第 **6** 章 **Hadoop HA** 高可用

# 6.1 HA 概述

1. 所谓 HA（High Availablity），即高可用（7\*24 小时不中断服务）。
2. 实现高可用最关键的策略是消除单点故障。HA 严格来说应该分成各个组件的 HA 机制：HDFS 的 HA 和 YARN 的 HA。
3. NameNode 主要在以下两个方面影响 HDFS 集群

* NameNode 机器发生意外，如宕机，集群将无法使用，直到管理员重启
* NameNode 机器需要升级，包括软件、硬件升级，此时集群也将无法使用

HDFS HA 功能通过配置多个 NameNodes(Active/Standby)实现在集群中对 NameNode 的热备来解决上述问题。如果出现故障，如机器崩溃或机器需要升级维护，这时可通过此种方式将 NameNode 很快的切换到另外一台机器。

# 6.2 HDFS-HA 集群搭建

当前 HDFS 集群的规划

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| NameNode |  | Secondarynamenode |
| DataNode | DataNode | DataNode |

HA 的主要目的是消除 namenode 的单点故障,需要将 hdfs 集群规划成以下模样

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| NameNode | NameNode | NameNode |
| DataNode | DataNode | DataNode |

## 6.2.1 HDFS-HA 核心问题

1. 怎么保证三台 **namenode** 的数据一致

a.Fsimage:让一台 nn 生成数据,让其他机器 nn 同步

b.Edits:需要引进新的模块 JournalNode 来保证 edtis 的文件的数据一致性

1. 怎么让同时只有一台 **nn** 是 **active**，其他所有是 **standby** 的

a.手动分配

b.自动分配

1. **2nn** 在 **ha** 架构中并不存在，定期合并 **fsimage** 和 **edtis** 的活谁来干

由 standby 的 nn 来干

1. 如果 **nn** 真的发生了问题，怎么让其他的 **nn** 上位干活

a.手动故障转移

b.自动故障转移

# 6.3 HDFS-HA 手动模式

## 6.3.1 环境准备

1. 修改 IP
2. 修改主机名及主机名和 IP 地址的映射
3. 关闭防火墙
4. ssh 免密登录
5. 安装 JDK，配置环境变量等

## 6.3.2 规划集群

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| NameNode | NameNode | NameNode |
| JournalNode | JournalNode | JournalNode |
| DataNode | DataNode | DataNode |

## 6.3.3 配置 HDFS-HA 集群

1. 官方地址：[**http://hadoop.apache.org/**](http://hadoop.apache.org/)
2. 在 **opt** 目录下创建一个 **ha** 文件夹

[atguigu@hadoop102 ~]$ cd /opt

[atguigu@hadoop102 opt]$ sudo mkdir ha

[atguigu@hadoop102 opt]$ sudo chown atguigu:atguigu /opt/ha

1. 将**/opt/module/**下的 **hadoop-3.1.3** 拷贝到**/opt/ha** 目录下（记得删除 **data** 和 **log** 目录）

[atguigu@hadoop102 opt]$ cp -r /opt/module/hadoop-3.1.3 /opt/ha/

### 4）配置 core-site.xml

<configuration>

<!-- 把多个NameNode的地址组装成一个集群mycluster -->

<property>

<name>fs.defaultFS</name>

<value>hdfs://mycluster</value>

</property>

<!-- 指定hadoop运行时产生文件的存储目录 -->

<property>

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

<value>/opt/ha/hadoop-3.1.3/data</value>

</property> </configuration>

### 5）配置 hdfs-site.xml

<configuration>

<!-- NameNode数据存储目录 -->

<property>

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

<value>file://${hadoop.tmp.dir}/name</value>

</property>

<!-- DataNode数据存储目录 -->

<property>

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

<value>file://${hadoop.tmp.dir}/data</value>

</property>

<!-- JournalNode数据存储目录 -->

<property>

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

<value>${hadoop.tmp.dir}/jn</value>

</property>

<!-- 完全分布式集群名称 -->

<property>

<name>dfs.nameservices</name>

<value>mycluster</value>

</property>

<!-- 集群中NameNode节点都有哪些 -->

<property>

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

<value>nn1,nn2,nn3</value> </property>

<!-- NameNode的RPC通信地址 -->

<property>

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

<value>hadoop102:8020</value>

</property>

<property>

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

<value>hadoop103:8020</value>

</property>

|  |
| --- |
| <property>  <name>dfs.namenode.rpc-address.mycluster.nn3</name>  <value>hadoop104:8020</value>  </property>  <!-- NameNode的http通信地址 -->  <property>  <name>dfs.namenode.http-address.mycluster.nn1</name>  <value>hadoop102:9870</value>  </property>  <property>  <name>dfs.namenode.http-address.mycluster.nn2</name>  <value>hadoop103:9870</value>  </property>  <property>  <name>dfs.namenode.http-address.mycluster.nn3</name>  <value>hadoop104:9870</value>  </property>  <!-- 指定NameNode元数据在JournalNode上的存放位置 -->  <property>  <name>dfs.namenode.shared.edits.dir</name>  <value>qjournal://hadoop102:8485;hadoop103:8485;hadoop104:8485/myclus ter</value>  </property>  <!-- 访问代理类：client用于确定哪个NameNode为Active -->  <property>  <name>dfs.client.failover.proxy.provider.mycluster</name>  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyP rovider</value>  </property>  <!-- 配置隔离机制，即同一时刻只能有一台服务器对外响应 -->  <property>  <name>dfs.ha.fencing.methods</name>  <value>sshfence</value>  </property>  <!-- 使用隔离机制时需要ssh秘钥登录-->  <property>  <name>dfs.ha.fencing.ssh.private-key-files</name>  <value>/home/atguigu/.ssh/id\_rsa</value>  </property>  </configuration> |

**6**）分发配置好的 **hadoop** 环境到其他节点

## 6.3.4 启动 HDFS-HA 集群

**1**）将 **HADOOP\_HOME** 环境变量更改到 **HA** 目录**(**三台机器**)**

[atguigu@hadoop102 ~]$ sudo vim /etc/profile.d/my\_env.sh

将 HADOOP\_HOME 部分改为如下

#HADOOP\_HOME

export HADOOP\_HOME=/opt/ha/hadoop-3.1.3 export PATH=$PATH:$HADOOP\_HOME/bin export PATH=$PATH:$HADOOP\_HOME/sbin

去三台机器上 **source** 环境变量

[atguigu@hadoop102 ~]$source /etc/profile

### 2）在各个 JournalNode 节点上，输入以下命令启动 journalnode 服务

[atguigu@hadoop102 ~]$ hdfs --daemon start journalnode [atguigu@hadoop103 ~]$ hdfs --daemon start journalnode

[atguigu@hadoop104 ~]$ hdfs --daemon start journalnode

**3**）在**[nn1]**上，对其进行格式化，并启动

[atguigu@hadoop102 ~]$ hdfs namenode -format

[atguigu@hadoop102 ~]$ hdfs --daemon start namenode

### 4）在[nn2]和[nn3]上，同步 nn1 的元数据信息

[atguigu@hadoop103 ~]$ hdfs namenode -bootstrapStandby

[atguigu@hadoop104 ~]$ hdfs namenode -bootstrapStandby

### 5）启动[nn2]和[nn3]

[atguigu@hadoop103 ~]$ hdfs --daemon start namenode [atguigu@hadoop104 ~]$ hdfs --daemon start namenode

**6**

）查看

**web**

页面显示



图

hadoop102(

standby

)

图

hadoop103(

standby

)

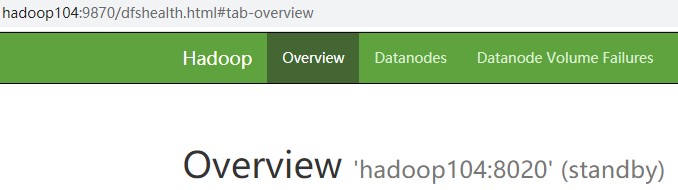


图

hadoop104(

standby

)



1. 在所有节点上，启动 **datanode**

[atguigu@hadoop102 ~]$ hdfs --daemon start datanode [atguigu@hadoop103 ~]$ hdfs --daemon start datanode

[atguigu@hadoop104 ~]$ hdfs --daemon start datanode

1. 将**[nn1]**切换为 **Active**

[atguigu@hadoop102 ~]$ hdfs haadmin -transitionToActive nn1

1. 查看是否 **Active**

[atguigu@hadoop102 ~]$ hdfs haadmin -getServiceState nn1

# 6.4 HDFS-HA 自动模式

## 6.4.1 HDFS-HA 自动故障转移工作机制

自动故障转移为 HDFS 部署增加了两个新组件：ZooKeeper 和 ZKFailoverController （ZKFC）进程，如图所示。ZooKeeper 是维护少量协调数据，通知客户端这些数据的改变和监视客户端故障的高可用服务。

HDFS-HA故障转移机制



Name node

Name node

active

standby

内存中的元数据

内存中的元数据

fsimage

fsimage

edits

edits

edits

edits

edits

zk1

zk2

zk3

Zkfc

Failover

controller

Zkfc

Failover

controller

Zookeeper

服务端

Edits

文件管理系统：

qjournal

5

如果

ssh

补

刀失败则调用

用户自定义脚

本程序

6

获取命令运

行结果

/home/atguigu/kill/

poweroff.sh

1

假死

2

检测到假死

3

通知另一台

NameNode

的

zkfc

4

强行杀死

namenode

,

防止脑裂

ssh

kill -9

namenode

进程号

同时出现两个

Active

状态

namenode

的术语叫脑裂

brain

split

。

防止脑裂的两种方式：

1

）

ssh

发送

kill

指令

2

）调用用户自定义脚本程序

7

激活本台

namenode

，切换为

Active

Zookeeper

客户端

Zookeeper

客户端

写

读

## 6.4.2 HDFS-HA 自动故障转移的集群规划

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| NameNode | NameNode | NameNode |
| JournalNode | JournalNode | JournalNode |
| DataNode | DataNode | DataNode |
| Zookeeper | Zookeeper | Zookeeper |
| ZKFC | ZKFC | ZKFC |

## 6.4.3 配置 HDFS-HA 自动故障转移

1. 具体配置
   1. 在 hdfs-site.xml 中增加

<!-- 启用nn故障自动转移 -->

<property>

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

<value>true</value>

</property>

* 1. 在 core-site.xml 文件中增加

<!-- 指定zkfc要连接的zkServer地址 -->

<property>

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

<value>hadoop102:2181,hadoop103:2181,hadoop104:2181</value>

</property>

* 1. 修改后分发配置文件

[atguigu@hadoop102 etc]$ pwd

/opt/ha/hadoop-3.1.3/etc

[atguigu@hadoop102 etc]$ xsync hadoop/

1. 启动
   1. 关闭所有 HDFS 服务：

[atguigu@hadoop102 ~]$ stop-dfs.sh

* 1. 启动 Zookeeper 集群：

[atguigu@hadoop102 ~]$ zkServer.sh start [atguigu@hadoop103 ~]$ zkServer.sh start

[atguigu@hadoop104 ~]$ zkServer.sh start

* 1. 启动 Zookeeper 以后，然后再初始化 HA 在 Zookeeper 中状态：

[atguigu@hadoop102 ~]$ hdfs zkfc -formatZK

* 1. 启动 HDFS 服务：

[atguigu@hadoop102 ~]$ start-dfs.sh

* 1. 可以去 zkCli.sh 客户端查看 Namenode 选举锁节点内容：

|  |
| --- |
| [zk: localhost:2181(CONNECTED) 7] get -s  /hadoop-ha/mycluster/ActiveStandbyElectorLock  myclusternn2 hadoop103 >(>  cZxid = 0x10000000b  ctime = Tue Jul 14 17:00:13 CST 2020 mZxid = 0x10000000b  mtime = Tue Jul 14 17:00:13 CST 2020 pZxid = 0x10000000b cversion = 0 dataVersion = 0 aclVersion = 0  ephemeralOwner = 0x40000da2eb70000 dataLength = 33 numChildren = 0 |

1. 验证
   1. 将 Active NameNode 进程 kill，查看网页端三台 Namenode 的状态变化

[atguigu@hadoop102 ~]$ kill -9 namenode的进程id

### 6.4.3 解决 NN 连接不上 JN 的问题

自动故障转移配置好以后，然后使用 start-dfs.sh 群起脚本启动 hdfs 集群，有可能会遇到 NameNode 起来一会后，进程自动关闭的问题。查看 NameNode 日志，报错信息如下：

2020-08-17 10:11:40,658 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)

2020-08-17 10:11:40,659 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)

2020-08-17 10:11:40,659 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 0 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)

2020-08-17 10:11:41,660 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 1 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)

2020-08-17 10:11:41,660 INFO org.apache.hadoop.ipc.Client: Retrying connect

|  |
| --- |
| to server: hadoop102/192.168.6.102:8485. Already tried 1 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:41,665 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 1 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:42,661 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 2 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:42,661 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 2 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:42,667 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 2 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:43,662 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 3 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:43,662 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 3 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:43,668 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 3 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:44,663 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 4 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:44,663 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 4 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:44,670 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 4 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS) 2020-08-17 10:11:45,467 INFO  org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager: Waited 6001 |

|  |
| --- |
| ms (timeout=20000 ms) for a response for selectStreamingInputStreams. No responses yet.  2020-08-17 10:11:45,664 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 5 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:45,664 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 5 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:45,672 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 5 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS) 2020-08-17 10:11:46,469 INFO  org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager: Waited 7003 ms (timeout=20000 ms) for a response for selectStreamingInputStreams. No responses yet.  2020-08-17 10:11:46,665 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 6 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:46,665 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 6 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:46,673 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 6 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS) 2020-08-17 10:11:47,470 INFO  org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager: Waited 8004 ms (timeout=20000 ms) for a response for selectStreamingInputStreams. No responses yet.  2020-08-17 10:11:47,666 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 7 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:47,667 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 7 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:47,674 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 7 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, |

|  |
| --- |
| sleepTime=1000 MILLISECONDS) 2020-08-17 10:11:48,471 INFO  org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager: Waited 9005 ms (timeout=20000 ms) for a response for selectStreamingInputStreams. No responses yet.  2020-08-17 10:11:48,668 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 8 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:48,668 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 8 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:48,675 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 8 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:49,669 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop102/192.168.6.102:8485. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:49,673 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop104/192.168.6.104:8485. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)  2020-08-17 10:11:49,676 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop103/192.168.6.103:8485. Already tried 9 time(s); retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS) 2020-08-17 10:11:49,678 WARN  org.apache.hadoop.hdfs.server.namenode.FSEditLog: Unable to determine input streams from QJM to [192.168.6.102:8485, 192.168.6.103:8485, 192.168.6.104:8485]. Skipping. org.apache.hadoop.hdfs.qjournal.client.QuorumException: Got too many exceptions to achieve quorum size 2/3. 3 exceptions thrown:  192.168.6.103:8485: Call From hadoop102/192.168.6.102 to hadoop103:8485 failed on connection exception: java.net.ConnectException: 拒绝连接; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused  192.168.6.102:8485: Call From hadoop102/192.168.6.102 to hadoop102:8485 failed on connection exception: java.net.ConnectException: 拒绝连接; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused  192.168.6.104:8485: Call From hadoop102/192.168.6.102 to hadoop104:8485 failed on connection exception: java.net.ConnectException: 拒绝连接; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused |

查看报错日志，可分析出报错原因是因为 NameNode 连接不上 JournalNode，而利用 jps 命令查看到三台 JN 都已经正常启动，为什么 NN 还是无法正常连接到 JN 呢？这是因为 start-dfs.sh 群起脚本默认的启动顺序是先启动 NN，再启动 DN，然后再启动 JN，并且默认的 rpc 连接参数是重试次数为 10，每次重试的间隔是 1s，也就是说启动完 NN 以后的 10s 中内，JN 还启动不起来，NN 就会报错了。

core-default.xml 里面有两个参数如下：

<!-- NN连接JN重试次数，默认是10次 -->

<property>

<name>ipc.client.connect.max.retries</name>

<value>10</value>

</property>

<!-- 重试时间间隔，默认1s -->

<property>

<name>ipc.client.connect.retry.interval</name>

<value>1000</value>

</property>

解决方案：遇到上述问题后，可以稍等片刻，等 JN 成功启动后，手动启动下三台

NN：

[atguigu@hadoop102 ~]$ hdfs --daemon start namenode

[atguigu@hadoop103 ~]$ hdfs --daemon start namenode

[atguigu@hadoop104 ~]$ hdfs --daemon start namenode

也可以在 core-site.xml 里面适当调大上面的两个参数：

<!-- NN连接JN重试次数，默认是10次 -->

<property>

<name>ipc.client.connect.max.retries</name>

<value>20</value>

</property>

<!-- 重试时间间隔，默认1s -->

<property>

<name>ipc.client.connect.retry.interval</name>

<value>5000</value>

</property>

# 6.5 YARN-HA 配置

## 6.5.1 YARN-HA 工作机制

**1**）官方文档：

[http://hadoop.apache.org/docs/r3.1.3/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.htm](http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html)

[l](http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html)

### 2）YARN-HA 工作机制

**6**

**.**

**5**

**.2**

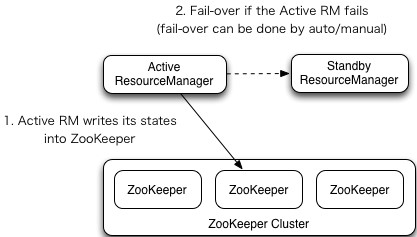
配置

**YARN**

**-**

**HA**

集群



1. 环境准备
   1. 修改 IP
   2. 修改主机名及主机名和 IP 地址的映射
   3. 关闭防火墙
   4. ssh 免密登录
   5. 安装 JDK，配置环境变量等
   6. 配置 Zookeeper 集群
2. 规划集群

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| ResourceManager | ResourceManager | ResourceManager |
| NodeManager | NodeManager | NodeManager |
| Zookeeper | Zookeeper | Zookeeper |

1. 核心问题

**a .**如果当前 **active rm** 挂了，其他 **rm** 怎么将其他 **standby rm** 上位核心原理跟 hdfs 一样，利用了 zk 的临时节点

**b.** 当前 **rm** 上有很多的计算程序在等待运行**,**其他的 **rm** 怎么将这些程序接手过来接着跑

rm 会将当前的所有计算程序的状态存储在 zk 中,其他 rm 上位后会去读取，然后接着跑

**4**）具体配置

1. yarn-site.xml

|  |
| --- |
| <configuration>    <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>    <!-- 启用resourcemanager ha -->  <property>  <name>yarn.resourcemanager.ha.enabled</name>  <value>true</value>  </property>    <!-- 声明两台resourcemanager的地址 -->  <property>  <name>yarn.resourcemanager.cluster-id</name>  <value>cluster-yarn1</value>  </property>  <!--指定resourcemanager的逻辑列表-->  <property>  <name>yarn.resourcemanager.ha.rm-ids</name> <value>rm1,rm2,rm3</value>  </property>  <!-- ========== rm1的配置 ========== -->  <!-- 指定rm1的主机名 -->  <property>  <name>yarn.resourcemanager.hostname.rm1</name> <value>hadoop102</value>  </property>  <!-- 指定rm1的web端地址 -->  <property>  <name>yarn.resourcemanager.webapp.address.rm1</name>  <value>hadoop102:8088</value>  </property>  <!-- 指定rm1的内部通信地址 -->  <property>  <name>yarn.resourcemanager.address.rm1</name>  <value>hadoop102:8032</value>  </property>  <!-- 指定AM向rm1申请资源的地址 -->  <property> |

|  |
| --- |
| <name>yarn.resourcemanager.scheduler.address.rm1</name>  <value>hadoop102:8030</value>  </property>  <!-- 指定供NM连接的地址 -->  <property>  <name>yarn.resourcemanager.resource-tracker.address.rm1</name>  <value>hadoop102:8031</value>  </property>  <!-- ========== rm2的配置 ========== -->  <!-- 指定rm2的主机名 -->  <property>  <name>yarn.resourcemanager.hostname.rm2</name> <value>hadoop103</value>  </property>  <property>  <name>yarn.resourcemanager.webapp.address.rm2</name>  <value>hadoop103:8088</value>  </property>  <property>  <name>yarn.resourcemanager.address.rm2</name>  <value>hadoop103:8032</value>  </property>  <property>  <name>yarn.resourcemanager.scheduler.address.rm2</name>  <value>hadoop103:8030</value>  </property>  <property>  <name>yarn.resourcemanager.resource-tracker.address.rm2</name>  <value>hadoop103:8031</value>  </property>  <!-- ========== rm3的配置 ========== -->  <!-- 指定rm1的主机名 -->  <property>  <name>yarn.resourcemanager.hostname.rm3</name> <value>hadoop104</value>  </property>  <!-- 指定rm1的web端地址 -->  <property>  <name>yarn.resourcemanager.webapp.address.rm3</name>  <value>hadoop104:8088</value>  </property>  <!-- 指定rm1的内部通信地址 -->  <property>  <name>yarn.resourcemanager.address.rm3</name> |
| <value>hadoop104:8032</value>  </property>  <!-- 指定AM向rm1申请资源的地址 -->  <property>  <name>yarn.resourcemanager.scheduler.address.rm3</name>  <value>hadoop104:8030</value>  </property>  <!-- 指定供NM连接的地址 -->  <property>  <name>yarn.resourcemanager.resource-tracker.address.rm3</name>  <value>hadoop104:8031</value>  </property>  <!-- 指定zookeeper集群的地址 -->  <property>  <name>yarn.resourcemanager.zk-address</name>  <value>hadoop102:2181,hadoop103:2181,hadoop104:2181</value>  </property>    <!-- 启用自动恢复 -->  <property>  <name>yarn.resourcemanager.recovery.enabled</name>  <value>true</value>  </property>    <!-- 指定resourcemanager的状态信息存储在zookeeper集群 -->  <property>  <name>yarn.resourcemanager.store.class</name>  <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateSt ore</value> </property>  <!-- 环境变量的继承 -->  <property>  <name>yarn.nodemanager.env-whitelist</name>  <value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLAS  SPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>  </property>    </configuration> |

1. 同步更新其他节点的配置信息，分发配置文件

[atguigu@hadoop102 etc]$ xsync hadoop/

### 4）启动 YARN

1. 在 hadoop102 或者 hadoop103 中执行：

[atguigu@hadoop102 ~]$ start-yarn.sh

1. 查看服务状态

[atguigu@hadoop102 ~]$ yarn rmadmin -getServiceState rm1

1. 可以去 zkCli.sh 客户端查看 ResourceManager 选举锁节点内容：

|  |
| --- |
| [atguigu@hadoop102 ~]$ zkCli.sh  [zk: localhost:2181(CONNECTED) 16] get -s  /yarn-leader-election/cluster-yarn1/ActiveStandbyElectorLock    cluster-yarn1rm1 cZxid = 0x100000022  ctime = Tue Jul 14 17:06:44 CST 2020 mZxid = 0x100000022  mtime = Tue Jul 14 17:06:44 CST 2020 pZxid = 0x100000022 cversion = 0 dataVersion = 0 aclVersion = 0  ephemeralOwner = 0x30000da33080005 dataLength = 20 numChildren = 0 |

1. web 端查看 hadoop102:8088 和 hadoop103:8088 的 YARN 的状态

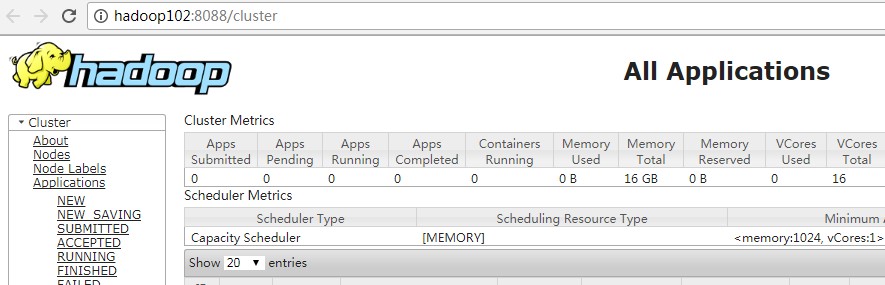
**6**

**.**

**6**

**HADOOP HA**

的最终规划



将整个 ha 搭建完成后,集群将形成以下模样

|  |  |  |
| --- | --- | --- |
| hadoop102 | hadoop103 | hadoop104 |
| NameNode | NameNode | NameNode |
| JournalNode | JournalNode | JournalNode |
| DataNode | DataNode | DataNode |
| Zookeeper | Zookeeper | Zookeeper |
| ZKFC | ZKFC | ZKFC |
| ResourceManager | ResourceManager | ResourceManager |
| NodeManager | NodeManager | NodeManager |