电子科技大学信息与软件工程学院

实 验 报 告

学　　号 202009120****

姓　　名 ***

（实验） 课程名称 大数据分析与智能计算

理论教师 罗瑜

实验教师 罗瑜

电子科技大学教务处制表

电 子 科 技 大 学

实　验　报　告

学生姓名：***　　学号：202009120****指导教师：罗瑜

实验地点：xxxxxx　　实验时间：XX.XX.XX

一、实验名称：实验 1-环境安装及 hadoop 和 spark 比较

二、实验学时：4 学时

三、实验目的：

了解掌握 hadoop 与 spark 在 Linux 上的安装、配置以及实例运行

四、实验原理：

适用于 Linux 的 Windows 子系统 (WSL) 是 Windows 的一项功能，可用于在 Windows 计算机上运行 Linux 环境，而无需单独的虚拟机或双引导。 WSL 旨在为希望同时使用 Windows 和 Linux 的开发人员提供无缝高效的体验。WSL 的优点包括 Windows 和 Linux 之间的无缝集成，启动时间短，资源占用量少，并且无需 VM 配置或管理，等等。学生曾用 WSL 作为在 Windows 系统下开发 Linux 项目的主要工具，如学生在企业实习环节参与的 LarkSDK 项目。

Hadoop 的设计思路来源于 Google 的 GFS 和 MapReduce。它是一个开源软件框架，通过在集群计算机中使用简单的编程模型，可编写和运行分布式应用程序处理大规模数据。

Spark 是一个基于内存计算的开源的集群计算系统，目的是让数据分析更加快速。Spark 提供了基于内存的计算集群，在分析数据时将数据导入内存以实现快速查询，“速度比”基于磁盘的系统，如比 Hadoop 快很多。Spark 最初是为了处理迭代算法，如机器学习、图挖掘算法等，以及交互式数据挖掘算法而开发的。在这两种场景下，Spark 的运行速度可以达到 Hadoop 的几百倍。

五、实验内容：

hadoop 单机模式安装、Spark 安装

测试安装

使用 Hadoop MapReduce、Spark 框架分别运行 wordcount 分析程序，来对 MapReduce 和 Spark 的性能进行对比。

六、实验器材（设备、元器件）：

适用于 Linux 的 Windows 子系统 (WSL)

jdk-8u391-linux-x64.tar.gz

hadoop-3.3.6.tar.gz

spark-3.5.0-bin-hadoop3.tgz

本实验使用的数据集，保存成 word.txt 文件

Python(3.7 以上版本)

七、实验步骤：

一、Hadoop 安装配置

需要先添加用来运行 Hadoop 进程的用户组 hadoop 及用户 hadoop。

1、添加用户及用户组

创建用户和用户组 hadoop

```
$ sudo mkdir -p /hadoop
$ sudo groupadd hadoop
$ sudo useradd -g hadoop -G hadoop -d /hadoop hadoop
$ sudo chown -R hadoop:hadoop /hadoop
$ sudo usermod -s /bin/bash hadoop
```

按照提示输入 hadoop 用户的密码，例如密码设定为 hadoop。注意输入密码的时候是不显示的。

2、添加 sudo 权限

将 hadoop 用户添加进 sudo 用户组：

```
$ sudo usermod -G sudo hadoop
```

上述步骤执行结果如下

```
jackchan@LAPTOP-HU42FJIU:~$ sudo mkdir -p /hadoop
[sudo] password for jackchan:
jackchan@LAPTOP-HU42FJIU:~$ sudo groupadd hadoop
jackchan@LAPTOP-HU42FJIU:~$ sudo useradd -g hadoop -G hadoop -d /hadoop hadoop
jackchan@LAPTOP-HU42FJIU:~$ sudo chown -R hadoop:hadoop /hadoop
jackchan@LAPTOP-HU42FJIU:~$ sudo usermod -s /bin/bash hadoop
jackchan@LAPTOP-HU42FJIU:~$ sudo passwd hadoop
New password:
Retype new password:
passwd: password updated successfully
jackchan@LAPTOP-HU42FJIU:~$ sudo usermod -G sudo hadoop
```

3、安装及配置依赖的软件包

（1）hadoop 环境需要预安装 openssh-server、java 等，这些软件包在实验环境中如果没有，需要手工安装。

更新 Ubuntu:

```
$ sudo apt-get update

$ sudo apt-get upgrade
```

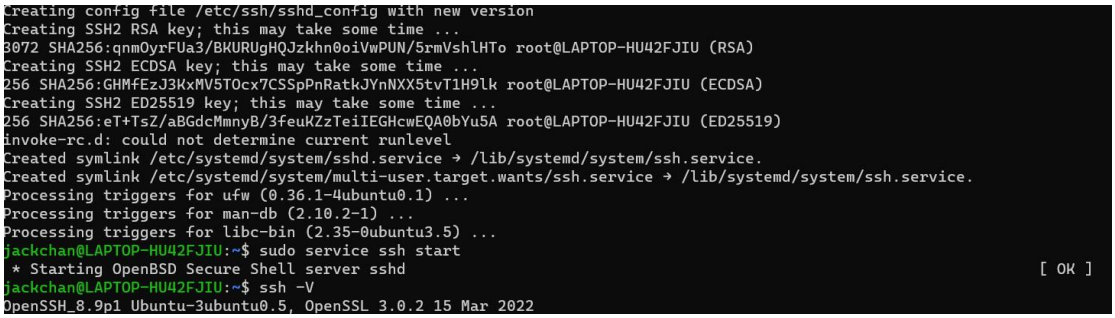
安装并启动 openssh-server：

```
$ sudo apt-get install openssh-server -y

$ sudo service ssh start
```

```
jackchan@LAPTOP-HU42FJIU:~$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libwrap0 ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  libwrap0 ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 5 newly installed, 0 to remove and 1 not upgraded.
Need to get 800 kB of archives.
After this operation, 6161 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-sftp-server amd64 1:8.9p1-3ubuntu0.5 [38.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwrap0 amd64 7.6.q-31build2 [47.9 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-server amd64 1:8.9p1-3ubuntu0.5 [435 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-term all 6.3-2ubuntu0.1 [267 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 ssh-import-id all 5.11-0ubuntu1 [10.1 kB]
Fetched 800 kB in 5s (171 kB/s)
Preconfiguring packages ...
Selecting previously unselected package openssh-sftp-server.
(Reading database ... 33325 files and directories currently installed.)
Preparing to unpack .../openssh-sftp-server_1%3a8.9p1-3ubuntu0.5_amd64.deb ...
Unpacking openssh-sftp-server (1:8.9p1-3ubuntu0.5) ...
Selecting previously unselected package libwrap0:amd64.
Preparing to unpack .../libwrap0_7.6.q-31build2_amd64.deb ...
Unpacking libwrap0:amd64 (7.6.q-31build2) ...
Selecting previously unselected package openssh-server.
Preparing to unpack .../openssh-server_1%3a8.9p1-3ubuntu0.5_amd64.deb ...
```

验证环境执行下列指令：

```
$ ssh -V
```



（2）配置 ssh 免密码登录

在配置 ssh 免密登录之前，首先修改 ssh 的配置文件。

```
$ sudo nano /etc/ssh/sshd_config
```

搜索“PermitRootLogin prohibit-password”，如果此句被注释则打开注释。

搜索“PubkeyAuthentication yes”，如果此句被注释则打开注释。

```
sudo nano /etc/ssh/sshd_config
```

检查

```
PermitRootLogin yes      # 远程 root 密码登录开启
StrictModes no
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile          .ssh/authorized_keys
```

sudo nano /etc/ssh/ssh_config

增加一句

```
PubkeyAcceptedKeyTypes +ssh-rsa
```

然后重启 sshd 服务：

```
service sshd.service restart
```

随后配置 ssh 的免密登录。切换到 hadoop 用户，需要输入添加 hadoop 用户时配置的密码。后续步骤都将在 hadoop 用户的环境中执行。

```
$ su - hadoop
```

配置 ssh 环境免密码登录，在/hadoop 目录下执行：

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 600 ~/.ssh/id_rsa ~/.ssh/id_rsa.pub
```

检查权限可用 ls -l -a 命令

```
jackchan@LAPTOP-HU42FJIU:~$ su - hadoop
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/hadoop/.hushlogin file.
hadoop@LAPTOP-HU42FJIU:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/hadoop/.ssh/id_rsa):
Created directory '/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /hadoop/.ssh/id_rsa
Your public key has been saved in /hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Ypxb3ohjm53BpOv/djIxLDNrEXdCAVOny8NMdmriJlA hadoop@LAPTOP-HU42FJIU
The key's randomart image is:
+---[RSA 3072]----+
|        ooo..    |
|         ..o     |
|      E  .+ .    |
|     o ..*o+.    |
|    . = S+Oo     |
|     o @==+.     |
|      B B*.o     |
|     . Ooo+ .    |
|     .=o+o.+     |
+----[SHA256]-----+
```

验证登录本机是否还需要密码，第一次需要输入 yes，以后不需要密码就可以登录。

$ ssh localhost

```
hadoop@LAPTOP-HU42FJIU:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
hadoop@LAPTOP-HU42FJIU:~$ chmod 600 .ssh/authorized_keys
hadoop@LAPTOP-HU42FJIU:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:eT+TsZ/aBGdcMmnyB/3feuKZzTeiIEGHcwEQA0bYu5A.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

hadoop@LAPTOP-HU42FJIU:~$
```

验证完后退出远程登录

$ exit

4、下载并安装 JAVA 和 Hadoop

开启 WSL 的镜像网络功能：在 Windows 的 %UserProfile%目录下(C:\Users\<UserName>)

创建.wslconfig 文件并设置 networkingMode=mirrored，以启用镜像模式网络。启用此功能会将 WSL 更改为全新的网络体系结构，将 Windows 上的网络接口“镜像”到 Linux 中，以添加新的网络功能并提高兼容性。开启镜像网络以后，WSL 内的网络连接可以享受与 Windows 下相同的速率。

JAVA 安装包下载地址（jdk1.8.0_XX 最新版本）：

https://www.oracle.com/java/technologies/downloads/

Hadoop 安装包下载地址：

https://archive.apache.org/dist/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz

开启镜像网络后，使用 wget 命令下载文件，下载完成后在 hadoop 用户登录的环境中进行下列操作：

（1）复制压缩包到/hadoop 目录下(theses are all in root dir)

```
$ sudo cp /home/xxx/Desktop/hadoop-3.3.6.tar.gz /hadoop/
$ sudo cp /home/xxx/Desktop/jdk-8u391-linux-x64.tar.gz /hadoop/
$ sudo cp /home/xxx/Desktop/spark-3.5.0-bin-hadoop3.tgz /hadoop/
```

（2）解压并安装

```
$ cd /hadoop
$ sudo chmod 777 /hadoop
$ tar -zxvf hadoop-3.3.6.tar.gz
$ tar -zxvf jdk-8u391-linux-x64.tar.gz
```

（3）打开一个新的终端并配置 Hadoop 环境变量

```
$ sudo nano /hadoop/.bash_profile
```

增加以下内容：

```
#HADOOP START
export JAVA_HOME=/hadoop/jdk1.8.0_391
export HADOOP_HOME=/hadoop/hadoop-3.3.6
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$JAVA_HOME/bin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

```
#HADOOP END
```

配置成功后，激活新加的环境变量

```
$ source ~/.bash_profile
```

```
hadoop@LAPTOP-HU42FJIU:~$ bash
hadoop@LAPTOP-HU42FJIU:~$ source .bash_profile
hadoop@LAPTOP-HU42FJIU:~$ java
Usage: java [-options] class [args...]
           (to execute a class)
   or  java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -server       to select the "server" VM
                  The default VM is server,
                  because you are running on a server-class machine.


    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A : separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D<name>=<value>
                  set a system property
    -verbose:[class|gc|jni]
                  enable verbose output
    -version      print product version and exit
    -version:<value>
                  Warning: this feature is deprecated and will be removed
                  in a future release.
                  require the specified version to run
    -showversion  print product version and continue
    -jre-restrict-search | -no-jre-restrict-search
                  Warning: this feature is deprecated and will be removed
                  in a future release.
                  include/exclude user private JREs in the version search
    -? -help      print this help message
    -X            print help on non-standard options
    -ea[:<packagename>...|:<classname>]
    -enableassertions[:<packagename>...|:<classname>]
                  enable assertions with specified granularity
    -da[:<packagename>...|:<classname>]
```

（4）查看 JAVA 和 Hadoop 版本

```
$ java -version

$ hadoop version
```

```
hadoop@LAPTOP-HU42FJIU:~$  java -version
java version "1.8.0_391"
Java(TM) SE Runtime Environment (build 1.8.0_391-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.391-b13, mixed mode)
hadoop@LAPTOP-HU42FJIU:~$ hadoop version
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /hadoop/hadoop-3.3.6/share/hadoop/common/hadoop-common-3.3.6.jar
```

至此，Hadoop 单机模式安装完成，可以通过下述步骤的测试来验证安装是否成功。

5、实验结果

创建输入的数据，暂时采用/etc/protocols 文件作为测试。

```
$ mkdir /hadoop/hadoop_project
```

```
$ cd /hadoop/hadoop_project

$ cp /etc/protocols input
```

执行 Hadoop WordCount 应用（词频统计）：

```
$ /hadoop/hadoop-3.3.6/bin/hadoop jar /hadoop/hadoop-3.3.6/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.3.6-sources.jar org.apache.hadoop.examples.WordCount input output
```

查看生成的单词统计数据：

```
$ cat output/*
```

```
hadoop@LAPTOP-HU42FJIU:~/hadoop_project$ cat output/*
"reliable       1
#       64
(Cisco) 2
(IP)    1
(officially     2
0       2
1       1
103     1
108     1
112     1
115     1
12      1
124     1
132     1
133     1
135     1
136     1
137     1
138     1
139     1
140     1
141     1
142     1
17      1
2       1
20      1
22      1
27      1
29      1
3       1
33      1
36      1
37      1
38      1
4       2
41      1
43      1
44      1
```

练习题

请使用 hadoop 的 wordcount 对日志文件/var/log/dpkg.log 进行词频统计。将你执行的命令，和输出的结果粘贴到下面。

```
hadoop@LAPTOP-HU42FJIU:~/hadoop_project$ rm -r output
hadoop@LAPTOP-HU42FJIU:~/hadoop_project$ rm input
hadoop@LAPTOP-HU42FJIU:~/hadoop_project$ cp /var/log/dpkg.log input
hadoop@LAPTOP-HU42FJIU:~/hadoop_project$ /hadoop/hadoop-3.3.6/bin/hadoop jar /hadoop/hadoop-3.3.6/share/hadoop/mapreduce
/sources/hadoop-mapreduce-examples-3.3.6-sources.jar org.apache.hadoop.examples.WordCount input output
```

```
hadoop@LAPTOP-HU42FJIU:~/hadoop_project$ cat output/*
0.0.17  7
0.0.7-1build2   7
0.04-10build3   7
0.06-9  7
0.0~2022.01.22-1        7
0.1.10-2.1build3        7
0.1.29-1build1  14
0.1.3-1 7
0.104-0ubuntu2  22
0.105-0ubuntu2~22.04.1  20
0.105-33        35
0.106.1-7ubuntu0.22.04.2        12
0.11.0-1build2  7
0.11.5-1        7
0.13-4build2    7
0.14.4-1        7
0.15-2build4    11
0.15-3~ubuntu1.22.04.1  10
0.15-3~ubuntu1.22.04.2  6
0.15.2-2        7
0.16-3  7
0.17-2  7
0.17-2ubuntu4   7
0.17-44build1   7
0.17029-1       7
0.18+nmu1       8
0.186-1build1   28
0.2     7
0.2.13-2        7
```

二、Hadoop 伪分布式安装

1、hadoop 伪分布式配置

hadoop 的配置文件存放在/hadoop/hadoop-3.3.6/etc/hadoop 下，要修改该目录下的文件 core-site.xml 和 hdfs-site.xml 来达到实现伪分布式配置。

在用户 xxx（Ubuntu 的初始登录账号）下，修改 core-site.xml，将<configuration></configuration>修改为：

```
sudo nano /hadoop/hadoop-3.3.6/etc/hadoop/core-site.xml
```

```
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>file:/hadoop/hadoop-3.3.6/tmp</value>
        <description>Abase for other temporary directories.</description>
```

```
    </property>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

修改 hdfs-site.xml，将<configuration></configuration>修改为：

```
sudo nano /hadoop/hadoop-3.3.6/etc/hadoop/hdfs-site.xml
```

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/hadoop/hadoop-3.3.6/tmp/dfs/name</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/hadoop/hadoop-3.3.6/tmp/dfs/data</value>
    </property>
</configuration>
```

配置完成后在/hadoop/hadoop-3.3.6 下使用命令：

```
# 实现 namenode 的格式化
$ /hadoop/hadoop-3.3.6/bin/hdfs namenode -format
```

注意不要多次格式化，否则会在后面运行实例时报错“could only be written to 0 of the 1 minReplication nodes.”

解决方法如下：

1、 停止集群所有的服务。指令为：stop-all.sh

2、删除 hdfs 中配置的 data 目录下的所有文件（级 core-site.xml 中配置的 hadoop.tmp.dir）。指令为：rm -rf /hadoop/hadoop-3.3.6/tmp/*

3、重新格式化 namenode。指令为：hadoop namenode -format

4、重新启动 hadoop 集群。指令为：start-all.sh

另外，需要修改/etc/hadoop/hadoop-env.sh 中搜索并设置 JAVA_HOME。

```
sudo nano /hadoop/hadoop-3.3.6/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/hadoop/jdk1.8.0_391
```

2、启动 hadoop（namenode 节点）（start-all.sh 在 sbin 里面）

启动命令为：

```
$ /hadoop/hadoop-3.3.6/sbin/start-all.sh
```

检查是否运行成功，通过执行 jps 命令可以查看到 hadoop 的几个主要进程：

```
$ jps
```

```
hadoop@LAPTOP-HU42FJIU:~$ jps
5040 NodeManager
4498 DataNode
4722 SecondaryNameNode
7846 Jps
4381 NameNode
4927 ResourceManager
```

三、安装与配置 Spark

1、解压并安装 Spark

本次实验我们将 spark 安装在/hadoop 下。下载安装包：

$ wget https://archive.apache.org/dist/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz

解压

```
$ cd /hadoop
$ tar -zxvf spark-3.5.0-bin-hadoop3.tgz
```

删除安装文件

```
$ rm -r spark-3.5.0-bin-hadoop3.tgz
```

修改文件名称

```
$ mv spark-3.5.0-bin-hadoop3 spark
```

2、配置 Hadoop 环境变量

在 Yarn 上运行 Spark 需要配置 HADOOP_CONF_DIR、YARN_CONF_DIR 和 HDFS_CONF_DIR 环境变量

命令：

```
$ sudo nano /hadoop/.bash_profile
```

在下面添加如下代码：

```
# SPARK START
export SPARK_HOME=/hadoop/spark
export PATH=$PATH:$SPARK_HOME/bin
# check the version of py4j
export
PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.9.7-src.zip:$PYTH
ONPATH
# if run pyspark on Ubuntu 20.04 LTS, it shows the error “python: command not found”
export PYSPARK_PYTHON=python3
# SPARK END
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HDFS_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

保存关闭后，执行以下命令使得环境变量生效：

```
$ source /hadoop/.bash_profile
```

3、修改配置文件

```
$ cd /hadoop/spark/conf/

$ cp spark-env.sh.template spark-env.sh

$ sudo nano /hadoop/spark/conf/spark-env.sh
```

在第一行“#!/usr/bin/env bash”下，写入以下内容

```
export SPARK_MASTER_HOST=127.0.0.1
export SPARK_MASTER_PORT=7077
export SPARK_WORKER_CORES=1
export SPARK_WORKER_MEMORY=512M
```

4、Spark 的启动

（1）进入 spark-shell。进入 Spark 安装主目录，执行命令进入 spark 的 shell 界面：

```
$ /hadoop/spark/bin/spark-shell
```

```
hadoop@LAPTOP-HU42FJIU:~$ sudo nano /hadoop/.bash_profile
hadoop@LAPTOP-HU42FJIU:~$ source /hadoop/.bash_profile
hadoop@LAPTOP-HU42FJIU:~$ cd /hadoop/spark/conf/
hadoop@LAPTOP-HU42FJIU:~/spark/conf$ cp spark-env.sh.template spark-env.sh
hadoop@LAPTOP-HU42FJIU:~/spark/conf$ sudo nano /hadoop/spark/conf/spark-env.sh
hadoop@LAPTOP-HU42FJIU:~/spark/conf$  /hadoop/spark/bin/spark-shell
23/12/28 02:10:24 WARN Utils: Your hostname, LAPTOP-HU42FJIU resolves to a loopback address: 127.0.1.1; using 172.20.16.
142 instead (on interface eth0)
23/12/28 02:10:24 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/12/28 02:10:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Spark context Web UI available at http://172.20.16.142:4040
Spark context available as 'sc' (master = local[*], app id = local-1703700631255).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_391)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

使用“CTRL+C”键退出 spark-shell 界面。

（2）启动 spark

1）首先启动 master

```
$ /hadoop/spark/sbin/start-master.sh
```

```
scala> hadoop@LAPTOP-HU42FJIU:~/spark/conf$ /hadoop/spark/sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /hadoop/spark/logs/spark-hadoop-org.apache.spark.deploy.maste
r.Master-1-LAPTOP-HU42FJIU.out
hadoop@LAPTOP-HU42FJIU:~/spark/conf$ jps
10035 Master
10104 Jps
27898 ResourceManager
hadoop@LAPTOP-HU42FJIU:~/spark/conf$
```

2）启动 slave

```
$ /hadoop/spark/sbin/start-worker.sh spark://127.0.0.1:7077
```

```
hadoop@LAPTOP-HU42FJIU:~/spark/conf$  /hadoop/spark/sbin/start-worker.sh spark://127.0.0.1:7077
starting org.apache.spark.deploy.worker.Worker, logging to /hadoop/spark/logs/spark-hadoop-org.apache.spark.deploy.worke
r.Worker-1-LAPTOP-HU42FJIU.out
hadoop@LAPTOP-HU42FJIU:~/spark/conf$  jps
10035 Master
10132 Worker
27898 ResourceManager
10205 Jps
hadoop@LAPTOP-HU42FJIU:~/spark/conf$
```

5、验证 Spark

运行 pi（π）的实例

```
$ /hadoop/spark/bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://127.0.0.1:7077 --driver-memory 512M --executor-memory 512M --executor-cores 1 /hadoop/spark/examples/jars/spark-examples*.jar 2>&1 | grep "Pi is roughly"
```

```
hadoop@LAPTOP-HU42FJIU:~/spark/conf$  /hadoop/spark/bin/spark-submit --class org.apache.spark.examples.SparkPi --master
spark://127.0.0.1:7077  --driver-memory 512M --executor-memory 512M --executor-cores 1 /hadoop/spark/examples/jars/spark
-examples*.jar 2>&1 | grep "Pi is roughly"
Pi is roughly 3.1379756898784494
```

四、将本次实验的数据文件上传到 HDFS 文件系统

先建立一个/hadoop/data 目录，把实验分析的 word.txt 文件放在此文件夹中。

```
$ mkdir /hadoop/data

$ cd /hadoop/data
```

然后将 word.txt 上传至该目录下

将文件上传到 HDFS/wordcount：

```
$ hadoop fs -mkdir /wordcount

$ hadoop fs -put /hadoop/data/word.txt /wordcount

$ hadoop fs -ls -R    /wordcount
```

```
hadoop@LAPTOP-HU42FJIU:~$ mkdir /hadoop/data
hadoop@LAPTOP-HU42FJIU:~$ cd /hadoop/data
hadoop@LAPTOP-HU42FJIU:~/data$ sudo cp /home/jackchan/word.txt /hadoop/data
[sudo] password for hadoop:
hadoop@LAPTOP-HU42FJIU:~/data$ ls -l
total 4
-rw-r--r-- 1 root root 2932 Dec 28 02:32 word.txt
hadoop@LAPTOP-HU42FJIU:~/data$ du -h word.txt
4.0K    word.txt
hadoop@LAPTOP-HU42FJIU:~/data$ wc -c word.txt
2932 word.txt
hadoop@LAPTOP-HU42FJIU:~/data$ hadoop fs -mkdir /wordcount
hadoop@LAPTOP-HU42FJIU:~/data$ hadoop fs -put /hadoop/data/word.txt /wordcount
hadoop@LAPTOP-HU42FJIU:~/data$  hadoop fs -ls -R  /wordcount
-rw-r--r--   1 hadoop supergroup       2932 2023-12-28 02:33 /wordcount/word.txt
```

3、MapReduce 实现 WordCount 实例（Python）

进入目录/hadoop/data。

（1）首先编写 MapReduce WordCount 代码。

a. 编写 map 阶段的代码，创建一个 Python 程序，命名为“count_mapper.py”，写入如下内容：

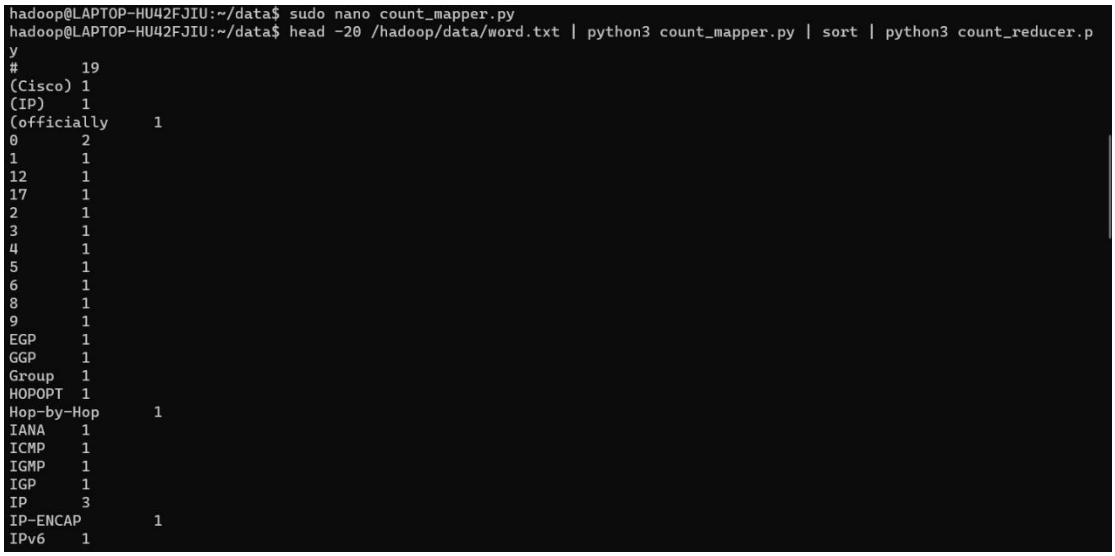
```
#!/usr/bin/env python3
import sys

# 从标准输入过来的数据
for line in sys.stdin:
    # 将首位的空格去掉
    line = line.strip()
    # 将这一行文本切分成单词（按空格）
    words = line.split()
    # 读一个单词写出一个<单词,1>
    for word in words:
        print("%s\t%s" % (word, 1))
```

b. 编写 Reduce 阶段的代码，创建一个 Python 程序，命名为“count_reducer.py”，写入如下内容：

```
#!/usr/bin/env python3

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# 从标准输入过来的数据
for line in sys.stdin:
    # 去除左右空格
     line = line.strip()
          # 按照 tab 键进行切分，得到 word 和次数 1
    word, count = line.split('\t', 1)

    # 得到的 1 是一个字符串，需要类型转化
    try:
        count = int(count)
    except ValueError:
        continue

    # 如果本次读取的单词和上一次一样，对次数加 1
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # 输出统计结果
            print ("%s\t%s" % (current_word, current_count))
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print ("%s\t%s" % (current_word, current_count))
```

（2）程序编写完成后，首先在本地测试一下 map 和 reduce，命令及图片如下：

```
$ head -20 /hadoop/data/word.txt | python3 count_mapper.py | sort | python3 count_reducer.py
```

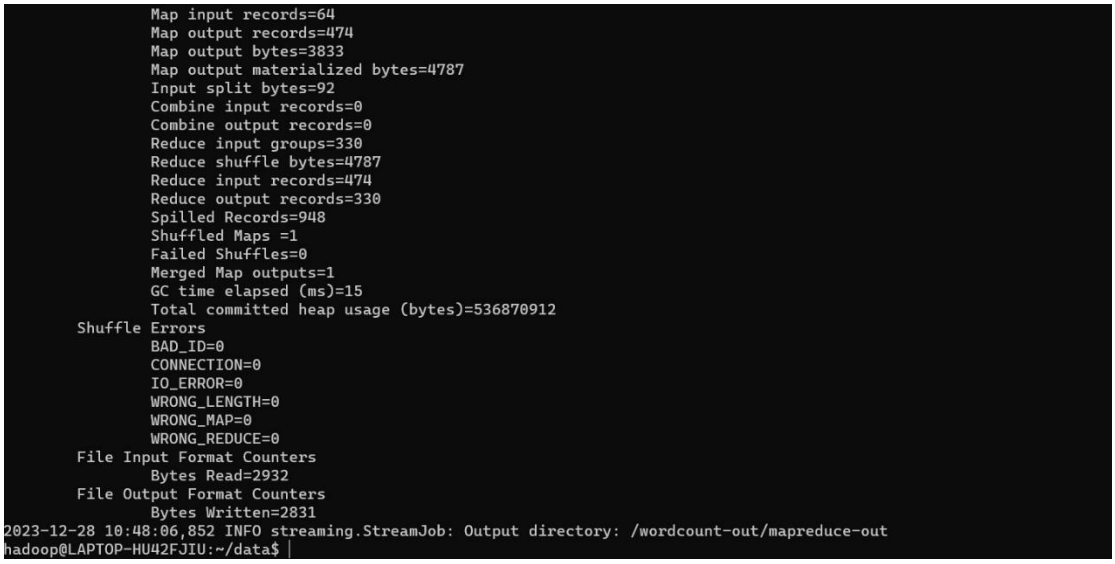


（3）运行该实例，命令如下：

```
$ hadoop jar /hadoop/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -file count_mapper.py -mapper count_mapper.py -file count_reducer.py -reducer count_reducer.py -input /wordcount/word.txt -output /wordcount-out/mapreduce-out
```

（4）查看结果

```
$ hadoop fs -tail /wordcount-out/mapreduce-out/part-00000
```



4、Spark 实现 WordCount 实例（python）

注意：使用 spark-3.1.2-bin-hadoop3.2 以上版本可能会提示“'NoneType' object has no attribute 'items'”错误，应该使用 spark-3.5.0-bin-hadoop3 版本。

（1）首先编写 Spark WordCount 代码，创建一个 Python 程序，命名为“wordcount.py”，写入如下内容：

```
#!/usr/bin/env python3
#导入包
from pyspark import SparkContext
# 输入输出路径，输出路径不需要自己创建，系统会自动生成
inputFile = 'hdfs://localhost:9000/wordcount/word.txt'
outputFile = 'hdfs://localhost:9000/wordcount-out/spark-out'

sc = SparkContext('local', 'wordcount')
text_file = sc.textFile(inputFile)

counts = text_file.flatMap(lambda line: line.split(' ')).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)
counts.saveAsTextFile(outputFile)
```

（2）运行该实例

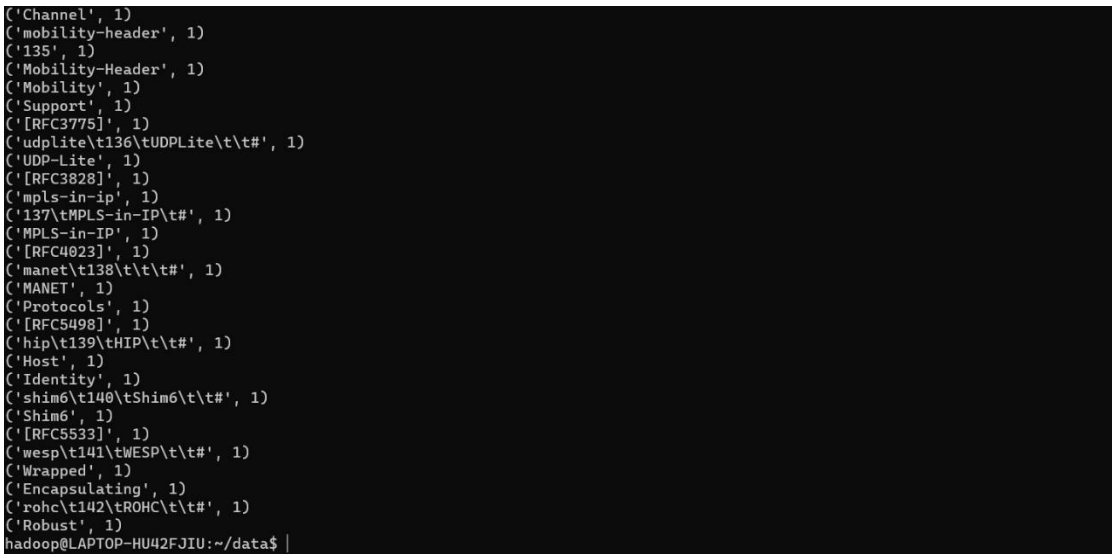
```
$ /hadoop/spark/bin/spark-submit --master spark://localhost:7077 /hadoop/data/wordcount.py
```

注：如果目录存在则先删除

```
$ hadoop fs -rm -r /wordcount-out/spark-out
```

（3）查看运行结果

```
$ hadoop fs -tail /wordcount-out/spark-out/part-00000
```



5、实验结果

分析对比使用 Hadoop 的 MapReduce 和 Spark 两者的计算速度。

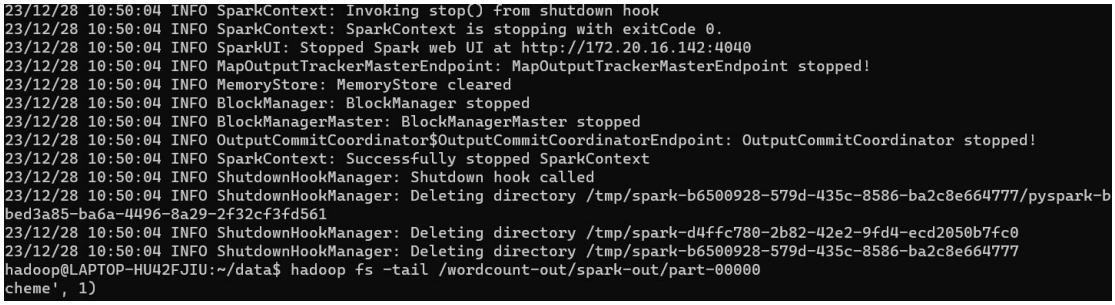
在命令行的回显中分别找到 Hadoop 与 Spark 命令运行的起止时间：

```
hadoop@LAPTOP-HU42FJIU:~/data$ hadoop fs -rm -r /wordcount-out/mapreduce-out
Deleted /wordcount-out/mapreduce-out
hadoop@LAPTOP-HU42FJIU:~/data$ hadoop jar /hadoop/hadoop-3.3.6/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -file c
ount_mapper.py -mapper count_mapper.py  -file count_reducer.py -reducer count_reducer.py -input /wordcount/word.txt -out
put /wordcount-out/mapreduce-out
2023-12-28 10:48:04,014 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [count_mapper.py, count_reducer.py] [] /tmp/streamjob2789613242644769 67.jar tmpDir=null
2023-12-28 10:48:04,933 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-12-28 10:48:05,023 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-12-28 10:48:05,023 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-12-28 10:48:05,035 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2023-12-28 10:48:05,377 INFO mapred.FileInputFormat: Total input files to process : 1
2023-12-28 10:48:05,436 INFO mapreduce.JobSubmitter: number of splits:1
2023-12-28 10:48:05,545 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local573292594_0001
2023-12-28 10:48:05,545 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-12-28 10:48:05,734 INFO mapred.LocalDistributedCacheManager: Localized file:/hadoop/data/count_mapper.py as file:/h
adoop/hadoop-3.3.6/tmp/mapred/local/job_local573292594_0001_92f32817-96cf-43e3-985e-f095f6846aff/count_mapper.py
2023-12-28 10:48:05,750 INFO mapred.LocalDistributedCacheManager: Localized file:/hadoop/data/count_reducer.py as file:/
hadoop/hadoop-3.3.6/tmp/mapred/local/job_local573292594_0001_9947b18f-dfcf-4129-8dcf-abb697238d2e/count_reducer.py
2023-12-28 10:48:05,828 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-12-28 10:48:05,829 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-12-28 10:48:05,829 INFO mapreduce.Job: Running job: job_local573292594_0001
2023-12-28 10:48:05,831 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2023-12-28 10:48:05,836 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-28 10:48:05,836 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under outpu
t directory:false, ignore cleanup failures: false
2023-12-28 10:48:05,896 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-12-28 10:48:05,902 INFO mapred.LocalJobRunner: Starting task: attempt_local573292594_0001_m_000000_0
2023-12-28 10:48:05,933 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-28 10:48:05,933 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under outpu
```

```
                Merged Map outputs=1
                GC time elapsed (ms)=15
                Total committed heap usage (bytes)=536870912
        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=2932
        File Output Format Counters
                Bytes Written=2831
2023-12-28 10:48:06,852 INFO streaming.StreamJob: Output directory: /wordcount-out/mapreduce-out
```

Hadoop 命令运行的起止时间为 10:48:04~10:48:06，约 2s

```
hadoop@LAPTOP-HU42FJIU:~/data$ sudo nano wordcount.py
[sudo] password for hadoop:
hadoop@LAPTOP-HU42FJIU:~/data$  /hadoop/spark/bin/spark-submit --master spark://localhost:7077 /hadoop/data/wordcount.py
23/12/28 10:49:56 WARN Utils: Your hostname, LAPTOP-HU42FJIU resolves to a loopback address: 127.0.1.1; using 172.20.16.
142 instead (on interface eth0)
23/12/28 10:49:56 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
23/12/28 10:49:58 INFO SparkContext: Running Spark version 3.5.0
23/12/28 10:49:58 INFO SparkContext: OS info Linux, 5.15.133.1-microsoft-standard-WSL2, amd64
23/12/28 10:49:58 INFO SparkContext: Java version 1.8.0_391
23/12/28 10:49:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
23/12/28 10:49:58 INFO ResourceUtils: ==============================================================
23/12/28 10:49:58 INFO ResourceUtils: No custom resources configured for spark.driver.
23/12/28 10:49:58 INFO ResourceUtils: ==============================================================
23/12/28 10:49:58 INFO SparkContext: Submitted application: wordcount
23/12/28 10:49:58 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, a
mount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount
: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
23/12/28 10:49:58 INFO ResourceProfile: Limiting resource is cpu
23/12/28 10:49:58 INFO ResourceProfileManager: Added ResourceProfile id: 0
23/12/28 10:49:58 INFO SecurityManager: Changing view acls to: hadoop
23/12/28 10:49:58 INFO SecurityManager: Changing modify acls to: hadoop
23/12/28 10:49:58 INFO SecurityManager: Changing view acls groups to:
23/12/28 10:49:58 INFO SecurityManager: Changing modify acls groups to:
23/12/28 10:49:58 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view perm
issions: hadoop; groups with view permissions: EMPTY; users with modify permissions: hadoop; groups with modify permissi
ons: EMPTY
23/12/28 10:49:58 INFO Utils: Successfully started service 'sparkDriver' on port 34747.
23/12/28 10:49:59 INFO SparkEnv: Registering MapOutputTracker
23/12/28 10:49:59 INFO SparkEnv: Registering BlockManagerMaster
```



Spark 命令运行的起止时间为 10:49:56~10:50:04，约 6s

故在该实例下 Hadoop 的运行速度快于 Spark

八、实验结果与分析（含重要数据结果分析或核心代码流程分析）

在该实例下，Hadoop 的运行速度快于 Spark

九、总结及心得体会：

本实验进行了 hadoop 单机模式安装、Spark 安装、测试安装并使用 Hadoop MapReduce、Spark 框架分别运行 wordcount 分析程序，来对 MapReduce 和 Spark 的性能进行对比，学生收获颇丰。

十、对本实验过程及方法、手段的改进建议：无

报告评分：

指导教师签字：