Homework2: Write a WordCount on Hadoop

组员：王宇琪、胡晨旭、张智为

Hadoop 架构理解

- **三大核心**

1. HDFS:分布式文件系统
2. YARN:资源管理调度系统
3. Mapreduce:分布式运算框架

- **HDFS 架构**

HDFS 遵循主从架构，Hadoop 启动的时候创建了 1 个 NameNode 和 4 个 DataNode

- NameNode：主节点（负责接收请求、维护文件系统、管理文件与 block 以及 block 和 datanode）
- DataNode：从节点（存储文件、文件分成 block 进行存储）
- Secondary NameNode：合并文件来更新 NameNode 的 metadata

- **MapReduce 框架**

MapReduce 的最大工作单元是 job，每个 job 又被分割成不同的 task

- Map 将输入数据划分为 key-value 的集合
- Shuffle 将 Map 划分的结果传给 Reduce
- Reduce 对接到的 key-value 集合处理

WordCount 理解

由于 MapReduce 已经提供了数据处理的框架，我们只需要提供相应的 map 和 reduce 函数即可完成 WordCount。

```
public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {
    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public TokenizerMapper() {
    }
    public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
            context) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while(itr.hasMoreTokens()) {
            this.word.set(itr.nextToken());
            context.write(this.word, one);
        }
    }
}
```

以上代码定义了继承自 Mapper 类的 TokenizerMapper，来实现 Map 的功能：通过 map 函数读入字符串，计入字符串中的单词，并且计数，直到完成全部输入的读取。

```
public static class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();
    public IntSumReducer() {
    }
    public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
            IntWritable>.Context context) throws IOException, InterruptedException {
        int sum = 0;
        IntWritable val;
        for(Iterator i$ = values.iterator(); i$.hasNext(); sum += val.get()) {
            val = (IntWritable)i$.next();
        }
        this.result.set(sum);
        context.write(key, this.result);
    }
}
```

以上代码定义了 IntSumReducer 类实现 Reduce：通过 reduce 函数，统计每个单词的出现次数记录在变量 sum 中，并将 sum 写入到结果中。

云服务器使用流程

按照助教给的教程，申请相应的服务器和公网 IP 即可，然后需要让 hadoop1 服务器绑定自己电脑的公钥，方便使用 ssh 登录到服务器。

1. 恢复服务器

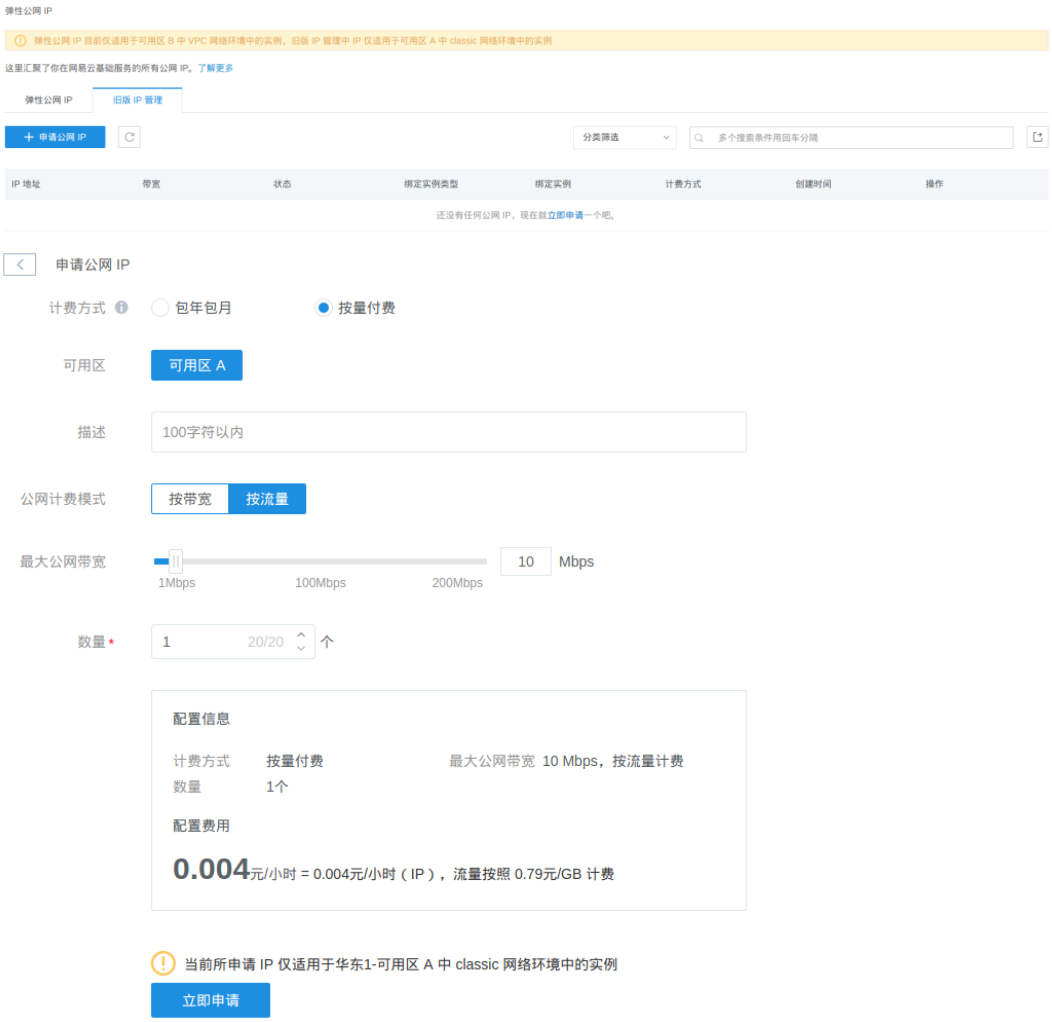
首先选择某一镜像，选择创建云服务器

选择相应的计费方式和可用区

选择相应的配置，绑定相应的 ssh 密钥

确认之后如下图所示：

2. 申请公网 IP



将 IP 与 hadoop1 绑定

这样就可用通过公网 IP 连接到 hadoop1 了，之后通过私有网可连接 hadoop2--hadoop5

```
# huchenxu @ huchenxu-ThinkPad-X1-Carbon-4th in ~ [19:25:03]
$ ssh root@59.111.99.29
The authenticity of host '59.111.99.29 (59.111.99.29)' can't be established.
ECDSA key fingerprint is SHA256:Y19VezPy+bXpRBCKqAfgfZDhKiJO+Ez4DzEoG/cNbEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '59.111.99.29' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-21-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Sep 27 21:14:29 2019 from 183.157.160.50
root@hadoop1:~#
```

因为每次重新申请云服务器，私有网 IP 都会变，所以要更改"/etc/hosts"中各个节点的 IP

```
127.0.0.1       localhost
127.0.1.1       localhost.localdomain   localhost
10.173.32.16     hadoop1
10.173.32.17     hadoop2
10.173.32.18     hadoop3
10.173.32.19     hadoop4
10.173.32.20     hadoop5
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

写了一个脚本来进行：旧的 host key 的删除，和将更改好的"/etc/hosts"文件上传到各个 datanode

```
#!/bin/bash
ssh-keygen -f "/root/.ssh/known_hosts" -R hadoop1
ssh-keygen -f "/root/.ssh/known_hosts" -R hadoop2
ssh-keygen -f "/root/.ssh/known_hosts" -R hadoop3
ssh-keygen -f "/root/.ssh/known_hosts" -R hadoop4
ssh-keygen -f "/root/.ssh/known_hosts" -R hadoop5

scp -r /etc/hosts root@hadoop2:/etc
scp -r /etc/hosts root@hadoop3:/etc
scp -r /etc/hosts root@hadoop4:/etc
scp -r /etc/hosts root@hadoop5:/etc
echo 'update hosts successful'
```

```
root@hadoop1:~# ./update_hosts.sh
# Host hadoop1 found: line 15
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
# Host hadoop2 found: line 7
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
# Host hadoop3 found: line 8
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
# Host hadoop4 found: line 9
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
# Host hadoop5 found: line 10
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
The authenticity of host 'hadoop2 (10.173.32.17)' can't be established.
ECDSA key fingerprint is SHA256:5uAFylcv8QHX3MNcqNRJ0Va4letWJa/MkzLo2dACAew.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hadoop2,10.173.32.17' (ECDSA) to the list of known hosts.
hosts                                         100%  335     0.3KB/s   00:00
The authenticity of host 'hadoop3 (10.173.32.18)' can't be established.
ECDSA key fingerprint is SHA256:NMrRwRluAvp5F+MF0qnkWgYZnXpl4RVPeV2HpalY9XQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hadoop3,10.173.32.18' (ECDSA) to the list of known hosts.
hosts                                         100%  335     0.3KB/s   00:00
The authenticity of host 'hadoop4 (10.173.32.19)' can't be established.
ECDSA key fingerprint is SHA256:l8W4MXpPk1E8tZOeo4zlM5AjCN3LRxUJMyeFPXP32OQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hadoop4,10.173.32.19' (ECDSA) to the list of known hosts.
hosts                                         100%  335     0.3KB/s   00:00
The authenticity of host 'hadoop5 (10.173.32.20)' can't be established.
ECDSA key fingerprint is SHA256:TajMg7s2Cf0kOY4DZq/40I+IUigRtRz57IukPFK2ezc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hadoop5,10.173.32.20' (ECDSA) to the list of known hosts.
hosts                                         100%  335     0.3KB/s   00:00
update hosts successful
```

到此为止，从镜像初始化云服务器的工作就完成了。

3. 云服务器的释放工作

选择一个云服务器，点击更多，选择保存为镜像

在镜像保存完毕后，就可以删除所有的云服务器，最后释放公网 IP

Hadoop 使用的流程

首先调用 start-dfs.sh 和 start-yarn.sh 开启 "dfs" 和 "yarn"

```
root@hadoop1:~# start-dfs.sh
Starting namenodes on [hadoop1]
hadoop1: Warning: Permanently added 'hadoop1,10.173.32.16' (ECDSA) to the list of known hosts.
Starting datanodes
Starting secondary namenodes [hadoop1]
```

在"namenode" 和 "datanode"中调用"jps"查看相应的进程

```
root@hadoop1:~# start-yarn.sh
Starting resourcemanager
Starting nodemanagers
root@hadoop1:~# jps
1232 NameNode
1859 ResourceManager
1580 SecondaryNameNode
2205 Jps
```

```
root@hadoop2:~# jps
3808 Jps
3649 NodeManager
3521 DataNode
```

使用"*hdfs dfsadmin -report*"进一步查看是否正确开启，主要查看"Live datanodes"的个数。

```
root@hadoop2:~# hdfs dfsadmin -report
Configured Capacity: 84412891136 (78.62 GB)
Present Capacity: 68361363456 (63.67 GB)
DFS Remaining: 68345913344 (63.65 GB)
DFS Used: 15450112 (14.73 MB)
DFS Used%: 0.02%
Replicated Blocks:
        Under replicated blocks: 12
        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

-------------------------------------------------
Live datanodes (4):

Name: 10.173.32.17:9866 (hadoop2)
Hostname: hadoop2
Decommission Status : Normal
Configured Capacity: 21103222784 (19.65 GB)
DFS Used: 3862528 (3.68 MB)
Non DFS Used: 3117731840 (2.90 GB)
DFS Remaining: 17084551168 (15.91 GB)
DFS Used%: 0.02%
DFS Remaining%: 80.96%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sun Sep 29 19:29:16 CST 2019
Last Block Report: Sun Sep 29 19:27:50 CST 2019
Num of Blocks: 42


Name: 10.173.32.18:9866 (hadoop3)
Hostname: hadoop3
Decommission Status : Normal
Configured Capacity: 21103222784 (19.65 GB)
DFS Used: 3862528 (3.68 MB)
Non DFS Used: 3115077632 (2.90 GB)
DFS Remaining: 17087205376 (15.91 GB)
DFS Used%: 0.02%
DFS Remaining%: 80.97%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sun Sep 29 19:29:16 CST 2019
```

使用 hadoop1 的公网 IP，加上端口号 9870 即可在 web 端查看运行情况。

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
hadoop2:9866 (10.173.32.17:9866)	http://hadoop2:9864	1s	7m	19.65 GB	72	108.88 MB (0.54%)	3.1.2
hadoop3:9866 (10.173.32.18:9866)	http://hadoop3:9864	1s	42m	19.65 GB	72	108.88 MB (0.54%)	3.1.2
hadoop4:9866 (10.173.32.19:9866)	http://hadoop4:9864	1s	42m	19.65 GB	72	108.88 MB (0.54%)	3.1.2
hadoop5:9866 (10.173.32.20:9866)	http://hadoop5:9864	1s	42m	19.65 GB	72	108.88 MB (0.54%)	3.1.2

Hadoop3.1.2 配置

1. 关于系统环境变量设置

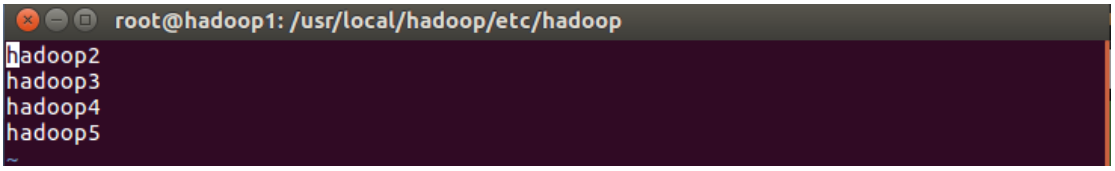
不论是 Java 还是 Hadoop 的环境变量，都需要在五台服务器的 bashrc 上进行配置，否则运行会有错误。

2. 关于 Hadoop3.x 版本的 xml 文件配置

大部分 xml 文件可以按照助教的文档进行配置，有以下几点和助教教程配置有所区别：

- 配置文件中的 slaves 文件变成了 workers 文件

("/usr/local/hadoop/etc/hadoop/workers")，在里面配置 datanode 节点



- 在 start-dfs.sh、start-yarn.sh 以及相应的 stop-x.sh 中添加如下变量

 # start-dfs.sh

HDFS_DATANODE_USER=root

HADOOP_SECURE_DN_USER=hdfs

HDFS_NAMENODE_USER=root

HDFS_SECONDARYNAMENODE_USER=root

 # start-yarn.sh

YARN_RESOURCEMANAGER_USER=root

HADOOP_SECURE_DN_USER=yarn

YARN_NODEMANAGER_USER=root

- 配置 mapred-site.xml 在运行例程或自己的程序前需要设置其中的路径

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

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

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

<property>
    <name>mapreduce.jobhistory.address</name>
    <value>hadoop1:10020</value>
</property>

<property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>hadoop1:19888</value>
</property>

<property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

<property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

<property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

<property>
    <name>mapred.child.java.opts</name>
    <value>-Djava.awt.headless=true</value>
</property>
<!-- add headless to default -Xmx1024m -->
<property>
    <name>yarn.app.mapreduce.am.command-opts</name>
    <value>-Djava.awt.headless=true -Xmx1024m</value>
</property>
<property>
    <name>yarn.app.mapreduce.am.admin-command-opts</name>
    <value>-Djava.awt.headless=true</value>
</property>

</configuration>
```

注意路径直接写成 HADOOP_HOME 即可，若写成绝对路径，后续运行会产生错误。

之后重新打包，传给各个"datanode"

3. 关于重启服务器运行 Hadoop

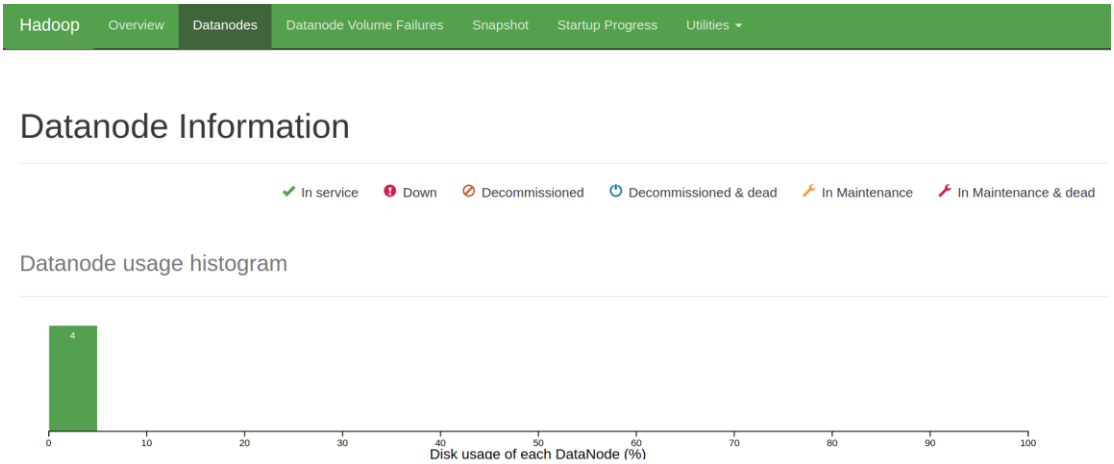
节点执行 NameNode 的初始化只需一次即可，之后的启用不要再运行这条指令

```
hdfs namenode -format
```

如果再次运行，可能会导致 hadoop 启动报错。这个时候需要把每台服务器 hadoop 里的 tmp 文件夹删掉，再重新运行，重新启动。

4. 关于 web 端的端口

端口号为 9870，使用 hadoop1 的公网 ip 加上端口号即可在 web 端查看运行情况



5. 运行 wordcount

运行时注意 input 和 output 的路径设置（联系当前所在目录），如果报内存超限可以使用更高级别的服务器运行程序。

Hadoop 运行 wordcount

首先重新创建个 hdfs 用户目录

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

把本地文件上传到分布式 HDFS 上

```
hadoop fs -put input/ /user/hadoop/
```

"input"文件夹里有我们事先预处理好的"spam-email-dataset"，即将其中所有的文件都输出到一个文件之中，**这样可以极大地加快 MapReduce 的速度**。

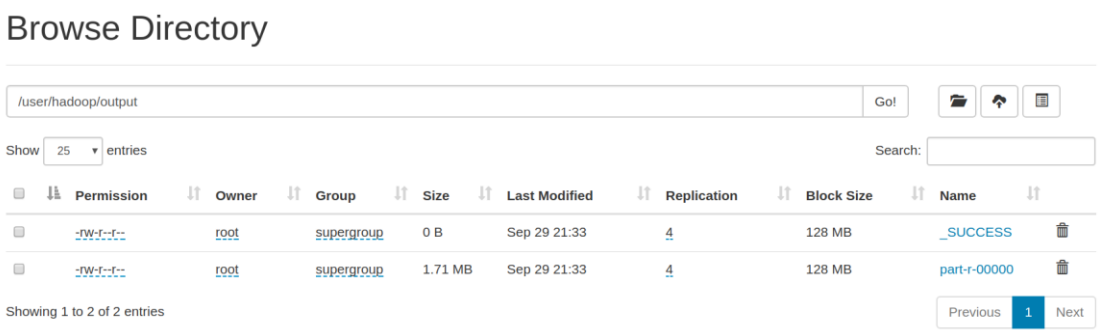
用我们提前打包好的"WordCount Java"程序进行运行：

hadoop jar WordCount.jar WordCount /user/hadoop/input /user/hadoop/output

出现如下情况，说明运行成功！

```
2019-09-29 21:32:36,724 INFO mapreduce.Job: The url to track the job: http://hadoop1:8088/proxy/application_1569761704909_0002/
2019-09-29 21:32:36,725 INFO mapreduce.Job: Running job: job_1569761704909_0002
2019-09-29 21:32:44,858 INFO mapreduce.Job: Job job_1569761704909_0002 running in uber mode : false
2019-09-29 21:32:44,859 INFO mapreduce.Job:  map 0% reduce 0%
2019-09-29 21:33:03,005 INFO mapreduce.Job:  map 47% reduce 0%
2019-09-29 21:33:09,045 INFO mapreduce.Job:  map 67% reduce 0%
2019-09-29 21:33:13,080 INFO mapreduce.Job:  map 100% reduce 0%
2019-09-29 21:33:20,125 INFO mapreduce.Job:  map 100% reduce 100%
2019-09-29 21:33:20,145 INFO mapreduce.Job: Job job_1569761704909_0002 completed successfully
2019-09-29 21:33:20,302 INFO mapreduce.Job: Counters: 53
```

可以在 web 端查看到生成的文件"part-r-00000"，这就是 WordCount 的输出文件。



使用"*hdfs dfs -cat /user/hadoop/output/part-r-00000*" 查看相应的结果

```
werbung 8
werbungen       1
werckle 2
werd    1
werde   12
werden  59
were    7090
werebr  1
wered   1
werefi  2
weren   120
wereselected    1
werfen  1
werhlksdhb      1
werk    2
werken  6
werkmeister     1
werknemers      1
werks   1
werner  33
wernerSubject:  2
weron   2
werona  1
weronem 1
wers    1
wersji  2
werst   1
wert    3
wertheimersoftware      1
werther 4
weruxvnnww      1
werwer  3
```

```
wes     353
wesSubject:     3
wesb    1
wescl   1
wesco   15
wesentliche     1
wesj    2
wesley  28
wesleyan        4
wesloski        2
weslypipesl     1
wesner  10
wesneske        1
wesom   1
wess    1
wessels 1
wessex  92
wessexwater     2
wesson  3
west    1377
westSubject:    2
westafrica      1
westar  4
westborough     1
westbound       7
westboundchalice        1
westbrhong      2
westbro 4
westbrook       51
westchase       2
```

可以看到 WordCount 的结果，说明程序执行成功。

最后使用 "*stop-all.sh*" 关闭"dfs"和"yarn"，并使用 "*jps*" 确认已经完全关闭

```
root@hadoop1:/usr/local/hadoop/sbin# stop-all.sh
WARNING: HADOOP_SECURE_DN_USER has been replaced by HDFS_DATANODE_SECURE_USER. Using value of HADOOP_SECUR
E_DN_USER.
Stopping namenodes on [hadoop1]
Stopping datanodes
Stopping secondary namenodes [hadoop1]
Stopping nodemanagers
Stopping resourcemanager
root@hadoop1:/usr/local/hadoop/sbin# jps
11402 Jps
```

```
Last login: Sun Sep 29 22:26:57 2019 from 10.173.32.16
root@hadoop2:~# jps
4003 Jps
root@hadoop2:~#
```

遇到的问题

1. 队友的 Mac 无法在网上创建面密登陆，总是显示无效秘钥

解决方法：将公钥直接写进 hadoop1 的 authorized key 文件里

2. 启动 hadoop 时显示 publickey 的权限问题

解决方法：将 hadoop1 的公钥写进自己的 authorized key 文件里

3. 使用原始的"spam-email-dataset"作为"WordCount"的"input"，运行速度非常慢

原因：这是因为原始的"dataset"中有接近三万个文件，使用 Hadoop 对其进行一个个的读取是非常慢的

解决方法：将" dataset"中所有的文件都输出到一个文件之中，操作脚本如下：

```
#!/bin/bash
for file in ../train/ham/*
do
    if test -f $file
    then
        cat $file >> merge-all
    fi
done
```

4. 第一次对所有的`spam-email-dataset`进行 map reduce 时，报出了如下错误：

```
2019-09-29 20:51:45,680 INFO mapreduce.Job: Task Id : attempt_1569756487553_0007
_m_000000_2, Status : FAILED
[2019-09-29 20:51:45.402]Container [pid=2115,containerID=container_1569756487553
_0007_01_000004] is running 332110336B beyond the 'VIRTUAL' memory limit. Curren
t usage: 462.7 MB of 1 GB physical memory used; 2.4 GB of 2.1 GB virtual memory
used. Killing container.
```

原因：可以看到，是因为"virtual memory"不足导致相关的"container"直接被"kill"掉了。

解决方法：经过查阅资料，我们知道默认条件下：$\frac{\text{virtual memory}}{\text{physical memory}} = 2.1$，所以要增加$\frac{\text{virtual memory}}{\text{physical memory}}$可以从以下两个方面入手：

- 增大$\frac{\text{virtual memory}}{\text{physical memory}}$
- 增大给每个"container"分配的最小虚拟内存

修改"yarn-site.xml"文件，增加两个"property"

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>hadoop1</value>
    </property>

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

    <property>
        <description>Ratio between virtual memory to physical memory when
        setting memory limits for containers. Container allocations are
        expressed in terms of physical memory, and virtual memory usage
        is allowed to exceed this allocation by this ratio.
        </description>
        <name>yarn.nodemanager.vmem-pmem-ratio</name>
        <value>3.5</value>
    </property>

    <property>
        <description>The minimum allocation for every container request at the
        RM,in MBs. Memory requests lower than this won't take effect,
        and the specified value will get allocated at minimum.
        </description>
        <name>yarn.scheduler.minimum-allocation-mb</name>
        <value>2048</value>
    </property>
```

我们两个方法都使用了：

- 增大$\frac{\text{virtual memory}}{\text{physical memory}} = 3.5$
- 增大"minimum allocation for every container"到2048MB 即 2GB

组员分工

- **实验部分**

三人共同完成，在9月25日，9月27日和9月29日的上机课上一起完成任务。

- **报告部分**

王宇琪：Hadoop架构理解，Hadoop3.1.2配置

胡晨旭：云服务器使用流程，Hadoop使用流程，Hadoop运行WordCount

张智为：WordCount理解，所有文本的整合排版