

大数据技术 笔记本虚拟机搭建 Hadoop 集群操作文档

网络科学与智能系统研究所

2018 年 1 月

修订记录

版本号	修订状态	简要说明修订内容和范围	修订日期	修订人
V1.0	A	完成 MySQL, zookeeper, Kafka 三部分	2018.1.30	宋超
V1.1	M	修改 SSH 章节的 bug	2018.1.30	宋超
V1.2	A	完成所有的 Hadoop 集群的安装说明	2018.1.31	宋超
V1.3	A	添加了封面、目录、页眉	2018.1.31	宋超
V1.4	A	新增了第二块网卡的配置方式（3.1 到 3.2.3）	2018.1.31	宋超
V1.5	A	修改 hadoop 安装过程的配置文件内容，增加 HDFS 的初始化， 修改 hive 的配置文件，增加 spark 安装过程的描述	2018.2.5	曹仲、 宋超
V1.6	A	增加了架构图与引言	2018.2.26	宋超
V1.7	A	增加了对于虚拟机网络配置的解释	2018.3.5	宋超
V1.8	M	对 ZooKeeper 以及 Kafka 配置代码中的一些问题进行了修改	2018.3.6	纪宇泽
V1.9	M	修改 HBase 的环境变量设置， 修改 Storm 配置文件内容	2018.3.9	王贝贝
V1.91	M	修改了网卡配置的内容	2018.3.12	宋超
V1.92	M	修改了新建 hadoop 用户的内容	2018.3.12	宋超
V1.93	M	修改了错别字	2018.3.14	宋超
V1.94	M	修改了改 hosts 的描述	2018.3.14	宋超
V1.95	M	修改了 Hive 的启动条件	2018.3.15	宋超
V1.96	A	增加了 3.2.5 检查虚拟机间能否正常通信的内容	2018.3.16	宋超
V2.0	A	增加了各组件的测试内容以及 web 界面的进入方法	2018.3.17	宋超
V2.01	M	修复了同步 hbase 工作目录的小 bug	2018.3.17	宋超
V2.02	M	增加了使 hive 用户可以在 cluster2 上登陆 MySQL 的命令	2018.3.19	宋超
V2.03	M	修复了小 bug	2018.3.27	宋超
V2.1	M	增加 Kafka 的新配置	2018.3.29	宋超
V2.2	M	修改了 MySQL 的安装方法	2018.6.10	宋超
V2.3	M	修复了小 bug	2018.6.17	宋超

注：修订记录在体系文件发布后换版时使用，修订状态栏填写：A—增加，M—修改，D—删除

目录

1. 概述	1
2. 服务列表	2
3. 搭建步骤	4
3.1 安装虚拟机	4
3.2 准备工作	6
3.2.1 关闭防火墙和 Selinux	6
3.2.2 安装软件	6
3.2.3 检查网卡是否开机自启	6
3.2.4 修改 hosts	8
3.2.5 检查网络是否正常	8
3.2.6 新建 hadoop 用户	9
3.2.7 生成 ssh 密钥并分发	10
3.2.8 安装 NTP 服务	10
3.3 安装 MySQL	11
3.3.1 安装	11
3.3.2 测试	12
3.4 安装 JDK	13
3.4.1 安装	13
3.4.2 测试	14
3.5 安装 ZooKeeper	14
3.6 安装 Kafka	17
3.6.1 安装	17
3.6.2 测试	18
3.6.3 可能遇到的问题	19
3.7 安装 Hadoop	20
3.7.1 安装	20
3.7.2 测试	25
3.8 安装 HBase	29
3.8.1 安装	29
3.8.2 测试	32
3.8.3 可能遇到的问题	33
3.9 安装 Hive	33
3.9.1 安装	33
3.9.2 测试	36
3.10 安装 Scala	37
3.11 安装 Spark	37
3.11.1 安装	37
3.11.2 测试	39
3.12 安装 Storm	40

《大数据技术》实验一需要学生在笔记本上搭建 Hadoop 集群，本文档对笔记本上创建虚拟机搭建 Hadoop 集群的步骤进行了说明。包含所有需要安装的软件与服务的版本，安装路径，安装方法等。

主要流程为：安装虚拟机管理程序，创建三台虚拟服务器，在三台虚拟服务器上搭建以 Hadoop 集群为核心的大数据平台。

2. 服务列表

实验一要求学生搭建的大数据平台以 Hadoop 为核心，HDFS、MySQL、HBase 组成了存储系统，通过 Kafka 实时接入数据，使用 Storm 对数据进行实时分析，Map/Reduce 和 Spark 负责离线数据分析。Zookeeper 负责为分布式应用程序协调服务。

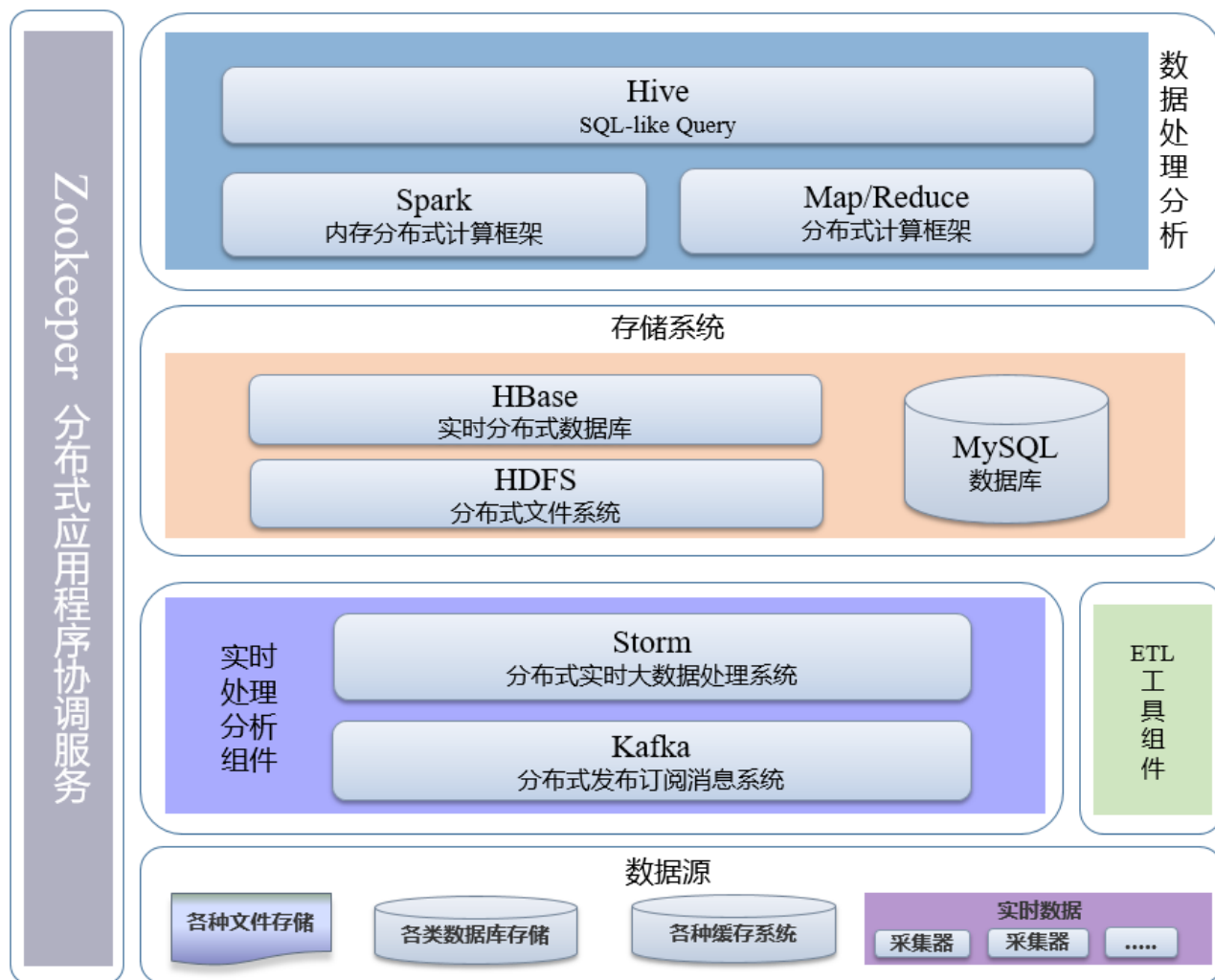


图 2-1 大数据平台架构图

用于搭建笔记本集群的虚拟机工具为 Oracle VM VirtualBox 5.1.26，使用其他工具也可用。

Hadoop 大数据平台所需工具、软件包、版本及安装目录见表 2-1。

表 2-1 Hadoop 大数据平台所需的软件包

软件包名	软件名称	版本	安装目录
jdk-7u80-linux-x64.tar.gz	java 软件开发工具包	1.7.80	/usr/local/jdk1.7.0_80
mysql-5.6.37-linux-glibc2.12-x86_64.tar.gz	MySQL	5.6.37	/usr/local/mysql
zookeeper-3.4.6.tar.gz	Zookeeper	3.4.6	/usr/local/zookeeper-3.4.6
kafka_2.10-0.8.2.1.tgz	Kafka	2.10.-0.8.2.1	/usr/local/kafka_2.10.-0.8.2.1

hadoop-2.6.5.tar.gz	Hadoop	2.6.5	/usr/local/hadoop-2.6.5
hbase-1.2.6-bin.tar.gz	HBase	1.2.6	/usr/local/hbase-1.2.6
apache-hive-1.1.0-bin.tar.gz	Hive	1.1.0	/usr/local/apache-hive-1.1.0-bin
mysql-connector-java-5.1.43-bin.jar	MySQL JDBC 驱动	5.1.43	
scala-2.10.6.tgz	Scala	2.10.6	/usr/local/scala-2.10.6
spark-1.6.3-bin-hadoop2.6.tgz	Spark	1.6.3	/usr/local/spark-1.6.3-bin-hadoop2.6
apache-storm-1.1.1.tar.gz	Storm	1.1.1	/usr/local/apache-storm-1.1.1

所有虚拟机上需要安装的软件与服务如表 2-2 所示。

表 2-2 虚拟机运行的服务列表

主机名	服务
cluster1	zookeeper, Kafka, HDFS(主), YARN(主), HBase(主), Hive, Spark(主), storm(主)
cluster2	zookeeper, Kafka, MySQL, HDFS, YARN, HBase, Hive, Spark, storm
cluster3	zookeeper, Kafka, HDFS, YARN, HBase, Hive, Spark, storm

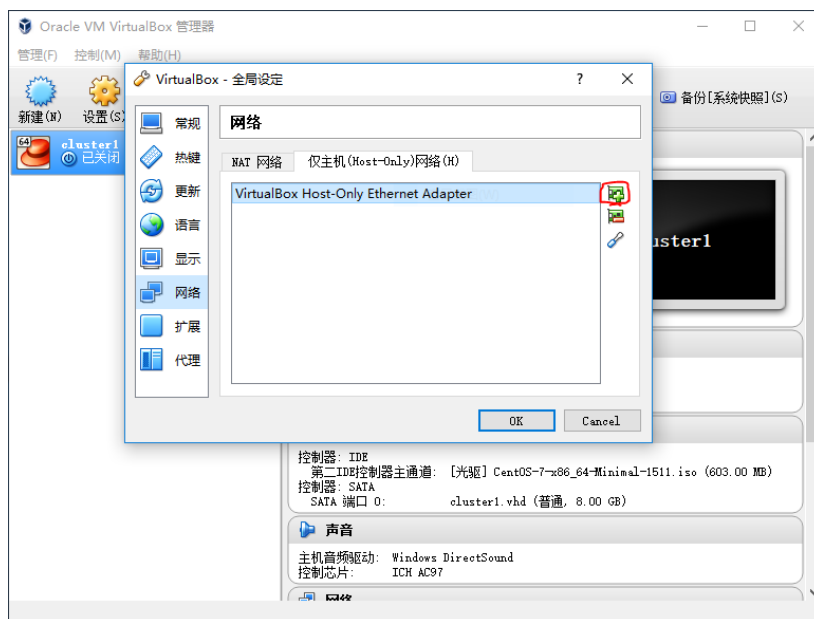
3. 搭建步骤

3.1 安装虚拟机

注意：请关闭 360 安全卫士等软件！

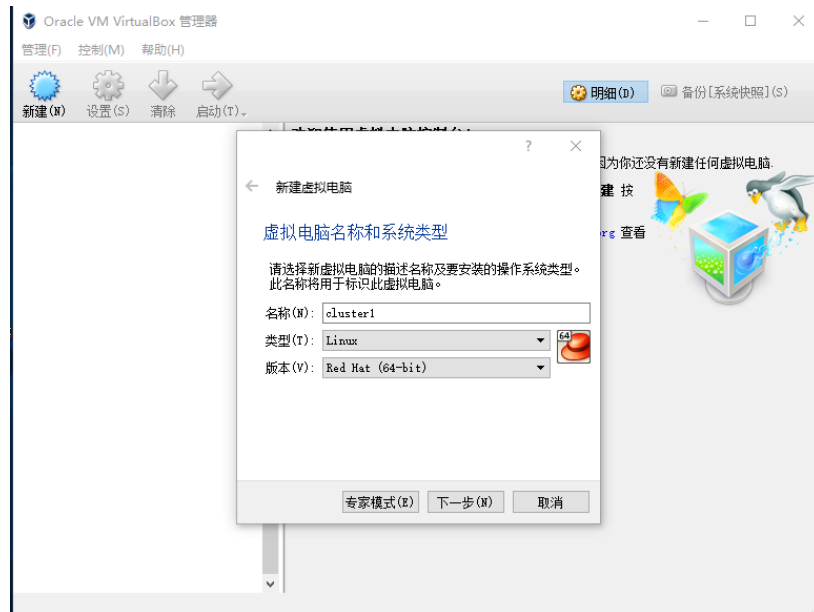
下载系统镜像，可从交大知行论坛上下载，CentOS-7-x86_64-Minimal-1511.iso。

虚拟机软件使用 Oracle VM VirtualBox 5.1.26。



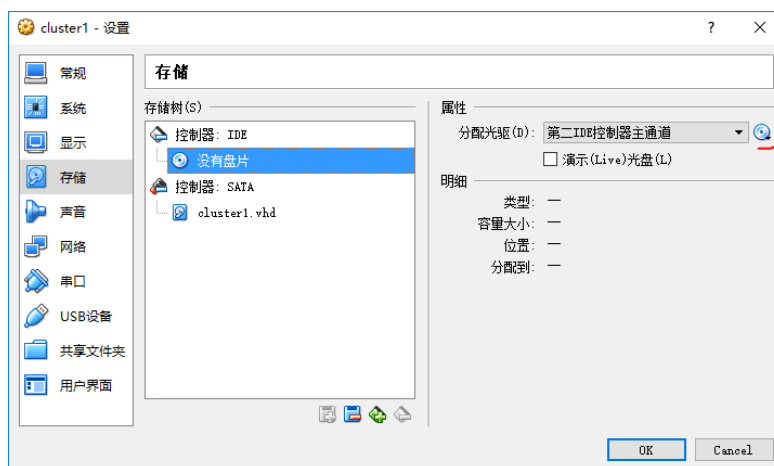
网络设置：选择左上角的管理->全局设定->网络->NAT 网络->右侧添加网卡->双击新增的网卡->网络 CIDR: 10.0.2.0/24->选 OK->仅主机(Host-Only)网络->右侧添加网卡->双击新增的网卡->IPv4 地址：192.168.56.1->IPv4 网络掩码：255.255.255.0->DHCP 服务器->取消勾选“启用服务器”->OK->点 OK 保存。

这里的 NAT 网络是为了创建出一个包含三台虚拟机的局域网环境，而 Host-Only 网络是为了使笔记本能与虚拟机通信，这样在虚拟机成功安装后，就可以使用 putty 或 secureCRT 等 SSH 工具从笔记本连接到虚拟机，然后将文档中提供的配置文件内容复制粘贴进去，若不配置 Host-Only 网络，本文档所有的配置文件全部需要手动输入。



内存 1536M，选择现在创建虚拟硬盘，虚拟硬盘文件类型选择 VHD(虚拟硬盘)，存储分配选择动态分配 (D)，大小选择 8G，然后选择创建。

右键选择创建好的虚拟机 cluster1，选择设置



存储->没有盘片->右侧的光盘图标，选择下载好的 CentOS-7-x86_64-Minimal-1511.iso，选择 OK。选择网络->网卡 1->启用网络连接->连接方式->NAT->界面名称->选择刚才先建立的那块虚拟网卡即可->网卡 2->启用网络连接->连接方式->仅主机(Host-Only)网络->界面名称->选择刚才后建立的那块虚拟网卡-> OK。

双击 cluster1 进入虚拟机，选择 Install CentOS7，选择 English->continue。

DATE&TIME，Region 选择 Asia，City 选择 Shanghai，左上角 Done。

INSTALLATION DESTINATION，点进去后直接点左上角的 Done。

NETWORK & HOST NAME，两张网卡右侧都选择 ON，下面的 Host name 改为 cluster1，左上角 Done。

选择右下角 Begin installation。

进入安装界面后选择 root password 随意。

右侧 user creation，Full name 设置为 cluster1，密码随意，不要选择 make this user administrator。

待下方进度条结束后，会有 reboot 这个选项，点击重启后，即可登陆。

再做两台这样的虚拟机分别是 cluster2 和 cluster3 即可。

3.2 准备工作

注：以下内容“//”后面的内容为下一行语句的注释，“#”后面的语句是使用 root 用户执行的，“\$”后面的语句是使用普通用户（3.2.5 创建的 hadoop 用户）执行的。**红色是必须要注意的内容！**

3.2.1 关闭防火墙和 Selinux

每台都要执行（我们安装的 centOS 最小版没有防火墙，在其他 centOS 上操作时必须关闭防火墙）

// 关闭防火墙和 selinux

systemctl stop firewalld.service

// 禁止 firewall 开机启动

systemctl disable firewalld.service

// 开机关闭 Selinux，编辑 Selinux 配置文件

vi /etc/selinux/config

将 SELINUX 设置为 disabled

如下：

SELINUX=disabled

// 重启

reboot

// 重启机器后 root 用户查看 Selinux 状态

getenforce

3.2.2 安装软件

以下软件是安装时需要的依赖环境，安装 MySQL 时需要使用 perl 和 libaio，ntupdate 负责集群内服务器时间，screen 用于新建后台任务。

每台都要执行

yum install perl*

yum install ntpdate

yum install libaio

yum install screen

3.2.3 检查网卡是否开机自启

每台都要执行

// 查看网卡名，看里面一个 enp0s 开头的是多少，由于我们开启了两块网卡，注意这两张都是什么名字

// 我的网卡名分别是 enp0s3 和 enp0s8，还有一个 lo,这个可以忽略不计。

ip addr

使用这条命令可以查看所有网卡的信息，注意记录每张网卡的网卡名：前面序号后面的即为网卡名。还需记录每张网卡的 ip 地址，inet 后面的就是。

```
[root@cluster1 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
   qlen 1000
    link/ether 08:00:27:82:6f:89 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 1079sec preferred_lft 1079sec
    inet6 fe80::a00:27ff:fe82:6f89/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
   qlen 1000
    link/ether 08:00:27:8f:05:ae brd ff:ff:ff:ff:ff:ff
[root@cluster1 ~]# _
```

注意：此处有两张网卡，分别是 2: enp0s3 和 3:enp0s8，如果没有，可能是因为在安装系统的过程中，没有打开网络服务，或是网卡没有设定为开机自启。

接下来编辑网卡配置文件

编辑的第一个网卡的配置文件，应该是 ip 为 10.0.2 开头的那张网卡，网卡名为 enp0s3

// 编辑网卡配置文件

```
# vi /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

// 确认 ONBOOT 为 yes，这个设置为 yes 后，该网卡会开机自启，不会开机连不上网
ONBOOT=yes

接下来编辑第二张网卡的配置文件，是 enp0s8

```
# vi /etc/sysconfig/network-scripts/ifcfg-enp0s8
```

将 BOOTPROTO 设置为 none

ONBOOT=yes

新增 IPADDR=192.168.56.121 (cluster2 设置为 192.168.56.122, cluster3 为 192.168.56.123)

NETMASK=255.255.255.0

NETWORK=192.168.56.0

保存后关闭文件。

// 重启网络服务

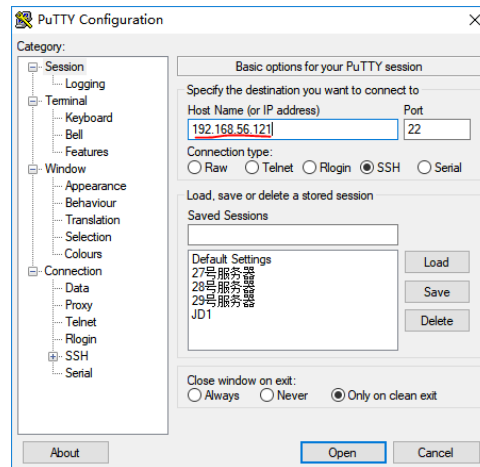
```
# service network restart
```

重启成功后会出现一个绿色的 OK，失败则显示红色的 failed，若失败，则使用 reboot 重启服务器即可。

重启后，就可以使用 putty 或其他 SSH 工具连接虚拟机了。

以下以 putty 为例：

在 windows 中，下载 putty，打开后，输入 192.168.56.121



然后点击右下角的 Open，即可连接到 cluster1 上。同样的，可以使用 Filezilla 传输文件。



主机为:sftp://192.168.56.121，用户名为 root，端口号不用设定，快速连接即可。

3.2.4 修改 hosts

每台都要执行

// 记录当前 ip 地址，要记录第一张网卡的 ip 地址

ip addr

// 修改 hosts

vi /etc/hosts

// 在最下面添加以下几行内容，下面的红色的 ip 地址写你用 ip addr 里面显示的第一张网卡(enp0s3)的 ip

10.0.2.6 cluster1

10.0.2.8 cluster2

10.0.2.7 cluster3

3.2.5 检查网络是否正常

// 在 cluster1 上

ping cluster2

如果出现如下界面

```
[root@cluster1 ~]# ping cluster2
PING cluster2 (10.0.2.5) 56(84) bytes of data.
64 bytes from cluster2 (10.0.2.5): icmp_seq=1 ttl=64 time=0.599 ms
64 bytes from cluster2 (10.0.2.5): icmp_seq=2 ttl=64 time=0.370 ms
64 bytes from cluster2 (10.0.2.5): icmp_seq=3 ttl=64 time=0.376 ms
64 bytes from cluster2 (10.0.2.5): icmp_seq=4 ttl=64 time=0.374 ms
^C
--- cluster2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 0.370/0.429/0.599/0.100 ms
[root@cluster1 ~]# _
```

可以看到 time=多少秒，说明可以从 cluster1 连接到 cluster2，同理，检查能否连接到 cluster3，使用 Ctrl+C 中断命令 ping。

```
// 检查能否连接到 cluster3
# ping cluster3
```

如果出现如下界面

```
[root@cluster1 ~]# ping cluster2
PING cluster2 (10.0.2.7) 56(84) bytes of data.
From cluster1 (10.0.2.4) icmp_seq=1 Destination Host Unreachable
From cluster1 (10.0.2.4) icmp_seq=2 Destination Host Unreachable
From cluster1 (10.0.2.4) icmp_seq=3 Destination Host Unreachable
From cluster1 (10.0.2.4) icmp_seq=4 Destination Host Unreachable
^C
--- cluster2 ping statistics ---
5 packets transmitted, 0 received, +4 errors, 100% packet loss, time 4003ms
pipe 4
```

可能导致该问题的原因：

1. hosts 配置有误，hosts 中的 cluster2 的 ip 地址，与 cluster2 的实际 ip 地址不同，请返回检查 cluster2 的 ip 地址。
2. cluster2 处于关机状态，没有开机。

3.2.6 新建 hadoop 用户

每台都要执行

新建 hadoop 用户，这个用户专门用来维护集群，因为实际中使用 root 用户的机会很少，而且不安全。

```
// 新建 hadoop 组
# groupadd hadoop

// 新建 hadoop 用户
# useradd -s /bin/bash -g hadoop -d /home/hadoop -m hadoop

// 修改 hadoop 这个用户的密码
# passwd hadoop
```

3.2.7 生成 ssh 密钥并分发

只在 cluster1 上执行

// 生成 ssh 密钥 (cluster1 上), 切换到 hadoop 用户

```
$ ssh-keygen -t rsa
```

然后一路回车

// 接下来分发密钥, 请仔细观察显示的内容, 会让你输入 yes

```
$ ssh-copy-id cluster1
```

```
$ ssh-copy-id cluster2
```

```
$ ssh-copy-id cluster3
```

3.2.8 安装 NTP 服务

// 三台都要安装

```
# yum install ntpdate
```

// cluster1 上装 ntp

```
# yum install ntp
```

// cluster1 上执行以下操作

```
# vi /etc/ntp.conf
```

注释掉以下 4 行, 也就是在这 4 行前面加 #

```
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
```

最下面加入以下内容, 红色部分分别为网关和掩码

```
restrict default ignore
```

```
restrict 10.0.2.0 mask 255.255.255.0 nomodify notrap
```

```
server 127.127.1.0
```

// 重启 ntp 服务

```
# service ntpd restart
```

// 设置 ntp 服务器开机自动启动

```
# chkconfig ntpd on
```

// 以下为客户端的配置 (除 cluster1 外其他所有的机器, 即 cluster2 和 cluster3):

设定每天 00:00 向服务器同步时间, 并写入日志

```
# crontab -e
```

输入以下内容后保存并退出:

```
0 0 * * * /usr/sbin/ntpdate cluster1 >> /root/ntpd.log
```

```
// 手动同步时间，需要在每台机器上（除 ntp server），使用 ntpdate cluster1 同步时间  
# ntpdate cluster1
```

3.3 安装 MySQL

3.3.1 安装

只在 cluster2 上做以下内容，因为我们的集群中，只有 cluster2 上需要安装一个 MySQL

```
# yum remove mysql mysql-server mysql-libs compat-mysql51  
# rm -rf /var/lib/mysql  
# rm -rf /etc/my.cnf
```

下载 [mysql-5.6.37-linux-glibc2.12-x86_64](#)

```
# cp mysql-5.6.37-linux-glibc2.12-x86_64.tar.gz /usr/local/
```

```
// 解压到/usr/local/
```

```
# tar -zxvf mysql-5.6.37-linux-glibc2.12-x86_64.tar.gz
```

```
// 改名为 mysql
```

```
# mv mysql-5.6.37-linux-glibc2.12-x86_64 mysql
```

```
// 删除安装包
```

```
# rm mysql-5.6.37-linux-glibc2.12-x86_64.tar.gz
```

```
// 修改环境变量
```

```
# vi /etc/profile
```

在最下面添加

```
export MYSQL_HOME=/usr/local/mysql
```

```
export PATH=$MYSQL_HOME/bin:$PATH
```

```
// 刷新环境变量
```

```
# source /etc/profile
```

```
// 新建 mysql 用户
```

```
# groupadd mysql      在/etc/group 中可以看到
```

```
# useradd -r -g mysql -s /bin/false mysql 在/etc/passwd 中可以看到
```

```
# cd /usr/local/mysql
```

```
# chown -R mysql:mysql .
```

```
# scripts/mysql_install_db --user=mysql
```

```
// 修改当前目录拥有者为 root 用户
```

```
# chown -R root .
```

```
// 修改当前 data 目录拥有者为 mysql 用户
```

```
# chown -R mysql data
```

```
# bin/mysqld_safe --user=mysql &
```

```
# cd /usr/local/mysql
```

```
// 登陆 mysql
```

```
# bin/mysql
```

```
// 登陆成功后退出即可
```

```
mysql> exit;
```

```
// 进行 root 账户密码的修改等操作
```

```
# bin/mysql_secure_installation
```

首先要求输入 root 密码，由于我们没有设置过 root 密码，括号里面说了，如果没有 root 密码就直接按回车。是否设定 root 密码，选 y，设定密码为 **cluster**，是否移除匿名用户：y。然后有个是否关闭 root 账户的远程登录，选 n，删除 test 这个数据库？y，更新权限？y，然后 ok。

```
# cp support-files/mysql.server /etc/init.d/mysql.server
```

```
// 查看 mysql 的进程号
```

```
# ps -ef | grep mysql
```

```
// 如果有的话就 kill 掉，保证 mysql 已经中断运行了，一般 kill 掉/usr/local/mysql/bin/mysqld 开头的即可
```

```
# kill 进程号
```

```
// 启动 mysql
```

```
# /etc/init.d/mysql.server start -user=mysql
```

```
# exit
```

还需要配置一下访问权限：

```
$ mysql -u root -p
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'cluster' WITH GRANT OPTION;
```

```
mysql> FLUSH PRIVILEGES;
```

```
// 关闭 mysql 的指令（不需要执行）
```

```
# mysqladmin -u root -p shutdown
```

3.3.2 测试

```
mysql> create database test_table;
```

```
mysql> use test_table;
```

```
mysql> create table userinfo(id int not null);
mysql> insert into userinfo values(1);
mysql> select * from userinfo;
mysql> drop database test_table;
mysql> show databases;
```

3.4 安装 JDK

3.4.1 安装

每台都要安装

```
$ su root
```

```
# cp jdk-7u80-linux-x64.tar.gz /usr/local/
```

```
# tar -zxvf jdk-7u80-linux-x64.tar.gz
```

```
// 修改环境变量
```

```
# vi /etc/profile
```

```
// 添加以下内容
```

```
export JAVA_HOME=/usr/local/jdk1.7.0_80/
```

```
export JRE_HOME=/usr/local/jdk1.7.0_80/jre
```

```
export CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
```

```
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$JAVA_HOME:$PATH
```

```
// 复制 jdk 到其他的服务器上
```

```
# scp -r /usr/local/jdk1.7.0_80/ cluster2:/usr/local/
```

```
# scp -r /usr/local/jdk1.7.0_80/ cluster3:/usr/local/
```

```
// cluster2 上
```

```
# vi /etc/profile
```

```
// 添加以下内容
```

```
export JAVA_HOME=/usr/local/jdk1.7.0_80/
```

```
export JRE_HOME=/usr/local/jdk1.7.0_80/jre
```

```
export CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
```

```
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$JAVA_HOME:$PATH
```

```
// cluster3 上
```

```
# vi /etc/profile
```

```
// 添加以下内容
```

```
export JAVA_HOME=/usr/local/jdk1.7.0_80/
```

```
export JRE_HOME=/usr/local/jdk1.7.0_80/jre
```



```
export CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$JAVA_HOME:$PATH
```

3.4.2 测试

```
$ java -version
```

可以看到 java 版本为 1.7.0_80 即为安装成功

3.5 安装 ZooKeeper

每台都要安装

// cluster1 上

将 zookeeper 解压到/usr/local 目录下，配置环境变量

```
# vi /etc/profile
```

// 添加以下内容

```
export ZOOKEEPER_HOME=/usr/local/zookeeper-3.4.6
```

```
export PATH=$ZOOKEEPER_HOME/bin:$PATH
```

```
# cd /usr/local/zookeeper-3.4.6
```

// 在 conf 中新建 zoo.cfg 文件

```
# vi conf/zoo.cfg
```

// 输入以下内容

客户端心跳时间(毫秒)

```
tickTime=2000
```

允许心跳间隔的最大时间

```
initLimit=10
```

同步时限

```
syncLimit=5
```

数据存储目录

```
dataDir=/home/hadoop_files/hadoop_data/zookeeper
```

数据日志存储目录

```
dataLogDir=/home/hadoop_files/hadoop_logs/zookeeper/dataLog
```

端口号

```
clientPort=2181
```

集群节点和服务端口配置

```
server.1=cluster1:2888:3888
```

```
server.2=cluster2:2888:3888
```

```
server.3=cluster3:2888:3888
```

// 创建 zookeeper 的数据存储目录和日志存储目录

```
# mkdir -p /home/hadoop_files/hadoop_data/zookeeper
# mkdir -p /home/hadoop_files/hadoop_logs/zookeeper/dataLog
# mkdir -p /home/hadoop_files/hadoop_logs/zookeeper/logs

// 修改文件夹的权限
# chown -R hadoop:hadoop /home/hadoop_files
# chown -R hadoop:hadoop /usr/local/zookeeper-3.4.6

// 在 cluster1 号服务器的 data 目录中创建一个文件 myid，输入内容为 1
// myid 应与 zoo.cfg 中的集群节点相匹配
# echo "1" >> /home/hadoop_files/hadoop_data/zookeeper/myid

// 修改 zookeeper 的日志输出路径(注意 CDH 版与原生版配置文件不同)
# vi bin/zkEnv.sh

// 将配置文件里面的以下项替换为红字的内容
if [ "x${ZOO_LOG_DIR}" = "x" ]
then
    ZOO_LOG_DIR="/home/hadoop_files/hadoop_logs/zookeeper/logs"
fi
if [ "x${ZOO_LOG4J_PROP}" = "x" ]
then
    ZOO_LOG4J_PROP="INFO,ROLLINGFILE"
fi

// 修改 zookeeper 的日志配置文件
# vi conf/log4j.properties

// 修改为以下内容:
zookeeper.root.logger=INFO,ROLLINGFILE
log4j.appender.ROLLINGFILE=org.apache.log4j.DailyRollingFileAppender

将这个 zookeeper-3.4.6 的目录复制到其他两个节点上
# scp -r /usr/local/zookeeper-3.4.6 cluster2:/usr/local/
# scp -r /usr/local/zookeeper-3.4.6 cluster3:/usr/local/

// 退回 hadoop 用户
# exit

// 刷新环境变量
$ source /etc/profile

// 启动 zookeeper
$ zkServer.sh start
```

```
// cluster2 上面 改环境变量，加入以下内容
export ZOOKEEPER_HOME=/usr/local/zookeeper-3.4.6
export PATH=$ZOOKEEPER_HOME/bin:$PATH

// 创建 zookeeper 的数据存储目录和日志存储目录
$ su root
# mkdir -p /home/hadoop_files/hadoop_data/zookeeper
# mkdir -p /home/hadoop_files/hadoop_logs/zookeeper/dataLog
# mkdir -p /home/hadoop_files/hadoop_logs/zookeeper/logs

// 添加 myid
# echo "2" >> /home/hadoop_files/hadoop_data/zookeeper/myid

// 修改文件夹的权限
# chown -R hadoop:hadoop /home/hadoop_files
# chown -R hadoop:hadoop /usr/local/zookeeper-3.4.6

// 退回 hadoop 用户
# exit

// 刷新环境变量
$ source /etc/profile

// 启动 zookeeper
$ zkServer.sh start

// cluster3 上面 加环境变量
# vi /etc/profile
export ZOOKEEPER_HOME=/usr/local/zookeeper-3.4.6
export PATH=$ZOOKEEPER_HOME/bin:$PATH

// 创建 zookeeper 的数据存储目录和日志存储目录
# mkdir -p /home/hadoop_files/hadoop_data/zookeeper
# mkdir -p /home/hadoop_files/hadoop_logs/zookeeper/dataLog
# mkdir -p /home/hadoop_files/hadoop_logs/zookeeper/logs

// 添加 myid
# echo "3" >> /home/hadoop_files/hadoop_data/zookeeper/myid

// 修改文件夹的权限
# chown -R hadoop:hadoop /home/hadoop_files
# chown -R hadoop:hadoop /usr/local/zookeeper-3.4.6
```

```
// 退回 hadoop 用户
```

```
# exit
```

```
// 刷新环境变量
```

```
$ source /etc/profile
```

```
// 启动 zookeeper（每台都要执行，而且三台要接连执行，都启动后再做下面的）
```

```
$ zkServer.sh start
```

```
// 三台 zookeeper 都启动后，使用 jps 命令查看进程是否启动
```

```
# jps
```

可以看到一个叫 QuorumPeerMain 的进程，说明 zookeeper 启动成功

```
// 退出 root 用户
```

```
# exit
```

```
// 查看 zookeeper 状态
```

```
$ zkServer.sh status
```

可以看到三台中有一个是 leader，两个是 follower

```
// 关闭 zookeeper 的命令（关机前在每台上都要执行，这里不需要执行）
```

```
$ zkServer.sh stop
```

3.6 安装 Kafka

3.6.1 安装

```
// cluster1 上
```

```
kafka_2.10-0.8.2.1 解压到/usr/local
```

```
//添加环境变量
```

```
export KAFKA_HOME=/usr/local/kafka_2.10-0.8.2.1
```

```
export PATH=$KAFKA_HOME/bin:$PATH
```

```
// 修改配置文件
```

```
# vi /usr/local/kafka_2.10-0.8.2.1/config/server.properties
```

```
// 修改下面 3 项
```

```
// 第一项：这个值要唯一，不同的机器不能相同，cluster1 就写 1，cluster2 就写 2，cluster3 就写 3  
broker.id=1
```

```
// 第二项：修改日志路径
```

```
log.dirs=/home/hadoop_files/hadoop_logs/kafka
```

// 第三项：此处要写 zookeeper 集群的 ip+端口号，逗号隔开

```
zookeeper.connect=cluster1:2181,cluster2:2181,cluster3:2181
```

// 第四项：此处要写对应机器的 ip 地址！

```
advertised.host.name=192.168.56.121
```

//修改完环境变量，更新配置文件

```
#source /etc/profile
```

// 保存退出后创建 logs 文件夹

```
# mkdir -p /home/hadoop_files/hadoop_logs/kafka
```

// 修改权限

```
# chown -R hadoop:hadoop /home/hadoop_files
```

```
# chown -R hadoop:hadoop /usr/local/kafka_2.10-0.8.2.1
```

// 复制文件夹

```
# scp -r /usr/local/kafka_2.10-0.8.2.1 cluster2:/usr/local/
```

```
# scp -r /usr/local/kafka_2.10-0.8.2.1 cluster3:/usr/local/
```

// cluster2 上

```
# vi /usr/local/kafka_2.10-0.8.2.1/config/server.properties
```

```
broker.id=2
```

// cluster3 上

```
# vi /usr/local/kafka_2.10-0.8.2.1/config/server.properties
```

```
broker.id=3
```

// 使用 **hadoop** 用户启动 kafka 集群

先启动 **zookeeper** 集群，然后在 kafka 集群中的每个节点使用

```
$ kafka-server-start.sh /usr/local/kafka_2.10-0.8.2.1/config/server.properties &
```

启动完成后按回车即可

3.6.2 测试

// 创建 topic

```
$ kafka-topics.sh --create --zookeeper cluster1:2181,cluster2:2181,cluster3:2181 --replication-factor 3 --partitions 1 --topic mykafka
```

// 查看 Topic:

```
$ kafka-topics.sh --list --zookeeper cluster1:2181,cluster2:2181,cluster3:2181
```

此时会显示 Topic: mykafka

// 查看详细信息

```
$ kafka-topics.sh --describe --zookeeper cluster1:2181,cluster2:2181,cluster3:2181
```

```
Topic:mykafka    PartitionCount:1    ReplicationFactor:3 Configs:
```

```
Topic: mykafka    Partition: 0    Leader: 133    Replicas: 133,134,132    Isr: 134
```

// 发送消息(cluster1 上执行)

```
$ kafka-console-producer.sh --broker-list localhost:9092 --topic mykafka
```

// 接收消息 (cluster2 上执行)

```
$ kafka-console-consumer.sh -zookeeper cluster1:2181,cluster2:2181,cluster3:2181 --topic mykafka --from-beginning
```

// 在 cluster1 输入以下内容

```
test
```

```
mycluster test
```

可以在 cluster2 上看到相应的信息

按 Ctrl+C 退出

// 关闭 kafka, 在每台上执行

```
$ kafka-server-stop.sh
```

//新建虚拟窗口 kafka, 在每台上执行

```
$ screen -S kafka
```

// 启动 kafka 集群, 在每台上执行

```
$ kafka-server-start.sh /usr/local/kafka_2.10-0.8.2.1/config/server.properties
```

// 退出虚拟窗口 kafka, 在每台上执行

```
$ Ctrl+A+D
```

在每台服务器上执行 jps 可以看到 Kafka 进程在运行

// 关闭 kafka 的命令为, 在每台服务器上进入虚拟窗口 kafka, 然后使用 kafka-server-stop.sh 即可。

3.6.3 可能遇到的问题

1. 如果出现以下信息, 则需要下载 slf4j-nop-1.5.jar, 并将其复制至 kafka 的 libs 目录下:

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
```

```
SLF4J: Defaulting to no-operation (NOP) logger implementation
```

```
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

2. Error: while executing topic command replication factor: 3 larger than available brokers: 1

这是因为没有将 3 台机器的 kafka 都启动。

3.7 安装 Hadoop

3.7.1 安装

Hadoop 启动的先决条件是 zookeeper 已经成功启动

// 在 cluster1 节点/usr/local/解压 hadoop 安装包

```
$ su root
```

```
# tar -zxvf hadoop-2.6.5.tar.gz
```

// 删除安装包

```
# rm hadoop-2.6.5.tar.gz
```

// 切换到存有 hadoop 配置文件的目录

```
# cd /usr/local/hadoop-2.6.5/etc/hadoop
```

// 修改 hadoop-env.sh 文件

```
# vi hadoop-env.sh
```

将 export JAVA_HOME=\${JAVA_HOME} 替换为 export JAVA_HOME=/usr/local/jdk1.7.0_80

```
export HADOOP_PID_DIR=/home/hadoop_files
```

// 配置 mapred-env.sh

```
# vi mapred-env.sh
```

```
export HADOOP_MAPRED_PID_DIR=/home/hadoop_files
```

// 配置 core-site.xml 文件

```
# vi core-site.xml
```

```
<configuration>
```

```
<!-- 指定 hdfs 的 nameservices 名称为 mycluster，与 hdfs-site.xml 的 HA 配置相同 -->
```

```
<property>
```

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

```
<value>hdfs://cluster1:9000</value>
```

```
</property>
```

```
<!-- 指定缓存文件存储的路径 -->
```

```
<property>
```

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

```
<value>/home/hadoop_files/hadoop_tmp/hadoop/data/tmp</value>
```

```
</property>
```

```
<!-- 配置 hdfs 文件被永久删除前保留的时间（单位：分钟），默认值为 0 表明垃圾回收站功能关闭 -->
```

```
<property>
```

```
<name>fs.trash.interval</name>
<value>1440</value>
</property>
```

```
<!-- 指定 zookeeper 地址，配置 HA 时需要 -->
<property>
  <name>ha.zookeeper.quorum</name>
  <value>cluster1:2181,cluster2:2181,cluster3:2181</value>
</property>
</configuration>
```

// 配置 hdfs-site.xml 文件

vi hdfs-site.xml

```
<configuration>
  <!-- 指定 hdfs 元数据存储的路径 -->
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/hadoop_files/hadoop_data/hadoop/namenode</value>
  </property>

  <!-- 指定 hdfs 数据存储的路径 -->
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/hadoop_files/hadoop_data/hadoop/datanode</value>
  </property>

  <property>
    <name>dfs.secondary.http.address</name>
    <value>cluster1:50090</value>
  </property>

  <!-- 数据备份的个数 -->
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>

  <!-- 关闭权限验证 -->
  <property>
    <name>dfs.permissions.enabled</name>
    <value>false</value>
  </property>

  <!-- 开启 WebHDFS 功能（基于 REST 的接口服务） -->
```



```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
</configuration>
```

// 配置 mapred-site.xml 文件

vi mapred-site.xml

```
<configuration>
  <!-- 指定 MapReduce 计算框架使用 YARN -->
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <!-- 指定 jobhistory server 的 rpc 地址 -->
  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>cluster1:10020</value>
  </property>

  <!-- 指定 jobhistory server 的 http 地址 -->
  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>cluster1:19888</value>
  </property>
</configuration>
```

// 配置 yarn-site.xml 文件

vi yarn-site.xml

```
<configuration>
  <!-- NodeManager 上运行的附属服务，需配置成 mapreduce_shuffle 才可运行 MapReduce 程序 -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>

  <!-- 配置 Web Application Proxy 安全代理（防止 yarn 被攻击） -->
  <property>
    <name>yarn.web-proxy.address</name>
    <value>cluster2:8888</value>
  </property>

  <!-- 开启日志 -->
```

```
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
</property>

<!-- 配置日志删除时间为 7 天，-1 为禁用，单位为秒 -->
<property>
  <name>yarn.log-aggregation.retain-seconds</name>
  <value>604800</value>
</property>

<!-- 修改日志目录 -->
<property>
  <name>yarn.nodemanager.remote-app-log-dir</name>
  <value>/home/hadoop_files/hadoop_logs/yarn</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>cluster1:8032</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>cluster1:8030</value>
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>cluster1:8031</value>
</property>
</configuration>
```

// 配置 slaves 文件

vi slaves

删除 localhost

添加以下内容：

cluster1

cluster2

cluster3

// 创建配置文件中涉及的目录(在所有结点上)

mkdir -p /home/hadoop_files/hadoop_data/hadoop/namenode

mkdir -p /home/hadoop_files/hadoop_data/hadoop/datanode

mkdir -p /home/hadoop_files/hadoop_tmp/hadoop/data/tmp

mkdir -p /home/hadoop_files/hadoop_logs/yarn

// 修改文件夹权限（在所有结点上）

```
# chown -R hadoop:hadoop /home/hadoop_files/
```

```
# chown -R hadoop:hadoop /usr/local/hadoop-2.6.5/
```

// 将 cluster1 的 hadoop 工作目录同步到集群其它节点

```
$ scp -r /usr/local/hadoop-2.6.5 cluster2:/usr/local/
```

```
$ scp -r /usr/local/hadoop-2.6.5 cluster3:/usr/local/
```

// 修改文件夹权限（在所有结点上）

```
# chown -R hadoop:hadoop /usr/local/hadoop-2.6.5/
```

// 在集群各节点上修改环境变量

```
# vi /etc/profile
```

```
export HADOOP_HOME=/usr/local/hadoop-2.6.5
```

```
export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native
```

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

// 使修改的环境变量生效

```
$ source /etc/profile
```

// 启动 zookeeper 集群（分别在 cluster1, cluster2 和 cluster3 上执行）

```
$ zkServer.sh start
```

接下来开始格式化：

// 启动 journalnode（在所有 datanode 上执行，也就是 cluster1, cluster2, cluster3）

```
$ hadoop-daemon.sh start journalnode
```

启动后使用 jps 命令可以看到 JournalNode 进程

// 格式化 HDFS（在 cluster1 上执行）

```
$ hdfs namenode -format
```

// 格式化完毕后可关闭 journalnode（在所有 datanode 上执行）

```
$ hadoop-daemon.sh stop journalnode
```

// 启动 HDFS（cluster1 上）

```
$ start-dfs.sh
```

// 启动后 cluster1 上使用 jps 可以看到 NameNode, DataNode, SecondaryNameNode

cluster2 和 cluster3 上可以看到 DataNode

```
$ jps
```

// 启动 YARN（cluster1 上）

```
$ start-yarn.sh
```

```
// 启动后 cluster1 上使用 jps 可以看到 NodeManager, ResourceManager  
cluster2 和 cluster3 上可以看到 NodeManager  
$ jps
```

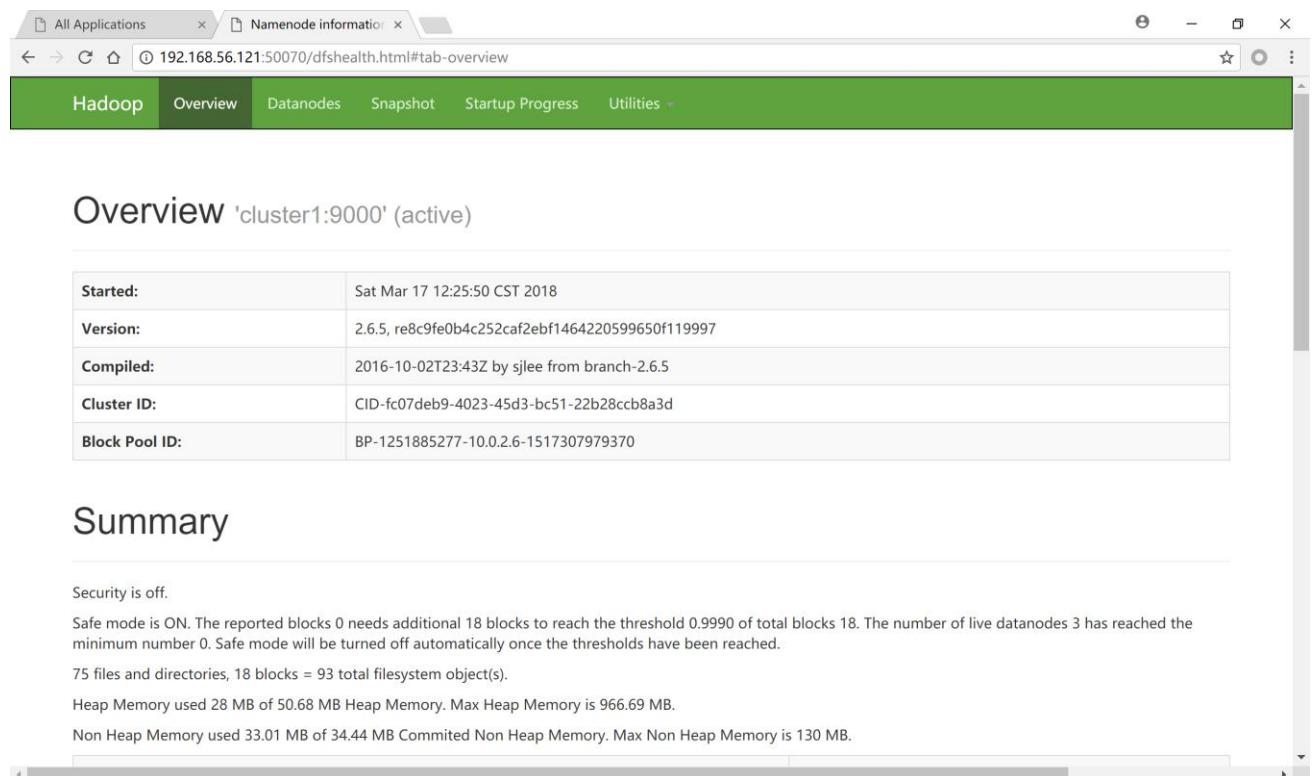
// 以下两条命令是关闭 YARN 和 HDFS 的命令，重启服务器或关机前一定要执行，否则有可能导致数据或服务损坏

```
// 关闭 YARN 的命令（cluster1 上）  
$ stop-yarn.sh
```

```
// 关闭 HDFS 的命令（cluster1 上）  
$ stop-dfs.sh
```

3.7.2 测试

启动 HDFS 后，可以在浏览器中，打开 192.168.56.121:50070，可以看到 HDFS 的 web 界面



The screenshot shows the Hadoop DFS Health Overview page. The browser address bar shows the URL: 192.168.56.121:50070/dfshealth.html#tab-overview. The page has a green header with tabs: Hadoop, Overview (selected), Datanodes, Snapshot, Startup Progress, and Utilities. The main content area is titled 'Overview 'cluster1:9000' (active)'. It contains a table with the following information:

Started:	Sat Mar 17 12:25:50 CST 2018
Version:	2.6.5, re8c9fe0b4c252caf2ebf1464220599650f119997
Compiled:	2016-10-02T23:43Z by sjlee from branch-2.6.5
Cluster ID:	CID-fc07deb9-4023-45d3-bc51-22b28ccb8a3d
Block Pool ID:	BP-1251885277-10.0.2.6-1517307979370

Below the table is a 'Summary' section. It states: 'Security is off.' and 'Safe mode is ON. The reported blocks 0 needs additional 18 blocks to reach the threshold 0.9990 of total blocks 18. The number of live datanodes 3 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.' It also shows: '75 files and directories, 18 blocks = 93 total filesystem object(s).', 'Heap Memory used 28 MB of 50.68 MB Heap Memory. Max Heap Memory is 966.69 MB.', and 'Non Heap Memory used 33.01 MB of 34.44 MB Committed Non Heap Memory. Max Non Heap Memory is 130 MB.'

第一页 Overview 是当前 HDFS 的概况，里面显示了 HDFS 的启动时间，版本等信息。
点击上面标签栏的第二项 Datanodes，可以看到如下界面

The screenshot shows the Hadoop web interface with the 'Datanode Information' tab selected. The page title is 'Datanode Information'. Below the title, there is a section 'In operation' which contains a table of datanodes. The table has columns: Node, Last contact, Admin State, Capacity, Used, Non DFS Used, Remaining, Blocks, Block pool used, Failed Volumes, and Version. There are three rows of data for cluster1, cluster2, and cluster3. Below this, there is a section 'Decommissioning' which is currently empty. At the bottom, there is a footer with 'Hadoop, 2016.' and 'Legacy UI'.

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
cluster1 (10.0.2.6:50010)	2	In Service	6.66 GB	4 KB	5.81 GB	871.8 MB	0	4 KB (0%)	0	2.6.5
cluster2 (10.0.2.8:50010)	1	In Service	6.66 GB	4 KB	4.53 GB	2.13 GB	0	4 KB (0%)	0	2.6.5
cluster3 (10.0.2.7:50010)	1	In Service	6.66 GB	4 KB	4.26 GB	2.4 GB	0	4 KB (0%)	0	2.6.5

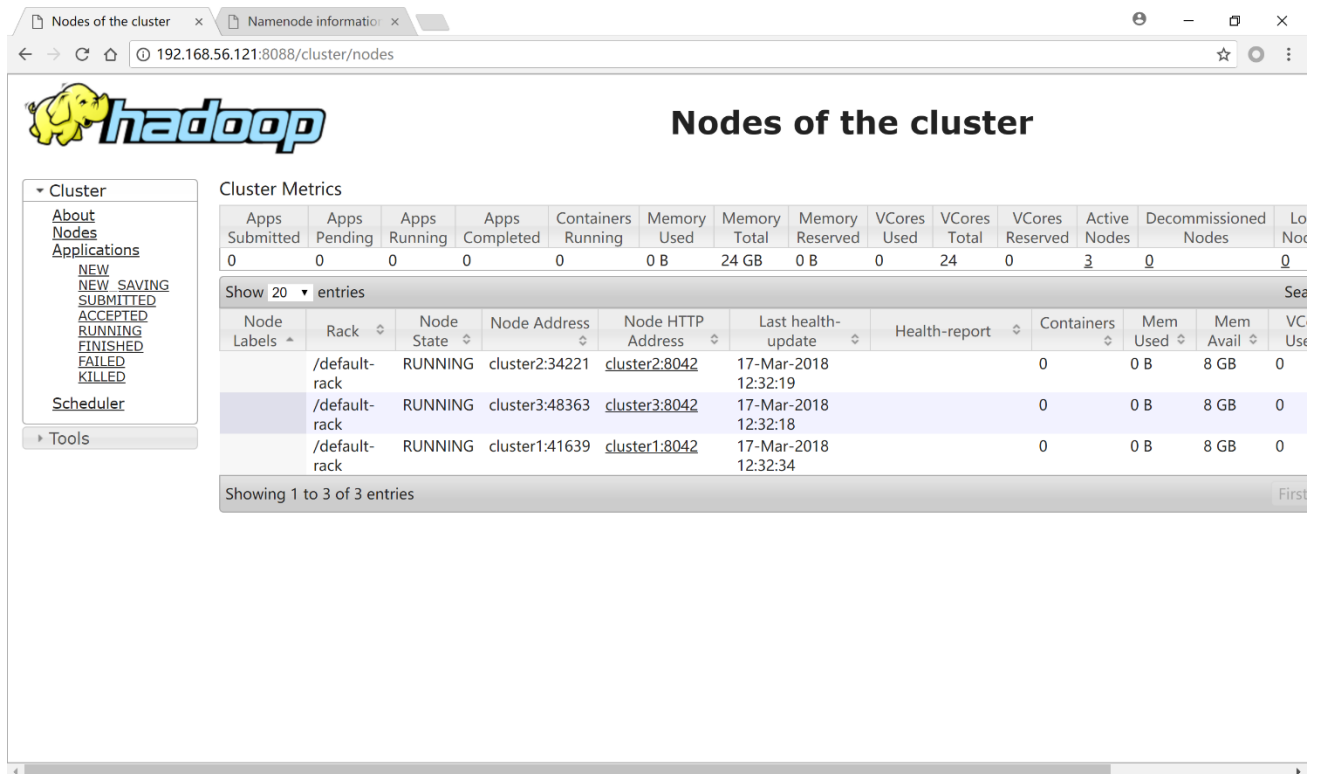
这个页面显示了当前 HDFS 中的可用节点。

启动 YARN 后，可以通过浏览器访问 192.168.56.121:8088，查看 YARN 的 web 界面

The screenshot shows the Hadoop YARN web interface with the 'All Applications' tab selected. The page title is 'All Applications'. On the left, there is a sidebar with a 'Cluster' section containing links for 'About', 'Nodes', 'Applications', 'NEW', 'NEW SAVING', 'SUBMITTED', 'ACCEPTED', 'RUNNING', 'FINISHED', 'FAILED', 'KILLED', 'Scheduler', and 'Tools'. The main content area shows 'Cluster Metrics' with a table of application metrics. Below this, there is a table of applications with columns: ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, FinalStatus, Progress, and Tracking UI. The table is currently empty, showing 'Showing 0 to 0 of 0 entries'.

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes
0	0	0	0	0	0 B	16 GB	0 B	0	16	0	2	0	0

该页面展示了所有提交到 YARN 上的程序，点击左侧的 Nodes 可以看到 YARN 的节点



The screenshot shows the Hadoop web interface at 192.168.56.121:8088/cluster/nodes. The title is "Nodes of the cluster". On the left is a navigation menu with "Cluster" selected, showing sub-items like "About", "Nodes", "Applications", and "Scheduler". The main content area displays "Cluster Metrics" and a table of nodes.

Cluster Metrics													
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Local Storage
0	0	0	0	0	0 B	24 GB	0 B	0	24	0	3	0	0

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used
/default-rack		RUNNING	cluster2:34221	cluster2:8042	17-Mar-2018 12:32:19		0	0 B	8 GB	0
/default-rack		RUNNING	cluster3:48363	cluster3:8042	17-Mar-2018 12:32:18		0	0 B	8 GB	0
/default-rack		RUNNING	cluster1:41639	cluster1:8042	17-Mar-2018 12:32:34		0	0 B	8 GB	0

Showing 1 to 3 of 3 entries

注意，此处可以看到每个节点的可用内存 **Mem Avail** 为 8G，我们的虚拟机每台内存只有 1.5G，之所以出现这个问题是因为我们没有在 `yarn-site.xml` 这个文件中对节点的可用内存进行配置，可以增加一下内容进行配置：

```
<!-- 配置 nodemanager 可用的资源内存 -->
<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>20480</value>
</property>

<!-- 配置 nodemanager 可用的资源 CPU -->
<property>
  <name>yarn.nodemanager.resource.cpu-vcores</name>
  <value>24</value>
</property>
```

由于我们的虚拟机没有那么大的内存空间以及 CPU 资源，所以我们暂时不设置这两个参数，在后续实验中，需要同学们设置这个参数以达到集群性能的最大化。

命令行测试：

```
// cluster1
// 切换至 hadoop 用户的主目录
$ cd ~/

// 新建一个测试文件
$ vi testfile
输入
```

1

2

3

// 保存退出

// 查看 HDFS 根目录的文件

\$ hdfs dfs -ls /

// 在 HDFS 的根目录创建 test 目录

\$ hdfs dfs -mkdir /test

// 如果出现了 mkdir: Cannot create directory /test. Name node is in safe mode.说明 HDFS 刚启动不久，还在安全检查中。由于我们的笔记本性能不够强，安全检查的时间会很长，可以使用命令退出安全模式，看到 Safe mode is OFF，再执行上面的创建目录的命令

\$ hdfs dfsadmin -safemode leave

```
[hadoop@cluster1 ~]$ hdfs dfsadmin -safemode leave
Safe mode is OFF
```

// 创建完文件夹后再次查看根目录，查看目录是否新建成功

\$ hdfs dfs -ls /

// 将测试文件 testfile 上传至 HDFS 根目录下的 test 目录中

\$ hdfs dfs -put testfile /test

// 在 cluster2 上

// 切换至 hadoop 用户的主目录

\$ cd ~/

// 查看 HDFS 根目录

\$ hdfs dfs -ls /

// 查看 HDFS 根目录下的 test 目录，可以看到我们刚才在 cluster1 上上传的文件 testfile

\$ hdfs dfs -ls /test

// 查看 HDFS 上的/test/testfile 文件的内容

\$ hdfs dfs -cat /test/testfile

// 将 HDFS 上的/test/testfile 下载到本地

\$ hdfs dfs -get /test/testfile

// 查看本地当前目录内的内容，可以看到刚才下载的 testfile

\$ ls

3.8 安装 HBase

3.8.1 安装

HBase 启动的先决条件是 zookeeper 和 Hadoop 已经启动

// 切换至 root 用户

```
$ su root
```

// 在 cluster1 节点/usr/local/解压 hbase 安装包

```
# tar -zxvf hbase-1.2.6-bin.tar.gz
```

// 进入 hbase 的 conf 目录

```
#cd /usr/local/hbase-1.2.6/conf/
```

// 修改 hbase-env.sh

```
# vi hbase-env.sh
```

将 JAVA_HOME, HADOOP_HOME, HBASE_LOG_DIR, HBASE_MANAGES_ZK 修改为以下内容:
记得去掉前面的#

```
# 配置 JDK 安装路径
```

```
export JAVA_HOME=/usr/local/jdk1.7.0_80
```

```
# 配置 Hadoop 安装路径
```

```
export HADOOP_HOME=/usr/local/hadoop-2.6.5
```

```
# 设置 HBase 的日志目录
```

```
export HBASE_LOG_DIR=/home/hadoop_files/hadoop_logs/hbase/logs
```

```
# 使用独立的 ZooKeeper 集群
```

```
export HBASE_MANAGES_ZK=false
```

```
# 设置 pid 的路径
```

```
export HBASE_PID_DIR=/home/hadoop_files
```

// 配置 hbase-site.xml

```
# vi hbase-site.xml
```

```
<configuration>
```

```
    <property>
```

```
        <name>hbase.rootdir</name>
```

```
        <value>hdfs://cluster1:9000/hbase</value>
```

```
    </property>
```

```
    <property>
```

```
        <name>hbase.cluster.distributed</name>
```

```
        <value>true</value>
```

```
    </property>
```

```
</property>
```



```
<name>hbase.master</name>
<value>60000</value>
</property>

<property>
  <name>hbase.tmp.dir</name>
  <value>/home/hadoop_files/hadoop_tmp/hbase/tmp</value>
</property>

<property>
  <name>hbase.zookeeper.quorum</name>
  <value>cluster1,cluster2,cluster3</value>
</property>

<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/home/hadoop_files/hadoop_data/zookeeper</value>
</property>

<property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2181</value>
</property>

<property>
  <name>zookeeper.session.timeout</name>
  <value>120000</value>
</property>

<property>
  <name>hbase.regionserver.restart.on.zk.expire</name>
  <value>true</value>
</property>

<property>
  <name>hbase.master.info.port</name>
  <value>60010</value>
</property>

</configuration>

// 配置 regionservers
# vi regionservers
删除 localhost
```

添加:

cluster1

cluster2

cluster3

// 删除 hbase 的 slf4j-log4j12-1.7.5.jar, 解决 hbase 和 hadoop 的 LSF4J 包冲突

cd /usr/local/hbase-1.2.6/lib

mv slf4j-log4j12-1.7.5.jar slf4j-log4j12-1.7.5.jar.bk

// 将 hbase 工作目录同步到集群其它节点

scp -r /usr/local/hbase-1.2.6/ cluster2:/usr/local/

scp -r /usr/local/hbase-1.2.6/ cluster3:/usr/local/

// 创建 hbase 的缓存文件目录 (所有节点)

mkdir -p /home/hadoop_files/hadoop_tmp/hbase/tmp

// 创建 hbase 的日志文件目录 (所有节点)

\$ mkdir -p /home/hadoop_files/hadoop_logs/hbase/logs

// 改权限 (所有节点)

chown -R hadoop:hadoop /usr/local/hbase-1.2.6

chown -R hadoop:hadoop /home/hadoop_files

// 在集群各节点上修改环境变量

vi /etc/profile

export HBASE_HOME=/usr/local/hbase-1.2.6

export PATH=\$HBASE_HOME/bin:\$PATH

\$ source /etc/profile

// 启动 HBase (cluster1 上)

先启动 zookeeper, Hadoop 的 HDFS 和 YARN, 然后才能启动 HBase

// 启动 HDFS (cluster1 上)

\$ start-dfs.sh

// 启动 YARN (cluster1 上)

\$ start-yarn.sh

// 启动 HBase (cluster1 上)

\$ start-hbase.sh

启动后 cluster1 上使用 jps 可以看到 HMaster 和 HRegionServer

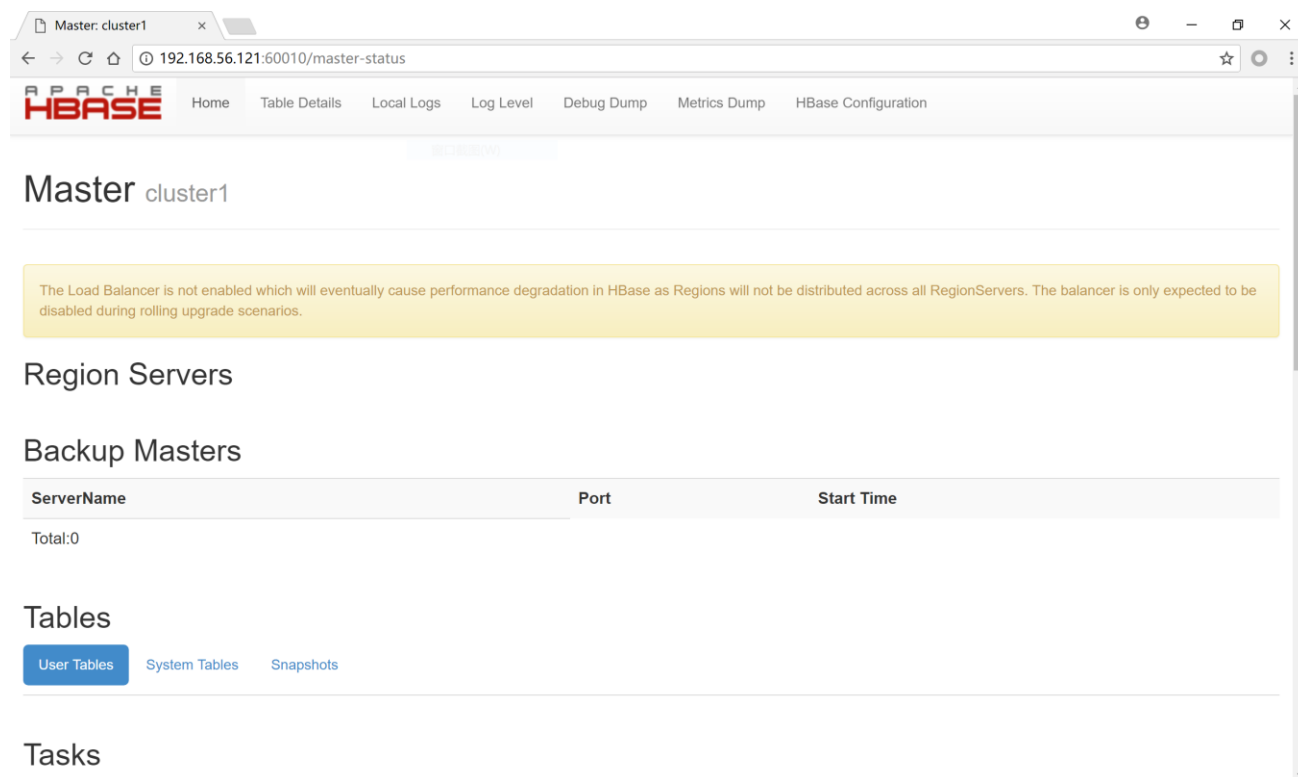
cluster2 和 cluster3 上可以看到 HRegionServer

// 关机前执行关闭 HBase 的命令 (cluster1 上)

\$ stop-hbase.sh

3.8.2 测试

用浏览器打开 192.168.56.121:60010，可以看到 HBase 的 web 界面



测试 HBase:

// cluster1 上

\$ hbase shell

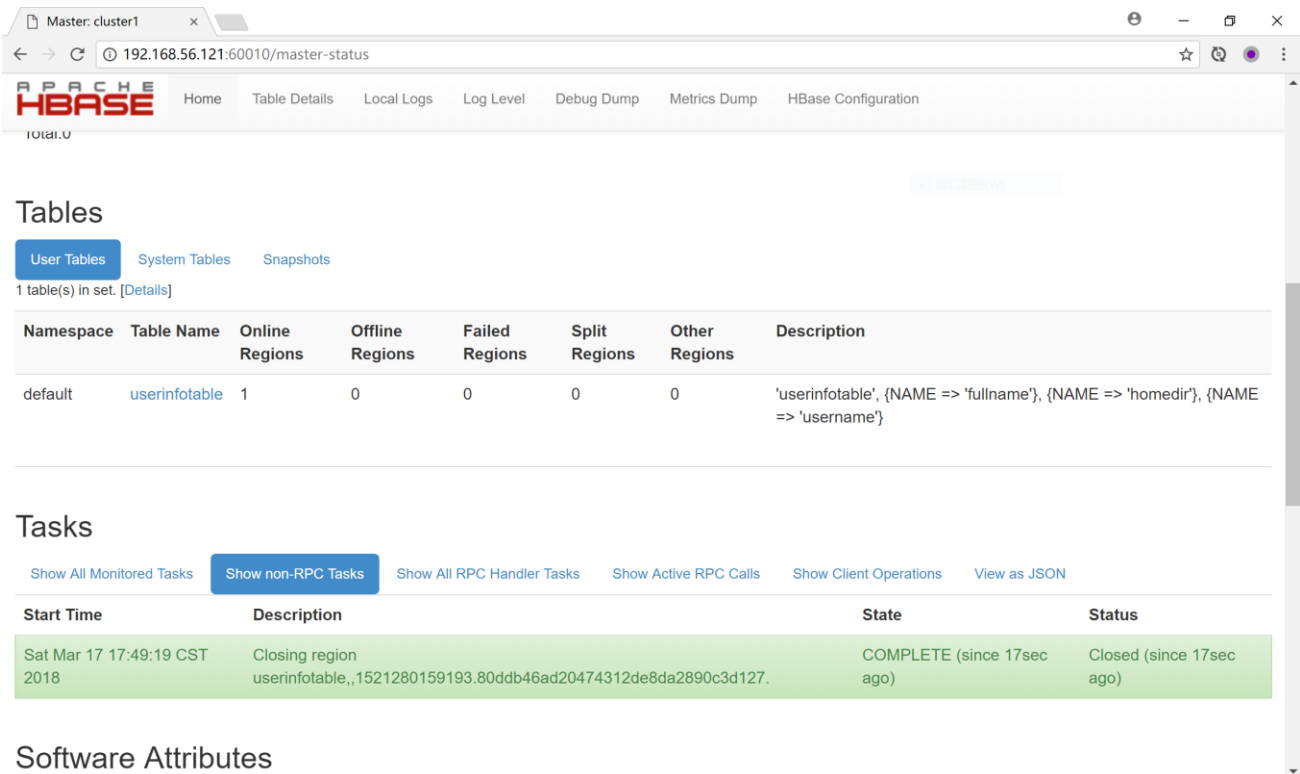
```
hbase> create 'userinfotable',{NAME=>'username'},{NAME=>'fullname'},{NAME=>'homedir'}
```

```
hbase> put 'userinfotable','r1','username','vcsa'
```

```
hbase> put 'userinfotable','r2','username','sasuser'
```

```
hbase> scan 'userinfotable'
```

可以看到刚才插入的信息，在 web 界面也可以看到刚才建立的表。



Master: cluster1

192.168.56.121:60010/master-status

APACHE HBASE

Home Table Details Local Logs Log Level Debug Dump Metrics Dump HBase Configuration

Tables

User Tables System Tables Snapshots

1 table(s) in set. [Details]

Namespace	Table Name	Online Regions	Offline Regions	Failed Regions	Split Regions	Other Regions	Description
default	userinfotable	1	0	0	0	0	'userinfotable', {NAME => 'fullname'}, {NAME => 'homedir'}, {NAME => 'username'}

Tasks

Show All Monitored Tasks Show non-RPC Tasks Show All RPC Handler Tasks Show Active RPC Calls Show Client Operations View as JSON

Start Time	Description	State	Status
Sat Mar 17 17:49:19 CST 2018	Closing region userinfotable,,1521280159193.80ddb46ad20474312de8da2890c3d127.	COMPLETE (since 17sec ago)	Closed (since 17sec ago)

Software Attributes

删除刚才建立的表：

```
hbase> disable 'userinfotable'
hbase> drop 'userinfotable'
hbase> exit
```

3.8.3 可能遇到的问题

1. HRegionServer 闪退，在 cluster1 上使用 start-hbase.sh 启动后，用 jps 查看进程，可以看到 HRegionServer，过一会儿再看就消失了。
解决方法：有可能是时间不同步导致的问题，检查三台服务器的时间是否还同步。

3.9 安装 Hive

3.9.1 安装

以下内容除在 MySQL 中创建 hive 用户和创建 hive 数据库只用操作一次，其他操作需要在每个 Hadoop 结点上都执行一次。

注：hive 能启动的先决条件是 MySQL 已经安装并配置完成，而且 HDFS 也要启动之后才能运行 hive

```
$ su root
# cp apache-hive-1.1.0-bin.tar.gz /usr/local
# cd /usr/local
# tar -zxvf ./apache-hive-1.1.0-bin.tar.gz
# vi /etc/profile
```

```
// 在下面加上两行:
export HIVE_HOME=/usr/local/apache-hive-1.1.0-bin
export PATH=$HIVE_HOME/bin:$HIVE_HOME/conf:$PATH

// root 用户登陆 MySQL
# mysql -u root -p
// 创建用户 hive, 密码 hive
mysql> GRANT USAGE ON *.* TO 'hive'@'%' IDENTIFIED BY 'hive' WITH GRANT OPTION;
// 创建数据库 hive
mysql> create database hive;
// 允许任意 ip 以 hive 登陆数据库
mysql> grant all on hive.* to hive@'%' identified by 'hive';
mysql> grant all on hive.* to hive@'localhost' identified by 'hive';
mysql> grant all on hive.* to hive@'cluster2' identified by 'hive';
// 刷新权限
mysql> flush privileges;
// 退出
mysql> exit;

// 验证 hive 用户, 密码 hive
# mysql -u hive -p
// 查看当前的数据库
mysql> show databases;
若看到下面有 hive 这个库, 则说明创建成功
+-----+
| Database                |
+-----+
| information_schema      |
| hive                    |
+-----+
2 rows in set (0.00 sec)
// 退出 mysql
mysql> exit;

// 修改 hive-site.xml
# cp apache-hive-1.1.0-bin/conf/hive-default.xml.template apache-hive-1.1.0-bin/conf/hive-site.xml
# vi apache-hive-1.1.0-bin/conf/hive-site.xml
找到以下 property 项, value 值修改成如下, 其他的在 hive-site.xml 中出现, 但是下文没出现的就不需要更改了:
<property>
  <name>javax.jdo.option.ConnectionURL </name>
  <value>jdbc:mysql://cluster2:3306/hive</value>
</property>
```

```
<property>
  <name>javax.jdo.option.ConnectionDriverName </name>
  <value>com.mysql.jdbc.Driver </value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword </name>
  <value>hive</value>
</property>

<property>
  <name>hive.hwi.listen.port</name>
  <value>9999 </value>
  <description>This is the port the Hive Web Interface will listen on</description>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>true</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>false</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hive</value>
  <description>Username to use against metastore database</description>
</property>

<property>
  <name>hive.exec.local.scratchdir</name>
  <value>/home/hadoop_files/hadoop_tmp/hive/iotmp</value>
  <description>Local scratch space for Hive jobs</description>
</property>

<property>
  <name>hive.downloaded.resources.dir</name>
  <value>/home/hadoop_files/hadoop_tmp/hive/iotmp</value>
  <description>Temporary local directory for added resources in the remote file system.</description>
</property>
```

```
<property>
  <name>hive.querylog.location</name>
  <value>/home/hadoop_files/hadoop_logs/hive/querylog</value>
  <description>Location of Hive run time structured log file</description>
</property>
```

// 拷贝 mysql-connector-java-5.1.43-bin.jar 到 hive 的 lib 下面

```
# cp mysql-connector-java-5.1.43-bin.jar /usr/local/apache-hive-1.1.0-bin/lib/
```

// 把 jline-2.12.jar 拷贝到 hadoop 相应的目录下，替代 jline-0.9.94.jar，否则启动会报错

```
# cp /usr/local/apache-hive-1.1.0-bin/lib/jline-2.12.jar /usr/local/hadoop-2.6.5/share/hadoop/yarn/lib/
```

// 切换到 hadoop 目录中 share/hadoop/yarn/lib

```
# cd /usr/local/hadoop-2.6.5/share/hadoop/yarn/lib/
```

// 将 hadoop-2.6.5/share/hadoop/yarn/lib/里面的 jline-0.9.94 重命名

```
# mv jline-0.9.94.jar jline-0.9.94.jar.bak
```

// 创建 hive 临时文件夹和日志目录

```
# mkdir -p /home/hadoop_files/hadoop_tmp/hive/iotmp
```

```
# mkdir -p /home/hadoop_files/hadoop_logs/hive/querylog
```

// 改一下权限

```
# chown -R hadoop:hadoop /home/hadoop_files/
```

```
# chown -R hadoop:hadoop /usr/local/apache-hive-1.1.0-bin
```

3.9.2 测试

// 打开 hive 客户端

```
$ hive
```

```
hive> create table test_table(id int, name string);
```

```
hive> insert into test_table values(1,"test");
```

// 换台服务器

```
hive> show tables;
```

应该可以看到刚才创建的 test_table

```
hive> select * from test_table;
```

```
hive> drop table test_table;
```

```
hive> show tables;
```

```
hive> exit;
```

3.10 安装 Scala

在 cluster1 上

```
$ su root
```

```
# mv scala-2.10.6.tgz /usr/local
```

```
# tar -zxvf scala-2.10.6.tgz
```

```
# vi /etc/profile
```

最下面加两行:

```
export SCALA_HOME=/usr/local/scala-2.10.6
```

```
export PATH=$SCALA_HOME/bin:$PATH
```

```
// 刷新环境变量
```

```
# source /etc/profile
```

```
// 查看版本, 验证安装是否成功
```

```
# scala -version
```

```
// 复制到所有的服务器上
```

```
# scp -r /usr/local/scala-2.10.6 cluster2:/usr/local/
```

```
# scp -r /usr/local/scala-2.10.6 cluster3:/usr/local/
```

```
//在每一个节点上修改环境变量
```

```
#vi /etc/profile
```

```
export SCALA_HOME=/usr/local/scala-2.10.6
```

```
export PATH=$SCALA_HOME/bin:$PATH
```

```
// 刷新环境变量
```

```
# source /etc/profile
```

```
// 修改文件夹权限(每一个节点都要操作)
```

```
# chown -R hadoop:hadoop /usr/local/scala-2.10.6
```

3.11 安装 Spark

3.11.1 安装

(所有节点都要操作)

下载 spark-1.6.3-bin-hadoop2.6.tgz

```
// 解压
```

```
# cp spark-1.6.3-bin-hadoop2.6.tgz /usr/local
```

```
# cd /usr/local/
```

```
# tar -zxvf spark-1.6.3-bin-hadoop2.6.tgz
```

```
# cd spark-1.6.3-bin-hadoop2.6
```

```
// 环境变量
```



```
# vi /etc/profile
```

添加以下内容：

```
export SPARK_HOME=/usr/local/spark-1.6.3-bin-hadoop2.6
```

```
export PATH=$SPARK_HOME/bin:$PATH
```

主节点要再加一行（cluster1）：

```
export PATH=$SPARK_HOME/sbin:$PATH
```

```
// 复制 conf 文件夹里面 template 一份，改名为 spark-env.sh
```

```
# cp conf/spark-env.sh.template conf/spark-env.sh
```

```
// 在 conf/spark-env.sh 最下面加入以下 7 行：
```

```
#vi conf/spark-env.sh
```

```
export JAVA_HOME=/usr/local/jdk1.7.0_80
```

```
export SCALA_HOME=/usr/local/scala-2.10.6
```

```
export SPARK_MASTER_IP=cluster1
```

```
export HADOOP_CONF_DIR=/usr/local/hadoop-2.6.5/etc/hadoop
```

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop-2.6.5/bin/hadoop classpath)
```

```
export SPARK_CLASSPATH=$HIVE_HOME/lib/mysql-connector-java-5.1.43-bin.jar
```

```
export SPARK_PID_DIR=/home/hadoop_files
```

```
// 在 conf 下面新建一个叫 slaves 的文件
```

```
# vi conf/slaves
```

添加以下几行

```
cluster1
```

```
cluster2
```

```
cluster3
```

```
// 将 hive 目录下 conf 文件夹中的 hive-site.xml 复制到 spark 的 conf 目录下
```

```
# cd /usr/local/
```

```
# cp apache-hive-1.1.0-bin/conf/hive-site.xml spark-1.6.3-bin-hadoop2.6/conf/
```

```
// 将 hadoop/etc/hadoop 文件中的 hdfs-site.xml 和 core-site.xml 文件复制到 spark 的 conf 目录下
```

```
# cp hadoop-2.6.5/etc/hadoop/hdfs-site.xml hadoop-2.6.5/etc/hadoop/core-site.xml spark-1.6.3-bin-hadoop2.6/conf/
```

```
// 将 conf 目录下的 spark-defaults.conf.template 复制一份，改名为 spark-default.conf
```

```
# cd spark-1.6.3-bin-hadoop2.6/conf/
```

```
# cp spark-defaults.conf.template spark-defaults.conf
```

```
# vi spark-defaults.conf
```

```
// 在最下面加上下面这一行
```

```
spark.files file:///usr/local/spark-1.6.3-bin-hadoop2.6/conf/hdfs-site.xml,file:///usr/local/spark-1.6.3-bin-hadoop2.6/conf/core-site.xml
```

保存后退出即可。

```
// 复制到所有的服务器上
# scp -r /usr/local/spark-1.6.3-bin-hadoop2.6 cluster2:/usr/local/
# scp -r /usr/local/spark-1.6.3-bin-hadoop2.6 cluster3:/usr/local/

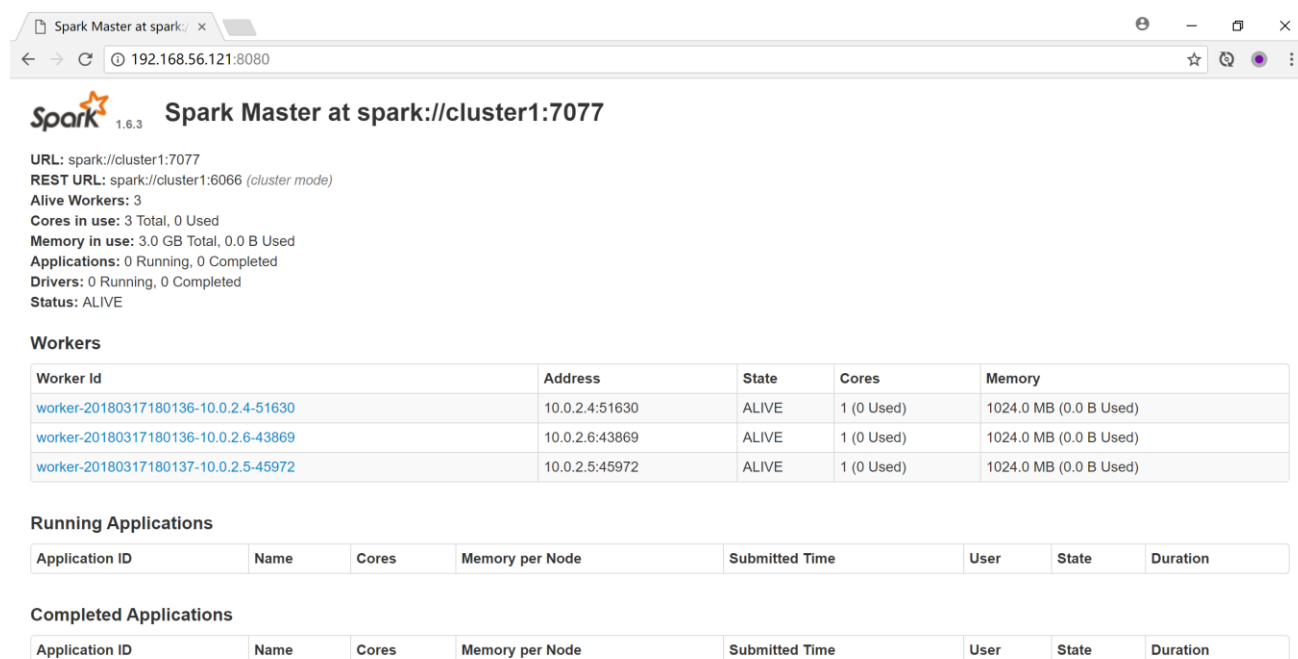
// 修改 spark 文件夹的权限（每个 spark 结点）
# chown -R hadoop:hadoop /usr/local/spark-1.6.3-bin-hadoop2.6

// 运行 Spark（cluster1 上）
运行 spark 前需启动 hadoop 的 HDFS 和 YARN
$ start-master.sh
$ start-slaves.sh

// 关闭 Spark 的命令（cluster1 上）
$ stop-slaves.sh
$ stop-master.sh
```

3.11.2 测试

在 cluster1 上使用 jps 命令可以看到 Master 和 Worker，cluster2 和 3 上可以看到 Worder
用浏览器访问 192.168.56.121:8080 可以看到 Spark 的 web 界面，可以看到 3 个 worker



Spark Master at spark://cluster1:7077

URL: spark://cluster1:7077
 REST URL: spark://cluster1:6066 (cluster mode)
 Alive Workers: 3
 Cores in use: 3 Total, 0 Used
 Memory in use: 3.0 GB Total, 0.0 B Used
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20180317180136-10.0.2.4-51630	10.0.2.4:51630	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20180317180136-10.0.2.6-43869	10.0.2.6:43869	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20180317180137-10.0.2.5-45972	10.0.2.5:45972	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
----------------	------	-------	-----------------	----------------	------	-------	----------

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
----------------	------	-------	-----------------	----------------	------	-------	----------

3.12 安装 Storm

storm 需要 python2.6 以上的版本

// 查看 python 版本

python

可以在最上面一行看到 python 的版本

// 退出 python 交互式界面

>>> exit()

如果版本低于 2.6, 使用 yum install python, 安装 Python2.7

(以下都要在所有节点上操作)

将下载的 storm 解压到/usr/local 下

// 添加环境变量

vi /etc/profile

export STORM_HOME=/usr/local/apache-storm-1.1.1

export PATH=\$STORM_HOME/bin:\$PATH

// 改一下权限

chown -R hadoop:hadoop apache-storm-1.1.1

// 更改配置文件

vi apache-storm-1.1.1/conf/storm.yaml

里面有两个要改的地方

第一个是

storm.zookeeper.servers :

- "cluster1"
- "cluster2"
- "cluster3"

第二个是加入一行

storm.local.dir : "/home/hadoop_files/hadoop_tmp/storm/tmp"

切记: 冒号左右要有空格, 否则会报错 could not found expected ' : '

storm.local.dir 的最左边也要有一个空格

// 然后新建 tmp 文件夹, 改权限

mkdir -p /home/hadoop_files/hadoop_tmp/storm/tmp

chown -R hadoop:hadoop /home/hadoop_files

chown -R hadoop:hadoop /usr/local/apache-storm-1.1.1

// 新建 storm-master 的虚拟窗口(cluster1)

\$ screen -S storm-master

\$ storm nimbus

\$ Ctrl+A+D

// 新建 storm-supervisor 的虚拟窗口 (cluster2, cluster3)

\$ screen -S storm-supervisor

\$ storm supervisor

\$ Ctrl+A+D

// 新建 storm-ui 的虚拟窗口(cluster1)

\$ screen -S storm-ui

\$ storm ui

\$ Ctrl+A+D

// 新建 storm-logviewer 的虚拟窗口 (cluster1, cluster2, cluster3)

\$ screen -S storm-logviewer

\$ storm logviewer

\$ Ctrl+A+D

使用 jps 可以看到以下进程

cluster1: nimbus, core, logviewer

cluster2: Supervisor, logviewer

cluster3: Supervisor, logviewer

关闭的过程就是按逆向的顺序进入虚拟窗口后, 使用 Ctrl+C 退出即可。