NSD ARCHITECTURE DAY07

1 案例1：Zookeeper安装

1.1 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

1.2 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）

```
01.     [root@nn01 hadoop]# vim /etc/hosts
02.     192.168.1.21  nn01
03.     192.168.1.22  node1
04.     192.168.1.23  node2
05.     192.168.1.24  node3
06.     192.168.1.25  node4
07.
08.     [root@nn01 hadoop]# for i in {22..24}  \
09.     do  \
10.     scp /etc/hosts 192.168.1.$i:/etc/   \
11.     done      //同步配置
12.     hosts      100%  253   639.2KB/s   00:00
13.     hosts      100%  253   497.7KB/s   00:00
14.     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

```
01.    [root@nn01 ~]# tar -xf zookeeper-3.4.10.tar.gz
02.    [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

```
01.    [root@nn01 ~]# cd /usr/local/zookeeper/conf/
02.    [root@nn01 conf]# ls
03.    configuration.xsl log4j.properties zoo_sample.cfg
04.    [root@nn01 conf]# mv zoo_sample.cfg zoo.cfg
05.    [root@nn01 conf]# chown root.root zoo.cfg
06.    [root@nn01 conf]# vim zoo.cfg
07.    server.1=node1:2888:3888
08.    server.2=node2:2888:3888
09.    server.3=node3:2888:3888
10.    server.4=nn01:2888:3888:observer
```

5）拷贝 /usr/local/zookeeper 到其他集群主机

```
01.    [root@nn01 conf]# for i in {22..24}; do rsync -aSH --delete /usr/local/zookeeper/ 192.:
02.    [4] 4956
03.    [5] 4957
04.    [6] 4958
```

6）创建 mkdir /tmp/zookeeper，每一台都要

```
01.    [root@nn01 conf]# mkdir /tmp/zookeeper
02.    [root@nn01 conf]# ssh node1 mkdir /tmp/zookeeper
03.    [root@nn01 conf]# ssh node2 mkdir /tmp/zookeeper
04.    [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

Top

```
01.    [root@nn01 conf]# echo 4 >/tmp/zookeeper/myid
02.    [root@nn01 conf]# ssh node1 'echo 1 >/tmp/zookeeper/myid'
```

03.   [root@nn01 conf] # ssh node2 'echo 2 >/tmp/zookeeper/myid'

04.   [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）

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

02.   ZooKeeper JMX enabled by default

03.   Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

04.   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

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

02.   ZooKeeper JMX enabled by default

03.   Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

04.   Mode: observe

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

06.   //关闭之后查看状态其他服务器的角色

07.   ZooKeeper JMX enabled by default

08.   Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg

09.   Stopping zookeeper ... STOPPED

10.

11.   [root@nn01 conf] # yum -y install telnet

12.   [root@nn01 conf] # telnet node3 2181

13.   Trying 192.168.1.24...

14.   Connected to node3.

15.   Escape character is '^]'.

16.   ruok   //发送

17.   imokConnection closed by foreign host.   //imok回应的结果

10）利用 api 查看状态（nn01上面操作）

01.   [root@nn01 conf] # /usr/local/zookeeper/bin/zkServer.sh start   **Top**

02.   [root@nn01 conf] # vim api.sh

03.   #! /bin/bash

```
04.    function getstatus(){
05.        exec 9<>/dev/tcp/$1/2181 2>/dev/null
06.        echo stat >&9
07.        MODE=$(cat <&9 | grep -Po "(?<=Mode:).*")
08.        exec 9<&-
09.        echo ${MODE:-NULL}
10.    }
11.    for i in node{1..3} nn01;do
12.        echo -ne "${i}\t"
13.        getstatus ${i}
14.    done
15.    [root@nn01 conf]# chmod 755 api.sh
16.    [root@nn01 conf]# ./api.sh
17.    node1    follower
18.    node2    leader
19.    node3    follower
20.    nn01    observer
```

3 案例2：Kafka集群实验

3.1 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

3.2 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

```
01.    [root@node1 ~]# tar -xf kafka_2.10-0.10.2.1.tgz
```

2）把 kafka 拷贝到 /usr/local/kafka 下面

```
01.    [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

```
01.     [root@node1 ~]# cd /usr/local/kafka/config
02.     [root@node1 config]# vim server.properties
03.     broker.id=22
04.     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

```
01.     [root@node1 config]# for i in 23 24; do rsync -aSH --delete /usr/local/kafka 192.168.1.$
02.     [1] 27072
03.     [2] 27073
04.
05.     [root@node2 ~]# vim /usr/local/kafka/config/server.properties
06.     //node2主机修改
07.     broker.id=23
08.     [root@node3 ~]# vim /usr/local/kafka/config/server.properties
09.     //node3主机修改
10.     broker.id=24
```

5）启动 kafka 集群（node1，node2，node3启动）

```
01.     [root@node1 local]# /usr/local/kafka/bin/kafka-server-start.sh -daemon /usr/local/kaf
02.     [root@node1 local]# jps        //出现kafka
03.     26483 DataNode
04.     27859 Jps
05.     27833 Kafka
06.     26895 QuorumPeerMain
```

6）验证配置，创建一个 topic

```
01.     [root@node1 local]# /usr/local/kafka/bin/kafka-topics.sh --create --partitions 1--replic
02.     Created topic "aa".
```

7) 模拟生产者，发布消息

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

9）模拟消费者，接收消息

```
01.    [ root@node3 ~] # /usr/local/kafka/bin/kafka- console- consumer.sh \
02.    - - bootstrap- server node1:9092 - - topic aa      //这边会直接同步
03.    ccc
04.    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

4.1 问题

本案例要求：

* 配置Hadoop的高可用
* 修改配置文件

4.2 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

Top

图-1

4.3 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）

```
01.      [ root@nn01 ~] # cd /usr/local/hadoop/
02.      [ 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

```
01.      [ root@nn01 hadoop] # /usr/local/zookeeper/bin/zkServer.sh start
02.      [ root@nn01 hadoop] # sh /usr/local/zookeeper/conf/api.sh //利用之前写好的脚本查看
03.      node1   follower
04.      node2   leader
05.      node3   follower
06.      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

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

4）修改vim /etc/hosts

```
01.      [ root@nn01 hadoop] # vim /etc/hosts
02.      192.168.1.21 nn01
03.      192.168.1.25 nn02
04.      192.168.1.22 node1
05.      192.168.1.23 node2
06.      192.168.1.24 node3
```

5）同步到nn02，node1，node2，node3

Top

```
01.    [root@nn01 hadoop]# for i in {22..25}; do rsync -aSH --delete /etc/hosts 192.168.1.$i:/
02.    [1]  14355
03.    [2]  14356
04.    [3]  14357
05.    [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

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

7）所有的主机删除/var/hadoop/*

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

8）配置 core-site

```
01.    [root@nn01 .ssh]# vim /usr/local/hadoop/etc/hadoop/core-site.xml
02.    <configuration>
03.    <property>
04.        <name>fs.defaultFS</name>
05.        <value>hdfs://nsdcluster</value>
06.    //nsdcluster是随便起的名。相当于一个组，访问的时候访问这个组          Top
07.        </property>
08.        <property>
```

```
09.            <name>hadoop.tmp.dir</name>
10.            <value>/var/hadoop</value>
11.        </property>
12.        <property>
13.            <name>ha.zookeeper.quorum</name>
14.            <value>node1:2181,node2:2181,node3:2181</value>   //zookeepe的地址
15.        </property>
16.        <property>
17.            <name>hadoop.proxyuser.nfs.groups</name>
18.            <value>*</value>
19.        </property>
20.        <property>
21.            <name>hadoop.proxyuser.nfs.hosts</name>
22.            <value>*</value>
23.        </property>
24.    </configuration>
```

9）配置 hdfs-site

```
01.    [root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml
02.    <configuration>
03.        <property>
04.            <name>dfs.replication</name>
05.            <value>2</value>
06.        </property>
07.        <property>
08.            <name>dfs.nameservices</name>
09.            <value>nsdcluster</value>
10.        </property>
11.        <property>
12.            <name>dfs.ha.namenodes.nsdcluster</name>
13.    //nn1,nn2名称固定，是内置的变量，nsdcluster里面有nn1，nn2
14.            <value>nn1,nn2</value>
15.        </property>
16.        <property>
17.            <name>dfs.namenode.rpc-address.nsdcluster.nn1</name>
18.    //声明nn1 8020为通讯端口，是nn01的rpc通讯端口
19.            <value>nn01:8020</value>
20.        </property>
21.        <property>
```

Top

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

23.   //声明nn2是谁，nn02的rpc通讯端口

24.   <value>nn02:8020</value>

25.   </property>

26.   <property>

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

28.   //nn01的http通讯端口

29.   <value>nn01:50070</value>

30.   </property>

31.   <property>

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

33.   //nn01和nn02的http通讯端口

34.   <value>nn02:50070</value>

35.   </property>

36.   <property>

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

38.   //指定namenode元数据存储在journalnode中的路径

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

40.   </property>

41.   <property>

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

43.   //指定journalnode日志文件存储的路径

44.   <value>/var/hadoop/journal</value>

45.   </property>

46.   <property>

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

48.   //指定HDFS客户端连接active namenode的java类

49.   <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvi

50.   </property>

51.   <property>

52.   <name>dfs.ha.fencing.methods</name>   //配置隔离机制为ssh

53.   <value>sshfence</value>

54.   </property>

55.   <property>

56.   <name>dfs.ha.fencing.ssh.private- key- files</name>   //指定密钥的位置

57.   <value>/root/.ssh/id_rsa</value>

58.   </property>

59.   <property>

60.   <name>dfs.ha.automatic- failover.enabled</name>   //开启自动故障转移

Top

61.   <value>true</value>

62.   </property>

63.   </configuration>

10）配置yarn-site

```
01.    [root@nn01 ~]# vim /usr/local/hadoop/etc/hadoop/yarn-site.xml
02.    <configuration>
03.
04.    <!-- Site specific YARN configuration properties -->
05.      <property>
06.        <name>yarn.nodemanager.aux-services</name>
07.        <value>mapreduce_shuffle</value>
08.      </property>
09.      <property>
10.        <name>yarn.resourcemanager.ha.enabled</name>
11.        <value>true</value>
12.      </property>
13.      <property>
14.        <name>yarn.resourcemanager.ha.rm-ids</name>        //rm1,rm2代表nn01和nn02
15.        <value>rm1,rm2</value>
16.      </property>
17.      <property>
18.        <name>yarn.resourcemanager.recovery.enabled</name>
19.        <value>true</value>
20.      </property>
21.      <property>
22.        <name>yarn.resourcemanager.store.class</name>
23.        <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore
24.      </property>
25.      <property>
26.        <name>yarn.resourcemanager.zk-address</name>
27.        <value>node1:2181,node2:2181,node3:2181</value>
28.      </property>
29.      <property>
30.        <name>yarn.resourcemanager.cluster-id</name>
31.        <value>yarn-ha</value>
32.      </property>
33.      <property>
34.        <name>yarn.resourcemanager.hostname.rm1</name>
35.        <value>nn01</value>
```

Top

```
36.        </property>
37.        <property>
38.            <name>yarn.resourcemanager.hostname.rm2</name>
39.            <value>nn02</value>
40.        </property>
41.    </configuration>
```

11）同步到nn02，node1，node2，node3

```
01.    [root@nn01 ~]# for i in {22..25}; do rsync -aSH --delete /usr/local/hadoop/ 192.168.1.9
02.    [1] 25411
03.    [2] 25412
04.    [3] 25413
05.    [4] 25414
```

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

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

13）同步配置

```
01.    [root@nn01 ~]# for i in {22..25}; do rsync -aSH --delete /usr/local/hadoop 192.168.1.$i
02.
03.    [1] 28235
04.    [2] 28236
05.    [3] 28237
06.    [4] 28238
```

5 案例4：高可用验证

5.1 问题

本案例要求：

- 初始化集群
- 验证集群

5.2 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

```
01.    [root@nn01 ~]# /usr/local/hadoop/bin/hdfs zkfc -formatZK
02.    …
03.    18/09/11 15:43:35 INFO ha.ActiveStandbyElector: Successfully created /hadoop-ha/nsdcl
04.    …
```

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

```
01.    [root@node1 ~]# /usr/local/hadoop/sbin/hadoop-daemon.sh start journalnode
02.    starting journalnode, logging to /usr/local/hadoop/logs/hadoop-root-journalnode-node1
03.    [root@node1 ~]# jps
04.    29262 JournalNode
05.    26895 QuorumPeerMain
06.    29311 Jps
```

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

```
01.    [root@nn01 ~]# /usr/local/hadoop//bin/hdfs namenode -format
02.    //出现Successfully即为成功
03.    [root@nn01 hadoop]# ls /var/hadoop/
04.    dfs
```

4）nn02数据同步到本地 /var/hadoop/dfs

```
01.    [root@nn02 ~]# cd /var/hadoop/
02.    [root@nn02 hadoop]# ls
03.    [root@nn02 hadoop]# rsync -aSH nn01:/var/hadoop/ /var/hadoop/
04.    [root@nn02 hadoop]# ls
05.    dfs
```

Top

5）初始化 JNS

```
01.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hdfs namenode - initializeSharedEdits
02.    18/09/11 16:26:15 INFO client.QuorumJournalManager: Successfully started new epoch 1
```

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

```
01.    [ root@node1 hadoop] # /usr/local/hadoop/sbin/hadoop- daemon.sh stop journalnode
02.    stopping journalnode
03.    [ root@node1 hadoop] # jps
04.    29346 Jps
05.    26895 QuorumPeerMain
```

步骤二：启动集群

1）nn01上面操作

```
01.    [ root@nn01 hadoop] # /usr/local/hadoop/sbin/start- all.sh  //启动所有集群
02.    This script is Deprecated. Instead use start- dfs.sh and start- yarn.sh
03.    Starting namenodes on [ nn01 nn02]
04.    nn01: starting namenode, logging to /usr/local/hadoop/logs/hadoop- root- namenode- nn
05.    nn02: starting namenode, logging to /usr/local/hadoop/logs/hadoop- root- namenode- nn
06.    node2: starting datanode, logging to /usr/local/hadoop/logs/hadoop- root- datanode- no
07.    node3: starting datanode, logging to /usr/local/hadoop/logs/hadoop- root- datanode- no
08.    node1: starting datanode, logging to /usr/local/hadoop/logs/hadoop- root- datanode- no
09.    Starting journal nodes [ node1 node2 node3]
10.    node1: starting journalnode, logging to /usr/local/hadoop/logs/hadoop- root- journalnode
11.    node3: starting journalnode, logging to /usr/local/hadoop/logs/hadoop- root- journalnode
12.    node2: starting journalnode, logging to /usr/local/hadoop/logs/hadoop- root- journalnode
13.    Starting ZK Failover Controllers on NN hosts [ nn01 nn02]
14.    nn01: starting zkfc, logging to /usr/local/hadoop/logs/hadoop- root- zkfc- nn01.out
15.    nn02: starting zkfc, logging to /usr/local/hadoop/logs/hadoop- root- zkfc- nn02.out
16.    starting yarn daemons
17.    starting resourcemanager, logging to /usr/local/hadoop/logs/yarn- root- resourcemanage
18.    node2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn- root- nodemanage
19.    node1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn- root- nodemanage
20.    node3: starting nodemanager, logging to /usr/local/hadoop/logs/yarn- root- nodemanage
```

Top

2）nn02上面操作

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

◄   ►

3）查看集群状态

```
01.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1
02.    active
03.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn2
04.    standby
05.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1
06.    active
07.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm2
08.    standby
```

4）查看节点是否加入

```
01.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hdfs dfsadmin -report
02.    …
03.    Live datanodes ( 3 ):     //会有三个节点
04.    …
05.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/yarn node -list
06.    Total Nodes: 3
07.         Node-Id        Node-State     Node-Http-Address     Number-of-Running-Containers
08.       node2: 43307        RUNNING          node2: 8042                 0
09.       node1: 34606        RUNNING          node1: 8042                 0
10.       node3: 36749        RUNNING          node3: 8042                 0
```

步骤三：访问集群

1）查看并创建

```
01.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hadoop fs -ls /
02.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hadoop fs -mkdir /aa //创建
03.    [ root@nn01 hadoop] # /usr/local/hadoop/bin/hadoop fs -ls /       //再次查看
04.
```

Top

05.   Found 1 items

06.   drwxr-xr-x   - root supergroup   0 2018-09-11 16:54 /aa

07.

08.   [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -put *.txt /aa

09.   [root@nn01 hadoop]# /usr/local/hadoop/bin/hadoop fs -ls hdfs://nsdcluster/aa

10.   //也可以这样查看

11.   Found 3 items

12.   -rw-r--r--   2 root supergroup 86424 2018-09-11 17:00 hdfs://nsdcluster/aa/LICENSE.txt

13.   -rw-r--r--   2 root supergroup 14978 2018-09-11 17:00 hdfs://nsdcluster/aa/NOTICE.txt

14.   -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

01.   [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1

02.   active

03.   [root@nn01 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh stop namenode

04.   stopping namenode

05.   [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1

06.   //再次查看会报错

07.   [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn2

08.   //nn02由之前的standby变为active

09.   active

10.

11.   [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm1

12.   active

13.   [root@nn01 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh stop resourcemanager

14.   //停止resourcemanager

15.   [root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin -getServiceState rm2

16.   active

3） 恢复节点

01.   [root@nn01 hadoop]# /usr/local/hadoop/sbin/hadoop-daemon.sh start namenode

02.   //启动namenode

03.   [root@nn01 hadoop]# /usr/local/hadoop/sbin/yarn-daemon.sh start resourcemanager

04.   //启动resourcemanager

Top

05.   [root@nn01 hadoop]# /usr/local/hadoop/bin/hdfs haadmin -getServiceState nn1

06.   //查看

07.　　[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin - getServiceState rm1

08.　　//查看

07.　　[root@nn01 hadoop]# /usr/local/hadoop/bin/yarn rmadmin - getServiceState rm1

08.　　//查看