TTS 10.0 COOKBOOK

（NSD ARCHITECTURE DAY07）

版本编号 10.0

2018-08

达内 IT 培训集团

NSD ARCHITECTURE DAY07

1. 案例 1：Zookeeper 安装

- **问题**

本案例要求：

- 搭建 Zookeeper 集群并查看各服务器的角色
- 停止 Leader 并查看各服务器的角色

- **步骤**

实现此案例需要按照如下步骤进行。

2. 步骤一：安装 Zookeeper

1）编辑/etc/hosts ,所有集群主机可以相互 ping 通（在 nn01 上面配置，同步到 node1，node2，node3）

```
[root@nn01 hadoop]# vim /etc/hosts
192.168.1.21  nn01
192.168.1.22  node1
192.168.1.23  node2
192.168.1.24  node3
192.168.1.25  node4

[root@nn01 hadoop]# for i in {22..24}  \
do   \
scp /etc/hosts 192.168.1.$i:/etc/  \
done      //同步配置
hosts       100%  253   639.2KB/s   00:00
hosts       100%  253   497.7KB/s   00:00
hosts       100%  253   662.2KB/s   00:00
```

2）安装 java-1.8.0-openjdk-devel,由于之前的 hadoop 上面已经安装过，这里不再安装，若是新机器要安装

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

```
[root@nn01 ~]# tar -xf zookeeper-3.4.10.tar.gz
[root@nn01 ~]# mv zookeeper-3.4.10 /usr/local/zookeeper
```

4）配置文件改名，并在最后添加配置

```
[root@nn01 ~]# cd /usr/local/zookeeper/conf/
[root@nn01 conf]# ls
configuration.xsl  log4j.properties  zoo_sample.cfg
[root@nn01 conf]# mv zoo_sample.cfg  zoo.cfg
[root@nn01 conf]# chown root.root zoo.cfg
```

```
[root@nn01 conf]# vim zoo.cfg
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]# for i in {22..24}; do rsync -aSH --delete /usr/local/zookeeper/
192.168.1.$i:/usr/local/zookeeper  -e 'ssh' & done
[4] 4956
[5] 4957
[6] 4958
```

6）创建 mkdir /tmp/zookeeper，每一台都要

```
[root@nn01 conf]# mkdir /tmp/zookeeper
[root@nn01 conf]# ssh node1 mkdir /tmp/zookeeper
[root@nn01 conf]# ssh node2 mkdir /tmp/zookeeper
[root@nn01 conf]# ssh node3 mkdir /tmp/zookeeper
```

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]# echo 4 >/tmp/zookeeper/myid
[root@nn01 conf]# ssh node1 'echo 1 >/tmp/zookeeper/myid'
[root@nn01 conf]# ssh node2 'echo 2 >/tmp/zookeeper/myid'
[root@nn01 conf]# 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，每一台上面都要手工启动（以 nn01 为例子）

```
[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

注意：刚启动 zookeeper 查看状态的时候报错，启动的数量要保证半数以上，这时再去看就成功了

9）查看状态

```
[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Mode: observe
[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh stop
//关闭之后查看状态其他服务器的角色
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED

[root@nn01 conf]# yum -y install telnet
[root@nn01 conf]# telnet node3 2181
Trying 192.168.1.24...
Connected to node3.
Escape character is '^]'.
ruok     //发送
imokConnection closed by foreign host.      //imok 回应的结果
```

10）利用 api 查看状态（nn01 上面操作）

```
[root@nn01 conf]# /usr/local/zookeeper/bin/zkServer.sh start
[root@nn01 conf]# vim api.sh
#!/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
[root@nn01 conf]# chmod 755 api.sh
[root@nn01 conf]# ./api.sh
node1    follower
node2    leader
node3    follower
nn01 observer
```

3. 案例 2：Kafka 集群实验

- **问题**

本案例要求：

- 利用 Zookeeper 搭建一个 Kafka 集群
- 创建一个 topic
- 模拟生产者发布消息
- 模拟消费者接收消息

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：搭建 Kafka 集群

1）解压 kafka 压缩包

Kafka 在 node1，node2，node3 上面操作即可

```
[root@node1 ~]# tar -xf 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
```

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

```
[root@node1 ~]# cd /usr/local/kafka/config
[root@node1 config]# vim server.properties
broker.id=22
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]# for i in 23 24; do rsync -aSH --delete /usr/local/kafka
192.168.1.$i:/usr/local/; done
[1] 27072
[2] 27073

[root@node2 ~]# vim /usr/local/kafka/config/server.properties
//node2 主机修改
broker.id=23
[root@node3 ~]# vim /usr/local/kafka/config/server.properties
//node3 主机修改
broker.id=24
```

5）启动 kafka 集群（node1，node2，node3 启动）

```
[root@node1 local]# /usr/local/kafka/bin/kafka-server-start.sh -daemon
/usr/local/kafka/config/server.properties
[root@node1 local]# jps        //出现 kafka
26483 DataNode
27859 Jps
27833 Kafka
26895 QuorumPeerMain
```

6）验证配置，创建一个 topic

```
[root@node1 local]# /usr/local/kafka/bin/kafka-topics.sh --create --partitions 1
--replication-factor 1 --zookeeper node3:2181 --topic aa
Created topic "aa".
```

7）模拟生产者，发布消息

```
[root@node2 ~]# /usr/local/kafka/bin/kafka-console-producer.sh \
--broker-list node2:9092 --topic aa        //写一个数据
ccc
ddd
```

9）模拟消费者，接收消息

```
[root@node3 ~]# /usr/local/kafka/bin/kafka-console-consumer.sh \
--bootstrap-server node1:9092 --topic aa        //这边会直接同步
ccc
ddd
```

注意：kafka 比较吃内存，做完这个 kafka 的实验可以把它停了

4. 案例 3：Hadoop 高可用

- **问题**

本案例要求：

- 配置 Hadoop 的高可用
- 修改配置文件

- **方案**

配置 Hadoop 的高可用，解决 NameNode 单点故障问题，使用之前搭建好的 hadoop 集群，新添加一台 nn02，ip 为 192.168.1.25，之前有一台 node4 主机，可以用这台主机，具体要求如图-1 所示：

主机	角色	软件
192.168.1.21	NameNode1	Hadoop
192.168.1.25	NameNode2	Hadoop
192.168.1.22 Node1	DataNode journalNode Zookeeper	HDFS Zookeeper
192.168.1.23 Node2	DataNode journalNode Zookeeper	HDFS Zookeeper
192.168.1.24 Node3	DataNode journalNode Zookeeper	HDFS Zookeeper

图-1

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：hadoop 的高可用

1）停止所有服务（由于 kafka 的实验做完之后就已经停止，这里不在重复）

```
[root@nn01 ~]# cd /usr/local/hadoop/
[root@nn01 hadoop]# ./sbin/stop-all.sh  //停止所有服务
```

2）启动 zookeeper（需要一台一台的启动）这里以 nn01 为例子

```
[root@nn01 hadoop]# /usr/local/zookeeper/bin/zkServer.sh start
[root@nn01 hadoop]# sh /usr/local/zookeeper/conf/api.sh //利用之前写好的脚本查看
node1    follower
node2    leader
node3    follower
nn01 observer
```

3）新加一台机器 nn02，这里之前有一台 node4，可以用这个作为 nn02

```
[root@node4 ~]# echo nn02 > /etc/hostname
[root@node4 ~]# hostname nn02
```

4）修改 vim /etc/hosts

```
[root@nn01 hadoop]# vim /etc/hosts
192.168.1.21  nn01
192.168.1.25  nn02
192.168.1.22  node1
192.168.1.23  node2
192.168.1.24  node3
```

5）同步到 nn02，node1，node2，node3

```
[root@nn01 hadoop]# for i in {22..25}; do rsync -aSH --delete /etc/hosts 192.168.1.$i:/etc/hosts -e 'ssh' & done
[1] 14355
[2] 14356
[3] 14357
[4] 14358
```

6）配置 SSH 信任关系

注意：nn01 和 nn02 互相连接不需要密码，nn02 连接自己和 node1，node2，node3 同样不需要密码

```
[root@nn02 ~]# vim /etc/ssh/ssh_config
Host *
        GSSAPIAuthentication yes
        StrictHostKeyChecking no
[root@nn01 hadoop]# cd /root/.ssh/
[root@nn01 .ssh]# scp id_rsa id_rsa.pub  nn02:/root/.ssh/
//把 nn01 的公钥私钥考给 nn02
```

7）所有的主机删除/var/hadoop/*

```
[root@nn01 .ssh]# rm -rf /var/hadoop/*
[root@nn01 .ssh]# ssh nn02 rm -rf /var/hadoop/*
[root@nn01 .ssh]# ssh node1 rm -rf /var/hadoop/*
[root@nn01 .ssh]# ssh node2 rm -rf /var/hadoop/*
[root@nn01 .ssh]# ssh node3 rm -rf /var/hadoop/*
```

8）配置 core-site

```
[root@nn01 .ssh]# vim /usr/local/hadoop/etc/hadoop/core-site.xml
<configuration>
<property>
        <name>fs.defaultFS</name>
        <value>hdfs://nsdcluster</value>
//nsdcluster 是随便起的名。相当于一个组，访问的时候访问这个组
    </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>  //zookeepe 的地址
    </property>
    <property>
        <name>hadoop.proxyuser.nfs.groups</name>
        <value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.nfs.hosts</name>
        <value>*</value>
```

```
        </property>
    </configuration>
```

9）配置 hdfs-site

```
    [root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/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>
    //nn1,nn2 名称固定，是内置的变量，nsdcluster 里面有 nn1，nn2
            <value>nn1,nn2</value>
        </property>
        <property>
            <name>dfs.namenode.rpc-address.nsdcluster.nn1</name>
    //声明 nn1  8020 为通讯端口，是 nn01 的 rpc 通讯端口
            <value>nn01:8020</value>
        </property>
        <property>
            <name>dfs.namenode.rpc-address.nsdcluster.nn2</name>
    //声明 nn2 是谁，nn02 的 rpc 通讯端口
            <value>nn02:8020</value>
        </property>
        <property>
            <name>dfs.namenode.http-address.nsdcluster.nn1</name>
    //nn01 的 http 通讯端口
            <value>nn01:50070</value>
        </property>
        <property>
            <name>dfs.namenode.http-address.nsdcluster.nn2</name>
    //nn01 和 nn02 的 http 通讯端口
            <value>nn02:50070</value>
        </property>
        <property>
            <name>dfs.namenode.shared.edits.dir</name>
    //指定 namenode 元数据存储在 journalnode 中的路径
            <value>qjournal://node1:8485;node2:8485;node3:8485/nsdcluster</value>
        </property>
        <property>
            <name>dfs.journalnode.edits.dir</name>
    //指定 journalnode 日志文件存储的路径
            <value>/var/hadoop/journal</value>
        </property>
        <property>
            <name>dfs.client.failover.proxy.provider.nsdcluster</name>
    //指定 HDFS 客户端连接 active  namenode 的 java 类

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</v
alue>
        </property>
        <property>
            <name>dfs.ha.fencing.methods</name>                    //配置隔离机制为 ssh
            <value>sshfence</value>
        </property>
```

```
        <property>
            <name>dfs.ha.fencing.ssh.private-key-files</name>     //指定密钥的位置
            <value>/root/.ssh/id_rsa</value>
        </property>
        <property>
            <name>dfs.ha.automatic-failover.enabled</name>        //开启自动故障转移
            <value>true</value>
        </property>
    </configuration>
```

10）配置 yarn-site

```
    [root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/yarn-site.xml
    <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>      //rm1,rm2 代表 nn01 和 nn02
            <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>
```

11）同步到 nn02，node1，node2，node3

```
    [root@nn01 ~]# for i in {22..25}; do rsync -aSH --delete /usr/local/hadoop/
192.168.1.$i:/usr/local/hadoop  -e 'ssh' & done
    [1] 25411
    [2] 25412
```

```
    [3] 25413
    [4] 25414
```

12）删除所有机器上面的/user/local/hadoop/logs，方便排错

```
    [root@nn01 ~]# for i in {21..25}; do ssh 192.168.1.$i rm -rf /usr/local/hadoop/logs ;
done
```

13）同步配置

```
    [root@nn01 ~]# for i in {22..25}; do rsync -aSH --delete /usr/local/hadoop
192.168.1.$i:/usr/local/hadoop -e 'ssh' & done

    [1] 28235
    [2] 28236
    [3] 28237
    [4] 28238
```

5. 案例 4：高可用验证

- **问题**

本案例要求：

- 初始化集群
- 验证集群

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：验证 hadoop 的高可用

1）初始化 ZK 集群

```
    [root@nn01 ~]# /usr/local/hadoop/bin/hdfs zkfc -formatZK
    ...
    18/09/11 15:43:35 INFO ha.ActiveStandbyElector: Successfully created
/hadoop-ha/nsdcluster in ZK    //出现 Successfully 即为成功
    ...
```

2）在 node1，node2，node3 上面启动 journalnode 服务（以 node1 为例）

```
    [root@node1 ~]# /usr/local/hadoop/sbin/hadoop-daemon.sh start journalnode
    starting journalnode, logging to
/usr/local/hadoop/logs/hadoop-root-journalnode-node1.out
    [root@node1 ~]# jps
    29262 JournalNode
    26895 QuorumPeerMain
    29311 Jps
```

3）格式化，先在 node1，node2，node3 上面启动 journalnode 才能格式化

```
    [root@nn01 ~]# /usr/local/hadoop//bin/hdfs  namenode  -format
    //出现 Successfully 即为成功
```

```
[root@nn01 hadoop]# ls /var/hadoop/
dfs
```

4）nn02 数据同步到本地 /var/hadoop/dfs

```
[root@nn02 ~]# cd /var/hadoop/
[root@nn02 hadoop]# ls
[root@nn02 hadoop]# rsync -aSH  nn01:/var/hadoop/  /var/hadoop/
[root@nn02 hadoop]# ls
dfs
```

5）初始化 JNS

```
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs namenode -initializeSharedEdits
18/09/11 16:26:15 INFO client.QuorumJournalManager: Successfully started new epoch
1       //出现 Successfully，成功开启一个节点
```

6）停止 journalnode 服务（node1，node2，node3）

```
[root@node1 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh stop journalnode
stopping journalnode
[root@node1 hadoop]# jps
29346 Jps
26895 QuorumPeerMain
```

步骤二：启动集群

1）nn01 上面操作

```
[root@nn01 hadoop]# /usr/local/hadoop/sbin/start-all.sh  //启动所有集群
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [nn01 nn02]
nn01:            starting            namenode,            logging            to
/usr/local/hadoop/logs/hadoop-root-namenode-nn01.out
nn02:            starting            namenode,            logging            to
/usr/local/hadoop/logs/hadoop-root-namenode-nn02.out
node2:            starting            datanode,            logging            to
/usr/local/hadoop/logs/hadoop-root-datanode-node2.out
node3:            starting            datanode,            logging            to
/usr/local/hadoop/logs/hadoop-root-datanode-node3.out
node1:            starting            datanode,            logging            to
/usr/local/hadoop/logs/hadoop-root-datanode-node1.out
Starting journal nodes [node1 node2 node3]
node1:            starting           journalnode,           logging            to
/usr/local/hadoop/logs/hadoop-root-journalnode-node1.out
node3:            starting           journalnode,           logging            to
/usr/local/hadoop/logs/hadoop-root-journalnode-node3.out
node2:            starting           journalnode,           logging            to
/usr/local/hadoop/logs/hadoop-root-journalnode-node2.out
Starting ZK Failover Controllers on NN hosts [nn01 nn02]
nn01: starting zkfc, logging to /usr/local/hadoop/logs/hadoop-root-zkfc-nn01.out
nn02: starting zkfc, logging to /usr/local/hadoop/logs/hadoop-root-zkfc-nn02.out
starting yarn daemons
starting               resourcemanager,               logging               to
/usr/local/hadoop/logs/yarn-root-resourcemanager-nn01.out
node2:            starting           nodemanager,           logging            to
/usr/local/hadoop/logs/yarn-root-nodemanager-node2.out
node1:            starting           nodemanager,           logging            to
/usr/local/hadoop/logs/yarn-root-nodemanager-node1.out
node3:            starting           nodemanager,           logging            to
/usr/local/hadoop/logs/yarn-root-nodemanager-node3.out
```

2）nn02 上面操作

```
[root@nn02 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh start resourcemanager
starting               resourcemanager,               logging               to
/usr/local/hadoop/logs/yarn-root-resourcemanager-nn02.out
```

3）查看集群状态

```
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
active
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn2
standby
[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
active
[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm2
standby
```

4）查看节点是否加入

```
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs dfsadmin -report
...
Live datanodes (3):    //会有三个节点
...
[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn  node  -list
Total Nodes:3
        Node-Id       Node-State   Node-Http-Address
 Number-of-Running-Containers
     node2:43307          RUNNING         node2:8042                            0
     node1:34606          RUNNING         node1:8042                            0
     node3:36749          RUNNING         node3:8042                            0
```

步骤三：访问集群

1）查看并创建

```
[root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop  fs -ls  /
[root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop  fs -mkdir /aa //创建 aa
[root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop  fs -ls  /      //再次查看

Found 1 items
drwxr-xr-x   - root supergroup          0 2018-09-11 16:54 /aa

[root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop  fs -put *.txt /aa
[root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop  fs -ls hdfs://nsdcluster/aa
//也可以这样查看
Found 3 items
-rw-r--r--          2     root     supergroup     86424     2018-09-11     17:00
hdfs://nsdcluster/aa/LICENSE.txt
-rw-r--r--          2     root     supergroup     14978     2018-09-11     17:00
hdfs://nsdcluster/aa/NOTICE.txt
-rw-r--r--  2 root supergroup 1366 2018-09-11 17:00 hdfs://nsdcluster/aa/README.txt
```

2）验证高可用，关闭 active namenode

```
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
active
[root@nn01 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh stop namenode
stopping namenode
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
//再次查看会报错
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn2
//nn02 由之前的 standby 变为 active
```

```
active

[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
active
[root@nn01 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh stop resourcemanager
//停止 resourcemanager
[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm2
active
```

3） 恢复节点

```
[root@nn01 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh start namenode
//启动 namenode
[root@nn01 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh start resourcemanager
//启动 resourcemanager
[root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
//查看
[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
//查看
```