# 大数据

**Hadoop 大数据分析**

虚拟机4台 2CPU 2G内存， 硬盘16G

192.168.1.10 nn01

192.168.1.11 node1

192.168.1.12 node2

192.168.1.13 node3

[root@nn01 ~]# yum install -y java-1.8.0-openjdk-devel #所有机器都做

#安装这个包其他依赖包 都会装上

[root@nn01 ~]# java -version

[root@nn01 ~]# jps #查看进程

在真机中提取这个包

[root@nn01 ~]# lftp 192.168.1.254

lftp 192.168.1.254:~> cd public

lftp 192.168.1.254:/public> get hadoop-2.7.6.tar.gz #下载到本地

[root@nn01 ~]# tar zxf hadoop-2.7.6.tar.gz

[root@nn01 ~]# mv hadoop-2.7.6 /usr/local/hadoop

[root@nn01 ~]# rpm -ql java-1.8.0-openjdk #查询java安装路径

[root@nn01 ~]# cd /usr/local/hadoop

[root@nn01 hadoop]# cd etc/

[root@nn01 etc]# cd hadoop/

[root@nn01 hadoop]# vim hadoop-env.sh **#修改配置文件**

25

export JAVA\_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-11.b12.el7.x86\_ 64/jre" #java的安装路径

33 export HADOOP\_CONF\_DIR="/usr/local/hadoop/etc/hadoop" #修改为当前路径

[root@nn01 hadoop]# pwd -P **#查看当前路径**

/usr/local/hadoop/etc/hadoop

[root@nn01 hadoop]# cd /usr/local/hadoop/

[root@nn01 hadoop]# ./bin/hadoop

[root@nn01 hadoop]# ./bin/hadoop version #查看版本

**分析数据**

[root@nn01 hadoop]# mkdir oo

[root@nn01 hadoop]# cp \*.txt oo/ #将txt文件拷贝到oo下

[root@nn01 hadoop]# cd oo

[root@nn01 hadoop]# cd /usr/local/hadoop

[root@nn01 hadoop]# bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount oo xx

[root@nn01 hadoop]# cat xx/\*

**命令解析：**

bin/hadoop #调用bin下的hadoop

jar #参数

share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar

#java脚本路径

wordcount #调用java脚本的方法，单词频率出现的总和的统计

oo #统计oo这个文件夹

xx #统计完后放到xx这个文件夹里

测试

[root@nn01 hadoop]# vim a.sh

[root@nn01 hadoop]# /bin/bash a.sh #只能用/bin/bash来运行脚本

hello world

[root@nn01 hadoop]# rm -rf a.sh

[root@nn01 hadoop]# bin/hadoop

version #查看版本信息

jar #运行jar文件

# file share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar

#jar文件在这个路径下

**hadoop 单机配置**

/usr/local/hadoop/etc/hadoop/hadoop-env.sh

export JAVA\_HOME="/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-11.b12.

el7.x86\_64/jre"

export HADOOP\_CONF\_DIR="/usr/local/hadoop/etc/hadoop"

分析单词出现的次数

./bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.

6.jar wordcount oo xx

#-----------------------------------------------------#

**ALL 表示所有主机， 本文档规则**

**NODE 表示 node1,node2,node3**

**NN1: 表示 namenode**

#-----------------------------------------------------#

完全分布式集群搭建 -- HDFS

192.168.1.10 nn01 namenode,secondarynamenode

192.168.1.11 node1 datanode

192.168.1.12 node2 datanode

192.168.1.13 node3 datanode

ALL: 配置 /etc/hosts

ALL: 安装 java-1.8.0-openjdk-devel

[root@nn01 ~]# vim /etc/ssh/ssh\_config #要求远程不能输入yes

35 # StrictHostKeyChecking ask #复制该行

58 Host \*

60 StrictHostKeyChecking no #插入该行

**完全分布式**

**配置ssh信任关系（namenode）**

[root@nn01 ~]# cd .ssh/ #免密登陆

[root@nn01 .ssh]# ssh-keygen -t rsa -b 2048 -N '' -f id\_rsa

[root@nn01 .ssh]# ssh-copy-id -i id\_rsa.pub nn01

[root@nn01 .ssh]# ssh-copy-id -i id\_rsa.pub node1

[root@nn01 .ssh]# ssh-copy-id -i id\_rsa.pub node2

[root@nn01 .ssh]# ssh-copy-id -i id\_rsa.pub node3

[root@nn01 ~]# cd /usr/local/hadoop/etc/hadoop/

[root@nn01 hadoop]# vim slaves #改为以下主机名就可以

node1

node2

node3

<http://hadoop.apache.org/> #官方手册

**core-site.xml 配置**

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://nn01:9000</value>

</property>

<property>

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

<value>/var/hadoop</value>

</property>

</configuration>

**ALL: 创建 /var/hadoop**

[root@nn01 hadoop]# mkdir /var/hadoop

[root@nn01 hadoop]# ssh node1 mkdir /var/hadoop

[root@nn01 hadoop]# ssh node2 mkdir /var/hadoop

[root@nn01 hadoop]# ssh node3 mkdir /var/hadoop

**hdfs-site.xml 配置**

<configuration>

<property>

<name>dfs.namenode.http-address</name>

<value>nn01:50070</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>nn01:50090</value>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

</configuration>

**ALL: 同步配置到所有主机**

如果报这个错需要两边都要安装rsync 命令（新的服务器需要执行yum install rsync -y）**：**rsync error: remote command not found,rsync执行的时候两端都要有rsync命令。

[root@nn01 hadoop]# for i in node{1..3}; do rsync -aSH --delete /usr/local/hadoop ${i}:/usr/local/ -e 'ssh' & done

[root@nn01 hadoop]# wait

#**wait**命令用来等待指令的指令，直到其执行完毕后返回终端。该指令常用于shell脚本编程中，待指定的指令执行完成后，才会继续执行后面的任务。

[root@nn01 hadoop]# vim rsync #编写一个备份脚本

#!/bin/bash

#$\* 和 $@ 的区别

#$\* 和 $@ 都表示传递给函数或脚本的所有参数，不被双引号(" ")包含时，都以"$1" "$2" … "$n" 的形式输出所有参数。

#但是当它们被双引号(" ")包含时，"$\*" 会将所有的参数作为一个整体，以"$1 $2 … $n"的形式输出所有参数；"$@" 会将各个参数分开，以"$1" "$2" … "$n" 的形式输出所有>参数。

for i in $@;do

rsync -aSH --delete /usr/local/hadoop/etc ${i}:/usr/local/hadoop/ -e 'ssh'

&

done

wait

[root@nn01 hadoop]# chmod 755 rsync

[root@nn01 hadoop]# rsync node{1..3} #运行同步脚本

**NN01: 格式化 namenode**

[root@nn01 hadoop]# cd /usr/local/hadoop/

[root@nn01 hadoop]# ./bin/hdfs namenode -format

#看见successfully formatted.表示成功

**NN01: 启动集群**

[root@nn01 hadoop]# ./sbin/start-dfs.sh

停止集群可以使用 ./sbin/stop-dfs.sh

**ALL: 验证角色 jps**

**NN01: 验证集群是否组建成功**

./bin/hdfs dfsadmin -report

服务启动日志路径 /usr/local/hadoop/logs

[root@nn01 ~]# cd /usr/local/hadoop/

[root@nn01 hadoop]# cd etc/hadoop/

[root@nn01 hadoop]# mv mapred-site.xml.template mapred-site.xml

#修改为.xml文件（有效配置）

**mapred-site.xml 配置**

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

**yarn-site.xml 配置**

<configuration>

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

<property>

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

<value>nn01</value>

</property>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

</configuration>

**ALL: 同步配置到主机**

**NN1: 启动服务 ./sbin/start-yarn.sh**

**ALL: 验证角色 jps**

**NN1: 验证节点状态 ./bin/yarn node -list**

# ./sbin/stop-all.sh 关闭所有服务

# ./sbin/start-all.sh 开启所有服务

**namenode web 地址 http://192.168.1.10:50070**

**sedondary namenode web 地址 http://192.168.1.10:50090**

**yarn resourcemanager 管理地址 http://192.168.1.10:8088**

**datanode web 地址 http://192.168.1.11:50075**

**nodemanager web 地址 http://192.168.1.11:8042**

[root@nn01 hadoop]# ./bin/hadoop fs -mkdir /aaa #创建文件夹

[root@nn01 hadoop]# ./bin/hadoop fs -touchz /f1 #创建f1文件

[root@nn01 hadoop]# ./bin/hadoop fs -put \*.txt /aaa/ #上传数据

[root@nn01 hadoop]# ./bin/hadoop fs -get /f1 ./f2 #下载

[root@nn01 hadoop]# ./bin/hadoop fs -ls / #查看

[root@nn01 hadoop]# ./bin/hadoop fs -rm /f1 #删除文件

**分析数据**

[root@nn01 hadoop]# ./bin/hadoop fs -mkdir /oo

[root@nn01 hadoop]# ./bin/hadoop fs -put \*.txt /oo

[root@nn01 hadoop]# bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount **/**oo **/**xx

[root@nn01 hadoop]# ./bin/hadoop fs -cat /xx/\* #查看

### Hadoop 节点管理

**HDFS增加修复节点**

**1、按照单机方法安装一台机器，部署运行的 java 环境**

**2、设置 ssh 免密码登录**

[root@nn01 hadoop]# cd /root/.ssh/

[root@nn01 .ssh]# ssh-copy-id -i id\_rsa.pub 192.168.1.15

**3、在所有节点增加新新节点的主机信息 /etc/hosts**

[root@nn01 .ssh]# vim /etc/hosts

……

192.168.1.15 newnode

**4、修改namenode的slaves文件增加该节点**

[root@nn01 hadoop]# vim etc/hadoop/slaves

……

newnode

[root@nn01 hadoop]# rsync node{1..3} #同步

[root@nn01 ~]# scp -r /usr/local/hadoop/ root@192.168.1.15:/usr/local/

[root@newnode hadoop]# rm -rf logs #删除logs重新创建（选做）

[root@newnode hadoop]# mkdir /var/hadoop

**5、拷贝namnode的 /usr/local/hadoop 到本机**  #namenode指nn01机器

**6、在该节点启动Datanode**

./sbin/hadoop-daemons.sh start datanode

**7、回到namenode上检查是否集群连接成功**

[root@nn01 hadoop]# ./bin/hdfs dfsadmin -report

**8、设置同步带宽,并同步数据**

./bin/hdfs dfsadmin -setBalancerBandwidth 60000000

./sbin/start-balancer.sh

**9、查看集群状态**

./bin/hdfs dfsadmin -report

**删除节点**

[root@nn01 hadoop]# vim etc//hadoop/hdfs-site.xml

……

<property>

<name>dfs.hosts.exclude</name>

<value>/usr/local/hadoop/etc/hadoop/exclude</value>

</property>

#将排除的主机写到这个文件里 /usr/local/hadoop/etc/hadoop/exclude

增加 exclude 配置文件,写入要删除的节点 主机名

[root@nn01 hadoop]# touch /usr/local/hadoop/etc/hadoop/exclude

[root@nn01 hadoop]# vim /usr/local/hadoop/etc/hadoop/exclude

newnode

**开始导出数据**

./bin/hdfs dfsadmin -refreshNodes

**查看状态 ./bin/hdfs dfsadmin -report**

Normal 正常状态

Decommissioned in Program 数据正在迁移

Decommissioned 数据迁移完成

**下线newnode**

**注意:只有当状态变成 Decommissioned 才能 down机下线**

[root@newnode hadoop]# ./sbin/hadoop-daemon.sh stop datanode

**Hadoop节点管理**

yarn 增加 nodemanager

./sbin/yarn-daemon.sh start nodemanager

yarn 停止 nodemanager

./sbin/yarn-daemon.sh stop nodemanager

yarn 查看节点状态

./bin/yarn node -list

**NFS 网关**

**1 配置 /etc/hosts (NFSGW**)

192.168.1.10 nn01

192.168.1.11 node1

192.168.1.12 node2

192.168.1.13 node3

192.168.1.16 nfsgw

**2 添加用户(nfsgw, nn01)**

groupadd -g 200 nsd1803

useradd -u 200 -g 200 nsd1803

**NN01: 3 停止集群**

./sbin/stop-all.sh

**NN01: 4 增加配置 core-site.xml**

<property>

<name>hadoop.proxyuser.nsd1803.groups</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.nsd1803.hosts</name>

<value>\*</value>

</property>

**NN01: 5 同步配置到 node1 node2 node3**

NN01: vim slaves 删除里面的newnode 主机名

[root@nn01 hadoop]# > exclude #清除newnode

**NN01: 6 启动集群 ./sbin/start-dfs.sh**

**NN01: 7 查看状态**

./bin/hdfs dfsadmin -report

**NFS 网关**

• 配置步骤

– 1、启动一个新的系统,禁用 selinux、禁用 firewalld

– 2、卸载 rpcbind 、nfs-utils

– 3、配置 /etc/hosts,添加所有 namenode 和datanode 的主机名与ip 对应关系

– 4、安装 java openjdk 的运行环境

– 5、同步 namenode 的 /usr/local/hadoop 到本机

– 6、配置 hdfs-site.xml

– 7、启动服务

**NFSGW:** 安装 java-1.8.0-openjdk-devel

**NFSGW:** 同步 nn01 的 /usr/local/hadoop 到NFSGW的相同目录下

# scp -r /usr/local/hadoop/ root@192.168.1.16:/usr/local/

**NFSGW:** hdfs-site.xml 增加配置

<property>

<name>nfs.exports.allowed.hosts</name>

<value>\* rw</value>

</property>

<property>

<name>nfs.dump.dir</name>

<value>/var/nfstmp</value>

</property>

**NFSGW: 创建转储目录，并给用户 nsd1803 赋权**

mkdir /var/nfstmp

chown nsd1803:nsd1803 /var/nfstmp

**NFSGW: 给 /usr/local/hadoop/logs 赋权**

setfacl -m u:nsd1803:rwx logs

getfacl logs

**NFSGW:创建数据根目录 /var/hadoop**

[root@nfsgw ~]# mkdir /var/hadoop

**NFSGW:测试是否成功**

[root@nfsgw hadoop]# sudo -u nsd1803 -s

[nsd1803@nfsgw hadoop]$ cd /var/nfstmp/

[nsd1803@nfsgw nfstmp]$ touch aaa

[nsd1803@nfsgw nfstmp]$ rm aaa

[nsd1803@nfsgw nfstmp]$ cd /usr/local/hadoop/logs/

[nsd1803@nfsgw logs]$ touch fff

[nsd1803@nfsgw logs]$ rm -rf fff

• 启动与挂载

**– 设置 /usr/local/hadoop/logs 权限,为代理用户赋予读写执行的权限**

[root@nfsgw hadoop]# setfacl -m user:proxyuser:rwx /usr/local/hadoop/logs

**– 使用 root 用户启动 portmap 服务**

[root@nfsgw hadoop]# ./sbin/hadoop-daemon.sh --script ./bin/hdfs start portmap

[root@nfsgw hadoop]# jps

1453 Jps

1407 Portmap

**– 使用代理用户启动 nfs3**

[nsd1803@nfsgw hadoop]$ ./sbin/hadoop-daemon.sh --script ./bin/hdfs start nfs3

[nsd1803@nfsgw hadoop]$ jps

1520 Nfs3

1573 Jps

NFS 网关

• 这里要特别注意:

– 启动 portmap 需要使用 root 用户

– 启动 nfs3 需要使用 core-site 里面设置的代理用户

– **必须先启动 portmap 之后再启劢 nfs3**

– 如果 portmap 重启了,在重启之后 nfs3 也需要重启

**Client: 安装 nfs-utils**

[root@newnode ~]# rm -rf /usr/local/hadoop/ #模拟客户端

[root@newnode ~]# yum -y install nfs-utils

**mount 共享目录**

mount -t nfs -o vers=3,proto=tcp,nolock,noatime,sync,noacl 192.168.1.16:/ /mnt/

[root@newnode ~]# cd /mnt/

[root@newnode mnt]# ls

**Client: 实现开机自动挂载**

[root@newnode mnt]# vim /etc/fstab

**查看注册服务**

rpcinfo -p 192.168.1.15

**查看共享目录**

showmount -e 192.168.1.15

# Linux 大数据

# NSD HADOOP DAY03

<http://zookeeper.apache.org/doc/r3.4.10/zookeeperAdmin.html> #官方文档

**zookeeper 安装**

[root@nn01 ~]# lftp 192.168.1.254

lftp 192.168.1.254:/public> get zookeeper-3.4.10.tar.gz

[root@nn01 ~]# tar -zxf zookeeper-3.4.10.tar.gz

[root@nn01 ~]# mv zookeeper-3.4.10 /usr/local/zookeeper

[root@nn01 ~]# cd /usr/local/zookeeper

[root@nn01 local]# chown -R root:root zookeeper

[root@nn01 zookeeper]# cd conf/

[root@nn01 conf]# mv zoo\_sample.cfg zoo.cfg

**1 配置 /etc/hosts ,所有集群主机可以相互 ping 通**

**2 安装 java-1.8.0-openjdk-devel**

**3 zookeeper 解压拷贝到 /usr/local/zookeeper**

**4 配置文件改名，并在最后添加配置**

mv zoo\_sample.cfg zoo.cfg

zoo.cfg 最后添加配置

server.1=node1:2888:3888

server.2=node2:2888:3888

server.3=node3:2888:3888

server.4=nn01:2888:3888:observer

**5 拷贝 /usr/local/zookeeper 到其他集群主机**

[root@nn01 conf]# scp -r /usr/local/zookeeper/ 192.168.1.11:/usr/local/

[root@nn01 conf]# scp -r /usr/local/zookeeper/ 192.168.1.12:/usr/local/

[root@nn01 conf]# scp -r /usr/local/zookeeper/ 192.168.1.13:/usr/local/

**6 创建 mkdir /tmp/zookeeper**

ALL:**7 创建 myid 文件，id 必须与配置文件里主机名对应的 server.(id)一致**

[root@nn01 conf]# mkdir /tmp/zookeeper

[root@nn01 conf]# echo 4 > /tmp/zookeeper/myid

[root@nn01 conf]# ssh node1 mkdir /tmp/zookeeper

[root@nn01 conf]# ssh node2 mkdir /tmp/zookeeper

[root@nn01 conf]# ssh node3 mkdir /tmp/zookeeper

[root@nn01 ~]# ssh node1 "echo 1 >/tmp/zookeeper/myid"

[root@nn01 ~]# ssh node2 "echo 2 >/tmp/zookeeper/myid"

[root@nn01 ~]# ssh node3 "echo 3 >/tmp/zookeeper/myid"

**8 启动服务，单启动一台无法查看状态，需要启动全部集群以后才能查看状态**

/usr/local/zookeeper/bin/zkServer.sh start

[root@nn01 conf]# cd

[root@nn01 ~]# /usr/local/zookeeper/bin/zkServer.sh start

[root@nn01 ~]# ssh node1 /usr/local/zookeeper/bin/zkServer.sh start

[root@nn01 ~]# ssh node2 /usr/local/zookeeper/bin/zkServer.sh start

[root@nn01 ~]# ssh node3 /usr/local/zookeeper/bin/zkServer.sh start

**9 查看状态**

/usr/local/zookeeper/bin/zkServer.sh status

[root@nn01 ~]# /usr/local/zookeeper/bin/zkServer.sh status

Mode: observer #在nn01上看见是observer表示成功

**查看其他机器的状态**

[root@nn01 ~]# yum -y install telnet

[root@nn01 ~]# telnet node3 2181 #2181端口

ruok #输入

imokConnection closed by foreign host. #回复inok证明活着

**利用 api 查看状态的脚本**

#!/bin/bash

function getstatus(){

exec 9<>/dev/tcp/$1/2181 2>/dev/null

echo stat >&9

MODE=$(cat <&9 |grep -Po "(?<=Mode:).\*")

exec 9<&-

echo ${MODE:-NULL}

}

for i in node{1..3} nn01;do

echo -ne "${i}\t"

getstatus ${i}

done

### kafka 搭建

**1 下载解压 kafka 压缩包**

[root@node1 ~]# lftp 192.168.1.254

lftp 192.168.1.254:/public> get kafka\_2.10-0.10.2.1.tgz

[root@node1 ~]# tar -zxf kafka\_2.10-0.10.2.1.tgz

**2 把 kafka 拷贝到 /usr/local/kafka 下面**

[root@node1 ~]# mv kafka\_2.10-0.10.2.1 /usr/local/kafka

[root@node1 ~]# cd /usr/local/kafka

[root@node1 kafka]# cd config/

**3 修改配置文件** /usr/local/kafka/config/server.properties

21 broker.id=11

119 zookeeper.connect=node1:2181,node2:2181,node3:2181

**4 拷贝 kafka 到其他主机，并修改 broker.id ,不能重复**

[root@node1 config]# scp -r /usr/local/kafka/ node2:/usr/local/

[root@node1 config]# scp -r /usr/local/kafka/ node3:/usr/local/

[root@node2 ~]# vim /usr/local/kafka/config/server.properties

broker.id=12

[root@node3 ~]# vim /usr/local/kafka/config/server.properties

broker.id=13

**5 启动 kafka 集群(node1、2、3)**

/usr/local/kafka/bin/kafka-server-start.sh -daemon /usr/local/kafka/config/server.properties #启动

[root@node3 ~]# jps

1854 Kafka #看见有kafka

/usr/local/kafka/bin/kafka-server-stop.sh -daemon /usr/local/kafka/config/server.properties **#关闭kafka集群**

**验证集群**

创建一个 topic

./bin/kafka-topics.sh --create --partitions 1 --replication-factor 1 --zookeeper node3:2181 --topic nsd1803

生产者

./bin/kafka-console-producer.sh --broker-list node2:9092 --topic nsd1803

消费者

./bin/kafka-console-consumer.sh --bootstrap-server node1:9092 --topic nsd1803

## hadoop 高可用

**ALL: 配置 /etc/hosts**

192.168.1.10 nn01

192.168.1.15 nn02

192.168.1.11 node1

192.168.1.12 node2

192.168.1.13 node3

**ALL: 除了 zookeeper 其他 hadoop ，kafka 服务全部停止**

[root@nn01 hadoop]# ./sbin/stop-all.sh

[root@nn01 hadoop]# rm -rf logs

[root@nn01 hadoop]# /usr/local/zookeeper/bin/zkServer.sh stop

[root@node2 kafka]# /usr/local/kafka/bin/kafka-server-stop.sh -daemon /usr/local/kafka/config/server.properties #关闭kafka集群

**ALL: 初始化 hdfs 集群，删除 /var/hadoop/\***

[root@nn01 ~]# rm -rf /var/hadoop/\*

[root@nn01 ~]# ssh node1 rm -rf /var/hadoop/\*

[root@nn01 ~]# ssh node2 rm -rf /var/hadoop/\*

[root@nn01 ~]# ssh node3 rm -rf /var/hadoop/\*

[root@nn01 ~]# ssh nn02 rm -rf /var/hadoop/\*

[root@nn01 ~]# rm -rf /var/hadoop/\*

**NN2: 关闭 ssh key 验证，部署公钥私钥**

[root@nn02 ~]# vim /etc/ssh/ssh\_config

35 # StrictHostKeyChecking #复制该行

58 Host \* #添加在该行后面

StrictHostKeyChecking no

[root@nn01 ~]# scp /root/.ssh/id\_rsa nn02:/root/.ssh/

[root@nn01 ~]# scp /root/.ssh/authorized\_keys nn02:/root/.ssh/

**NN1: 配置 core-site.xml**

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://nsdcluster</value>

</property>

<property>

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

<value>/var/hadoop</value>

</property>

<property>

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

<value>node1:2181,node2:2181,node3:2181</value>

</property>

<property>

<name>hadoop.proxyuser.nsd1803.groups</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.nsd1803.hosts</name>

<value>\*</value>

</property>

</configuration>

**配置 hdfs-site.xml**

<configuration>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.nameservices</name>

<value>nsdcluster</value>

</property>

<property>

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

<value>nn1,nn2</value>

</property>

<property>

<name>dfs.namenode.rpc-address.nsdcluster.nn1</name>

<value>nn01:8020</value>

</property>

<property>

<name>dfs.namenode.rpc-address.nsdcluster.nn2</name>

<value>nn02:8020</value>

</property>

<property>

<name>dfs.namenode.http-address.nsdcluster.nn1</name>

<value>nn01:50070</value>

</property>

<property>

<name>dfs.namenode.http-address.nsdcluster.nn2</name>

<value>nn02:50070</value>

</property>

<property>

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

<value>qjournal://node1:8485;node2:8485;node3:8485/nsdcluster</value>

</property>

<property>

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

<value>/var/hadoop/journal</value>

</property>

<property>

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

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

</property>

<property>

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

<value>sshfence</value>

</property>

<property>

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

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

</property>

<property>

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

<value>true</value>

</property>

</configuration>

**yarn-site.xml 配置文件**

<configuration>

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

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

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

<value>true</value>

</property>

<property>

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

<value>rm1,rm2</value>

</property>

<property>

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

<value>true</value>

</property>

<property>

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

<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>

</property>

<property>

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

<value>node1:2181,node2:2181,node3:2181</value>

</property>

<property>

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

<value>yarn-ha</value>

</property>

<property>

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

<value>nn01</value>

</property>

<property>

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

<value>nn02</value>

</property>

</configuration>

#-----------------------------------------------------#

**初始化启动集群**

**ALL: 所有机器**

**nodeX： node1 node2 node3**

**NN1: nn01**

**NN2: nn02**

#-----------------------------------------------------#

**ALL: 同步配置到所有集群机器**

[root@nn01 ~]# scp -r /usr/local/hadoop/ root@192.168.1.11:/usr/local/

[root@nn01 ~]# scp -r /usr/local/hadoop/ root@192.168.1.12:/usr/local/

[root@nn01 ~]# scp -r /usr/local/hadoop/ root@192.168.1.13:/usr/local/

[root@nn01 ~]# scp -r /usr/local/hadoop/ root@192.168.1.15:/usr/local/

[root@nn01 ~]# ssh node1 rm -rf /usr/local/hadoop/logs #清空好排错

[root@nn01 ~]# ssh node2 rm -rf /usr/local/hadoop/logs

[root@nn01 ~]# ssh node3 rm -rf /usr/local/hadoop/logs

**NN1: 初始化ZK集群**  ./bin/hdfs zkfc -formatZK

**nodeX: 启动 journalnode 服务**

./sbin/hadoop-daemon.sh start journalnode

[root@node3 hadoop]# jps

2240 JournalNode #看见有JournalNode

**NN1: 格式化**  ./bin/hdfs namenode -format

[root@nn01 hadoop]# ./bin/hdfs namenode -format

**NN2: 数据同步到本地** /var/hadoop/dfs

[root@nn01 hadoop]# scp -r /var/hadoop/dfs/ root@192.168.1.15:/var/hadoop/

**NN1: 初始化 JNS** ./bin/hdfs namenode -initializeSharedEdits

[root@nn01 hadoop]# ./bin/hdfs namenode -initializeSharedEdits

#输入y

nodeX: 停止 journalnode 服务

./sbin/hadoop-daemon.sh stop journalnode

[root@node1 hadoop]# ./sbin/hadoop-daemon.sh stop journalnode

[root@node2 hadoop]# ./sbin/hadoop-daemon.sh stop journalnode

[root@node3 hadoop]# ./sbin/hadoop-daemon.sh stop journalnode

#集群初始化完毕

#-----------------------------------------------------#

启动集群

NN1: ./sbin/start-all.sh

NN2: ./sbin/yarn-daemon.sh start resourcemanager

查看集群状态

./bin/hdfs haadmin -getServiceState nn1

./bin/hdfs haadmin -getServiceState nn2

./bin/yarn rmadmin -getServiceState rm1

./bin/yarn rmadmin -getServiceState rm2

./bin/hdfs dfsadmin -report

./bin/yarn node -list

访问集群：

./bin/hadoop fs -ls /

./bin/hadoop fs -mkdir hdfs://nsdcluster/input

[root@nn01 hadoop]# ./bin/hadoop fs -mkdir /oo

#初始化后里面都是空的，新创建一个文件夹

验证高可用，关闭 active namenode

./sbin/hadoop-daemon.sh stop namenode

./sbin/yarn-daemon.sh stop resourcemanager

恢复节点

./sbin/hadoop-daemon.sh start namenode

./sbin/yarn-daemon.sh start resourcemanager