

Hadoop HA 集群搭建

目录

1、Hadoop HA 原理概述	1
2、集群规划	2
3、集群服务器准备	3
4、集群安装	3
5、集群启动测试	14

1、Hadoop HA 原理概述

为什么会有 hadoop HA 机制呢？

HA: High Available, 高可用

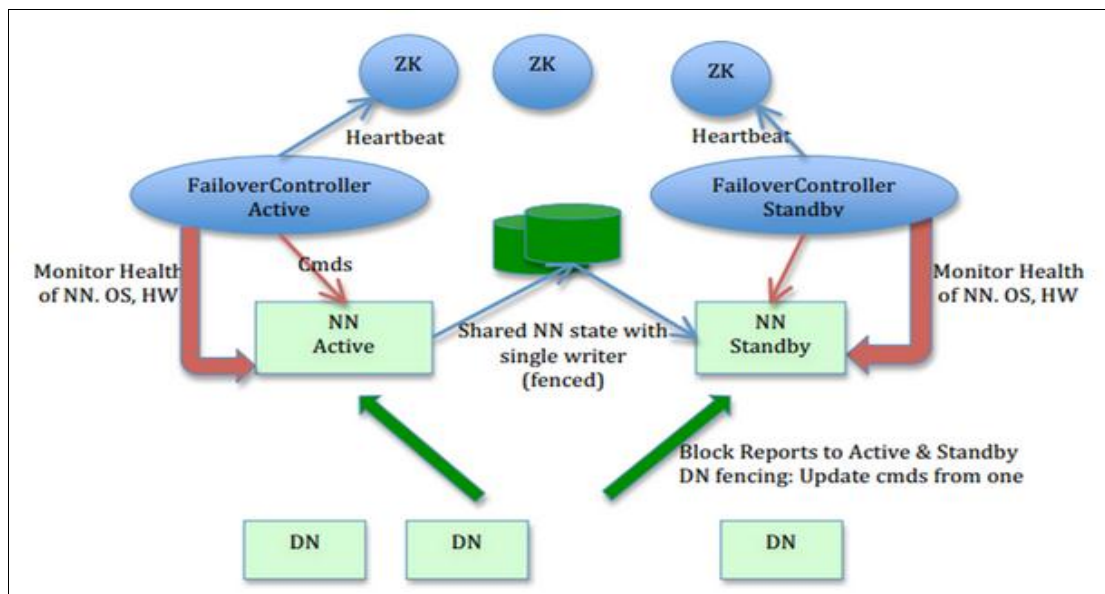
在 Hadoop 2.0 之前,在 HDFS 集群中 NameNode 存在单点故障 (SPOF: A Single Point of Failure)。对于只有一个 NameNode 的集群,如果 NameNode 机器出现故障(比如宕机或是软件、硬件升级),那么整个集群将无法使用,直到 NameNode 重新启动

那如何解决呢？

HDFS 的 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。

在一个典型的 HDFS(HA) 集群中,使用两台单独的机器配置为 NameNodes。在任何时间点,确保 NameNodes 中只有一个处于 Active 状态,其他的处在 Standby 状态。其中 ActiveNameNode 负责集群中的所有客户端操作,StandbyNameNode 仅仅充当备机,保证一旦 ActiveNameNode 出现问题能够快速切换。

为了能够实时同步 Active 和 Standby 两个 NameNode 的元数据信息(实际上 editlog),需提供一个共享存储系统,可以是 NFS、QJM (Quorum Journal Manager) 或者 Zookeeper, Active Namenode 将数据写入共享存储系统,而 Standby 监听该系统,一旦发现有新数据写入,则读取这些数据,并加载到自己内存中,以保证自己内存状态与 Active NameNode 保持基本一致,如此这般,在紧急情况下 standby 便可快速切为 active namenode。为了实现快速切换,Standby 节点获取集群的最新文件块信息也是很有必要的。为了实现这一目标,DataNode 需要配置 NameNodes 的位置,并同时给他们发送文件块信息以及心跳检测。



思考问题: **SecondaryNameNode** 和 **Standby Namenode** 的区别?

2、集群规划

描述: hadoop HA 集群的搭建依赖于 zookeeper, 所以选取三台当做 zookeeper 集群
我总共准备了四台主机, 分别是 hadoop02, hadoop03, hadoop04, hadoop05
其中 hadoop02 和 hadoop03 做 namenode 的主备切换, hadoop04 和 hadoop05 做 resourcemanager 的主备切换

四台:

	hadoop02	hadoop03	hadoop04	hadoop05
namenode	√	√		
datanode	√	√	√	√
resourcemanager			√	√
nodemanager	√	√	√	√
zookeeper	√	√	√	
journalnode	√	√	√	
zkfc	√	√		

五台:

	hadoop01	hadoop02	hadoop03	hadoop04	hadoop05
namenode	√	√			
datanode	√	√	√	√	√
resourcemanager			√	√	
nodemanager	√	√	√	√	√
zookeeper	√	√	√	√	√
journalnode		√	√	√	

zkfc	√	√				
------	---	---	--	--	--	--

七台：

	01	02	03	04	05	06	07
namenode	√	√					
datanode					√	√	√
resourcemanager			√	√			
nodemanager					√	√	√
zookeeper					√	√	√
journalnode					√	√	√
zkfc	√	√					

具体怎样规划，其实可以自由设置。

3、集群服务器准备

- 1、修改主机名
- 2、修改 IP 地址
- 3、添加主机名和 IP 映射
- 4、添加普通用户 `hadoop` 用户并配置 `sudoer` 权限
- 5、设置系统启动级别
- 6、关闭防火墙/关闭 `Selinux`
- 7、安装 `JDK`

两种准备方式：

- 1、每个节点都单独设置，这样比较麻烦。线上环境可以编写脚本实现
- 2、虚拟机环境可是在做完以上 7 步之后，就进行克隆
- 3、然后接着再给你的集群配置 `SSH` 免密登陆和搭建时间同步服务
- 8、配置 `SSH` 免密登录
- 9、同步服务器时间

4、集群安装

- 1、安装 `Zookeeper` 集群
在此略过，见 `zookeeper` 安装文档

- 2、安装 `hadoop` 集群

- 1、上传安装包 `hadoop-2.6.5-centos-6.7.tar.gz`
- 2、解压到对应的安装目录
`[hadoop@hadoop02 ~]# tar -zxvf hadoop-2.6.5-centos-6.7.tar.gz -C /home/hadoop/apps/`
- 3、修改配置文件

1、修改 hadoo-env.sh

修改一行

```
export JAVA_HOME= /usr/local/jdk1.8.0_73
```

2、修改 core-site.xml

```
<configuration>
```

```
<!-- 指定 hdfs 的 nameservice 为 myha01 -->
```

```
<property>
```

```
<name>fs.defaultFS</name>
```

```
<value>hdfs://myha01</value>
```

```
</property>
```

```
<!-- 指定 hadoop 工作目录 -->
```

```
<property>
```

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

```
<value>/home/hadoop/data/hadoopdata</value>
```

```
</property>
```

```
<!-- 指定 zookeeper 集群访问地址 -->
```

```
<property>
```

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

```
<value>hadoop02:2181,hadoop03:2181,hadoop04:2181</value>
```

```
</property>
```

```
</configuration>
```

3、修改 hdfs-site.xml

```
<configuration>
```

```
<!-- 指定副本数 -->
```

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>2</value>
```

```
</property>
```

```
<!--指定 hdfs 的 nameservice 为 myha01，需要和 core-site.xml 中保持一致-->
```

```
<property>
```

```
<name>dfs.nameservices</name>
```

```
<value>myha01</value>
```

```
</property>
```

```
<!-- myha01 下面有两个 NameNode，分别是 nn1，nn2 -->
```

```
<property>
```

```
<name>dfs.ha.namenodes.myha01</name>
```

```
<value>nn1,nn2</value>
```

```
</property>
```

```
<!-- nn1 的 RPC 通信地址 -->
<property>
  <name>dfs.namenode.rpc-address.myha01.nn1</name>
  <value>hadoop02:9000</value>
</property>

<!-- nn1 的 http 通信地址 -->
<property>
  <name>dfs.namenode.http-address.myha01.nn1</name>
  <value>hadoop02:50070</value>
</property>

<!-- nn2 的 RPC 通信地址 -->
<property>
  <name>dfs.namenode.rpc-address.myha01.nn2</name>
  <value>hadoop03:9000</value>
</property>

<!-- nn2 的 http 通信地址 -->
<property>
  <name>dfs.namenode.http-address.myha01.nn2</name>
  <value>hadoop03:50070</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://hadoop02:8485;hadoop03:8485;hadoop04:8485/myha01</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/data/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 -->
```

```
<!-- 此处配置在安装的时候切记检查不要换行-->
<property>
  <name>dfs.client.failover.proxy.provider.myha01</name>
  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverPr
oxyProvider</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>
```

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>

  <!-- 设置 mapreduce 的历史服务器地址和端口号 -->
  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>hadoop02:10020</value>
  </property>

  <!-- mapreduce 历史服务器的 web 访问地址 -->
```

```
<property>
  <name>mapreduce.jobhistory.webapp.address</name>
  <value>hadoop02:19888</value>
</property>
</configuration>
```

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>ycrc</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>hadoop04</value>
</property>

<property>
  <name>yarn.resourcemanager.hostname.rm2</name>
  <value>hadoop05</value>
</property>

<!-- 指定 zk 集群地址 -->
<property>
  <name>yarn.resourcemanager.zk-address</name>
  <value>hadoop02:2181,hadoop03:2181,hadoop04:2181</value>
</property>

<!-- 要运行 MapReduce 程序必须配置的附属服务 -->
```

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>

<!-- 开启 YARN 集群的日志聚合功能 -->
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
</property>

<!-- YARN 集群的聚合日志最长保留时长 -->
<property>
  <name>yarn.log-aggregation.retain-seconds</name>
  <value>86400</value>
</property>

<!-- 启用自动恢复 -->
<property>
  <name>yarn.resourcemanager.recovery.enabled</name>
  <value>true</value>
</property>

<!-- 制定 resourcemanager 的状态信息存储在 zookeeper 集群上-->
<property>
  <name>yarn.resourcemanager.store.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>
</property>
</configuration>
```

6、修改 slaves

vi slaves

添加 datanode 的节点地址:

```
hadoop02
hadoop03
hadoop04
hadoop05
```

7、分发安装包到其他机器

```
scp -r hadoop-2.6.5 hadoop@hadoop03:$PWD
scp -r hadoop-2.6.5 hadoop@hadoop04:$PWD
scp -r hadoop-2.6.5 hadoop@hadoop05:$PWD
```


8、并分别配置环境变量

vi ~/.bashrc

添加两行：

export HADOOP_HOME=/home/hadoop/apps/hadoop-2.6.5

export PATH=\$PATH:\$HADOOP_HOME/bin:\$HADOOP_HOME/sbin

保存退出

4、集群初始化操作（记住：严格按照以下步骤执行）

1、先启动 zookeeper 集群

启动：zkServer.sh start

检查启动是否正常：zkServer.sh status

2、分别在每个 zookeeper（也就是规划的三个 journalnode 节点，不一定跟 zookeeper 节点一样）节点上启动 journalnode 进程

[hadoop@hadoop02 ~]\$ hadoop-daemon.sh start journalnode

[hadoop@hadoop03 ~]\$ hadoop-daemon.sh start journalnode

[hadoop@hadoop04 ~]\$ hadoop-daemon.sh start journalnode

然后用 jps 命令查看是否各个 datanode 节点上都启动了 journalnode 进程

如果报错，根据错误提示改进

3、在第一个 namenode 上执行格式化操作

[hadoop@hadoop02 ~]\$ hadoop namenode -format

```

hadoop02 x | hadoop03 | hadoop04 | hadoop05
17/07/11 08:01:48 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
17/07/11 08:01:48 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension = 30000
17/07/11 08:01:48 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
17/07/11 08:01:48 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 milliseconds
17/07/11 08:01:48 INFO util.GSet: Computing capacity for map NameNodeRetryCache
17/07/11 08:01:48 INFO util.GSet: VM type = 64-bit
17/07/11 08:01:48 INFO util.GSet: 0.029999999329447746% max memory 966.7 MB = 297.0 KB
17/07/11 08:01:48 INFO util.GSet: capacity = 2^15 = 32768 entries
17/07/11 08:01:48 INFO namenode.NNConf: ACLs enabled? false
17/07/11 08:01:48 INFO namenode.NNConf: XAttrs enabled? true
17/07/11 08:01:48 INFO namenode.NNConf: Maximum size of an xattr: 16384
17/07/11 08:01:49 INFO namenode.FSImage: Allocated new BlockPoolId: BP-1914185581-192.168.123.102-1499731309485
17/07/11 08:01:49 INFO common.Storage: Storage directory /home/hadoop/data/hadoopdata/dfs/name has been successfully formatted.
17/07/11 08:01:49 INFO namenode.FSImageFormatProtobuf: Saving image file /home/hadoop/data/hadoopdata/dfs/name/current/fsimage.ckpt_00000000000000000000 using no compression
17/07/11 08:01:49 INFO namenode.FSImageFormatProtobuf: Image file /home/hadoop/data/hadoopdata/dfs/name/current/fsimage.ckpt_00000000000000000000 of size 323 bytes saved in 0 seconds.
17/07/11 08:01:49 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
17/07/11 08:01:49 INFO util.ExitUtil: Exiting with status 0
17/07/11 08:01:49 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at hadoop02/192.168.123.102
*****/
[hadoop@hadoop02 ~]$

```

然后会在 core-site.xml 中配置的临时目录中生成一些集群的信息
把他拷贝的第二个 namenode 的相同目录下

```

<name>hadoop.tmp.dir</name>
<value>/home/hadoop/data/hadoopdata/</value>

```

这个目录下，千万记住：两个 namenode 节点该目录中的数据结构是一致的

[hadoop@hadoop02 ~]\$ scp -r ~/data/hadoopdata/ hadoop03:~/data

或者也可以在另一个 namenode 上执行：hadoop namenode -bootstrapStandby

```

hadoop02 x | hadoop03 | hadoop04 | hadoop05
[hadoop@hadoop02 data]$ scp -r ~/data/hadoopdata/ hadoop03:~/data
seen_txid
VERSION
fsimage_00000000000000000000.md5
fsimage_00000000000000000000
[hadoop@hadoop02 data]$

```

4、格式化 ZKFC

[hadoop@hadoop02 ~]\$ hdfs zkfc -formatZK

在第一台机器上即可

```

hadoop02 x | hadoop03 | hadoop04 | hadoop05
re-2.6.5.jar:/home/hadoop/apps/hadoop-2.6.5/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.6.5.jar:/home/hadoop/apps/hadoop-2.6.5/share/hadoop/mapreduce/hadoop-mapreduce-client-hs-2.6.5.jar:/home/hadoop/apps/hadoop-2.6.5/contrib/capacity-scheduler/*.jar
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:java.library.path=/home/hadoop/apps/hadoop-2.6.5/lib/native
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:java.io.tmpdir=/tmp
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:java.compiler=<NA>
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:os.name=Linux
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:os.arch=amd64
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:os.version=2.6.32-573.el6.x86_64
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:user.name=hadoop
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:user.home=/home/hadoop
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Client environment:user.dir=/home/hadoop/data
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=hadoop02:2181,hadoop03:2181,hadoop04:2181 sessionTimeout=5000 watcher=org.apache.hadoop.ha.ActiveStandbyElector$WatcherWithClientRef@158d2680
17/07/11 08:05:10 INFO zookeeper.ClientCnxn: Opening socket connection to server hadoop03/192.168.123.103:2181. Will not attempt to authenticate using SASL (unknown error)
17/07/11 08:05:10 INFO zookeeper.ClientCnxn: Socket connection established to hadoop03/192.168.123.103:2181, initiating session
17/07/11 08:05:10 INFO zookeeper.ClientCnxn: Session establishment complete on server hadoop03/192.168.123.103:2181, sessionId = 0x35d2ef0fd310000, negotiated timeout = 5000
17/07/11 08:05:10 INFO ha.ActiveStandbyElector: Session connected.
17/07/11 08:05:10 INFO ha.ActiveStandbyElector: Successfully created /hadoop-ha/myha01 in ZK.
17/07/11 08:05:10 INFO zookeeper.ZooKeeper: Session: 0x35d2ef0fd310000 closed
17/07/11 08:05:10 INFO zookeeper.ClientCnxn: EventThread shut down
[hadoop@hadoop02 data]$

```

5、启动 HDFS

[hadoop@hadoop02 ~]\$ start-dfs.sh

```

[had00p@hadoop02 data]$ start-dfs.sh
Starting namenodes on [hadoop02 hadoop03]
hadoop03: starting namenode, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-namenode-hadoop03.out
hadoop02: starting namenode, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-namenode-hadoop02.out
hadoop04: starting datanode, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-datanode-hadoop04.out
hadoop05: starting datanode, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-datanode-hadoop05.out
hadoop03: starting datanode, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-datanode-hadoop03.out
hadoop02: starting datanode, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-datanode-hadoop02.out
Starting journal nodes [hadoop02 hadoop03 hadoop04]
hadoop04: journalnode running as process 2640. Stop it first.
hadoop03: journalnode running as process 2637. Stop it first.
hadoop02: journalnode running as process 2788. Stop it first.
Starting ZK Failover Controllers on NN hosts [hadoop02 hadoop03]
hadoop03: starting zkfc, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-zkfc-hadoop03.out
hadoop02: starting zkfc, logging to /home/hadoop/apps/hadoop-2.6.5/logs/hadoop-hadoop-zkfc-hadoop02.out
[had00p@hadoop02 data]$

```

查看各节点进程是否启动正常：依次为 2345 四台机器的进程

```

[had00p@hadoop02 data]$ jps
3520 DFSZKFailoverController
3587 Jps
2788 JournalNode
3237 DataNode
2685 QuorumPeerMain
3103 NameNode
[had00p@hadoop02 data]$

[had00p@hadoop04 ~]$ jps
2640 JournalNode
2837 Jps
2742 DataNode
2526 QuorumPeerMain
[had00p@hadoop04 ~]$

[had00p@hadoop03 ~]$ jps
2784 NameNode
2531 QuorumPeerMain
3075 Jps
3019 DFSZKFailoverController
2637 JournalNode
2862 DataNode
[had00p@hadoop03 ~]$

[had00p@hadoop05 hadoop]$ jps
2658 QuorumPeerMain
2867 Jps
2796 DataNode
[had00p@hadoop05 hadoop]$

```

访问 web 页面 <http://hadoop02:50070>

Overview 'hadoop02:9000' (active)

Started:	Tue Jul 11 08:06:15 CST 2017
Version:	2.6.5, rUnknown
Compiled:	2017-01-22T01:18Z by root from Unknown
Cluster ID:	CID-6dcd7b67-3e15-4bbf-a3bf-aa25f258b82f
Block Pool ID:	BP-1914185581-192.168.123.102-1499731309485

访问 web 页面: <http://hadoop03:50070>

Started:	Tue Jul 11 08:06:15 CST 2017
Version:	2.6.5, rUnknown
Compiled:	2017-01-22T01:18Z by root from Unknown
Cluster ID:	CID-6dcd7b67-3e15-4bbf-a3bf-aa25f258b82f
Block Pool ID:	BP-1914185581-192.168.123.102-1499731309485

6、启动 YARN

[hadoop@hadoop04 ~]\$ start-yarn.sh

在主备 resourcemanager 中随便选择一台进行启动，正常启动之后，检查各节点的进程：

```

hadoop02:
3520 DFSZKFailoverController
2788 JournalNode
3780 Jps
3237 DataNode
2685 QuorumPeerMain
3646 NodeManager
3103 NameNode

hadoop03:
2784 NameNode
2531 QuorumPeerMain
3290 Jps
3147 NodeManager
3019 DFSZKFailoverController
2637 JournalNode
2862 DataNode

hadoop04:
2640 JournalNode
2993 NodeManager
3330 Jps
2742 DataNode
2889 ResourceManager
2526 QuorumPeerMain

hadoop05:
2658 QuorumPeerMain
2915 NodeManager
3044 Jps
2796 DataNode
  
```

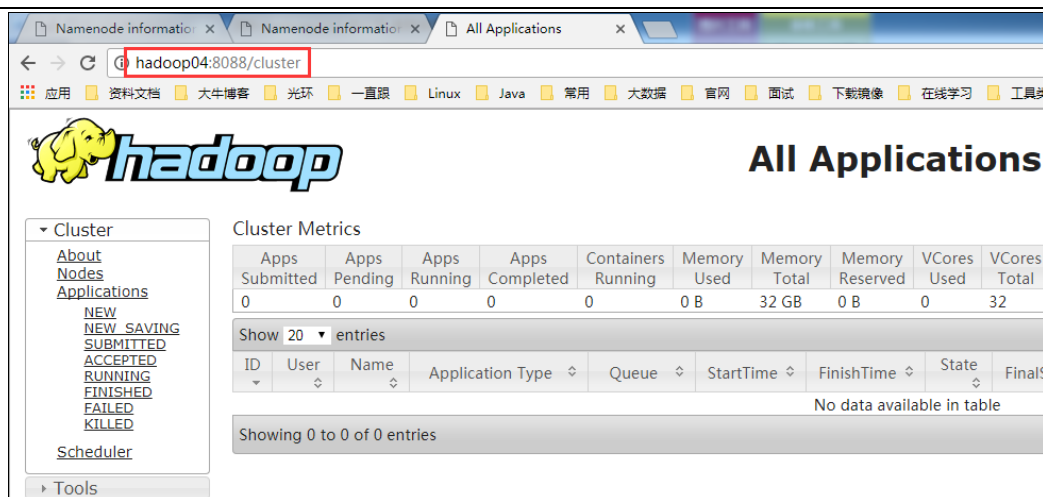
若备用节点的 resourcemanager 没有启动起来，则手动启动起来

[hadoop@hadoop04 ~]\$ yarn-daemon.sh start resourcemanager

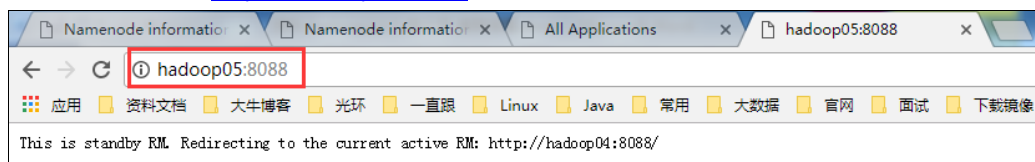
```

[hadoop@hadoop05 hadoop]$ yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/hadoop/apps/hadoop-2.6.5/
[hadoop@hadoop05 hadoop]$ jps
2658 QuorumPeerMain
2915 NodeManager
3080 ResourceManager
2796 DataNode
3133 Jps
  
```

之后打开浏览器访问页面：<http://hadoop04:8088>



访问 web 页面: <http://hadoop05:8088>



hadoop05 是 standby resourcemanager, 会自动跳转到 hadoop04

7、查看各主节点的状态

HDFS:

`hdfs haadmin -getServiceState nn1`

`hdfs haadmin -getServiceState nn2`

```
[hadoop@hadoop04 mapreduce]$ hdfs haadmin -getServiceState nn1
standby
[hadoop@hadoop04 mapreduce]$ hdfs haadmin -getServiceState nn2
active
```

YARN:

`yarn rmadmin -getServiceState rm1`

`yarn rmadmin -getServiceState rm2`

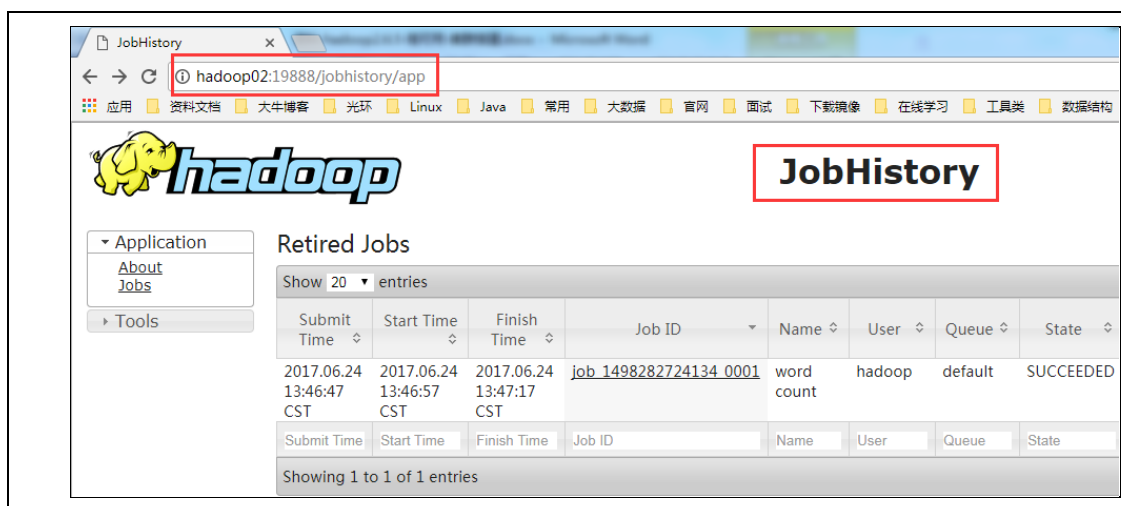
```
[hadoop@hadoop04 mapreduce]$ yarn rmadmin -getServiceState rm1
active
[hadoop@hadoop04 mapreduce]$ yarn rmadmin -getServiceState rm2
standby
```

8、启动 mapreduce 任务历史服务器

`[hadoop@hadoop02 ~]$ mr-jobhistory-daemon.sh start historyserver`

按照配置文件配置的历史服务器的 web 访问地址去访问:

<http://hadoop02:19888>



5、集群启动测试

- 1、干掉 active namenode， 看看集群有什么变化
- 2、在上传文件的时候干掉 active namenode， 看看有什么变化
- 3、干掉 active resourcemanager， 看看集群有什么变化
- 4、在执行任务的时候干掉 active resourcemanager， 看看集群有什么变化