

```
export PATH=$PATH:/usr/local/hadoop/sbin
```

在后面的学习过程中,如果要继续把其他命令的路径也加入到 PATH 变量中,也需要继续修改 `~/.bashrc` 这个文件。当后面要继续加入新的路径时,只要用英文冒号“:”隔开,把新的路径加到后面即可,例如,如果要继续把 `/usr/local/hadoop/bin` 路径增加到 PATH 中,只要继续追加到后面,如下所示:

```
export PATH=$PATH:/usr/local/hadoop/sbin:/usr/local/hadoop/bin
```

添加后,执行命令 `source ~/.bashrc` 使设置生效。设置生效后,在任何目录下启动 Hadoop,都只要直接输入 `start-dfs.sh` 命令即可。同理,停止 Hadoop,也只需要在任何目录下输入 `stop-dfs.sh` 命令即可。

3.3.4 分布式模式配置

当 Hadoop 采用分布式模式部署和运行时,存储采用分布式文件系统 HDFS,而且, HDFS 的名称节点和数据节点位于不同机器上。这时,数据就可以分布到多个节点上,不同数据节点上的数据计算可以并行执行,这时的 MapReduce 分布式计算能力才能真正发挥作用。

为了降低分布式模式部署的难度,本书简单使用两个节点(两台物理机器)来搭建集群环境:一台机器作为 Master 节点,局域网 IP 地址为 192.168.1.121;另一台机器作为 Slave 节点,局域网 IP 地址为 192.168.1.122。由 3 个以上节点构成的集群,也可以采用类似的方法完成安装部署。

Hadoop 集群的安装配置大致包括以下步骤。

- (1) 选定一台机器作为 Master。
- (2) 在 Master 节点上创建 hadoop 用户、安装 SSH 服务端、安装 Java 环境。
- (3) 在 Master 节点上安装 Hadoop,并完成配置。
- (4) 在其他 Slave 节点上创建 hadoop 用户、安装 SSH 服务端、安装 Java 环境。
- (5) 将 Master 节点上的 `/usr/local/hadoop` 目录复制到其他 Slave 节点上。
- (6) 在 Master 节点上开启 Hadoop。

上述这些步骤中,关于如何创建 hadoop 用户、安装 SSH 服务端、安装 Java 环境、安装 Hadoop 等过程,已经在前面介绍伪分布式安装的时候做了详细介绍,按照之前介绍的方法完成步骤(1)到步骤(4),这里不再赘述。在完成步骤(1)到步骤(4)的操作以后,才可以继续进行下面的操作。

1. 网络配置

假设集群所用的两个节点(机器)都位于同一个局域网内。如果两个节点使用的是虚拟机安装的 Linux 系统,那么两者都需要更改网络连接方式为“桥接网卡”模式(可以参考第 2 章中介绍的网络连接设置方法),才能实现多个节点互连,如图 3-11 所示。此外,一定要确保各个节点的 MAC 地址不能相同,否则会出现 IP 冲突。在第 2 章曾介绍过采用导入虚拟

机镜像文件的方式安装 Linux 系统,如果是采用这种方式安装 Linux 系统,则有可能出现两台机器的 MAC 地址是相同的,因为一台机器复制了另一台机器的配置。因此,需要改变机器的 MAC 地址,如图 3-11 所示,可以单击界面右边的“刷新”按钮随机生成 MAC 地址,这样就可以让两台机器的 MAC 地址不同了。

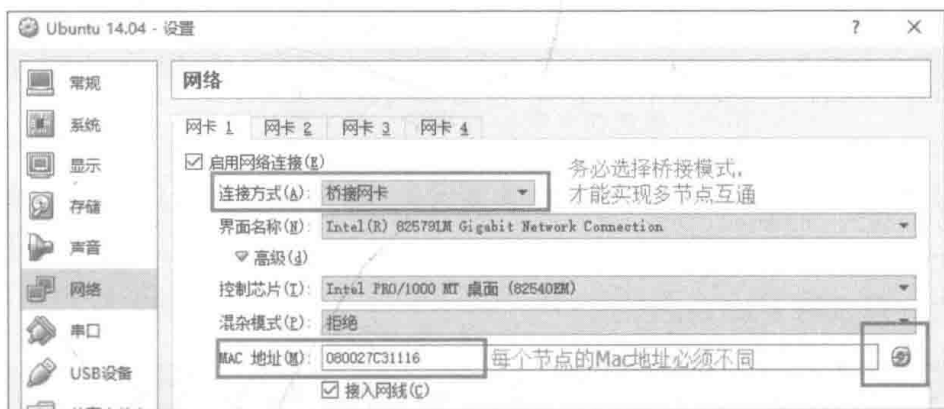


图 3-11 网络连接方式设置

网络配置完成以后,可以查看一下机器的 IP 地址,可以使用在第 2 章介绍过的 `ifconfig` 命令查看。本书在同一个局域网内部的两台机器的 IP 地址分别是 192.168.1.121 和 192.168.1.122。

由于集群中有两台机器需要设置,所以,在接下来的操作中,一定要注意区分 Master 节点和 Slave 节点。为了便于区分 Master 节点和 Slave 节点,可以修改各个节点的主机名,这样,在 Linux 系统中打开一个终端以后,在终端窗口的标题和命令行中都可以看到主机名,就比较容易区分当前是对哪台机器进行操作。在 Ubuntu 中,在 Master 节点上执行如下命令修改主机名:

```
$sudo vim /etc/hostname
```

执行上面命令后,就打开了 `/etc/hostname` 这个文件,这个文件里面记录了主机名,例如,本书在第 2 章中安装 Ubuntu 系统时,设置的主机名是 `dblab-VirtualBox`,因此,打开这个文件以后,里面就只有 `dblab-VirtualBox` 这一行内容,可以直接删除,并修改为 Master (注意是区分大小写的);然后保存并退出 vim 编辑器,这样就完成了主机名的修改,需要重启 Linux 系统才能看到主机名的变化。

要注意观察主机名修改前后的变化。在修改主机名之前,如果用 hadoop 登录 Linux 系统,打开终端,进入 Shell 命令提示符状态,会显示如下内容:

```
hadoop@dblab-VirtualBox:~$
```

修改主机名并且重启系统之后,用 hadoop 登录 Linux 系统,打开终端,进入 Shell 命令提示符状态,会显示如下内容:

```
hadoop@Master:~$
```

可以看出,这时就很容易辨认出当前是处于 Master 节点上进行操作,不会和 Slave 节点产生混淆。

然后执行如下命令打开并修改 Master 节点中的/etc/hosts 文件:

```
$sudo vim /etc/hosts
```

可以在 hosts 文件中增加如下两条 IP 和主机名映射关系:

```
192.168.1.121 Master
192.168.1.122 Slave1
```

修改后的效果如图 3-12 所示。

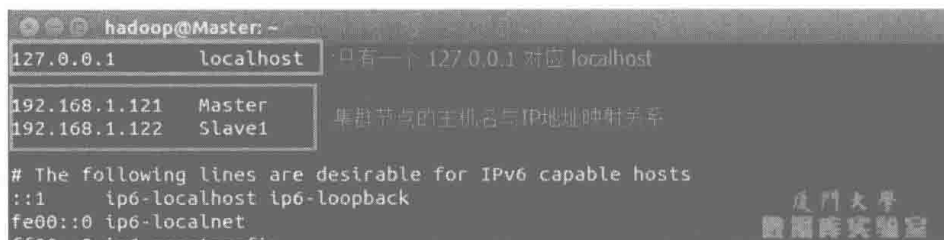


图 3-12 修改 IP 和主机名映射关系后的效果

需要注意的是,一般 hosts 文件中只能有一个 127.0.0.1,其对应主机名为 localhost,如果有多余 127.0.0.1 映射,应删除,特别是不能存在 127.0.0.1 Master 这样的映射记录。修改后需要重启 Linux 系统。

上面完成了 Master 节点的配置,接下来要继续完成对其他 Slave 节点的配置修改。本书只有一个 Slave 节点,主机名为 Slave1。参照上面的方法,把 Slave 节点上的/etc/hostname 文件中的主机名修改为 Slave1,同时,修改/etc/hosts 的内容,在 hosts 文件中增加如下两条 IP 和主机名映射关系:

```
192.168.1.121 Master
192.168.1.122 Slave1
```

修改完成以后,重新启动 Slave 节点的 Linux 系统。

这样就完成了 Master 节点和 Slave 节点的配置,然后,需要在各个节点上都执行如下命令,测试是否相互 ping 得通,如果 ping 不通,后面就无法顺利配置成功:

```
$ping Master-c 3      #只 ping 3 次就会停止,否则要按 Ctrl+C 键中断 ping 命令
$ping Slave1-c 3
```

例如,在 Master 节点上 ping Slave1,如果 ping 通的话,会显示如图 3-13 所示的结果。

2. SSH 无密码登录节点

必须要让 Master 节点可以 SSH 无密码登录到各个 Slave 节点上。首先,生成 Master

```

hadoop@Master: ~
hadoop@Master:~$ ping Slave1 -c 3
PING Slave1 (192.168.1.122) 56(84) bytes of data:
64 bytes from Slave1 (192.168.1.122): icmp_seq=1 ttl=64 time=0.315 ms
64 bytes from Slave1 (192.168.1.122): icmp_seq=2 ttl=64 time=0.427 ms
64 bytes from Slave1 (192.168.1.122): icmp_seq=3 ttl=64 time=0.338 ms

--- Slave1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.315/0.360/0.427/0.048 ms

```

图 3-13 使用 ping 命令的效果

节点的公匙,如果之前已经生成过公匙,必须要删除原来生成的公匙,重新生成一次,因为前面对主机名进行了修改。具体命令如下:

```

$cd ~/.ssh          #如果没有该目录,先执行一次 ssh localhost
$rm ./id_rsa *      #删除之前生成的公匙(如果已经存在)
$ssh-keygen -t rsa   #执行该命令后,遇到提示信息,一直按 Enter 键就可以

```

为了让 Master 节点能够无密码 SSH 登录本机,需要在 Master 节点上执行如下命令:

```
$cat ./id_rsa.pub>>./authorized_keys
```

完成后可以执行命令 `ssh Master` 来验证一下,可能会遇到提示信息,只要输入 `yes` 即可,测试成功后,执行 `exit` 命令返回原来的终端。

接下来在 Master 节点将上公匙传输到 Slave1 节点:

```
scp ~/.ssh/id_rsa.pub hadoop@ Slave1:/home/hadoop/
```

上面的命令中,scp 是 secure copy 的简写,用于在 Linux 下进行远程复制文件,类似于 cp 命令,不过,cp 只能在本机中复制。执行 scp 时会要求输入 Slave1 上 hadoop 用户的密码,输入完成后会提示传输完毕,如图 3-14 所示。

```

hadoop@Master: ~/.ssh
hadoop@Master:~/.ssh$ scp ~/.ssh/id_rsa.pub hadoop@Slave1:/home/hadoop/
The authenticity of host 'slave1 (192.168.1.122)' can't be established.
ECDSA key fingerprint is e3:40:14:58:1c:37:4d:21:a0:24:bf:00:e6:a0:fb:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave1,192.168.1.122' (ECDSA) to the list of known h
osts.
hadoop@slave1's password:
id_rsa.pub                                100% 395    0.4KB/s   00:00
hadoop@Master:~/.ssh$

```

图 3-14 执行 scp 命令的效果

接着在 Slave1 节点上将 SSH 公匙加入授权:

```

$mkdir ~/.ssh          #如果不存在该文件夹需先创建,若已存在,则忽略本命令
$cat ~/id_rsa.pub>>~/ .ssh/authorized_keys
$rm ~/id_rsa.pub        #用完以后就可以删掉

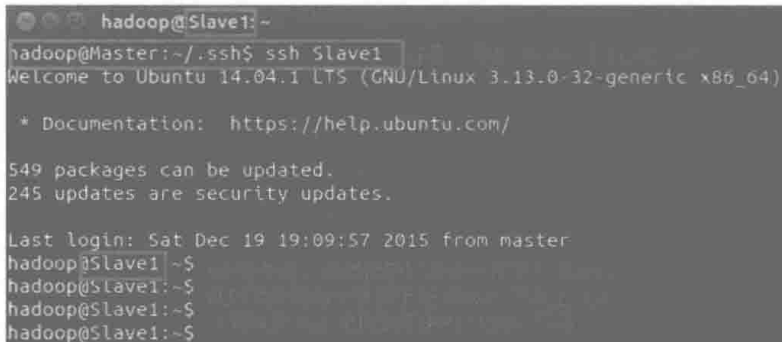
```

如果有其他 Slave 节点,也要执行将 Master 公匙传输到 Slave 节点以及在 Slave 节点上加入授权这两步操作。

这样,在 Master 节点上就可以无密码 SSH 登录到各个 Slave 节点了,可在 Master 节点上执行如下命令进行检验:

```
$ssh Slave1
```

执行该命令的效果如图 3-15 所示。



```
hadoop@Slave1:~  
hadoop@Master:~/.ssh$ ssh Slave1  
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)  
  
* Documentation: https://help.ubuntu.com/  
  
549 packages can be updated.  
245 updates are security updates.  
  
Last login: Sat Dec 19 19:09:57 2015 from master  
hadoop@Slave1 ~$  
hadoop@Slave1:~$  
hadoop@Slave1:~$  
hadoop@Slave1:~$
```

图 3-15 ssh 命令执行效果

3. 配置 PATH 变量

在前面的伪分布式安装内容中,已经介绍过 PATH 变量的配置方法。可以按照同样的方法进行配置,这样就可以在任意目录中直接使用 hadoop、hdfs 等命令了。如果还没有配置 PATH 变量,那么需要在 Master 节点上进行配置。首先执行命令 `vim ~/.bashrc`,也就是使用 vim 编辑器打开 `~/.bashrc` 文件;然后,在该文件最上面的位置加入下面一行内容:

```
export PATH=$PATH:/usr/local/hadoop/bin:/usr/local/hadoop/sbin
```

保存后执行命令 `source ~/.bashrc`,使配置生效。

4. 配置集群/分布式环境

在配置集群/分布式模式时,需要修改 `/usr/local/hadoop/etc/hadoop` 目录下的配置文件,这里仅设置正常启动所必需的设置项,包括 `slaves`、`core-site.xml`、`hdfs-site.xml`、`mapred-site.xml`、`yarn-site.xml` 共 5 个文件,更多设置项可查看官方说明。

1) 修改文件 slaves

需要把所有数据节点的主机名写入该文件,每行一个,默认为 localhost(即把本机作为数据节点),所以,在伪分布式配置时,就采用了这种默认的配置,使得节点既作为名称节点也作为数据节点。在进行分布式配置时,可以保留 localhost,让 Master 节点同时充当名称节点和数据节点,或者也可以删掉 localhost 这行,让 Master 节点仅作为名称节点使用。

本书让 Master 节点仅作为名称节点使用,因此将 slaves 文件中原来的 localhost 删除,只添加如下一行内容:

Slave1

2) 修改文件 core-site.xml

把 core-site.xml 文件修改为如下内容:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://Master:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
</configuration>
```

各个配置项的含义可以参考前面伪分布式模式时的介绍,这里不再赘述。

3) 修改文件 hdfs-site.xml

对于 Hadoop 的分布式文件系统 HDFS 而言,一般都是采用冗余存储,冗余因子通常为 3,也就是说,一份数据保存 3 份副本。但是,本书只有一个 Slave 节点作为数据节点,即集群中只有一个数据节点,数据只能保存一份,所以,dfs.replication 的值还是设置为 1。hdfs-site.xml 的具体内容如下:

```
<configuration>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>Master:50090</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/tmp/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/tmp/dfs/data</value>
  </property>
</configuration>
```

4) 修改文件 mapred-site.xml

/usr/local/hadoop/etc/hadoop 目录下有一个 mapred-site.xml.template, 需要修改文件名称, 把它重命名为 mapred-site.xml, 然后把 mapred-site.xml 文件配置成如下内容:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>Master:10020</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>Master:19888</value>
  </property>
</configuration>
```

5) 修改文件 yarn-site.xml

把 yarn-site.xml 文件配置成如下内容:

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>Master</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

上述 5 个文件全部配置完成以后, 需要把 Master 节点上的 /usr/local/hadoop 文件夹复制到各个节点上。如果之前已经运行过伪分布式模式, 建议在切换到集群模式之前首先删除之前在伪分布式模式下生成的临时文件。具体来说, 需要首先在 Master 节点上执行如下命令:

```
$cd /usr/local
$sudo rm -r ./hadoop/tmp           #删除 Hadoop 临时文件
$sudo rm -r ./hadoop/logs/*       #删除日志文件
$tar -zcf ~/hadoop.master.tar.gz ./hadoop  #先压缩再复制
$cd ~
$scp ./hadoop.master.tar.gz Slave1:/home/hadoop
```

然后在 Slave1 节点上执行如下命令：

```
$sudo rm -r /usr/local/hadoop      #删掉旧的(如果存在)
$sudo tar -zxf ~/hadoop.master.tar.gz -C /usr/local
$sudo chown -R hadoop /usr/local/hadoop
```

同样,如果有其他 Slave 节点,也要执行将 hadoop.master.tar.gz 传输到 Slave 节点以及在 Slave 节点解压文件的操作。

首次启动 Hadoop 集群时,需要先在 Master 节点执行名称节点的格式化(只需要执行这一次,后面再启动 Hadoop 时,不要再次格式化名称节点),命令如下:

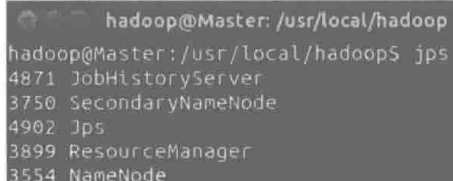
```
$hdfs namenode -format
```

现在就可以启动 Hadoop 了,启动需要在 Master 节点上进行,执行如下命令:

```
$start-dfs.sh
$start-yarn.sh
$mr-jobhistory-daemon.sh start historyserver
```

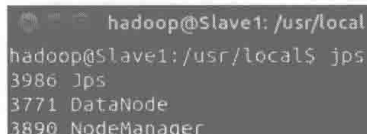
通过命令 jps 可以查看各个节点所启动的进程。如果已经正确启动,则在 Master 节点上可以看到 NameNode、ResourceManager、SecondaryNameNode 和 JobHistoryServer 进程,如图 3-16 所示。

在 Slave 节点可以看到 DataNode 和 NodeManager 进程,如图 3-17 所示。



```
hadoop@Master: /usr/local/hadoop
hadoop@Master: /usr/local/hadoop$ jps
4871 JobHistoryServer
3750 SecondaryNameNode
4902 Jps
3899 ResourceManager
3554 NameNode
```

图 3-16 Master 节点上启动的进程



```
hadoop@Slave1: /usr/local
hadoop@Slave1: /usr/local$ jps
3986 Jps
3771 DataNode
3890 NodeManager
```

图 3-17 Slave 节点上启动的进程

缺少任一进程都表示出错。另外还需要在 Master 节点上通过命令 `hdfs dfsadmin -report` 查看数据节点是否正常启动,如果屏幕信息中的 Live datanodes 不为 0,则说明集群启动成功。由于本书只有一个 Slave 节点充当数据节点,因此,数据节点启动成功以后,会显示如图 3-18 所示的信息。

也可以在 Linux 系统的浏览器中输入地址 `http://master:50070/`,通过 Web 页面看到查看名称节点和数据节点的状态。如果不成功,可以通过启动日志排查原因。

这里再次强调,伪分布式模式和分布式模式切换时需要注意以下事项。

(1) 从分布式切换到伪分布式时,不要忘记修改 slaves 配置文件。

(2) 在两者之间切换时,若遇到无法正常启动的情况,可以删除所涉及节点的临时文件夹,这样虽然之前的数据会被删掉,但能保证集群正确启动。所以,如果集群以前能启动,但后来启动不了,特别是数据节点无法启动,不妨试着删除所有节点(包括 Slave 节点)上的 `/usr/local/hadoop/tmp` 文件夹,再重新执行一次 `hdfs namenode -format`,再次启动即可。


```

hadoop@Master: /usr/local/hadoop
-----
Live datanodes (1):
Name: 192.168.1.122:50010 (Slave1)
Hostname: Slave1
Decommission Status : Normal
Configured Capacity: 7262953472 (6.76 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 5365833728 (5.00 GB)
DFS Remaining: 1897095168 (1.77 GB)
DFS Used%: 0.00%
DFS Remaining%: 26.12%
Configured Cache Capacity: 0 (0 B)

```

图 3-18 通过 dfsadmin 查看数据节点的状态

5. 执行分布式实例

执行分布式实例过程与伪分布式模式一样,首先创建 HDFS 上的用户目录,命令如下:

```
$hdfs dfs -mkdir -p /user/hadoop
```

然后在 HDFS 中创建一个 input 目录,并把 /usr/local/hadoop/etc/hadoop 目录中的配置文件作为输入文件复制到 input 目录中,命令如下:

```
$hdfs dfs -mkdir input
$hdfs dfs -put /usr/local/hadoop/etc/hadoop/* .xml input
```

接着就可以运行 MapReduce 作业了,命令如下:

```
$hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar grep input output 'dfs[a-z.]+'
```

运行时的输出信息与伪分布式类似,会显示 MapReduce 作业的进度,如图 3-19 所示。

```

hadoop@Master: /usr/local/hadoop
15/12/19 20:09:53 INFO mapreduce.Job: Running job: job_1450525734982_0001
15/12/19 20:10:07 INFO mapreduce.Job: Job job_1450525734982_0001 running in uber
mode : false
15/12/19 20:10:07 INFO mapreduce.Job: map 0% reduce 0%
15/12/19 20:10:59 INFO mapreduce.Job: map 67% reduce 0%
15/12/19 20:11:38 INFO mapreduce.Job: map 89% reduce 0%
15/12/19 20:11:40 INFO mapreduce.Job: map 100% reduce 0%
15/12/19 20:11:42 INFO mapreduce.Job: map 100% reduce 67%
15/12/19 20:11:45 INFO mapreduce.Job: map 100% reduce 100%
15/12/19 20:11:46 INFO mapreduce.Job: Job job_1450525734982_0001 completed succe
ssfully

```

图 3-19 运行 MapReduce 作业时的输出信息

执行过程可能会有点慢,但是,如果迟迟没有进度,例如 5 分钟都没看到进度变化,那么不妨重启 Hadoop 再次测试。若重启还不行,则很有可能是内存不足引起,建议增大虚拟机的内存,或者通过更改 YARN 的内存配置来解决。

在执行过程中,可以在 Linux 系统中打开浏览器,在地址栏输入 <http://master:8088/>

cluster,通过 Web 界面查看任务进度,在 Web 界面中单击 Tracking UI 这一列的 History 连接,可以看到任务的运行信息,如图 3-20 所示。

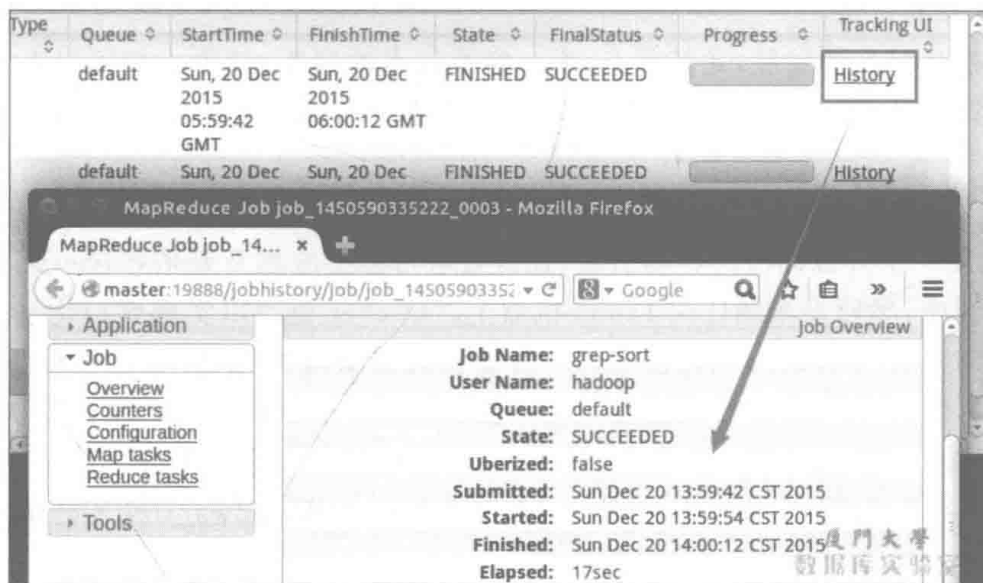


图 3-20 通过 Web 页面查看集群和 MapReduce 作业的信息

执行完毕后的输出结果如图 3-21 所示。

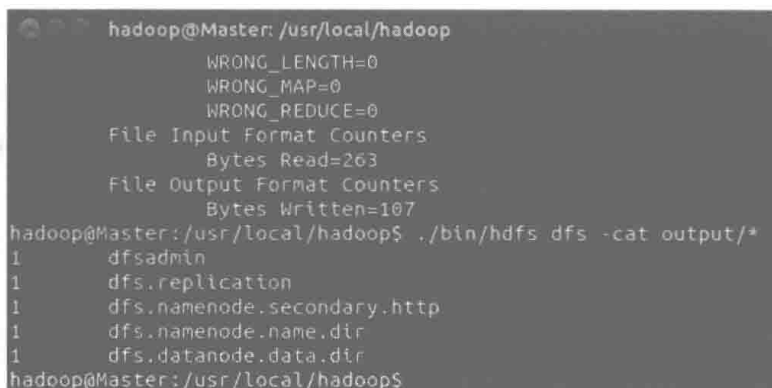


图 3-21 MapReduce 作业执行完毕后的信息

最后关闭 Hadoop 集群,需要在 Master 节点执行如下命令:

```
$stop-yarn.sh
$stop-dfs.sh
$mr-jobhistory-daemon.sh stop historyserver
```

至此,就顺利完成了 Hadoop 集群搭建。

3.3.5 使用 Docker 搭建 Hadoop 分布式集群

搭建 Hadoop 分布式集群通常容易想到的两种方法如下:

- (1) 采用多台机器构建分布式集群;
- (2) 在一台机器上,安装多个虚拟机,每个虚拟机上运行一个 Hadoop 节点。

但是,上述两种方式都有缺点。如果采用第一种方法,通常需要有多台机器,对于很多

大数据学习者而言,通常难以找到多台机器用于构建分布式实验环境;如果采用第二种方法,通常对单台机器的配置要求很高,如果机器配置较低,那么在一台机器上同时运行多个虚拟机,速度会非常慢。

随着虚拟化技术的发展,尤其是 Docker 容器技术的诞生,使得人们可以有了第 3 种构建 Hadoop 分布式集群的方法——使用 Docker 搭建 Hadoop 分布式集群。

1. Docker 简介

Docker 是一个开源的应用容器引擎,让开发者可以把应用及其依赖包一起打包到一个可移植的容器中,然后发布到任何 Linux 机器上。Docker 也可以实现虚拟化,而且,Docker 是不同于 VMware 等传统虚拟化技术的一种新型轻量级虚拟化技术(也被称为“容器型虚拟化技术”)。与 VMware 等传统虚拟化技术相比,Docker 容器具有启动速度快、资源利用率高、性能开销小等优点,受到业界青睐,并得到了越来越广泛的应用。不过,需要注意的是,Docker 是基于 64 位 Linux 的,无法在 32 位的 Linux/Windows/UNIX 环境下使用。

2. 安装 Docker

安装 Docker 之前,必须首先保证机器上安装的是 64 位 Linux 系统;其次,内核版本必须大于 3.10。可以用如下命令来检测机器上已经安装好的 Ubuntu 系统的内核版本:

```
$uname -r
```

然后需要先更新 apt,安装 CA 证书,因为访问 Docker 使用的是 HTTPS 协议,命令如下:

```
$sudo apt-get update
$sudo apt-get install apt-transport-https ca-certificates
```

执行如下命令添加新的 GPG key:

```
$sudo apt-key adv \
    --keyserver hkp://ha.pool.sks-keyservers.net:80 \
    --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

执行如下命令为 Ubuntu 系统添加 Docker 安装源(也就是可以获得 Docker 安装包的地方):

```
$echo deb https://apt.dockerproject.org/repo ubuntu-xenial main | sudo tee
/etc/apt/sources.list.d/docker.list
```

执行如下命令更新 apt 软件列表:

```
$sudo apt-get update
```

接着可以用如下命令验证一下是否从正确的仓库拉取安装包：

```
$apt-cache policy docker-engine
```

如果有类似于下面的输出，则说明从正确的仓库获取包：

```
docker-engine:
  Installed: 1.12.2-0~trusty
  Candidate: 1.12.2-0~trusty
  Version table:
*** 1.12.2-0~trusty 0
    500 https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
    100 /var/lib/dpkg/status
 1.12.1-0~trusty 0
    500 https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
 1.12.0-0~trusty 0
    500 https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
```

接下来可以直接安装 Docker，命令如下：

```
$sudo apt-get install docker-engine
```

等这个命令结束之后，Docker 即安装完成。可以通过下面命令开启 Docker 服务：

```
$sudo service docker start
```

然后可以运行 Docker 官方提供的 hello-world 程序来检测 Docker 是否能够顺利运行程序：

```
$sudo docker run hello-world
```

执行上述命令以后，屏幕上会返回很多信息，如果其中包含如下所示信息，则表示安装成功：

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

对于 Docker 而言，默认情况下只有 root 用户才能执行 Docker 命令，因此，还需要添加用户权限。首先需要使用如下命令创建 Docker 用户组：

```
$sudo groupadd docker
```

然后添加当前用户到 Docker 用户组，命令如下：

```
$sudo usermod -aG docker hadoop
```

由于本书全部采用 hadoop 用户登录 Linux 系统,所以,上面的命令中使用了 hadoop 用户名。执行上述命令以后,需要注销当前登录,然后再次用 hadoop 用户登录系统,就可以顺畅地执行 Docker 命令了。

3. 在 Docker 上安装 Ubuntu 系统

安装好 Docker 之后,接下来就要在 Docker 上安装 Ubuntu。可以直接从 Docker 上下载 Ubuntu 镜像文件,命令如下:

```
$docker pull ubuntu
```

docker pull 命令表示从 Docker hub 上拉取 Ubuntu 镜像到本地。执行该命令以后,可以在执行下面命令查看是否安装成功:

```
$docker images
```

docker images 表示列出 Docker 上所有的镜像。镜像也是一堆文件,如果输出类似如下的信息则表示安装成功:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	4ca3a192ff2a	11days ago	128.2 MB

然后需要在 Docker 上开启 Ubuntu 系统,也就是启动 Ubuntu 镜像。在启动 Ubuntu 镜像之前,需要先在当前登录用户(hadoop)的用户目录下创建一个子目录,用于向 Docker 内部的 Ubuntu 系统传输文件,请在本地 Ubuntu 系统(不是 Docker 中的 Ubuntu 系统)中执行如下命令:

```
$cd ~  
$mkdir build
```

现在就可以在 Docker 上启动运行 Ubuntu 系统了,在本地 Ubuntu 系统中执行如下命令:

```
$docker run -it -v /home/hadoop/build:/root/build--name ubuntu ubuntu
```

上面命令的含义如下。

(1) docker run 表示运行一个镜像。

(2) -i 表示开启交互式;-t 表示分配一个 tty(可以理解为一个控制台);因此,-it 可以理解为在当前终端上与 Docker 内部的 Ubuntu 系统进行交互。

(3) -v 表示 Docker 内部的 Ubuntu 系统中的/root/build 目录与本地的/home/

hadoop/build 目录共享,这样就可以很方便地将本地文件上传到 Docker 内部的 Ubuntu 系统中。

(4) `--name ubuntu` 表示 Ubuntu 镜像启动名称,如果没有指定,那么 Docker 将会随机分配一个名字。

(5) `ubuntu`: 命令中的最后一个 `ubuntu`,表示 `docker run` 启动的镜像文件。

4. Ubuntu 系统初始化

刚安装好的 Ubuntu 系统,是一个很纯净的系统,很多软件是没有安装的,所以需要先更新一下 Ubuntu 系统的软件源并安装一些必备的软件。

1) 更新系统软件源

在 Docker 上的 Ubuntu 系统(不是本地 Ubuntu 系统)中执行下面命令更新软件源:

```
$apt-get update
```

2) 安装 vim 编辑器

在 Docker 上的 Ubuntu 系统中执行下面命令安装 vim 编辑器:

```
$apt-get install vim
```

3) 安装和配置 sshd

接着安装 `sshd`,因为在开启分布式 Hadoop 时,需要用到 SSH 连接 Slave 机器,在 Docker 上的 Ubuntu 系统中执行如下命令:

```
$apt-get install ssh
```

然后在 Docker 上的 Ubuntu 系统中运行如下脚本即可开启 `sshd` 服务器:

```
$/etc/init.d/ssh start
```

但是,这样每次在启动镜像时,都需要手动开启 `sshd` 服务,因此,可以把启动命令写进 `~/.bashrc` 文件,从而在每次登录 Ubuntu 系统时,都能自动启动 `sshd` 服务。用 vim 编辑器打开 `~/.bashrc` 文件,在该文件最后一行添加如下内容:

```
/etc/init.d/ssh start
```

安装好 `sshd` 之后,需要配置 SSH 无密码连接本地 `sshd` 服务,在 Docker 上的 Ubuntu 系统中执行如下命令:

```
$ssh-keygen -t rsa #一直按 Enter 键即可  
$cd ~/.ssh  
$cat id_dsa.pub>>authorized_keys
```

执行完上述命令之后,即可无密码访问本地 sshd 服务。

4) 安装 Java

因为 Hadoop 需要使用 Java 环境,因此,需要安装 JDK。可以在 Docker 上的 Ubuntu 系统中直接输入以下命令来安装 JDK:

```
$apt-get install default-jdk
```

这个命令会安装比较多的库,并会消耗较长时间。等待这个命令运行结束之后,JDK 安装就顺利完成了。然后,需要配置环境变量,打开 `~/.bashrc` 文件,在文件的最开始位置增加如下内容:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/  
export PATH=$PATH:$JAVA_HOME/bin
```

接着在 Docker 上的 Ubuntu 系统中执行如下命令使 `~/.bashrc` 配置文件生效:

```
$source ~/.bashrc
```

5) 保存镜像文件

在 Docker 内部对容器做的修改,是不会自动保存到镜像中的,也就是说,上面尽管对容器进行了大量配置,但是,一旦把容器关闭,然后重新开启容器,则之前的设置会全部消失;因此,需要保存当前的容器配置。为了达到复用容器的配置信息,在每个步骤完成之后,都应该保存成一个新的镜像,然后开启并运行保存后的新镜像即可。为了把容器配置保存成新的镜像,首先需要到 Docker 官网(<https://hub.docker.com/>)注册一个账号,账号注册成功后,然后在本地 Ubuntu 系统(不是 Docker 中的 Ubuntu 系统)中新建一个终端,输入如下命令:

```
$docker login
```

然后会有如下提示信息,输入上面注册的用户名和密码即可:

```
Login with your Docker ID to push and pull images from Docker Hub. If you don't  
have a Docker ID, head over to https://hub.docker.com to create one.  
Username:  
Password:  
Login Succeeded
```

登录之后,首先在本地 Ubuntu 系统中使用如下命令查看当前运行的容器的信息:

```
$docker ps
```

输出结果如下：

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
fd1fc69d75a3	ubuntu	"/bin/bash"	About an hour ago	Up	About an hour	ubuntu

从上面的结果信息可以看出,当前运行的 Ubuntu 镜像的 ID 是 fd1fc69d75a3。然后在本地 Ubuntu 系统中输入以下命令,把修改后的容器保存成为一个新的镜像,新镜像的名称是 ubuntu/jdkinstalled,表示该镜像中已经包含了 JDK:

```
$docker commit fd1fc69d75a3 ubuntu/jdkinstalled
```

其中,docker commit 后面的 fd1fc69d75a3 是当前运行的 Ubuntu 镜像的 ID。输出结果如下:

```
sha256:05c9bc2b849359d029d417421d6968bdd239aad447ac6054c429b630378118aa
```

最后在本地 Ubuntu 系统的另一个终端中使用如下命令查看所有镜像,确认一下新的镜像是否保存成功:

```
$docker images
```

输出结果如下:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu/jdkinstalled	latest	05c9bc2b8493	9 seconds ago	899.3 MB
ubuntu	latest	4ca3a192ff2a	12 days ago	128.2 MB

从上面输出结果可以看出,目前系统中已经有两个镜像:一个是 ubuntu 镜像;另一个是刚才新建的镜像 ubuntu/jdkinstalled。

5. 安装 Hadoop

下面开始安装 Hadoop。需要首先在本地 Ubuntu 系统中启动之前保存的镜像 ubuntu/jdkinstalled,命令如下:

```
$docker run -it -v /home/hadoop/build:/root/build--name ubuntu-jdkinstalled  
ubuntu/jdkinstalled
```

可以在本地 Ubuntu 系统中的另一个终端中使用如下命令查看启动的容器:

```
$docker ps
```

输出结果如下:


```
$CONTAINER ID
325a1c2f2f93
IMAGE
ubuntu/jdkinstalled
COMMAND
"/bin/bash"
CREATED
6 seconds ago
STATUS
Up 5 seconds
PORTS NAMES
ubuntu-jdkinstalled
```

启动容器以后,在这个容器中就运行了 Ubuntu 系统(包含 JDK)。前面已经下载了 Hadoop 安装文件 `hadoop-2.7.1.tar.gz`,现在需要把该安装文件放到本地 Ubuntu 系统的共享目录 `/home/hadoop/build` 下面;然后在 Docker 内部 Ubuntu 系统的 `/root/build` 目录即可获取到该 Hadoop 安装文件。实际上,在 Docker 内部的 Ubuntu 系统上面安装 Hadoop 的过程,和之前介绍的单机模式安装 Hadoop 是一样的。具体而言,需要首先在 Docker 内部的 Ubuntu 系统上面输入如下命令对安装文件进行解压缩操作:

```
$cd /root/build
$tar -zxvf hadoop-2.7.1.tar.gz -C /usr/local
```

如果是单机模式安装 Hadoop,到这里就已经顺利完成了安装,可以运行如下命令测试是否安装正确(在 Docker 内部的 Ubuntu 系统上面运行命令,不是在本机 Ubuntu 系统上面运行命令):

```
$cd /usr/local
$mv ./hadoop-2.7.1 ./hadoop
$cd hadoop
$./bin/hadoop version
```

输出的信息如下:

```
Hadoop 2.7.1
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git-r
15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled by jenkins on 2015-06-29T06:04Z
Compiled with protoc 2.5.0
From source with checksum fc0a1a23fc1868e4d5ee7fa2b28a58a
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-
common-2.7.1.jar
```

6. 配置 Hadoop 集群

现在介绍如何配置 Hadoop 集群,需要修改 `/usr/local/hadoop/etc/hadoop` 目录下的配

置文件 `hadoop_env.sh`、`core-site.xml`、`hdfs-site.xml`、`mapred-site.xml` 和 `yarn-site.xml`。首先使用 vim 编辑器打开 `hadoop_env.sh` 文件,命令如下(在 Docker 内部的 Ubuntu 系统中运行命令):

```
$cd /usr/local/hadoop
$vim etc/hadoop/hadoop-env.sh
```

打开 `hadoop_env.sh` 文件以后,在该文件的最前面增加如下内容:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
```

然后保存该文件,退出 vim 编辑器。类似地,使用 vim 编辑器打开 `core-site.xml` 文件,并在文件中输入以下内容:

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

使用 vim 编辑器打开 `hdfs-site.xml` 文件,并在文件中输入以下内容:

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/namenode_dir</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/datanode_dir</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
</configuration>
```

再使用 vim 编辑器打开 `mapred-site.xml` 文件(可以复制 `mapred-site.xml.template`,再修改文件名得到 `mapred-site.xml` 文件),并在文件中输入以下内容:

```
<configuration>
  <property>
```

```

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

```

最后修改 yarn-site.xml 文件,在该文件中输入以下内容:

```

<configuration>
  <!--Site specific YARN configuration properties-->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master</value>
  </property>
</configuration>

```

至此,Hadoop 集群配置顺利结束。需要首先保存这个镜像,需要在本地 Ubuntu 系统(不是 Docker 内部的 Ubuntu 系统)的终端中输入如下命令:

```
$docker commit a40b99f869ae ubuntu/hadoopinstalled
```

接下来在本地 Ubuntu 系统中(不是 Docker 内部的 Ubuntu 系统)打开 3 个终端窗口,每个终端上分别启动一个容器运行 ubuntu/hadoopinstalled 镜像,分别表示 Hadoop 集群中的 Master、Slave01 和 Slave02:

```

#在第一个终端中执行下面命令
$docker run -it -h master --name master ubuntu/hadoopinstalled
#在第二个终端中执行下面命令
$docker run -it -h slave01 --name slave01 ubuntu/hadoopinstalled
#在第三个终端中执行下面命令
$docker run -it -h slave02 --name slave02 ubuntu/hadoopinstalled

```

接着需要配置 Master、Slave01 和 Slave02 的地址信息,使得它们可以找到彼此。在 3 个终端中分别打开/etc/hosts 文件,可以查看本机的 IP 和主机名信息,最后得到 3 个 IP 和主机地址信息如下:

172.18.0.2	Master
172.18.0.3	Slave01
172.18.0.4	Slave02

最后把上述 3 个地址信息分别复制到 Master、Slave01 和 Slave02 的 /etc/hosts 文件中,也就是复制到各自 Docker 容器内部的 Ubuntu 系统的 /etc/hosts 文件中。然后,可以在作为 Master 节点的 Docker 容器内部的 Ubuntu 系统的终端中,使用如下命令来检测一下 Master 是否可以成功连上 Slave01 和 Slave02:

```
$ssh slave01
$ssh slave02
```

最后还需要打开 Master 节点上的 slaves 文件,输入两个 Slave 节点的主机名,请在 Master 节点中的终端中执行如下命令:

```
$cd /usr/local/hadoop
$vim etc/hadoop/slaves
```

上面命令执行后,就使用 vim 编辑器打开了 slaves 文件,需要修改该文件的配置,把文件中的 localhost 替换成两个 Slave 节点的主机名,分别放在两行上,最终,slaves 文件的内容只有如下两行:

```
Slave01
Slave02
```

至此,Hadoop 集群已经配置完成,现在可以启动集群,在 Master 节点的终端中执行如下命令:

```
$cd /usr/local/hadoop
$bin/hdfs namenode -format
$sbin/start-all.sh
```

这时 Hadoop 集群就已经启动,可以在 Master、Slave01 和 Slave02 这 3 个节点上分别运行命令 jps 查看运行结果,如图 3-22~图 3-24 所示。

```
root@master:/usr/local/hadoop# jps
789 Jps
199 NameNode
381 SecondaryNameNode
526 ResourceManager
```

图 3-22 在 Master 节点上运行 jps 命令的结果

```
root@slave01:/usr/local/hadoop# jps
70 DataNode
278 Jps
169 NodeManager
```

图 3-23 在 Slave01 节点上运行 jps 命令的结果

```
root@slave02:/usr/local/hadoop# jps
71 DataNode
170 NodeManager
270 Jps
```

图 3-24 在 Slave02 节点上运行 jps 命令的结果

7. 运行 Hadoop 程序实例

到目前为止,我们已经采用容器的方式成功启动 Hadoop 分布式集群,接下来,可以运行 Hadoop 自带的 grep 实例进行测试。因为要用到 HDFS,所以,需要先在 HDFS 上创建一个目录,在 Master 节点的终端中执行如下命令:

```
$cd /usr/local/hadoop
$./bin/hdfs dfs -mkdir -p /user/hadoop/input
```

然后将 /usr/local/hadoop/etc/hadoop/ 目录下的所有文件复制到 HDFS 中,需要在 Master 节点的终端中继续执行如下命令:

```
$./bin/hdfs dfs -put ./etc/hadoop/*.xml /user/hadoop/input
```

然后通过 ls 命令查看下是否正确将文件上传到 HDFS,需要在 Master 节点的终端中继续执行如下命令:

```
$./bin/hdfs dfs -ls /user/hadoop/input
```

输出结果如下:

```
Found 9 items
-rw-r--r--  3 root supergroup    4436 2016-12-26 07:40 /user/hadoop/input/
capacity-scheduler.xml
-rw-r--r--  3 root supergroup    1090 2016-12-26 07:40 /user/hadoop/input/
core-site.xml
-rw-r--r--  3 root supergroup    9683 2016-12-26 07:40 /user/hadoop/input/
hadoop-policy.xml
-rw-r--r--  3 root supergroup    1133 2016-12-26 07:40 /user/hadoop/input/
hdfs-site.xml
-rw-r--r--  3 root supergroup     620 2016-12-26 07:40 /user/hadoop/input/
httpfs-site.xml
-rw-r--r--  3 root supergroup    3518 2016-12-26 07:40 /user/hadoop/input/
kms-acls.xml
-rw-r--r--  3 root supergroup    5511 2016-12-26 07:40 /user/hadoop/input/
kms-site.xml
-rw-r--r--  3 root supergroup     866 2016-12-26 07:40 /user/hadoop/input/
mapred-site.xml
-rw-r--r--  3 root supergroup     947 2016-12-26 07:40 /user/hadoop/input/
yarn-site.xml
```

接下来在 Master 节点的终端中继续运行下面命令执行实例程序:

```
$/bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar grep  
/user/hadoop/input output 'dfs[a-z.]+'
```

等待这个程序运行结束之后,就可以在 HDFS 上的 output 目录下查看到运行结果,在 Master 节点的终端中执行如下命令:

```
$/bin/hdfs dfs -cat output/*
```

运行结果如下:

```
1 dfsadmin  
1 dfs.replication  
1 dfs.namenode.name.dir  
1 dfs.datanode.data.dir
```

如果得到类似上述的结果,则说明在 Hadoop 分布式集群中顺利执行了 grep 程序。

3.4 本章小结

Hadoop 是当前流行的分布式计算框架,在企业中得到了广泛的部署和应用。本章重点介绍如何安装 Hadoop,从而为后续章节开展 HDFS 和 MapReduce 编程实践奠定基础。

Hadoop 是基于 Java 开发的,需要运行在 JVM 中,因此,需要为 Hadoop 配置相应的 Java 环境。Hadoop 包含 3 种安装模式,即单机模式、伪分布式模式和分布式模式。本章分别介绍 3 种不同模式的安装配置方法。在初学阶段,建议采用伪分布式模式配置,这样可以快速构建起 Hadoop 实战环境,有效开展基础编程工作。

Docker 是目前比较热门的容器技术,相对于 VMware 和 VirtualBox 等虚拟化技术而言,Docker 也可以被视为一种轻量级虚拟化技术,可以在普通 PC 上快速构建起 Hadoop 集群,完成相关大数据实验,本章简要介绍了使用 Docker 搭建 Hadoop 分布式集群的方法。