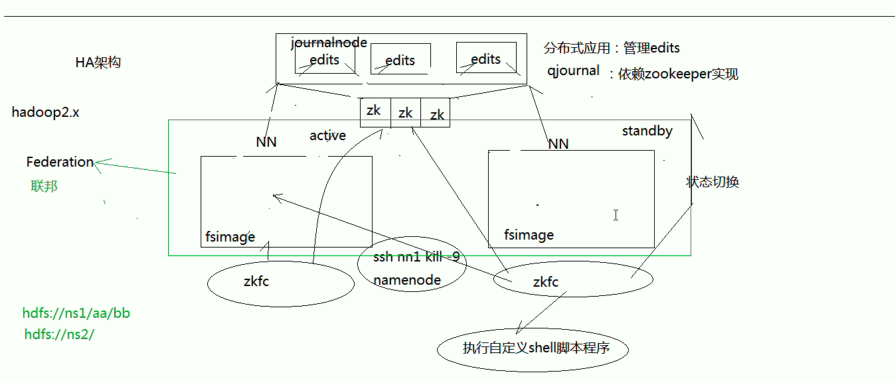


3.如何避免状态切换时的脑裂（brain split）



先杀掉他（没有返回，执行自定义shell脚本程序），然后切换状态。

fencing机制 ：

ssh发送kill指令

执行自定义shell脚本

Namenode1 namenode2

Zkfc1 zkfc2

Datanode 1

Nn1 nn2

Zkfc1 zkfc2

Zk1 zk2 zk3

Jn1 jn2 jn3

Dn1

Hadoop-HA安装

1.修改Linux主机名

2.修改IP

3.修改主机名和IP的映射关系

######注意######如果你们公司是租用的服务器或是使用的云主机（如华为用主机、阿里云主机等）

/etc/hosts里面要配置的是内网IP地址和主机名的映射关系

4.关闭防火墙

5.ssh免登陆

6.安装JDK，配置环境变量等

集群规划：

主机名 IP 安装的软件 运行的进程

weekend01 192.168.1.201 jdk、hadoop NameNode、DFSZKFailoverController(zkfc)

weekend02 192.168.1.202 jdk、hadoop NameNode、DFSZKFailoverController(zkfc)

weekend03 192.168.1.203 jdk、hadoop ResourceManager

weekend04 192.168.1.204 jdk、hadoop ResourceManager

weekend05 192.168.1.205 jdk、hadoop、zookeeper DataNode、NodeManager、JournalNode、QuorumPeerMain

weekend06 192.168.1.206 jdk、hadoop、zookeeper DataNode、NodeManager、JournalNode、QuorumPeerMain

weekend07 192.168.1.207 jdk、hadoop、zookeeper DataNode、NodeManager、JournalNode、QuorumPeerMain

说明：

1.在hadoop2.0中通常由两个NameNode组成，一个处于active状态，另一个处于standby状态。Active NameNode对外提供服务，而Standby NameNode则不对外提供服务，仅同步active namenode的状态，以便能够在它失败时快速进行切换。

hadoop2.0官方提供了两种HDFS HA的解决方案，一种是NFS，另一种是QJM。这里我们使用简单的QJM。在该方案中，主备NameNode之间通过一组JournalNode同步元数据信息，一条数据只要成功写入多数JournalNode即认为写入成功。通常配置奇数个JournalNode

这里还配置了一个zookeeper集群，用于ZKFC（DFSZKFailoverController）故障转移，当Active NameNode挂掉了，会自动切换Standby NameNode为standby状态

2.hadoop-2.2.0中依然存在一个问题，就是ResourceManager只有一个，存在单点故障，hadoop-2.4.1解决了这个问题，有两个ResourceManager，一个是Active，一个是Standby，状态由zookeeper进行协调

安装步骤：

1.安装配置zooekeeper集群（在weekend05上）

1.1解压

tar -zxvf zookeeper-3.4.5.tar.gz -C /weekend/

1.2修改配置

cd /weekend/zookeeper-3.4.5/conf/

cp zoo\_sample.cfg zoo.cfg

vim zoo.cfg

修改：dataDir=/weekend/zookeeper-3.4.5/tmp

在最后添加：

server.1=weekend05:2888:3888

server.2=weekend06:2888:3888

server.3=weekend07:2888:3888

保存退出

然后创建一个tmp文件夹

mkdir /weekend/zookeeper-3.4.5/tmp

再创建一个空文件

touch /weekend/zookeeper-3.4.5/tmp/myid

最后向该文件写入ID

echo 1 > /weekend/zookeeper-3.4.5/tmp/myid

1.3将配置好的zookeeper拷贝到其他节点(首先分别在weekend06、weekend07根目录下创建一个weekend目录：mkdir /weekend)

scp -r /weekend/zookeeper-3.4.5/ weekend06:/weekend/

scp -r /weekend/zookeeper-3.4.5/ weekend07:/weekend/

注意：修改weekend06、weekend07对应/weekend/zookeeper-3.4.5/tmp/myid内容

weekend06：

echo 2 > /weekend/zookeeper-3.4.5/tmp/myid

weekend07：

echo 3 > /weekend/zookeeper-3.4.5/tmp/myid

2.安装配置hadoop集群（在weekend01上操作）

2.1解压

tar -zxvf hadoop-2.4.1.tar.gz -C /weekend/

2.2配置HDFS（hadoop2.0所有的配置文件都在$HADOOP\_HOME/etc/hadoop目录下）

#将hadoop添加到环境变量中

vim /etc/profile

export JAVA\_HOME=/usr/java/jdk1.7.0\_55

export HADOOP\_HOME=/weekend/hadoop-2.4.1

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin

#hadoop2.0的配置文件全部在$HADOOP\_HOME/etc/hadoop下

cd /home/hadoop/app/hadoop-2.4.1/etc/hadoop

2.2.1修改hadoo-env.sh

export JAVA\_HOME=/home/hadoop/app/jdk1.7.0\_55

2.2.2修改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>

2.2.3修改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>

2.2.4修改mapred-site.xml

<configuration>

<!-- 指定mr框架为yarn方式 -->

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

2.2.5修改yarn-site.xml

<configuration>

<!-- 开启RM高可用 -->

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

<!-- 指定RM的cluster id -->

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>yrc</value>

</property>

<!-- 指定RM的名字 -->

<property>

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

<value>rm1,rm2</value>

</property>

<!-- 分别指定RM的地址 -->

<property>

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

<value>weekend03</value>

</property>

<property>

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

<value>weekend04</value>

</property>

<!-- 指定zk集群地址 -->

<property>

<name>yarn.resourcemanager.zk-address</name>

<value>weekend05:2181,weekend06:2181,weekend07:2181</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

</configuration>

2.2.6修改slaves(slaves是指定子节点的位置，因为要在weekend01上启动HDFS、在weekend03启动yarn，所以weekend01上的slaves文件指定的是datanode的位置，weekend03上的slaves文件指定的是nodemanager的位置)

weekend05

weekend06

weekend07

2.2.7配置免密码登陆

#首先要配置weekend01到weekend02、weekend03、weekend04、weekend05、weekend06、weekend07的免密码登陆

#在weekend01上生产一对钥匙

ssh-keygen -t rsa

#将公钥拷贝到其他节点，包括自己

ssh-coyp-id weekend01

ssh-coyp-id weekend02

ssh-coyp-id weekend03

ssh-coyp-id weekend04

ssh-coyp-id weekend05

ssh-coyp-id weekend06

ssh-coyp-id weekend07

#配置weekend03到weekend04、weekend05、weekend06、weekend07的免密码登陆

#在weekend03上生产一对钥匙

ssh-keygen -t rsa

#将公钥拷贝到其他节点

ssh-coyp-id weekend04

ssh-coyp-id weekend05

ssh-coyp-id weekend06

ssh-coyp-id weekend07

#注意：两个namenode之间要配置ssh免密码登陆，别忘了配置weekend02到weekend01的免登陆

在weekend02上生产一对钥匙

ssh-keygen -t rsa

ssh-coyp-id -i weekend01

2.4将配置好的hadoop拷贝到其他节点

scp -r /weekend/ weekend02:/

scp -r /weekend/ weekend03:/

scp -r /weekend/hadoop-2.4.1/ hadoop@weekend04:/weekend/

scp -r /weekend/hadoop-2.4.1/ hadoop@weekend05:/weekend/

scp -r /weekend/hadoop-2.4.1/ hadoop@weekend06:/weekend/

scp -r /weekend/hadoop-2.4.1/ hadoop@weekend07:/weekend/

###注意：严格按照下面的步骤

2.5启动zookeeper集群（分别在weekend05、weekend06、tcast07上启动zk）

cd /weekend/zookeeper-3.4.5/bin/

./zkServer.sh start

#查看状态：一个leader，两个follower

./zkServer.sh status

2.6启动journalnode（分别在在weekend05、weekend06、tcast07上执行）

cd /weekend/hadoop-2.4.1

sbin/hadoop-daemon.sh start journalnode

#运行jps命令检验，weekend05、weekend06、weekend07上多了JournalNode进程

2.7格式化HDFS

#在weekend01上执行命令:

hdfs namenode -format

#格式化后会在根据core-site.xml中的hadoop.tmp.dir配置生成个文件，这里我配置的是/weekend/hadoop-2.4.1/tmp，然后将/weekend/hadoop-2.4.1/tmp拷贝到weekend02的/weekend/hadoop-2.4.1/下。

scp -r tmp/ weekend02:/home/hadoop/app/hadoop-2.4.1/

##也可以这样，建议hdfs namenode -bootstrapStandby

2.8格式化ZKFC(在weekend01上执行即可)

hdfs zkfc -formatZK

2.9启动HDFS(在weekend01上执行)

sbin/start-dfs.sh

2.10启动YARN(#####注意#####：是在weekend03上执行start-yarn.sh，把namenode和resourcemanager分开是因为性能问题，因为他们都要占用大量资源，所以把他们分开了，他们分开了就要分别在不同的机器上启动)

sbin/start-yarn.sh

到此，hadoop-2.4.1配置完毕，可以统计浏览器访问:

http://192.168.1.201:50070

NameNode 'weekend01:9000' (active)

http://192.168.1.202:50070

NameNode 'weekend02:9000' (standby)

验证HDFS HA

首先向hdfs上传一个文件

hadoop fs -put /etc/profile /profile

hadoop fs -ls /

然后再kill掉active的NameNode

kill -9 <pid of NN>

通过浏览器访问：http://192.168.1.202:50070

NameNode 'weekend02:9000' (active)

这个时候weekend02上的NameNode变成了active

在执行命令：

hadoop fs -ls /

-rw-r--r-- 3 root supergroup 1926 2014-02-06 15:36 /profile

刚才上传的文件依然存在！！！

手动启动那个挂掉的NameNode

sbin/hadoop-daemon.sh start namenode

通过浏览器访问：http://192.168.1.201:50070

NameNode 'weekend01:9000' (standby)

验证YARN：

运行一下hadoop提供的demo中的WordCount程序：

hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.4.1.jar wordcount /profile /out

OK，大功告成！！！

测试集群工作状态的一些指令 ：

bin/hdfs dfsadmin -report 查看hdfs的各节点状态信息

bin/hdfs haadmin -getServiceState nn1 获取一个namenode节点的HA状态

sbin/hadoop-daemon.sh start namenode 单独启动一个namenode进程

./hadoop-daemon.sh start zkfc 单独启动一个zkfc进程

Hdfs动态增加节点和副本数量管理

HA的java api 访问要点

package cn.itacst.hadoop.hdfs;

import java.io.FileInputStream;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.URI;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IOUtils;

public class HDFS\_HA {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://ns1");

conf.set("dfs.nameservices", "ns1");

conf.set("dfs.ha.namenodes.ns1", "nn1,nn2");

conf.set("dfs.namenode.rpc-address.ns1.nn1", "itcast01:9000");

conf.set("dfs.namenode.rpc-address.ns1.nn2", "itcast02:9000");

//conf.setBoolean(name, value);

conf.set("dfs.client.failover.proxy.provider.ns1", "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider");

FileSystem fs = FileSystem.get(new URI("hdfs://ns1"), conf, "hadoop");

InputStream in =new FileInputStream("c://eclipse.rar");

OutputStream out = fs.create(new Path("/eclipse"));

IOUtils.copyBytes(in, out, 4096, true);

}

}