS 已经启动。可以使用 start-dfs.sh 来启动 HDFS。我们需要先运行 schematool 命令来执行初始化操作。

```
schematool -dbType mysql -initSchema
```

要使用 Hive CLI（Hive command line interface），可以在终端输入以下命令:

```
hive
```