## 第5章  Hadoop基本操作——完全分布式

#### 1.实验步骤

##### 1.1  集群节点基本配置

步骤1.   **节点IP地址查询**

1.   在***节点1，节点2，节点3***中分别通过下面的命令查询节点IP地址：

ifconfig

需要记录IP地址，在后文中我们需要根据此IP地址进行相关操作。

步骤2.   **节点1主机名配置**

需要在***节点1***进行下列操作，将节点1的主机名配置为realtime-1。

1.   通过下列命令使用vi编辑器编辑主机名配置文件：

vi /etc/sysconfig/network

打开后的文件内容如下所示：

NETWORKING=yes

HOSTNAME=CentOS6.5      (注：需要将此行内容修改为实际的主机名realtime-1)

2.   在文件中进行内容更改，将HOSTNAME字段内容配置成realtime-1：

HOSTNAME=realtime-1

编辑完成后保存文件并退出vi编辑器。

3.     更改后的内容会在下次系统重启的时候生效，通过下列命令重新启动系统：

reboot

步骤3.   用相同办法进行***节点2、节点3***的**主机名配置**

将节点2的主机名配置为realtime-2，将节点3的主机名配置为realtime-3。

步骤4.   **节点1主机名与IP地址映射文件配置**

1.   在***节点1***中，通过下列命令使用vi编辑器编辑hosts文件：

vi /etc/hosts

打开后的文件内容如下所示：

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 (注：在此行增加内容)

::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

2.   增加节点1的IP地址与主机名的映射关系、节点2的IP地址与主机名的映射关系、节点3的IP地址与主机名的映射关系，IP地址与主机名之间用空格分隔（主机名填写为前文配置的节点实际主机名称，IP地址需要根据上文中的查询结果来进行填写，并与实际的主机名相对应）：

10.1.1.4 realtime-1

10.1.1.3 realtime-2

10.1.1.206 realtime-3

更改后的文件内容如下所示：

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4

::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.1.1.4 realtime-1

10.1.1.3 realtime-2

10.1.1.206 realtime-3

编辑完成后保存文件并退出vi编辑器。

3.   通过下列命令检测主机名与IP映射是否配置成功：

ping realtime-1 -c 2

ping realtime-2 -c 2

ping realtime-3 -c 2

***如果无法进行正常的报文发送，请检查主机名是否配置正确，同时请检查主机名与IP地址映射是否配置正确。***

步骤5. **相同方法在*节点2*、***节点3***中进行主机名与IP地址映射文件配置**

##### 1.2  配置SSH免密码登录

步骤1.   **节点1秘钥配置及分发**

需要在***节点1***进行下列操作，在节点1中生成秘钥文件，然后将公钥文件分发到节点2和节点3中，实现在节点1可以免密码登录到集群中的其他主机中。

1.   通过下面的命令在***节点1***中生成密钥（使用rsa加密方式）：

echo -e "\n"|ssh-keygen -t rsa -N "" >/dev/null 2>&1

默认情况下会在~/.ssh/文件夹下生成公钥文件id\_rsa.pub和私钥文件id\_rsa，通过下面的命令对~/.ssh/内容进行查看：

ll ~/.ssh/

2.   通过下面的命令将公钥文件发送到***本机***，创建root免密钥通道（需要输入密码：111111）：

ssh-copy-id -i /root/.ssh/id\_rsa.pub root@realtime-1

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# ssh-copy-id -i /root/.ssh/id\_rsa.pub root@realtime-1

The authenticity of host 'realtime-1 (10.1.1.4)' can't be established.

RSA key fingerprint is 9f:3b:30:10:65:46:c9:c3:2b:fb:e5:28:38:39:9c:84.

Are you sure you want to continue connecting (yes/no)? yes    (注：此处需要输入yes)

Warning: Permanently added 'realtime-1,10.1.1.4' (RSA) to the list of known hosts.

root@realtime-1's password:                      （注：此处需要输入root用户密码，为111111）

Now try logging into the machine, with "ssh 'root@realtime-1'", and check in:

  .ssh/authorized\_keys

to make sure we haven't added extra keys that you weren't expecting.

[root@realtime-1 ~]#

3.   通过下面的命令将公钥文件发送到***节点2***，创建root免密钥通道（需要输入密码：111111）：

ssh-copy-id -i /root/.ssh/id\_rsa.pub root@realtime-2

4.   同理将公钥文件发送到***节点3***，创建root免密钥通道（需要输入密码：111111）：

步骤2.   **同上面方法，对节点2、节点3秘钥配置及分发**

步骤3.   **SSH免密码登录测试**

集群中各个节点秘钥分发完毕后，可以通过ssh远程登录命令来测试免密码登录是否配置成功。为了操作统一，我们在***节点3***中进行下面的操作（在其他节点操作所实现的效果也是一样的）。

1.   在节点3中通过下面的命令可以实现免密码远程登录到节点1：

ssh realtime-1

命令运行后的返回结果如下所示：

[root@realtime-3 ~]# ssh realtime-1

Last login: Thu Nov 29 14:08:34 2018 from realtime-3

[root@realtime-1 ~]#

2.   登录到节点1后，通过下面的命令可以实现免密码远程登录到节点2：

ssh realtime-2

3.   登录到节点2后，通过下面的命令可以实现免密码远程登录到节点3：

ssh realtime-3

##### 1.3  安装配置JDK1.8

JDK需要在集群3个节点都进行安装，为了操作方便，我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

**在模板中，我们已经将相应的JDK安装包jdk-8u60-linux-x64.tar.gz放到了/usr/software/目录下，同学们不需要再次下载，可以直接使用。**

步骤1.   **创建工作路径**

1.   首先需要在终端中输入下列命令，在/usr目录下建立cx工作路径：

mkdir /usr/cx

2.   通过下面的命令实现在节点2的/usr目录下建立cx工作路径：

ssh realtime-2 "mkdir /usr/cx"

3.   通过下面的命令实现在节点3的/usr目录下建立cx工作路径：

ssh realtime-3 "mkdir /usr/cx"

步骤2.   **解压安装包**

1.   我们可以在/usr/software/目录下找到jdk-8u60-linux-x64.tar.gz安装包，通过下列命令将其解压到/usr/cx/目录下：

tar -zxvf /usr/software/jdk-8u60-linux-x64.tar.gz -C /usr/cx

2.   通过下列命令实现在节点2中将jdk-8u60-linux-x64.tar.gz安装包解压到/usr/cx/目录下：

ssh realtime-2 "tar -zxvf /usr/software/jdk-8u60-linux-x64.tar.gz -C /usr/cx"

3.   通过下列命令实现在节点3中将jdk-8u60-linux-x64.tar.gz安装包解压到/usr/cx/目录下：

ssh realtime-3 "tar -zxvf /usr/software/jdk-8u60-linux-x64.tar.gz -C /usr/cx"

步骤3.   **配置环境变量**

1.   通过下列命令使用vi编辑器打开 ~/.bashrc文件：

vi ~/.bashrc

打开的~/.bashrc文件内容如下所示：

# .bashrc

# User specific aliases and functions

alias rm='rm -i'

alias cp='cp -i'

alias mv='mv -i'

# Source global definitions

if [ -f /etc/bashrc ]; then

        . /etc/bashrc

fi

(----------------注：需要在此处增加内容-------------------)

2.   在文件中写入下列内容：

export JAVA\_HOME=/usr/cx/jdk1.8.0\_60

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

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

编辑完成后保存文件并退出vi编辑器。

3.   通过下面的命令将环境变量配置文件分发到节点2：

scp ~/.bashrc [root@realtime-2:~/.bashrc](mailto:root@realtime-2:~/.bashrc)

命令执行后的输出内容如下所示：

[root@realtime-1 ~]# scp ~/.bashrc root@realtime-2:~/.bashrc

.bashrc                                      100%  320     0.3KB/s   00:00

[root@realtime-1 ~]#

4.   通过下面的命令将环境变量配置文件分发到节点3：

scp ~/.bashrc [root@realtime-3:~/.bashrc](mailto:root@realtime-3:~/.bashrc)

步骤4.   **更新环境变量**

1.   执行如下命令，更新环境变量：

source  ~/.bashrc

2.   执行如下命令，更新节点2的环境变量：

ssh realtime-2 "source  ~/.bashrc"

3.   执行如下命令，更新节点3的环境变量：

ssh realtime-3 "source  ~/.bashrc"

步骤5.   **验证JDK是否配置成功**

1.   通过下面的命令验证JDK是否安装并配置成功：

java -version

如果出现如下JDK版本信息，则说明安装配置成功：

2.   通过下面的命令验证节点2的JDK是否安装并配置成功：

ssh realtime-2 "java -version"

如果出现如下JDK版本信息，则说明安装配置成功：

3.   通过下面的命令验证节点3的JDK是否安装并配置成功：

ssh realtime-3 "java -version"

***如果没有正确输出相关版本信息，请检查~/.bashrc文件中的JDK环境变量是否配置正确，同时请确定是否使用source  ~/.bashrc命令更新环境变量配置。***

##### 1.4  NTP服务配置

需要在集群的3台节点中都进行NTP服务的配置。

步骤1.   **NTP服务配置**

1.   在***节点1***中通过下面的命令打开NTP配置文件：

vi /etc/ntp.conf

打开后的文件内容如下所示：

（---------------省略----------------）

# Permit all access over the loopback interface.  This could

# be tightened as well, but to do so would effect some of

# the administrative functions.

restrict 127.0.0.1

restrict -6 ::1

# Hosts on local network are less restricted.

#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.

# Please consider joining the pool (http://www.pool.ntp.org/join.html).

server 0.centos.pool.ntp.org iburst  （注：注释此行内容）

server 1.centos.pool.ntp.org iburst  （注：注释此行内容）

server 2.centos.pool.ntp.org iburst  （注：注释此行内容）

server 3.centos.pool.ntp.org iburst  （注：注释此行内容）

（注：在此处增加内容）

#broadcast 192.168.1.255 autokey        # broadcast server

（---------------省略----------------）

在文件中进行下列内容更改（通过server字段设置本机为NTP Serevr服务器，通过restrict限制realtime-2和realtime-3主机名对应的主机可以同步时间）：

#server 0.centos.pool.ntp.org iburst

#server 1.centos.pool.ntp.org iburst

#server 2.centos.pool.ntp.org iburst

#server 3.centos.pool.ntp.org iburst

server 127.127.1.0

fudge 127.127.1.0 stratum 10

restrict realtime-2 nomodify notrap

restrict realtime-3 nomodify notrap

更改完成后保存文件并退出编辑器。

2.   在***节点2***中通过下面的命令打开NTP配置文件：

vi /etc/ntp.conf

打开后的文件内容如下所示：

（---------------省略----------------）

# Permit all access over the loopback interface.  This could

# be tightened as well, but to do so would effect some of

# the administrative functions.

restrict 127.0.0.1

restrict -6 ::1

# Hosts on local network are less restricted.

#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.

# Please consider joining the pool (http://www.pool.ntp.org/join.html).

server 0.centos.pool.ntp.org iburst  （注：注释此行内容）

server 1.centos.pool.ntp.org iburst  （注：注释此行内容）

server 2.centos.pool.ntp.org iburst  （注：注释此行内容）

server 3.centos.pool.ntp.org iburst  （注：注释此行内容）

（注：在此处增加内容）

#broadcast 192.168.1.255 autokey        # broadcast server

（---------------省略----------------）

在文件中进行下列内容更改（通过iburst字段指明通过主机名为realtime-1的服务器进行时间同步）：

#server 0.centos.pool.ntp.org iburst

#server 1.centos.pool.ntp.org iburst

#server 2.centos.pool.ntp.org iburst

#server 3.centos.pool.ntp.org iburst

server realtime-1  iburst

更改完成后保存文件并退出编辑器。

3.   ***节点3***中同***节点2***操作。

步骤2.   **启动NTP服务**

为了操作方便，我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

1.   通过下面的命令在节点1中设定NTP服务自启动：

chkconfig ntpd on

2.   通过下面的命令在节点1中启动NTP服务：

service ntpd start

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# service ntpd start

正在启动 ntpd：                                            [确定]

[root@realtime-1 ~]#

3.   通过下面的命令在节点2中设定NTP服务自启动：

ssh realtime-2 "chkconfig ntpd on"

4.   通过下面的命令在节点2中启动NTP服务：

ssh realtime-2 "service ntpd start"

5.   通过下面的命令在节点3中设定NTP服务自启动：

ssh realtime-3 "chkconfig ntpd on"

6.   通过下面的命令在节点3中启动NTP服务：

ssh realtime-3 "service ntpd start"

***如果服务无法正常启动，会出现相关的错误提示信息，只需要根据错误提示进行更改即可。***

步骤3.   **NTP服务状态查看**

为了操作方便，我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

1.   通过下面的命令查看节点1中NTP服务的运行状态：

ntpstat

命令运行后的返回结果如下所示（由于节点1是作为Server节点，所以其状态会很快变成synchronised，此时说明服务已经正常启动）：

[root@realtime-1 ~]# ntpstat

synchronised to local net at stratum 11

   time correct to within 449 ms

   polling server every 64 s

[root@realtime-1 ~]#

2.   通过下面的命令查看节点2与节点3中NTP服务的运行状态：

ssh realtime-2 "ntpstat"

命令运行后的返回结果如下所示（由于节点2需要同步节点1的时间，因此需要大概15分钟其状态才会由unsynchronised会变成synchronised，当状态变为synchronised时说明服务已经正常启动）：

3．***节点3***同***节点2***操作。

***当3个节点的状态都显示为synchronised时，则表示ntp服务已经启动成功；如果一直显示unsynchronised,可能是配置文件有错误，因此需要检查IP地址是否配置正确。***

同学们不必一直等待，可以先进行下文的实验，然后过后再查看NTP服务状态。

##### 1.5  SElinux安全配置

需要在集群3个节点都进行SElinux配置，为了操作方便，我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

1.   通过下面的命令，关闭节点1的SElinux安全设置：

/bin/sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config

2.   通过下面的命令，关闭节点2的SElinux安全设置：

ssh realtime-2 "/bin/sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config"

3.   通过下面的命令，关闭节点3的SElinux安全设置：

ssh realtime-3 "/bin/sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config"

##### 1.6  安装配置ZooKeeper集群

由于我们需要搭建一套具备高可用性的Hadoop集群，因此需要通过ZooKeeper来进行集群中服务的协调。ZooKeeper需要在集群3个节点进行安装配置，为了操作方便，我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

**在模板中我们已经将ZooKeeper安装文件zookeeper-3.4.6.tar.gz放到了/usr/software目录下，同学们可以直接使用。**

步骤1.   **解压安装包**

1.   通过下列命令将ZooKeeper安装包解压到/usr/cx目录下：

tar -zxvf /usr/software/zookeeper-3.4.6.tar.gz -C /usr/cx

2.   解压完成后，我们可以查看解压后的文件夹内容：

ls /usr/cx/zookeeper-3.4.6/

步骤2.   **数据存储目录创建**

1.   通过下面的命令创建ZooKeeper数据存储目录：

mkdir -p /home/data

通过下面的命令创建ZooKeeper日志存储目录：

mkdir -p /home/logs

2.   通过下面的命令在节点2中创建ZooKeeper数据存储目录：

ssh realtime-2 "mkdir -p /home/data"

通过下面的命令在节点2中创建ZooKeeper日志存储目录：

ssh realtime-2 "mkdir -p /home/logs"

3.   ***节点3***中同***节点2***操作。

步骤3.   **主机myid编号文件创建**

1.   通过下面的命令创建myid文件，并设置节点1对应的编号为1（集群启动后会通过此编号来进行主机识别）：

echo "1" > /home/data/myid

2.   通过下面的命令在节点2中创建myid文件，并设置节点2对应的编号为2（集群启动后会通过此编号来进行主机识别）：

ssh realtime-2 "echo "2" > /home/data/myid"

3.   通过下面的命令在节点3中创建myid文件，并设置节点3对应的编号为3（集群启动后会通过此编号来进行主机识别）：

ssh realtime-3 "echo "3" > /home/data/myid"

步骤4.   **ZooKeeper配置文件编辑**

1.   通过下列命令创建并打开zoo.cfg配置文件：

vi /usr/cx/zookeeper-3.4.6/conf/zoo.cfg

在文件中写入下列内容：

tickTime=2000

dataDir=/home/data

clientPort=2181

dataLogDir=/home/logs

initLimit=5

syncLimit=2

server.1=realtime-1:2888:3888

server.2=realtime-2:2888:3888

server.3=realtime-3:2888:3888

编辑完成后保存文件并退出vi编辑器。

在上述配置中，我们设置心跳时间为2000毫秒，设置ZooKeeper在本地保存数据的目录为/home/data，ZooKeeper监听客户端连接的端口为2181,设置所有Follower和Leader进行同步的时间为5s，设置一个Follower和Leader进行同步的时间为2s。同时设定集群中有3台主机，其中realtime-1对应的主机编号为1，Follower与Leader之间交换信息的端口为2888，进行Leader选举的端口为3888；realtime-2对应的主机编号为2，Follower与Leader之间交换信息的端口为2888，进行Leader选举的端口为3888；realtime-3对应的主机编号为3，Follower与Leader之间交换信息的端口为2888，进行Leader选举的端口为3888。

步骤5.   **文件分发**

1.   通过下面的命令将节点1的ZooKeeper文件包分发到节点2中：

scp -r /usr/cx/zookeeper-3.4.6 root@realtime-2:/usr/cx/

2.   通过下面的命令将节点1的ZooKeeper文件包分发到节点3中：

scp -r /usr/cx/zookeeper-3.4.6 root@realtime-3:/usr/cx/

步骤6.   **ZooKeeper环境变量配置**

1.   通过下列命令使用vi编辑器打开 ~/.bashrc文件：

vi ~/.bashrc

打开的~/.bashrc文件内容如下所示：

# .bashrc

# User specific aliases and functions

alias rm='rm -i'

alias cp='cp -i'

alias mv='mv -i'

# Source global definitions

if [ -f /etc/bashrc ]; then

        . /etc/bashrc

fi

export JAVA\_HOME=/usr/cx/jdk1.8.0\_60

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

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

(----------------注：需要在此处增加内容-------------------)

2.   在文件中写入下列内容：

export ZK\_HOME=/usr/cx/zookeeper-3.4.6

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

编辑完成后保存文件并退出vi编辑器。

3.   通过下面的命令将环境变量配置文件分发到节点2：

scp ~/.bashrc [root@realtime-2:~/.bashrc](mailto:root@realtime-2:~/.bashrc)

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# scp ~/.bashrc root@realtime-2:~/.bashrc

.bashrc                                       100%  391     0.4KB/s   00:00

[root@realtime-1 ~]#

4.   ***节点3***中同***节点2***操作。

步骤7.   **更新环境变量**

1.   执行如下命令，更新环境变量：

source  ~/.bashrc

2.   执行如下命令，更新节点2的环境变量：

ssh realtime-2 "source  ~/.bashrc"

3.   ***节点3***中同***节点2***操作。

步骤8.   **验证环境变量是否配置成功**

1.   通过下面的命令验证环境变量是否配置成功：

zkServer.sh

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# zkServer.sh

JMX enabled by default

Using config: /usr/cx/zookeeper-3.4.6/bin/../conf/zoo.cfg

Usage: /usr/cx/zookeeper-3.4.6/bin/zkServer.sh {start|start-foreground|stop|restart|status|upgrade|print-cmd}

[root@realtime-1 ~]#

由输出内容可以看出，ZooKeeper环境变量已经配置正确，并且可以正常执行。

2.   通过下面的命令验证节点2的环境变量是否配置成功：

ssh realtime-2 "zkServer.sh"

3.   ***节点3***中同***节点2***操作。

##### 1.7  ZooKeeper启动及状态查看

步骤1.   **ZooKeeper启动**

1.   通过下面的命令启动ZooKeeper服务：

zkServer.sh start

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# zkServer.sh start

JMX enabled by default

Using config: /usr/cx/zookeeper-3.4.6/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

[root@realtime-1 ~]#

2.   通过下面的命令在节点2中启动ZooKeeper服务：

ssh realtime-2 "zkServer.sh start"

3.   ***节点3***中同***节点2***操作。

步骤2.   **ZooKeeper运行状态查看**

ZooKeeper运行之后会随机进行follower角色以及leader角色选举，当leader角色节点出现异常后，会从其他节点中选举出新的leader角色。

1.   通过下面的命令可以查看节点1的ZooKeeper运行状态：

zkServer.sh status

命令运行后的返回结果如下所示（由返回结果的Mode字段可以看出，当前节点是作为follower角色运行的）：

[root@realtime-1 ~]# zkServer.sh status

JMX enabled by default

Using config: /usr/cx/zookeeper-3.4.6/bin/../conf/zoo.cfg

Mode: follower

[root@realtime-1 ~]#

2.   通过下面的命令可以查看节点2的ZooKeeper运行状态：

ssh realtime-2 "zkServer.sh status"

##### 3.*节点3*中同*节点2*操作。

##### 1.8  安装配置Hadoop集群

Hadoop需要在集群3个节点进行安装配置，为了操作方便，我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

**在模板中，我们已经将相应的Hadoop安装包hadoop-2.7.1.tar.gz放到/usr/software/目录下，同学们不需要再次下载，可以直接使用。**

步骤1.   **数据存储目录创建**

1.   执行下面的命令创建Hadoop的元数据存储目录namenode：

mkdir -p /hdfs/namenode

2.   执行下面的命令创建Hadoop的数据存储目录datanode：

mkdir -p /hdfs/datanode

3.   执行下面的命令创建JournalNode的数据存储目录journalnode：

mkdir -p /hdfs/journalnode

4.   执行下面的命令创建任务调度的日志存储目录hadoop-yarn：

mkdir -p /var/log/hadoop-yarn

5.   执行下面的命令在节点2中创建Hadoop的元数据存储目录namenode：

ssh realtime-2 "mkdir -p /hdfs/namenode"

6.   执行下面的命令在节点2中创建Hadoop的数据存储目录datanode：

ssh realtime-2 "mkdir -p /hdfs/datanode"

7.   执行下面的命令在节点2中创建JournalNode的数据存储目录journalnode：

ssh realtime-2 "mkdir -p /hdfs/journalnode"

8.   执行下面的命令在节点2中创建任务调度的日志存储目录hadoop-yarn：

ssh realtime-2 "mkdir -p /var/log/hadoop-yarn"

9.   执行下面的命令在节点3中创建Hadoop的元数据存储目录namenode：

ssh realtime-3 "mkdir -p /hdfs/namenode"

10.  执行下面的命令在节点3中创建Hadoop的数据存储目录datanode：

ssh realtime-3 "mkdir -p /hdfs/datanode"

11.  执行下面的命令在节点3中创建JournalNode的数据存储目录journalnode：

ssh realtime-3 "mkdir -p /hdfs/journalnode"

12.  执行下面的命令在节点3中创建任务调度的日志存储目录hadoop-yarn：

ssh realtime-3 "mkdir -p /var/log/hadoop-yarn"

步骤2.   **解压安装文件**

通过下列命令解压Hadoop安装文件，将文件解压到/usr/cx目录下：

tar -zxvf /usr/software/hadoop-2.7.1.tar.gz -C /usr/cx

步骤3.   **编辑Hadoop配置文件：**

1.   使用vi命令打开hadoop-env.sh配置文件进行编辑：

vi /usr/cx/hadoop-2.7.1/etc/hadoop/hadoop-env.sh

打开后的文件内容如下所示：

(-------------------省略------------------------)

# Set Hadoop-specific environment variables here.

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

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

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

# remote nodes.

# The java implementation to use.

export JAVA\_HOME=${JAVA\_HOME}    (注：需要对此行内容进行更改，为Hadoop绑定Java运行环境)

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

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

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

# data transfer protocol using non-privileged ports.

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

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

(-------------------省略------------------------)

在文件中进行下列内容更改，将JAVA\_HOME对应的值改成实际的JDK安装路径：

export  JAVA\_HOME=/usr/cx/jdk1.8.0\_60

编辑完成后保存文件并退出vi编辑器。

2.   使用vi命令打开hdfs-site.xml文件进行配置：

vi /usr/cx/hadoop-2.7.1/etc/hadoop/hdfs-site.xml

打开后的文件内容如下所示：

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

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

<!--

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

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

  You may obtain a copy of the License at

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

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

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

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

  See the License for the specific language governing permissions and

  limitations under the License. See accompanying LICENSE file.

-->

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

<configuration>

(注：需要在此处进行相关内容配置)

</configuration>

在文件中<configuration>和</configuration>之间增加下列内容：

/\*配置DataNode的数据存储目录，需要与上文创建的目录相对应\*/

<property>

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

<value>/hdfs/datanode</value>

</property>

/\*配置数据块大小为256M\*/

<property>

<name>dfs.blocksize</name>

<value>268435456</value>

</property>

/\*自定义的HDFS服务名，在高可用集群中，无法配置单一HDFS服务器入口，所以需要指定一个逻辑上的服务名，当访问服务名时，会自动选择NameNode节点进行访问\*/

<property>

<name>dfs.nameservices</name>

<value>HDFScluster</value>

</property>

/\*配置NameNode的数据存储目录，需要与上文创建的目录相对应\*/

<property>

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

<value>/hdfs/namenode</value>

</property>

/\*定义HDFS服务名所指向的NameNode主机名称\*/

<property>

<name>dfs.ha.namenodes.HDFScluster</name>

<value>realtime-1,realtime-2</value>

</property>

/\*设置NameNode的完整监听地址\*/

<property>

<name>dfs.namenode.rpc-address.HDFScluster.realtime-1</name>

<value>realtime-1:8020</value>

</property>

/\*设置NameNode的完整监听地址\*/

<property>

<name>dfs.namenode.rpc-address.HDFScluster.realtime-2</name>

<value>realtime-2:8020</value>

</property>

/\*设置NameNode的HTTP访问地址\*/

<property>

<name>dfs.namenode.http-address.HDFScluster.realtime-1</name>

<value>realtime-1:50070</value>

</property>

/\*设置NameNode的HTTP访问地址\*/

<property>

<name>dfs.namenode.http-address.HDFScluster.realtime-2</name>

<value>realtime-2:50070</value>

</property>

/\*设置主从NameNode元数据同步地址，官方推荐将nameservice作为最后的journal ID\*/

<property>

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

<value>qjournal://realtime-1:8485;realtime-2:8485;realtime-3:8485/HDFScluster</value>

</property>

/\*设置HDFS客户端用来连接集群中活动状态NameNode节点的Java类\*/

<property>

<name>dfs.client.failover.proxy.provider.HDFScluster</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

/\*设置SSH登录的私钥文件地址\*/

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_rsa</value>

</property>

/\*启动fence过程，确保集群高可用性\*/

<property>

<name>dfs.ha.fencing.methods</name>

<value>shell(/bin/true)</value>

</property>

/\*配置JournalNode的数据存储目录，需要与上文创建的目录相对应\*/

<property>

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

<value>/hdfs/journalnode</value>

</property>

/\*设置自动切换活跃节点，保证集群高可用性\*/

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

/\*配置数据块副本数\*/

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

/\*将dfs.webhdfs.enabled属性设置为true，否则就不能使用webhdfs的LISTSTATUS、LIST FILESTATUS等需要列出文件、文件夹状态的命令，因为这些信息都是由namenode保存的\*/

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

编辑完成后保存文件并退出vi编辑器。

3.   使用vi命令打开core-site.xml配置文件进行编辑：

vi /usr/cx/hadoop-2.7.1/etc/hadoop/core-site.xml

打开后的文件内容如下所示：

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

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

<!--

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

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

  You may obtain a copy of the License at

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

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

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

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

  See the License for the specific language governing permissions and

  limitations under the License. See accompanying LICENSE file.

-->

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

<configuration>

(注：需要在此处进行相关内容配置)

</configuration>

在文件中<configuration>和</configuration>之间增加下列内容：

/\*设置默认的HDFS访问路径，需要与hdfs-site.xml中的HDFS服务名相一致\*/

<property>

<name>fs.defaultFS</name>

<value>hdfs://HDFScluster</value>

</property>

/\*临时文件夹路径设置\*/

<property>

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

<value>/usr/tmp</value>

</property>

/\*配置ZooKeeper服务集群，用于活跃NameNode节点的选举\*/

<property>

<name>ha.zookeeper.quorum</name>

<value>realtime-1:2181,realtime-2:2181,realtime-3:2181</value>

</property>

/\*设置数据压缩算法\*/

<property>

<name>io.compression.codecs</name>

<value>org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.BZip2Codec,com.hadoop.compression.lzo.LzoCodec,com.hadoop.compression.lzo.LzopCodec,org.apache.hadoop.io.compress.SnappyCodec</value>

</property>

<property>

<name>io.compression.codec.lzo.class</name>

<value>com.hadoop.compression.lzo.LzoCodec</value>

</property>

/\*设置使用hduser用户可以代理所有主机用户进行任务提交\*/

<property>

<name>hadoop.proxyuser.hduser.host</name>

<value>\*</value>

</property>

/\*设置使用hduser用户可以代理所有组用户进行任务提交\*/

<property>

<name>hadoop.proxyuser.hduser.groups</name>

<value>\*</value>

</property>

编辑完成后保存文件并退出vi编辑器。

4.   使用vi命令打开yarn-site.xml文件进行配置：

vi /usr/cx/hadoop-2.7.1/etc/hadoop/yarn-site.xml

打开后的文件内容如下所示：

<?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>

(注：需要在此处进行相关内容配置)

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

</configuration>

在文件中<configuration>和</configuration>之间增加下列内容：

/\*设置NodeManager上运行的附属服务，需配置成mapreduce\_shuffle才可运行MapReduce程序\*/

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce\_shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

/\*设置任务日志存储目录\*/

<property>

<name>yarn.nodemanager.log-dirs</name>

<value>file:///var/log/hadoop-yarn </value>

</property>

/\*设置Hadoop依赖包地址\*/

<property>

<name>yarn.application.classpath</name>

<value>

$HADOOP\_HOME/share/hadoop/common/\*,$HADOOP\_HOME/share/hadoop/common/lib/\*,

$HADOOP\_HOME/share/hadoop/hdfs/\*,$HADOOP\_HOME/share/hadoop/hdfs/lib/\*,

$HADOOP\_HOME/share/hadoop/mapreduce/\*,$HADOOP\_HOME/share/hadoop/mapreduce/lib/\*,

$HADOOP\_HOME/share/hadoop/yarn/\*,$HADOOP\_HOME/share/hadoop/yarn/lib/\*

</value>

</property>

/\*开启resourcemanager 的高可用性功能\*/

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

/\*标识集群中的resourcemanager，如果设置该选项，需要确保所有的resourcemanager节点在配置中都有自己的逻辑id\*/

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>YARNcluster</value>

</property>

/\*设置resourcemanager节点的逻辑id\*/

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

</property>

/\*为每个逻辑id绑定实际的主机名称\*/

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>realtime-1</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>realtime-2</value>

</property>

/\*指定ZooKeeper服务地址\*/

<property>

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

<value>realtime-1:2181,realtime-2:2181,realtime-3:2181</value>

</property>

/\*指定resourcemanager的WEB访问地址\*/

<property>

<name>yarn.resourcemanager.webapp.address.rm1</name>

<value>realtime-1:8089</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm2</name>

<value>realtime-2:8089</value>

</property>

/\*设定虚拟内存与实际内存的比例，比例值越高，则可用虚拟内存就越多\*/

<property>

<name>yarn.nodemanager.vmem-pmem-ratio</name>

<value>3</value>

</property>

/\*设定单个容器可以申领到的最小内存资源\*/

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>32</value>

</property>

/\*设置当任务运行结束后，日志文件被转移到的HDFS目录\*/

<property>

<name>yarn.nodemanager.remote-app-log-dir</name>

<value>hdfs://HDFScluster/var/log/hadoop-yarn/apps</value>

</property>

/\*设定资源调度策略，目前可用的有FIFO、Capacity Scheduler和Fair Scheduler\*/

<property>

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

<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>

</property>

/\*设定每个任务能够申领到的最大虚拟CPU数目\*/

<property>

<name>yarn.scheduler.maximum-allocation-vcores</name>

<value>8</value>

</property>

/\*设置任务完成指定时间（秒）之后，删除任务的本地化文件和日志目录\*/

<property>

<name>yarn.nodemanager.delete.debug-delay-sec</name>

<value>600</value>

</property>

/\*设置志在HDFS上保存多长时间（秒）\*/

<property>

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

<value>86400</value>

</property>

/\*设定物理节点有2G内存加入资源池\*/

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>2048</value>

</property>

编辑完成后保存文件并退出vi编辑器。

5.   使用下列命令复制mapred-site.xml.template文件并重命名为mapred-site.xml：

cp /usr/cx/hadoop-2.7.1/etc/hadoop/mapred-site.xml.template /usr/cx/hadoop-2.7.1/etc/hadoop/mapred-site.xml

6.   使用vi命令打开mapred-site.xml文件进行配置：

vi /usr/cx/hadoop-2.7.1/etc/hadoop/mapred-site.xml

打开后的文件内容如下所示：

<?xml version="1.0"?>

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

<!--

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

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

  You may obtain a copy of the License at

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

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

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

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

  See the License for the specific language governing permissions and

  limitations under the License. See accompanying LICENSE file.

-->

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

<configuration>

(注：需要在此处进行相关内容配置)

</configuration>

在文件中<configuration>和</configuration>之间增加下列内容：

/\*Hadoop对MapReduce运行框架一共提供了3种实现，在mapred-site.xml中通过"mapreduce.framework.name"这个属性来设置为"classic"、"yarn"或者"local"\*/

<property>

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

<value>yarn</value>

</property>

编辑完成后保存文件并退出vi编辑器。

7.   使用vi命令打开slaves文件进行配置（要与我们前文设置的主机名相互一致，否则将会引起Hadoop相关进程无法正确启动）：

vi /usr/cx/hadoop-2.7.1/etc/hadoop/slaves

打开后的文件内容如下所示：

localhost        （注：需要对此内容进行更改，配置为Slave节点的实际主机名）

将文件中的内容更改为下列内容：

realtime-1

realtime-2

realtime-3

编辑完成后保存文件并退出vi编辑器。

步骤4.   **文件分发**

1.   通过下面的命令将节点1的Hadoop文件包分发到节点2中：

scp -r /usr/cx/hadoop-2.7.1 root@realtime-2:/usr/cx/

命令运行后的返回结果如下所示：

（---------------------省略-----------------------）

external.png                                  100%  230     0.2KB/s   00:00

banner.jpg                                    100%  872     0.9KB/s   00:00

maven-feather.png                             100% 3330     3.3KB/s   00:00

build-by-maven-white.png                      100% 2260     2.2KB/s   00:00

build-by-maven-black.png                      100% 2294     2.2KB/s   00:00

bg.jpg                                        100%  486     0.5KB/s   00:00

icon\_error\_sml.gif                            100% 1010     1.0KB/s   00:00

logo\_apache.jpg                               100%   33KB  32.7KB/s   00:00

collapsed.gif                                 100%  820     0.8KB/s   00:00

apache-maven-project-2.png                    100%   33KB  32.7KB/s   00:00

icon\_success\_sml.gif                          100%  990     1.0KB/s   00:00

icon\_info\_sml.gif                             100%  606     0.6KB/s   00:00

h3.jpg                                        100%  431     0.4KB/s   00:00

maven-logo-2.gif                              100%   26KB  25.8KB/s   00:00

h5.jpg                                        100%  357     0.4KB/s   00:00

newwindow.png                                 100%  220     0.2KB/s   00:00

icon\_warning\_sml.gif                          100%  576     0.6KB/s   00:00

expanded.gif                                  100%   52     0.1KB/s   00:00

dependency-analysis.html                      100%   21KB  21.3KB/s   00:00

[root@realtime-1 ~]#

2.   通过下面的命令将节点1的Hadoop文件包分发到节点3中，***节点3***同***节点2***操作。

步骤5.   **配置Hadoop环境变量**

1.   通过下列命令使用vi编辑器编辑~/.bashrc文件：

vi  ~/.bashrc

打开后的文件内容如下所示：

# .bashrc

# User specific aliases and functions

alias rm='rm -i'

alias cp='cp -i'

alias mv='mv -i'

# Source global definitions

if [ -f /etc/bashrc ]; then

        . /etc/bashrc

fi

export JAVA\_HOME=/usr/cx/jdk1.8.0\_60

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

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

export ZK\_HOME=/usr/cx/zookeeper-3.4.6

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

(----------------在此处增加内容-------------------)

2.   在~/.bashrc文件中增加以下内容：

export HADOOP\_HOME=/usr/cx/hadoop-2.7.1

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

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

编辑完成后保存文件并退出vi编辑器。

3.   通过下面的命令将节点1的环境变量文件分发到节点2中：

scp -r ~/.bashrc root@realtime-2:~/.bashrc

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# scp -r ~/.bashrc root@realtime-2:~/.bashrc

.bashrc                                       100%  502     0.5KB/s   00:00

[root@realtime-1 ~]#

4.   通过下面的命令将节点1的环境变量文件分发到节点3中，***节点3***同***节点2***操作。：

步骤6.   **更新环境变量**

1.   执行如下命令，更新环境变量：

source  ~/.bashrc

2.   执行如下命令，更新节点2的环境变量：

ssh realtime-2 "source  ~/.bashrc"

3.   更新节点3的环境变量，***节点3***同***节点2***操作。：

步骤7.   **验证Hadoop环境变量是否配置成功**

1.   通过下列命令验证Hadoop环境变量是否配置成功：

hadoop

如果出现如下提示信息，则说明Hadoop安装配置成功：

[root@realtime-1 ~]# hadoop

Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]

  CLASSNAME            run the class named CLASSNAME

 or

  where COMMAND is one of:

  fs                   run a generic filesystem user client

  version              print the version

  jar <jar>            run a jar file

                       note: please use "yarn jar" to launch

                             YARN applications, not this command.

  checknative [-a|-h]  check native hadoop and compression libraries availability

  distcp <srcurl> <desturl> copy file or directories recursively

  archive -archiveName NAME -p <parent path> <src>\* <dest> create a hadoop archive

  classpath            prints the class path needed to get the

  credential           interact with credential providers

                       Hadoop jar and the required libraries

  daemonlog            get/set the log level for each daemon

  trace                view and modify Hadoop tracing settings

Most commands print help when invoked w/o parameters.

[root@realtime-1 ~]#

2.   通过下列命令验证节点2的Hadoop环境变量是否配置成功：

ssh realtime-2 "hadoop"

3.   验证节点3的Hadoop环境变量是否配置成功，***节点3***同***节点2***操作。：

***如果没有正确输出相关信息，请检查~/.bashrc文件中的Hadoop环境变量是否配置正确，同时请确定是否使用source  ~/.bashrc命令更新环境变量配置。***

步骤8.   **格式化HDFS**

通过下列命令格式化HDFS文件系统（***如果格式化失败，会有相关的错误日志输出，根据输出内容进行更改即可***）：

hadoop namenode -format

命令运行后的部分显示内容如下所示：

(-------------------省略------------------------)

18/11/30 11:07:15 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25

18/11/30 11:07:15 INFO namenode.FSNamesystem: Retry cache on namenode is enabled

18/11/30 11:07:15 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis

18/11/30 11:07:15 INFO util.GSet: Computing capacity for map NameNodeRetryCache

18/11/30 11:07:15 INFO util.GSet: VM type       = 64-bit

18/11/30 11:07:15 INFO util.GSet: 0.029999999329447746% max memory 966.7 MB = 297.0 KB

18/11/30 11:07:15 INFO util.GSet: capacity      = 2^15 = 32768 entries

18/11/30 11:07:16 INFO namenode.FSImage: Allocated new BlockPoolId: BP-348760827-10.1.1.4-1543547236332

18/11/30 11:07:16 INFO common.Storage: Storage directory /hdfs/namenode has been successfully formatted.

18/11/30 11:07:16 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0

18/11/30 11:07:16 INFO util.ExitUtil: Exiting with status 0

18/11/30 11:07:16 INFO namenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHUTDOWN\_MSG: Shutting down NameNode at realtime-1/10.1.1.4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

[root@realtime-1 ~]#

步骤9.   **格式化zkfc元数据**

格式化DFSZKFailoverController(ZKFC)元数据（在一个节点中进行处理即可）：

hdfs zkfc -formatZK

命令运行后的返回结果如下所示：

（---------------省略------------------）

tString=realtime-1:2181,realtime-2:2181,realtime-3:2181 sessionTimeout=5000 watcher=org.apache.hadoop.ha.ActiveStandbyElector$WatcherWithClientRef@10e31a9a

18/11/30 11:37:46 INFO zookeeper.ClientCnxn: Opening socket connection to server realtime-2/10.1.1.3:2181. Will not attempt to authenticate using SASL (unknown error)

18/11/30 11:37:47 INFO zookeeper.ClientCnxn: Socket connection established to realtime-2/10.1.1.3:2181, initiating session

18/11/30 11:37:47 INFO zookeeper.ClientCnxn: Session establishment complete on server realtime-2/10.1.1.3:2181, sessionid = 0x2675e9a37a90000, negotiated timeout = 5000

18/11/30 11:37:47 INFO ha.ActiveStandbyElector: Successfully created /hadoop-ha/HDFScluster in ZK.

18/11/30 11:37:47 INFO ha.ActiveStandbyElector: Session connected.

18/11/30 11:37:47 INFO zookeeper.ZooKeeper: Session: 0x2675e9a37a90000 closed

18/11/30 11:37:47 INFO zookeeper.ClientCnxn: EventThread shut down

[root@realtime-1 ~]#

##### 1.9  Hadoop集群启动运行

我们在***节点1***进行下列操作，在节点1中通过ssh远程登录到节点2和节点3中，实现命令的分发与运行。

步骤1.   **启动HDFS相关服务**

1.   通过下面的命令可以启动HDFS相关服务：

start-dfs.sh

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# start-dfs.sh

18/11/30 11:55:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting namenodes on [realtime-2 realtime-1]

realtime-2: starting namenode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-namenode-realtime-2.out

realtime-1: starting namenode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-namenode-realtime-1.out

realtime-1: starting datanode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-datanode-realtime-1.out

realtime-2: starting datanode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-datanode-realtime-2.out

realtime-3: starting datanode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-datanode-realtime-3.out

Starting journal nodes [realtime-1 realtime-2 realtime-3]

realtime-1: starting journalnode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-journalnode-realtime-1.out

realtime-2: starting journalnode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-journalnode-realtime-2.out

realtime-3: starting journalnode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-journalnode-realtime-3.out

18/11/30 11:56:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting ZK Failover Controllers on NN hosts [realtime-2 realtime-1]

realtime-2: starting zkfc, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-zkfc-realtime-2.out

realtime-1: starting zkfc, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-zkfc-realtime-1.out

[root@realtime-1 ~]#

2.   通过下面的命令查看节点1中对应的相关服务：

jps

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# jps

8048 DataNode

7937 NameNode

8435 DFSZKFailoverController

8245 JournalNode

2597 QuorumPeerMain

8502 Jps

1743 VmServer.jar

[root@realtime-1 ~]#

由返回结果可以看出，在节点1中已经成功启动了NameNode、DataNode 以及JournalNode进程，同时由于节点1作为NameNode节点，因此也启动了ZKFC进程DFSZKFailoverController。

3.   通过下面的命令查看节点2中对应的相关服务：

ssh realtime-2 "jps"

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# ssh realtime-2 "jps"

5447 DFSZKFailoverController

1703 VmServer.jar

5495 Jps

5356 JournalNode

2686 QuorumPeerMain

5262 DataNode

[root@realtime-1 ~]#

由返回结果可以看出，在节点2中已经成功启动了DataNode以及JournalNode进程，同时由于节点2作为NameNode节点，因此也启动了ZKFC进程DFSZKFailoverController，然而此时节点2中并没有启动NameNode进程，所以需要通过后文独立进行启动操作。

4.   查看节点3中对应的相关服务，***节点3***中同***节点2***操作。

由于节点3只是作为存储节点，因此并没有启动ZKFC进程DFSZKFailoverController以及NameNode进程。

5.   通过下列命令格式化Standby节点（节点2）：

ssh realtime-2 "hadoop namenode -bootstrapStandby"

命令运行后的返回结果如下所示：

18/11/30 13:48:31 INFO namenode.NameNode: registered UNIX signal handlers for [TERM, HUP, INT]

18/11/30 13:48:31 INFO namenode.NameNode: createNameNode [-bootstrapStandby]

18/11/30 13:48:31 WARN common.Util: Path /hdfs/namenode should be specified as a URI in configuration files. Please update hdfs configuration.

18/11/30 13:48:31 WARN common.Util: Path /hdfs/namenode should be specified as a URI in configuration files. Please update hdfs configuration.

18/11/30 13:48:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

=====================================================

About to bootstrap Standby ID realtime-2 from:

           Nameservice ID: HDFScluster

        Other Namenode ID: realtime-1

  Other NN's HTTP address: http://realtime-1:50070

  Other NN's IPC  address: realtime-1/10.1.1.4:8020

             Namespace ID: 1029343182

            Block pool ID: BP-348760827-10.1.1.4-1543547236332

               Cluster ID: CID-2b5734df-2bd3-451b-a186-9029edd7e31d

           Layout version: -63

       isUpgradeFinalized: true

=====================================================

18/11/30 13:48:32 INFO common.Storage: Storage directory /hdfs/namenode has been successfully formatted.

18/11/30 13:48:32 WARN common.Util: Path /hdfs/namenode should be specified as a URI in configuration files. Please update hdfs configuration.

18/11/30 13:48:32 WARN common.Util: Path /hdfs/namenode should be specified as a URI in configuration files. Please update hdfs configuration.

18/11/30 13:48:33 INFO namenode.TransferFsImage: Opening connection to http://realtime-1:50070/imagetransfer?getimage=1&txid=0&storageInfo=-63:1029343182:0:CID-2b5734df-2bd3-451b-a186-9029edd7e31d

18/11/30 13:48:33 INFO namenode.TransferFsImage: Image Transfer timeout configured to 60000 milliseconds

18/11/30 13:48:34 INFO namenode.TransferFsImage: Transfer took 0.01s at 0.00 KB/s

18/11/30 13:48:34 INFO namenode.TransferFsImage: Downloaded file fsimage.ckpt\_0000000000000000000 size 351 bytes.

18/11/30 13:48:34 INFO util.ExitUtil: Exiting with status 0

18/11/30 13:48:34 INFO namenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHUTDOWN\_MSG: Shutting down NameNode at realtime-2/10.1.1.3

6.   格式化Standby节点后，通过下面的命令启动Standby节点的NameNode进程：

ssh realtime-2 "hadoop-daemon.sh start namenode"

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# ssh realtime-2 "hadoop-daemon.sh start namenode"

starting namenode, logging to /usr/cx/hadoop-2.7.1/logs/hadoop-root-namenode-realtime-2.out

[root@realtime-1 ~]#

7.   Standby节点的NameNode进程启动完成后，通过下面的命令查看节点2中对应的相关服务：

ssh realtime-2 "jps"

命令运行后的返回结果如下所示（可见此时节点2也成功启动了NameNode进程）：

[root@realtime-1 ~]# ssh realtime-2 "jps"

5792 DataNode

6164 NameNode

1703 VmServer.jar

6251 Jps

5981 DFSZKFailoverController

2686 QuorumPeerMain

5887 JournalNode

[root@realtime-1 ~]#

步骤2.   **启动YARN相关服务**

1.   通过下面的命令启动YARN相关服务：

start-yarn.sh

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# start-yarn.sh

starting yarn daemons

starting resourcemanager, logging to /usr/cx/hadoop-2.7.1/logs/yarn-root-resourcemanager-realtime-1.out

realtime-2: starting nodemanager, logging to /usr/cx/hadoop-2.7.1/logs/yarn-root-nodemanager-realtime-2.out

realtime-3: starting nodemanager, logging to /usr/cx/hadoop-2.7.1/logs/yarn-root-nodemanager-realtime-3.out

realtime-1: starting nodemanager, logging to /usr/cx/hadoop-2.7.1/logs/yarn-root-nodemanager-realtime-1.out

[root@realtime-1 ~]#

2.   通过下面的命令查看节点1中对应的相关服务：

jps

3.   通过下面的命令查看节点2中对应的相关服务：

ssh realtime-2 "jps"

4.   通过下面的命令查看节点3中对应的相关服务：

ssh realtime-3 "jps"

5.   通过下面的命令在节点2中启动ResourceManager进程：

ssh realtime-2 "yarn-daemon.sh start resourcemanager"

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# ssh realtime-2 "yarn-daemon.sh start resourcemanager"

starting resourcemanager, logging to /usr/cx/hadoop-2.7.1/logs/yarn-root-resourcemanager-realtime-2.out

[root@realtime-1 ~]#

6.   通过下面的命令查看节点2中对应的相关服务：

ssh realtime-2 "jps"

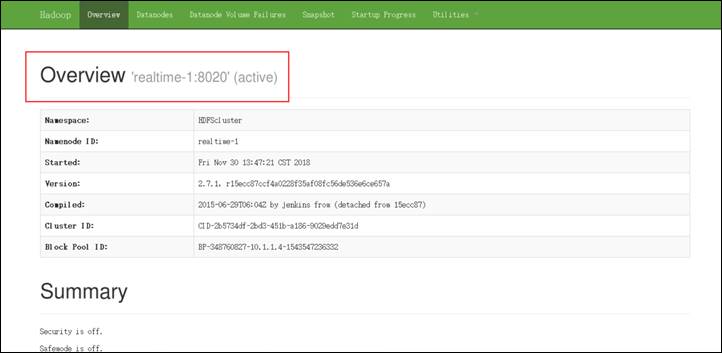
##### 1.10  Hadoop 高可用性测试

笔者在写作过程中是在***节点1***中进行下面的操作，同学们可以在任意节点中进行下面的操作，所实现的效果是一致的。

步骤1.   **NodeManager状态查看**

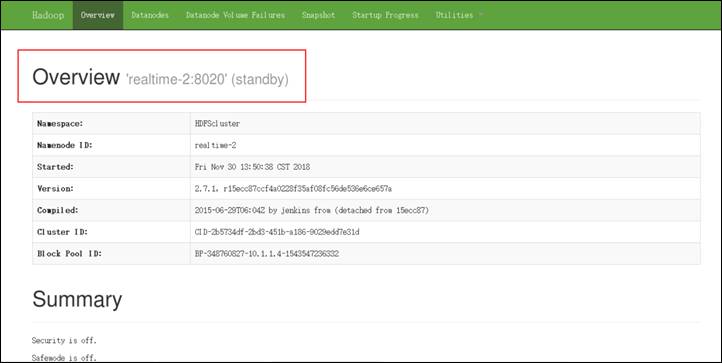
由于设置了2个NameNode，因此必然会有一个处于Active状态，一个处于StandBy状态，至于具体哪个节点处于Active状态，需要根据实际情况确定，并不是千篇一律的。

1.   当Hadoop成功启动后，我们打开浏览器，输入网址***http://realtime-1:50070***便可以访问HDFS的Web管理页面（此时可以看到realtime-1节点是处于active状态的）：



**图 1.5 通过realtime-1访问HDFS Web管理页面**

2.   输入网址***[http://realtime-2:50070](http://realtime-2:50070/)***依然可以访问HDFS的Web管理页面（此时可以看到realtime-2节点是处于standby状态的）：



**图 1.5 通过realtime-2访问HDFS Web管理页面**

步骤2.   **ResourceManager状态查看**

由于设置了2个ResourceManager，因此必然会有一个处于Active状态，一个处于StandBy状态，至于具体哪个节点处于Active状态，需要根据实际情况确定，并不是千篇一律的。

1.   在终端模拟器中，通过下面的命令可以查看逻辑ID为rm1（实际映射的节点为realtime-1）的节点对应的ResourceManager状态：

yarn rmadmin -getServiceState rm1

命令运行后的返回结果如下所示（可见当前节点是active状态）：

[root@realtime-1 ~]# yarn rmadmin -getServiceState rm1

18/11/30 16:35:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

active

[root@realtime-1 ~]#

2.   在终端模拟器中，通过下面的命令可以查看逻辑ID为rm2（实际映射的节点为realtime-2）的节点对应的ResourceManager状态：

yarn rmadmin -getServiceState rm2

命令运行后的返回结果如下所示（可见当前节点是standby状态）：

[root@realtime-1 ~]# yarn rmadmin -getServiceState rm2

18/11/30 16:35:44 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

standby

[root@realtime-1 ~]#

步骤3.   **HDFS高可用测试**

1.   通过下面的命令在HDFS中创建测试文件夹/test：

hadoop fs -mkdir /test

2.   通过下面的命令查看HDFS中创建的测试文件夹/test：

hadoop fs -ls /

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# hadoop fs -ls /

18/11/30 16:40:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 1 items

drwxr-xr-x   - root supergroup          0 2018-11-30 16:39 /test

[root@realtime-1 ~]#

3.   通过下面的命令停止Active状态节点对应的NameNode进程：

ssh realtime-1 "hadoop-daemon.sh stop namenode"

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# ssh realtime-1 "hadoop-daemon.sh stop namenode"

stopping namenode

[root@realtime-1 ~]#

4.   通过下面的命令查看对应节点的进程信息：

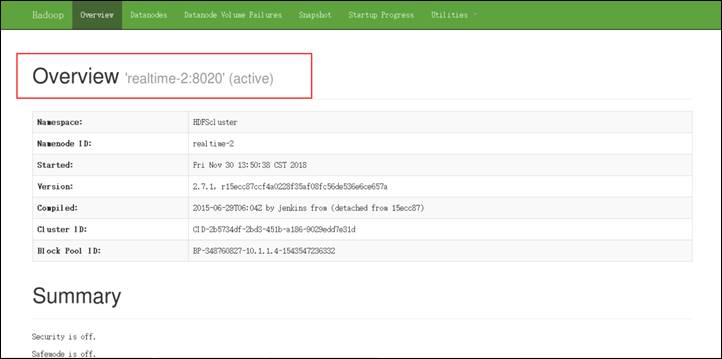
ssh realtime-1 "jps"

通过命令运行后的返回结果判断NameNode进程是否已经被停止。

5.   通过下面的命令查看HDFS中创建的测试文件夹/test：

hadoop fs -ls /

6.   打开浏览器，输入网址***http://realtime-2:50070***访问HDFS的Web管理页面，此时可以看到realtime-2节点已经成功接替成为NameNode并处于active状态：



**图 1.5 查看节点状态**

步骤4.   **YARN高可用测试**

1.   通过下面的命令，使用Hadoop自带的案例测试MapReduce应用程序的运行：

hadoop jar /usr/cx/hadoop-2.7.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar wordcount /test /output

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# hadoop jar /usr/cx/hadoop-2.7.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar wordcount /test /output

18/11/30 16:47:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

18/11/30 16:47:15 INFO input.FileInputFormat: Total input paths to process : 0

18/11/30 16:47:15 INFO mapreduce.JobSubmitter: number of splits:0

18/11/30 16:47:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1543557487449\_0001

18/11/30 16:47:16 INFO impl.YarnClientImpl: Submitted application application\_1543557487449\_0001

18/11/30 16:47:16 INFO mapreduce.Job: The url to track the job: http://realtime-1:8089/proxy/application\_1543557487449\_0001/

18/11/30 16:47:16 INFO mapreduce.Job: Running job: job\_1543557487449\_0001

18/11/30 16:47:26 INFO mapreduce.Job: Job job\_1543557487449\_0001 running in uber mode : false

18/11/30 16:47:26 INFO mapreduce.Job:  map 0% reduce 0%

18/11/30 16:47:37 INFO mapreduce.Job:  map 0% reduce 100%

18/11/30 16:47:38 INFO mapreduce.Job: Job job\_1543557487449\_0001 completed successfully

18/11/30 16:47:39 INFO mapreduce.Job: Counters: 38

   File System Counters

        FILE: Number of bytes read=0

        FILE: Number of bytes written=119357

        FILE: Number of read operations=0

        FILE: Number of large read operations=0

        FILE: Number of write operations=0

        HDFS: Number of bytes read=0

        HDFS: Number of bytes written=0

        HDFS: Number of read operations=3

        HDFS: Number of large read operations=0

        HDFS: Number of write operations=2

   Job Counters

        Launched reduce tasks=1

        Total time spent by all maps in occupied slots (ms)=0

        Total time spent by all reduces in occupied slots (ms)=227232

        Total time spent by all reduce tasks (ms)=7101

        Total vcore-seconds taken by all reduce tasks=7101

        Total megabyte-seconds taken by all reduce tasks=7271424

   Map-Reduce Framework

        Combine input records=0

        Combine output records=0

        Reduce input groups=0

        Reduce shuffle bytes=0

        Reduce input records=0

        Reduce output records=0

        Spilled Records=0

        Shuffled Maps =0

        Failed Shuffles=0

        Merged Map outputs=0

        GC time elapsed (ms)=67

        CPU time spent (ms)=290

        Physical memory (bytes) snapshot=94629888

        Virtual memory (bytes) snapshot=2064699392

        Total committed heap usage (bytes)=30474240

   Shuffle Errors

        BAD\_ID=0

        CONNECTION=0

        IO\_ERROR=0

        WRONG\_LENGTH=0

        WRONG\_MAP=0

        WRONG\_REDUCE=0

   File Output Format Counters

        Bytes Written=0

[root@realtime-1 ~]#

2.   通过下面的命令停止Active状态节点对应的ResourceManager进程

ssh realtime-1 "yarn-daemon.sh stop resourcemanager"

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# ssh realtime-1 "yarn-daemon.sh stop resourcemanager"

stopping resourcemanager

[root@realtime-1 ~]#

3.   通过下面的命令查看对应节点的进程信息：

ssh realtime-1 "jps"

通过命令运行后的返回结果判断ResourceManager进程已经被停止。

4.   通过下面的命令，再次使用Hadoop自带的案例测试MapReduce应用程序的运行：

hadoop jar /usr/cx/hadoop-2.7.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar wordcount /test /output1

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# hadoop jar /usr/cx/hadoop-2.7.1/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar wordcount /test /output1

18/11/30 16:50:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

18/11/30 16:50:31 INFO client.ConfiguredRMFailoverProxyProvider: Failing over to rm2

18/11/30 16:50:32 INFO input.FileInputFormat: Total input paths to process : 0

18/11/30 16:50:32 INFO mapreduce.JobSubmitter: number of splits:0

18/11/30 16:50:32 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1543567750404\_0001

18/11/30 16:50:33 INFO impl.YarnClientImpl: Submitted application application\_1543567750404\_0001

18/11/30 16:50:33 INFO mapreduce.Job: The url to track the job: http://realtime-2:8089/proxy/application\_1543567750404\_0001/

18/11/30 16:50:33 INFO mapreduce.Job: Running job: job\_1543567750404\_0001

18/11/30 16:50:45 INFO mapreduce.Job: Job job\_1543567750404\_0001 running in uber mode : false

18/11/30 16:50:45 INFO mapreduce.Job:  map 0% reduce 0%

18/11/30 16:50:53 INFO mapreduce.Job:  map 0% reduce 100%

18/11/30 16:50:54 INFO mapreduce.Job: Job job\_1543567750404\_0001 completed successfully

18/11/30 16:50:54 INFO mapreduce.Job: Counters: 38

   File System Counters

        FILE: Number of bytes read=0

        FILE: Number of bytes written=119358

        FILE: Number of read operations=0

        FILE: Number of large read operations=0

        FILE: Number of write operations=0

        HDFS: Number of bytes read=0

        HDFS: Number of bytes written=0

        HDFS: Number of read operations=3

        HDFS: Number of large read operations=0

        HDFS: Number of write operations=2

   Job Counters

        Launched reduce tasks=1

        Total time spent by all maps in occupied slots (ms)=0

        Total time spent by all reduces in occupied slots (ms)=147936

        Total time spent by all reduce tasks (ms)=4623

        Total vcore-seconds taken by all reduce tasks=4623

        Total megabyte-seconds taken by all reduce tasks=4733952

   Map-Reduce Framework

        Combine input records=0

        Combine output records=0

        Reduce input groups=0

        Reduce shuffle bytes=0

        Reduce input records=0

        Reduce output records=0

        Spilled Records=0

        Shuffled Maps =0

        Failed Shuffles=0

        Merged Map outputs=0

        GC time elapsed (ms)=81

        CPU time spent (ms)=280

        Physical memory (bytes) snapshot=94146560

        Virtual memory (bytes) snapshot=2064695296

        Total committed heap usage (bytes)=30474240

   Shuffle Errors

        BAD\_ID=0

        CONNECTION=0

        IO\_ERROR=0

        WRONG\_LENGTH=0

        WRONG\_MAP=0

        WRONG\_REDUCE=0

   File Output Format Counters

        Bytes Written=0

[root@realtime-1 ~]#

由返回结果可以看出，此时YARN依然可以可靠的实现任务的调度。

5.   在终端模拟器中，通过下面的命令可以查看逻辑ID为rm2（实际映射的节点为realtime-2）的节点对应的ResourceManager状态（同学们需要根据实际情况来确定）：

yarn rmadmin -getServiceState rm2

命令运行后的返回结果如下所示：

[root@realtime-1 ~]# yarn rmadmin -getServiceState rm2

18/11/30 16:51:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

active

[root@realtime-1 ~]#

由返回结果可以看出，当前节点已经自动成功接替变成了active状态。