zookeeper 3

DataNode 3

journalnode 3 #两个NameNode为了数据同步，会通过一组称作JournalNodes的独立进程进行相互通信。

regionservers 3

hbase.master 2

namenode 2

zkfc 2

一、 组件

**JAVA**：jdk7 注：必须是jdk7

**Hadoop**：

**HBase**：

**Zookeeper**：

系统环境：创建hadoop用户，配置免密登陆，格式化数据盘(ext4)并在根下创建data1-N目录并挂载。

1、集群规划介绍

master slave1 slave2

NameNode NameNode

DataNode DataNode DataNode

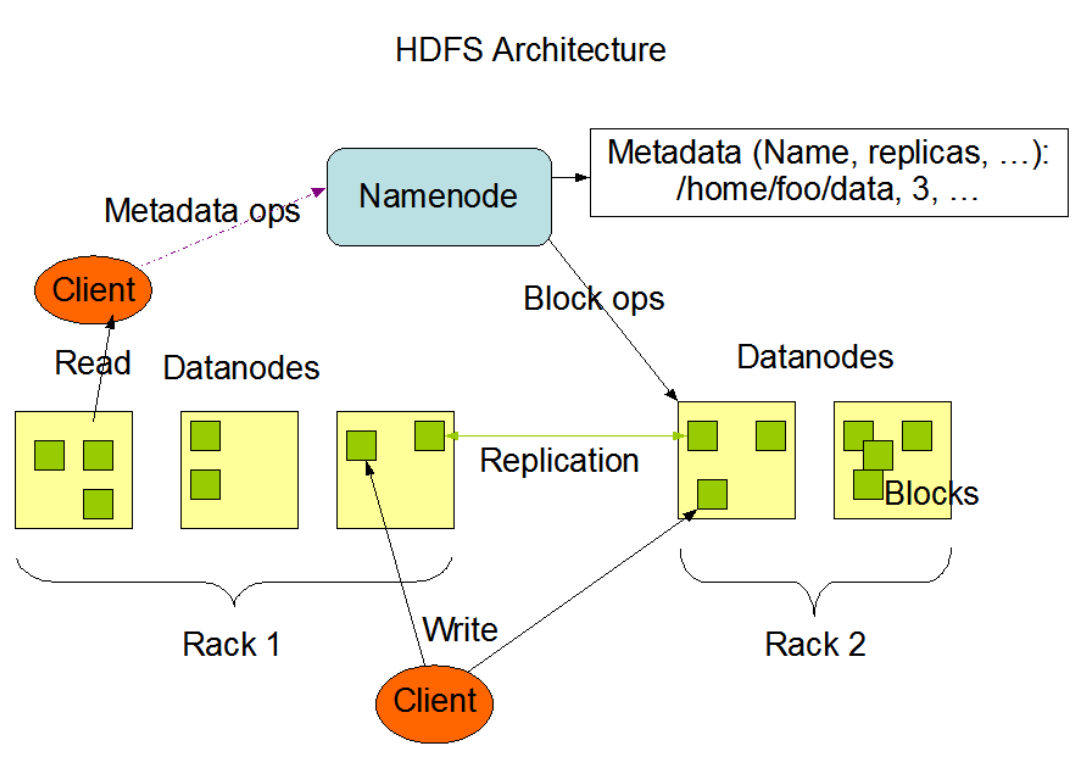
Zookeeper Zookeeper Zookeeper

ResourceManager ResourceManager

NodeManager NodeManager NodeManager

HMaster HMasetr

HRegionServer HRegionServer HRegionServer



HDFS采用master/slave架构。一个HDFS集群是由一个Namenode和一定数目的Datanodes组成

Namenode是一个中心服务器，负责管理文件系统的名字空间(namespace)以及客户端对文件的访问

集群中的Datanode一般是一个节点一个，负责管理它所在节点上的存储。HDFS暴露了文件系统的名字空间，用户能够以文件的形式在上面存储数据

从内部看，一个文件其实被分成一个或多个数据块，这些块存储在一组Datanode上。Namenode执行文件系统的名字空间操作，比如打开、关闭、重命名文件或目录。它也负责确定数据块到具体Datanode节点的映射。

Datanode负责处理文件系统客户端的读写请求。在Namenode的统一调度下进行数据块的创建、删除和复制

Namenode和Datanode被设计成可以在普通的商用机器上运行。这些机器一般运行着GNU/Linux操作系统(OS)。HDFS采用Java语言开发，因此任何支持Java的机器都可以部署Namenode或Datanode。

由于采用了可移植性极强的Java语言，使得HDFS可以部署到多种类型的机器上。一个典型的部署场景是一台机器上只运行一个Namenode实例，而集群中的其它机器分别运行一个Datanode实例。这种架构并不排斥在一台机器上运行多个Datanode，只不过这样的情况比较少见。

集群中单一Namenode的结构大大简化了系统的架构。Namenode是所有HDFS元数据的仲裁者和管理者，这样，用户数据永远不会流过Namenode。

Namenode负责维护文件系统的名字空间，任何对文件系统名字空间或属性的修改都将被Namenode记录下来。应用程序可以设置HDFS保存的文件的副本数目。文件副本的数目称为文件的副本系数，这个信息也是由Namenode保存的。

## 数据复制

HDFS被设计成能够在一个大集群中跨机器可靠地存储超大文件。它将每个文件存储成一系列的数据块，除了最后一个，所有的数据块都是同样大小的。为了容错，文件的所有数据块都会有副本。每个文件的数据块大小和副本系数都是可配置的。应用程序可以指定某个文件的副本数目。副本系数可以在文件创建的时候指定，也可以在之后改变。HDFS中的文件都是一次性写入的，并且严格要求在任何时候只能有一个写入者。

Namenode全权管理数据块的复制，它周期性地从集群中的每个Datanode接收心跳信号和块状态报告(Blockreport)。接收到心跳信号意味着该Datanode节点工作正常。块状态报告包含了一个该Datanode上所有数据块的列表。

ResourceManage 即资源管理，在YARN中，ResourceManager负责集群中所有资源的统一管理和分配，它接收来自各个节点（NodeManager）的资源汇报信息，并把这些信息按照一定的策略分配给各个应用程序（实际上是ApplicationManager）。

NodeManager介绍

NM是ResourceManager在每台机器上的代理，负责容器管理，并监控它们的资源使用情况，以及向ResourceManager/Scheduler提供资源使用报告

HBASE:开源数据库

HMaster的作用：

为Region server分配region(区域)

负责Region server的负载均衡

发现失效的Region server并重新分配其上的region

HDFS上的垃圾文件回收

处理schema更新请求

HRegionServer作用：

维护master分配给他的region，处理对这些region的io请求

负责切分正在运行过程中变的过大的region

可以看到，client访问[HBase](http://www.dataguru.cn/article-9771-1.html?union_site=innerlink" \t "http://www.dataguru.cn/_blank)上的数据并不需要master参与（寻址访问zookeeper和region server，数据读写访问region server），master仅仅维护table和region的元数据信息（table的元数据信息保存在zookeeper上），负载很低。

HRegionServer存取一个子表时，会创建一个HRegion对象，然后对表的每个列族创建一个Store实例，每个Store都会有一个MemStore和0个或多个StoreFile与之对应，每个StoreFile都会对应一个HFile， HFile就是实际的存储文件。因此，一个HRegion有多少个列族就有多少个Store。

一个HRegionServer会有多个HRegion和一个HLog。

QuorumPeerMain(zookeeper)

Zookeeper主要可以干哪些事情：配置管理，名字服务，提供分布式同步以及集群管理。

Zookeeper集群启动的入口类是QuorumPeerMain来加载配置启动QuorumPeer线程。首先我们来看下QuorumPeer， 谷歌翻译quorum是法定人数，定额的意思， peer是对等的意思，那么QuorumPeer中quorum代表的意思就是每个zookeeper集群启动的时候集群中zookeeper服务数量就已经确定了，在每个zookeeper的配置文件中配置集群中的所有机器

server.1=127.0.0.1:2886:3886

server.2=127.0.0.1:2887:3887

server.3=127.0.0.1:2888:3888

事实上着也确定zookeeper在运行中是不能动态扩容的，必须停下服务修改配置才可以；QuorumPeer中peer代表就是集中每个zookeeper角色是对等的没有主从之分，每个zookeeper服务都可以成为leader， follower，observer。

2、 添加Hosts映射关系（分别在三个节点上添加hosts映射关系：）

# vim /etc/hosts

添加的内容如下：

192.168.0.226 master

192.168.0.181 slave1

192.168.0.235 slave2

2、集群之间SSH无密码登陆（CentOS默认安装了ssh，如果没有你需要先安装ssh ）

集群环境的使用必须通过ssh无密码登陆来执行，本机登陆本机必须无密码登陆，主机与从机之间必须可以双向无密码登陆，从机与从机之间无限制。

2.3.1 设置master无密码自动登陆slave1和slave2

# ssh master

# ssh slave1

# ssh slave2

三台集群执行生成秘钥

[hadoop@master ~]$ ssh-keygen -t rsa //执行命令一路回车，生成秘钥

[hadoop@master .ssh]$ cat id\_rsa.pub >>authorized\_keys

[hadoop@master .ssh]$ chmod 600 authorized\_keys

[hadoop@slave1 ~]$

cd /root/.ssh

# scp authorized\_keys master:/root/.ssh/

# scp authorized\_keys slave1:/root/.ssh/

# scp authorized\_keys slave2:/root/.ssh/

重启服务

#systemctl restart sshd.service

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*改正hadoop用户目录的权限为 chmod 700 hadoop\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

三台机器ssh 都要互相拷贝 ，互相执行

2.1、修改主机名和创建账户（所有集群都要操作）

[root@master .ssh]# vim /etc/hostname

master

[root@master src]# useradd hadoop

[root@master src]# passwd hadoop

jianjiechen123

2.3、安装Java

在三台机器上配置JDK环境，下载 jdk-8u111-linux-x64.rpm 文件直接安装：

[root@master src]# rpm -ivh jre-8u77-linux-x64.rpm

[root@master src]# cat /etc/profile

export JAVA\_HOME=/usr/java/jre1.8.0\_77 # 不同的jdk路径需要修改此项

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

export HADOOP\_HOME=/data/yunva/hadoop-2.7.3

export HADOOP\_INSTALL=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

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

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

#export HADOOP\_SSH\_OPTS="-p 48490" # 非默认ssh的22号端口需要添加此项，表示端口为48490

[root@slave1 java]# source /etc/profile

需要单独安装jps java命令 ,还可以使用 jps 命令

java-1.7.0-openjdk-1.7.0.171-2.6.13.0.el7\_4.x86\_64

[hadoop@nm2oos18 ~]$ vim .bashrc

export HADOOP\_HOME=/home/hadoop/hdfs

export HBASE\_HOME=/home/hadoop/hbase

export ZK\_HOME=/home/hadoop/zookeeper

export PATH=/home/appuser/oos/bin:/opt/java/jdk1.7.0\_65/bin:/opt/apache-ant-1.8.4/bin:/home/hadoop/hbase/bin:/home/hadoop/hbase/sbin:/home/hadoop/hdfs/bin:/home/hadoop/hdfs/sbin:/home/hadoop/zookeeper/bin:/home/hadoop/zookeeper/sbin:/opt/java/jdk1.7.0\_65/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1. 安装zookeeper （安装顺序ZK>HDFS>HBase） 元数据服务器都安装zookeeper

**Zookeeper**：配置文件所在目录为$ZK\_HOME/conf，需要配置的配置文件有java.env、 zoo.cfg。

1. java.env

需要配置以下参数，其余参数采用默认值。

JAVA\_HOME、 JVMFLAGS，其中JAVA\_HOME为java的安装目录，JVMFLAGS为jvm的参数，采用默认值即可。

2. zoo.cfg

需要配置以下参数，其余参数采用默认值。

3.1、解压文件到目录

#unzip -n zookeeper.zip -d /data/yunva/zookeeper

3.2、修改配置文件

（dataDir、dataLogDir分别为数据目录、日志目录。）

（clientPort=2181 端口不改变）

（主机名，心跳端口，数据端口 ）

#cd /usr/local/src/zookeeper/conf

[root@master conf]# vim zoo.cfg

dataDir=/home/hadoop/zookeeper/data/zkdata

dataLogDir=/home/hadoop/zookeeper/data/zkdata/datastore\_log

clientPort=2181

maxClientCnxns=200

server.1=master:2888:3888

server.2=slave1:2888:3888

server.3=slave2:2888:3888

2888端口号是zookeeper服务之间通信的端口，而3888是zookeeper与其他应用程序通信的端口。而zookeeper是在hosts中已映射了本机的ip。

3.3、在主目录下建立zookeeper/data目录（与之前配置文件中dataDir一致），并在data目录里面，新建myid文件并且写入“1” ，其他节点依次是2,3,

[root@master data]# touch myid

[root@master data]# echo "1" >myid

/home/hadoop/zookeeper/data/zkdata

[hadoop@master zkdata]$ ll

总用量 12

-rw-r--r-- 1 hadoop hadoop 0 5月 2 15:42 datastore\_log

-rw-r--r-- 1 hadoop hadoop 2 5月 3 09:44 myid

drwxrwxr-x 2 hadoop hadoop 4096 5月 4 09:07 version-2

-rw-rw-r-- 1 hadoop hadoop 4 5月 4 09:06 zookeeper\_server.pid

3.4、配置Java 环境变量

# vim java.env

export JAVA\_HOME=/usr/java/jre1.8.0\_77

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

export JVMFLAGS="-Xms1024m -Xmx2000m $JVMFLAGS"

3.5、配置好，拷贝到其他节点

#chown -R hadoop. zookeeper/

#scp -r zookeeper/ hadoop@slave1:/home/hadoop/

#scp -r zookeeper/ hadoop@slave2:/home/hadoop/

3.6启动服务(三个节点都启动)

在各个zk节点上运行zkServer.sh start，所有zk节点上的zk进程启动之后，可运行zkServer.sh status查看各个zk节点的状态，只有一个是leader，其余的都是follower。

#启动命令

#/home/hadoop/zookeeper/bin$ zkServer.sh start

#查看节点状态

[hadoop@master bin]$ ./zkServer.sh status

JMX enabled by default

Using config: /home/hadoop/zookeeper/bin/../conf/zoo.cfg

Mode: follower（追随者）

#查看节点状态

[hadoop@slave1 bin]$ ./zkServer.sh status

JMX enabled by default

Using config: /home/hadoop/zookeeper/bin/../conf/zoo.cfg

Mode: leader （领导）

#查看节点状态

[hadoop@slave2 bin]$ ./zkServer.sh status

JMX enabled by default

Using config: /home/hadoop/zookeeper/bin/../conf/zoo.cfg

Mode: follower

//查看进程，若有QuorumpeerMain，则启动成功

[root@master bin]# jps

3615 JournalNode

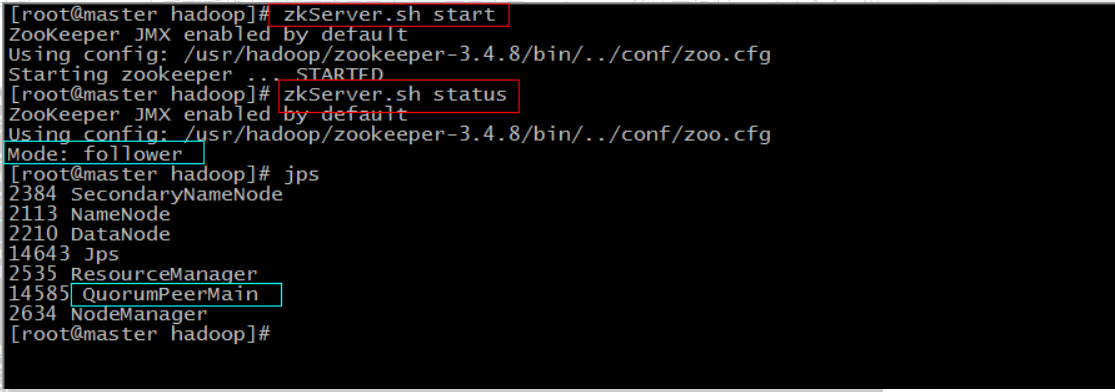
2305 NameNode

9131 Jps

5602 HRegionServer

2409 DataNode

2109 QuorumPeerMain ------zookeeper 进程



###################排查问题查看日志 zookeeper.out#############################

#/home/hadoop/zookeeper/bin

-rwxrwxr-x 1 hadoop hadoop 26674 5月 4 10:16 zookeeper.out

5、安装hadoop（安装 hadoop 集群安装配置）安装和配置HDFS

安装配置hdfs--配置Hadoop-env.sh

(1)、Hadoop2所有的配置都位于$HADOOP\_HOME/etc/hadoop目录下，我们使用的配置文件目录为$HADOOP\_HOME/conf /目录下，手动安装若没有conf目录就需要手动创建，然后将core-site.xml、hadoop-env.sh 、hdfs-site.xml 、masters和slaves配置文件从$HADOOP\_HOME/etc/hadoop中拷贝到$HADOOP\_HOME/conf /目录下,

修改完之后通过scp命令拷贝到元数据所有节点相同的目录下。

**Hadoop**：需要启动以下进程NameNode、DataNode、JournalNode。

依次在各节点上启动qjournal，启动方法为hadoop-daemon.sh start journalnode

在主节点上，运行hdfs zkfc –formatZK

5.1 解压缩

[root@master src]# mkdir -p /home/hadoop/

[root@master src]# unzip -n hadoop.zip -d /home/hadoop/

/home/hadoop/hadoop/conf #conf 目录没有可以新建

[root@master conf]# ll

总用量 24

-rw-r--r-- 1 hadoop hadoop 2682 5月 7 20:05 core-site.xml

-rw-r--r-- 1 hadoop hadoop 3510 5月 7 20:05 hadoop-env.sh

-rw-r--r-- 1 hadoop hadoop 4133 5月 7 20:05 hdfs-site.xml

-rw-r--r-- 1 hadoop hadoop 10 5月 7 20:06 masters

-rw-r--r-- 1 hadoop hadoop 21 5月 7 20:04 slaves

5.2、配置文件

#/home/hadoop/hadoop/etc/hadoop (这里是元数据服务器使用的机器列表，如果在集群里面，就是元数据和存储节点共有使用)

$ vim slaves ##配置所有的DataNode节点，每个节点一行，格式如下所示：

master

slave1

slave2

$ vim hadoop-env.sh #配置Java环境变量

export HADOOP\_HOME=/home/hadoop/hadoop

export HADOOP\_CONF=$HADOOP\_HOME/conf

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

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

export JAVA\_HOME=/usr/java/jre1.8.0\_77

export JAVA\_LIBRARY\_PATH=${HADOOP\_HOME}/lib/native

hdfs-site.xml配置：



#【使用federation时，HDFS集群别名。名字可以随便起，多个集群时相互不重复即可】

<name>dfs.nameservices</name>

<value>oos-hbase</value>

【指定该集群的namenode的机器】

<name>dfs.ha.namenodes.oos-hbase</name>

<value>nn1,nn2</value>

#说明：该参数为NameNode主节点

<name>dfs.namenode.rpc-address.oos-hbase.nn1</name>

<value>master:8020</value>

#说明：该参数为NameNode备节点

<name>dfs.namenode.rpc-address.oos-hbase.nn2</name>

<value>slave1:8020</value>

【指定hadoop100的RPC地址】

#Hadoop的进程间交互都是通过RPC来进行的，比如Namenode与Datanode直接，Jobtracker与Tasktracker之间等。

**#透明性**：远程调用其他机器上的程序，对用户来说就像是调用本地方法一样；

#高性能：RPC Server能够并发处理多个来自Client的请求；

#可控性：jdk中已经提供了一个RPC框架—RMI，但是该PRC框架过于重量级并且可控之处比较少，所以Hadoop RPC实现了自定义的PRC框架。

<name>dfs.namenode.rpc-address.oos-hbase.nn1</name>

<value>master:8020</value>

【指定hadoop101的http地址】

<name>dfs.namenode.http-address.cluster1.hadoop101</name> <value>hadoop101:50070</value> </property>

【指定该集群的两个NameNode共享edits文件目录时，使用的JournalNode集群信息】

该参数为JournalNode部署的节点

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

<value>qjournal://master:8485;slave1:8485;slave2:8485/oos-hbase</value>

【指定该集群是否启动自动故障恢复，即当NameNode出故障时，是否自动切换到另一台NameNode】<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

【指定该集群出故障时，哪个实现类负责执行故障切换】

<name>dfs.client.failover.proxy.provider.oos-hbase</name> <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

【一旦需要NameNode切换，使用ssh方式进行操作】

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

<value>sshfence</value>

【如果使用ssh进行故障切换，使用ssh通信时用的密钥存储的位置】

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

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

【connect-timeout连接超时】

<name>ha.zookeeper.session-timeout.ms</name>

<value>30000</value>

#指定DataNode存储block的副本数量。默认值是3个，我们现在有4个DataNode，该值不大于4即可。

<name>dfs.replication</name>

<value>3</value>

#datanode 最大链接

<name>dfs.datanode.max.xcievers</name>

<value>4096</value>

#两个NameNode为了数据同步，会通过一组称作JournalNodes的独立进程进行相互通信。

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

<value>/mnt/hhzData/journal/data</value>

#datanode 该参数为JournalNode存储数据的目录

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

<value>/mnt/hhzData/hdfs/data</value>

#zookeeper集群的URL配置，多个host中间用逗号该参数为zk所在的节点

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

<value>master:2181,slave1:2181,slave2:2181</value>

#zk ,区域node的地址

<name>rsRegion.zk.node</name>

<value>/oos/metaRegion</value>

$core-site.xml配置：



【这里的值指的是默认的HDFS路径。当有多个HDFS集群同时工作时，集群名称在这里指定！该值来自于hdfs-site.xml中的配置】

<name>fs.defaultFS</name>

<value>hdfs://oos-hbase</value>

【这里是ZooKeeper集群的地址和端口。注意，数量一定是奇数，且不少于三个节点】该参数为zk节点

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

<value>master:2181,slave1:2181,slave2:2181</value>

【这里的路径默认是NameNode、DataNode、JournalNode等存放数据的公共目录。用户也可以自己单独指定这三类节点的目录。该参数为hadoop临时目录

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

<value>/home/hadoop/hadoop/data/hhzData/hdfs/tmp</value>

#hadoop http认证签名目录地址

<name>hadoop.http.authentication.signature.secret.file</name> <value>/home/hadoop/hadoop/etc/hadoop/hadoop-http-auth-signature-secret</value>

#创建相关目录并授权

mkdir -p /mnt/hhzData/journal/data

chown -R hadoop. /mnt/hhzData/journal/data

mkdir -p /oos/metaRegion

chown -R hadoop. /oos/metaRegion

5.3修改权限

chown -R hadoop. hadoop/

5.4、启动hdfs

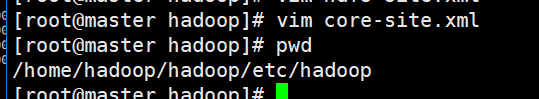
在启动HDFS之前，先启动qjournal，启动方法为nohup sh hdfs journalnode &，与journalnode有关的配置为hdfs-site.xml中标红的部分，在上面配置中需要在fjmeta001、fjmeta002、fjmeta003上启动qjournal

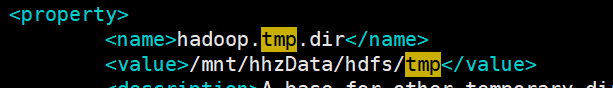
三台节点启动完 ，查看服务就是多了这个

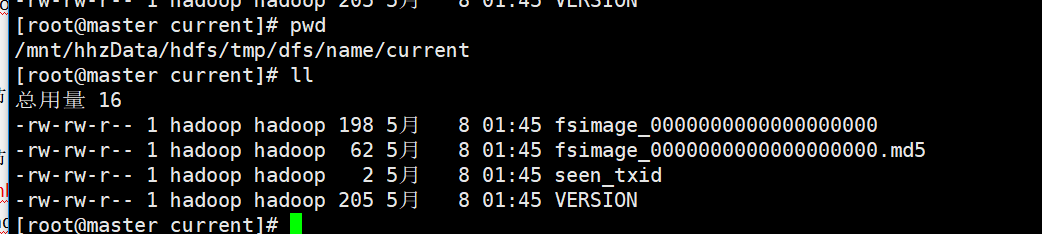
[hadoop@slave2 bin]$ jps |grep -v 'Jps' |grep Journ

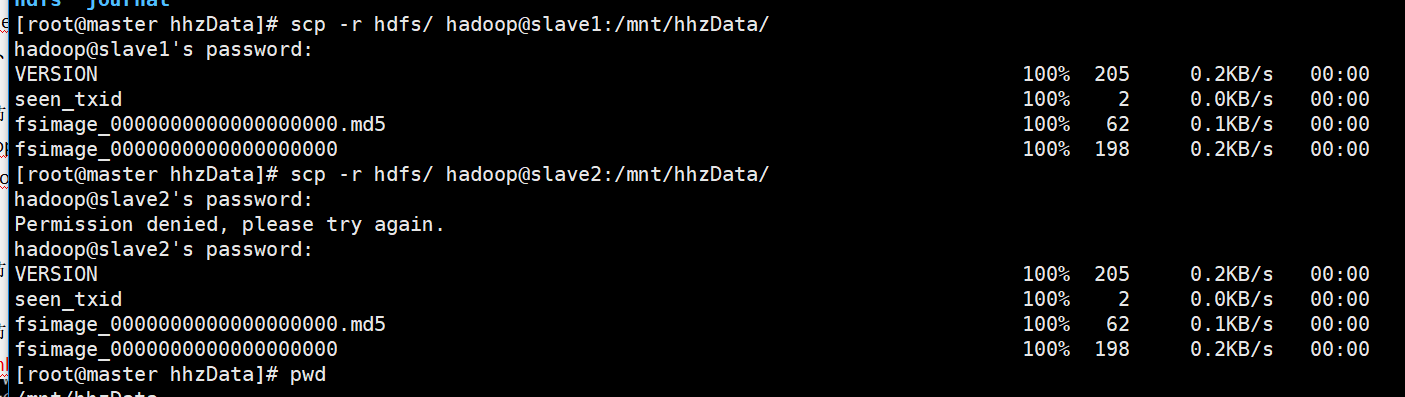
2536 JournalNode

在主节点上，在这里是fjmeta001，运行hdfs zkfc -formatZK （格式化zk）

在主节点上运行hdfs namenode -format进行格式化(格式化nomenode)，然后将fjmeta001的/data1/hbase/hdfs/tmp目录拷贝拷贝到fjmeta002的相同目录下（这个目录是自己设置的目录）







启动hdfs，start-dfs.sh

启动之后可通过：hdfs dfsadmin –report查看集群的状态

停止：stop-dfs.sh

格式花命令：两个要看一下，了解一下 具体哪个使用

hdfs namenode -format

[hadoop@master bin]$ hdfs zkfc -formatZK

18/05/03 14:42:58 INFO zookeeper.ZooKeeper: Client environment:user.dir=/home/hadoop/hadoop/bin

18/05/03 14:42:58 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=master:2181,slave1:2181,slave2:2181 sessionTimeout=30000 watcher=org.apache.hadoop.ha.ActiveStandbyElector$WatcherWithClientRef@12028586

18/05/03 14:42:58 INFO zookeeper.ClientCnxn: Opening socket connection to server slave2/192.168.0.235:2181. Will not attempt to authenticate using SASL (unknown error)

18/05/03 14:42:58 INFO zookeeper.ClientCnxn: Socket connection established to slave2/192.168.0.235:2181, initiating session

18/05/03 14:42:58 INFO zookeeper.ClientCnxn: Session establishment complete on server slave2/192.168.0.235:2181, sessionid = 0x36323bfd53e0001, negotiated timeout = 30000

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

The configured parent znode /hadoop-ha/oos-hbase already exists.

Are you sure you want to clear all failover information from

ZooKeeper?

WARNING: Before proceeding, ensure that all HDFS services and

failover controllers are stopped!

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

Proceed formatting /hadoop-ha/oos-hbase? (Y or N) 18/05/03 14:42:58 INFO ha.ActiveStandbyElector: Session connected.

y

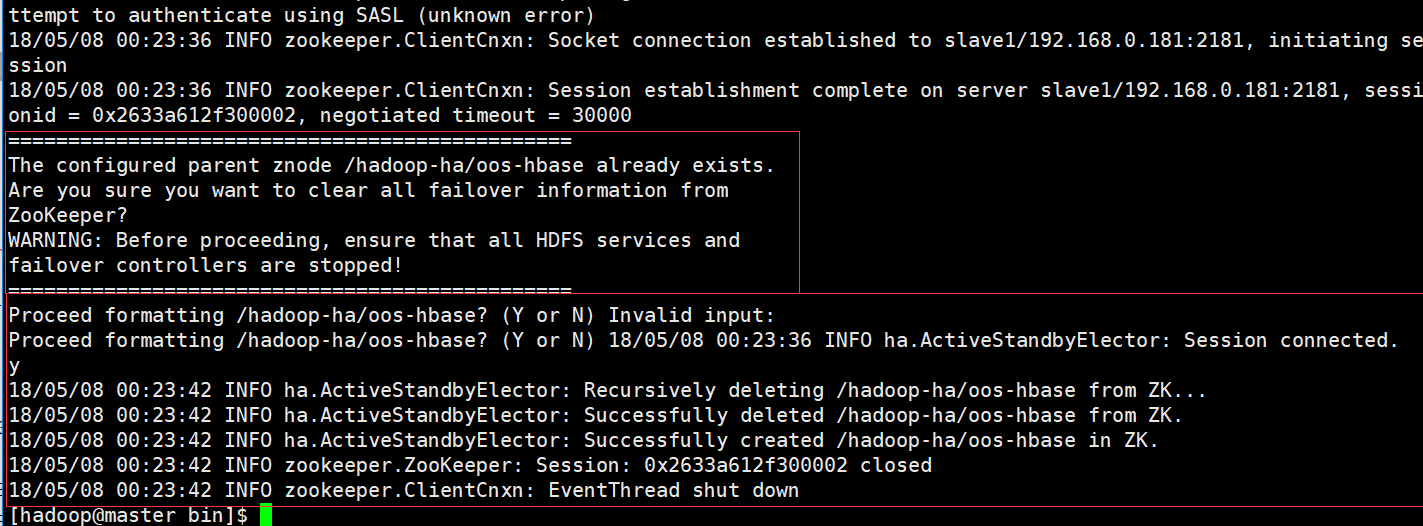
18/05/03 14:43:04 INFO ha.ActiveStandbyElector: Recursively deleting /hadoop-ha/oos-hbase from ZK...

18/05/03 14:43:04 INFO ha.ActiveStandbyElector: Successfully deleted /hadoop-ha/oos-hbase from ZK.

18/05/03 14:43:04 INFO ha.ActiveStandbyElector: Successfully created /hadoop-ha/oos-hbase in ZK.

18/05/03 14:43:04 INFO zookeeper.ZooKeeper: Session: 0x36323bfd53e0001 closed

18/05/03 14:43:04 INFO zookeeper.ClientCnxn: EventThread shut down



[hadoop@master bin]$ hdfs namenode -format

18/05/03 15:03:52 INFO util.GSet: VM type = 64-bit

18/05/03 15:03:52 INFO util.GSet: 0.029999999329447746% max memory = 6.9 GB

18/05/03 15:03:52 INFO util.GSet: capacity = 2^18 = 262144 entries

Re-format filesystem in Storage Directory /home/hadoop/hadoop/data/hhzData/hdfs/tmp/dfs/name ? (Y or N) y

Re-format filesystem in QJM to [192.168.0.226:8485, 192.168.0.181:8485, 192.168.0.235:8485] ? (Y or N) y

18/05/03 15:04:00 INFO common.Storage: Storage directory /home/hadoop/hadoop/data/hhzData/hdfs/tmp/dfs/name has been successfully formatted.

18/05/03 15:04:00 INFO namenode.FSImage: Saving image file /home/hadoop/hadoop/data/hhzData/hdfs/tmp/dfs/name/current/fsimage.ckpt\_0000000000000000000 using no compression

18/05/03 15:04:00 INFO namenode.FSImage: Image file /home/hadoop/hadoop/data/hhzData/hdfs/tmp/dfs/name/current/fsimage.ckpt\_0000000000000000000 of size 198 bytes saved in 0 seconds.

18/05/03 15:04:00 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0

18/05/03 15:04:00 INFO util.ExitUtil: Exiting with status 0

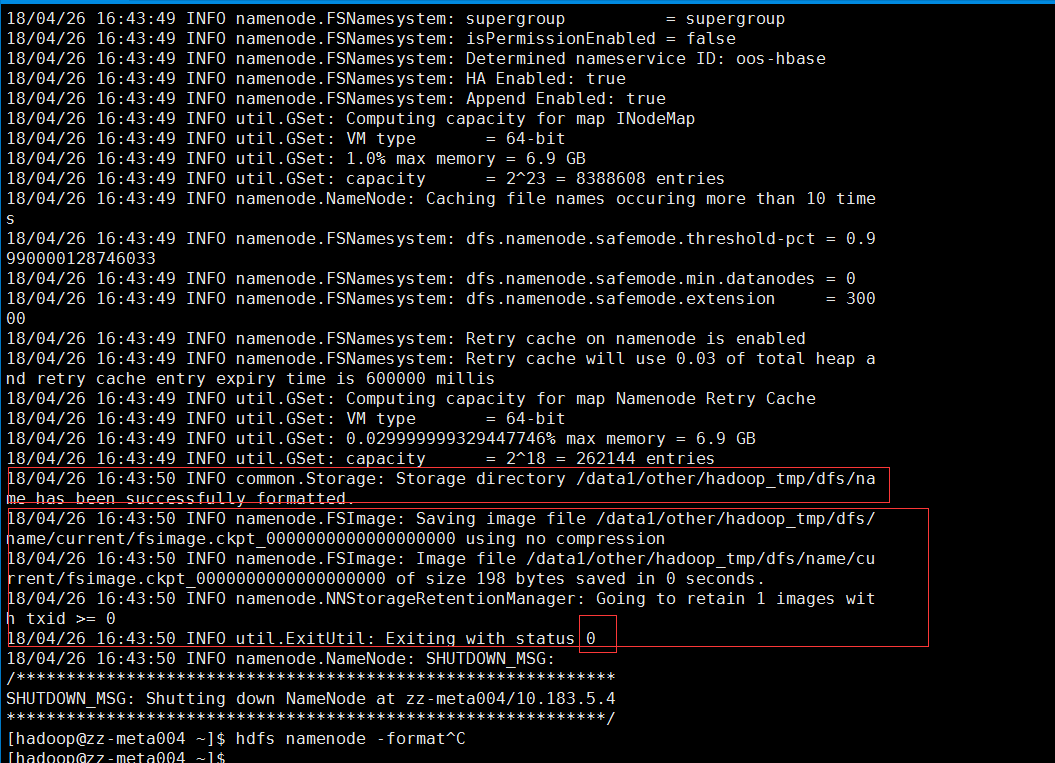
18/05/03 15:04:00 INFO namenode.NameNode: SHUTDOWN\_MSG:

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

SHUTDOWN\_MSG: Shutting down NameNode at master/192.168.0.226

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

[hadoop@master bin]$



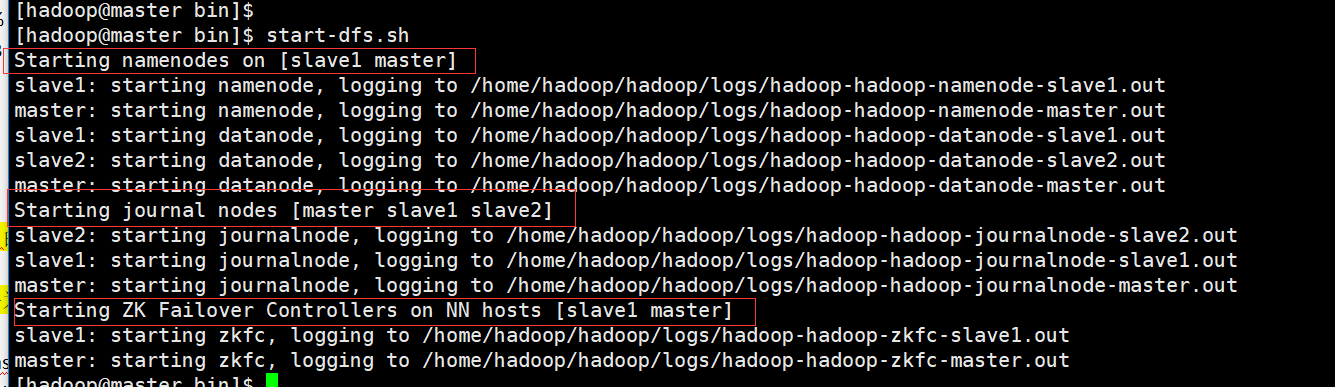
内容拷贝到从节点

[hadoop@master data]$ scp -r hhzData/ hadoop@slave1:/home/hadoop/hadoop/data/

[hadoop@master data]$ scp -r hhzData/ hadoop@slave2:/home/hadoop/hadoop/data/

启动hadoop hdfs #在节点1 启动

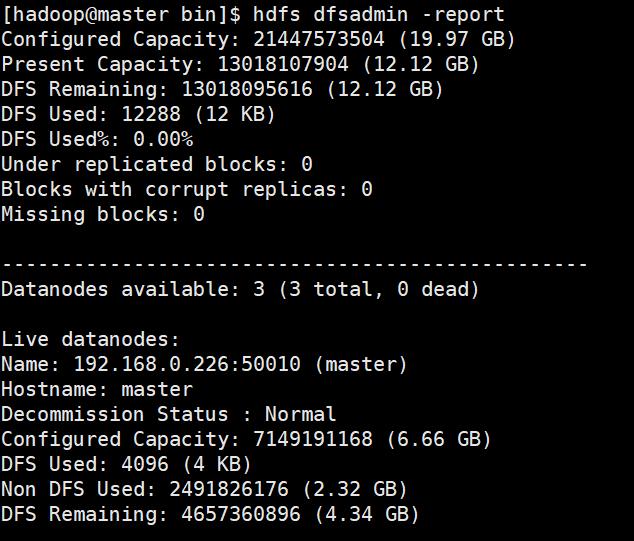
[hadoop@master hadoop]$ sbin/start-dfs.sh

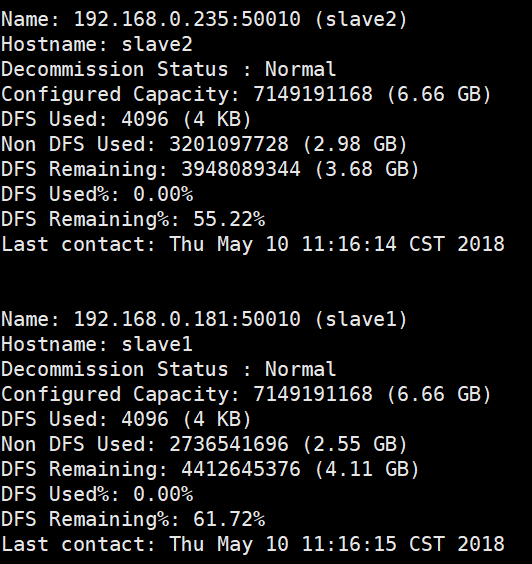


[hadoop@master bin]$

查看集群状态 （这个是看slaves 里面配置了几个节点，就显示几个，这里配置了3个节点，显示了3个）

hadoop@master bin]$ hdfs dfsadmin -report





查看服务

[hadoop@master bin]$ jps |grep -v 'Jps'

2886 JournalNode

21546 DataNode -启动服务

21421 NameNode -启动服务

22829 DFSZKFailoverController -启动服务

2550 QuorumPeerMain

节点2

[hadoop@slave1 logs]$ jps |grep -v 'Jps'

12890 DFSZKFailoverController -启动服务

11756 DataNode -启动服务

11661 NameNode -启动服务

2585 JournalNode

2393 QuorumPeerMain

节点3

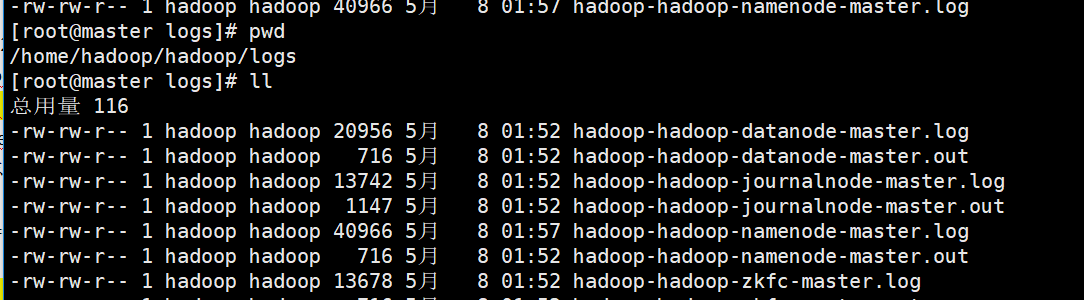
[hadoop@slave2 ~]$ jps |grep -v 'Jps'

2536 JournalNode

9798 DataNode -启动服务

2349 QuorumPeerMain

排查 查看日志信息



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

5、安装hbase ，Hbase的配置文件/home/Hadoop/hbase/conf/hbase-site.xml，配置中标绿的为新增的配置项，

对应的新增元数据文件为metaRegion，修改配置文件之前，需要在所有元数据节点上手动创建如下目录：

$mkdir –pv /data2/hbase/hbse/data

$mkdir –pv /var/hbase/tmp

$mkdir –pv /var/hbase/local

#解压缩到文件

[root@master src]# unzip -n hbase.zip -d /home/hadoop/hbase

配置环境变量

# cat /etc/profile

export HBASE\_HOME=/home/hadoop/hbase

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

配置文件信息如下：

/home/hadoop/hbase/conf

vim hbase-env.sh

export HBASE\_HOME=/home/hadoop/hbase

export HADOOP\_HOME=/home/hadoop/hadoop

export JAVA\_HOME=/usr/java/jre1.8.0\_77

export JAVA\_LIBRARY\_PATH=${HBASE\_HOME}/lib/native:${HADOOP\_HOME}/lib/native

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

export HADOOP\_OPTS="-Djava.library.path=${HADOOP\_HOME}/lib/native"

#日志文件路径

export HBASE\_LOG\_DIR=/home/hadoop/hbase/log

#备份masters 节点（当前hbase集群有活动的master节点，该master节点为hadoop01，所以hadoop02节点开始等待，直到hadoop01上的master挂掉。hadoop02会变成新的hmaster节点。）

vim backup-masters

slave1

需要配置regionservers（区域服务器）信息如下：RegionServer是HBase集群运行在每个工作节点上的服务。它是整个HBase系统的关键所在

$ vim regionservers

master

slave1

slave2

vim hdfs-site.xml 拷贝文件过来

/home/hadoop/hbase/conf

[root@master conf]# cp ../../hadoop/etc/hadoop/hdfs-site.xml .

(4)、启动hbase之前先要启动FailoverTracker,切记！启动方式如下：

$nohup hbase failoverTracker &

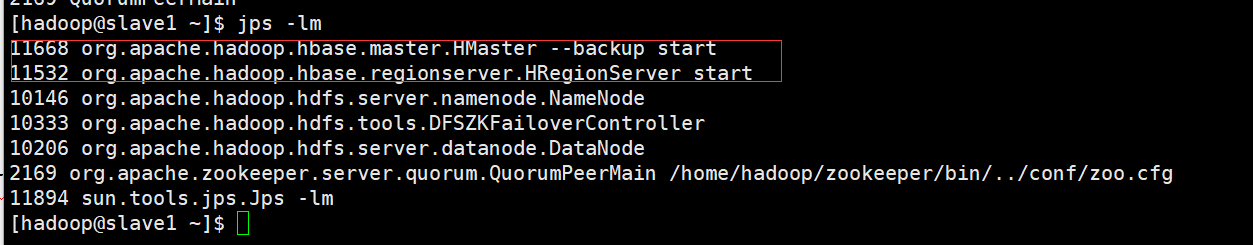
[hadoop@master bin]$ jobs

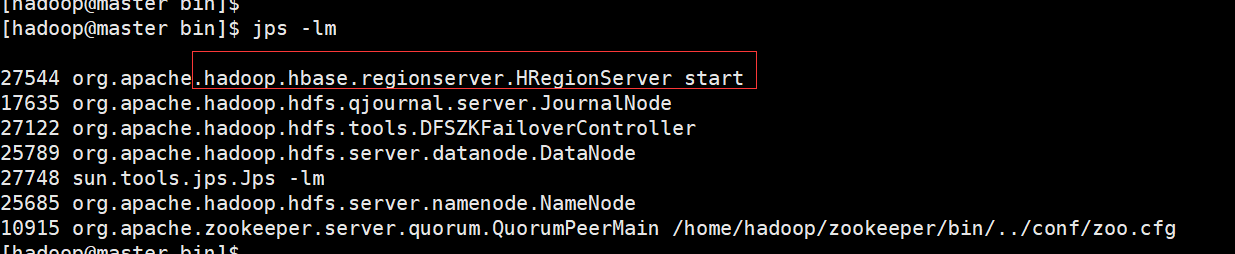
[2]- 运行中 nohup sh hdfs journalnode &(工作目录：~/hadoop/bin)

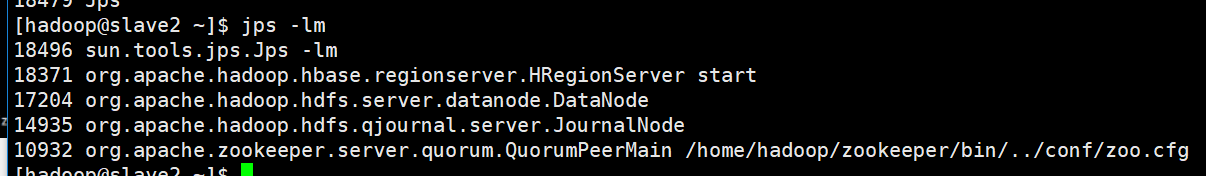
[3]+ 运行中 nohup hbase failoverTracker &

(5)、将hadoop的hdfs-site.xml拷贝到$HBASE\_HOME/conf下

启动：start-hbase.sh







# 停止：stop-hbase.sh

启动服务

[hadoop@master bin]$ start-hbase.sh

starting master, logging to /home/hadoop/hbase/log/hbase-hadoop-master-master.out

slave2: regionserver running as process 3692. Stop it first.

hadoop@master's password: slave1: regionserver running as process 4055. Stop it first.

master: regionserver running as process 5602. Stop it first.

slave2: starting master, logging to /home/hadoop/hbase/log/hbase-hadoop-master-slave2.out

slave1: starting master, logging to /home/hadoop/hbase/log/hbase-hadoop-master-slave1.out

hadoop@master's password:

master: master running as process 5898. Stop it first.

查看服务

[hadoop@slave2 conf]$ jps

\3930 Jps

3692 HRegionServer -----------

2101 QuorumPeerMain

3847 HMaster -----------

3328 JournalNode

2213 DataNode

报错：ssh 秘钥没有设置好

hadoop@master's password: master: Permission denied, please try again.

systemctl restart sshd.service

#关闭服务顺序

hbase-daemon.sh stop master

一个节点关闭

|  |
| --- |
| stop-hbase.sh  三个节点都要关闭  hadoop-daemons.sh stop datanode |

hadoop-daemons.sh stop datanode

hadoop-daemons.sh stop datanode

hadoop-daemon.sh stop journalnode

三个节点

zkServer.sh stop