

Hadoop 实训任务书

目录

实训任务一：

Hadoop 完全分布式环境安装部署.....	2
一、实训说明.....	2
二、实训环境.....	2
三、实训内容.....	3

实训任务二：

分布式文件系统 HDFS shell 命令.....	16
一、实训说明.....	17
二、实训环境.....	17
三、实训内容.....	17

实训任务三：

分布式离线计算框架 MapReduce.....	25
一、实训环境.....	25
二、实训内容.....	25

Hadoop 完全分布式环境安装部署

一、实训说明

本次实训，搭建的是完全分布式的 **hadoop** 环境，数据存储于由多个节点组成的 **hdfs** 分布式文件系统中。数据之间存在多个副本备份，当一个 **datanode** 宕机之后，另外的节点还存在数据副本，不影响数据的读取。本次实训目的，就是搭建分布式 **hadoop** 环境，实现数据的读取的高并发和高安全性。

二、实训环境

操作系统：Centos7.5_x86_64

Hadoop 版本号：3.1.0

JDK 版本号：jdk-8

集群中包括 3 个节点：1 个 **master**，2 个 **slave**，节点之间网络可以相互 **Ping** 通。节点的 IP 地址分布如下：（IP 地址可以自行设置）：

节点主机名	IP 地址
master	192.168.0.100
slave1	192.168.0.101
slave2	192.168.0.102

三、实训内容

1.网络配置（所有节点都要配置）

1）静态网络配置

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-ens192
```

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no 静态地址
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=ens192
UUID=f8c303f0-be3d-45a1-b963-770b61be4573
DEVICE=ens192
ONBOOT=yes 网卡开机自启动
IPADDR=192.168.0.100 ip地址
GATEWAY=192.168.0.1 网关地址
NETMASK=255.255.255.0 子网掩码
DNS1=192.168.0.1 DNS服务器地址
```

重启网卡，使配置为了将生效：

```
[root@localhost ~]# systemctl restart network
```

2）关闭 Selinux，将 SELINUX 修改为 disabled

```
[root@localhost ~]# vi /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three two values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3）关闭防火墙和防火墙自启，查看防火墙状态

```
[root@localhost ~]# systemctl stop firewalld
```

```
[root@localhost ~]# systemctl disable firewalld
```

```
[root@localhost ~]# systemctl status firewalld
```

```

[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@localhost ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)

Jul 25 10:01:24 localhost.localdomain systemd[1]: Starting firewa...
Jul 25 10:01:24 localhost.localdomain systemd[1]: Started firewal...
Jul 25 12:40:55 localhost.localdomain systemd[1]: Stopping firewa...
Jul 25 12:40:56 localhost.localdomain systemd[1]: Stopped firewal...
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost ~]#

```

4) 修改节点的主机名

```
[root@localhost ~]# hostnamectl set-hostname master
```

```

[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-ens192
[root@localhost ~]# hostnamectl set-hostname master
[root@localhost ~]#

```

修改完节点的主机名要重启才能生效:

```
[root@localhost ~]# reboot
```

5) 配置 host 文件

添加 3 个节点的 IP 和主机名的映射

```
[root@master ~]# vi /etc/hosts
```

```

127.0.0.1    localhost localhost.localdomain localhost4 localhost4.lo
caldomain4
::1         localhost localhost.localdomain localhost6 localhost6.lo
caldomain6
192.168.0.100 master
192.168.0.101 slave1
192.168.0.102 slave2
~
~

```

2.SSH 免密配置及验证（所有节点）

Hadoop 运行过程中需要管理远端 Hadoop 守护进程，在 Hadoop 启动以后，NameNode 是通过 SSH(Secure Shell)来启动和停止各个 DataNode 上的各种守护进程的。这就必须在节点之间执行指令的时候是不需要输入密码来执行的形式。我们需要配置 SSH 运用无密码公钥认证的形式，这样 NameNode 就可以使用 SSH 无密码登录并启动 DataNode 上的各种守护进程，同样原理 DataNode 上也能使用 SSH 无密码登录到 NameNode。

1）创建 hadoop 用户

```
[root@master ~]# useradd hadoop
[root@master ~]# passwd hadoop
```

```
[root@master ~]# useradd hadoop
[root@master ~]# passwd hadoop
Changing password for user hadoop.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@master ~]# █
```

2）对 hadoop 用户启用 sudo 命令

```
[root@master ~]# vim /etc/sudoers
```

添加内容：

```
hadoop  ALL=(ALL)        ALL
```

```
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
hadoop  ALL=(ALL)        ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PR
OCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)        ALL
```

3）安装和启动 SSH 协议

首先查看 ssh 服务是否已经安装：

```
[root@master ~]# rpm -qa |grep openssh
```

```
[root@master ~]# rpm -qa |grep openssh
openssh-clients-7.4p1-16.el7.x86_64
openssh-7.4p1-16.el7.x86_64
openssh-server-7.4p1-16.el7.x86_64
[root@master ~]# █
```

如果没有安装，我们可以通过以下命令来安装：

```
[root@master ~]# yum install openssh* -y
```

4) 切换到 hadoop 用户

```
su hadoop
```

```
[root@master ~]# su hadoop
[hadoop@master root]$
```

5) 每个节点生成密钥对

```
[hadoop@master root]$ ssh-keygen -t rsa
```

一直回车默认就可以了

```
[hadoop@master root]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:nIfew0+31DKlRxrtc96el5p+w250+u3oNS0pQRfX98 hadoop@master
The key's randomart image is:
+---[RSA 2048]---+
|
|      .    o. |
|    . o  o ..B|
|   S .  + +E|
|  . +  + **o|
|   . +..*0=+|
|    .+.o+OX|
|     .. *%@|
+-----[SHA256]-----+
[hadoop@master root]$
```

6) 将公钥发送到要免密登陆的节点上 (包括本机)

```
[hadoop@master root]$ ssh-copy-id -i master
[hadoop@master root]$ ssh-copy-id -i slave1
[hadoop@master root]$ ssh-copy-id -i slave2
```

要输入 yes, 然后输入免密节点的密码


```
[hadoop@master root]$ ssh-copy-id -i master
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home
/hadoop/.ssh/id_rsa.pub"
The authenticity of host 'master (192.168.0.100)' can't be establish
ed.
ECDSA key fingerprint is SHA256:f/w9KYhZDU+nRbvayp1mm5gR1mRE8gvTMqkf
9BIN+QI.
ECDSA key fingerprint is MD5:3a:e1:3a:f3:db:f1:ac:e9:a1:47:c3:30:bf:
06:22:85.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s)
, to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if yo
u are prompted now it is to install the new keys
hadoop@master's password:

Number of key(s) added: 1

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

[hadoop@master root]$ █
```

7) 验证是否成功

```
[hadoop@master root]$ ssh master
[hadoop@master root]$ ssh slave1
[hadoop@master root]$ ssh slave2
```

```
[hadoop@master root]$ ssh slave1
Last failed login: Thu Jul 25 13:08:21 EDT 2019 from slave1 on ssh:n
otty
There was 1 failed login attempt since the last successful login.
Last login: Thu Jul 25 13:07:35 2019
[hadoop@slave1 ~]$ exit
logout
Connection to slave1 closed.
[hadoop@master root]$ ssh slave2
Last login: Thu Jul 25 13:08:31 2019
[hadoop@slave2 ~]$ exit
logout
Connection to slave2 closed.
[hadoop@master root]$ ssh master
Last login: Thu Jul 25 13:01:07 2019
[hadoop@master ~]$ exit
logout
Connection to master closed.
[hadoop@master root]$ █
```

3. Java 环境安装（所有节点都要配置）

1）新建 java 目录

```
[hadoop@master root]$ sudo mkdir /usr/java
```

2）将 java 的安装包解压到 java 目录

```
[hadoop@master root]$ sudo tar -zxvf /home/package/jdk-8.tar.gz -C /usr/java
```

3）配置环境变量

```
[hadoop@master root]$ sudo vim /etc/profile
```

在文件后面添加环境变量：

```
export JAVA_HOME=/usr/java/jdk1.8.0_201
export JRE_HOME=/usr/java/jdk1.8.0_201/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

```
unset i
unset -f pathmunge

export JAVA_HOME=/usr/java/jdk1.8.0_201
export JRE_HOME=/usr/java/jdk1.8.0_201/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

4）使环境变量生效

```
[hadoop@master root]$ source /etc/profile
```

5）验证安装

```
[hadoop@master ~]$ java -version
[hadoop@master ~]$ javac -version
```

```
[hadoop@master ~]$ java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
[hadoop@master ~]$ javac -version
javac 1.8.0_201
[hadoop@master ~]$
```


4. 在 Master 节点上安装 Hadoop

1) 将 hadoop 安装包解压到/usr 目录下

```
[hadoop@master ~]$ sudo tar -zxvf /home/package/hadoop-3.1.2.tar.gz -C /usr
```

2) 重命名安装路径

```
[hadoop@master ~]$ sudo mv /usr/hadoop-3.1.2/ /usr/hadoop
```

3) 配置 Hadoop 的环境变量

```
[hadoop@master ~]$ sudo vim /etc/profile
```

在文件最后面加上 hadoop 环境变量:

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

4) 使配置的环境变量生效

```
[hadoop@master ~]$ source /etc/profile
```

5) 配置 hadoop-env.sh 配置文件

```
[hadoop@master ~]$ sudo vim $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

修改配置文件里面的 JAVA_HOME:

```
# Many of the options here are built from the perspective that users
# may want to provide OVERWRITING values on the command line.
# For example:
#
JAVA_HOME=/usr/java/jdk1.8.0_201
#
# Therefore, the vast majority (BUT NOT ALL!) of these defaults
# are configured for substitution and not append. If append
# is desired, the configuration file should be modified.
```

6) 配置 core-site.xml 配置文件

```
[hadoop@master ~]$ sudo vim $HADOOP_HOME/etc/hadoop/core-site.xml
```

在配置文件中添加: hdfs 的数据访问地址和设置临时数据存放目录

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoopData/tmp/</value>
  </property>
</configuration>
```

7) 配置 hdfs-site.xml 配置文件

```
[hadoop@master ~]$sudo vim $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

在配置文件添加：文件副本数量、hdfs 的 namenode 数据存储目录、hdfs 的 datanode 数据存储目录、hdfs 的 web 访问地址

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/hadoopData/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/hadoopData/dfs/data</value>
  </property>
  <property>
    <name>dfs.namenode.http-address</name>
    <value>0.0.0.0:50070</value>
  </property>
</configuration>
```

8) 配置 yarn-site.xml 配置文件

```
[hadoop@master ~]$sudo vim $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

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

9) 配置 mapred-site.xml 配置文件

```
[hadoop@master ~]$sudo vim $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

在配置文件中添加以下配置：

```
<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>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/usr/hadoop</value>
  </property>
</configuration>
```

1 0) 配置 hadoop-config.sh 配置文件

```
[hadoop@master hadoop]$ vim $HADOOP_HOME/libexec/hadoop-config.sh
```

在配置文件中加入 JAVA 环境变量

```
export JAVA_HOME=/usr/java/jdk1.8.0_201
export JRE_HOME=/usr/java/jdk1.8.0_201/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

```
export JAVA_HOME=/usr/java/jdk1.8.0_201
export JRE_HOME=/usr/java/jdk1.8.0_201/jre
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JRE_HOME/lib
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
-- INSERT --
```

163,11

Bot

1 1) 配置 workers 文件

```
[hadoop@master ~]$ sudo vim $HADOOP_HOME/etc/hadoop/workers
```

在配置文件里面添加 datanode 数据节点的主机名

```
slave1
slave2
~
```

1 2) 将 master 节点上的安装文件同步到 slave1、slave2 节点上

```
[hadoop@master ~]$ sudo scp -r /usr/hadoop/ slave1:/usr/  
[hadoop@master ~]$ sudo scp -r /usr/hadoop/ slave2:/usr/
```

1 3) 新建数据文件目录（所有节点）

```
[hadoop@master ~]$ sudo mkdir -p /home/hadoopData/tmp  
[hadoop@master ~]$ sudo mkdir -p /home/hadoopData/dfs/name  
[hadoop@master ~]$ sudo mkdir -p /home/hadoopData/dfs/data
```

1 4) 修改 hadoop 文件属主权限（所有节点）

```
[hadoop@master ~]$ sudo chown -R hadoop:hadoop /usr/hadoop  
[hadoop@master ~]$ sudo chown -R hadoop:hadoop /home/hadoopData/
```

```
[hadoop@master ~]$ sudo chown -R hadoop:hadoop /usr/hadoop  
[hadoop@master ~]$ sudo chown -R hadoop:hadoop /home/hadoopData/  
[hadoop@master ~]$
```


5. 初始化 Hadoop

1) 格式化 namenode (master 节点)

```
[hadoop@master ~]$ hdfs namenode -format
```

```
2019-07-25 14:22:35,140 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
2019-07-25 14:22:35,140 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
2019-07-25 14:22:35,142 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2019-07-25 14:22:35,142 INFO util.GSet: VM type = 64-bit
2019-07-25 14:22:35,142 INFO util.GSet: 0.029999999329447746% max memory 1.7 GB = 534.2 KB
2019-07-25 14:22:35,142 INFO util.GSet: capacity = 2^16 = 65536 entries
2019-07-25 14:22:35,167 INFO namenode.FSImage: Allocated new BlockPoolId: BP-553268938-192.168.0.100-1564078955161
2019-07-25 14:22:35,183 INFO common.Storage: Storage directory /home/hadoopData/dfs/name has been successfully formatted.
2019-07-25 14:22:35,191 INFO namenode.FSImageFormatProtobuf: Saving image file /home/hadoopData/dfs/name/current/fsimage.ckpt_000000000000000000 using no compression
2019-07-25 14:22:35,285 INFO namenode.FSImageFormatProtobuf: Image file /home/hadoopData/dfs/name/current/fsimage.ckpt_000000000000000000 of size 393 bytes saved in 0 seconds .
2019-07-25 14:22:35,291 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2019-07-25 14:22:35,296 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at master/192.168.0.100
*****/
```

2) 启动 hadoop (master 节点)

```
[hadoop@master ~]$ start-all.sh
```

```
[hadoop@master ~]$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [master]
Starting datanodes
slave1: WARNING: /usr/hadoop/logs does not exist. Creating.
slave2: WARNING: /usr/hadoop/logs does not exist. Creating.
Starting secondary namenodes [master]
Starting resourcemanager
Starting nodemanagers
[hadoop@master ~]$ jps
3873 Jps
3106 NameNode
3334 SecondaryNameNode
3566 ResourceManager
```


3) 查看 hadoop 的守护进程

master 节点:

```
[hadoop@master ~]$ jps
3873 Jps
3106 NameNode
3334 SecondaryNameNode
3566 ResourceManager
[hadoop@master ~]$
```

slave1 节点:

```
[hadoop@slave1 jdk1.8.0_201]$ jps
2402 NodeManager
2505 Jps
2284 DataNode
[hadoop@slave1 jdk1.8.0_201]$
```

slave2 节点:

```
[hadoop@slave1 jdk1.8.0_201]$ jps
2402 NodeManager
2505 Jps
2284 DataNode
[hadoop@slave1 jdk1.8.0_201]$
```

6. 验证测试

1) 在浏览器中打开 <http://192.168.0.100:50070>，查看 NameNode 节点的状态

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Overview 'master:9000' (active)

Started:

Fri Jul 26 02:23:17 +0800 2019

Version:

3.1.2, r1019dde65bcf12e05ef48ac71e84550d589e5d9a

Compiled:

Tue Jan 29 09:39:00 +0800 2019 by sunilg from branch-3.1.2

Cluster ID:

CID-d236edcb-6667-45b0-92db-3d75425da04f

Block Pool ID:

BP-553268938-192.168.0.100-1564078955161

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 76.2 MB of 241.5 MB Heap Memory. Max Heap Memory is 1.7 GB.

Non Heap Memory used 47.36 MB of 48.46 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:

282.1 GB

Configured Remote Capacity:

0 B

DFS Used:

8 KB (0%)

Non DFS Used:

2.81 GB

DFS Remaining:

279.29 GB (99%)

Block Pool Used:

8 KB (0%)

DataNodes usages% (Min/Median/Max/stdDev):

0.00% / 0.00% / 0.00% / 0.00%

2) 浏览 DataNode 数据节点

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Datanode Information

✔ In service

❗ Down

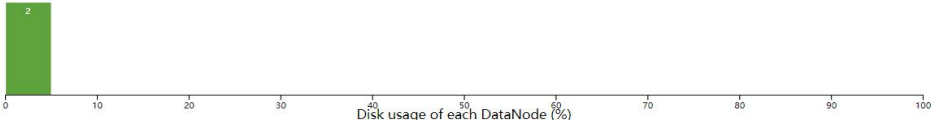
🗑 Decommissioned

🔗 Decommissioned & dead

🔧 In Maintenance

🔪 In Maintenance & dead

Datanode usage histogram



In operation

Show 25 entries

Search:

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✔ slave1:9866 (192.168.0.101:9866)	http://slave1:9864	0s	2m	141.05 GB	0	4 KB (0%)	3.1.2
✔ slave2:9866 (192.168.0.102:9866)	http://slave2:9864	0s	8m	141.05 GB	0	4 KB (0%)	3.1.2


Showing 1 to 2 of 2 entries

Previous

1

Next

3) 在浏览器打开 <http://192.168.0.100:8088>，查看所有的应用



All Applications

Logged in as: dr

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	16 GB	0 B	0	16	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
2	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=M), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0


Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklist Nodes
No data available in table																			

Showing 0 to 0 of 0 entries

First Previous Next Last

4) 浏览 Nodes



Nodes of the cluster

Logged in as: dr:who

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
1	0	0	1	0	0 B	16 GB	0 B	0	16	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
2	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=M), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack		RUNNING	slave2:46686	slave2:8042	Tue Jul 30 22:57:29 -0400 2019		0		0 B	8 GB	0	8	3.1.2
/default-rack		RUNNING	slave1:44034	slave1:8042	Tue Jul 30 22:57:28 -0400 2019		0		0 B	8 GB	0	8	3.1.2

Showing 1 to 2 of 2 entries

First Previous 1 Next Last

5) 关闭 Hadoop

```
[hadoop@master ~]$ stop-all.sh
```

```
[hadoop@master ~]$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [master]
Stopping datanodes
Stopping secondary namenodes [master]
Stopping nodemanagers
Stopping resourcemanager
[hadoop@master ~]$
```

分布式文件系统 HDFS shell 命令

一、实训说明

本次实训是针对 hdfs shell 命令的使用。Hadoop fs 与 hdfs dfs 的命令的使用是相似的，本次实训使用的是 hdfs dfs 命令，所有命令的操作都是在 Hadoop 用户下进行。

二、实训环境

已经完成配置的 Hadoop 伪分布式或完全分布式环境。

三、实训内容

1. ls：查看指定目录文件和目录

使用方法：hdfs dfs -ls [-d][-h][-R] <paths>

操作选项	功能说明
-d	返回文件 paths
-h	按照 KMG 数据大小单位显示文件大小，如果没有单位，默认认为 B
-R	级联显示 paths 下文件，这里 paths 是个多级目录

示例 1：查看 hdfs 目录和文件的大小

```
[hadoop@master ~]$ hdfs dfs -ls -h /  
  
[hadoop@master ~]$ hdfs dfs -ls -h /  
Found 1 items  
drwxr-xr-x - hadoop supergroup 0 2019-07-26 07:59 /hdfs  
[hadoop@master ~]$
```

示例 2：列出 hdfs 目录级子目录下面的文件

```
[hadoop@master ~]$ hdfs dfs -ls -R /hdfs
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs/dir
1
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs/dir
1/dir11
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs/dir
1/dir12
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs/dir
2
[hadoop@master ~]$
```

2. mkdir：新建目录

使用方法：hdfs dfs -mkdir [-p] <paths>

当创建目录的父目录不存在时，使用-p 参数可以连同父目录一起创建

示例 1：在/目录下创建一个 shixun 文件夹

```
[hadoop@master ~]$ hdfs dfs -mkdir /shixun
```

示例 2：创建目录/shixun/mapreduce/data

```
[hadoop@master ~]$ hdfs dfs -mkdir -p /shixun/mapreduce/data
```

```
[hadoop@master ~]$ hdfs dfs -mkdir /shixun
[hadoop@master ~]$ hdfs dfs -mkdir -p /shixun/mapreduce/data
[hadoop@master ~]$
```

3. touch：新建文件

使用方法：hdfs dfs -touch <paths>

示例 1：在/shixun/mapreduce/data 目录下新建一个空文件 word.txt

```
[hadoop@master ~]$ hdfs dfs -touch /shixun/mapreduce/data/word.txt
```

```
[hadoop@master ~]$ hdfs dfs -touch /shixun/mapreduce/data/word.txt
[hadoop@master ~]$ hdfs dfs -ls /shixun/mapreduce/data/
Found 1 items
-rw-r--r--   2 hadoop supergroup          0 2019-07-26 10:10 /shixun/m
apreduce/data/word.txt
[hadoop@master ~]$
```

4. cp：复制文件

使用方法：hdfs fs -cp SRC [SRC ...] DST

文件从 SRC 复制到 DST，当指定 SRC 为多个文件时，DST 必须为目录

示例 1：将/shixun 目录复制为/shixun2

```
[hadoop@master ~]$ hdfs dfs -cp /shixun /shixun2
```

示例 2：将/hdfs 目录和/shixun 目录复制到/shixun2 目录

```
[hadoop@master ~]$ hdfs dfs -cp /hdfs /shixun /shixun2
```



```
[hadoop@master ~]$ hdfs dfs -cp /shixun /shixun2
[hadoop@master ~]$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 10:04 /shixun
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 10:22 /shixun2
[hadoop@master ~]$ hdfs dfs -cp /hdfs /shixun /shixun2
[hadoop@master ~]$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 10:04 /shixun
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 10:23 /shixun2
[hadoop@master ~]$
```

5. rm：删除文件和目录

使用方法：hdfs dfs -rm [-f] [-r|-R] [-skip Trash] <paths>

参数选下个	参数说明
-f	如果要删除的文件不存在，不显示提示和错误信息
-r -R	-r R 级联删除目录下的所有文件和子目录文件
-skipTrash	-skipTrash 直接删除，不进入垃圾回收站

示例 1：删除/shixun2/mapreduce/data/目录下面的所有文件

```
[hadoop@master ~]$ hdfs dfs -rm /shixun2/mapreduce/data/word.txt
```

删除前：

```
[hadoop@master ~]$ hdfs dfs -ls /shixun2/mapreduce/data
Found 1 items
-rw-r--r--   2 hadoop supergroup          0 2019-07-26 10:32 /shixun2/
mapreduce/data/word.txt
[hadoop@master ~]$
```

删除后：

```
[hadoop@master ~]$ hdfs dfs -rm /shixun2/mapreduce/data/*
Deleted /shixun2/mapreduce/data/word.txt
[hadoop@master ~]$ hdfs dfs -ls /shixun2/mapreduce/data
[hadoop@master ~]$
```

示例 2：删除/shixun2 目录

```
[hadoop@master ~]$ hdfs dfs -rm -r /shixun2
```

```
[hadoop@master ~]$ hdfs dfs -rm -r /shixun2
Deleted /shixun2
[hadoop@master ~]$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 09:57 /hdfs
drwxr-xr-x   - hadoop supergroup          0 2019-07-26 10:04 /shixun
[hadoop@master ~]$
```

6. put/get：上传/下载文件

使用方法：hdfs dfs -put [-f] [-p] <localsrc> ... <dst>

hdfs dfs -get [-p] <src> ... <localdst>

put 把本地文件系统的文件上传到 HDFS 分布式文件系统

get 把 HDFS 分布式文件系统上的文件下载到本地文件系统

参数选项	参数说明
-f	如果文件在分布式文件系统上已经存在，则覆盖存储
-p	保持源文件的 属性（组、拥有者、创建时间、权限等）

示例 1：将本地文件上传到分布式文件系统

```
[hadoop@master ~]$ hdfs dfs -ls /home/package/movies.csv /shixun/mapreduce/data/
```

```
[hadoop@master ~]$ hdfs dfs -put /home/package/movies.csv /shixun/mapreduce/data/
[hadoop@master ~]$ hdfs dfs -ls /shixun/mapreduce/data/
Found 2 items
-rw-r--r--  2 hadoop supergroup    1397542 2019-07-26 10:48 /shixun/mapreduce/data/movies.csv
-rw-r--r--  2 hadoop supergroup      0 2019-07-26 10:10 /shixun/mapreduce/data/word.txt
[hadoop@master ~]$
```

实例 2：将分布式文件系统的文件下载到本地

```
hdfs dfs -get /shixun/mapreduce/data/word.txt /home/hadoop/
```

```
[hadoop@master ~]$ hdfs dfs -get /shixun/mapreduce/data/word.txt /home/hadoop/
[hadoop@master ~]$ ll /home/hadoop
total 0
-rw-r--r--  1 hadoop hadoop 0 Jul 26 10:58 word.txt
[hadoop@master ~]$
```

实例 3：将文件上传到分布式文件系统，同时保存文件属组等信息

```
[hadoop@master ~]$ hdfs dfs -put -p /home/package/movies.csv /shixun/mapreduce/data/movies2.csv
```

```
[hadoop@master ~]$ hdfs dfs -put -p /home/package/movies.csv /shixun/mapreduce/data/movies2.csv
[hadoop@master ~]$ hdfs dfs -ls /shixun/mapreduce/data/
Found 3 items
-rw-r--r--  2 hadoop supergroup    1397542 2019-07-26 10:48 /shixun/mapreduce/data/movies.csv
-rw-r--r--  2 root root            1397542 2019-07-26 10:45 /shixun/mapreduce/data/movies2.csv
-rw-r--r--  2 hadoop supergroup      0 2019-07-26 10:10 /shixun/mapreduce/data/word.txt
[hadoop@master ~]$
```

7. cat、text、tail ： 查看文件内容

使用方法：hdfs dfs -cat/text [-ignoreCrc] <src>

Hdfs dfs -tail [-f] <file>

其中：-ignoreCrc 忽略循环检验失败的文件；-f 动态更新显示数据，如查看某个不断增长的文件的日志文件。

3 个命令都是在命令行窗口查看指定文件内容。区别是 text 不仅可以查看文本文件，还可以查看压缩文件和 Avro 序列化的文件，其他两个不可以；tail 查看的是最后 1KB 的文件（Linux 上的 tail 默认查看最后 10 行记录）。

示例 1: 查看文件/shixun/mapreduce/data/movies.csv 内容

```
[hadoop@master ~]$ hdfs dfs -tail /shixun/mapreduce/data/movies.csv
```

```
[hadoop@master ~]$ hdfs dfs -tail /shixun/mapreduce/data/movies.csv
uñado de besos (2014),Drama|Romance
131164,Vietnam in HD (2011),War
131166,WVII IN HD (2009),(no genres listed)
131168,Phoenix (2014),Drama
131170,Parallels (2015),Sci-Fi
131172,Closed Curtain (2013),(no genres listed)
131174,Gentlemen (2014),Drama|Romance|Thriller
131176,A Second Chance (2014),Drama
131180,Dead Rising: Watchtower (2015),Action|Horror|Thriller
131231,Standby (2014),Comedy|Romance
131237,What Men Talk About (2010),Comedy
131239,Three Quarter Moon (2011),Comedy|Drama
131241,Ants in the Pants (2000),Comedy|Romance
131243,Werner - Gekotzt wird später (2003),Animation|Comedy
131248,Brother Bear 2 (2006),Adventure|Animation|Children|Comedy|Fantasy
131250,No More School (2000),Comedy
131252,Forklift Driver Klaus: The First Day on the Job (2001),Comedy|Horror
131254,Kein Bund für's Leben (2007),Comedy
131256,"Feuer, Eis & Dosenbier (2002)",Comedy
131258,The Pirates (2014),Adventure
131260,Rentun Ruusu (2001),(no genres listed)
131262,Innocence (2014),Adventure|Fantasy|Horror
[hadoop@master ~]$
```

8. appendToFile : 追加文件

使用方法: `hdfs dfs -appendToFile <localrc> ... <dst>`

示例 1.把本地文件系统文件追加到分布式文件系统中

```
[hadoop@master ~]$ cat /home/hadoop/file1.txt
```

```
[hadoop@master ~]$ cat /home/hadoop/file1.txt
Hello HDFS~
[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/mapreduce/data/word.txt
```

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/mapreduce/data/word.txt
[hadoop@master ~]$
```

```
[hadoop@master ~]$ hdfs dfs -appendToFile /home/hadoop/file1.txt /shixun/mapreduce/data/word.txt
```

```
[hadoop@master ~]$ hdfs dfs -appendToFile /home/hadoop/file1.txt /shixun/mapreduce/data/word.txt
[hadoop@master ~]$ hdfs dfs -cat /shixun/mapreduce/data/word.txt
Hello HDFS~
[hadoop@master ~]$
```


9. du : 显示占用磁盘大小

使用方法: `hdfs dfs -du [-s] [-h] <path> ...`

默认按字节显示指定目录所占空间大小。其中,

-s 显示指定目录下文件总大小;

-h 按照 KMG 数据大小单位显示文件大小, 如果没有单位, 默认为 B

示例 1: 显示分布式主目录下文件和目录大小

```
[hadoop@master ~]$ hdfs dfs -du /shixun
```

```
[hadoop@master ~]$ hdfs dfs -du /shixun
2795096 5590192 /shixun/mapreduce
[hadoop@master ~]$
```

示例 2: 显示指定目录下文件所占空间大小

```
[hadoop@master ~]$ hdfs dfs -du -h /shixun/mapreduce/data/
```

```
[hadoop@master ~]$ hdfs dfs -du -h /shixun/mapreduce/data/
1.3 M 2.7 M /shixun/mapreduce/data/movies.csv
1.3 M 2.7 M /shixun/mapreduce/data/movies2.csv
12    24    /shixun/mapreduce/data/word.txt
[hadoop@master ~]$
```

示例 3: 显示指定文件目录总数据量大小

```
[hadoop@master ~]$ hdfs dfs -du -s /shixun/mapreduce/data
```

```
[hadoop@master ~]$ hdfs dfs -du -s /shixun/mapreduce/data/
2795096 5590192 /shixun/mapreduce/data
```

10. chmod : 修改文件权限

使用方法: `hdfs fs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]`

改变文件的权限。使用-R 将递归改变目录结构下的所有文件

示例 1: 改变/hdfs/dir1 的权限

```
[hadoop@master ~]$ hdfs dfs -chmod 777 /hdfs/dir1
```

```
[hadoop@master ~]$ hdfs dfs -ls /hdfs
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2019-07-26 09:57 /hdfs/dir1
drwxr-xr-x - hadoop supergroup 0 2019-07-26 09:57 /hdfs/dir2
[hadoop@master ~]$ hdfs dfs -chmod 777 /hdfs/dir1
[hadoop@master ~]$ hdfs dfs -ls /hdfs
Found 2 items
drwxrwxrwx - hadoop supergroup 0 2019-07-26 09:57 /hdfs/dir1
drwxr-xr-x - hadoop supergroup 0 2019-07-26 09:57 /hdfs/dir2
[hadoop@master ~]$
```

11. chown : 修改文件属主和属组

使用方法: `hdfs dfs -chown [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]`

改变文件的属主和属组。使用-R 将递归改变目录结构下的所有文件

命令的使用者必须是文件的所 有者或者超级用户

示例 1：改变目录/hdfs/dir1 的属组

```
[hadoop@master ~]$ hdfs dfs -chown hadoop:dfs /hdfs/dir1

[hadoop@master ~]$ hdfs dfs -ls /hdfs
Found 2 items
drwxrwxrwx - hadoop supergroup      0 2019-07-26 09:57 /hdfs/dir1
drwxr-xr-x - hadoop supergroup      0 2019-07-26 09:57 /hdfs/dir2
[hadoop@master ~]$
[hadoop@master ~]$
[hadoop@master ~]$ hdfs dfs -chown hadoop:dfs /hdfs/dir1
[hadoop@master ~]$ hdfs dfs -ls /hdfs
Found 2 items
drwxrwxrwx - hadoop dfs              0 2019-07-26 09:57 /hdfs/dir1
drwxr-xr-x - hadoop supergroup      0 2019-07-26 09:57 /hdfs/dir2
[hadoop@master ~]$
```

12. dfsadmin：显示 HDFS 的运行状态和管理 HDFS

dfsadmin 是一个多任务客服端工具，用来显示 HDFS 运行状态和管理 HDFS，文件的命令

命令参数	参数说明
-report	查看文件系统的基本信息和统计信息
-safeadmin enter leave get wait	安全模式命令。
-saveNamespace	可以强制创建检查点，仅仅在安全模式下面运行
-refreshNodes	重新读取 hosts 和 exclude 文件，使新的节点或需要退出集群的节点能够被 NameNode 重新识别。这个命令在新增节点或注销节点时用到。
-finalizeUpgrade	终结 HDFS 的升级操作。DataNode 删除前一个版本的工作目录，之后 NameNode 也这样做。
-upgradeProgress status details force	请求当前系统的升级状态 升级状态的细节 强制升级操作
-metasave filename	保存 NameNode 的主要数据结构到 Hadoop.log.dir 属性指定的目录下的<filename>文件中。
-setQuota <quota><dirname>.....<dirname>:	为每个目录 <dirname> 设定配额 57<quota>。目录配额是一个长整形整数，强制设定目录树下的名字个数。
-clrQuota <dirname>.....<dirname>	为每个目录<dirname>清除配额设定
-fetchImage <local directory>	把最新的文件系统镜像文件从元数据节点上下载到本地指定目录
-clrQuota <dirname> <dirname>	清除每个目录 dirname 的配额。以下情况会报错： (1) 这个目录不存在或者是一个文件 (2) 用户不是管理员
-restoreFailedStorage true false check	此选项将关闭自动尝试恢复故障的存储副本。如果故障的存储可用，再次尝试还原检查点期间的日志编

	辑文件或文件系统镜像文件。 “check”选项将返回当前设置
--	-----------------------------------

示例 1：显示文件系统的基本信息和统计信息

```
[hadoop@master ~]$ hdfs dfsadmin -report
```

```
[hadoop@master ~]$ hdfs dfsadmin -report
Configured Capacity: 302898880512 (282.10 GB)
Present Capacity: 299881127936 (279.29 GB)
DFS Remaining: 299875434496 (279.28 GB)
DFS Used: 5693440 (5.43 MB)
DFS Used%: 0.00%
Replicated Blocks:
    Under replicated blocks: 0
    Blocks with corrupt replicas: 0
    Missing blocks: 0
    Missing blocks (with replication factor 1): 0
    Low redundancy blocks with highest priority to recover: 0
    Pending deletion blocks: 0
Erasure Coded Block Groups:
    Low redundancy block groups: 0
    Block groups with corrupt internal blocks: 0
    Missing block groups: 0
    Low redundancy blocks with highest priority to recover: 0
    Pending deletion blocks: 0
```

示例 2：获取安全模式状态

```
[hadoop@master ~]$ hdfs dfsadmin -safemode get
```

```
[hadoop@master ~]$ hdfs dfsadmin -safemode get
Safe mode is OFF
[hadoop@master ~]$
```

开启安全模式

```
[hadoop@master ~]$ hdfs dfsadmin -safemode enter
```

关闭安全模式

```
[hadoop@master ~]$ hdfs dfsadmin -safemode leave
```

示例 3：刷新 datanode 数据节点

```
[hadoop@master ~]$ hdfs dfsadmin -refreshNodes
```

```
[hadoop@master ~]$ hdfs dfsadmin -refreshNodes
Refresh nodes successful
[hadoop@master ~]$
```

分布式离线计算框架 MapReduce

一、实训环境

已经配置完成的 Hadoop 完全分布式环境

二、实训内容

1. wordcount 程序

wordcount 顾名思义就是单词计数，用于统计文本单词出现的次数，它是 Hadoop 的入门程序

文本输入如下：

wordcount.txt

```
The Younger Generation In the Twenty-first Century

Good evening, ladies and gentlemen.

We always say "we are the future". Indeed. We, the younger generation represents modern knowledge, new concepts, ambition and great desire for success. But, have you ever thought of this question: how can we be successful in the 21st century, which is full of great challenges and fierce competitions? In my opinion, there are two important factors.

First, we have to catch the opportunities and face the challenges. China has successfully entered the World Trade Organization, Beijing has won the bid for the 2008 Olympic Games. If we want to be successful, we have to seize these opportunities. SARS and AIDS, environmental pollution, growing population We have no choice but to solve these problems with our own hands. Thus, catching the opportunities and trying our best to cope with the difficulties is necessary for us to succeed in the 21st century.

Secondly, we have to learn to cooperate and compete. Last year, when SARS was spreading rapidly in China, scientists all over the world cooperated with each other and prevented the disease from spreading in a short period of time. At the same time, they were also competing against each other to see who could conquer the disease first. By cooperating and competing, the world develops day by day. Therefore, we, the younger generation, must know how to cooperate and compete.

Ladies and gentlemen, we are standing at the outset of a new century, we are at the gate of a new era. Looking back, the 20th century is a century of great developments and remarkable changes. Looking forward, the 21st century is a century full of opportunities, challenges, cooperation and competition. I believe, if we can seize the opportunities, face the challenges, cooperate and compete, success in the 21st century belongs to him, to you and to me.
```

2. 在集群中运行

(1) 将文本上传到 HDFS 分布式文件系统

```
[hadoop@master ~]$ hdfs dfs -put /home/hadoop/wordcount.txt /shixun/word_in/
```

(2) 运行 WordCount

格式: Hadoop jar [jar 文件位置] [jar 主类] [HDFS 输入位置] [HDFS 输出位置]

注意: 程序运行结果的输出位置一定时一个不存在的目录, 否则会报错

```
[hadoop@master ~]$ hadoop jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar
wordcount /shixun/word_in/wordcount.txt /shixun/word_out
```

```
[hadoop@master ~]$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar wordcount /shixun/word_in/wordcount.txt /shixun/word_out
2019-07-26 12:39:55,943 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-07-26 12:39:55,996 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-07-26 12:39:55,996 INFO impl.MetricsSystemImpl: JobTracker metrics initialized
```

```
Map-Reduce Framework
  Map input records=24
  Map output records=309
  Map output bytes=3113
  Map output materialized bytes=2431
  Input split bytes=112
  Combine input records=309
  Combine output records=185
  Reduce input groups=185
  Reduce shuffle bytes=2431
  Reduce input records=185
  Reduce output records=185
  Spilled Records=370
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=0
  Total committed heap usage (bytes)=538968064
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=1897
File Output Format Counters
  Bytes Written=1688
```

(3) 查看结果:

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/word_out/part-r-00000
```

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/word_out/part-r-00000
2008      1
20th      1
21st      4
85        1
AIDS,     1
At        1
Beijing   1
But,      1
By        1
Century   1
China     1
China,    1
First,    1
Games.If      1
Generation    1
Good         1
I           1
In          2
Ladies      1
Last        1
Looking     2
Olympic     1
Organization, 1
```

3. Java API 实现 wordcount

WordCountMap 类:

```
package mr;

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

public class WordCountMap extends Mapper<LongWritable, Text, Text, IntWritable>
{
    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String line = value.toString();
        String[] words = line.split(" ");
        for (String word: words) {
            context.write(new Text(word), new IntWritable(1));
        }
    }
}
```

WordCountReduce 类:

```
package mr;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import javax.xml.soap.Text;

public class WordCountReduce extends Reducer<Text, IntWritable, Text,
IntWritable> {
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
        int count = 0;
        for (IntWritable value: values) {
            count += value.get();
        }
        context.write(key, new IntWritable(count));
    }
}
```


WordCountMain 类:

```
package mr;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;

import java.io.IOException;

public class WordCountMain {
    public static void main(String[] args) throws IOException,
        ClassNotFoundException, InterruptedException {
        Configuration configuration=new Configuration();
        if(args.length !=2){
            System.err.println("Usage:wordcount <input> <output>");
            System.exit(2);
        }
        Job job=new Job(configuration,"word count");

        job.setJarByClass(WordCountMain.class);
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job,new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));

        Path outputPath=new Path(args[1].toString());
        outputPath.getFileSystem(configuration).delete(outputPath,true);

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

将程序打包放到集群中运行：

```
hadoop jar /home/hadoop/wordcount.jar mr.WordCountMain  
/shixun/word_in/wordcount.txt /shixun/word_out2
```

```
[hadoop@master ~]$ hadoop jar /home/hadoop/wordcount.jar mr.WordCountMain /shixun/word_in/wordcount.txt /shixun/word_out2  
2019-07-26 13:30:44,336 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties  
2019-07-26 13:30:44,420 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).  
2019-07-26 13:30:44,420 INFO impl.MetricsSystemImpl: JobTracker metrics system started  
2019-07-26 13:30:44,619 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.  
2019-07-26 13:30:44,680 INFO input.FileInputFormat: Total input file
```

Map-Reduce Framework

```
Map input records=24  
Map output records=309  
Map output bytes=3113  
Map output materialized bytes=3737  
Input split bytes=112  
Combine input records=0  
Combine output records=0  
Reduce input groups=185  
Reduce shuffle bytes=3737  
Reduce input records=309  
Reduce output records=185  
Spilled Records=618  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=0  
Total committed heap usage (bytes)=537919488
```

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=1897
```

File Output Format Counters

```
Bytes Written=1688
```

查看结果:

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/word_out2/part-r-00000
```

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/word_out2/part-r-00000
2008      1
20th      1
21st      4
85        1
AIDS,     1
At        1
Beijing   1
But,      1
By        1
Century   1
China     1
China,    1
First,    1
Games.If          1
Generation        1
Good             1
I               1
In              2
Ladies         1
Last           1
Looking        2
Olympic        1
Organization,   1
SARS           2
Secondly,       1
The            1
Therefore,      1
Thus,          1
Trade          1
Twenty-first    1
We             2
We,            1
World          1
Younger        1
```

4. Python 实现 wordcount

这里使用 Hadoop streaming API 通过 STDIN 和 STDOUT 在 Map 和 Reduce 间传递数据

mapper.py 文件

```
#!/usr/bin/python
import sys
for line in sys.stdin:
    line=line.strip()
    words= line.split(' ')
    for word in words:
        print(word)
```

reducer.py 文件

```
#!/usr/bin/python
from collections import Counter
import sys
data=[line.strip() for line in sys.stdin]
result = Counter(data).items()
for rs in result:
    print(rs[0],rs[1])
```

赋予两个文件可执行权限:

```
[hadoop@master ~]$ chmod 777 /home/hadoop/mapper.py
[hadoop@master ~]$ chmod 777 /home/hadoop/reducer.py
```

将程序放到集群上面去运行:

```
[hadoop@master ~]$ hadoop jar
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -mapper
/home/hadoop/mapper.py -reducer /home/hadoop/reducer.py -input
/shixun/word_in/wordcount.txt -output /shixun/word_out3/
```

```
[hadoop@master ~]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -mapper /home/hadoop/mapper.py -reducer /home/hadoop/reducer.py -input /shixun/word_in/wordcount.txt -output /shixun/word_out3/
2019-07-26 06:41:59,004 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-07-26 06:41:59,056 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
```

查看结果:

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/word_out3/part-00000
```

```
[hadoop@master ~]$ hdfs dfs -cat /shixun/word_out3/part-00000
('all', 1)
('remarkable', 1)
('over', 1)
('competing', 1)
('outset', 1)
('ever', 1)
('concepts,', 1)
('Organization,', 1)
('generation,', 1)
('21st', 4)
('rapidly', 1)
('how', 2)
('time.', 1)
('opportunities', 2)
('to', 13)
('belongs', 1)
('hands.', 1)
('World', 1)
('has', 2)
('By', 1)
('Last', 1)
('successful,', 1)
('prevented', 1)
('bid', 1)
('We,', 1)
('year,', 1)
```