手把手教你搭建 5 节点 Hadoop 分布式集群（HA）

1 目录

1.1 写在前面的话

本文章我们使用 hadoop2.6.0 版本配置 Hadoop 集群，同时配置 NameNode+HA、ResourceManager+HA，并使用 zookeeper 来管理 Hadoop 集群。

在开始讲解之前，先告诉大家一个小秘密，本文有对应的配套视频讲解哦，详情参考：http://www.dajiangtai.com/course/27.do?origin=invite-register&code=a8EYNf

1.2 (一)HDFS 概述

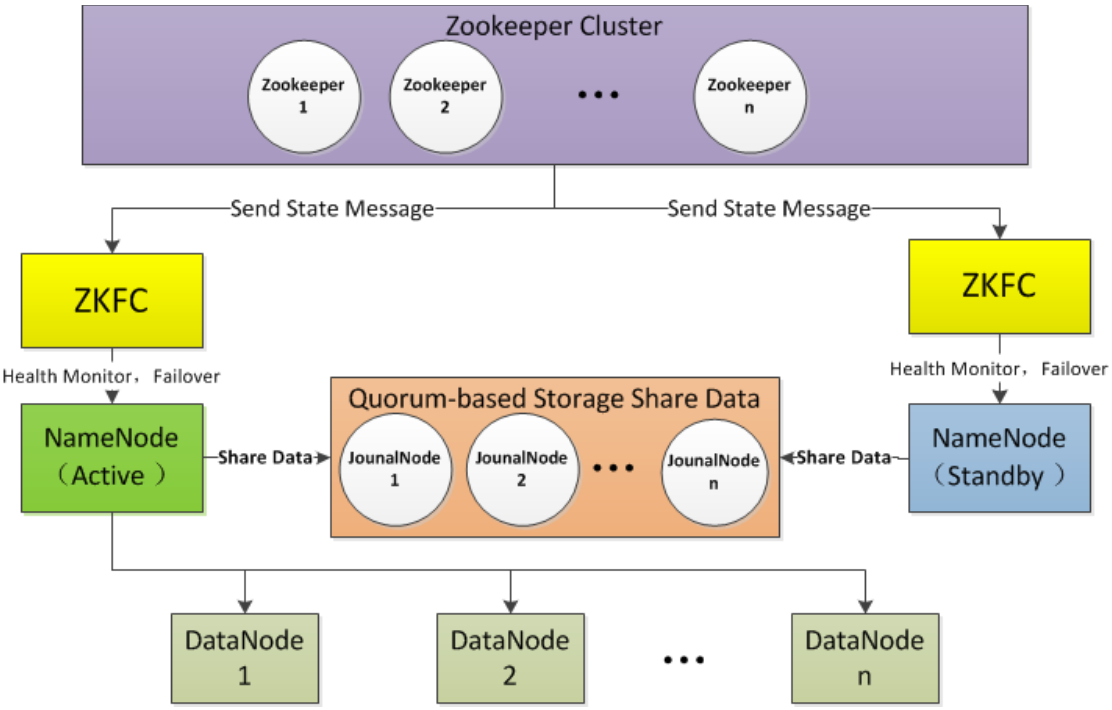
1.2.1 基础架构

1、NameNode（Master）

1)命名空间管理：命名空间支持对 HDFS 中的目录、文件和块做类似文件系统的创建、修改、删除、列表文件和目录等基本操作。

2)块存储管理。

1.2.2 HA 架构



从上面的架构图可以看出，使用 Active NameNode，Standby NameNode 两个节点可以解决单点问题，两个节点通过 JounalNode 共享状态，通过 ZKFC 选举 Active ，监控状态，自动备份。

1、Active NameNode

接受 client 的 RPC 请求并处理，同时写自己的 Editlog 和共享存储上的 Editlog，接收 DataNode 的 Block report, block location updates 和 heartbeat。

2、Standby NameNode

同样会接到来自 DataNode 的 Block report, block location updates 和 heartbeat，同时会从共享存储的 Editlog 上读取并执行这些 log 操作，保持自己 NameNode 中的元数据（Namespcae information + Block locations map）和 Active NameNode 中的元数据是同步的。所以说 Standby 模式的 NameNode 是一个热备（Hot Standby NameNode），一旦切换成 Active 模式，马上就可以提供 NameNode 服务。

3、JounalNode

用于 Active NameNode ， Standby NameNode 同步数据，本身由一组 JounnalNode 节点组成，该组节点奇数个。

4、ZKFC

监控 NameNode 进程，自动备份。

1.3 (二)YARN 概述

1.3.1 基础架构

1、ResourceManager(RM)
接收客户端任务请求，接收和监控 NodeManager(NM)的资源情况汇报，负责资源的分配与调度，启动和监控 ApplicationMaster(AM)。

2、NodeManager
节点上的资源管理，启动 Container 运行 task 计算，上报资源、container 情况汇报给 RM 和任务处理情况汇报给 AM。

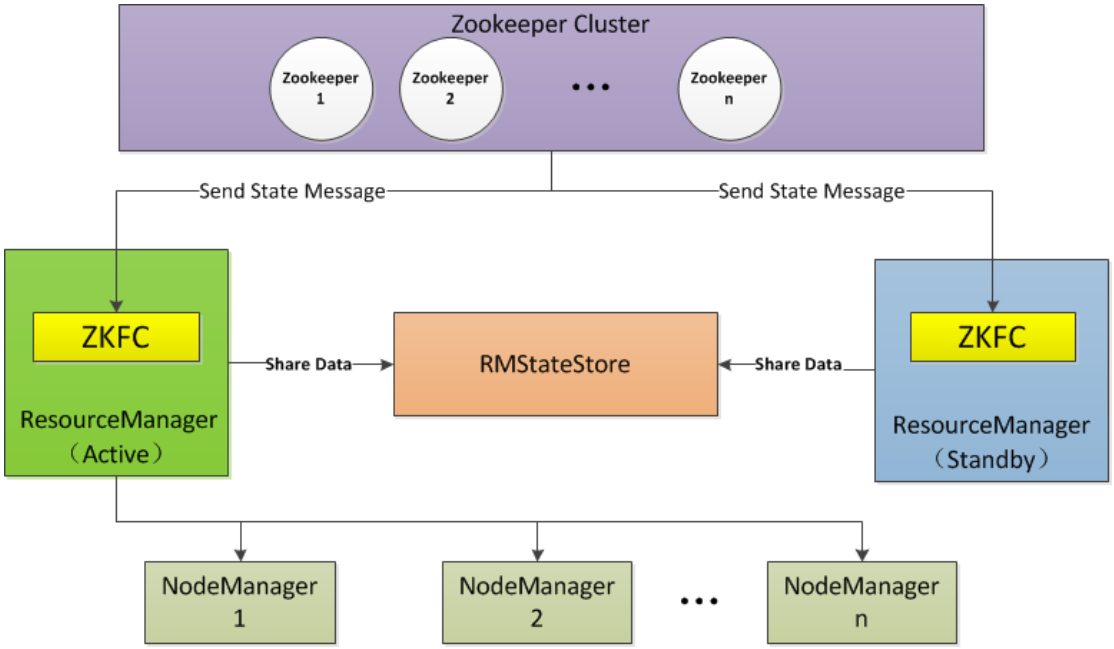
3、ApplicationMaster
单个 Application(Job)的 task 管理和调度，向 RM 进行资源的申请，向 NM 发出 launch Container 指令，接收 NM 的 task 处理状态信息。

4、Web Application Proxy
用于防止 Yarn 遭受 Web 攻击 ,本身是 ResourceManager 的一部分 ,可通过配置独立进程。ResourceManager Web 的访问基于守信用户，当 Application Master 运行于一个非受信用户 ,其提供给 ResourceManager 的可能是非受信连接 ,Web Application Proxy 可以阻止这种连接提供给 RM。

5、Job History Server
NodeManager 在启动的时候会初始化 LogAggregationService 服务，该服务会在把本机执行的 container log（在 container 结束的时候)收集并存放到 hdfs 指定的目录下. ApplicationMaster 会把 jobhistory 信息写到 hdfs 的 jobhistory 临时目录下，并在结束的时候

把 jobhisoty 移动到最终目录，这样就同时支持了 job 的 recovery.History 会启动 web 和 RPC 服务，用户可以通过网页或 RPC 方式获取作业的信息。

1.3.2 HA 架构



ResourceManager HA 由一对 Active，Standby 结点构成，通过 RMStateStore 存储内部数据和主要应用的数据及标记。目前支持的可替代的 RMStateStore 实现有：基于内存的 MemoryRMStateStore，基于文件系统的 FileSystemRMStateStore，及基于 zookeeper 的 ZKRMStateStore。 ResourceManager HA 的架构模式同 NameNode HA 的架构模式基本一致，数据共享由 RMStateStore，而 ZKFC 成为 ResourceManager 进程的一个服务，非独立存在。

1.4 (三)规划

1.4.1 主机规划

这里我们使用 5 台主机来配置 Hadoop 集群。

	djt11/192.168.1.171	djt17/192.168.1.172	djt13/192.168.1.173	djt14/192.168.1.174	djt15/192.168.1.175
namenode	是	是	否	否	否
datanode	否	否	是 • • •	是	是
resourcemanager	是	是	否	否	否
journalnode	是	是	是	是	是
zookeeper	是	是	是	是	是

Journalnode 和 ZooKeeper 保持奇数个，这点大家要有个概念，最少不少于 3 个节点。Zookeeper 课程中我们已经讲解过，这里就不再赘叙。

1.4.2 软件规划

软件	版本	位数	说明
jdk	jdk1.7	64 位	最新稳定版本
centos	centos6.5	64 位	
zookeeper	Apache zookeeper3.4.6		稳定版本
hadoop	Apache hadoop2.6.0		稳定版本

1.4.3 用户规划

每个节点的 hadoop 用户组和用户需要大家自己创建，单节点已经讲过，这里就不耽误大家时间。

节点名称	用户组	用户
djt11	hadoop	hadoop
djt12	hadoop	hadoop
djt13	hadoop	hadoop
djt14	hadoop	hadoop
djt15	hadoop	hadoop

1.4.4 目录规划

名称	路径
所有软件目录	/home/hadoop/app/
所有数据和日志目录	/home/hadoop/data/

1.5 (四)集群安装前的环境检查

1.5.1 时钟同步

所有节点的系统时间要与当前时间保持一致。

查看当前系统时间

```
date
Tue Nov  3 06:06:04 CST 2015
```

如果系统时间与当前时间不一致,进行以下操作。

```
[root@djt11 ~]# cd /usr/share/zoneinfo/
[root@djt11 zoneinfo]# ls          //找到 Asia
[root@djt11 zoneinfo]# cd Asia/            //进入 Asia 目录
[root@djt11 Asia]# ls              //找到 Shanghai
[root@djt11 Asia]# cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

```
//当前时区替换为上海
```

我们可以同步当前系统时间和日期与 NTP（网络时间协议）一致。

```
[root@djt11  Asia]# yum install ntp   //如果 ntp 命令不存在，在线安装 ntp
[root@djt11 Asia]# ntpdate pool.ntp.org            //执行此命令同步日期时间
[root@djt11 Asia]# date           //查看当前系统时间
```

1.5.2 hosts 文件检查

所有节点的 hosts 文件都要配置静态 ip 与 hostname 之间的对应关系。

```
[root@djt11 Asia]# vi /etc/hosts
192.168.1.171 djt11
192.168.1.172 djt12
192.168.1.173 djt13
192.168.1.174 djt14
192.168.1.175 djt15
```

1.5.3 禁用防火墙

所有节点的防火墙都要关闭。

查看防火墙状态

```
[root@djt11 Asia]# service iptables status
iptables: Firewall is not running.
```

如果不是上面的关闭状态，则需要关闭防火墙。

```
[root@djt11 Asia]#  chkconfig iptables off         //永久关闭防火墙
[root@djt11 Asia]#  service iptables stop          //临时关闭防火墙
```

1.6 (五)配置 SSH 免密码通信

这里我们以 djt11 来配置 ssh。

```
[root@djt11 ~]# su hadoop          //切换到 hadoop 用户下
[hadoop@djt11 root]$ cd            //切换到 hadoop 用户目录
[hadoop@djt11 ~]$ mkdir .ssh
[hadoop@djt11 ~]$ ssh-keygen -t rsa                //执行命令一路回车，生成秘钥
[hadoop@djt11 ~]$cd .ssh
[hadoop@djt11 .ssh]$ ls
```

```
id_rsa  id_rsa.pub
[hadoop@djt11 .ssh]$ cat id_rsa.pub >> authorized_keys
//将公钥保存到 authorized_keys 认证文件中
[hadoop@djt11 .ssh]$ ls
authorized_keys  id_rsa  id_rsa.pub
[hadoop@djt11 .ssh]$ cd ..
[hadoop@djt11 ~]$ chmod 700 .ssh
[hadoop@djt11 ~]$ chmod 600 .ssh/*
[hadoop@djt11 ~]$ ssh djt11                //第一次执行需要输入 yes
[hadoop@djt11 ~]$ ssh djt11                //第二次以后就可以直接访问
```

集群所有节点都要行上面的操作。

将所有节点中的共钥 id_ras.pub 拷贝到 djt11 中的 authorized_keys 文件中。

```
cat ~/.ssh/id_rsa.pub  | ssh hadoop@djt11 'cat >> ~/.ssh/authorized_keys'   所有节点都需要执行这条命令
```

然后将 djt11 中的 authorized_keys 文件分发到所有节点上面。

```
scp -r authorized_keys hadoop@djt12:~/.ssh/

scp -r authorized_keys hadoop@djt13:~/.ssh/

scp -r authorized_keys hadoop@djt14:~/.ssh/

scp -r authorized_keys hadoop@djt15:~/.ssh/
```

大家通过 ssh 相互访问，如果都能无密码访问，代表 ssh 配置成功。

1.7 (六)脚本工具的使用

在 djt11 节点上创建/home/hadoop/tools 目录。

```
[hadoop@djt11 ~]$ mkdir /home/hadoop/tools
cd /home/hadoop/tools
```

将本地脚本文件上传至/home/hadoop/tools 目录下。这些脚本大家如果能看懂也可以自己写，如果看不懂直接使用就可以，后面慢慢补补 Linux 相关的知识。

```
[hadoop@djt11 tools]$ rz deploy.conf
[hadoop@djt11 tools]$ rz deploy.sh
[hadoop@djt11 tools]$ rz runRemoteCmd.sh
[hadoop@djt11 tools]$ ls
```

```
deploy.conf  deploy.sh  runRemoteCmd.sh
```

查看一下 deploy.conf 配置文件内容。

```
[hadoop@djt11 tools]$ cat deploy.conf
djt11,all,namenode,zookeeper,resourcemanager,
djt12,all,slave,namenode,zookeeper,resourcemanager,
djt13,all,slave,datanode,zookeeper,
djt14,all,slave,datanode,zookeeper,
djt15,all,slave,datanode,zookeeper,
```

查看一下 deploy.sh 远程复制文件脚本内容。

```
[hadoop@djt11 tools]$ cat deploy.sh
#!/bin/bash
#set -x

if [ $# -lt 3 ]
then
  echo "Usage: ./deply.sh srcFile(or Dir) descFile(or Dir) MachineTag"
  echo "Usage: ./deply.sh srcFile(or Dir) descFile(or Dir) MachineTag confFi
le"
  exit
fi

src=$1
dest=$2
tag=$3
if [ 'a'$4'a' == 'aa' ]
then
  confFile=/home/hadoop/tools/deploy.conf
else
  confFile=$4
fi

if [ -f $confFile ]
then
  if [ -f $src ]
  then
```

```
    for server in `cat $confFile|grep -v '^#'|grep ','$tag','|awk -F',' '{pr
int $1}'`
    do
       scp $src $server":"${dest}
    done
  elif [ -d $src ]
  then
    for server in `cat $confFile|grep -v '^#'|grep ','$tag','|awk -F',' '{pr
int $1}'`
    do
       scp -r $src $server":"${dest}
    done
  else
     echo "Error: No source file exist"
  fi


else
  echo "Error: Please assign config file or run deploy.sh command with deplo
y.conf in same directory"
fi
```

查看一下 runRemoteCmd.sh 远程执行命令脚本内容。

```
[hadoop@djt11 tools]$ cat runRemoteCmd.sh
#!/bin/bash
#set -x


if [ $# -lt 2 ]
then
  echo "Usage: ./runRemoteCmd.sh Command MachineTag"
  echo "Usage: ./runRemoteCmd.sh Command MachineTag confFile"
  exit
fi


cmd=$1
tag=$2
if [ 'a'$3'a' == 'aa' ]
then
```

```
  confFile=/home/hadoop/tools/deploy.conf
else
  confFile=$3
fi


if [ -f $confFile ]
then
    for server in `cat $confFile|grep -v '^#'|grep ','$tag','|awk -F',' '{pr
int $1}'`
    do
       echo "******************$server**************************"
       ssh $server "source /etc/profile; $cmd"
    done
else
  echo "Error: Please assign config file or run deploy.sh command with deplo
y.conf in same directory"
fi
```

以上三个文件，方便我们搭建 hadoop 分布式集群。具体如何使用看后面如何操作。
如果我们想直接使用脚本，还需要给脚本添加执行权限。

```
[hadoop@djt11 tools]$ chmod u+x deploy.sh
[hadoop@djt11 tools]$ chmod u+x runRemoteCmd.sh
```

同时我们需要将/home/hadoop/tools 目录配置到 PATH 路径中。

```
[hadoop@djt11 tools]$ su root
Password:
[root@djt11 tools]# vi /etc/profile
PATH=/home/hadoop/tools:$PATH
export PATH
```

我们在 djt11 节点上，通过 runRemoteCmd.sh 脚本，一键创建所有节点的软件安装目录/home/hadoop/app。

```
[hadoop@djt11 tools]$ runRemoteCmd.sh "mkdir /home/hadoop/app" all
```

我们可以在所有节点查看到/home/hadoop/app 目录已经创建成功。

1.8 (七)jdk 安装

将本地下载好的 jdk1.7,上传至 djt11 节点下的/home/hadoop/app 目录。

```
[root@djt11 tools]# su hadoop
[hadoop@djt11 tools]$ cd /home/hadoop/app/
[hadoop@djt11 app]$ rz            //选择本地的下载好的 jdk-7u79-linux-x64.tar.gz
[hadoop@djt11 app]$ ls
jdk-7u79-linux-x64.tar.gz
[hadoop@djt11 app]$ tar zxvf jdk-7u79-linux-x64.tar.gz              //解压
[hadoop@djt11 app]$ ls
jdk1.7.0_79 jdk-7u79-linux-x64.tar.gz
[hadoop@djt11 app]$ rm jdk-7u79-linux-x64.tar.gz           //删除安装包
```

添加 jdk 环境变量。

```
[hadoop@djt11 app]$ su root
Password:
[root@djt11 app]# vi /etc/profile
JAVA_HOME=/home/hadoop/app/jdk1.7.0_79
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
PATH=$JAVA_HOME/bin:$PATH
export JAVA_HOME CLASSPATH PATH
[root@djt11 app]# source /etc/profile              //使配置文件生效
```

查看 jdk 是否安装成功。

```
[root@djt11 app]# java -version
java version "1.7.0_79"
Java(TM) SE Runtime Environment (build 1.7.0_79-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
```

出现以上结果就说明 djt11 节点上的 jdk 安装成功。

然后将 djt11 下的 jdk 安装包复制到其他节点上。

```
[hadoop@djt11 app]$ deploy.sh jdk1.7.0_79 /home/hadoop/app/ slave
```

djt12,djt13,djt14,djt15 节点重复 djt11 节点上的 jdk 配置即可。

1.9 (八)Zookeeper 安装

将本地下载好的 zookeeper-3.4.6.tar.gz 安装包，上传至 djt11 节点下的/home/hadoop/app 目录下。

```
[hadoop@djt11 app]$ rz            //选择本地下载好的 zookeeper-3.4.6.tar.gz
[hadoop@djt11 app]$ ls
jdk1.7.0_79 zookeeper-3.4.6.tar.gz
[hadoop@djt11 app]$ tar zxvf zookeeper-3.4.6.tar.gz                  //解压
[hadoop@djt11 app]$ ls
jdk1.7.0_79 zookeeper-3.4.6.tar.gz zookeeper-3.4.6
[hadoop@djt11 app]$ rm zookeeper-3.4.6.tar.gz              //删除 zookeeper-
3.4.6.tar.gz 安装包
[hadoop@djt11 app]$ mv zookeeper-3.4.6 zookeeper           //重命名
```

修改 Zookeeper 中的配置文件。

```
[hadoop@djt11 app]$ cd /home/hadoop/app/zookeeper/conf/
[hadoop@djt11 conf]$ ls
configuration.xsl  log4j.properties  zoo_sample.cfg
[hadoop@djt11 conf]$ cp zoo_sample.cfg zoo.cfg             //复制一个 zoo.cfg
文件
[hadoop@djt11 conf]$ vi zoo.cfg
dataDir=/home/hadoop/data/zookeeper/zkdata                 //数据文件目录
dataLogDir=/home/hadoop/data/zookeeper/zkdatalog           //日志目录
# the port at which the clients will connect
clientPort=2181          //默认端口号
#server.服务编号=主机名称：Zookeeper 不同节点之间同步和通信的端口：选举端口（选举 l
eader）
server.1=djt11:2888:3888
server.2=djt12:2888:3888
server.3=djt13:2888:3888
server.4=djt14:2888:3888
server.5=djt15:2888:3888
```

通过远程命令 deploy.sh 将 Zookeeper 安装目录拷贝到其他节点上面。

```
[hadoop@djt11 app]$ deploy.sh zookeeer /home/hadoop/app  slave
```

通过远程命令 runRemoteCmd.sh 在所有的节点上面创建目录：

```
[hadoop@djt11 app]$ runRemoteCmd.sh "mkdir -p /home/hadoop/data/zookeeper/z
kdata" all   //创建数据目录
[hadoop@djt11 app]$ runRemoteCmd.sh "mkdir -p /home/hadoop/data/zookeeper/z
kdatalog" all   //创建日志目录
```

然后分别在 djt11、djt12、djt13、djt14、djt15 上面，进入 zkdata 目录下，创建文件 myid，里面的内容分别填充为：1、2、3、4、5， 这里我们以 djt11 为例。

```
[hadoop@djt11 app]$ cd /home/hadoop/data/zookeeper/zkdata
[hadoop@djt11 zkdata]$ vi myid
1       //输入数字 1
```

配置 Zookeeper 环境变量。

```
[hadoop@djt11 zkdata]$ su root
Password:
[root@djt11 zkdata]# vi /etc/profile
JAVA_HOME=/home/hadoop/app/jdk1.7.0_79
ZOOKEEPER_HOME=/home/hadoop/app/zookeeper
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
PATH=$JAVA_HOME/bin:$ZOOKEEPER_HOME/bin:$PATH
export JAVA_HOME CLASSPATH PATH ZOOKEEPER_HOME
[root@djt11 zkdata]# source /etc/profile          //使配置文件生效
```

在 djt11 节点上面启动 Zookeeper。

```
[hadoop@djt11 zkdata]$ cd /home/hadoop/app/zookeeper/
[hadoop@djt11 zookeeper]$ bin/zkServer.sh start
[hadoop@djt11 zookeeper]$ jps
3633 QuorumPeerMain
[hadoop@djt11 zookeeper]$ bin/zkServer.sh stop             //关闭 Zookeeper
```

使用 runRemoteCmd.sh 脚本，启动所有节点上面的 Zookeeper。

```
runRemoteCmd.sh "/home/hadoop/app/zookeeper/bin/zkServer.sh start" zookeepe
r
```

查看所有节点上面的 QuorumPeerMain 进程是否启动。

```
runRemoteCmd.sh "jps" zookeeper
```

查看所有 Zookeeper 节点状态。

```
runRemoteCmd.sh "/home/hadoop/app/zookeeper/bin/zkServer.sh status" zookeep
er
```

如果一个节点为 leader，另四个节点为 follower，则说明 Zookeeper 安装成功。

1.10 (九)hadoop 集群环境搭建

将下载好的 apache hadoop-2.6.0.tar.gz 安装包，上传至 djt11 节点下的/home/hadoop/app

目录下。

```
[hadoop@djt11 app]$ rz    //将本地的 hadoop-2.6.0.tar.gz 安装包上传至当前目录
[hadoop@djt11 app]$ ls
hadoop-2.6.0.tar.gz jdk1.7.0_79  zookeeper
[hadoop@djt11 app]$ tar zxvf hadoop-2.6.0.tar.gz           //解压
[hadoop@djt11 app]$ ls
hadoop-2.6.0 hadoop-2.6.0.tar.gz jdk1.7.0_79  zookeeper
[hadoop@djt11 app]$ rm hadoop-2.6.0.tar.gz                 //删除安装包
[hadoop@djt11 app]$ mv hadoop-2.6.0 hadoop         //重命名
```

切换到/home/hadoop/app/hadoop/etc/hadoop/目录下，修改配置文件。

```
[hadoop@djt11 app]$ cd /home/hadoop/app/hadoop/etc/hadoop/
```

1.10.1 配置 HDFS

配置 hadoop-env.sh

```
[hadoop@djt11 hadoop]$ vi hadoop-env.sh
export JAVA_HOME=/home/hadoop/app/jdk1.7.0_79
```

配置 core-site.xml

```
[hadoop@djt11 hadoop]$ vi core-site.xml
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://cluster1</value>
        </property>
        < 这里的值指的是默认的 HDFS 路径 ，取名为 cluster1>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/hadoop/data/tmp</value>
        </property>
        < hadoop 的临时目录，如果需要配置多个目录，需要逗号隔开，data 目录需要我们自己创建>
        <property>
                <name>ha.zookeeper.quorum</name>
                <value>djt11:2181,djt12:2181,djt13:2181,djt14:2181,djt15:2181</value>
```

```
        </property>
        < 配置 Zookeeper 管理 HDFS>
</configuration>
```

配置 hdfs-site.xml

```
[hadoop@djt11 hadoop]$ vi hdfs-site.xml
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
    </property>
        < 数据块副本数为 3>
        <property>
                <name>dfs.permissions</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.permissions.enabled</name>
                <value>false</value>
        </property>
        < 权限默认配置为 false>
        <property>
                <name>dfs.nameservices</name>
                <value>cluster1</value>
        </property>
        < 命名空间，它的值与 fs.defaultFS 的值要对应，namenode 高可用之后有两个 namenode，cluster1 是对外提供的统一入口>
        <property>
                <name>dfs.ha.namenodes.cluster1</name>
                <value>djt11,djt12</value>
        </property>
        < 指定 nameService 是 cluster1 时的 nameNode 有哪些，这里的值也是逻辑名称，名字随便起，相互不重复即可>
        <property>
                <name>dfs.namenode.rpc-address.cluster1.djt11</name>
                <value>djt11:9000</value>
        </property>
        < djt11 rpc 地址>
        <property>
```

```
                <name>dfs.namenode.http-address.cluster1.djt11</name>
                <value>djt11:50070</value>
        </property>
        < djt11 http 地址>
        <property>
                <name>dfs.namenode.rpc-address.cluster1.djt12</name>
                <value>djt12:9000</value>
        </property>
        < djt12 rpc 地址>
        <property>
                <name>dfs.namenode.http-address.cluster1.djt12</name>
                <value>djt12:50070</value>
        </property>
        < djt12 http 地址>
        <property>
                <name>dfs.ha.automatic-failover.enabled</name>
                <value>true</value>
    </property>
        < 启动故障自动恢复>
        <property>
                <name>dfs.namenode.shared.edits.dir</name>
                <value>qjournal://djt11:8485;djt12:8485;djt13:8485;djt14:8
485;djt15:8485/cluster1</value>
        </property>
        < 指定 journal>
        <property>
                <name>dfs.client.failover.proxy.provider.cluster1</nam
e>
                <value>org.apache.hadoop.hdfs.server.namenode.ha.Configure
dFailoverProxyProvider</value>
    </property>
        < 指定 cluster1 出故障时，哪个实现类负责执行故障切换>
    <property>
                <name>dfs.journalnode.edits.dir</name>
                <value>/home/hadoop/data/journaldata/jn</value>
    </property>
        < 指定 JournalNode 集群在对 nameNode 的目录进行共享时，自己存储数据的磁
盘路径 >
        <property>
                <name>dfs.ha.fencing.methods</name>
```

```
                <value>shell(/bin/true)</value>
    </property>
    <property>
        <name>dfs.ha.fencing.ssh.private-key-files</name>
        <value>/home/hadoop/.ssh/id_rsa</value>
    </property>
         <property>
        <name>dfs.ha.fencing.ssh.connect-timeout</name>
        <value>10000</value>
    </property>
         < 脑裂默认配置>
    <property>
                <name>dfs.namenode.handler.count</name>
                <value>100</value>
    </property>
</configuration>
```

配置 slave

```
[hadoop@djt11 hadoop]$ vi slaves
djt13
djt14
djt15
```

向所有节点分发 hadoop 安装包。

```
[hadoop@djt11 app]$ deploy.sh hadoop /home/hadoop/app/ slave
```

1.10.2 hdfs 配置完毕后启动顺序

1、启动所有节点上面的 Zookeeper 进程

```
[hadoop@djt11 hadoop]$ runRemoteCmd.sh "/home/hadoop/app/zookeeper/bin/zkSe
rver.sh start" zookeeper
```

2、启动所有节点上面的 journalnode 进程

```
[hadoop@djt11 hadoop]$ runRemoteCmd.sh "/home/hadoop/app/hadoop/sbin/hadoop
-daemon.sh start journalnode" all
```

3、首先在主节点上(比如,djt11)执行格式化

```
[hadoop@djt11 hadoop]$ bin/hdfs namenode -format / /namenode 格式化
[hadoop@djt11 hadoop]$ bin/hdfs zkfc -formatZK //格式化高可用
[hadoop@djt11 hadoop]$bin/hdfs namenode //启动 namenode
```

4、与此同时，需要在备节点（比如，djt12）上执行数据同步

```
[hadoop@djt12 hadoop]$ bin/hdfs namenode -bootstrapStandby //同步主节点和备节点之间的元数据
```

5、djt12 同步完数据后，紧接着在 djt11 节点上，按下 ctrl+c 来结束 namenode 进程。 然后关闭所有节点上面的 journalnode 进程

```
[hadoop@djt11 hadoop]$ runRemoteCmd.sh "/home/hadoop/app/hadoop/sbin/hadoop-daemon.sh stop journalnode" all  //然后停掉各节点的 journalnode
```

6、如果上面操作没有问题，我们可以一键启动 hdfs 所有相关进程

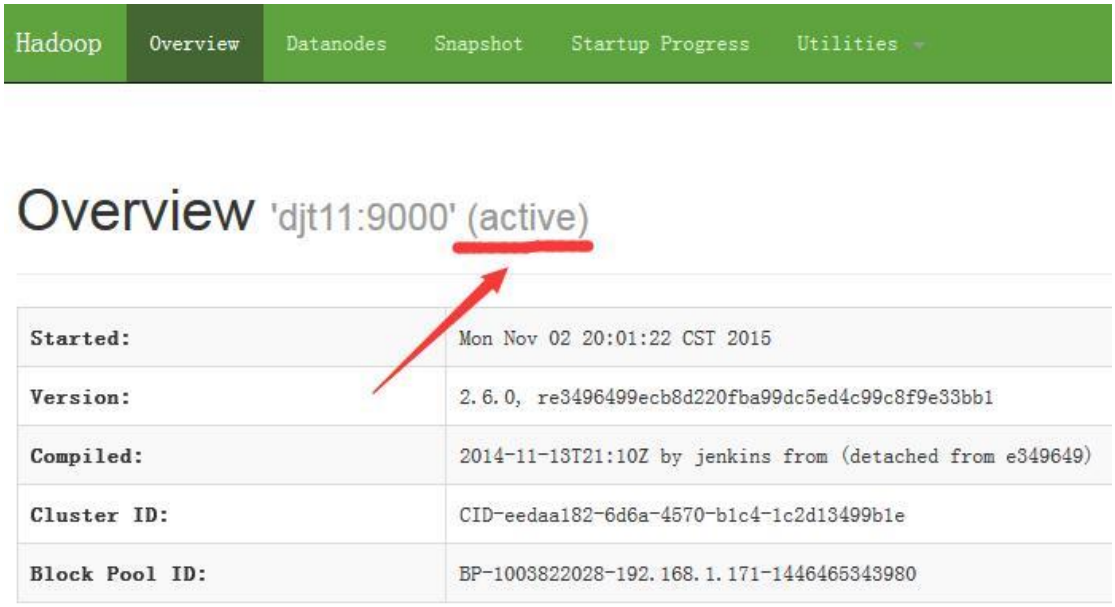
```
[hadoop@djt11 hadoop]$ sbin/start-dfs.sh
```

启动成功之后，关闭其中一个 namenode ，然后在启动 namenode 观察切换的状况。

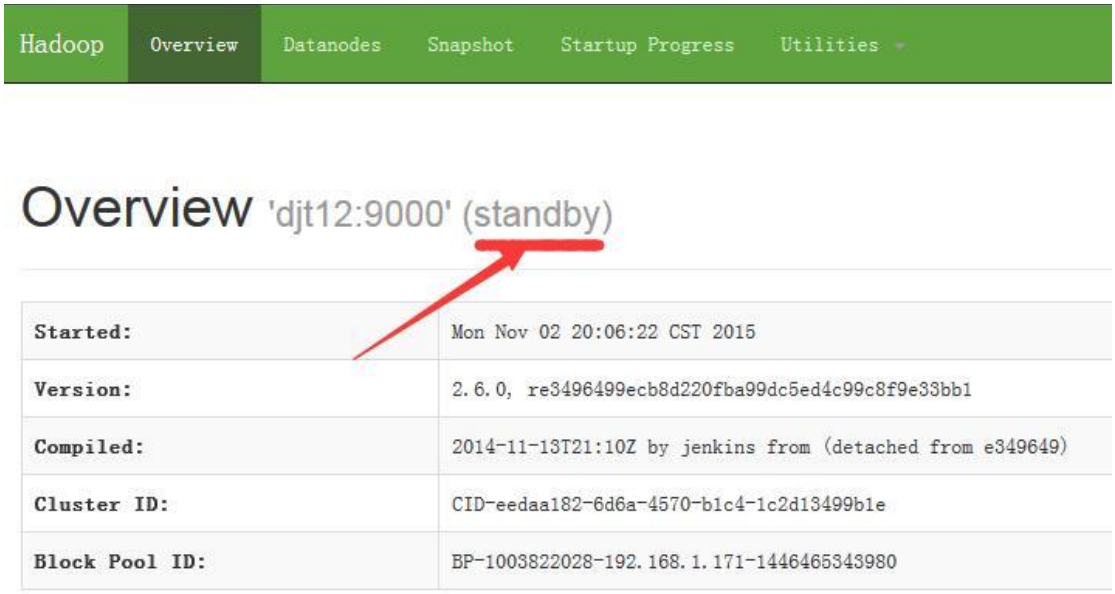
7、验证是否启动成功

通过 web 界面查看 namenode 启动情况。

```
http://djt11:50070
```




```
http://djt12:50070
```



上传文件至 hdfs

```
[hadoop@djt11 hadoop]$ vi djt.txt //本地创建一个 djt.txt 文件
hadoop dajiangtai
hadoop dajiangtai
hadoop dajiangtai
[hadoop@djt11 hadoop]$ hdfs dfs -mkdir /test   //在 hdfs 上创建一个文件目录
[hadoop@djt11 hadoop]$ hdfs dfs -put djt.txt /test          //向 hdfs 上传一个文件
[hadoop@djt11 hadoop]$ hdfs dfs -ls /test //查看 djt.txt 是否上传成功
```

如果上面操作没有问题说明 hdfs 配置成功。

1.10.3 YARN 安装配置

配置 mapred-site.xml

```
[hadoop@djt11 hadoop]$ vi mapred-site.xml
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <指定运行 mapreduce 的环境是 Yarn，与 hadoop1 不同的地方>
</configuration>
```

配置 yarn-site.xml

```
[hadoop@djt11 hadoop]$ vi yarn-site.xml
<configuration>
<property>
        <name>yarn.resourcemanager.connect.retry-interval.ms</name>
        <value>2000</value>
</property>
< 超时的周期>
<property>
        <name>yarn.resourcemanager.ha.enabled</name>
        <value>true</value>
</property>
< 打开高可用>
<property>
        <name>yarn.resourcemanager.ha.automatic-failover.enabled</name>
        <value>true</value>
</property>
<启动故障自动恢复>
<property>
        <name>yarn.resourcemanager.ha.automatic-failover.embedded</name>
        <value>true</value>
</property>

<property>
        <name>yarn.resourcemanager.cluster-id</name>
        <value>yarn-rm-cluster</value>
</property>
<给 yarn cluster 取个名字 yarn-rm-cluster>
<property>
        <name>yarn.resourcemanager.ha.rm-ids</name>
        <value>rm1,rm2</value>
</property>
<给 ResourceManager 取个名字 rm1,rm2>
<property>
        <name>yarn.resourcemanager.hostname.rm1</name>
        <value>djt11</value>
</property>
<配置 ResourceManager rm1 hostname>
```

```
<property>
        <name>yarn.resourcemanager.hostname.rm2</name>
        <value>djt12</value>
</property>
<配置 ResourceManager  rm2  hostname>
<property>
        <name>yarn.resourcemanager.recovery.enabled</name>
        <value>true</value>
</property>
<启用 resourcemanager  自动恢复>
<property>
        <name>yarn.resourcemanager.zk.state-store.address</name>
        <value>djt11:2181,djt12:2181,djt13:2181,djt14:2181,djt15:2181</valu
e>
</property>
<配置 Zookeeper 地址>
<property>
        <name>yarn.resourcemanager.zk-address</name>
        <value>djt11:2181,djt12:2181,djt13:2181,djt14:2181,djt15:2181</valu
e>
</property>
<配置 Zookeeper 地址>
<property>
        <name>yarn.resourcemanager.address.rm1</name>
        <value>djt11:8032</value>
</property>
< rm1 端口号>
<property>
        <name>yarn.resourcemanager.scheduler.address.rm1</name>
        <value>djt11:8034</value>
</property>
< rm1 调度器的端口号>
<property>
        <name>yarn.resourcemanager.webapp.address.rm1</name>
        <value>djt11:8088</value>
</property>
< rm1  webapp 端口号>
<property>
        <name>yarn.resourcemanager.address.rm2</name>
        <value>djt12:8032</value>
```

```
</property>
< rm2 端口号>
<property>
        <name>yarn.resourcemanager.scheduler.address.rm2</name>
        <value>djt12:8034</value>
</property>
< rm2 调度器的端口号>
<property>
        <name>yarn.resourcemanager.webapp.address.rm2</name>
        <value>djt12:8088</value>
</property>
< rm2 webapp 端口号>
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
</property>
<property>
        <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</nam
e>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<执行 MapReduce 需要配置的 shuffle 过程>
</configuration>
```

启动 YARN

1、在 djt11 节点上执行。

```
[hadoop@djt11 hadoop]$ sbin/start-yarn.sh
```

2、在 djt12 节点上面执行。

```
[hadoop@djt11 hadoop]$ sbin/yarn-daemon.sh start resourcemanager
```

同时打开一下 web 界面。

```
http://djt11:8088
http://djt12:8088
```

关闭其中一个 resourcemanager，然后再启动，看看这个过程的 web 界面变化。

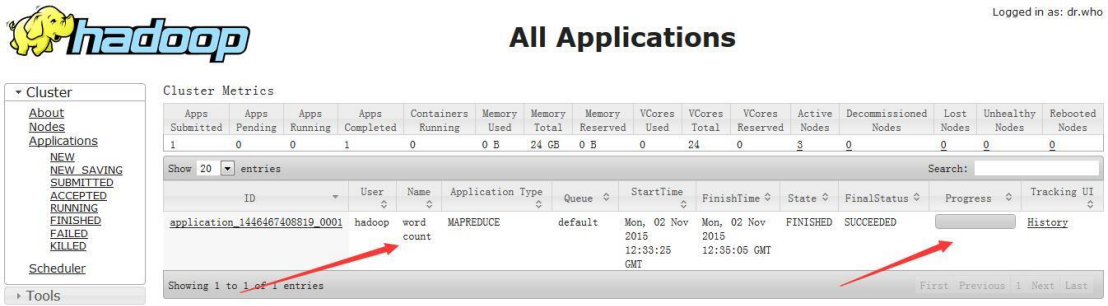
3、检查一下 ResourceManager 状态

```
[hadoop@djt11 hadoop]$ bin/yarn rmadmin -getServiceState rm1

[hadoop@djt11 hadoop]$ bin/yarn rmadmin -getServiceState rm2
```

4、Wordcount 示例测试

```
[hadoop@djt11 hadoop]$ hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /test/djt.txt /test/out/
```



如果上面执行没有异常，说明 YARN 安装成功。

hadoop 分布式集群安装，相关的 jar 包、脚本、配置文件都可以下载哦，具体参考：
http://www.dajiangtai.com/course/27.do?origin=invite-register&code=a8EYNf
至此，hadoop 分布式集群搭建完毕。