[大数据cdh5.3.6集群搭建手册 3](#_Toc8270)

[1环境 3](#_Toc29637)

[2软件清单 3](#_Toc29420)

[3文件目录 3](#_Toc15877)

[4集群规划 3](#_Toc9089)

[5虚拟机相关配置 4](#_Toc32101)

[5.1修改主机名： 4](#_Toc29855)

[5.2 Ip与主机名映射 4](#_Toc21613)

[5.3 配置固定Ip 4](#_Toc10245)

[5.4禁用IPv6 4](#_Toc18918)

[5.5新增普通用户 4](#_Toc27956)

[5.6配置普通用户的sudo权限 5](#_Toc2989)

[5.7防火墙 5](#_Toc28039)

[5.8禁用Selinux 5](#_Toc19668)

[5.9设置文件打开数量和用户最大进程数 5](#_Toc10832)

[5.10 SSH免密登录 5](#_Toc16868)

[5.11 SSH传输文件到另一条机器 5](#_Toc17536)

[5.12统一集群时间 5](#_Toc5200)

[6软件安装 6](#_Toc17124)

[6.1安装jdk-7u67-linux-x64.tar.gz 6](#_Toc11857)

[6.2安装集群zookeeper-3.4.5-cdh5.3.6.tar.gz 7](#_Toc21247)

[6.3安装集群hadoop-2.5.0-cdh5.3.6.tar.gz 8](#_Toc2752)

[6.3.1 HA高可用性配置 8](#_Toc8716)

[6.3.2 ResourcesManager高可用性 14](#_Toc20654)

[6.4安装集群hbase-0.98.6-cdh5.3.6.tar.gz 17](#_Toc5801)

[6.5安装集群hive-0.13.1-cdh5.3.6.tar.gz 19](#_Toc197)

[6.6安装sqoop-1.4.5-cdh5.3.6.tar.gz 21](#_Toc14647)

[6.7安装flume-ng-1.5.0-cdh5.3.6.tar.gz 23](#_Toc27327)

[6.8安装oozie-4.0.0-cdh5.3.6.tar.gz 26](#_Toc18430)

[6.9安装集群kafka\_2.10-0.8.2.1.tgz 27](#_Toc4974)

[6.10安装Oozie-4.0.0-cdh5.3.6.tar.gz 28](#_Toc7561)

[6.11安装Mysql5.6.24 29](#_Toc14378)

[6.12安装集群Nginx+Keepalived+Lvs 30](#_Toc734)

[6.12.1 Nginx配置 30](#_Toc17067)

[6.11.2 Keepalived配置 31](#_Toc5245)

[6.13安装Spark-1.5.1-bin-hadoop2.4.tgz 35](#_Toc22665)

[7 大数据框架简介 36](#_Toc22603)

[7.1 Hadoop基本语法及操作 36](#_Toc19460)

[7.1.1 Hadoop Hdfs 36](#_Toc21455)

[7.1.2 MapReduce 37](#_Toc8126)

[7.2 Hbase基本语法及操作  39](#_Toc11632)

[7.3 Spark基本语法及操作 43](#_Toc7562)

[7.4 Zookeeper基本语法及操作 44](#_Toc30873)

[7.4.1基本语法 44](#_Toc16518)

# 大数据cdh5.3.6集群搭建手册

作者：王雪强 QQ:592470261

# 1环境

Centos6 64、Window 64、JDK1.7.0\_67、Xftp5、Xshell5、WMware10

在新的或者没有开启虚拟化电脑中，需要注意先将其开启详情参考（https://jingyan.baidu.com/article/8ebacdf0df465b49f65cd5d5.html）。

# **2软件清单**

下载地址为：http://archive.cloudera.com/cdh5/cdh/5/

* hadoop-2.5.0-cdh5.3.6.tar.gz
* hbase-0.98.6-cdh5.3.6.tar.gz
* hive-0.13.1-cdh5.3.6.tar.gz
* flume-ng-1.5.0-cdh5.3.6.tar.gz
* sqoop-1.4.5-cdh5.3.6.tar.gz
* flume-ng-1.5.0-cdh5.3.6.tar.gz
* jdk-7u67-linux-x64.tar.gz
* zookeeper-3.4.5-cdh5.3.6.tar.gz

# **3文件目录**

* /opt/datas 数据存放文件夹
* /opt/cdh5.3.6 大数据相关软件
* /opt/modules 其他软件安装目录
* /opt/software 应用软件

# **4集群规划**

192.168.174.135 bigdata-cdh01.ibeifeng.com bigdata-cdh01

192.168.174.136 bigdata-cdh02.ibeifeng.com bigdata-cdh02

192.168.174.137 bigdata-cdh03.ibeifeng.com bigdata-cdh03

# **5虚拟机相关配置**

## **5.1修改主机名：**

# vi /etc/sysconfig/network

# hostname bigdata-cdh01.ibeifeng.com

## **5.2 Ip与主机名映射**

# vi /etc/hosts

127.0.0.1 localhost.localdomain localhost

192.168.174.135 bigdata-cdh01.ibeifeng.com bigdata-cdh01

192.168.174.136 bigdata-cdh02.ibeifeng.com bigdata-cdh02

192.168.174.137 bigdata-cdh03.ibeifeng.com bigdata-cdh03

## **5.3 配置固定Ip**

# vi /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE="eth0"

BOOTPROTO="static"

HWADDR="00:0C:29:3B:7E:81"

IPV6INIT="yes"

NM\_CONTROLLED="yes"

ONBOOT="yes"

TYPE="Ethernet"

UUID="1edd9c83-9df9-4d64-9c58-d5e23ffd02f4"

IPADDR="192.168.174.128"

NETMASK="255.255.255.0"

GATEWAY="192.168.174.1"

## **5.4禁用IPv6**

# echo "alias net-pf-10 off" >> /etc/modprobe.d/dist.conf

# echo "alias ipv6 off" >> /etc/modprobe.d/dist.conf

## **5.5新增普通用户**

# adduser beifeng

# passwd beifeng

## **5.6配置普通用户的sudo权限**

# chmod u+w /etc/sudoers

# vi /etc/sudoers

添加内容：beifeng ALL=(root)NOPASSWD:ALL

## **5.7防火墙**

# service iptables stop

# chkconfig iptables off

## **5.8禁用Selinux**

# vi /etc/sysconfig/selinux

添加内容：SELINUX=disabled

## **5.9设置文件打开数量和用户最大进程数**

# vi /etc/security/limits.conf

尾部添加内容：

\* soft nofile 65535

\* hard nofile 65535

\* soft nproc 32000

\* hard nproc 32000

## **5.10 SSH免密登录**

# ssh-keygen

# ssh-copy-id bigdata-cdh01.ibeifeng.com

# ssh-copy-id bigdata-cdh02.ibeifeng.com

# ssh-copy-id bigdata-cdh03.ibeifeng.com

## **5.11 SSH传输文件到另一条机器**

# scp -r modules [root@bigdata-cdh02.ibeifeng.com:/opt/modules](mailto:root@bigdata-cdh02.ibeifeng.com:/opt/modules)

## **5.12统一集群时间**

选用bigdata-cdh01.ibeifeng.com为时间服务器

# vi /etc/ntp.conf

打开

restrict 192.168.174.0 mask 255.255.255.0 nomodify notrap

注释

#server 0.centos.pool.ntp.org iburst

#server 1.centos.pool.ntp.org iburst

#server 2.centos.pool.ntp.org iburst

#server 3.centos.pool.ntp.org iburst

打开

server 127.127.1.0 # local clock

fudge 127.127.1.0 stratum 10

# vi /etc/sysconfig/ntpd

添加SYNC\_HWCLOCK=yes

启动ntp

# chkconfig ntpd on

# chkconfig --list|grep ntpd

在另外两台机器操作

# crontab -e

# 0-59/10 \* \* \* \* /usr/sbin/ntpdate bigdata-cdh01.ibeifeng.com

# **6软件安装**

## 6.1安装jdk-7u67-linux-x64.tar.gz

卸载默认jdk

# rpm -qa|grep java

# rpm -e --nodeps xxx yyy zzz

安装：

# chmod u+x jdk-7u67-linux-x64.tar.gz

# tar -zxvf jdk-7u67-linux-x64.tar.gz -C /opt/modules

配置环境变量：

# vi /etc/profile

添加内容

###### JAVA\_HOME

export JAVA\_HOME=/opt/modules/jdk1.7.0\_67

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

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

## 6.2安装集群zookeeper-3.4.5-cdh5.3.6.tar.gz

# chmod u+x zookeeper-3.4.5-cdh5.3.6.tar.gz

# tar -zxvf zookeeper-3.4.5-cdh5.3.6.tar.gz -C /opt/cdh5.3.6

修改conf下配置文件

# cp zoo\_sample.cfg zoo.cfg

修改 zoo.cfg

# vi zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/opt/cdh5.3.6/zookeeper-3.4.5-cdh5.3.6/data

dataLogDir=/opt/cdh5.3.6/zookeeper-3.4.5-cdh5.3.6/logs

clientPort=2181

server.1=bigdata-cdh01.ibeifeng.com:2888:3888

server.2=bigdata-cdh02.ibeifeng.com:2888:3888

server.3=bigdata-cdh03.ibeifeng.com:2888:3888

切换到/opt/cdh5.3.6/zookeeper-3.4.5-cdh5.3.6/data

# vi myid

输入对应的序号 1

依次启动

# bin/zkServer.sh start

## 6.3安装集群hadoop-2.5.0-cdh5.3.6.tar.gz

# chmod u+x hadoop-2.5.0-cdh5.3.6.tar.gz

# tar -zxvf hadoop-2.5.0-cdh5.3.6.tar.gz -C /opt/cdh5.3.6

切换到目录/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/etc/hadoop修改配置文件

### 6.3.1 HA高可用性配置

#### **core-site.xml**

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://ns1</value>

</property>

<property>

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

<value>/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/data/tmp</value>

</property>

<property>

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

<value>bigdata-cdh01.ibeifeng.com:2181,

bigdata-cdh02.ibeifeng.com:2181,

bigdata-cdh03.ibeifeng.com:2181

</value>

</property>

<property>

<name>fs.trash.interval</name>

<value>420</value>

</property>

</configuration>

#### **hdfs-site.xml**

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>dfs.nameservices</name>

<value>ns1</value>

</property>

<property>

<name>dfs.ha.namenodes.ns1</name>

<value>nn1,nn2</value>

</property>

<!-- ###NameNode RPC ADDRESS### -->

<property>

<name>dfs.namenode.rpc-address.ns1.nn1</name>

<value>bigdata-cdh01.ibeifeng.com:8020</value>

</property>

<property>

<name>dfs.namenode.rpc-address.ns1.nn2</name>

<value>bigdata-cdh02.ibeifeng.com:8020</value>

</property>

<!-- ###NameNode HTTP WEB ADDRESS### -->

<property>

<name>dfs.namenode.http-address.ns1.nn1</name>

<value>bigdata-cdh01.ibeifeng.com:50070</value>

</property>

<property>

<name>dfs.namenode.http-address.ns1.nn2</name>

<value>bigdata-cdh02.ibeifeng.com:50070</value>

</property>

<!-- ###NameNode SHARED EDITS ADDRESS### -->

<property>

<name>dfs.namenode.shared.edits.dir</name>

<value>qjournal://bigdata-cdh01.ibeifeng.com:8485;bigdata-cdh02.ibeifeng.com:8485;bigdata-cdh03.ibeifeng.com:8485/ns1</value>

</property>

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/opt/hadoop-2.5.0-cdh5.3.6/data/dfs/jn</value>

</property>

<!-- ###HDFS PROXY CLIENT### -->

<property>

<name>dfs.client.failover.proxy.provider.ns1</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

<!-- ### NameNodes FENCE ### -->

<property>

<name>dfs.ha.fencing.methods</name>

<value>sshfence</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_rsa</value>

</property>

<!-- ### automatic failover ### -->

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

</configuration>

#### **mapred-site.xml**

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

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

<value>yarn</value>

</property>

<!-- 配置 MapReduce JobHistory Server 地址 ，默认端口10020 -->

<property>

<name>mapreduce.jobhistory.address</name>

<value>0.0.0.0:10020</value>

</property>

<!-- 配置 MapReduce JobHistory Server web ui 地址， 默认端口19888 -->

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>0.0.0.0:19888</value>

</property>

</configuration>

#### **slaves**

bigdata-cdh01.ibeifeng.com

bigdata-cdh02.ibeifeng.com

bigdata-cdh03.ibeifeng.com

**其他文件**

另外需要在hadoop-env.sh、mapred-env.sh、yarn-env.sh文件中指定jdk版本。

export JAVA\_HOME=/opt/modules/jdk1.7.0\_6

#### **格式化namenode**

#cd /opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

#mkdir -p /opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/data/dfs/jn/ns1/current

#sbin/hadoop-daemon.sh start journalnode

启动第一个namenode节点

#bin/hdfs namenode -format

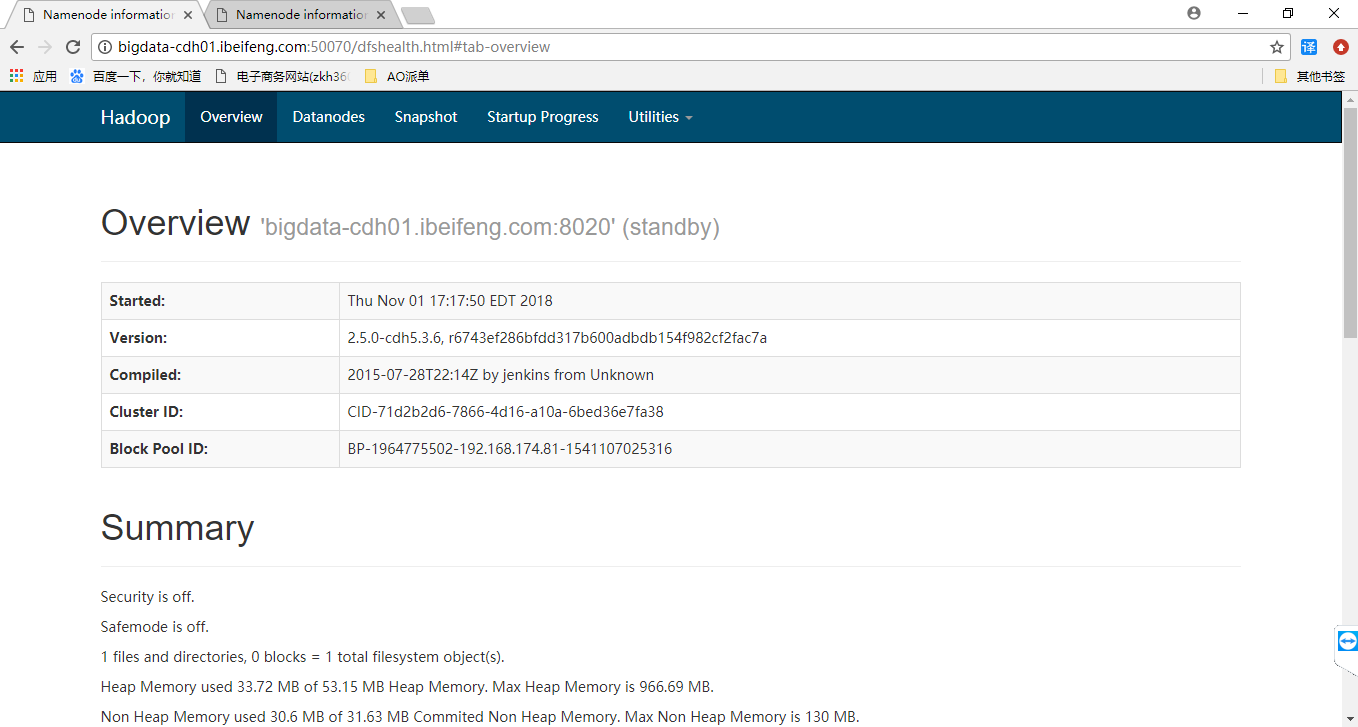
#sbin/hadoop-daemon.sh start namenode

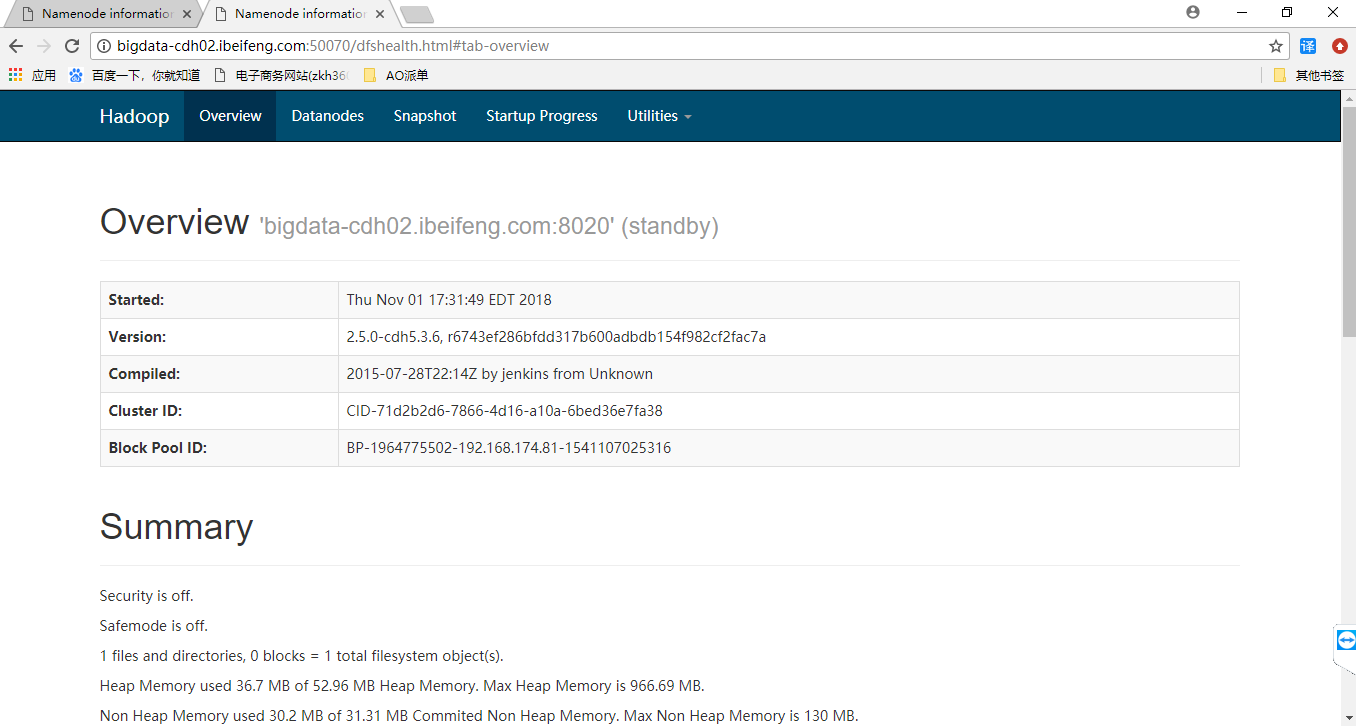
启动第二个namenode节点之前需同步第一个在启动

# bin/hdfs namenode -bootstrapStandby

#sbin/hadoop-daemon.sh start namenode

两个namenode启动完后状态都是standby需要指定一个为active





# bin/hdfs haadmin -transitionToActive nn2

执行此命令如果报错如下：

Automatic fail overisen abled for NameNode at xxx

Refusing to manually manage HA state, since it may cause

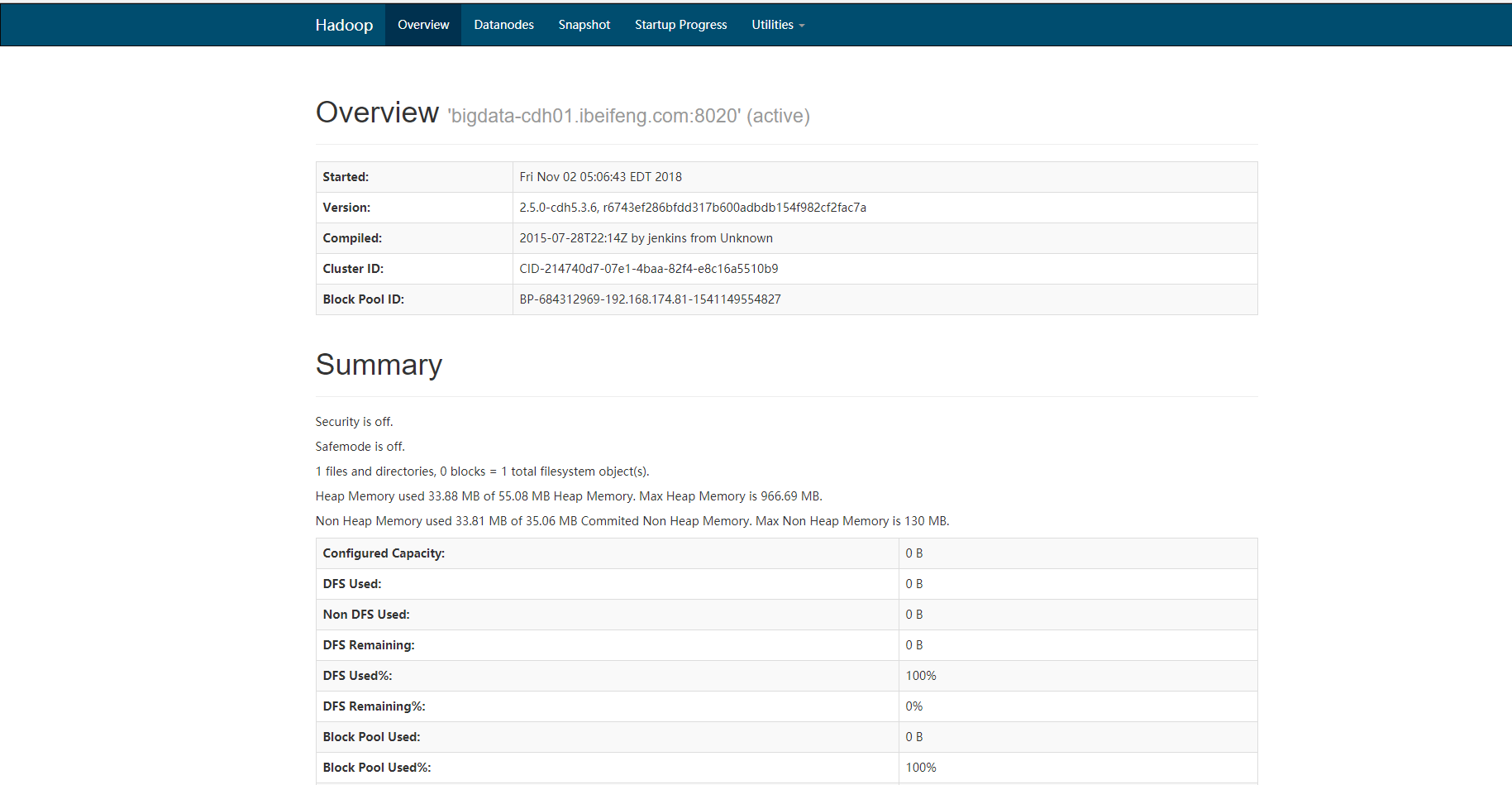
a split-brain scenario or other incorrect state.

If you are very sure you know what you are doing, please

specify the forcemanual flag.

执行命令

# bin/hdfs haadmin -transitionToActive --forcemanual nn1



第三台不需要namenode

最后启动datanode

# sbin/hadoop-daemon.sh start datanode

datanode启动失败：原因多次format namenode导致解决办法删除tmp/dfs/data下的current文件夹，然后重新启动hadoop。

#### **HA故障自动转移**

启动zookeeper集群

#/opt/cdh5.3.6/zookeeper-3.4.5-cdh5.3.6/bin/zkServer.sh start

第一次需初始化HA在zookeeper的状态

#/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/bin/hdfs zkfc -formatZK

启动HDFS服务

#/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/sbin/start-dfs.sh

验证

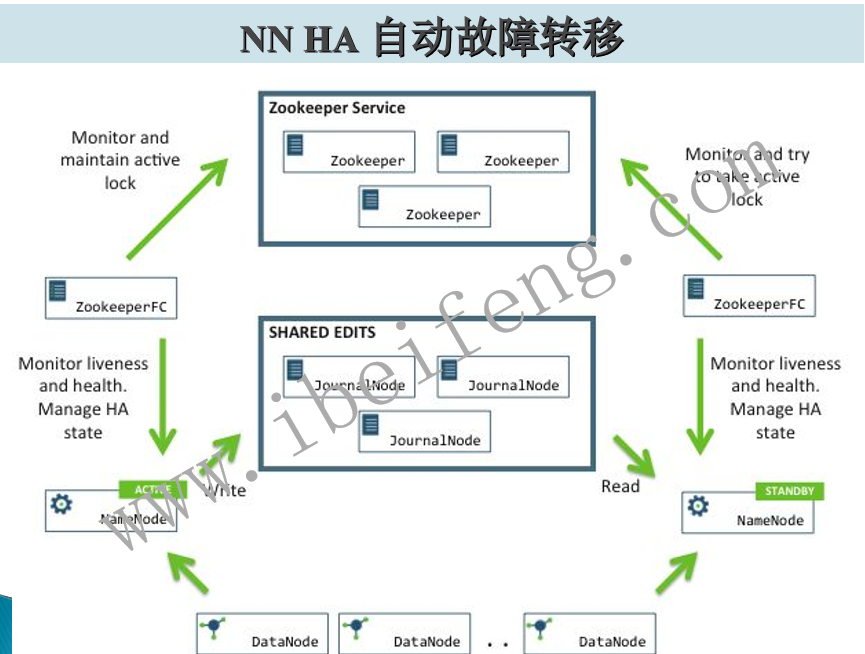
将Active NameNode进程kill

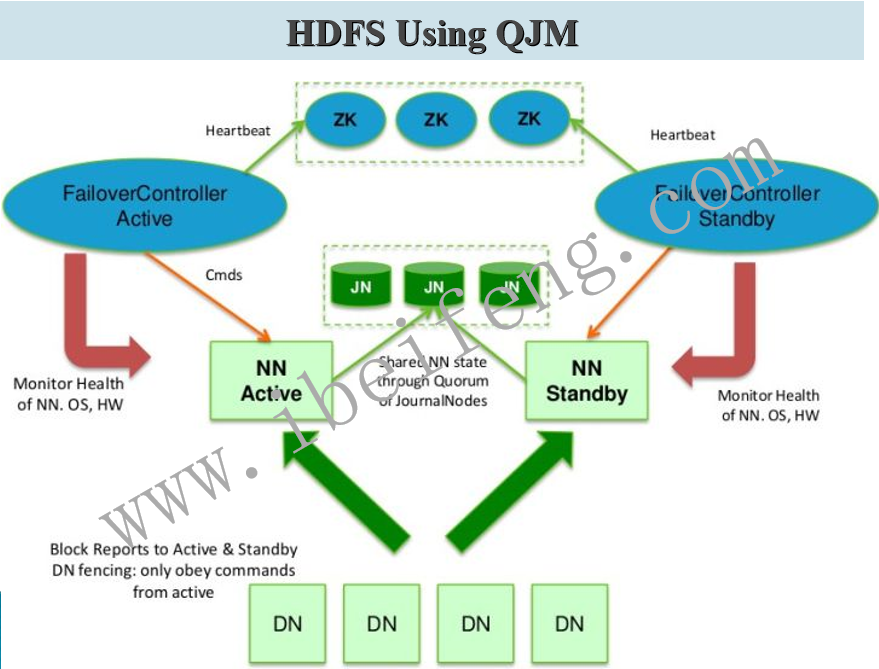
# jps

# kill -9 pid

或者使机器断网

# service net work stop





#### **启动方式**

逐一启动

#cd /opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

#sbin/hadoop-daemon.sh start namenode

#sbin/hadoop-daemon.sh start datanode

分类启动

#sbin/start-dfs.sh

一键启动

#sbin/start-all.sh

日志服务单独启用

#sbin/mr-jobhistory-daemon.sh start historyserver

### 6.3.2 ResourcesManager高可用性

#### **RM故障转移**

ResourceManager HA通过主动/备用架构实现 - 在任何时间点，其中一个RM是活动的，并且一个或多个RM处于待机模式，等待接管，如果活动发生任何事情。转换到活动的触发器来自管理员（通过CLI）或通过集成的故障转移控制器（当启用自动故障转移时）。

#### **手动转换和故障转移**

当未启用自动故障转移时，管理员必须手动将其中一个RM转换为活动。要从一个RM故障切换到另一个RM，需要首先将Active-RM转换为Standby并将Standby-RM转换为Active。所有这一切都可以使用“ yarn rmadmin ”CLI 来完成。

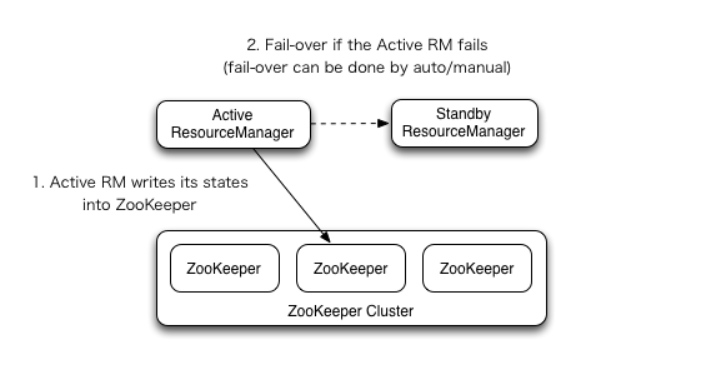
#### **自动故障转移**

RM可以选择嵌入基于Zookeeper的ActiveStandbyElector来决定哪个RM应该是活动的。当活动下降或变得无响应时，另一RM被自动选择为活动，然后接管。注意，不需要像HDFS那样运行单独的ZKFC守护程序，因为嵌入在RM中的ActiveStandbyElector用作故障检测器和领导选举人，而不是单独的ZKFC定义。

#### **客户端，ApplicationMaster和NodeManager在RM故障转移**

当有多个RM时，客户端和节点使用的配置（yarn-site.xml）应该列出所有的RM。客户端，ApplicationMasters（AM）和NodeManager（NM）尝试以循环方式连接到RM，直到它们达到Active RM。如果活动下降，他们恢复轮询轮询，直到他们击中“新”活动。此默认重试逻辑实现为org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider。您可以通过实现org.apache.hadoop.yarn.client.RMFailoverProxyProvider并将yarn.client.failover-proxy-provider的值设置为类名来覆盖逻辑。

在启用[ResourceManger重新启动](https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html" \t "https://blog.csdn.net/zhongguozhichuang/article/details/_blank)后，升级到活动状态的RM将加载RM内部状态，并根据RM重新启动功能，尽可能继续从上次活动停止的位置开始操作。为先前提交给RM的每个托管应用程序生成新尝试。应用程序可以定期检查点，以避免丢失任何工作。状态存储必须从Active / Standby RM两个可见。目前，有两个用于持久化的RMStateStore实现 - FileSystemRMStateStore和ZKRMStateStore。该ZKRMStateStore隐式地允许写在任何时间点访问单个RM，因此是推荐的商店HA群集使用。当使用ZKRMStateStore时，不需要单独的屏蔽机制来解决潜在的裂脑情况，其中多个RM可以潜在地承担活动角色。当使用ZKRMStateStore时，建议不要在Zookeeper集群上设置“ zookeeper.DigestAuthenticationProvider.superDigest ”属性，以确保zookeeper管理员无权访问YARN应用程序/用户凭据信息。



#### **恢复以前活跃的RM状态**

在启用[ResourceManger重新启动](https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html" \t "https://blog.csdn.net/zhongguozhichuang/article/details/_blank)后，升级到活动状态的RM将加载RM内部状态，并根据RM重新启动功能，尽可能继续从上次活动停止的位置开始操作。为先前提交给RM的每个托管应用程序生成新尝试。应用程序可以定期检查点，以避免丢失任何工作。状态存储必须从Active / Standby RM两个可见。目前，有两个用于持久化的RMStateStore实现 - FileSystemRMStateStore和ZKRMStateStore。该ZKRMStateStore隐式地允许写在任何时间点访问单个RM，因此是推荐的商店HA群集使用。当使用ZKRMStateStore时，不需要单独的屏蔽机制来解决潜在的裂脑情况，其中多个RM可以潜在地承担活动角色。当使用ZKRMStateStore时，建议不要在Zookeeper集群上设置“ zookeeper.DigestAuthenticationProvider.superDigest ”属性，以确保zookeeper管理员无权访问YARN应用程序/用户凭据信息。

#### **示例配置yarn-site.xml**

<property>

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

<value>true</value>

</property>

<property>

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

<value>cluster1</value>

</property>

<property>

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

<value>rm1,rm2</value>

</property>

<property>

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

<value>bigdata-cdh01.ibeifeng.com</value>

</property>

<property>

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

<value>bigdata-cdh02.ibeifeng.com</value>

</property>

<property>

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

<value>bigdata-cdh01.ibeifeng.com:2181,bigdata-cdh02.ibeifeng.com:2181,bigdata-cdh03.ibeifeng.com:2181</value>

</property>

#### **管理命令**

yarn rmadmin具有几个HA特定的命令选项，用于检查RM的运行状况/状态，并转换到Active / Standby。HA的命令采用由yarn.resourcemanager.ha.rm-ids作为参数设置的RM的服务标识。

# /opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/sbin/start-yarn.sh

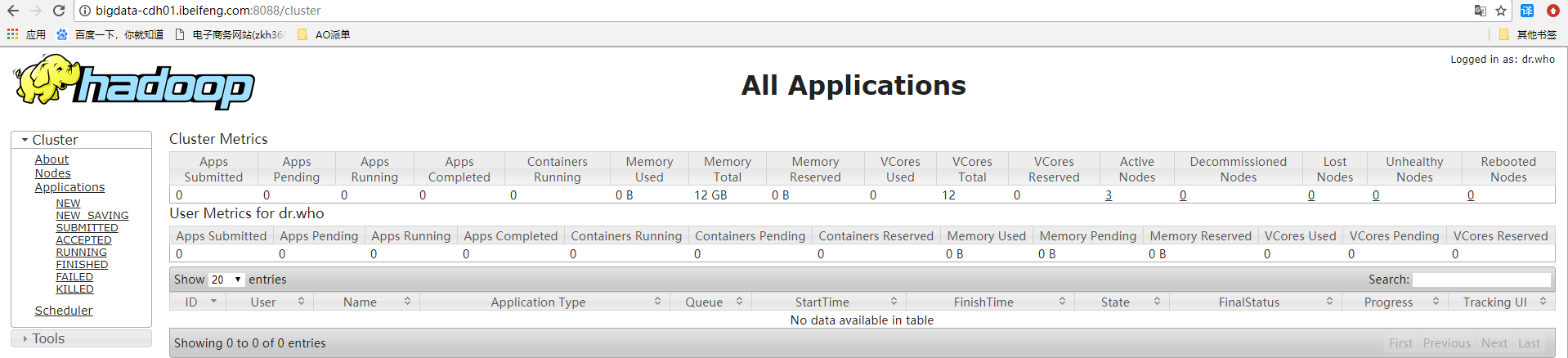
# yarn rmadmin -getServiceState rm1

# yarn rmadmin -getServiceState rm2

如果启用了自动故障转移，则不能使用手动转换命令。虽然你可以通过-forcemanual标志覆盖它，你需要谨慎。

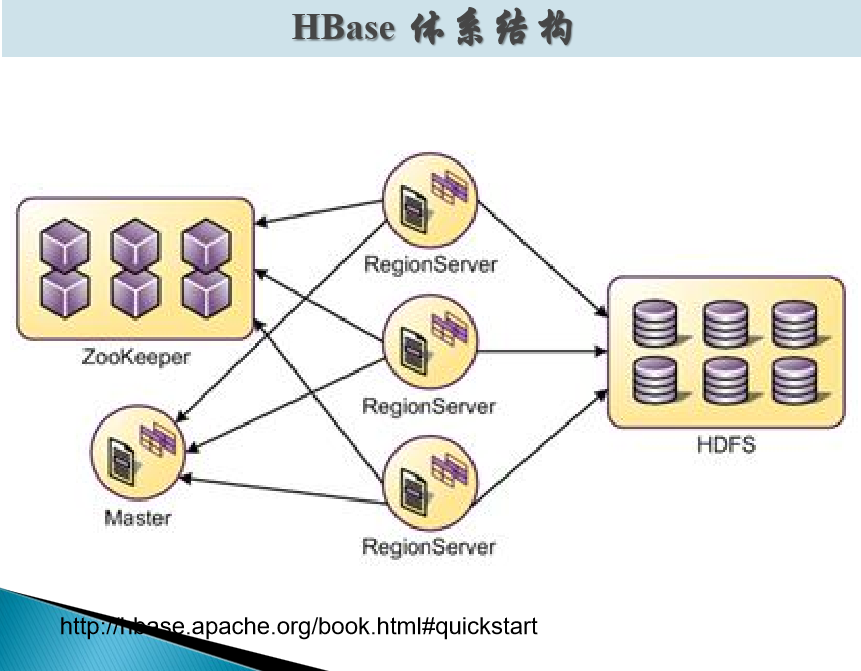
# yarn rmadmin -transitionToStandby rm1

对org.apache.hadoop.yarn.client.RMHAServiceTarget@1d8299fd启用自动故障转移拒绝手动管理HA状态，因为它可能导致裂脑情景或其他不正确的状态。如果你很确定你知道你在做什么，请指定forcemanual标志





## 6.4安装集群hbase-0.98.6-cdh5.3.6.tar.gz



#### **集群规划**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 机器 | NN1 | NN2 | DN | ZK | ZKFC | JNN | HMASTER | HregionServer |
| **01** | \* |  |  |  | \* | \* | 主 |  |
| **02** |  | \* | \* | \* | \* | \* | 备 | 从 |
| **03** |  |  | \* | \* |  | \* |  | 从 |
| 04 |  |  | \* | \* |  |  |  | 从 |

#### **配置hbase-env.sh**

JAVA\_HOME=/opt/modules/jdk1.7.0\_67

HBASE\_MANAGRS\_ZK =false(关闭hbase自身zookeeper集群)

#### 配置 hbase-site.xml

<configuration>  
<property>  
    <name>hbase.rootdir</name>  
    <value>hdfs://ns1/hbase</value>  
  </property>  
  <property>  
    <name>hbase.cluster.distributed</name>  
    <value>true</value>  
  </property>  
  <property>  
    <name>hbase.zookeeper.quorum</name>  
    <value>**bigdata-cdh02.ibeifeng.com,**

**bigdata-cdh03.ibeifeng.com,bigdata-cdh04.ibeifeng.com**

</value>**<--对应的zookeeper集群，不用加端口-->**  
  </property>  
</configuration>

#### **配置regionservers**

**bigdata-cdh02.ibeifeng.com**

**bigdata-cdh03.ibeifeng.com**

**bigdata-cdh04.ibeifeng.com**

#### **配置backup-masters**

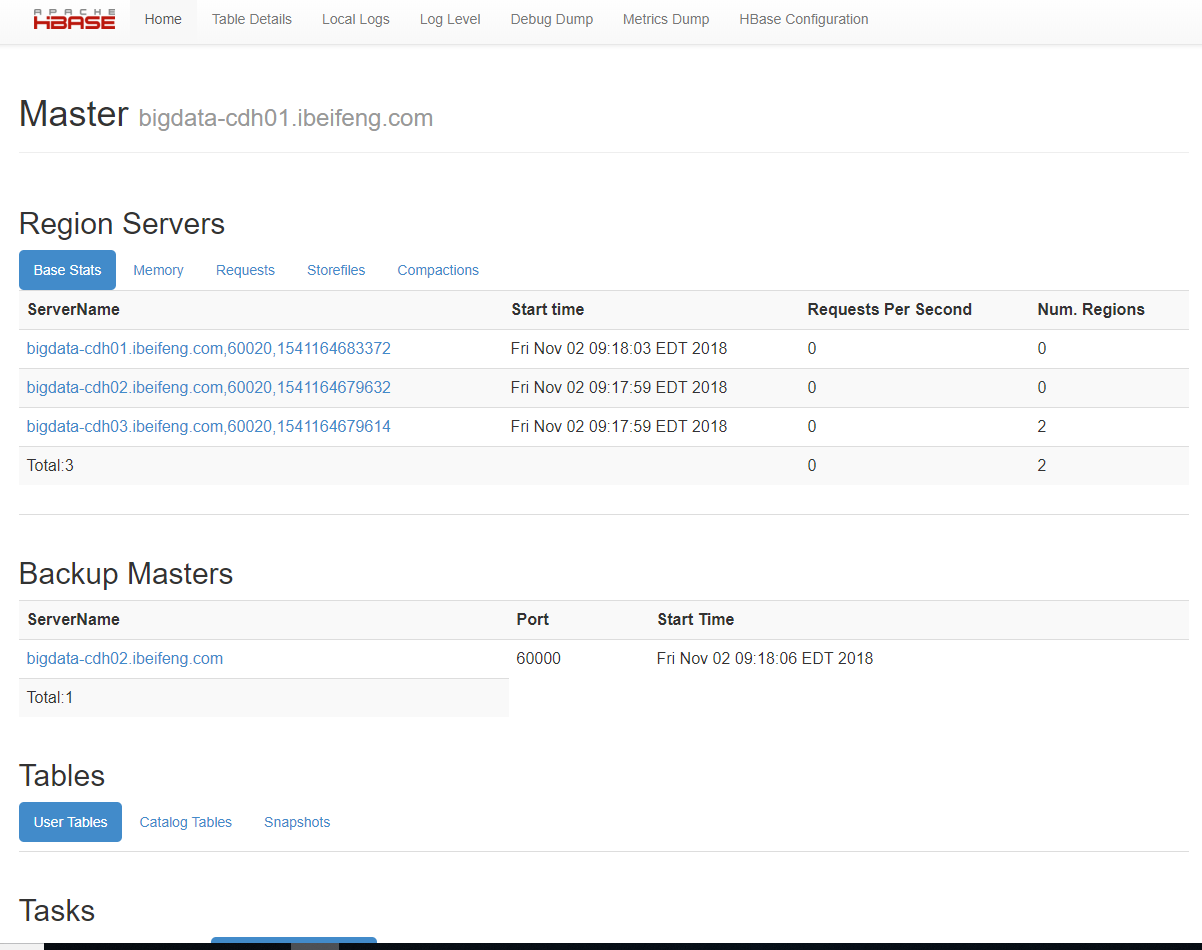
**bigdata-cdh02.ibeifeng.com**

**将hdfs当中的hdfs-site.xml 拷贝hbase的conf目录下，**保证集群当中每一台服务器配置文件相同**，在bigdata-cdh01.ibeifeng.com（Master节点） 执行 bin/start-hbase.sh启动主节点。bigdata-cdh02.ibeifeng.com(backup-Master节点) 执行 bin/start-hbase.sh启动备份节点。最好配置成全局，加载到环境变量中去。**

#### **启动命令**

**#/opt/cdh5.3.6/hbase-0.98.6-cdh5.3.6/bin/start-hbase.sh**

**访问http://bigdata-cdh01.ibeifeng.com:60010/master-status**



## 6.5安装集群hive-0.13.1-cdh5.3.6.tar.gz

安装hive前需要安装MySQL

#### **配置hive-env.sh**

# Set HADOOP\_HOME to point to a specific hadoop install directory

HADOOP\_HOME=/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

# Hive Configuration Directory can be controlled by:

export HIVE\_CONF\_DIR=/opt/cdh5.3.6/hive-0.13.1-cdh5.3.6/conf

# Folder containing extra ibraries required for hive compilation/execution can be controlled by:

export HIVE\_AUX\_JARS\_PATH=/opt/cdh5.3.6/hbase-0.98.6-cdh5.3.6/lib

#### **配置hive-sit.xml**

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>hive.metastore.uris</name>

<value>thrift://hadoop-senior.ibeifeng.com:9083</value>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/hive</value>

</property>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop-senior.ibeifeng.com:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123456</value>

</property>

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

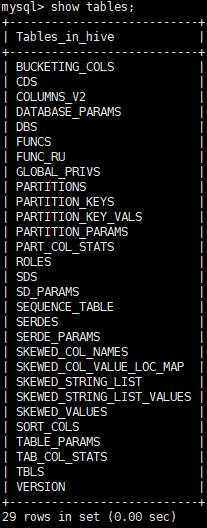
</property>

</configuration>

#### **启动metastore服务**

#/opt/cdh5.3.6/hive-0.13.1-cdh5.3.6/bin/hive --service metastore

查看mysql数据库hive中多了如下表



#### **启动hive shell**

#/opt/cdh5.3.6/hive-0.13.1-cdh5.3.6/bin/hive

## 6.6安装sqoop-1.4.5-cdh5.3.6.tar.gz

#### **配置sqoop-env.sh**

#Set path to where bin/hadoop is available

export HADOOP\_COMMON\_HOME=/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

#Set path to where hadoop-\*-core.jar is available

export HADOOP\_MAPRED\_HOME=/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

#set the path to where bin/hbase is available

export HBASE\_HOME=/opt/cdh5.3.6/hbase-0.98.6-cdh5.3.6

#Set the path to where bin/hive is available

export HIVE\_HOME=/opt/cdh5.3.6/hive-0.13.1-cdh5.3.6

#Set the path for where zookeper config dir is

#export ZOOCFGDIR=/opt/cdh5.3.6/zookeeper-3.4.5-cdh5.3.6/conf

#### 添加mysql驱动

cp /opt/modules/mysql-libs/mysql-connector-java-5.1.27/mysql-connector-java-5.1.27-bin.jar /opt/cdh5.3.6/sqoop-1.4.5-cdh5.3.6/lib

#### import语法将Mysql数据导入到hdfs

CREATE TABLE `my\_user` (

`id` tinyint(4) NOT NULL AUTO\_INCREMENT,

`account` varchar(255) DEFAULT NULL,

`passwd` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

);

INSERT INTO `my\_user` VALUES ('1', 'admin', 'admin');

INSERT INTO `my\_user` VALUES ('2', 'pu', '12345');

INSERT INTO `my\_user` VALUES ('3', 'system', 'system');

INSERT INTO `my\_user` VALUES ('4', 'zxh', 'zxh');

INSERT INTO `my\_user` VALUES ('5', 'test', 'test');

INSERT INTO `my\_user` VALUES ('6', 'pudong', 'pudong');

INSERT INTO `my\_user` VALUES ('7', 'qiqi', 'qiqi');

bin/sqoop import \

--connect jdbc:mysql://bigdata-cdh03.ibeifeng.com:3306/test \

--username root \

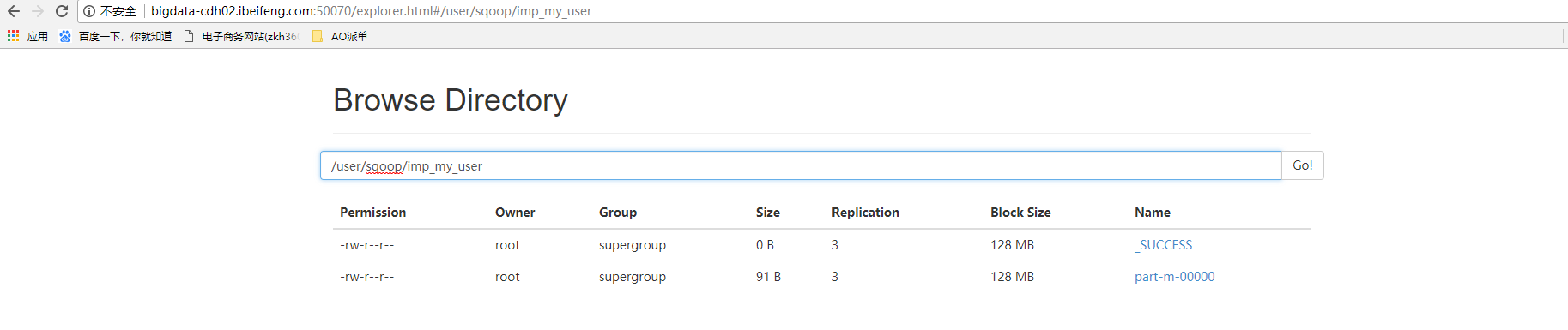
--password 123456 \

--table my\_user \

--target-dir /user/sqoop/imp\_my\_user \

--num-mappers 1

#### 数据压缩格式



#### export语法

创建文件添加数据

touch /opt/datas/user.txt

vi /opt/datas/user.txt

12,beifeng,beifeng

13,xuanyun,xuanyu

将上传的文件上传到hdfs

bin/hdfs dfs -mkdir -p /user/sqoop/exp/user/

bin/hdfs dfs -put /opt/datas/user.txt /user/sqoop/exp/user/

bin/sqoop export \

--connect jdbc:mysql://bigdata-cdh03.ibeifeng.com:3306/test \

--username root \

--password 123456 \

--table my\_user \

--export-dir /user/sqoop/exp/user/ \

--num-mappers 1

## 6.7安装flume-ng-1.5.0-cdh5.3.6.tar.gz

#### **配置flume-env.sh**

JAVA\_HOME=/opt/modules/jdk1.7.0\_67

#### **第一个例子收集日志到hdfs**

**flume-tail.conf内容：**

**### define agent**

**a2.sources = r2**

**a2.channels = c2**

**a2.sinks = k2**

**### define sources**

**a2.sources.r2.type = exec**

**a2.sources.r2.command=tail-f /opt/cdh-5.3.6/hive-0.13.1-cdh5.3.6/logs/hive.log**

**a2.sources.r2.shell = /bin/bash -c**

**### define channels**

**a2.channels.c2.type = memory**

**a2.channels.c2.capacity = 1000**

**a2.channels.c2.transactionCapacity = 100**

**### define sink**

**a2.sinks.k2.type = hdfs**

**a2.sinks.k2.hdfs.path = hdfs://ns1/user/flume/hive-logs/**

**a2.sinks.k2.hdfs.fileType = DataStream**

**a2.sinks.k2.hdfs.writeFormat = Text**

**a2.sinks.k2.hdfs.batchSize = 10**

**### bind the soures and sink to the channel**

**a2.sources.r2.channels = c2**

**a2.sinks.k2.channel = c2**

#### **第一个例子启动方式**

**#/opt/cdh5.3.6/apache-flume-1.5.0-cdh5.3.6-bin/bin/flume-ng agent \**

**-c conf \**

**-n a2 \**

**-f conf/flume-tail.conf \**

**-Dflume.root.logger=DEBUG,console**

#### **第二个例子收集日志到hdfs进行分区存储**

**监控某个日志文件的目录**

**/app/logs/2018-10-20**

**.....**

**/app/logs/2018-11-02**

**zz.log -> 不收集变化的日志文件**

**xx.log.comp 200MB**

**yy.log.comp 200MB**

**flume-app.conf 内容：**

**a1.sources = r1**

**a1.channels = c1**

**a1.sinks = k1**

**## common**

**a1.sources.r1.channels = c1**

**a1.sinks.k1.channel = c1**

**## sources config**

**a1.sources.r1.type = exec**

**a1.sources.r1.command = tail -F /usr/local/nginx/logs/access.log**

**## channels config**

**a1.channels.c1.type = memory**

**a1.channels.c1.capacity = 1000**

**a1.channels.c1.transactionCapacity = 1000**

**a1.channels.c1.byteCapacityBufferPercentage = 20**

**a1.channels.c1.byteCapacity = 1000000**

**a1.channels.c1.keep-alive = 60**

**#sinks config**

**a1.sinks.k1.type = hdfs**

**a1.sinks.k1.channel = c1**

**a1.sinks.k1.hdfs.path = hdfs://hadoop-senior.ibeifeng.com:8020/logs/%m/%d**

**a1.sinks.k1.hdfs.fileType = DataStream**

**a1.sinks.k1.hdfs.filePrefix = BF-%H**

**a1.sinks.k1.hdfs.fileSuffix=.log**

**a1.sinks.k1.hdfs.minBlockReplicas=1**

**a1.sinks.k1.hdfs.rollInterval=3600**

**a1.sinks.k1.hdfs.rollSize=132692539**

**a1.sinks.k1.hdfs.idleTimeout=10**

**a1.sinks.k1.hdfs.batchSize = 1**

**a1.sinks.k1.hdfs.rollCount=0**

**a1.sinks.k1.hdfs.round = true**

**a1.sinks.k1.hdfs.roundValue = 2**

**a1.sinks.k1.hdfs.roundUnit = minute**

**a1.sinks.k1.hdfs.useLocalTimeStamp = true**

#### **第二个例子启动方式**

**#/opt/cdh5.3.6/apache-flume-1.5.0-cdh5.3.6-bin/bin/flume-ng agent \**

**-c conf \**

**-n a1 \**

**-f conf/flume-app.conf \**

**-Dflume.root.logger=DEBUG,console**

## 6.8安装oozie-4.0.0-cdh5.3.6.tar.gz

## 6.9安装集群kafka\_2.10-0.8.2.1.tgz

**#tar -zxvf kafka\_2.10-0.8.2.1.tgz -C /opt/modules**

**#** **/opt/modules/kafka\_2.10-0.8.2.1/config**

# vi server.properties

#### **修改配置文件server.properties**

**host.name=bigdata-cdh01.ibeifeng.com**

**注意：主机修改响应的主机名**

**zookeeper.connect=****bigdata-cdh01.ibeifeng.com:2181,bigdata-cdh02.ibeifeng.com:2181,bigdata-cdh03.ibeifeng.com:2181**

**broker.id=01**

**注意的是第一台机器值为0,第二台1,第三台2其余修改不变**

#### **修改配置文件producer.properties**

**metadata.broker.list=bigdata-cdh01.ibeifeng.com:9092,bigdata-cdh02.ibeifeng.com:9092,bigdata-cdh03.ibeifeng.com:9092**

#### **修改配置文件consumer .properties**

**zookeeper.connect=bigdata-cdh01.ibeifeng.com:2181,bigdata-cdh02.ibeifeng.com:2181,bigdata-cdh03.ibeifeng.com:2181**

#### **启动kafka服务命令**

**#****cd /opt/modules/kafka\_2.10-0.8.2.1/**

**#****nohup bin/kafka-server-start.sh config/server.properties &**

#### **创建topic命令**

**#****bin/kafka-topics.sh --create --zookeeper** **bigdata-cdh01.ibeifeng.com:2181 --replication-factor 1 --partitions 1 --topic test**

#### **查看已用topic**

**#****bin/kafka-topics.sh --list --zookeeper bigdata-cdh01.ibeifeng.com:2181**

#### **生产数据**

**#****bin/kafka-console-producer.sh --broker-list** **bigdata-cdh01.ibeifeng.com:9092 --topic test**

#### **消费数据**

**#****bin/kafka-console-consumer.sh --zookeeper bigdata-cdh01.ibeifeng.com:2181 --topic test --from-beginning**

## 6.10安装Oozie-4.0.0-cdh5.3.6.tar.gz

**下载：**

**wget http://archive.cloudera.com/cdh5/cdh/5/oozie-4.0.0-cdh5.3.6.tar.gz**

**参考：**

**http://archive.cloudera.com/cdh5/cdh/5/oozie-4.0.0-cdh5.3.6/**

**https://cwiki.apache.org/confluence/display/OOZIE**

**oozie安装**

**1、下载oozie和ext-2.2.zip**

**2、设置环境变量**

**export OOZIE\_HOME=/home/hadoop/bigdater/oozie-4.0.0-cdh5.3.6**

**export PATH=$PATH:$OOZIE\_HOME/bin**

**3、进行conf配置信息修改，修改conf/oozie-site.xml文件。**

**主要就是进行元数据指定和service指定。**

**4、可以在conf/oozie-env.sh中进行参数修改，比如修改端口号，默认端口号为11000.**

**5、oozie根目录创建libext文件夹，复制mysql的driver压缩包到lib文件夹中。**

**6、执行sql创建，执行完成后，mysql中出现数据库和数据表**

**命令：ooziedb.sh create -sqlfile oozie.sql -run**

**7、设置hadoop代理用户。（hive安装的时候已经设置过）**

**hadoop.proxyuser.hadoop.hosts&hadoop.proxyuser.hadoop.groups**

**8、在hdfs上创建公用文件夹：执行命令**

**oozie-setup.sh sharelib create -fs hdfs://ns1**

**-locallib oozie-sharelib-4.0.0-cdh5.3.6-yarn.tar.gz**

**9、创建war文件，执行**

**addtowar.sh -inputwar ./oozie.war -outputwar ./oozie-server/webapps/oozie.war -hadoop 2.5.0 $HADOOP\_HOME -jars**

**./libext/mysql-connector-java-5.1.31.jar -extjs ../softs/ext-2.2.zip**

**10、运行：oozied.sh run 或者 oozied.sh start(前者在前端运行，后者在后台运行)**

**11、查看web界面&查看状态oozie admin -oozie http://主机名:11000/oozie -status**

## 6.11安装Mysql5.6.24

#### **卸载原有Mysql**

**#find / -name mysql**

**# rm -rf /var/lib/mysql/mysql**

**# rm -rf /var/lib/mysql/**

**# rm -rf /usr/lib64/mysql**

**#rpm -qa|grep mysql**

**#rpm -e --nodeps mysql-libs-5.1.66-2.el6\_3.x86\_64**

#### **安装****MySQL-server-5.6.24-1.el6.x86\_64.rpm**

**#rpm -ivh MySQL-server-5.6.24-1.el6.x86\_64.rpm**

**执行命令后仔细观察日志查看初始密码:**

**A RANDOM PASSWORD HAS BEEN SET FOR THE MySQL root USER !**

**You will find that password in '**/root/.mysql\_secret**'.**

**# cat /root/.mysql\_secret**

**The random password set for the root user at Fri Nov 2 10:25:24 2018 (local time): **JBuHOuPBqlE3BJJ7****

#### **安装MySQL-client-5.6.24-1.el6.x86\_64.rpm**

**#rpm -ivh MySQL-client-5.6.24-1.el6.x86\_64.rpm**

**#service mysql restart**

**# mysql -uroot -pJBuHOuPBqlE3BJJ7**

**SET PASSWORD=PASSWORD('123456');**

**#mysql -uroot -p123456**

**use mysql;**

**select HOST,User,Password from user;**

#### **设置远程连接**

**delete from user where Host='::1';**

**delete from user where Host='127.0.0.1';**

**delete from user where Host='bigdata-cdh03.ibeifeng.com';**

**update user set HOST='%' where User='root' and Host='localhost' ;**

**flush privileges;**

**quit;**

## 6.12安装集群Nginx+Keepalived+Lvs

### 6.12.1 Nginx配置

#### **集群规划**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vip | ip | 主机名 | 端口号 | 主从 |
| 192.168.174.100 | 192.168.174.135 | bigdata-cdh01.ibeifeng.com | 88 | MASTER |
| 192.168.174.136 | bigdata-cdh02.ibeifeng.com | 88 | BACKUP |
| 192.168.174.133 | bigdata-cdh02.ibeifeng.com | 88 | BACKUP |

#### **环境准备**

所需软件：keepalived-1.2.18.tar.gz 、nginx-1.6.2.tar.gz 安装编译 Nginx 所需的依赖包 :

# yum install gcc gcc-c++ make automake autoconf libtool pcre pcre-devel zlib zlib-devel openssl openssl-devel

上传 Nginx(nginx-1.6.2.tar.gz)到 /usr/local/src 目录

#### **编译安装 Nginx**

# cd /usr/local/src/

# tar -zxvf nginx-1.6.2.tar.gz

# cd nginx-1.6.2

# ./configure --prefix=/usr/local/nginx

# make && make install

#### **配置 Nginx**

# vi /usr/local/nginx/conf/nginx.conf

user root;

worker\_processes 1;

events { worker\_connections 1024; }

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 88;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html { root html; }

} }

#### **修改Nginx欢迎首页内容**

# vi /usr/local/nginx/html/index.html

192.168.174.135 中的标题加 1

<h1>Welcome to nginx! 1</h1>

192.168.174.136 中的标题加 2

<h1>Welcome to nginx! 2</h1>

192.168.174.137 中的标题加 3

<h1>Welcome to nginx! 3</h1>

#### **测试 Nginx 是否安装成功**

# /usr/local/nginx/sbin/nginx -t

nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

#### **启动 Nginx**

# /usr/local/nginx/sbin/nginx

#### **重启 Nginx**

# /usr/local/nginx/sbin/nginx -s reload

#### **设置 Nginx 开机启动**

# vi /etc/rc.local

加入： /usr/local/nginx/sbin/nginx

### 6.11.2 Keepalived配置

上传或下载 keepalived（keepalived-1.2.18.tar.gz）到 /usr/local/src 目录

**解压安装**

# cd /usr/local/src

# tar -zxvf keepalived-1.2.18.tar.gz

# cd keepalived-1.2.18

# ./configure --prefix=/usr/local/keepalived

# make && make install

#### **将 keepalived 安装成 Linux 系统服务：**

因为没有使用 keepalived 的默认路径安装（默认是/usr/local）,安装完成之后，需要做一些工作 。复制默认配置文件到默认路径

# mkdir /etc/keepalived

# cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

#### **复制 keepalived 服务脚本到默认的地址**

# cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/

# cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/

# ln -s /usr/local/sbin/keepalived /usr/sbin/

# ln -s /usr/local/keepalived/sbin/keepalived /sbin/

#### **设置 keepalived 服务开机启动**

# chkconfig keepalived on

#### **修改 Keepalived 配置文件**

(1) MASTER 节点配置文件（192.168.174.135）

# vi /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

## keepalived 自带的邮件提醒需要开启 sendmail 服务。建议用独立的监控或第三方 SMTP

router\_id bigdata-cdh01.ibeifeng.com ## 标识本节点的字条串，通常为 hostname

}

## keepalived 会定时执行脚本并对脚本执行的结果进行分析，动态调整 vrrp\_instance 的优先级。如果

## 脚本执行结果为 0，并且 weight 配置的值大于 0，则优先级相应的增加。如果脚本执行结果非 0，并且 weight

## 配置的值小于 0，则优先级相应的减少。其他情况，维持原本配置的优先级，即配置文件中 priority 对应

## 的值。

vrrp\_script chk\_nginx {

script "/etc/keepalived/nginx\_check.sh" ## 检测 nginx 状态的脚本路径

interval 2 ## 检测时间间隔

weight -20 ## 如果条件成立，权重-20

}

## 定义虚拟路由，VI\_1 为虚拟路由的标示符，自己定义名称

vrrp\_instance VI\_1 {

state MASTER ## 主节点为 MASTER，对应的备份节点为 BACKUP

interface eth0 ## 绑定虚拟 IP 的网络接口，与本机 IP 地址所在的网络接口相同，我的是 eth0

virtual\_router\_id 51 ## 虚拟路由的 ID 号，两个节点设置必须一样，可选 IP 最后一段使用, 相同的VRID 为一个组，他将决定多播的 MAC 地址

mcast\_src\_ip 192.168.174.135 ## 本机 IP 地址

priority 100 ## 节点优先级，值范围 0-254，MASTER 要比 BACKUP 高

nopreempt ## 优先级高的设置 nopreempt 解决异常恢复后再次抢占的问题

advert\_int 1 ## 组播信息发送间隔，两个节点设置必须一样，默认 1s

## 设置验证信息，两个节点必须一致

authentication {

auth\_type PASS

auth\_pass 1111 ## 真实生产，按需求对应该过来

}

## 将 track\_script 块加入 instance 配置块

track\_script {

chk\_nginx ## 执行 Nginx 监控的服务

}

## 虚拟 IP 池, 两个节点设置必须一样

virtual\_ipaddress {

192.168.174.100 ## 虚拟 ip，可以定义多个

}

}

(2)BACKUP 节点配置文件（192.168.174.136）:

# vi /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id bigdata-cdh02.ibeifeng.com

}

vrrp\_script chk\_nginx {

script "/etc/keepalived/nginx\_check.sh"

interval 2

weight -20

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth1

virtual\_router\_id 51

mcast\_src\_ip 192.168.174.136

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

track\_script {

chk\_nginx

}

virtual\_ipaddress {

192.168.174.100

}

}

1. BACKUP 节点配置文件（192.168.174.137）:略

#### **编写 Nginx 状态检测脚本**

# vi /etc/keepalived/nginx\_check.sh (已在 keepalived.conf 中配置)

脚本要求：如果 nginx 停止运行，尝试启动，如果无法启动则杀死本机的 keepalived 进程，keepalied将虚拟 ip 绑定到 BACKUP 机器上。内容如下：

#!/bin/bash

A=`ps -C nginx –no-header |wc -l`

if [ $A -eq 0 ];then

/usr/local/nginx/sbin/nginx

sleep 2

if [ `ps -C nginx --no-header |wc -l` -eq 0 ];then

killall keepalived

fi

fi

#### **给脚本赋执行权限：**

# chmod +x /etc/keepalived/nginx\_check.sh

启动 Keepalived

# service keepalived start

Starting keepalived: [ OK ]

#### **Keepalived+Nginx 的高可用测试**

(1)关闭 192.168.1.51 中的 Nginx，Keepalived 会将它重新启动

# /usr/local/nginx/sbin/nginx -s stop

(2)关闭 192.168.174.135 中的 Keepalived，VIP 会切换到 192.168.174.136或者 192.168.174.137中

# service keepalived stop

#### **Keepalived 服务管理命令**

停止：service keepalived stop

启动：service keepalived start

重启：service keepalived restart

查看状态：service keepalived status

## 6.13安装Spark-1.5.1-bin-hadoop2.4.tgz

#### **配置scala环境变量**

因为spark底层使用scala语言编写所以需要配置其黄静变量。

解压scala-2.11.4.tgz 到 /opt/modules/scala-2.10.4

# vi /etc/profile

export SCALA\_HOME=/opt/modules/scala-2.10.4

export PATH=$PATH:$SCALA\_HOME/bin

#source /etc/profile

#### **配置spark**

解压编译后的文件spark-1.5.1-bin-hadoop2.4.tgz

到/opt/cdh5.3.6/spark-1.3.0-bin-2.5.0-cdh5.3.6/

#### **配置slaves**

bigdata-cdh01.ibeifeng.com

bigdata-cdh02.ibeifeng.com

bigdata-cdh03.ibeifeng.com

#### **配置spark-env.sh**

#### JDK

JAVA\_HOME=/opt/modules/jdk1.7.0\_67

#### SCALA

SCALA\_HOME=/opt/modules/scala-2.10.4

#### Hadoop

HADOOP\_CONF\_DIR=/opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

#### 本机IP地址

SPARK\_LOCAL\_IP=bigdata-cdh01.ibeifeng.com

#### 设置spark主节点

SPARK\_MASTER\_IP=bigdata-cdh01.ibeifeng.com

#### 设置spark主节点端口号

SPARK\_MASTER\_PORT=7077

#### 设置spark主节点前台访问端口号

SPARK\_MASTER\_WEBUI\_PORT=8080

SPARK\_WORKER\_CORES=2

SPARK\_WORKER\_MEMORY=2g

#### 设置spark工作节点端口号

SPARK\_WORKER\_PORT=7078

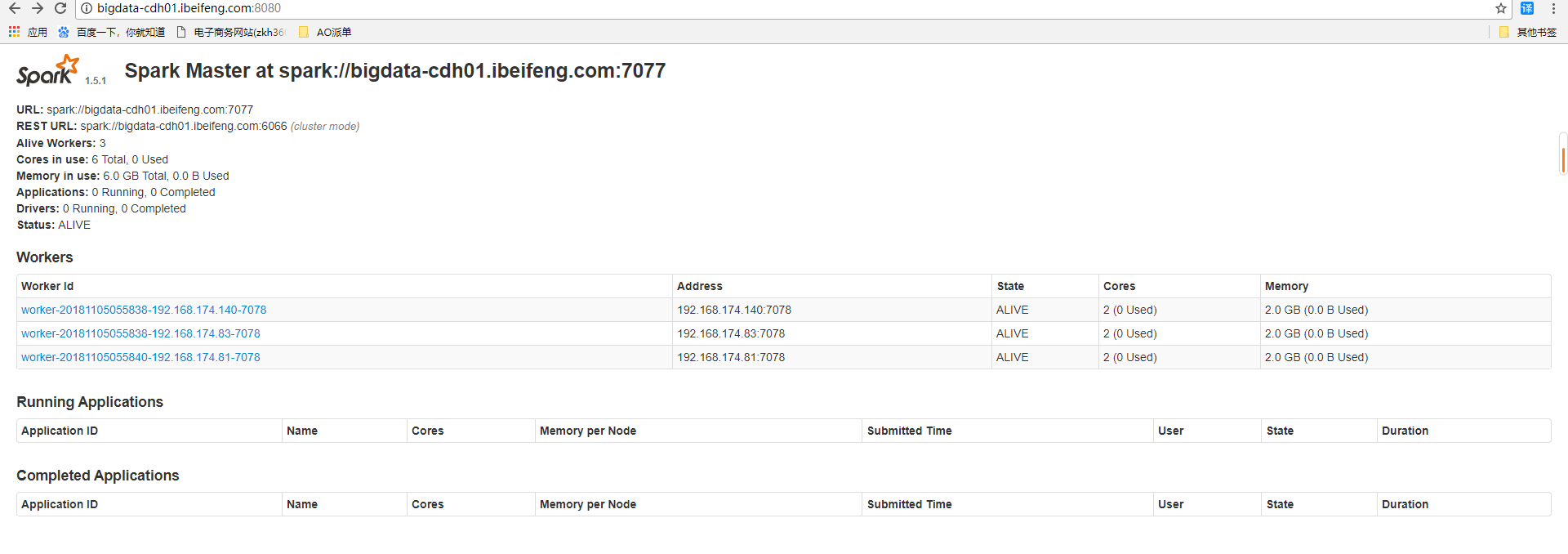
#### 设置spark工作节点前台访问端口号

SPARK\_WORKER\_WEBUI\_PORT=8081

SPARK\_WORKER\_INSTANCES=1

### zookeeper实现spark的高可用

SPARK\_MASTER\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER-Dspark.deploy.zookeeper.url=bigdata-cdh01.ibeifeng.com:2181,bigdata-cdh02.ibeifeng.com:2181,bigdata-cdh03.ibeifeng.com:2181-Dspark.deploy.zookeeper.dir=/spark"



# **7 大数据框架简介**

## 7.1 Hadoop基本语法及操作

### 7.1.1 Hadoop Hdfs

#cd /opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6

创建文件夹

#bin/hdfs dfs -mkdir -p /user/hadoop-data/input

编辑数据

#touch /opt/datas/input.txt

Tom hello

Jery hi

Alice hello

Bob hello

上传数据

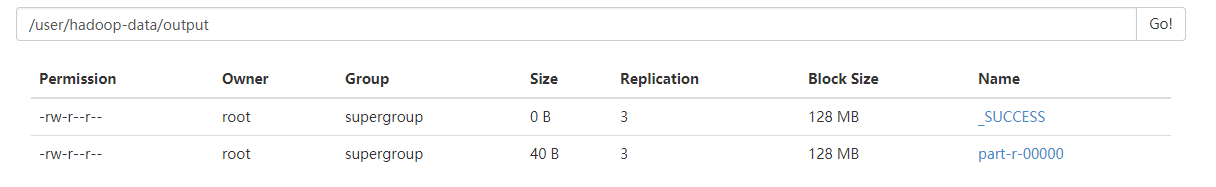
# bin/hdfs dfs -put /opt/datas/input.txt /user/hadoop-data/input

运行自带的wordcount程序

#bin/yarn jar share/hadoop/mapreduce/hadoop\*example\*.jar wordcount /user/hadoop-data/input /user/hadoop-data/output

注意：参数/user/hadoop-data/input 刚才上传的文件目录。

参数/user/hadoop-data/output 结果存储目录（必须是新建的，hdfs不存在的）。

结果

Alice 1

Bob 1

HI 1

Jery 1

Tom 1

Hello 3

### 7.1.2 MapReduce

用Java编写一个统计单词个数的wordcount程序

package com.zkh.mr.wordcount;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WordCountMapReduce

extends Configured implements Tool {

// step 1: Map Class

public static class WordCountMapper

extends Mapper<LongWritable, Text, Text, IntWritable> {

private Text mapOutputKey = new Text();

private final static IntWritable mapOuputValue = new IntWritable(1);

@Override

public void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String lineValue = value.toString();

// String[] strs = lineValue.split(" ");

StringTokenizer stringTokenizer = new StringTokenizer(lineValue);

while (stringTokenizer.hasMoreTokens()) {

String wordValue = stringTokenizer.nextToken();// get word value

mapOutputKey.set(wordValue);// set value

context.write(mapOutputKey, mapOuputValue);// output

}

}

}

// step 2: Reduce Class

public static class WordCountReducer

extends Reducer<Text, IntWritable, Text, IntWritable> {

private IntWritable outputValue = new IntWritable();

@Override

public void reduce(Text key, Iterable<IntWritable> values, Context context)

throws IOException, InterruptedException {

int sum = 0;// sum tmp

for (IntWritable value : values) {

sum += value.get();

}

outputValue.set(sum);

context.write(key, outputValue);

}

}

// step 3: Driver ,component job

public int run(String[] args) throws Exception {

// 1: get confifuration

Configuration configuration = getConf();

// 2: create Job

Job job = Job.getInstance(configuration, //

this.getClass().getSimpleName());

// run jar

job.setJarByClass(this.getClass());

// 3: set job

// input -> map -> reduce -> output

// 3.1: input

Path inPath = new Path(args[0]);

FileInputFormat.addInputPath(job, inPath);

// 3.2: map

job.setMapperClass(WordCountMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

// 3.3: reduce

job.setReducerClass(WordCountReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

// 3.4: output

Path outPath = new Path(args[1]);

FileOutputFormat.setOutputPath(job, outPath);

// 4: submit job

boolean isSuccess = job.waitForCompletion(true);

return isSuccess ? 0 : 1;

}

// step 4: run program

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

// 1: get confifuration

Configuration configuration = new Configuration();

// int status = new WordCountMapReduce().run(args);

int status = ToolRunner.run(configuration, //

new WordCountMapReduce(), //

args);

System.exit(status);

}

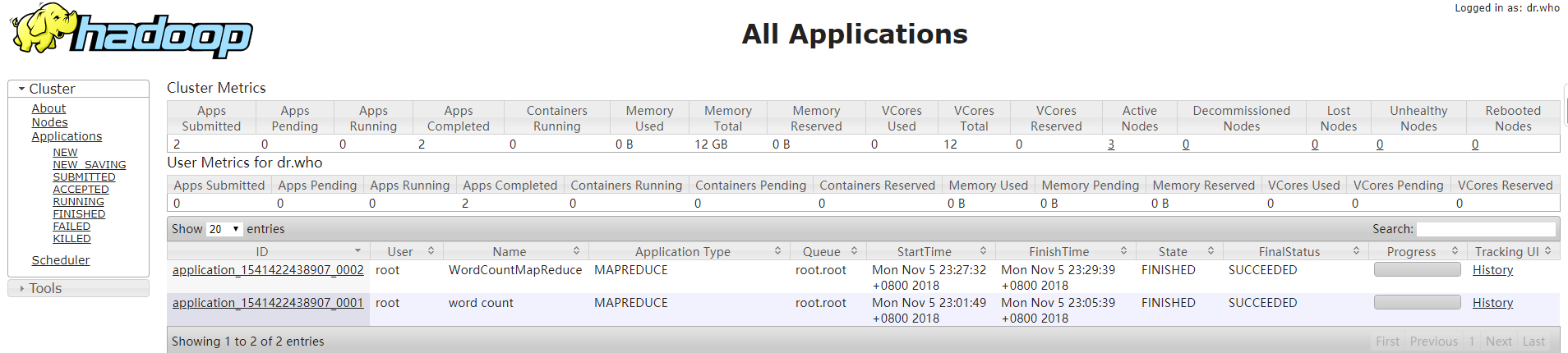
}

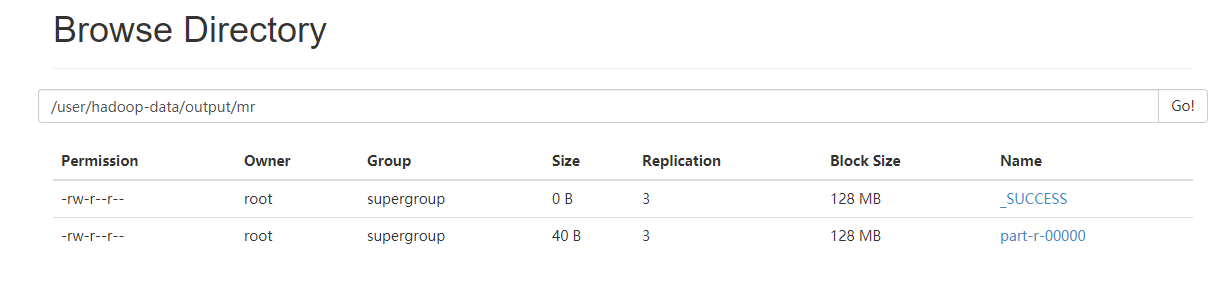
创建一个文件夹专门存放jar

#mkdir -p /opt/cdh5.3.6/hadoop-2.5.0-cdh5.3.6/jars

将该程序打包wordcount.jar上传此目录

#bin/yarn jar jars/wordcount.jar /user/hadoop-data/input /user/hadoop-data/output/mr





结果：

Alice 1

Bob 1

HI 1

Jery 1

Tom 1

hello 3

## 7.2 Hbase基本语法及操作

**进入hbase shell console**  
    $HBASE\_HOME/bin/hbase shell  
    如果有kerberos认证，需要事先使用相应的keytab进行一下认证（使用kinit命令），认证成功之后再使用hbase shell进入可以使用whoami命令可查看当前用户  
    hbase(main)> whoami

**表的管理**  
    1）查看有哪些表  
    hbase(main)> list  
    2）创建表  
    # 语法：create <table>, {NAME => <family>, VERSIONS => <VERSIONS>}  
    # 例如：创建表t1，有两个family name：f1，f2，且版本数均为2  
    hbase(main)> create 't1',{NAME => 'f1', VERSIONS => 2},{NAME => 'f2', VERSIONS => 2}  
    3）删除表  
    分两步：首先disable，然后drop  
    例如：删除表t1  
    hbase(main)> disable 't1'  
    hbase(main)> drop 't1'  
    4）查看表的结构  
    # 语法：describe <table>  
    # 例如：查看表t1的结构  
    hbase(main)> describe 't1'  
    5）修改表结构  
    修改表结构必须先disable  
    # 语法：alter 't1', {NAME => 'f1'}, {NAME => 'f2', METHOD => 'delete'}  
    # 例如：修改表test1的cf的TTL为180天  
    hbase(main)> disable 'test1'  
    hbase(main)> alter 'test1',{NAME=>'body',TTL=>'15552000'},{NAME=>'meta', TTL=>'15552000'}  
    hbase(main)> enable 'test1'

**权限管理**  
    1）分配权限  
    # 语法 : grant <user> <permissions> <table> <column family> <column qualifier> 参数后面用逗号分隔  
    # 权限用五个字母表示： "RWXCA".  
    # READ('R'), WRITE('W'), EXEC('X'), CREATE('C'), ADMIN('A')  
    # 例如，给用户‘test'分配对表t1有读写的权限，  
    hbase(main)> grant 'test','RW','t1'  
    2）查看权限  
    # 语法：user\_permission <table>  
    # 例如，查看表t1的权限列表  
    hbase(main)> user\_permission 't1'  
  
    3）收回权限  
    # 与分配权限类似，语法：revoke <user> <table> <column family> <column qualifier>  
    # 例如，收回test用户在表t1上的权限  
    hbase(main)> revoke 'test','t1'

**表数据的增删改查**  
    1）添加数据  
    # 语法：put <table>,<rowkey>,<family:column>,<value>,<timestamp>  
    # 例如：给表t1的添加一行记录：rowkey是rowkey001，family name：f1，column name：col1，value：value01，timestamp：系统默认  
    hbase(main)> put 't1','rowkey001','f1:col1','value01'  
    用法比较单一。  
    2）查询数据  
    a）查询某行记录  
    # 语法：get <table>,<rowkey>,[<family:column>,....]  
    # 例如：查询表t1，rowkey001中的f1下的col1的值  
    hbase(main)> get 't1','rowkey001', 'f1:col1'  
    # 或者：  
    hbase(main)> get 't1','rowkey001', {COLUMN=>'f1:col1'}  
    # 查询表t1，rowke002中的f1下的所有列值  
    hbase(main)> get 't1','rowkey001'  
    b）扫描表  
    # 语法：scan <table>, {COLUMNS => [ <family:column>,.... ], LIMIT => num}  
    # 另外，还可以添加STARTROW、TIMERANGE和FITLER等高级功能  
    # 例如：扫描表t1的前5条数据  
    hbase(main)> scan 't1',{LIMIT=>5}  
    c）查询表中的数据行数  
    # 语法：count <table>, {INTERVAL => intervalNum, CACHE => cacheNum}  
    # INTERVAL设置多少行显示一次及对应的rowkey，默认1000；CACHE每次去取的缓存区大小，默认是10，调整该参数可提高查询速度  
    # 例如，查询表t1中的行数，每100条显示一次，缓存区为500  
    hbase(main)> count 't1', {INTERVAL => 100, CACHE => 500}  
    3）删除数据  
    a )删除行中的某个列值  
    # 语法：delete <table>, <rowkey>,  <family:column> , <timestamp>,必须指定列名  
    # 例如：删除表t1，rowkey001中的f1:col1的数据  
    hbase(main)> delete 't1','rowkey001','f1:col1'  
    注：将删除改行f1:col1列所有版本的数据  
    b )删除行  
    # 语法：deleteall <table>, <rowkey>,  <family:column> , <timestamp>，可以不指定列名，删除整行数据  
    # 例如：删除表t1，rowk001的数据  
    hbase(main)> deleteall 't1','rowkey001'  
    c）删除表中的所有数据  
    # 语法： truncate <table>  
    # 其具体过程是：disable table -> drop table -> create table  
    # 例如：删除表t1的所有数据  
    hbase(main)> truncate 't1'

**Region管理**  
    1）移动region  
    # 语法：move 'encodeRegionName', 'ServerName'  
    # encodeRegionName指的regioName后面的编码，ServerName指的是master-status的Region Servers列表  
    # 示例  
hbase(main)>move'4343995a58be8e5bbc739af1e91cd72d', 'db-41.xxx.xxx.org,60020,1390274516739'  
    2）开启/关闭region  
    # 语法：balance\_switch true|false  
    hbase(main)> balance\_switch  
    3）手动split  
    # 语法：split 'regionName', 'splitKey'  
    4）手动触发major compaction  
    #语法：  
    #Compact all regions in a table:  
    #hbase> major\_compact 't1'  
    #Compact an entire region:  
    #hbase> major\_compact 'r1'  
    #Compact a single column family within a region:  
    #hbase> major\_compact 'r1', 'c1'  
    #Compact a single column family within a table:  
    #hbase> major\_compact 't1', 'c1'

## 7.3 Spark基本语法及操作

## 7.4 Zookeeper基本语法及操作

### 7.4.1基本语法

（1）使用zookeeper安装bin目录下的./zkCli.sh连接到zookeeper服务器上，基本语法如下：

./zkCli.sh -timeout 0 -r -server ip:port

其中，-timeout表示当前会话的超时时间，zookeper依靠与客户端的心跳来判断会话是否有效，单位是毫秒-r代表只读模式，zookeeper的只读模式指一个服务器与集群中过半机器失去连接以后，这个服务器就不在不处理客户端的请求，但我们仍然希望该服务器可以提供读服务。

-server，指定服务器ip地址和端口号

（2）h：查看帮助

（3）ls，列出某一节点下的子节点信息

（4）stat，查看节点的状态信息

在zookeeper中，每一次对节点的写操作都认为是一次事务，每一个事务，系统都会分配一个唯一的事务ID，czxid代表该节点被创建的事务ID，ctime表示创建的时间，mzxid表示最后一次被更新的事务ID，mtime是修改时间，pzxid表示子节点列表最后一次被更新的事务ID，cversion子节点的版本号，dataversion数据版本号，aclversion权限版本号，ephemeralOwner用于临时节点，代表临时节点的事务ID，如果为持久节点则为0，dataLength代表节点存储的数据的长度，numChildren当前节点的子节点个数。

（5）get，获取当前节点存储的数据内容

ls2，列出当前节点的子节点，同时列出节点状态

create，创建节点 -s，顺序节点-e，临时节点

set，修改节点数据，可携带版本号

修改的时候要么不携带版本号，要么携带的版本号要跟dataVersion的版本号一致，否则就会报错（类似于乐观锁机制）。

（8）delete，删除节点，只能删除没有子节点的节点，要删除含有子节点的时候需使用rmr命令。

（9）rmr，删除节点。

（10）setquota，配额，给节点限制值，比如限制子节点个数、节点数据的长度

-n，限制子节点个数 -b，限制值的长度

（11）listquota，查看配额，以及节点的配额状态

（12）delquota，删除配额

（13）close，关闭当前连接

（14）history，查看历史执行指令

（15）redo，重复执行指令

### 7.4.2典型使用场景

ZooKeeper是一个高可用的分布式数据管理与系统协调框架。基于对Paxos算法的实现，使该框架保证了分布式环境中数据的强一致性，也正是基 于这样的特性，使得zookeeper能够应用于很多场景。

**数据发布与订阅**

发布与订阅即所谓的配置管理，顾名思义就是将数据发布到zk节点上，供订阅者动态获取数据，实现配置信息的集中式管理和动态更新。例如全局的配置信息，地址列表等就非常适合使用。

1. 索引信息和集群中机器节点状态存放在zk的一些指定节点，供各个客户端订阅使用。

2. 系统日志（经过处理后的）存储，这些日志通常2-3天后被清除。

3. 应用中用到的一些配置信息集中管理，在应用启动的时候主动来获取一次，并且在节点上注册一个Watcher，以后每次配置有更新，实时通知到应用，获取最新配置信息。

4. 业务逻辑中需要用到的一些全局变量，比如一些消息中间件的消息队列通常有个offset，这个offset存放在zk上，这样集群中每个发送者都能知道当前的发送进度。

5. 系统中有些信息需要动态获取，并且还会存在人工手动去修改这个信息。以前通常是暴露出接口，例如JMX接口，有了zk后，只要将这些信息存放到zk节点上即可。

**分布通知/协调**

ZooKeeper 中特有watcher注册与异步通知机制，能够很好的实现分布式环境下不同系统之间的通知与协调，实现对数据变更的实时处理。使用方法通常是不同系统都对 ZK上同一个znode进行注册，监听znode的变化（包括znode本身内容及子节点的），其中一个系统update了znode，那么另一个系统能 够收到通知，并作出相应处理。

1. 另一种心跳检测机制：检测系统和被检测系统之间并不直接关联起来，而是通过zk上某个节点关联，大大减少系统耦合。

2. 另一种系统调度模式：某系统有控制台和推送系统两部分组成，控制台的职责是控制推送系统进行相应的推送工作。管理人员在控制台作的一些操作，实际上是修改 了ZK上某些节点的状态，而zk就把这些变化通知给他们注册Watcher的客户端，即推送系统，于是，作出相应的推送任务。

3. 另一种工作汇报模式：一些类似于任务分发系统，子任务启动后，到zk来注册一个临时节点，并且定时将自己的进度进行汇报（将进度写回这个临时节点），这样任务管理者就能够实时知道任务进度。

总之，使用zookeeper来进行分布式通知和协调能够大大降低系统之间的耦合。

**分布式锁**

分布式锁，这个主要得益于ZooKeeper为我们保证了数据的强一致性，即用户只要完全相信每时每刻，zk集群中任意节点（一个zk server）上的相同znode的数据是一定是相同的。锁服务可以分为两类，一个是保持独占，另一个是控制时序。

保持独占，就是所有试图来获取这个锁的客户端，最终只有一个可以成功获得这把锁。通常的做法是把zk上的一个znode看作是一把锁，通过create znode的方式来实现。所有客户端都去创建 /distribute\_lock 节点，最终成功创建的那个客户端也即拥有了这把锁。

控制时序，就是所有视图来获取这个锁的客户端，最终都是会被安排执行，只是有个全局时序了。做法和上面基本类似，只是这里 /distribute\_lock 已经预先存在，客户端在它下面创建临时有序节点（这个可以通过节点的属性控制：CreateMode.EPHEMERAL\_SEQUENTIAL来指定）。Zk的父节点（/distribute\_lock）维持一份sequence,保证子节点创建的时序性，从而也形成了每个客户端的全局时序。

**集群管理**

1. 集群机器监控：这通常用于那种对集群中机器状态，机器在线率有较高要求的场景，能够快速对集群中机器变化作出响应。这样的场景中，往往有一个监控系统，实时检测集群机器是否存活。过去的做法通常是：监控系统通过某种手段（比如ping）定时检测每个机器，或者每个机器自己定时向监控系统汇报“我还活着”。 这种做法可行，但是存在两个比较明显的问题：1. 集群中机器有变动的时候，牵连修改的东西比较多。2. 有一定的延时。

利用ZooKeeper有两个特性，就可以实时另一种集群机器存活性监控系统：a. 客户端在节点 x 上注册一个Watcher，那么如果 x 的子节点变化了，会通知该客户端。b. 创建EPHEMERAL类型的节点，一旦客户端和服务器的会话结束或过期，那么该节点就会消失。

2. Master选举则是zookeeper中最为经典的使用场景了。

在分布式环境中，相同的业务应用分布在不同的机器上，有些业务逻辑（例如一些耗时的计算，网络I/O处理），往往只需要让整个集群中的某一台机器进行执行， 其余机器可以共享这个结果，这样可以大大减少重复劳动，提高性能，于是这个master选举便是这种场景下的碰到的主要问题。

利用ZooKeeper的强一致性，能够保证在分布式高并发情况下节点创建的全局唯一性，即：同时有多个客户端请求创建 /currentMaster 节点，最终一定只有一个客户端请求能够创建成功。