数据库原理(2) 实验二

姓名：严昕宇　　学号：20121802　　实验日期：2023.05.03

一、基础环境配置

实验过程 1——系统准备：

1. 更新软件，确保所有存储库和 PPA 中的软件包列表都是最新的

```
sudo apt-get update
```

2. 安装 VIM——Linux 系统上一款文本编辑器

```
sudo apt-get install vim
```

3. 安装 ssh

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

4. 在 master 端创建私钥(id_rsa)与公钥(id_rsa.pub)

```
ssh-keygen -t rsa -C
"yanxinyu@shu.edu.cn"
```

5. 将公钥(id_rsa.pub)中的内容追加到 authorized_keys 中

```
cd ~/.ssh
cat id_rsa.pub >> authorized_keys
```

6. 尝试无密码访问自身

```
ssh localhost
```

实验过程 2——Java 环境安装：

1. 安装 JRE、JDK

```
sudo apt-get install openjdk-8-jre
openjdk-8-jdk
```

2. 配置 JAVA_HOME 环境变量，在 ~/.bashrc 中进行设置：

```
vim ~/.bashrc
```

3. 在文件最开头添加以下内容

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

4. 重新读取刚修改的文件，使变量设置生效

```
source ~/.bashrc
```

5. 检查环境变量是否已经配置成功

```
echo $JAVA_HOME
java -version
```

实验结果 1：

```
ubuntu@master:~$ vim ~/.bashrc
ubuntu@master:~$ source ~/.bashrc
ubuntu@master:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64
ubuntu@master:~$ java -version
openjdk version "1.8.0_362"
OpenJDK Runtime Environment (build 1.8.0_362-8u362-ga-0ubuntu1~20.04.1-b09)
OpenJDK 64-Bit Server VM (build 25.362-b09, mixed mode)
```

二、Hadoop 基础配置

实验过程：

1. 下载 Hadoop 2.7.0

```
wget https://archive.apache.org/dist/
hadoop/common/hadoop-2.7.0/hadoop-
2.7.0.tar.gz
```

2. 将 Hadoop 安装(解压)至 /usr/local/ 中

```
sudo tar -zxf hadoop-2.7.0.tar.gz -C
/usr/local
```

3. 修改文件夹名称

```
cd /usr/local
sudo mv hadoop-2.7.0 hadoop
```

4. 修改文件夹的拥有者

```
sudo chown -R ubuntu ./hadoop
```

5. 检查 Hadoop 是否可用

```
cd /usr/local/hadoop
./bin/hadoop version
```

实验结果：

```
ubuntu@VM-0-17-ubuntu:/usr/local/hadoop$ ./bin/hadoop version
Hadoop 2.7.0
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r d4c8d4d4d203c934e8074b31289a28724c0842cf
Compiled by jenkins on 2015-04-10T18:40Z
Compiled with protoc 2.5.0
From source with checksum a9e90912c37a35c3195d23951fd18f
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.0.jar
```

三、Hadoop 单机配置(非分布式)

实验内容：

- Hadoop 默认模式为非分布式模式，非分布式即单 Java 进程，无需进行其他配置即可运行。
- 此处运行 Hadoop 自带的 grep 例子，将 input 文件夹中的所有文件作为输入，筛选当中符合正则表达式 dfs[a-z.]+的单词并统计出现的次数，最后输出结果到 output 文件夹中。

实验过程：

1. 创建 input 文件夹

```
cd /usr/local/hadoop
mkdir ./input
```

2. 将配置文件作为输入文件

```
cp ./etc/hadoop/*.xml ./input
```

3. 运行程序

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

4. 查看运行结果，并在查看完后删除，避免再次运行时出现错误

```
cat ./output/*
rm -r ./output
```

实验结果：

```
        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=123
        File Output Format Counters
                Bytes Written=23
ubuntu@VM-0-17-ubuntu:/usr/local/hadoop$ cat ./output/*
1       dfsadmin
ubuntu@VM-0-17-ubuntu:/usr/local/hadoop$ rm -r ./output
```

四、Hadoop 伪分布式配置与运行

实验内容：

- Hadoop 可以在单节点上以伪分布式的方式运行，Hadoop 进程以分离的 Java 进程来运行，节点既作为 NameNode 也作为 DataNode，同时，读取的是 HDFS 中的文件。
- 在第三部分中，grep 例子读取的是本地数据，第四部分的伪分布式读取的则是 HDFS 上的数据。

实验过程 1——Hadoop 伪分布式配置：

1. 修改配置文件 core-site.xml（代码略）
2. 同样的，修改配置文件 hdfs-site.xml（代码略）
3. 执行 NameNode 的格式化

```
cd /usr/local/hadoop
bin/hdfs namenode -format
```

4. 开启 NaneNode 和 DataNode 守护进程

```
cd /usr/local/hadoop
./sbin/start-dfs.sh
```

5. 启动完成后，通过命令 jps 来判断是否成功启动

```
jps
```

实验结果 1：

```
23/05/02 23:14:45 INFO util.ExitUtil: Exiting with status 0
23/05/02 23:14:45 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost.localdomain/127.0.1.1
************************************************************/
```

```
ubuntu@VM-0-17-ubuntu:/usr/local/hadoop$ jps
14643 Jps
13963 NameNode
14190 DataNode
14463 SecondaryNameNode
```

实验过程 2——运行 Hadoop 伪分布式实例：

1. 要使用 HDFS，先要在 HDFS 中创建用户目录

```
./bin/hdfs dfs -mkdir -p /user/ubuntu
```

2. 将./etc/hadoop 中的 xml 文件作为输入文件复制到分布式文件系统中

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

3. 查看 HDFS 中的文件列表

```
./bin/hdfs dfs -ls input
```

4. 运行程序，查看结果

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

实验结果 2：

结果如下，注意到刚才已经更改了配置文件，所以运行结果不同：

```
ubuntu@VM-0-17-ubuntu:/usr/local/hadoop$ ./bin/hdfs dfs -cat output/*
1       dfsadmin
1       dfs.replication
1       dfs.namenode.name.dir
1       dfs.datanode.data.dir
```

实验过程 3—运行结果取回到本地：

1. 先删除本地的 output 文件夹(如果存在)

```
rm -r ./output
```

2. 将 HDFS 上的 output 文件夹拷贝到本机

```
./bin/hdfs dfs -get output ./output
```

3. 查看 HDFS 中的文件列表

```
cat ./output/*
```

4. Hadoop 运行程序时，输出目录不能存在，否则会提示错误。因此若要再次执行，需要执行如下命令删除 output 文件夹

```
./bin/hdfs dfs -rm -r output
```

5. 关闭 Hadoop

```
./sbin/stop-dfs.sh
```

实验结果 3：

```
ubuntu@VM-0-17-ubuntu:/usr/local/hadoop$ cat ./output/*
1       dfsadmin
1       dfs.replication
1       dfs.namenode.name.dir
1       dfs.datanode.data.dir
```

五、Hadoop 两地三中心配置与运行

实验过程 1——分布式基础配置：

1. 使用镜像创建实例，注意实例共分布在两地
2. 删除 known_host 文件

```
cd .ssh
rm known_host
```

3. 修改主机名(重启后生效)

```
sudo vim /etc/hostname
sudo hostname master(slave01、slave02)
```

4. 修改 hosts 文件

```
sudo vim /etc/hosts
格式：内网 IP + Name
```

实验过程 2——修改 Hadoop 配置文件：

1. 在 master 上修改配置文件 slaves、core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml（代码略）
2. 将 master 上修改后的配置文件发送到 slave01(slave02 同理)

```
scp 文件名 ubuntu@slave01:/usr/local/hadoop/etc/hadoop/文件名
```

3. 确认配置文件发送成功

```
cd /usr/local/hadoop/etc/hadoop
cat slaves
```

实验过程 3—执行分布式实例：

1. 启动 Hadoop

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

2. 启动完成后，判断各节点是否成功启动

```
jps
```

3. 创建 HDFS 上的用户目录

```
./bin/hdfs dfs -mkdir -p /user/ubuntu
```

4. 将配置文件作为输入文件，复制到 HDFS 系统中

```
./bin/hdfs dfs -put ./etc/hadoop/core-site.xml
```

5. 将刚上传的文件从 HDFS 系统中下载到本地

```
./bin/hdfs dfs -ls
./bin/hdfs dfs -get core-site.xml ./
```

6. 关闭与 master 同一中心的 slave01 后，再次尝试下载之前上传的文件

```
./bin/hdfs dfs -get core-site.xml ./
```

实验结果：

- 各个节点的进程启动情况

通过命令 jps 可以查看各个节点所启动的进程。
正确的话，在 master 节点上可以看到 JPS、NameNode、ResourceManager、SecondrryNameNode 进程，
在两个 slave 节点可以看到 JPS、DataNode 和 NodeManager 进程，如下图所示：

```
ubuntu@master:/usr/local/hadoop$ jps
7360 Jps
6929 SecondaryNameNode
6643 NameNode
7097 ResourceManager
```

```
ubuntu@slave01:/usr/local$ jps
6899 Jps
6549 DataNode
6751 NodeManager
```

```
ubuntu@slave02:/usr/local$ jps
5237 Jps
4582 DataNode
4781 NodeManager
```

- 关闭与 master 同一中心的 slave02 后，再次尝试下载之前上传的文件，仍能正常先下载。体现了分布式系统的容灾能力

```
ubuntu@master:/usr/local/hadoop$ ./bin/hdfs dfs -ls
Found 2 items
-rw-r--r--   3 ubuntu supergroup       1123 2023-05-03 21:15 core-site.xml
drwxr-xr-x   - ubuntu supergroup          0 2023-05-03 21:15 yanxinyu
ubuntu@master:/usr/local/hadoop$ ./bin/hdfs dfs -get core-site.xml ./
23/05/03 21:16:06 WARN hdfs.DFSClient: DFSInputStream has been closed already
ubuntu@master:/usr/local/hadoop$ ls
bin  core-site.xml  etc  include  lib  libexec  LICENSE.txt  logs  NOTICE.txt
```

```
ubuntu@master:/usr/local/hadoop$ rm core-site.xml
ubuntu@master:/usr/local/hadoop$ ./bin/hdfs dfs -get core-site.xml ./
23/05/03 21:18:56 WARN hdfs.DFSClient: DFSInputStream has been closed already
ubuntu@master:/usr/local/hadoop$ ls
bin  core-site.xml  etc  include  lib  libexec  LICENSE.txt  logs  NOTICE.txt
```