尚硅谷大数据技术之Hadoop（HA）

（作者：尚硅谷大数据研发部）

版本：V3.0

# 第4章 Hadoop HA高可用

## 4.1 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）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NameNodes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。

## 4.2 HDFS-HA工作机制

通过多个NameNode消除单点故障

### 4.2.1 HDFS-HA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；

所有的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）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。

3）必须保证两个NameNode之间能够ssh无密码登录

4）隔离（Fence），即同一时刻仅仅有一个NameNode对外提供服务

### 4.2.2 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3-20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。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。



## 4.3 HDFS-HA集群配置

### 4.3.1 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

### 4.3.2 规划集群

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

### 4.3.3 配置Zookeeper集群

1）集群规划

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

2）解压安装

（1）解压Zookeeper安装包到/opt/module/目录下

[atguigu@hadoop102 software]$ tar -zxvf zookeeper-3.4.14.tar.gz -C /opt/module/

（2）在/opt/module/zookeeper-3.4.14/这个目录下创建zkData

mkdir -p zkData

（3）重命名/opt/module/zookeeper-3.4.14/conf这个目录下的zoo\_sample.cfg为zoo.cfg

mv zoo\_sample.cfg zoo.cfg

3）配置zoo.cfg文件

（1）具体配置

dataDir=/opt/module/zookeeper-3.4.14/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。

4）集群操作

（1）在/opt/module/zookeeper-3.4.14/zkData目录下创建一个myid的文件

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

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

scp -r zookeeper-3.4.14/ [root@hadoop103.atguigu.com:/opt/app/](mailto:root@hadoop103.atguigu.com:/opt/app/)

scp -r zookeeper-3.4.14/ [root@hadoop104.atguigu.com:/opt/app/](mailto:root@hadoop104.atguigu.com:/opt/app/)

并分别修改myid文件中内容为3、4

（4）分别启动zookeeper

[root@hadoop102 zookeeper-3.4.14]# bin/zkServer.sh start

[root@hadoop103 zookeeper-3.4.14]# bin/zkServer.sh start

[root@hadoop104 zookeeper-3.4.14]# bin/zkServer.sh start

（5）查看状态

[root@hadoop102 zookeeper-3.4.14]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.14/bin/../conf/zoo.cfg

Mode: follower

[root@hadoop103 zookeeper-3.4.14]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.14/bin/../conf/zoo.cfg

Mode: leader

[root@hadoop104 zookeeper-3.4.5]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.14/bin/../conf/zoo.cfg

Mode: follower

### 4.3.4 配置HDFS-HA集群

1）官方地址：<http://hadoop.apache.org/>

2）在opt目录下创建一个ha文件夹

sudo mkdir ha

sudo chown atguigu:atguigu /opt/ha

3）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）

cp -r /opt/module/hadoop-3.1.3 /opt/ha/

4）配置hadoop-env.sh

export JAVA\_HOME=/opt/module/jdk1.8.0\_212

5）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>

6）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/mycluster</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.ConfiguredFailoverProxyProvider</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>

7）分发配置好的hadoop环境到其他节点

### 4.3.5 启动HDFS-HA集群

1）将HADOOP\_HOME环境变量更改到HA目录

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

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

hdfs --daemon start journalnode

3）在[nn1]上，对其进行格式化，并启动

hdfs namenode -format

hdfs --daemon start namenode

4）在[nn2]和[nn3]上，同步nn1的元数据信息

hdfs namenode -bootstrapStandby

5）启动[nn2]和[nn3]

hdfs --daemon start namenode

6）查看web页面显示

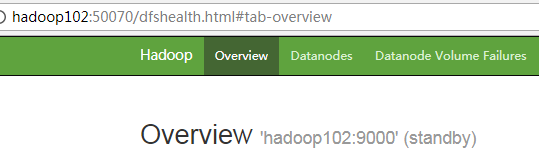


图 hadoop102(standby)

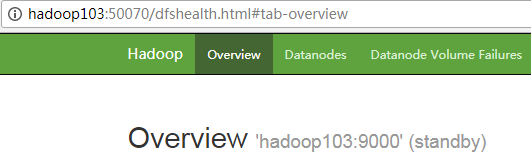


图 hadoop103(standby)

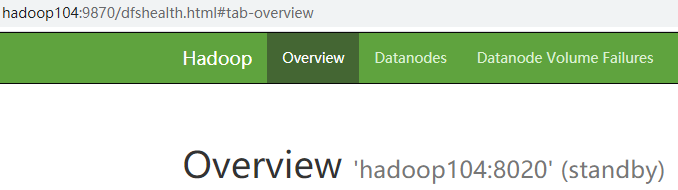


图 hadoop104(standby)

7）在所有节点上，启动datanode

hdfs --daemon start datanode

8）将[nn1]切换为Active

hdfs haadmin -transitionToActive nn1

9）查看是否Active

hdfs haadmin -getServiceState nn1

### 4.3.6 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>

（2）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>

2）启动

（1）关闭所有HDFS服务：

stop-dfs.sh

（2）启动Zookeeper集群：

zkServer.sh start

（3）初始化HA在Zookeeper中状态：

hdfs zkfc -formatZK

（4）启动HDFS服务：

start-dfs.sh

3）验证

（1）将Active NameNode进程kill

kill -9 namenode的进程id

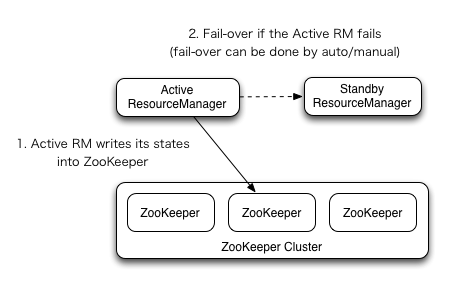
## 4.4 YARN-HA配置

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

2）YARN-HA工作机制



### 4.4.2 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 |
| NameNode | NameNode | NameNode |
| JournalNode | JournalNode | JournalNode |
| DataNode | DataNode | DataNode |
| ZK | ZK | ZK |
| ResourceManager | ResourceManager |  |
| NodeManager | NodeManager | NodeManager |

3）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</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>

<!--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>

<!-- 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,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_MAPRED\_HOME</value>

</property>

</configuration>

（2）同步更新其他节点的配置信息

4）启动hdfs

start-dfs.sh

5）启动YARN

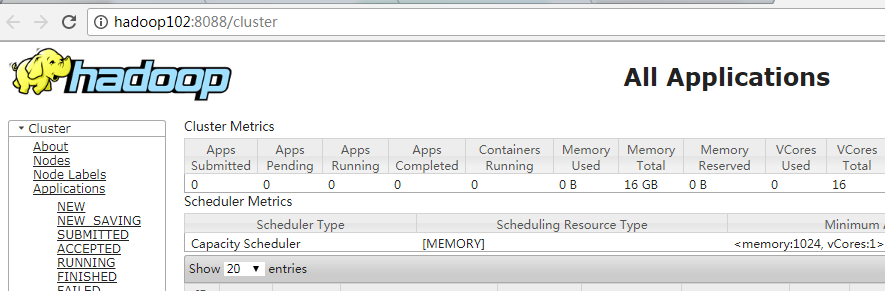
（1）在hadoop102或者hadoop103中执行：

start-yarn.sh

（2）查看服务状态

yarn rmadmin -getServiceState rm1

（3）web端查看hadoop102:8088和hadoop103:8088的YARN的状态



## 4.5 HDFS Federation架构设计

### 4.5.1 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。

### 4.5.2 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 |



图 HDFS Federation架构设计

### 4.5.3 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）

