# 第8章 HA高可用

## 8.1 HA概述

1）所谓HA（High Availability），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）Hadoop2.0之前，在HDFS集群中NameNode存在单点故障（SPOF）。

4）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Active/Standby两个NameNode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。

## 8.2 工作机制

配置两个NameNode，通过双NameNode消除单点故障。

## 8.3 HDFS-HA手动故障转移

### 8.3.1 工作要点

1）元数据管理方式需要改变：

内存中各自保存一份元数据；

Edits日志只有Active状态的NameNode节点可以做写操作；

两个NameNode都可以读取Edits；

共享的Edits放在一个共享存储中管理（qjournal和NFS两个主流实现）；

2）必须保证两个NameNode之间能够ssh无密码登录。

3）隔离（Fence），即同一时刻仅仅有一个NameNode对外提供服务

4）自动故障转移需要一个状态管理功能模块

实现了一个ZKfailover，常驻在每一个NameNode所在的节点，每一个ZKfailover负责监控自己所在NameNode节点，利用zk进行状态标识，当需要进行状态切换时，由ZKfailover来负责切换，切换时需要防止brain split现象的发生。

### 8.3.2 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

### 8.3.3 规划集群

hadoop102 hadoop103 hadoop104

NameNode NameNode

JournalNode JournalNode JournalNode

DataNode DataNode DataNode

ResourceManager

NodeManager NodeManager NodeManager

### 8.3.4 配置HDFS-HA集群

1）官方地址：<http://hadoop.apache.org/>

2）在/opt/module/目录下创建一个HA文件夹

[atguigu@hadoop102 module]$ mkdir HA

3）将/opt/module/下的 hadoop-2.7.2拷贝到/opt/module/HA目录下

[atguigu@hadoop102 module]$ cp -r hadoop-2.7.2/ /opt/module/HA/

4）配置core-site.xml

<configuration>

<!-- 把两个NameNode）的地址组装成一个集群mycluster -->

<property>

<name>fs.defaultFS</name>

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

</property>

<!-- 声明journalnode服务本地文件系统存储目录-->

<property>

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

<value>/opt/module/HA/hadoop-2.7.2/data/jn</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/module/HA/hadoop-2.7.2/data/tmp</value>

</property>

</configuration>

5）配置hdfs-site.xml

<configuration>

<!-- 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</value>

</property>

<!-- nn1的RPC通信地址 -->

<property>

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

<value>hadoop102:8020</value>

</property>

<!-- nn2的RPC通信地址 -->

<property>

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

<value>hadoop103:8020</value>

</property>

<!-- nn1的http通信地址 -->

<property>

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

<value>hadoop102:50070</value>

</property>

<!-- nn2的http通信地址 -->

<property>

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

<value>hadoop103:50070</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/mycluster</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无秘钥登录-->

<property>

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

<value>/home/atguigu/.ssh/id\_rsa</value>

</property>

<!-- 关闭权限检查-->

<property>

<name>dfs.permissions.enable</name>

<value>false</value>

</property>

<!-- 访问代理类：client，mycluster，active配置失败自动切换实现方式-->

<property>

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

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

</property>

</configuration>

6）拷贝配置好的hadoop环境到其他节点

### 8.3.5 启动HDFS-HA集群

1）在所有JournalNode节点上，输入以下命令启动JournalNode服务

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start journalnode

2）在[nn1]上，对其进行格式化，并启动

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hdfs namenode -format

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start namenode

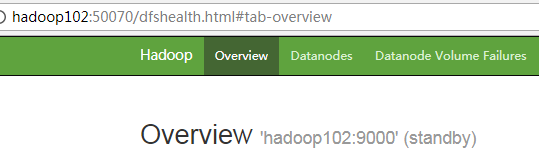
3）在[nn2]上，同步nn1的元数据信息

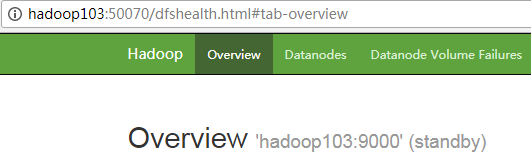
[atguigu@hadoop103 hadoop-2.7.2]$ bin/hdfs namenode -bootstrapStandby

4）启动[nn2]

[atguigu@hadoop103 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start namenode

5）查看web页面显示





6）在[nn1]上，启动所有DataNode

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemons.sh start datanode

7）将[nn1]切换为Active

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hdfs haadmin -transitionToActive nn1

8）查看是否Active

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hdfs haadmin -getServiceState nn1

## 8.4 HDFS-HA自动故障转移

### 8.4.1工作要点

前面学习了使用命令hdfs haadmin -failover手动进行故障转移，在该模式下，即使现役NameNode已经失效，系统也不会自动从现役NameNode转移到待机NameNode。

下面学习如何配置部署HA自动进行故障转移，自动故障转移为HDFS部署增加了两个新组件：ZooKeeper和ZKFailoverController（ZKFC）进程。

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。HA的自动故障转移依赖于ZooKeeper的以下功能：

**1）故障检测：**集群中的每个NameNode在ZooKeeper中维护了一个持久会话，如果机器崩溃，ZooKeeper中的会话将终止，ZooKeeper通知另一个NameNode需要触发故障转移。

**2）现役NameNode选择：**ZooKeeper提供了一个简单的机制用于唯一的选择一个节点为active状态。如果目前现役NameNode崩溃，另一个节点可能从ZooKeeper获得特殊的排外锁以表明它应该成为现役NameNode。

ZKFC是自动故障转移中的另一个新组件，是ZooKeeper的客户端，也监视和管理NameNode的状态。每个运行NameNode的主机也运行了一个ZKFC进程，ZKFC负责：

**1） 健康监测：**ZKFC使用一个健康检查命令定期地ping与之在相同主机的NameNode，只要该NameNode及时地回复健康状态，ZKFC认为该节点是健康的。如果该节点崩溃，冻结或进入不健康状态，健康监测器标识该节点为非健康的。

**2）ZooKeeper会话管理：**当本地NameNode是健康的，ZKFC保持一个在ZooKeeper中打开的会话。如果本地NameNode处于active状态，ZKFC也保持一个特殊的znode锁，该锁使用了ZooKeeper对短暂节点的支持，如果会话终止，锁节点将自动删除。

**3）基于ZooKeeper的选择：**如果本地NameNode是健康的，且ZKFC发现没有其它的节点当前持有znode锁，它将为自己获取该锁。如果成功，则它已经赢得了选择，并负责运行故障转移进程以使它的本地NameNode为active。故障转移进程与前面描述的手动故障转移相似，首先如果必要保护之前的现役NameNode，然后本地NameNode转换为Active状态。



### 8.4.2 规划集群

hadoop102 hadoop103 hadoop104

NameNode NameNode

JournalNode JournalNode JournalNode

DataNode DataNode DataNode

ZK ZK ZK

ResourceManager

NodeManager NodeManager NodeManager

### 8.4.3 配置Zookeeper集群

0）集群规划

在hadoop102、hadoop103和hadoop104三个节点上部署Zookeeper。

1）解压安装

（1）解压zookeeper安装包到/opt/module/目录下

[atguigu@hadoop102 software]$ tar -zxvf zookeeper-3.4.10.tar.gz -C /opt/module/

（2）在/opt/module/zookeeper-3.4.10/这个目录下创建zkData

[atguigu@hadoop102 zookeeper-3.4.10]$ mkdir -p zkData

（3）重命名/opt/module/zookeeper-3.4.10/conf这个目录下的zoo\_sample.cfg为zoo.cfg

[atguigu@hadoop102 conf]$ mv zoo\_sample.cfg zoo.cfg

2）配置zoo.cfg文件

（1）具体配置

dataDir=/opt/module/zookeeper-3.4.10/zkData

增加如下配置

#######################cluster##########################

server.2=hadoop102:2888:3888

server.3=hadoop103:2888:3888

server.4=hadoop104:2888:3888

（2）配置参数解读

Server.A=B:C:D。

A是一个数字，表示这个是第几号服务器；

B是这个服务器的ip地址；

C是这个服务器与集群中的Leader服务器交换信息的端口；

D是万一集群中的Leader服务器挂了，需要一个端口来重新进行选举，选出一个新的Leader，而这个端口就是用来执行选举时服务器相互通信的端口。

集群模式下配置一个文件myid，这个文件在dataDir目录下，这个文件里面有一个数据就是A的值，Zookeeper启动时读取此文件，拿到里面的数据与zoo.cfg里面的配置信息比较从而判断到底是哪个server。

3）集群操作

（1）在/opt/module/zookeeper-3.4.10/zkData目录下创建一个myid的文件

[atguigu@hadoop102 zkData]$ touch myid

添加myid文件，注意一定要在linux里面创建，在notepad++里面很可能乱码

（2）编辑myid文件

[atguigu@hadoop102 zkData]$ vi myid

在文件中添加与server对应的编号：如2

（3）拷贝配置好的zookeeper到其他机器上

[atguigu@hadoop102 zkData]$ scp -r zookeeper-3.4.10/ [hadoop103:/opt/module/](mailto:root@hadoop103.atguigu.com:/opt/app/)

[atguigu@hadoop102 zkData]$ scp -r zookeeper-3.4.10/ [hadoop104:/opt/module/](mailto:root@hadoop103.atguigu.com:/opt/app/)

并分别修改myid文件中内容为3、4

（4）分别启动zookeeper

[atguigu@hadoop102 zookeeper-3.4.10]$ bin/zkServer.sh start

[atguigu@hadoop103 zookeeper-3.4.10]$ bin/zkServer.sh start

[atguigu@hadoop104 zookeeper-3.4.10]$ bin/zkServer.sh start

（5）查看状态

[atguigu@hadoop102 zookeeper-3.4.10]$ bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: follower

[atguigu@hadoop102 zookeeper-3.4.10]$ bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: leader

[atguigu@hadoop102 zookeeper-3.4.10]$ bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: follower

### 8.4.4 配置HDFS-HA自动故障转移

1）具体配置

（1）在hdfs-site.xml中增加

<property>

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

<value>true</value>

</property>

（2）在core-site.xml文件中增加

<property>

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

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

</property>

2）启动（在手动故障转移的基础上）

（1）关闭所有HDFS服务：

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/stop-dfs.sh

（2）启动Zookeeper集群：

[atguigu@hadoop102 hadoop-2.7.2]$ bin/zkServer.sh start

（3）初始化HA在Zookeeper中状态：

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hdfs zkfc -formatZK

（4）启动HDFS服务（先启动JN，再启动NN以及DN）：

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemons.sh start journalnode

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start namenode

[atguigu@hadoop103 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start namenode

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemons.sh start datanode

（5）在各个NameNode节点上启动DFSZK Failover Controller，先在哪台机器启动，哪个机器的NameNode就是Active NameNode

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start zkfc

3）验证:将Active NameNode进程kill

[atguigu@hadoop102 hadoop-2.7.2]$ kill -9 namenode的进程id

### 8.4.5 直接进行自动故障转移

仅仅在第一次启动的时候做如下操作即可

1）在各个JournalNode节点上，输入以下命令启动JournalNode服务

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start journalnode

2）在[nn1]上，对其进行格式化，并启动

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hdfs namenode -format

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start namenode

3）在[nn2]上，同步nn1的元数据信息

[atguigu@hadoop103 hadoop-2.7.2]$ bin/hdfs namenode -bootstrapStandby

4）启动[nn2]

[atguigu@hadoop103 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start namenode

5）启动Zookeeper集群：

[atguigu@hadoop102 hadoop-2.7.2]$ bin/zkServer.sh start

6）初始化HA在Zookeeper中状态：

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hdfs zkfc -formatZK

7）在各个NameNode节点上启动DFSZK Failover Controller，先在哪台机器启动，哪个机器的NameNode就是Active NameNode

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemon.sh start zkfc

8）启动DataNode

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/hadoop-daemons.sh start datanode

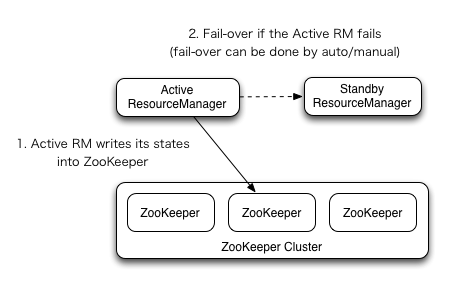
## 8.5 YARN-HA配置

### 8.5.1 YARN-HA工作机制

1）官方文档：

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html>

2）YARN-HA工作机制



### 8.5.2 配置YARN-HA集群

0）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

1）规划集群

hadoop102 hadoop103 hadoop104

NameNode NameNode

JournalNode JournalNode JournalNode

DataNode DataNode DataNode

ZK ZK ZK

ResourceManager ResourceManager

NodeManager NodeManager NodeManager

2）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>

<property>

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

<value>rm1,rm2</value>

</property>

<property>

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

<value>hadoop103</value>

</property>

<property>

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

<value>hadoop104</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.ZKRMStateStore</value>

</property>

</configuration>

（2）同步更新其他节点的配置信息

3）启动Yarn

（1）在hadoop103中执行：

[atguigu@hadoop103 hadoop-2.7.2]$ sbin/start-yarn.sh

（2）在hadoop104中执行：

[atguigu@hadoop104 hadoop-2.7.2]$ sbin/yarn-daemon.sh start resourcemanager

（3）查看服务状态

[atguigu@hadoop103 hadoop-2.7.2]$ bin/yarn rmadmin -getServiceState rm1

## 8.6 HDFS Federation架构设计

1. NameNode架构的局限性

（1）Namespace（命名空间）的限制

由于NameNode在内存中存储所有的元数据（metadata），因此单个namenode所能存储的对象（文件+块）数目受到namenode所在JVM的heap size的限制。50G的heap能够存储20亿（200million）个对象，这20亿个对象支持4000个datanode，12PB的存储（假设文件平均大小为40MB）。随着数据的飞速增长，存储的需求也随之增长。单个datanode从4T增长到36T，集群的尺寸增长到8000个datanode。存储的需求从12PB增长到大于100PB。

（2）隔离问题

由于HDFS仅有一个NameNode，无法隔离各个程序，因此HDFS上的一个实验程序就很有可能影响整个HDFS上运行的程序。

（3）性能的瓶颈

由于是单个NameNode的HDFS架构，因此整个HDFS文件系统的吞吐量受限于单个namenode的吞吐量。

2）HDFS Federation架构设计

能不能有多个NameNode

NameNode NameNode NameNode

元数据 元数据 元数据

Log machine 电商数据/话单数据



3）HDFS Federation应用思考

不同应用可以使用不同NameNode进行数据管理

图片业务、爬虫业务、日志审计业务

Hadoop生态系统中，不同的框架使用不同的namenode进行管理namespace。（隔离性）