华东师范大学软件工程学院实验报告

实验课程:	Cloud Computing	年　级:	2022 级	实验成绩:	
实验名称:	Headoop 搭建	姓　名:	李鹏达	实验日期:	2024 年 12 月 18 日
实验编号:	No. 3	学　号:	10225101460	实验时间:	第 15 - 16 周

1　实验内容

- Linux 系统安装及配置
- Hadoop 单例模式搭建
- Hadoop 伪分布式模式搭建
- 虚拟机克隆及相关网络配置
- 集群时间同步
- Hadoop 集群模式部署
- MapReduce 案例应用

2　实验关键步骤

2.1　Linux 系统的安装及配置

2.1.1　安装 Linux 系统

2.1.1.1　创建虚拟机

NOTE

由于在实验一中已经安装了 Oracle VirtualBox 虚拟机平台，因此，我选择使用 VirtualBox 创建虚拟机，而不是实验手册中的 VMware Workstation。

(1) 新建一个虚拟机，名称为 `lipengda001`
(2) 将虚拟光盘选择为预先下载好的 `CentOS7` 映像，勾选“跳过自动安装”。
(3) 为虚拟机分配 2GB 内存，20GB 硬盘空间。

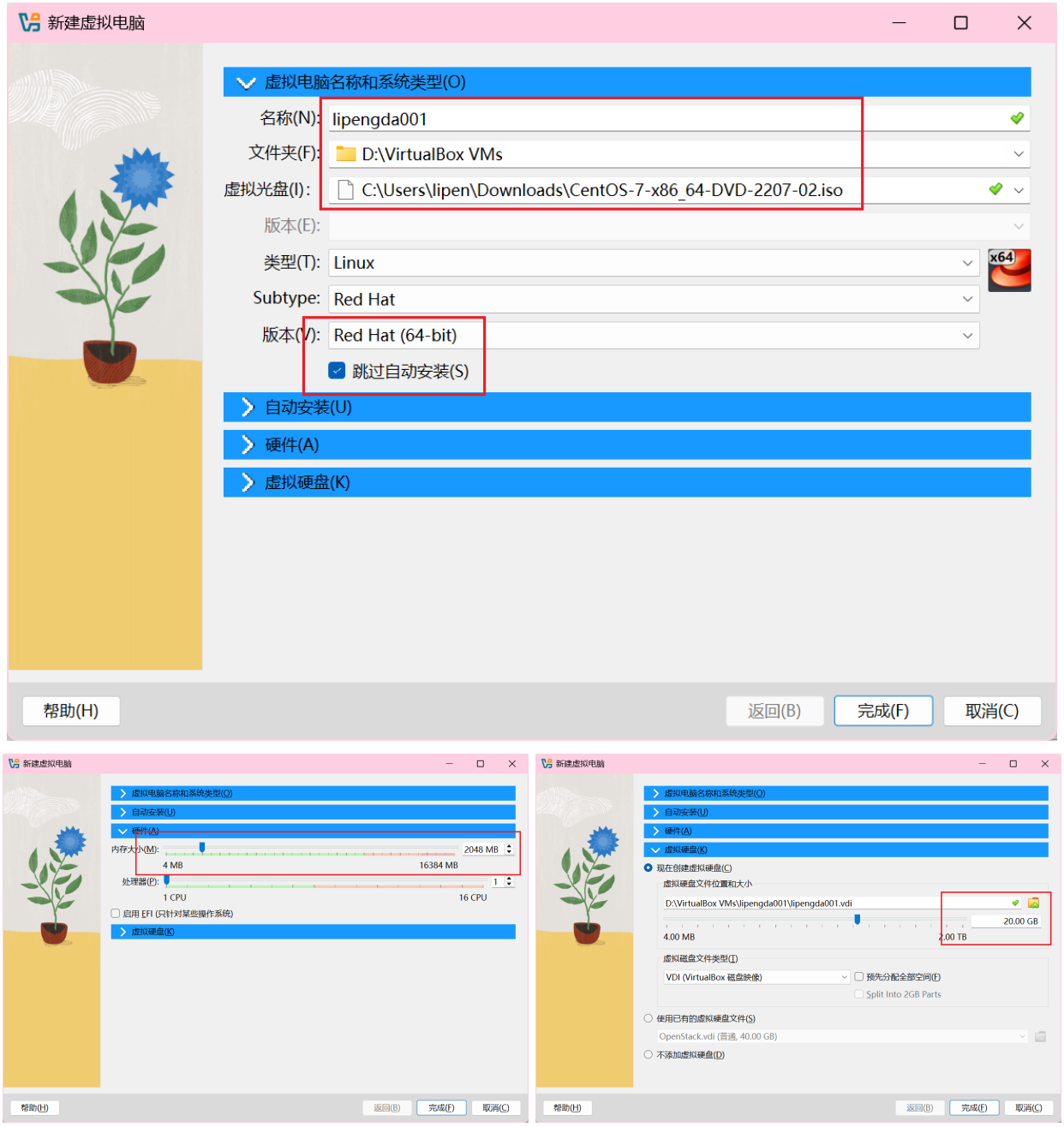

图 1: 创建虚拟机

2.1.1.2 启动虚拟机

启动虚拟机，选择 `Install CentOS Linux 7` 安装系统。

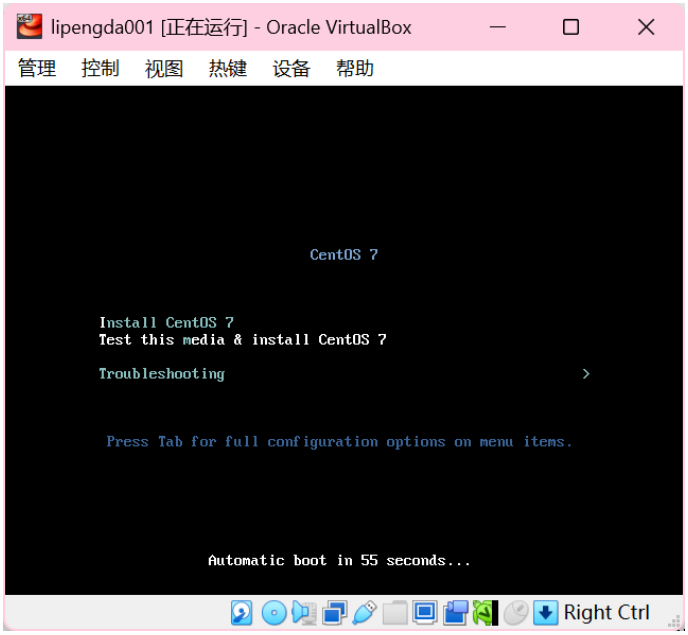

图 2: 安装系统 (1)

在软件选择中，选择“最小安装”。

NOTE

为减少资源占用并加快安装速度，我在此处选择了“最小安装”，而不是“带 GUI 的服务器”。

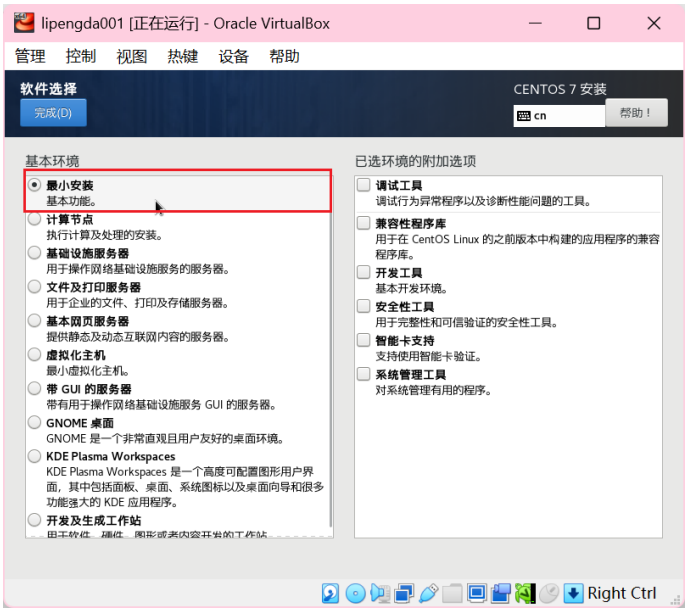

图 3: 安装系统 (2)

设置 root 账户密码。

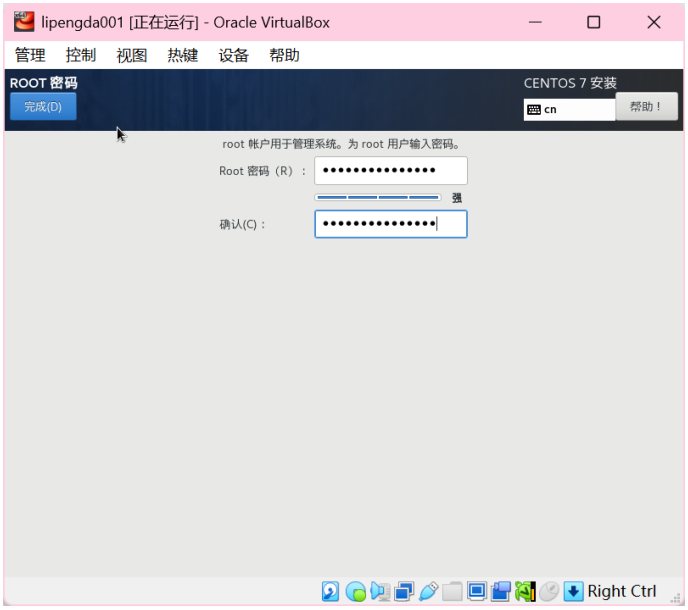

图 4: 安装系统 (3)

等待系统安装完成。

2.1.2 Linux 系统相关配置

2.1.2.1 网络配置

(1) 输入命令 `ip addr` 查看网络配置。

```
1 ip addr
```

```
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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 group default qlen 1
000
    link/ether 08:00:27:d1:39:c4 brd ff:ff:ff:ff:ff:ff
```

图 5: 查看网络配置

可以看到，我的网卡名称叫 `enp0s3`。

(2) 查看网卡 IP 信息的配置文件。

```
ls /etc/sysconfig/network-scripts/
```

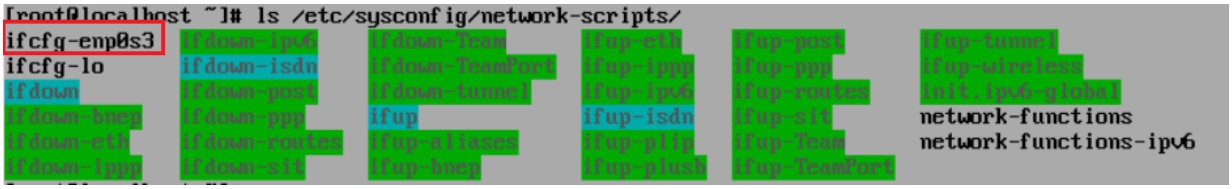

图 6: 查看配置文件

可以看到，网卡配置文件为 `ifcfg-enp0s3`。

(3) 查看网卡配置文件。

```
cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

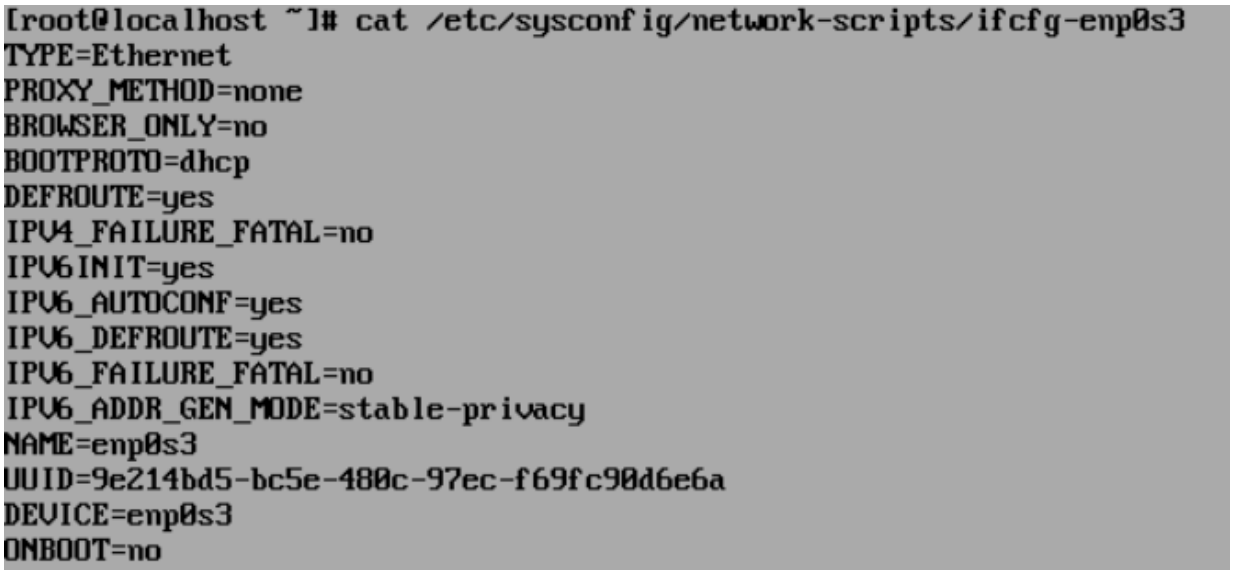

图 7: 查看网卡配置文件

(4) 修改网卡配置文件，将 `ONBOOT` 的值改为 `yes`。

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

```
"/etc/sysconfig/network-scripts/ifcfg-enp0s3" 15L, 282C written
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=9e214bd5-bc5e-480c-97ec-f69fc90d6e6a
DEVICE=enp0s3
ONBOOT=yes
```

图 8: 修改网卡配置文件

(5) 重启网络服务。

```
1 service network restart
```

(6) 再次查看网络配置。

```
1 ip addr
```

```
[root@localhost ~]# service network restart
Restarting network (via systemctl):                        [  OK  ]
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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 group default qlen 1
000
    link/ether 08:00:27:d1:39:c4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.111/24 brd 192.168.1.255 scope global noprefixroute dynamic enp0s3
       valid_lft 7198sec preferred_lft 7198sec
    inet6 fe80::6d2:58f3:39b6:9be0/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

图 9: 查看网络配置

可以看到，分配的 IP 地址为 192.168.1.111。

(7) 再次修改网卡配置文件，将 BOOTPROTO 的值改为 static，并添加 IPADDR、NETMASK、GATEWAY 等字段，将 IP 地址设置为 192.168.1.111。

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

```
"/etc/sysconfig/network-scripts/ifcfg-enp0s3" 18L, 347C written
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp0s3
UUID=9e214bd5-bc5e-480c-97ec-f69fc90d6e6a
DEVICE=enp0s3
ONBOOT=yes
IPADDR=192.168.1.111
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
```

图 10: 修改网卡配置文件

(8) 重启网络服务。

```
service network restart
```

(9) 查看网络配置，确认配置成功。

```
ip addr
```

```
[root@localhost ~]# service network restart
Restarting network (via systemctl):                        [  OK  ]
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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 group default qlen 1
000
    link/ether 08:00:27:d1:39:c4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.111/24 brd 192.168.1.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::6d2:58f3:39b6:9be0/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

图 11: 查看网络配置

(10) 测试网络连通性。

```
1 ping 192.168.1.111
```

```
[root@localhost ~]# ping 192.168.1.111
PING 192.168.1.111 (192.168.1.111) 56(84) bytes of data.
64 bytes from 192.168.1.111: icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from 192.168.1.111: icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from 192.168.1.111: icmp_seq=3 ttl=64 time=0.042 ms
64 bytes from 192.168.1.111: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from 192.168.1.111: icmp_seq=5 ttl=64 time=0.041 ms
64 bytes from 192.168.1.111: icmp_seq=6 ttl=64 time=0.061 ms
64 bytes from 192.168.1.111: icmp_seq=7 ttl=64 time=0.055 ms
^C
--- 192.168.1.111 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 5999ms
rtt min/avg/max/mdev = 0.037/0.046/0.061/0.009 ms
```

图 12: 测试网络连通性

2.1.2.2 修改主机名

(1) 查看主机名。

```
1 hostname
```

```
[root@localhost ~]# hostname
localhost.localdomain
```

图 13: 查看主机名

(2) 修改主机名。

```
1 vi /etc/sysconfig/network
```

设置 NETWORKING 为 yes，HOSTNAME 为 lipengda001。

```
"/etc/sysconfig/network" 3L, 58C written
[root@localhost ~]# cat /etc/sysconfig/network
# Created by anaconda
NETWORKING=YES
HOSTNAME=lipengda001
```

图 14: 修改主机名

```
1 hostnamectl set-hostname lipengda001
```

NOTE

此处实验手册有误，在 CentOS7 中，/etc/sysconfig/network 文件中的 HOSTNAME 字段已经被废弃。该项配置已经被移动到 /etc/hostname 文件中。使用 hostnamectl 命令修改主机名是一个更好的选择。

```
      IPV6_RADVD_PIDFILE="/some/other/location/radvd.pid"
  IPV6TO4_RADVD_PIDFILE=<pid-file> (obsolete)
    As above, still supported for a while for backward compatibility.
  IPV6_RADVD_TRIGGER_ACTION=startstop|reload|restart|SIGHUP (optional)
    How to trigger radvd in case of 6to4 or PPP action
     startstop: radvd starts if interface goes up and stops
       if interface goes down using initscript call of radvd with related parameter
     reload|restart: initscript of radvd is called with this parameter
     SIGHUP: signal HUP is sent to radvd, pidfile must be specified, if not the default
    Default: SIGHUP

  IPv6 options above can be overridden in interface-specific configuration.

  obsoleted values from earlier releases:

    FORWARD_IPV4=yes|no
      This setting has been moved into net.ipv4.ip_forward setting
      in /etc/sysctl.conf. Setting it to 1 there enables IP forwarding,
      setting it to 0 disables it (which is the default for RFC compliance).

    NETWORKWAIT=yes|no
      This is not used with the move to systemd.

    HOSTNAME=<fqdn by default, but whatever hostname you want>
      This is now configured in /etc/hostname.

/etc/sysconfig/static-routes-ipv6:
  Contains lines of the form:

    <device> IPv6-network IPv6-gateway
    <tunneldevice> IPv6-network

  <device> must be a device name to have the route brought up and
  down with the device

  For example:
```

图 15: /usr/share/doc/initscripts-9.49.53/sysconfig.txt 中的说明

(3) 重启虚拟机。

```
1 reboot
```

(4) 查看主机名是否修改成功。

```
1 hostname
```

使用 ping 命令进行验证。

```
1 ping lipengda001
```

```
[root@lipengda001 ~]# hostname
lipengda001
[root@lipengda001 ~]# ping lipengda001
PING lipengda001 (192.168.1.111) 56(84) bytes of data.
64 bytes from lipengda001 (192.168.1.111): icmp_seq=1 ttl=64 time=0.021 ms
64 bytes from lipengda001 (192.168.1.111): icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from lipengda001 (192.168.1.111): icmp_seq=3 ttl=64 time=0.041 ms
^C
--- lipengda001 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.021/0.034/0.042/0.011 ms
```

图 16: 验证主机名修改成功

2.1.2.3 建立 IP 地址与虚拟机名称的对应关系

修改域名解析映射文件，使得后续可以直接通过主机名访问。

```
1 vi /etc/hosts
```

在文件中添加 `192.168.1.111 lipengda001`。

```
"/etc/hosts" 3L, 184C written
[root@lipengda001 ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.111 lipengda001
```

图 17: 修改域名解析映射文件

2.1.2.4 Linux 系统与 Windows 系统进行网络通讯

(1) 在 Windos 系统中，使用 `ping` 命令测试与 Linux 系统的网络连通性。

```
1 ping 192.168.1.111
```

```
pwsh ~                                                    21:07:33
❯ ping 192.168.1.111

Pinging 192.168.1.111 with 32 bytes of data:
Reply from 192.168.1.111: bytes=32 time<1ms TTL=64
Reply from 192.168.1.111: bytes=32 time=1ms TTL=64
Reply from 192.168.1.111: bytes=32 time<1ms TTL=64
Reply from 192.168.1.111: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.111:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 18: Windows 系统与 Linux 系统网络连通性测试

(2) 在 Windos 端，尝试 `ping` 主机名 `lipengda001`。

```
1 ping lipengda001
```

显示无法解析主机名。

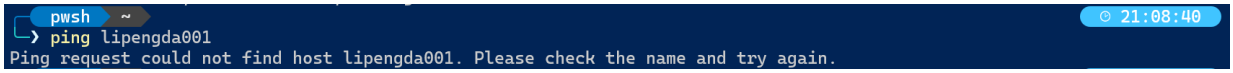

图 19: Windows 系统无法解析主机名

因此，需要在 Windows 系统中修改 `hosts` 文件 (位于 `C:\Windows\System32\drivers\etc`)，添加 `192.168.1.111 lipengda001`。

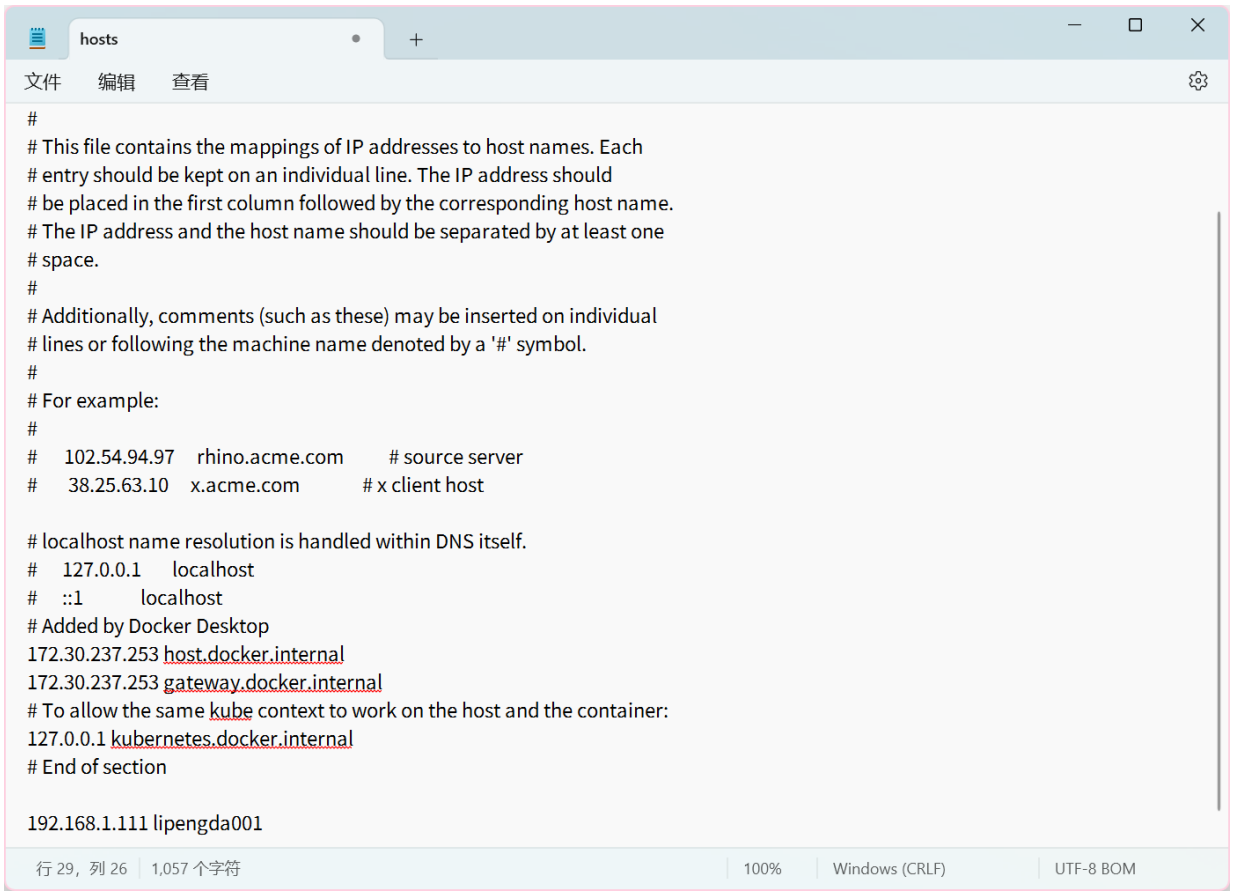

图 20: 修改 Windows 系统 hosts 文件

再次尝试 `ping` 主机名 `lipengda001`。

```
1 ping lipengda001
```

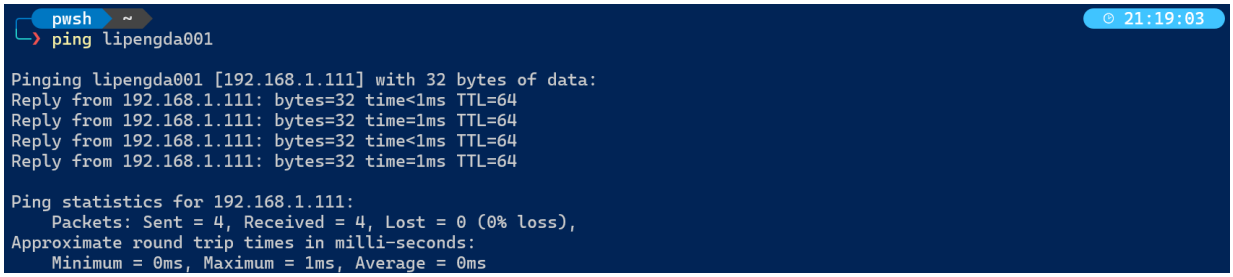

图 21: Windows 系统解析主机名成功

2.2 Hadoop 单例模式搭建

2.2.1 安装 JDK

2.2.1.1 下载解压

(1) 创建一个文件夹 app

```
1 mkdir app
```

(2) 检查系统是否存在 openjdk

```
1 rpm -qa | grep java
```

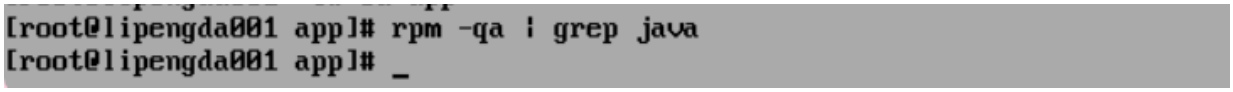

图 22: 检查系统是否存在 openjdk

此处没有输出，说明系统中没有安装 openjdk。

(3) 下载 JDK

```
1 curl -L -C - -b "oraclelicense=accept-securebackup-cookie" -O http://
      download.oracle.com/otn-pub/java/jdk/8u131-b11/
      d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz
```

```
[root@lipengda001 app]# curl -L -C - -b "oraclelicense=accept-securebackup-cookie" -O http://downloa
d.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100   533  100   533    0     0    325      0  0:00:01  0:00:01 --:--:--     0
100  176M  100  176M    0     0  3494k      0  0:00:51  0:00:51 --:--:-- 4994k
[root@lipengda001 app]# ls
jdk-8u131-linux-x64.tar.gz
[root@lipengda001 app]#
```

图 23: 下载 JDK

(4) 解压 JDK

```
1 tar -zxvf jdk-8u131-linux-x64.tar.gz -C .
```

```
jdk1.8.0_131/jre/lib/amd64/libavplugin-55.so
jdk1.8.0_131/jre/lib/amd64/libawt.so
jdk1.8.0_131/jre/lib/amd64/libjawt.so
jdk1.8.0_131/jre/lib/amd64/libverify.so
jdk1.8.0_131/jre/lib/amd64/libzip.so
jdk1.8.0_131/jre/lib/amd64/libjavafx_iio.so
jdk1.8.0_131/jre/lib/amd64/libjava_crw_demo.so
jdk1.8.0_131/jre/lib/amd64/libjfxmedia.so
jdk1.8.0_131/jre/lib/amd64/libnet.so
jdk1.8.0_131/jre/lib/amd64/libjavafx_font.so
jdk1.8.0_131/jre/lib/amd64/libprism_common.so
jdk1.8.0_131/jre/lib/amd64/libnio.so
jdk1.8.0_131/jre/lib/amd64/libprism_es2.so
jdk1.8.0_131/jre/lib/amd64/libinstrument.so
jdk1.8.0_131/jre/lib/amd64/libkcms.so
jdk1.8.0_131/jre/lib/amd64/libawt_xawt.so
jdk1.8.0_131/jre/lib/amd64/libmanagement.so
jdk1.8.0_131/jre/lib/amd64/libunpack.so
jdk1.8.0_131/jre/lib/amd64/libgstreamer-lite.so
jdk1.8.0_131/jre/lib/amd64/libawt_headless.so
jdk1.8.0_131/jre/lib/amd64/libsplashscreen.so
jdk1.8.0_131/jre/lib/fontconfig.properties.src
jdk1.8.0_131/jre/lib/psfont.properties.ja
jdk1.8.0_131/jre/lib/fontconfig.Turbo.properties.src
jdk1.8.0_131/jre/lib/jce.jar
jdk1.8.0_131/jre/lib/flavormap.properties
jdk1.8.0_131/jre/lib/jfxswt.jar
jdk1.8.0_131/jre/lib/fontconfig.SuSE.10.properties.src
jdk1.8.0_131/jre/lib/fontconfig.SuSE.11.bfc
jdk1.8.0_131/jre/COPYRIGHT
jdk1.8.0_131/jre/THIRDPARTYLICENSEREADME-JAVAFX.txt
jdk1.8.0_131/jre/Welcome.html
jdk1.8.0_131/jre/README
jdk1.8.0_131/README.html
[root@lipengda001 app]# ls
jdk1.8.0_131  jdk-8u131-linux-x64.tar.gz
[root@lipengda001 app]#
```

图 24: 解压 JDK

(5) 检验 JDK 是否安装成功

```
cd jdk1.8.0_131
./bin/java -version
```

```
[root@lipengda001 app]# cd jdk1.8.0_131/
[root@lipengda001 jdk1.8.0_131]# ./bin/java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
[root@lipengda001 jdk1.8.0_131]#
```

图 25: 检验 JDK 是否安装成功

2.2.2 配置环境变量

(1) 编辑 `/etc/profile` 文件

```
vi /etc/profile
```

在文件末尾添加以下内容

```
export JAVA_HOME=/root/app/jdk1.8.0_131
export PATH=$JAVA_HOME/bin:$PATH
```

```
HOSTNAME=`/usr/bin/hostname 2>/dev/null`
HISTSIZE=1000
if [ "$HISTCONTROL" = "ignorespace" ] ; then
    export HISTCONTROL=ignoreboth
else
    export HISTCONTROL=ignoredups
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

# By default, we want umask to get set. This sets it for login shell
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge

export JAVA_HOME=/root/app/jdk1.8.0_131
export PATH=$JAVA_HOME/bin:$PATH
-- INSERT --
```

图 26: 编辑/etc/profile 文件

(2) 使配置生效

```
source /etc/profile
```

(3) 检验环境变量是否配置成功

```
java -version
```

```
[root@lipengda001 jdk1.8.0_131]# source /etc/profile
[root@lipengda001 jdk1.8.0_131]# java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
[root@lipengda001 jdk1.8.0_131]#
```

图 27: 检验环境变量是否配置成功

编写一个简单的 Java 程序，检验 JDK 是否配置成功。

```
mkdir workspace
cd workspace
vi HelloWorld.java
```

输入以下内容

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

编译运行

```
javac HelloWorld.java
java HelloWorld
```

```
[root@lipengda001 workspace]# javac HelloWorld.java
[root@lipengda001 workspace]# java HelloWorld
HelloWorld!
[root@lipengda001 workspace]#
```

图 28: Java 程序运行结果

2.2.3 安装 Hadoop

2.2.3.1 下载 Hadoop

下载 Hadoop。

```
curl -O https://archive.apache.org/dist/hadoop/common/hadoop-2.5.0/hadoop-2.5.0.tar.gz
```

解压 Hadoop。

```
tar -zxvf hadoop-2.5.0.tar.gz -C .
```

```
hadoop-2.5.0/etc/hadoop/container-executor.cfg
hadoop-2.5.0/etc/hadoop/core-site.xml
hadoop-2.5.0/etc/hadoop/hadoop-env.cmd
hadoop-2.5.0/etc/hadoop/hadoop-env.sh
hadoop-2.5.0/etc/hadoop/hadoop-metrics.properties
hadoop-2.5.0/etc/hadoop/hadoop-metrics2.properties
hadoop-2.5.0/etc/hadoop/hadoop-policy.xml
hadoop-2.5.0/etc/hadoop/hdfs-site.xml
hadoop-2.5.0/etc/hadoop/httpfs-env.sh
hadoop-2.5.0/etc/hadoop/httpfs-log4j.properties
hadoop-2.5.0/etc/hadoop/httpfs-signature.secret
hadoop-2.5.0/etc/hadoop/httpfs-site.xml
hadoop-2.5.0/etc/hadoop/log4j.properties
hadoop-2.5.0/etc/hadoop/mapred-env.cmd
hadoop-2.5.0/etc/hadoop/mapred-env.sh
hadoop-2.5.0/etc/hadoop/mapred-queues.xml.template
hadoop-2.5.0/etc/hadoop/mapred-site.xml.template
hadoop-2.5.0/etc/hadoop/slaves
hadoop-2.5.0/etc/hadoop/ssl-client.xml.example
hadoop-2.5.0/etc/hadoop/ssl-server.xml.example
hadoop-2.5.0/etc/hadoop/yarn-env.cmd
hadoop-2.5.0/etc/hadoop/yarn-env.sh
hadoop-2.5.0/etc/hadoop/yarn-site.xml
hadoop-2.5.0/bin/container-executor
hadoop-2.5.0/bin/hadoop
hadoop-2.5.0/bin/hadoop.cmd
hadoop-2.5.0/bin/hdfs
hadoop-2.5.0/bin/hdfs.cmd
hadoop-2.5.0/bin/mapred
hadoop-2.5.0/bin/mapred.cmd
hadoop-2.5.0/bin/rcc
hadoop-2.5.0/bin/test-container-executor
hadoop-2.5.0/bin/yarn
hadoop-2.5.0/bin/yarn.cmd
[root@lipengda001 app]# ls
hadoop-2.5.0  hadoop-2.5.0.tar.gz  jdk1.8.0_131  jdk-8u131-linux-x64.tar.gz  workspace
[root@lipengda001 app]# _
```

图 29: 下载解压 Hadoop

2.2.3.2 配置环境变量

(1) 编辑 `/etc/profile` 文件

```
vi /etc/profile
```

在文件末尾添加以下内容

```
export HADOOP_HOME=/root/app/hadoop-2.5.0
```

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

```
else
    export HISTCONTROL=ignoredups
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

# By default, we want umask to get set. This sets it for login shell
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge

export JAVA_HOME=/root/app/jdk1.8.0_131
export PATH=$JAVA_HOME/bin:$PATH

export HADOOP_HOME=/root/app/hadoop-2.5.0
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
"/etc/profile" 82L, 1985C written
[root@lipengda001 app]# source /etc/profile
[root@lipengda001 app]# _
```

图 30: 编辑/etc/profile 文件

(2) 使配置生效

```
1 source /etc/profile
```

(3) 检验环境变量是否配置成功

```
1 hadoop
```

```
[root@lipengda001 app]# hadoop
Usage: hadoop [--config confdir] COMMAND
       where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
  checknative [-a|-h]  check native hadoop and compression libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
  classpath            prints the class path needed to get the
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
 or
  CLASSNAME            run the class named CLASSNAME

Most commands print help when invoked w/o parameters.
[root@lipengda001 app]# _
```

图 31: 检验环境变量是否配置成功

2.2.3.3 配置 hadoop-env.sh

修改 hadoop-env.sh 文件。

```
1 vi $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

将 JAVA_HOME 配置为 /root/app/jdk1.8.0_131。

```
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/root/app/jdk1.8.0_131

# The jsvc implementation to use. Jsvc is required to run secure datanodes.
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
"hadoop-env.sh" 78L, 3453C written
[root@lipengda001 hadoop]#
```

图 32: 修改 hadoop-env.sh 文件

2.2.3.4 测试

通过单词统计案例测试 Hadoop 是否配置成功。

进入 workspace 目录。

```
1 cd ~/app/workspace
```

运行 Hadoop。

```
1 hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.5.0.
      jar wordcount ./HelloWorld.java ./out
```

```
                FILE: Number of bytes written=1001700
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=5
                Map output records=13
                Map output bytes=158
                Map output materialized bytes=161
                Input split bytes=105
                Combine input records=13
                Combine output records=10
                Reduce input groups=10
                Reduce shuffle bytes=161
                Reduce input records=10
                Reduce output records=10
                Spilled Records=20
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=15
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=270671872
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=115
        File Output Format Counters
                Bytes Written=127
[root@lipengda001 workspace]#
```

图 33: 运行 Hadoop 测试案例

查看结果。

```
1 cd out
2 ls
3 cat part-r-00000
```

```
[root@lipengda001 workspace]# cd out
[root@lipengda001 out]# ls
part-r-00000  _SUCCESS
[root@lipengda001 out]# cat part-r-00000
HelloWorld      1
System.out.println("HelloWorld!");      1
args[]) 1
class   1
main(String     1
public  2
static  1
void    1
{       2
}       2
[root@lipengda001 out]#
```

图 34: 查看结果

2.3 Hadoop 伪分布式模式搭建

2.3.1 修改配置文件

进入 Hadoop 配置文件目录。

```
1 cd $HADOOP_HOME/etc/hadoop
```

2.3.1.1 配置 hadoop-env.sh

在 2.2.3.3 中已经配置过 hadoop-env.sh 文件。

2.3.1.2 配置 core-site.xml

```
1 vi core-site.xml
```

添加以下内容。

```
1 <configuration>
2     <property>
3         <name>fs.default.name</name>
4         <value>hdfs://lipengda001:9000</value>
5     </property>
6 </configuration>
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://lipengda001:9000</value>
  </property>
</configuration>
```

图 35: 配置 core-site.xml

2.3.1.3 配置 hdfs-site.xml

(1) 配置 hadoop 公共目录。

在 Hadoop 的配置文件目录下创建 data 目录。在 data 目录下创建 namenode、datanode 和 tmp 目录。

```
1 cd $HADOOP_HOME
2 mkdir data
3 mkdir data/namenode
4 mkdir data/datanode
5 mkdir data/tmp
```

```
[root@lipengda001 hadoop-2.5.0]# cd $HADOOP_HOME
[root@lipengda001 hadoop-2.5.0]# mkdir data
[root@lipengda001 hadoop-2.5.0]# mkdir data/namenode
[root@lipengda001 hadoop-2.5.0]# mkdir data/datanode
[root@lipengda001 hadoop-2.5.0]# mkdir data/tmp
[root@lipengda001 hadoop-2.5.0]# ls data
datanode  namenode  tmp
```

图 36: 配置 hadoop 公共目录

(2) 修改配置文件

```
1 cd $HADOOP_HOME/etc/hadoop
2 vi hdfs-site.xml
```

在 hdfs-site.xml 文件中添加以下内容。

```
1 <configuration>
2     <property>
3         <name>dfs.name.dir</name>
4         <value>/root/app/hadoop-2.5.0/data/namenode</value>
5     </property>
6     <property>
7         <name>dfs.data.dir</name>
```

```
        <value>/root/app/hadoop-2.5.0/data/datanode</value>
    </property>
    <property>
        <name>dfs.tmp.dir</name>
        <value>/root/app/hadoop-2.5.0/data/tmp</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
     <name>dfs.name.dir</name>
     <value>/root/app/hadoop-2.5.0/data/namenode</value>
  </property>
  <property>
     <name>dfs.data.dir</name>
     <value>/root/app/hadoop-2.5.0/data/datanode</value>
  </property>
  <property>
     <name>dfs.tmp.dir</name>
     <value>/root/app/hadoop-2.5.0/data/tmp</value>
  </property>
  <property>
     <name>dfs.replication</name>
     <value>1</value>
  </property>_
</configuration>

-- INSERT --
```

图 37: 修改配置文件

2.3.1.4 配置 mapred-site.xml

```
mv mapred-site.xml.template mapred-site.xml
vi mapred-site.xml
```

在 mapred-site.xml 文件中添加以下内容。

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

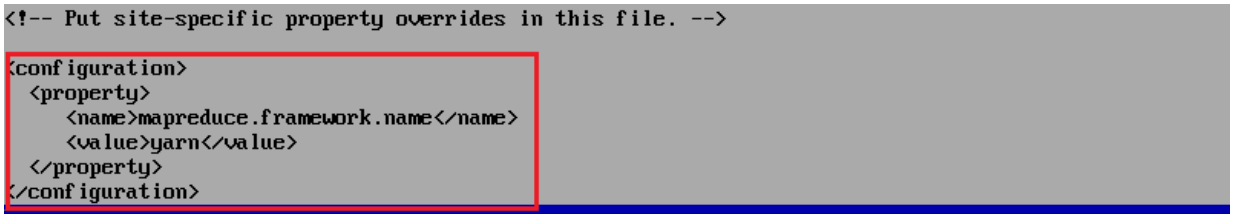

图 38: 配置 mapred-site.xml

2.3.1.5 配置 yarn-site.xml

```
1 vi yarn-site.xml
```

在 yarn-site.xml 文件中添加以下内容。

```
1  <configuration>
2      <property>
3          <name>yarn.resourcemanager.hostname</name>
4          <value>lipengda001</value>
5      </property>
6      <property>
7          <name>yarn.nodemanager.aux-services</name>
8          <value>mapreduce_shuffle</value>
9      </property>
10 </configuration>
```

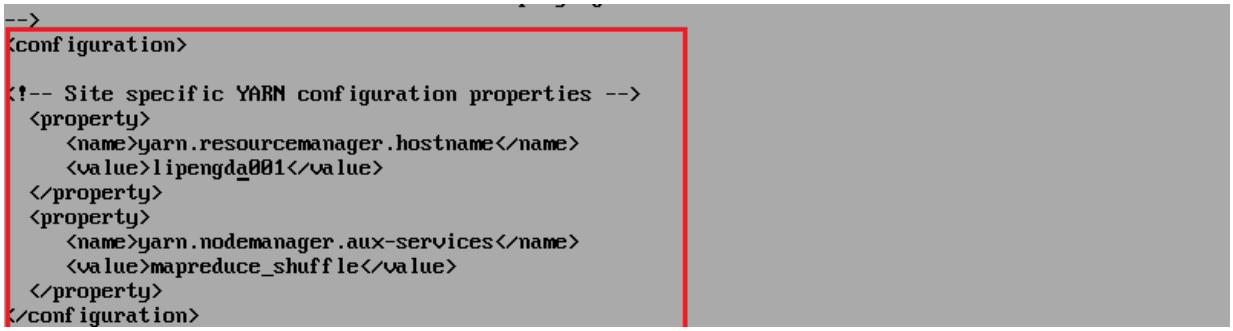

图 39: 配置 yarn-site.xml

2.3.1.6 配置 slaves

```
1 vi slaves
```

将 localhost 改为 lipengda001。

```
"slaves" 1L, 12C written
[root@lipengda001 hadoop]# cat slaves
lipengda001
[root@lipengda001 hadoop]# _
```

图 40: 配置 slaves

2.3.1.7 配置环境变量 PATH

在 2.2.3.2 中已经配置过环境变量。

2.3.2 启动

2.3.2.1 namenode 格式化

```
1 hdfs namenode -format
```

```
24/12/19 00:01:02 INFO namenode.FSNamesystem: isPermissionEnabled = true
24/12/19 00:01:02 INFO namenode.FSNamesystem: HA Enabled: false
24/12/19 00:01:02 INFO namenode.FSNamesystem: Append Enabled: true
24/12/19 00:01:02 INFO util.GSet: Computing capacity for map INodeMap
24/12/19 00:01:02 INFO util.GSet: VM type       = 64-bit
24/12/19 00:01:02 INFO util.GSet: 1.0% max memory 966.7 MB = 9.7 MB
24/12/19 00:01:02 INFO util.GSet: capacity      = 2^20 = 1048576 entries
24/12/19 00:01:02 INFO namenode.NameNode: Caching file names occuring more than 10 times
24/12/19 00:01:02 INFO util.GSet: Computing capacity for map cachedBlocks
24/12/19 00:01:02 INFO util.GSet: VM type       = 64-bit
24/12/19 00:01:02 INFO util.GSet: 0.25% max memory 966.7 MB = 2.4 MB
24/12/19 00:01:02 INFO util.GSet: capacity      = 2^18 = 262144 entries
24/12/19 00:01:02 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.99900001287460
33
24/12/19 00:01:02 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
24/12/19 00:01:02 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
24/12/19 00:01:02 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
24/12/19 00:01:02 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cach
e entry expiry time is 600000 millis
24/12/19 00:01:02 INFO util.GSet: Computing capacity for map NameNodeRetryCache
24/12/19 00:01:02 INFO util.GSet: VM type       = 64-bit
24/12/19 00:01:02 INFO util.GSet: 0.029999999329447746% max memory 966.7 MB = 297.0 KB
24/12/19 00:01:02 INFO util.GSet: capacity      = 2^15 = 32768 entries
24/12/19 00:01:02 INFO namenode.NNConf: ACLs enabled? false
24/12/19 00:01:02 INFO namenode.NNConf: XAttrs enabled? true
24/12/19 00:01:02 INFO namenode.NNConf: Maximum size of an xattr: 16384
24/12/19 00:01:02 INFO namenode.FSImage: Allocated new BlockPoolId: BP-991731584-192.168.1.111-17345
37662763
24/12/19 00:01:02 INFO common.Storage: Storage directory /root/app/hadoop-2.5.0/data/namenode has be
en successfully formatted.
24/12/19 00:01:03 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
24/12/19 00:01:03 INFO util.ExitUtil: Exiting with status 0
24/12/19 00:01:03 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at lipengda001/192.168.1.111
************************************************************/
[root@lipengda001 hadoop]# A_
```

图 41: namenode 格式化

2.3.2.2 启动

```
1 start-dfs.sh
2 start-yarn.sh
```

输入 `jps` 查看进程。

```
1 jps
```

出现以下进程，说明启动成功。

- NameNode
- DataNode
- SecondaryNameNode
- ResourceManager
- NodeManager

```
Are you sure you want to continue connecting (yes/no)? yes
lipengda001: Warning: Permanently added 'lipengda001,192.168.1.111' (ECDSA) to the list of known hos
ts.
root@lipengda001's password:
lipengda001: setterm: $TERM is not defined.
lipengda001: starting namenode, logging to /root/app/hadoop-2.5.0/logs/hadoop-root-namenode-lipengda
001.out
root@lipengda001's password:
lipengda001: setterm: $TERM is not defined.
lipengda001: starting datanode, logging to /root/app/hadoop-2.5.0/logs/hadoop-root-datanode-lipengda
001.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is SHA256:x+4IN5/5swWvMcVYw8trjMCZy/Q/yVg1ewdlBtsnJyk.
ECDSA key fingerprint is MD5:cf:cd:22:b5:0c:d8:a4:99:84:e5:a1:69:90:e8:2a:79.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
root@0.0.0.0's password:
0.0.0.0: setterm: $TERM is not defined.
0.0.0.0: starting secondarynamenode, logging to /root/app/hadoop-2.5.0/logs/hadoop-root-secondarynam
enode-lipengda001.out
[root@lipengda001 hadoop]# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /root/app/hadoop-2.5.0/logs/yarn-root-resourcemanager-lipengda0
01.out
root@lipengda001's password:
lipengda001: setterm: $TERM is not defined.
lipengda001: starting nodemanager, logging to /root/app/hadoop-2.5.0/logs/yarn-root-nodemanager-lipe
ngda001.out
[root@lipengda001 hadoop]# jps
2257 SecondaryNameNode
2118 DataNode
2391 ResourceManager
2760 Jps
2010 NameNode
2670 NodeManager
[root@lipengda001 hadoop]# _
```

图 42: 启动 Hadoop

2.4 虚拟机克隆及相关网络配置

NOTE

由于虚拟机中未安装后续步骤需要的 ntp，为减少重复工作，先在 lipengda001 虚拟机中安装 ntp 后再进行克隆。

NOTE

由于 CentOS7 在 2024 年 6 月 30 日已经停止维护，因此 yum 源已经失效。为了解决这个问题，需要先将 yum 源更换为 CentOS Vault 源。

```
sed -i s/^#.*baseurl=http/baseurl=http/g /etc/yum.repos.d/*.repo
sed -i s/^mirrorlist=http/#mirrorlist=http/g /etc/yum.repos.d/*.repo
sed -i s/mirror.centos.org/vault.centos.org/g /etc/yum.repos.d/*.repo
yum clean all
yum makecache
```

```
yum install -y ntp
```

NOTE

为方便后续主机名和 IP 地址的配置，我选择先创建一个 sh 脚本，将主机名和 IP 地址的配置写入脚本中，将来再在克隆的虚拟机中执行该脚本。

```
vi ~/config.sh
```

内容如下。

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Usage: $0 hostname ip"
    exit 1
fi

hostnamectl set-hostname $1
sed -i "s/HOSTNAME=.*/HOSTNAME=$1/g" /etc/sysconfig/network

echo "$2 $1" >> /etc/hosts

sed -i "s/IPADDR=.*/IPADDR=$2/g" /etc/sysconfig/network-scripts/ifcfg-enp0s3

```

```
service network restart

hostname
ip addr
```

```
chmod +x ~/config.sh
```

2.4.1 虚拟机克隆

在 VirtualBox 中选择 `lipengda001` 虚拟机，右键，然后点击 **复制**。

图 43: 虚拟机克隆 (1)

克隆两台虚拟机，分别命名为 `lipengda002` 和 `lipengda003`。

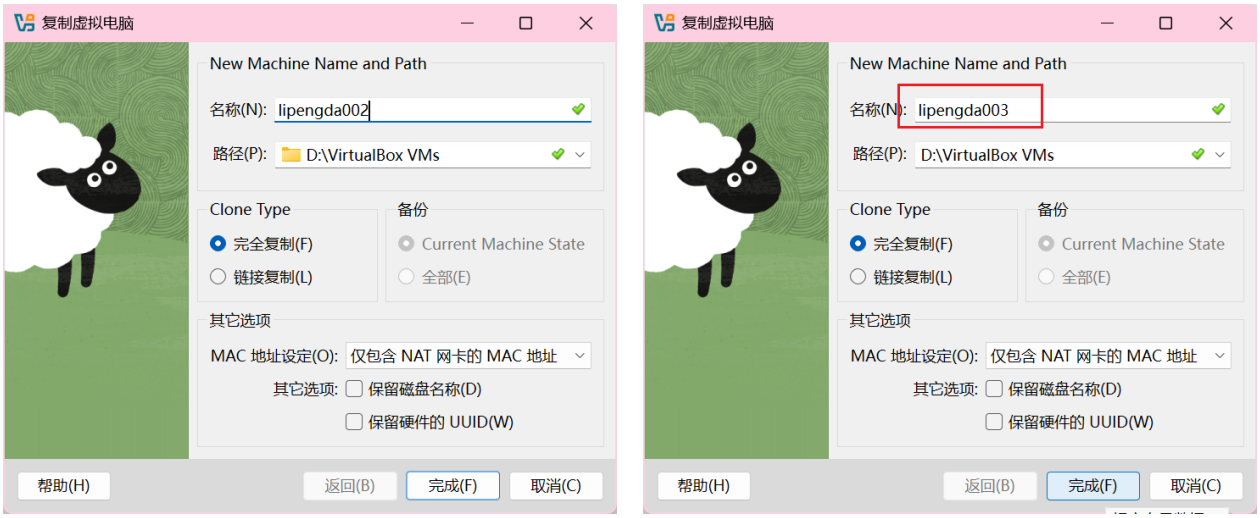

图 44: 虚拟机克隆 (2)

2.4.2 相关网络配置

在 `lipengda002` 和 `lipengda003` 虚拟机中执行 `config.sh` 脚本。

```
./config.sh lipengda002 192.168.1.222
```

```
./config.sh lipengda003 192.168.1.233
```

```
[root@lipengda001 ~]# ./config.sh lipengda002 192.168.1.222
Restarting network (via systemctl):                        [  OK  ]
lipengda002
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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 group default qlen 1
000
    link/ether 08:00:27:cb:27:70 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.222/24 brd 192.168.1.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::6d2:58f3:39b6:9be0/64 scope link tentative noprefixroute
       valid_lft forever preferred_lft forever
```

图 45: 002 相关网络配置

```
[root@lipengda001 ~]# ./config.sh lipengda003 192.168.1.233
Restarting network (via systemctl):                        [  OK  ]
lipengda003
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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 group default qlen 1
000
    link/ether 08:00:27:67:30:64 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.233/24 brd 192.168.1.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::6d2:58f3:39b6:9be0/64 scope link tentative noprefixroute
       valid_lft forever preferred_lft forever
```

图 46: 003 相关网络配置

修改 Windows 系统中的 `hosts` 文件。

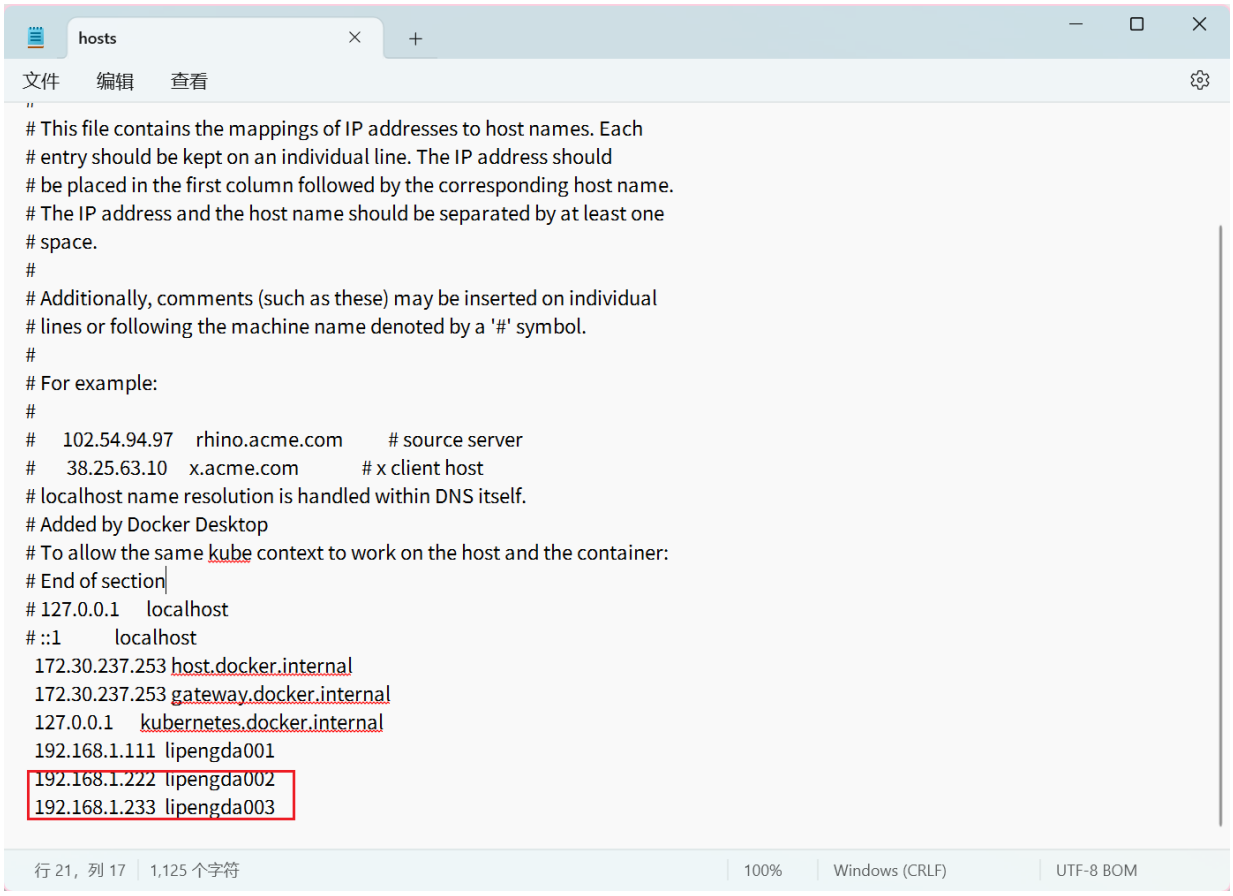

图 47: 修改 Windows 系统 hosts 文件

在 Windows 系统中测试网络连通性。

```
1 ping lipengda001
2 ping lipengda002
3 ping lipengda003
```

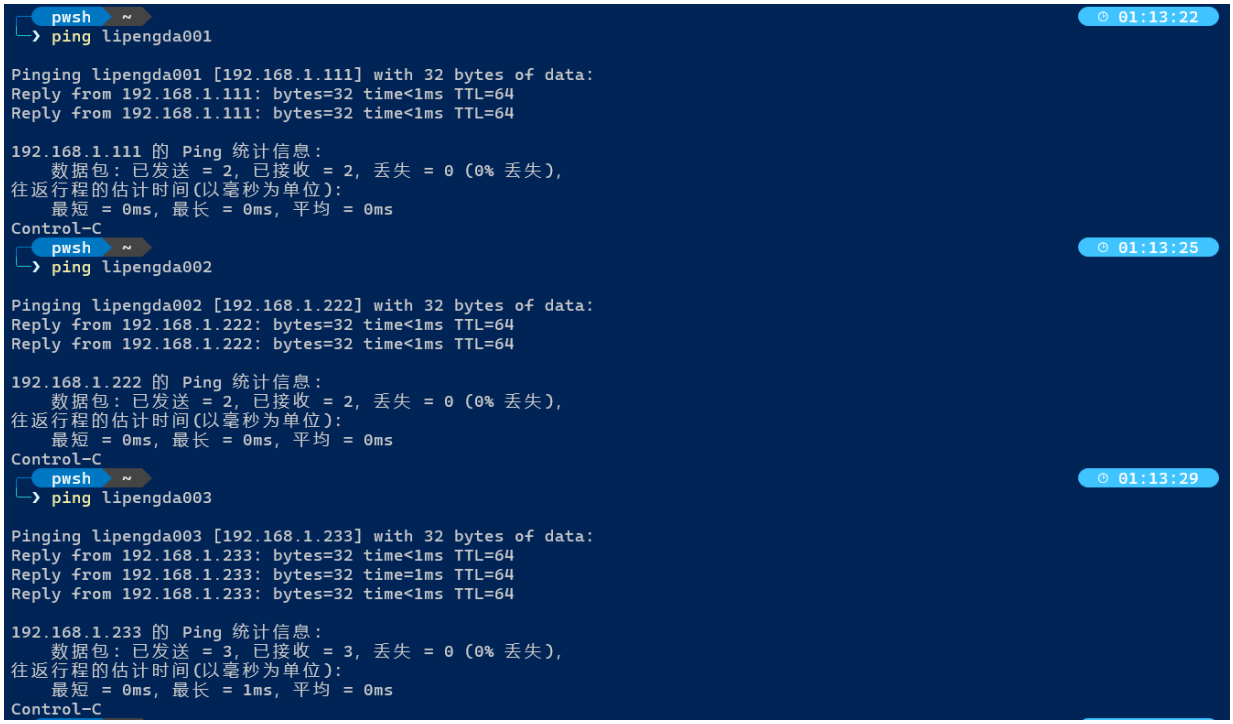

图 48: Windows 系统测试网络连通性

2.5 SSH 免密码登录

2.5.1 创建公钥/私钥

在 `lipengda001` 虚拟机中创建公钥/私钥。

```
1 cd ~/.ssh
2 ssh-keygen -t rsa
```

```
[root@lipengda001 ~]# cd .ssh
[root@lipengda001 .ssh]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:gIl1OuMx8ae6o3mxmEcN0TJYb3ysMleeiRXfUmLU0TM root@lipengda001
The key's randomart image is:
+---[RSA 2048]----+
|   o+.. ..+.oo   |
|  .o+X.. + +. E  |
|  . O+* * o .  o |
|   ..* @ o .     |
|    +o= S        |
|    o=.          |
|   +.o           |
|  o.=.           |
|  o+..           |
+----[SHA256]-----+
[root@lipengda001 .ssh]# ls
id_rsa  id_rsa.pub  known_hosts
```

图 49: 创建公钥/私钥

在 lipengda002 和 lipengda003 虚拟机中进行同样的操作。

2.5.2 对虚拟机自己实行免密码登录

```
1 ssh-copy-id lipengda001
```

```
[root@lipengda001 .ssh]# ssh-copy-id lipengda001
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
 installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install
the new keys
root@lipengda001's password:
setterm: $TERM is not defined.

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'lipengda001'"
and check to make sure that only the key(s) you wanted were added.

[root@lipengda001 .ssh]# ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
[root@lipengda001 .ssh]# _
```

图 50: 对虚拟机自己实行免密码登录

尝试免密码登录。

```
1 ssh lipengda001
```

```
[root@lipengda001 .ssh]# ssh lipengda001
Last login: Thu Dec 19 13:05:40 2024
[root@lipengda001 ~]# exit
logout
Connection to lipengda001 closed.
[root@lipengda001 .ssh]#
```

图 51: 尝试免密码登录

在 lipengda002 和 lipengda003 虚拟机中进行同样的操作。

2.5.3 虚拟机之间相互通讯

2.5.3.1 虚拟机 1 访问虚拟机 2 和 3

(1) 通过 IP 地址

```
1 ssh-copy-id 192.168.1.222
2 ssh 192.168.1.222
```

```
[root@lipengda001 .ssh]# ssh-copy-id 192.168.1.222
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '192.168.1.222 (192.168.1.222)' can't be established.
ECDSA key fingerprint is SHA256:x+4IN5/5swWvMcVYw8trjNCZy/Q/yVg1ewdlBtsnJyk.
ECDSA key fingerprint is MD5:cf:cd:22:b5:0c:d8:a4:99:84:e5:a1:69:90:e8:2a:79.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
 installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install
the new keys
root@192.168.1.222's password:
setterm: $TERM is not defined.

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.1.222'"
and check to make sure that only the key(s) you wanted were added.

[root@lipengda001 .ssh]# ssh 192.168.1.222
Last login: Thu Dec 19 13:11:23 2024 from lipengda002
[root@lipengda002 ~]# logout
Connection to 192.168.1.222 closed.
```

图 52: 通过 IP 地址访问虚拟机 2

```
1 ssh-copy-id 192.168.1.233
2 ssh 192.168.1.233
```

```
[root@lipengda001 .ssh]# ssh-copy-id 192.168.1.233
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '192.168.1.233 (192.168.1.233)' can't be established.
ECDSA key fingerprint is SHA256:x+4IN5/5swWvMcVYw8trjNCZy/Q/yVg1ewdlBtsnJyk.
ECDSA key fingerprint is MD5:cf:cd:22:b5:0c:d8:a4:99:84:e5:a1:69:90:e8:2a:79.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
 installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install
the new keys
root@192.168.1.233's password:
setterm: $TERM is not defined.

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '192.168.1.233'"
and check to make sure that only the key(s) you wanted were added.

[root@lipengda001 .ssh]# ssh 192.168.1.233
Last login: Thu Dec 19 13:12:11 2024 from lipengda003
[root@lipengda003 ~]# logout
Connection to 192.168.1.233 closed.
```

图 53: 通过 IP 地址访问虚拟机 3

对虚拟机 2 和 3 进行同样的操作。

(2) 通过主机名

编辑 `hosts` 文件。

```
1 vi /etc/hosts
```

添加以下内容。

```
1 192.168.1.222 lipengda002
2 192.168.1.233 lipengda003
```

```
"/etc/hosts" 5L, 236C written
[root@lipengda001 .ssh]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.111 lipengda001
192.168.1.222 lipengda002
192.168.1.233 lipengda003
```

图 54: 编辑 `hosts` 文件

尝试使用主机名访问。

```
1 ssh lipengda002
2 ssh lipengda003
```

```
[root@lipengda001 .ssh]# ssh lipengda002
The authenticity of host 'lipengda002 (192.168.1.222)' can't be established.
ECDSA key fingerprint is SHA256:x+4IN5/5swWvMcVYw8trjNCZy/Q/yVg1ewdlBtsnJyk.
ECDSA key fingerprint is MD5:cf:cd:22:b5:0c:d8:a4:99:84:e5:a1:69:90:e8:2a:79.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'lipengda002' (ECDSA) to the list of known hosts.
Last login: Thu Dec 19 13:15:42 2024 from lipengda001
[root@lipengda002 ~]# logout
Connection to lipengda002 closed.
[root@lipengda001 .ssh]# ssh lipengda003
The authenticity of host 'lipengda003 (192.168.1.233)' can't be established.
ECDSA key fingerprint is SHA256:x+4IN5/5swWvMcVYw8trjNCZy/Q/yVg1ewdlBtsnJyk.
ECDSA key fingerprint is MD5:cf:cd:22:b5:0c:d8:a4:99:84:e5:a1:69:90:e8:2a:79.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'lipengda003' (ECDSA) to the list of known hosts.
Last login: Thu Dec 19 13:17:57 2024 from lipengda001
[root@lipengda003 ~]# logout
Connection to lipengda003 closed.
```

图 55: 通过主机名访问虚拟机 2 和 3

在 `lipengda002` 和 `lipengda003` 虚拟机中进行同样的操作。

2.6 集群时间同步

2.6.1 设置时间

2.6.1.1 查看虚拟机中是否安装了 ntp

```
1 rpm -qa | grep ntp
```

```
[root@lipengda001 ~]# rpm -qa | grep ntp
ntpdate-4.2.6p5-29.el7.centos.2.x86_64
ntp-4.2.6p5-29.el7.centos.2.x86_64
```

图 56: 查看是否安装了 ntp

出现内容说明已经安装了 ntp。

2.6.1.2 配置 ntpd

```
1 vi /etc/sysconfig/ntpd
```

添加以下内容

```
1 SYNC_HWCLOCK=yes
```

```
"/etc/sysconfig/ntpd" 3L, 62C written
[root@lipengda001 ~]# cat /etc/sysconfig/ntpd
# Command line options for ntpd
SYNC_HWCLOCK=yes
OPTIONS="-g"
```

图 57: 配置 ntpd

2.6.1.3 开启 ntpd 服务

```
service ntpd status
service ntpd start
chkconfig ntpd on
```

```
[root@lipengda001 ~]# service ntpd status
Redirecting to /bin/systemctl status ntpd.service
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[root@lipengda001 ~]# service ntpd start
Redirecting to /bin/systemctl start ntpd.service
[root@lipengda001 ~]# chkconfig ntpd on
Note: Forwarding request to 'systemctl enable ntpd.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/ntpd.service to /usr/lib/systemd/sy
stem/ntpd.service.
```

图 58: 开启 ntpd 服务

2.6.1.4 修改 ntp 配置

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

取消注释以下内容

```
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

注释以下内容

```
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
```

在 `#crypto` 下添加以下内容

```
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

NOTE

此处实验手册遗漏了重启 ntpd 服务。

```
service ntpd restart
```

```
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default nomodify notrap nopeer noquery

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1
restrict ::1

# Hosts on local network are less restricted.
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#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

#broadcast 192.168.1.255 autokey        # broadcast server
#broadcastclient                        # broadcast client
#broadcast 224.0.1.1 autokey            # multicast server
#multicastclient 224.0.1.1              # multicast client
#manycastserver 239.255.254.254         # manycast server
#manycastclient 239.255.254.254 autokey # manycast client

# Enable public key cryptography.
#crypto
server 127.127.1.0
fudge  127.127.1.0 stratum 10
```

图 59: 修改 ntp 配置

2.6.1.5 设置具体时间

随便设置一个时间。

```
1 date -s 2024-12-01
2 date -s 12:00:00
3 date
```

```
[root@lipengda001 ~]# date -s 2024-12-01
Sun Dec  1 00:00:00 CST 2024
[root@lipengda001 ~]# date -s 12:00:00
Sun Dec  1 12:00:00 CST 2024
[root@lipengda001 ~]# date
Sun Dec  1 12:00:03 CST 2024
```

图 60: 设置具体时间

2.6.2 脚本同步

2.6.2.1 编写脚本

在其他虚拟机上写一个脚本与虚拟机 1 的时间同步，设置为每 10 分钟同步一次时间。

```
1 crontab -e
```

输入以下内容。

```
1 0-59/10 * * * * /usr/sbin/ntpdate lipengda001
```

2.6.2.2 同步

输入命令立刻进行一次时间同步。

```
1 /usr/sbin/ntpdate lipengda001
```

NOTE

如果同步失败，可以尝试关闭 `lipengda001` 虚拟机的防火墙。

`service firewalld stop`

```
[root@lipengda002 ~]# /usr/sbin/ntpdate lipengda001
 1 Dec 12:23:48 ntpdate[1888]: step time server 192.168.1.111 offset -1561936.365390 sec
```

图 61: 时间同步

2.7 Hadoop 集群模式部署

2.7.1 主节点部署

2.7.1.1 重新解压

进入 app 目录下，重新解压一个 hadoop 文件到其他文件夹。

```
1 cd ~/app
2 tar -zxvf hadoop-2.5.0.tar.gz -C /tmp
```

了与之前的 hadoop 伪分布式模式区分, 将其重命名为 `hadoop`。

```
1 mv /tmp/hadoop-2.5.0 ~/app/hadoop
```

```
hadoop-2.5.0/etc/hadoop/httpfs-signature.secret
hadoop-2.5.0/etc/hadoop/httpfs-site.xml
hadoop-2.5.0/etc/hadoop/log4j.properties
hadoop-2.5.0/etc/hadoop/mapred-env.cmd
hadoop-2.5.0/etc/hadoop/mapred-env.sh
hadoop-2.5.0/etc/hadoop/mapred-queues.xml.template
hadoop-2.5.0/etc/hadoop/mapred-site.xml.template
hadoop-2.5.0/etc/hadoop/slaves
hadoop-2.5.0/etc/hadoop/ssl-client.xml.example
hadoop-2.5.0/etc/hadoop/ssl-server.xml.example
hadoop-2.5.0/etc/hadoop/yarn-env.cmd
hadoop-2.5.0/etc/hadoop/yarn-env.sh
hadoop-2.5.0/etc/hadoop/yarn-site.xml
hadoop-2.5.0/bin/container-executor
hadoop-2.5.0/bin/hadoop
hadoop-2.5.0/bin/hadoop.cmd
hadoop-2.5.0/bin/hdfs
hadoop-2.5.0/bin/hdfs.cmd
hadoop-2.5.0/bin/mapred
hadoop-2.5.0/bin/mapred.cmd
hadoop-2.5.0/bin/rcc
hadoop-2.5.0/bin/test-container-executor
hadoop-2.5.0/bin/yarn
hadoop-2.5.0/bin/yarn.cmd
[root@lipengda001 app]# mv /tmp/hadoop-2.5.0 ~/app/hadoop
[root@lipengda001 app]# ls
hadoop  hadoop-2.5.0  hadoop-2.5.0.tar.gz  jdk1.8.0_131  jdk-8u131-linux-x64.tar.gz  workspace
```

图 62: 重新解压

2.7.1.2 修改配置文件

进入 hadoop/etc/hadoop 目录。

```
1 cd ~/app/hadoop/etc/hadoop
```

(1) 配置 hadoop-env.sh

```
1 vi hadoop-env.sh
```

修改 JAVA_HOME 为 /root/app/jdk1.8.0_131。

```
# The java implementation to use.
export JAVA_HOME=/root/app/jdk1.8.0_131
```

图 63: 配置 hadoop-env.sh

(2) 配置 core-site.xml

```
1 vi core-site.xml
```

添加以下内容。

```
1 <configuration>
2     <property>
```

```
3         <name>fs.default.name</name>
4         <value>hdfs://lipengda001:9000</value>
5     </property>
6
7     <property>
8         <name>hadoop.tmp.dir</name>
9         <value>/root/app/hadoop/data/tmp</value>
10    </property>
11
12    <property>
13        <name>fs.trash.interval</name>
14        <value>420</value>
15    </property>
16 </configuration>
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://lipengda001:9000</value>
    </property>

    <property>
        <name>hadoop.tmp.dir</name>
        <value>/root/app/hadoop/data/tmp</value>
    </property>

    <property>
        <name>fs.trash.interval</name>
        <value>420</value>
    </property>
</configuration>
~
```

图 64: 配置 core-site.xml

并在 /root/app/hadoop 目录下创建 data/tmp 目录。

```
1 mkdir -p /root/app/hadoop/data/tmp
```

(3) 配置 hdfs-site.xml

```
1 vi hdfs-site.xml
```

添加以下内容。

```
1 <configuration>
2     <property>
3         <name>dfs.namenode.secondary.http-address</name>
4         <value>lipengda003:50090</value>
5     </property>
```

```
6 </configuration>
```

```
<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>lipengda003:50090</value>
    </property>
</configuration>
```

图 65: 配置 `hdfs-site.xml`

(4) 配置 `mapred-site.xml`

```
1 mv mapred-site.xml.template mapred-site.xml
2 vi mapred-site.xml
```

添加以下内容。

```
1  <configuration>
2      <property>
3          <name>mapreduce.framework.name</name>
4          <value>yarn</value>
5      </property>
6
7      <property>
8          <name>mapreduce.jobhistory.address</name>
9          <value>lipengda001:10020</value>
10     </property>
11
12     <property>
13         <name>mapreduce.jobhistory.webapp.address</name>
14         <value>lipengda001:19888</value>
15     </property>
16 </configuration>
```

```
<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>lipengda001:10020</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>lipengda001:19888</value>
    </property>
</configuration>
```

图 66: 配置 mapred-site.xml

(5) 配置 yarn-site.xml

```
1 vi yarn-site.xml
```

添加以下内容。

```
1  <configuration>
2      <property>
3          <name>yarn.resourcemanager.hostname</name>
4          <value>lipengda002</value>
5      </property>
6
7      <property>
8          <name>yarn.nodemanager.aux-services</name>
9          <value>mapreduce_shuffle</value>
10     </property>
11
12     <property>
13         <name>yarn.log-aggregation-enable</name>
14         <value>true</value>
15     </property>
16
17     <property>
18         <name>yarn.log-aggregation.retain-seconds</name>
```

```
        <value>420</value>
    </property>
</configuration>
```

```
<!-- Site specific YARN configuration properties -->
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>lipengda002</value>
    </property>

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

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

    <property>
        <name>yarn.log-aggregation.retain-seconds</name>
        <value>420</value>
    </property>
</configuration>
```

图 67: 配置 `yarn-site.xml`

(6) 配置 `slaves`

```
vi slaves
```

添加以下内容。

```
lipengda001
lipengda002
lipengda003
```

```
lipengda001
lipengda002
lipengda003
~
~
```

图 68: 配置 `slaves`

2.7.2 集群节点分发与启动

2.7.2.1 修改/etc/profile 文件与分发

```
1 vi /etc/profile
```

注释掉 HADOOP_HOME 和相关 PATH 的配置。

```
unset i
unset -f pathmunge

export JAVA_HOME=/root/app/jdk1.8.0_131
export PATH=$JAVA_HOME/bin:$PATH

#export HADOOP_HOME=/root/app/hadoop-2.5.0
#export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

图 69: 修改/etc/profile 文件

```
1 source /etc/profile
```

同步到其他虚拟机。

```
1 scp -r /etc/profile lipengda002:/etc/profile
2 scp -r /etc/profile lipengda003:/etc/profile
```

```
[root@lipengda001 hadoop]# scp -r /etc/profile lipengda002:/etc/profile
setterm: $TERM is not defined.
profile                                                    100% 1991   436.1KB/s   00:00
[root@lipengda001 hadoop]# scp -r /etc/profile lipengda003:/etc/profile
setterm: $TERM is not defined.
profile                                                    100% 1991   694.8KB/s   00:00
[root@lipengda001 hadoop]#
```

图 70: 同步/etc/profile 文件

在其他虚拟机上执行 source /etc/profile。

将 hadoop 发送到其他两台虚拟机。

为方便传输，先将文件夹压缩。

```
1 tar -zcvf hadoop.tar.gz hadoop/
```

```
1 scp -r ~/app/hadoop.tar.gz lipengda002:~/app
2 scp -r ~/app/hadoop.tar.gz lipengda003:~/app
```

```
[root@lipengda001 app]# scp -r hadoop.tar.gz root@lipengda003:~/app
setterm: $TERM is not defined.
hadoop.tar.gz                                              100%  299MB  24.5MB/s   00:12
```

图 71: 发送 hadoop 文件

然后在其他两台虚拟机上解压。

```
1 tar -zxvf hadoop.tar.gz -C .
```

2.7.2.2 格式化

```
1 bin/hdfs namenode -format
```

出现以下内容说明格式化成功。

```
24/12/25 17:51:13 INFO namenode.NameNode: Caching file names occuring more than 10 times
24/12/25 17:51:13 INFO util.GSet: Computing capacity for map cachedBlocks
24/12/25 17:51:13 INFO util.GSet: VM type       = 64-bit
24/12/25 17:51:13 INFO util.GSet: 0.25% max memory 966.7 MB = 2.4 MB
24/12/25 17:51:13 INFO util.GSet: capacity      = 2^18 = 262144 entries
24/12/25 17:51:13 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
24/12/25 17:51:13 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
24/12/25 17:51:13 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
24/12/25 17:51:13 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
24/12/25 17:51:13 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expi
ry time is 600000 millis
24/12/25 17:51:13 INFO util.GSet: Computing capacity for map NameNodeRetryCache
24/12/25 17:51:13 INFO util.GSet: VM type       = 64-bit
24/12/25 17:51:13 INFO util.GSet: 0.029999999329447746% max memory 966.7 MB = 297.0 KB
24/12/25 17:51:13 INFO util.GSet: capacity      = 2^15 = 32768 entries
24/12/25 17:51:13 INFO namenode.NNConf: ACLs enabled? false
24/12/25 17:51:13 INFO namenode.NNConf: XAttrs enabled? true
24/12/25 17:51:13 INFO namenode.NNConf: Maximum size of an xattr: 16384
24/12/25 17:51:13 INFO namenode.FSImage: Allocated new BlockPoolId: BP-835559989-192.168.1.110-1735120273260
24/12/25 17:51:13 INFO common.Storage: Storage directory /root/app/hadoop/data/tmp/dfs/name has been successfull
y formatted.
24/12/25 17:51:13 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
24/12/25 17:51:13 INFO util.ExitUtil: Exiting with status 0
24/12/25 17:51:13 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at lipengda001/192.168.1.110
************************************************************/
[root@lipengda001 hadoop]#
```

图 72: 格式化

2.7.2.3 启动

(1) 在 `lipengda001` 虚拟机上启动。

```
1 sbin/start-dfs.sh
```

在三台虚拟机上分别输入 `jps` 查看进程。

```
[root@lipengda001 hadoop]# sbin/start-dfs.sh
Starting namenodes on [lipengda001]
lipengda001: setterm: $TERM is not defined.
lipengda001: starting namenode, logging to /root/app/hadoop/logs/hadoop-root-namenode-lipengda001.out
lipengda001: setterm: $TERM is not defined.
lipengda002: setterm: $TERM is not defined.
lipengda001: starting datanode, logging to /root/app/hadoop/logs/hadoop-root-datanode-lipengda001.out
lipengda003: setterm: $TERM is not defined.
lipengda002: starting datanode, logging to /root/app/hadoop/logs/hadoop-root-datanode-lipengda002.out
lipengda003: starting datanode, logging to /root/app/hadoop/logs/hadoop-root-datanode-lipengda003.out
Starting secondary namenodes [lipengda003]
lipengda003: setterm: $TERM is not defined.
lipengda003: starting secondarynamenode, logging to /root/app/hadoop/logs/hadoop-root-secondarynamenode-lipengda
003.out
[root@lipengda001 hadoop]# jps
3139 Jps
2843 NameNode
2926 DataNode
```

图 73: 启动 (1)(lipengda001)

```
[root@lipengda002 hadoop]# jps
1762 DataNode
1829 Jps
```

图 74: 启动 (1)(lipengda002)

```
[root@lipengda003 hadoop]# jps
1778 SecondaryNameNode
1704 DataNode
1848 Jps
```

图 75: 启动 (1)(lipengda003)

(2) 在 lipengda002 虚拟机上启动 yarn。

```
1 sbin/start-yarn.sh
```

在三台虚拟机上分别输入 jps 查看进程。

```
[root@lipengda002 hadoop]# sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /root/app/hadoop/logs/yarn-root-resourcemanager-lipengda002.out
lipengda003: setterm: $TERM is not defined.
lipengda001: setterm: $TERM is not defined.
lipengda003: starting nodemanager, logging to /root/app/hadoop/logs/yarn-root-nodemanager-lipengda003.out
lipengda001: starting nodemanager, logging to /root/app/hadoop/logs/yarn-root-nodemanager-lipengda001.out
lipengda002: setterm: $TERM is not defined.
lipengda002: starting nodemanager, logging to /root/app/hadoop/logs/yarn-root-nodemanager-lipengda002.out
[root@lipengda002 hadoop]# jps
2848 Jps
1762 DataNode
2504 ResourceManager
2618 NodeManager
```

图 76: 启动 (2)(lipengda002)

```
[root@lipengda001 hadoop]# jps
3921 Jps
3780 NodeManager
2843 NameNode
2926 DataNode
```

图 77: 启动 (2)(lipengda001)

```
[root@lipengda003 hadoop]# jps
1778 SecondaryNameNode
2933 Jps
1704 DataNode
2825 NodeManager
```

图 78: 启动 (2)(lipengda003)

(3) 在 lipengda001 虚拟机上启动 historyserver。

```
1 sbin/mr-jobhistory-daemon.sh start historyserver
```

在三台虚拟机上分别输入 jps 查看进程。

```
[root@lipengda001 hadoop]# jps
4356 NodeManager
4421 Jps
4248 JobHistoryServer
2843 NameNode
2926 DataNode
```

图 79: 启动 (3)(lipengda001)

```
[root@lipengda002 hadoop]# jps
1762 DataNode
3414 Jps
2504 ResourceManager
2618 NodeManager
```

图 80: 启动 (3)(lipengda002)

```
[root@lipengda003 hadoop]# jps
3153 NodeManager
1778 SecondaryNameNode
1704 DataNode
3260 Jps
```

图 81: 启动 (3)(lipengda003)

2.8 分布式离线计算框架—MapReduce

2.8.1 安装 Linux 版本的 eclipse

NOTE

由于我安装的是最小安装的 CentOS，没有安装图形化界面，因此无法安装 eclipse。跳过此步骤。

2.8.2 MapReduce 实例——单词统计

(1) 创建一个 Java 项目 TestHadoop

```
mkdir ~/app/workspace/TestHadoop
cd ~/app/workspace/TestHadoop
```

(2) 导入 jar 包

创建一个 `libs` 文件夹.

```
mkdir ~/libs
```

将 Hadoop 的 `jar` 包拷贝到 `libs` 文件夹。

```
cp ~/app/hadoop/share/hadoop/common/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/hdfs/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/mapreduce/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/yarn/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/common/lib/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/hdfs/lib/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/mapreduce/lib/*.jar ~/libs
cp ~/app/hadoop/share/hadoop/yarn/lib/*.jar ~/libs
```

(3) 创建一个文本文件 word.txt 做测试文件

```
vi word.txt
```

输入以下内容。

```
hello java
java hadoop
spark hbase
hello hadoop
hello word
```

(4) 创建包 `com.hadoop.mapreduce`

```
mkdir -p src/com/hadoop/mapreduce
```

(5) 创建 WordCountMapper.java 文件

```
cd src/com/hadoop/mapreduce
vi WordCountMapper.java
```

输入以下内容。

```
package com.hadoop.mapreduce;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text,
    NullWritable, LongWritable> {
    @Override
    protected void map(LongWritable key, Text value,Mapper<LongWritable,
        Text, NullWritable, LongWritable>.Context context) throws
        IOException, InterruptedException {
        String line = value.toString();
        //用空格进行分割
        String words[] = line.split(" ");
        context.write(NullWritable.get(), new LongWritable(words.length));
    }
}
```

(6) 创建 WordCountReduce.java 文件

```
vi WordCountReduce.java
```

输入以下内容。

```
package com.hadoop.mapreduce;

import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReduce extends Reducer<NullWritable, LongWritable,
    NullWritable, LongWritable> {
    //数组分组合并输出
    @Override
    protected void reduce(NullWritable key, Iterable<LongWritable> v2s,
```

```
        Reducer<NullWritable, LongWritable, NullWritable, LongWritable>.
        Context context) throws IOException, InterruptedException {
        long counter = 0;
        for(LongWritable v:v2s){
            counter += v.get();
        }
        context.write(NullWritable.get(), new LongWritable(counter));
    }
}
```

(7) 创建 WordCount.java 文件

```
vi WordCount.java
```

输入以下内容。

```
package com.hadoop.mapreduce;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

    public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(WordCount.class);
        // Mapper方法名
        job.setMapperClass(WordCountMapper.class);
        // Reducer方法名
        job.setReducerClass(WordCountReduce.class);
        // Map输出的key类型
        job.setMapOutputKeyClass(NullWritable.class);
        // Map输出的value类型
        job.setMapOutputValueClass(LongWritable.class);
        // Reduce输出的key类型
        job.setOutputKeyClass(NullWritable.class);
        // Reduce 输出的value类型
        job.setOutputValueClass(LongWritable.class);
        // 读取的文件位置
```

```
30          FileInputFormat.setInputPaths(job, new Path("file:///root/app/
                workspace/TestHadoop/word.txt"));
31          // 处理完之后的数据存放位置，注意输出的文件夹如果已经存在会报错
32          FileOutputFormat.setOutputPath(job, new Path("file:///root/app/
                workspace/TestHadoop/out"));
33          job.waitForCompletion(true);
34      }
35  }
```

(8) 编译运行

```
1  cd ~/app/workspace/TestHadoop
2  javac -cp ~/libs/*: src/com/hadoop/mapreduce/*.java -d .
3  java -cp ~/libs/*: com.hadoop.mapreduce.WordCount
```

(9) 查看结果

```
1  cd out
2  ls
3  cat part-r-00000
```

```
[root@lipengda001 TestHadoop]# cd out
[root@lipengda001 out]# ls
part-r-00000  _SUCCESS
[root@lipengda001 out]# cat part-r-00000
10
[root@lipengda001 out]#
```

图 82: 查看结果

2.8.3 MapReduce 实例——温度统计

根据 MapReduce 的原理，结合单词统计案例，实现温度统计。需求：找出每年每月的 2 个最低温度时刻并进行升序排列。

(1) 创建一个 Java 项目 TemperatureStats

```
1  mkdir ~/app/workspace/TemperatureStats
2  cd ~/app/workspace/TemperatureStats
```

(2) 创建一个文本文件 temperature.txt 做测试文件

```
1  vi temperature.txt
```

输入以下内容。

```
1  2017-10-05 12:15:30     26
2  2017-11-12 09:36:54     23
```

```
2017-11-16 15:12:12     29
2017-11-17 10:30:59     30
2017-11-23 11:23:15     19
2018-05-16 18:23:23     28
2018-05-21 12:56:30     33
2018-06-03 08:16:15     26
2018-10-15 16:15:20     25
2019-04-09 21:25:26     18
2019-07-16 13:15:16     34
2019-07-22 22:16:56     16
2019-07-23 05:26:11     11
```

(3) 创建包 com.hadoop.temperature

```
mkdir -p src/com/hadoop/temperature
```

(4) 创建 TemperatureMapper.java 文件

```
cd src/com/hadoop/temperature
vi TemperatureMapper.java
```

输入以下内容。

```
package com.hadoop.temperature;

import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TemperatureMapper extends Mapper<Object, Text, Text, Text> {
    @Override
    protected void map(Object key, Text value, Context context) throws
        IOException, InterruptedException {
        String line = value.toString();
        String[] parts = line.split("\\s+");
        if (parts.length == 3) {
            String date = parts[0];
            String time = parts[1];
            String temperature = parts[2];

            String yearMonth = date.substring(0, 7);
            String fullRecord = date + " " + time + " " + temperature;

            Text outputKey = new Text();
            Text outputValue = new Text();

```

```
            outputKey.set(yearMonth);
            outputValue.set(fullRecord);

            context.write(outputKey, outputValue);
        }
    }
}
```

(5) 创建 TemperatureReducer.java 文件

```
vi TemperatureReducer.java
```

输入以下内容。

```
package com.hadoop.temperature;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.PriorityQueue;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class TemperatureReducer extends Reducer<Text, Text, Text, Text> {
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context)
         throws IOException, InterruptedException {
        ArrayList<String> records = new ArrayList<>();

        for (Text value : values) {
            records.add(value.toString());
        }

        Collections.sort(records, new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                int temp1 = Integer.parseInt(o1.split("\\s+")[2]);
                int temp2 = Integer.parseInt(o2.split("\\s+")[2]);
                return Integer.compare(temp1, temp2);
            }
        });

        Text outputValue = new Text();

```

```
        int limit = Math.min(2, records.size());
        for (int i = 0; i < limit; i++) {
            outputValue.set(records.get(i));
            context.write(null, outputValue);
        }
    }
}
```

(6) 创建 TemperatureStats.java 文件

```
vi TemperatureStats.java
```

输入以下内容。

```
package com.hadoop.temperature;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TemperatureStats {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Temperature Stats");
        job.setJarByClass(TemperatureStats.class);
        job.setMapperClass(TemperatureMapper.class);
        job.setReducerClass(TemperatureReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path("file:///root/app/
            workspace/TemperatureStats/temperature.txt"));
        FileOutputFormat.setOutputPath(job, new Path("file:///root/app/
            workspace/TemperatureStats/out"));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

(7) 编译运行

```
cd ~/app/workspace/TemperatureStats
javac -cp ~/libs/*: src/com/hadoop/temperature/*.java -d .
```

```
3 java -cp ~/libs/*: com.hadoop.temperature.TemperatureStats
```

(8) 查看结果

```
1 cd out
2 ls
3 cat part-r-00000
```

```
[root@lipengda001 TemperatureStats]# cd out
[root@lipengda001 out]# ls
part-r-00000  _SUCCESS
[root@lipengda001 out]# cat part-r-00000
2017-10-05 12:15:30 26
2017-11-23 11:23:15 19
2017-11-12 09:36:54 23
2018-05-16 18:23:23 28
2018-05-21 12:56:30 33
2018-06-03 08:16:15 26
2018-10-15 16:15:20 25
2019-04-09 21:25:26 18
2019-07-23 05:26:11 11
2019-07-22 22:16:56 16
```

(按时间升序排列，相同的年月，按温度升序排列)

图 83: 查看结果

3 实验结果

3.1 成功安装虚拟机，并于 Windows 通信

```
pwsh ~                                                    21:19:03
❯ ping lipengda001

Pinging lipengda001 [192.168.1.111] with 32 bytes of data:
Reply from 192.168.1.111: bytes=32 time<1ms TTL=64
Reply from 192.168.1.111: bytes=32 time=1ms TTL=64
Reply from 192.168.1.111: bytes=32 time<1ms TTL=64
Reply from 192.168.1.111: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.111:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 84: Windows Ping 虚拟机

3.2 成功运行 Hadoop 单例模式

```
[root@lipengda001 workspace]# cd out
[root@lipengda001 out]# ls
part-r-00000  _SUCCESS
[root@lipengda001 out]# cat part-r-00000
HelloWorld      1
System.out.println("HelloWorld!");      1
args[]) 1
class   1
main(String     1
public  2
static  1
void    1
{       2
}       2
[root@lipengda001 out]#
```

图 85: 查看 Hadoop 单例模式运行结果

3.3 成功运行 Hadoop 伪分布式模式

```
Are you sure you want to continue connecting (yes/no)? yes
lipengda001: Warning: Permanently added 'lipengda001,192.168.1.111' (ECDSA) to the list of known hos
ts.
root@lipengda001's password:
lipengda001: setterm: $TERM is not defined.
lipengda001: starting namenode, logging to /root/app/hadoop-2.5.0/logs/hadoop-root-namenode-lipengda
001.out
root@lipengda001's password:
lipengda001: setterm: $TERM is not defined.
lipengda001: starting datanode, logging to /root/app/hadoop-2.5.0/logs/hadoop-root-datanode-lipengda
001.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is SHA256:x+4IN5/5swWvMcVYw8trjNCZy/Q/yVg1ewdlBtsnJyk.
ECDSA key fingerprint is MD5:cf:cd:22:b5:0c:d8:a4:99:84:e5:a1:69:90:e8:2a:79.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
root@0.0.0.0's password:
0.0.0.0: setterm: $TERM is not defined.
0.0.0.0: starting secondarynamenode, logging to /root/app/hadoop-2.5.0/logs/hadoop-root-secondarynam
enode-lipengda001.out
[root@lipengda001 hadoop]# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /root/app/hadoop-2.5.0/logs/yarn-root-resourcemanager-lipengda0
01.out
root@lipengda001's password:
lipengda001: setterm: $TERM is not defined.
lipengda001: starting nodemanager, logging to /root/app/hadoop-2.5.0/logs/yarn-root-nodemanager-lipe
ngda001.out
[root@lipengda001 hadoop]# jps
2257 SecondaryNameNode
2118 DataNode
2391 ResourceManager
2760 Jps
2010 NameNode
2670 NodeManager
[root@lipengda001 hadoop]# _
```

图 86: 启动 Hadoop 伪分布式模式

3.4 成功克隆虚拟机并与 Windows 通信

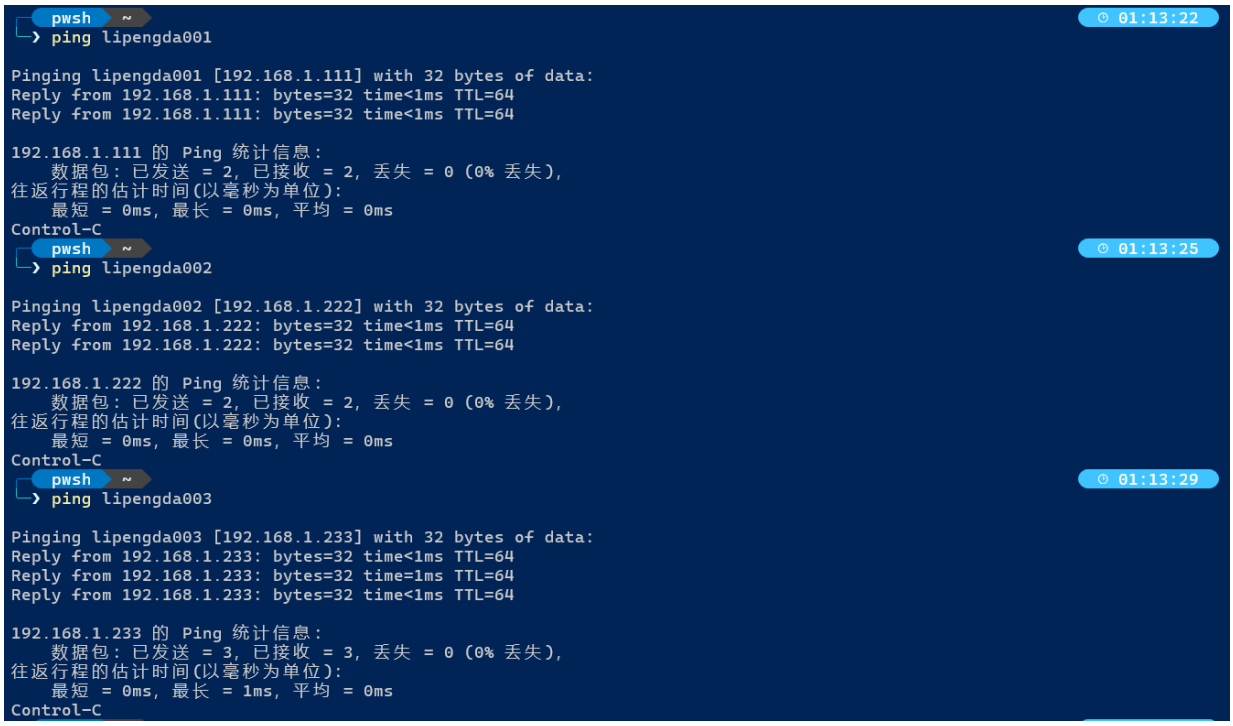

图 87: Windows ping 三个虚拟机

3.5 成功运行 Hadoop 集群模式

```
[root@lipengda001 hadoop]# jps
4356 NodeManager
4421 Jps
4248 JobHistoryServer
2843 NameNode
2926 DataNode
```

图 88: 集群模式 (`lipengda001`)

```
[root@lipengda002 hadoop]# jps
1762 DataNode
3414 Jps
2504 ResourceManager
2618 NodeManager
```

图 89: 集群模式 (`lipengda002`)

```
[root@lipengda003 hadoop]# jps
3153 NodeManager
1778 SecondaryNameNode
1704 DataNode
3260 Jps
```

图 90: 集群模式 (`lipengda003`)

3.6 成功使用 MapReduce 框架进行单词统计

```
[root@lipengda001 TestHadoop]# cd out
[root@lipengda001 out]# ls
part-r-00000  _SUCCESS
[root@lipengda001 out]# cat part-r-00000
10
[root@lipengda001 out]#
```

图 91: MapReduce 单词统计结果

3.7 成功使用 MapReduce 框架进行温度统计

```
[root@lipengda001 TemperatureStats]# cd out
[root@lipengda001 out]# ls
part-r-00000  _SUCCESS
[root@lipengda001 out]# cat part-r-00000
2017-10-05 12:15:30 26
2017-11-23 11:23:15 19
2017-11-12 09:36:54 23
2018-05-16 18:23:23 28
2018-05-21 12:56:30 33
2018-06-03 08:16:15 26
2018-10-15 16:15:20 25
2019-04-09 21:25:26 18
2019-07-23 05:26:11 11
2019-07-22 22:16:56 16
```

(按时间升序排列，相同的年月，按温度升序排列)

图 92: MapReduce 温度统计结果

4 实验总结

在本次实验中，我完成了 Linux 虚拟机的配置、Hadoop 的单例、伪分布式和集群模式的搭建，以及 MapReduce 框架的使用。在实验过程中，我学会了如何配置 Linux 虚拟机网络、ssh 与时钟同步，学习了搭建 Hadoop 集群，以及使用 MapReduce 框架进行了单词统计。通过本次实验，我对 Hadoop 有了更深入的了解，也对云计算有了更多的认识。

在实验中，我遇到的问题及解决方法、一些与实验手册不同的地方和我自己的思考，我记录在了蓝色“NOTE”框中。