# HADOOP高可用机制

|  |  |
| --- | --- |
| HA运作机制 | 什么是HA |
| HADOOP如何实现HA |
| HDFS-HA详解 |
| HA集群搭建 |

目标：

掌握分布式系统中HA机制的思想

掌握HADOOP内置HA的运作机制

掌握HADOOP2.x的HA集群机制配置

# Hadoop的HDFS的HA机制

前言：正式引入HA机制是从hadoop2.0开始，之前的版本中没有HA机制

## 1.1 HA的运作机制

（1）hadoop-HA集群运作机制介绍

所谓HA，即高可用（7\*24小时不中断服务）

实现高可用最关键的是消除单点故障

hadoop-ha严格来说应该分成各个组件的HA机制——HDFS的HA、YARN的HA

（2）HDFS的HA机制详解

通过双namenode消除单点故障

双namenode协调工作的要点：

A、元数据管理方式需要改变：

内存中各自保存一份元数据

Edits日志只能有一份，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）

B、需要一个状态管理功能模块

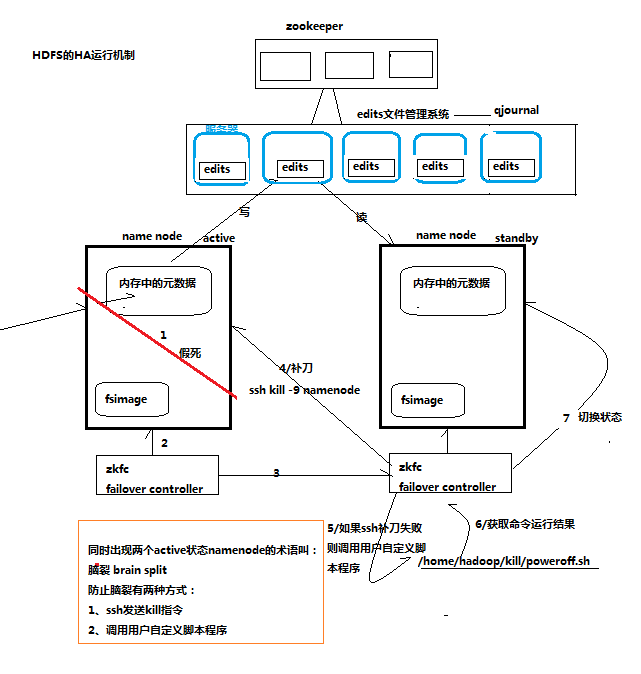
实现了一个ZKFC，常驻在每一个namenode所在的节点

每一个ZKFC(ZooKeeper Failover Controller)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ZKFC来负责切换

切换时需要防止brain split现象的发生

## 1.2 HDFS-HA图解：



## 1.2 HA集群的安装部署

### 1.2.1 集群节点规划

集群部署节点角色的规划（10节点）：

|  |
| --- |
| server01 namenode zkfc > start-dfs.sh  server02 namenode zkfc  server03 resourcemanager > start-yarn.sh  server04 resourcemanager  server05 datanode nodemanager  server06 datanode nodemanager  server07 datanode nodemanager  server08 journal node zookeeper  server09 journal node zookeeper  server10 journal node zookeeper |

集群部署节点角色的规划（3节点）

|  |
| --- |
| server01 namenode resourcemanager zkfc nodemanager datanode zookeeper journal node  server02 namenode resourcemanager zkfc nodemanager datanode zookeeper journal node  server05 datanode nodemanager zookeeper journal node |

### 1.2.2 环境准备

1、环境准备

a/linux系统准备

ip地址配置

hostname配置

hosts映射配置

防火墙关闭

init启动级别修改

sudoers加入hadoop用户

ssh免密登陆配置

b/java环境的配置

上传jdk，解压，修改/etc/profile

c/zookeeper集群的部署

### 1.2.3 配置文件

core-site.xml

|  |
| --- |
| <configuration>  <!-- 指定hdfs的nameservice为ns1 -->  <property>  <name>fs.defaultFS</name>  <value>hdfs://ns1/</value>  </property>  <!-- 指定hadoop临时目录 -->  <property>  <name>hadoop.tmp.dir</name>  <value>/home/hadoop/app/hadoop-2.4.1/tmp</value>  </property>  <!-- 指定zookeeper地址 -->  <property>  <name>ha.zookeeper.quorum</name>  <value>weekend05:2181,weekend06:2181,weekend07:2181</value>  </property>  </configuration> |

hdfs-site.xml

|  |
| --- |
| <configuration>  <!--指定hdfs的nameservice为ns1，需要和core-site.xml中的保持一致 -->  <property>  <name>dfs.nameservices</name>  <value>ns1</value>  </property>  <!-- ns1下面有两个NameNode，分别是nn1，nn2 -->  <property>  <name>dfs.ha.namenodes.ns1</name>  <value>nn1,nn2</value>  </property>  <!-- nn1的RPC通信地址 -->  <property>  <name>dfs.namenode.rpc-address.ns1.nn1</name>  <value>weekend01:9000</value>  </property>  <!-- nn1的http通信地址 -->  <property>  <name>dfs.namenode.http-address.ns1.nn1</name>  <value>weekend01:50070</value>  </property>  <!-- nn2的RPC通信地址 -->  <property>  <name>dfs.namenode.rpc-address.ns1.nn2</name>  <value>weekend02:9000</value>  </property>  <!-- nn2的http通信地址 -->  <property>  <name>dfs.namenode.http-address.ns1.nn2</name>  <value>weekend02:50070</value>  </property>  <!-- 指定NameNode的edits元数据在JournalNode上的存放位置 -->  <property>  <name>dfs.namenode.shared.edits.dir</name>  <value>qjournal://weekend05:8485;weekend06:8485;weekend07:8485/ns1</value>  </property>  <!-- 指定JournalNode在本地磁盘存放数据的位置 -->  <property>  <name>dfs.journalnode.edits.dir</name>  <value>/home/hadoop/app/hadoop-2.4.1/journaldata</value>  </property>  <!-- 开启NameNode失败自动切换 -->  <property>  <name>dfs.ha.automatic-failover.enabled</name>  <value>true</value>  </property>  <!-- 配置失败自动切换实现方式 -->  <property>  <name>dfs.client.failover.proxy.provider.ns1</name>  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>  </property>  <!-- 配置隔离机制方法，多个机制用换行分割，即每个机制暂用一行-->  <property>  <name>dfs.ha.fencing.methods</name>  <value>  sshfence  shell(/bin/true)  </value>  </property>  <!-- 使用sshfence隔离机制时需要ssh免登陆 -->  <property>  <name>dfs.ha.fencing.ssh.private-key-files</name>  <value>/home/hadoop/.ssh/id\_rsa</value>  </property>  <!-- 配置sshfence隔离机制超时时间 -->  <property>  <name>dfs.ha.fencing.ssh.connect-timeout</name>  <value>30000</value>  </property>  </configuration> |

### 1.2.4 集群运维测试

#### 1、Datanode动态上下线

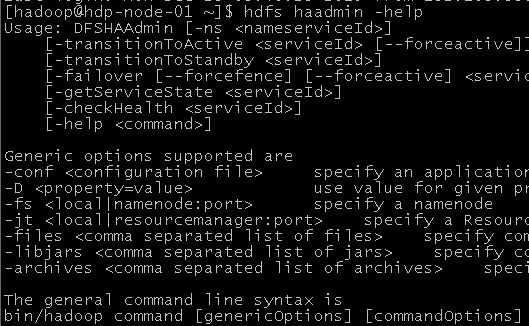
Datanode动态上下线很简单，步骤如下：

1. 准备一台服务器，设置好环境
2. 部署hadoop的安装包，并同步集群配置
3. 联网上线，新datanode会自动加入集群
4. 如果是一次增加大批datanode，还应该做集群负载重均衡

#### 2、Namenode状态切换管理

使用的命令上hdfs haadmin

可用 hdfs haadmin –help查看所有帮助信息



可以看到，状态操作的命令示例：

查看namenode工作状态

|  |
| --- |
| hdfs haadmin -getServiceState nn1 |

将standby状态namenode切换到active

|  |
| --- |
| hdfs haadmin –transitionToActive nn1 |

将active状态namenode切换到standby

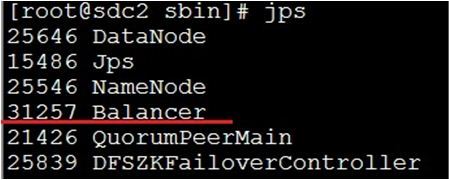
|  |
| --- |
| hdfs haadmin –transitionToStandby nn2 |

#### 3、数据块的balance

启动balancer的命令：

start-balancer.sh -threshold 8

运行之后，会有Balancer进程出现：



上述命令设置了Threshold为8%，那么执行balancer命令的时候，首先统计所有DataNode的磁盘利用率的均值，然后判断如果某一个DataNode的磁盘利用率超过这个均值Threshold，那么将会把这个DataNode的block转移到磁盘利用率低的DataNode，这对于新节点的加入来说十分有用。Threshold的值为1到100之间，不显示的进行参数设置的话，默认是10。

### 1.2.5 HA下hdfs-api变化

客户端需要nameservice的配置信息，其他不变

|  |
| --- |
| /\*\*  \* 如果访问的是一个ha机制的集群  \* 则一定要把core-site.xml和hdfs-site.xml配置文件放在客户端程序的classpath下  \* 以让客户端能够理解hdfs://ns1/中“ns1”是一个ha机制中的namenode对——nameservice  \* 以及知道ns1下具体的namenode通信地址  \* @author  \*  \*/  public class UploadFile {    public static void main(String[] args) throws Exception {    Configuration conf = new Configuration();  conf.set("fs.defaultFS", "hdfs://ns1/");    FileSystem fs = FileSystem.get(new URI("hdfs://ns1/"),conf,"hadoop");    fs.copyFromLocalFile(new Path("g:/eclipse-jee-luna-SR1-linux-gtk.tar.gz"), new Path("hdfs://ns1/"));    fs.close();  }  } |

\* Federation下 mr程序运行的staging提交目录问题

<property>

<name>yarn.app.mapreduce.am.staging-dir</name>

<value>/bi/tmp/hadoop-yarn/staging</value>

<description>The staging dir used while submitting jobs.

</description>

</property>

# Hadoop的Yarn的HA机制

2.1 Yarn的HA机制相对于HDFS的HA机制

还是比较简单的，依靠ZK来做即可，如果一台Yarn挂掉，那么ZK上的临时节点就会被删除，MRAppMaster就会重新分配一个节点重新运行这个Yarn上的程序

# Hadoop中HDFS的NameNode的横向扩展