大数据分析技术
华为云实验报告

学 生 姓 名	：	刘运昊
学 生 学 号	：	20201072008
联 系 方 式	：	18042688180

1 实验内容

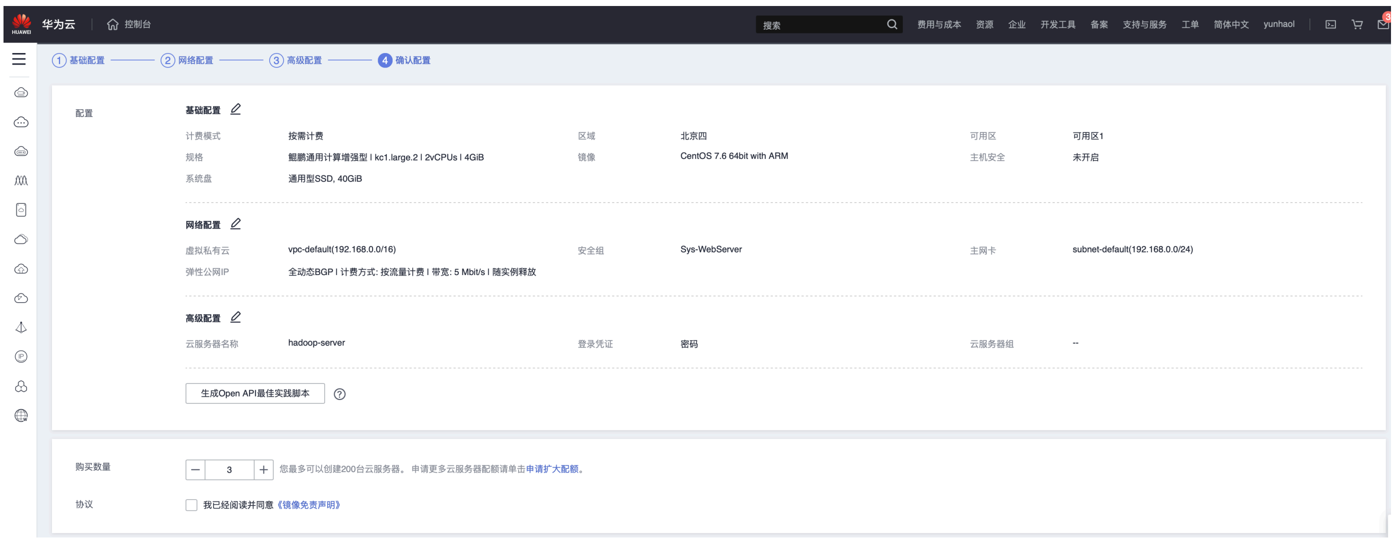
本次实验我主要完成了以下两项任务

- 创建了一组脚本，用于快速在多台 `Linux` 服务器上构建 `Hadoop` 集群，并集成华为的 `OBS` 插件，以实现大数据计算 `Spark`、`MapReduce`、`Hive`、`HBase` 等组件与OBS对象存储服务的对接。这些脚本已经上传到了我的 `Github` 仓库中 [链接]
- 基于 `Hadoop` 实现了一个简单的基于 `MapReduce` 的任务：清洗奥运会的统计数据，输出每位运动员获得金牌、银牌和铜牌的情况。

2 所采用的云服务说明

2.1 集群说明

本次实验中我使用了三台华为云服务器，操作系统为 `CentOS 7.6`，架构为 `ARM64` 鲲鹏通用计算增强型，截图如下图所示。



配置

基础配置

计费模式	按需计费	区域	北京四	可用区	可用区1
规格	鲲鹏通用计算增强型 I kc1.large.2 I 2vCPUs I 4GiB	镜像	CentOS 7.6 64bit with ARM	主机安全	未开启
系统盘	通用型SSD, 40GiB				

网络配置

虚拟私有云	vpc-default(192.168.0.0/16)	安全组	Sys-WebServer	主网卡	subnet-default(192.168.0.0/24)
弹性公网IP	全动态BGP I 计费方式: 按流量计费 I 带宽: 5 Mbit/s I 随实例释放				

高级配置

云服务器名称	hadoop-server	登录凭证	密码	云服务器组	--

购买数量 3 您最多可以创建200台云服务器。申请更多云服务器配额请单击申请扩大配额。

协议 ☐ 我已经阅读并同意《镜像免责声明》

2.2 所使用代码的说明

根据老师提供的配置文件已经网上的一些教程，我构建了一组脚本。可以自动完成节点互信，`host` 配置，`openjdk` 安装，插件安装，`hadoop` 部署和分发等任务。从而实现了 `Hadoop` 集群的快速配置。代码结构如下：

```
hadoop-setup
├── README.md
├── autohost.sh
├── autossh.sh
├── config
│   ├── core-site.xml
│   ├── hdfs-site.xml
│   ├── mapred-site.xml
│   ├── slaves
│   └── yarn-site.xml
├── distribute.sh
├── main.sh
└── set_env.sh

1 directory, 11 files
```

`config/` 该文件夹存储有配置文件，用于配置 hadoop 和华为的 OBS 插件

`autossh.sh` 在任意一台服务器上运行该脚本，则可以使指定服务器集群之间实现 SSH 互信

`autohost.sh` 配置 host 信息，默认命名为 master, slave1, slave2

`distribute.sh` 将master节点的 hadoop 包分发给 slave1, slave2

`set_env.sh` 配置 master 节点和 slave 节点的环境变量

`main.sh` 主脚本，用以最后执行，完成所有的步骤

2.3 脚本使用方法

2.3.1 添加华为云OBS服务个人配置信息

进入项目目录，打开 `./config/core-site.xml`，找到对应的 `access key`，`secret key` 和 `endpoint` 三个空项上。添加个人在`obs` 购买配置时，所记录的 `access key`，`secret key` 和 `endpoint` 的值。

2.3.2 运行脚本

以如下命令运行脚本，其中 `[IP1] [IP2] [IP3]` 为 `master`，`slave1`，`slave2` 的内网 `IP` 地址，`[PASSWORD1] [PASSWORD2] [PASSWORD3]` 为 `master`，`slave1`，`slave2` 的登录密码，这些信息都可在华为云控制台获得。

```
source main.sh [IP1] [IP2] [IP3] [PASSWORD1] [PASSWORD2] [PASSWORD3]
```

该脚本将自动完成节点互信，`host` 配置，`openjdk` 安装，插件安装，`hadoop` 部署和分发等任务

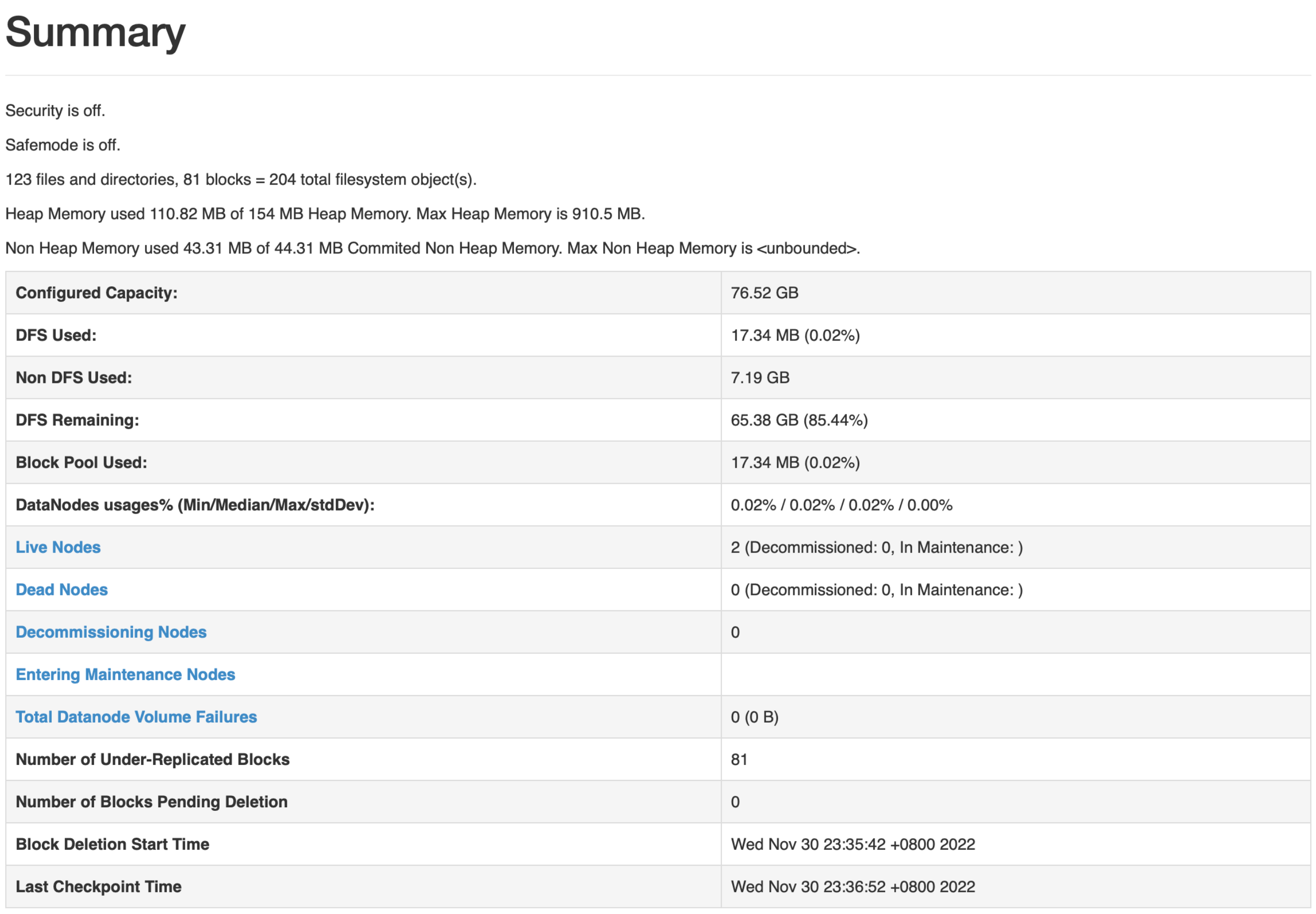
2.3.3 `hosts` 调整

由于 `CentOS` 在每次开机后都会将本机 `hostname` 映射到 `127.0.0.1`，导致 `hadoop` 的端口监听地址为`127.0.0.1`，外部无法访问。故最后我们需要打开每台服务器的 `/etc/hosts` 文件，删掉所有包含`127.0.0.1` 的行。这样就完成了设置。

2.3.4 启动 `Hadoop`

```
hdfs namenode -format
start-all.sh
```

经过测试，通过脚本自动构建的 `Hadoop` 环境运行正常，如下图所示



Summary

Security is off.

Safemode is off.

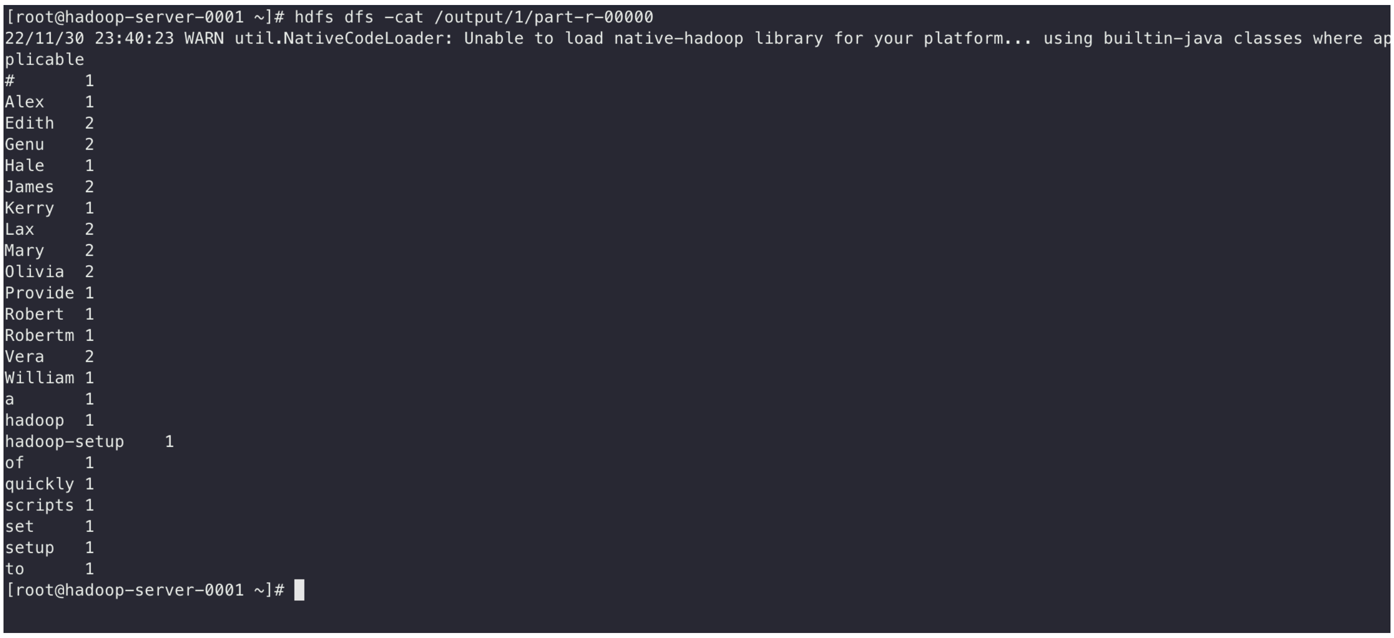
123 files and directories, 81 blocks = 204 total filesystem object(s).

Heap Memory used 110.82 MB of 154 MB Heap Memory. Max Heap Memory is 910.5 MB.

Non Heap Memory used 43.31 MB of 44.31 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	76.52 GB
DFS Used:	17.34 MB (0.02%)
Non DFS Used:	7.19 GB
DFS Remaining:	65.38 GB (85.44%)
Block Pool Used:	17.34 MB (0.02%)
DataNodes usages% (Min/Median/Max/stdDev):	0.02% / 0.02% / 0.02% / 0.00%
Live Nodes	2 (Decommissioned: 0, In Maintenance:)
Dead Nodes	0 (Decommissioned: 0, In Maintenance:)
Decommissioning Nodes	0
Entering Maintenance Nodes	
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	81
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Wed Nov 30 23:35:42 +0800 2022
Last Checkpoint Time	Wed Nov 30 23:36:52 +0800 2022

为了测试华为OBS插件的正常工作，使用该集群环境运行示例程序 `WordCount`，结果如下图所示。



```
[root@hadoop-server-0001 ~]# hdfs dfs -cat /output/1/part-r-00000
22/11/30 23:40:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where ap
plicable
#       1
Alex    1
Edith   2
Genu    2
Hale    1
James   2
Kerry   1
Lax     2
Mary    2
Olivia  2
Provide 1
Robert  1
Robertm 1
Vera    2
William 1
a       1
hadoop  1
hadoop-setup    1
of      1
quickly 1
scripts 1
set     1
setup   1
to      1
[root@hadoop-server-0001 ~]# 
```

3 MapReduce 任务实现及结果展示

3.1 任务名称

奥运会运动员的奖牌统计

3.2 任务目标描述

我们的输入是大量奥运会运动员的信息，在统计运动员的获奖数时，我们并不需要运动员诸如 `id`，`date_of_birth`，`info` 等项的数据，这就需要我们清洗奥运会的统计数据。这时我们的一个做法是使用 `Python` 的 `pandas` 库进行处理，但是使用 `Python` 执行具有速度较慢的缺点，应对这种海量数据时我们就需要使用 `MapReduce` 这种分布式的计算框架进行处理。

3.3 所用数据描述

该任务的输入是一个 `csv` 文档，其中包含大量奥运会运动员的信息，这些信息包含 `id`，`name`，`nationality`，`sex`，`date_of_birth`，`height`，`weight`，`sport`，`gold`，`silver`，`bronze`，`info` 等

3.4 任务实现过程

3.4.1 java代码

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

import java.io.*;
import java.io.IOException;
import java.util.*;
import java.nio.charset.StandardCharsets;



public class Medals
{
    /* input:  <byte_offset, line_of_dataset>
```

```
     * output: <(name,sex,country,sport), (gold#silver#bronze)>
     */
    public static class Map extends Mapper<Object, Text, Text, Text>
    {
        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException
        {
            try
            {
                if(value.toString().contains("nationality")) // remove header
                    return;
                else
                {
                    String record = value.toString();
                    String[] columns = record.split(",(?=([^\"]*\"[^\"]*\")*[^\"]*$)");

                    // extract athlete's main info
                    String name = columns[1];
                    String sex = columns[3];
                    String country = columns[2];
                    String sport = columns[7];

                    // extract athlete's stat info
                    String gold = columns[8];
                    String silver = columns[9];
                    String bronze = columns[10];

                    // set the main info as key and the stat info as value
                    context.write(new Text(name + "," + sex + "," + country + "," + sport),
new Text(gold + "#" + silver + "#" + bronze));
                }
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
    }

    /* input:  <(name,sex,country,sport), (gold#silver#bronze)>
     * output: <(NULL, (name,sex,,country,sport,,golds,silvers,bronzes)>
     */
    public static class Reduce extends Reducer<Text, Text, NullWritable, Text>
    {
        public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException
        {
            // extract athlete's main info
            String[] athlete_info = key.toString().split(",");
            String name = athlete_info[0];
            String sex = athlete_info[1];
            String country = athlete_info[2];
            String sport = athlete_info[3];

            int gold_cnt = 0;
            int silver_cnt = 0;
            int bronze_cnt = 0;

            // for a single athlete, compute their stats...
            for(Text value : values)
            {
                String[] medals = value.toString().split("#");

                gold_cnt += Integer.parseInt(medals[0]);
                silver_cnt += Integer.parseInt(medals[1]);
                bronze_cnt += Integer.parseInt(medals[2]);
            }

            context.write(NullWritable.get(), new Text(name + "," + sex + "," + "," +
country + "," + sport + "," + String.valueOf(gold_cnt) + "," + String.valueOf(silver_cnt) +
"," + String.valueOf(bronze_cnt)));
        }
    }


    public static void main(String[] args) throws Exception
    {
        // set the paths of the input and output directories in the HDFS
        Path input_dir = new Path("olympic_stats");
        Path output_dir = new Path("medals");

        // in case the output directory already exists, delete it
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        if(fs.exists(output_dir))
            fs.delete(output_dir, true);

        // configure the MapReduce job
        Job medals_job = Job.getInstance(conf, "Medals Counter");
        // medals_job.setJarByClass(Medals.class);
        medals_job.setJar("Medals.jar"); // 注意此处很重要，否则在代码运行时因无法识别class文件而报错
        medals_job.setMapperClass(Map.class);
        medals_job.setReducerClass(Reduce.class);
        medals_job.setMapOutputKeyClass(Text.class);
        medals_job.setMapOutputValueClass(Text.class);
        medals_job.setOutputKeyClass(NullWritable.class);
        medals_job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(medals_job, input_dir);
        FileOutputFormat.setOutputPath(medals_job, output_dir);
        medals_job.waitForCompletion(true);
    }
}
```

该代码在 `Map` 阶段将读入的每行 `csv` 文件转为 `<(name,sex,country,sport), (gold,silver,bronze)>` 的键值对，分别交给每个节点去处理。最后在 `Reduce` 阶段再将这些键值对合并为一行仅含有 `(name,sex,,country,sport,,golds,silvers,bronzes)` 值的序列并写入文件。这种分布式处理的方式极大地提高了数据清洗的效率。

3.4.2 配置过程

进入 `./Medals` 目录，运行以下命令编译 `java` 源文件，并将结果打包为 `jar` 包。

```
javac -classpath "$(yarn classpath)" -d . Medals.java
jar -cvf Medals.jar -C . .
```

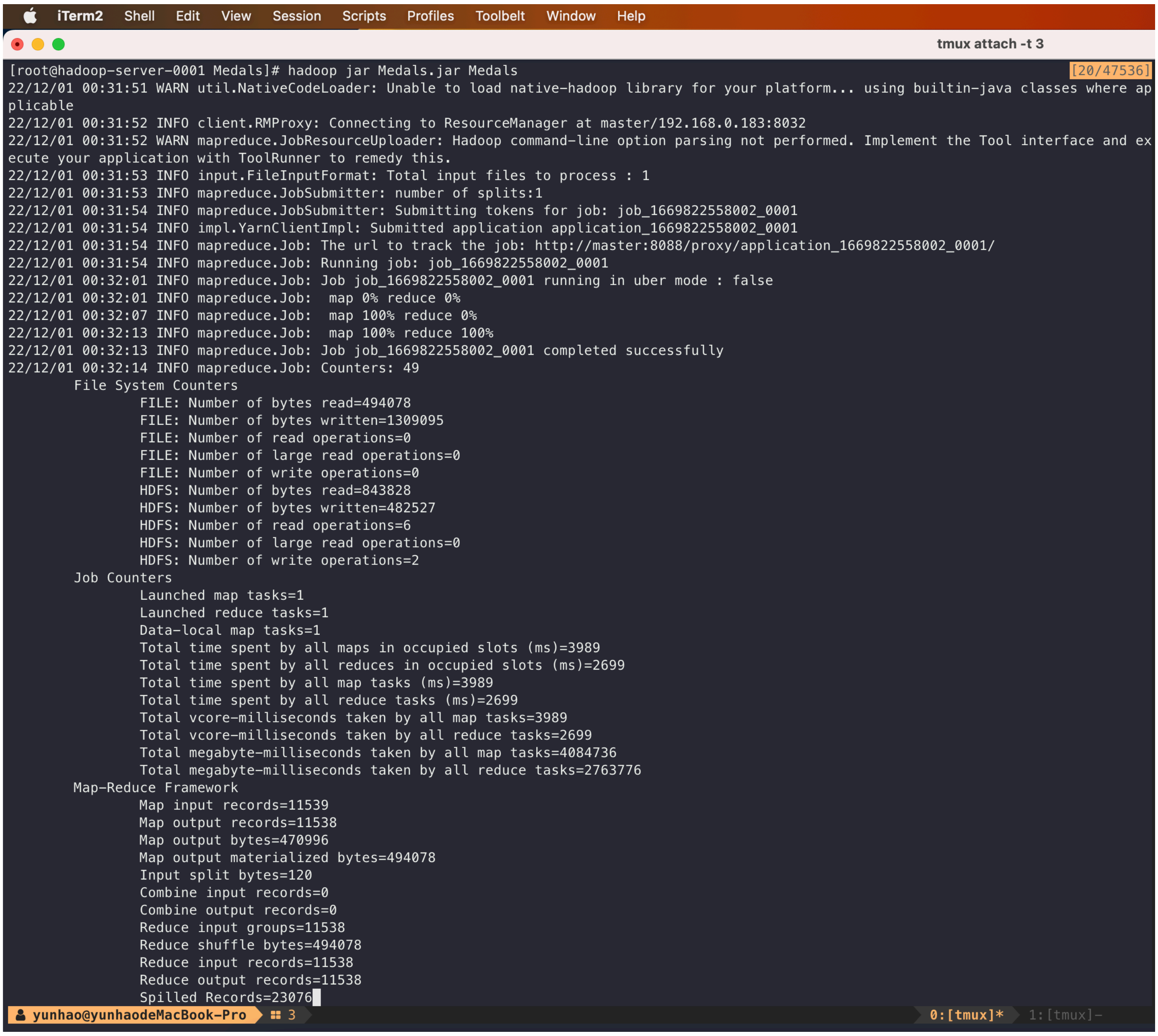
将 `java` 代码中制定的输入目录和输出目录上传到 `hdfs` 上

```
hdfs dfs -mkdir olympic_stats # input dir
hdfs dfs -mkdir medals        # output dir
```

3.4.3 运行结果说明及展示

编译得到了 `Medals.jar` 包之后，我们在 `./Medals` 目录下执行以下命令开启 `MapReduce` 任务，运行截图如下图所示

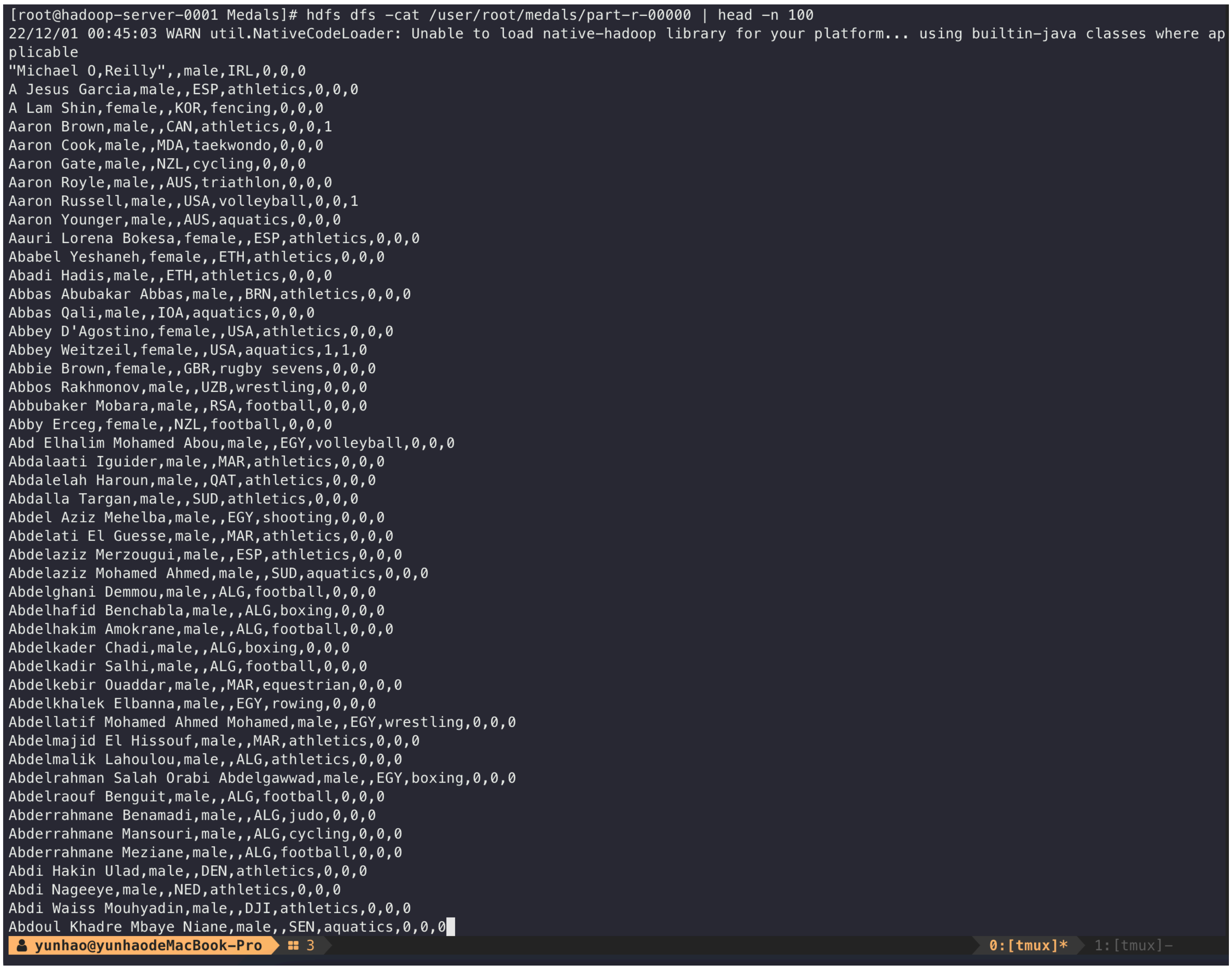
```
hadoop jar Medals.jar Medals
```



```
[root@hadoop-server-0001 Medals]# hadoop jar Medals.jar Medals
22/12/01 00:31:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where ap
plicable
22/12/01 00:31:52 INFO client.RMProxy: Connecting to ResourceManager at master/192.168.0.183:8032
22/12/01 00:31:52 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and ex
ecute your application with ToolRunner to remedy this.
22/12/01 00:31:53 INFO input.FileInputFormat: Total input files to process : 1
22/12/01 00:31:53 INFO mapreduce.JobSubmitter: number of splits:1
22/12/01 00:31:54 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1669822558002_0001
22/12/01 00:31:54 INFO impl.YarnClientImpl: Submitted application application_1669822558002_0001
22/12/01 00:31:54 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1669822558002_0001/
22/12/01 00:31:54 INFO mapreduce.Job: Running job: job_1669822558002_0001
22/12/01 00:32:01 INFO mapreduce.Job: Job job_1669822558002_0001 running in uber mode : false
22/12/01 00:32:01 INFO mapreduce.Job:  map 0% reduce 0%
22/12/01 00:32:07 INFO mapreduce.Job:  map 100% reduce 0%
22/12/01 00:32:13 INFO mapreduce.Job:  map 100% reduce 100%
22/12/01 00:32:13 INFO mapreduce.Job: Job job_1669822558002_0001 completed successfully
22/12/01 00:32:14 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=494078
                FILE: Number of bytes written=1309095
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=843828
                HDFS: Number of bytes written=482527
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3989
                Total time spent by all reduces in occupied slots (ms)=2699
                Total time spent by all map tasks (ms)=3989
                Total time spent by all reduce tasks (ms)=2699
                Total vcore-milliseconds taken by all map tasks=3989
                Total vcore-milliseconds taken by all reduce tasks=2699
                Total megabyte-milliseconds taken by all map tasks=4084736
                Total megabyte-milliseconds taken by all reduce tasks=2763776
        Map-Reduce Framework
                Map input records=11539
                Map output records=11538
                Map output bytes=470996
                Map output materialized bytes=494078
                Input split bytes=120
                Combine input records=0
                Combine output records=0
                Reduce input groups=11538
                Reduce shuffle bytes=494078
                Reduce input records=11538
                Reduce output records=11538
                Spilled Records=23076
```

根据我们的设置，输出目录的位置为 `/user/root/medals/`，故运行以下命令查看输出结果，因为输出数据量太大，故截取前100条数据，结果如下图所示

```
hdfs dfs -cat /user/root/medals/part-r-00000 | head -n 100
```



```
[root@hadoop-server-0001 Medals]# hdfs dfs -cat /user/root/medals/part-r-00000 | head -n 100
22/12/01 00:45:03 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where ap
plicable
"Michael O,Reilly",,male,IRL,0,0,0
A Jesus Garcia,male,,ESP,athletics,0,0,0
A Lam Shin,female,,KOR,fencing,0,0,0
Aaron Brown,male,,CAN,athletics,0,0,1
Aaron Cook,male,,MDA,taekwondo,0,0,0
Aaron Gate,male,,NZL,cycling,0,0,0
Aaron Royle,male,,AUS,triathlon,0,0,0
Aaron Russell,male,,USA,volleyball,0,0,1
Aaron Younger,male,,AUS,aquatics,0,0,0
Aauri Lorena Bokesa,female,,ESP,athletics,0,0,0
Ababel Yeshaneh,female,,ETH,athletics,0,0,0
Abadi Hadis,male,,ETH,athletics,0,0,0
Abbas Abubakar Abbas,male,,BRN,athletics,0,0,0
Abbas Qali,male,,IOA,aquatics,0,0,0
Abbey D'Agostino,female,,USA,athletics,0,0,0
Abbey Weitzeil,female,,USA,aquatics,1,1,0
Abbie Brown,female,,GBR,rugby sevens,0,0,0
Abbos Rakhmonov,male,,UZB,wrestling,0,0,0
Abbubaker Mobara,male,,RSA,football,0,0,0
Abby Erceg,female,,NZL,football,0,0,0
Abd Elhalim Mohamed Abou,male,,EGY,volleyball,0,0,0
Abdalaati Iguider,male,,MAR,athletics,0,0,0
Abdalelah Haroun,male,,QAT,athletics,0,0,0
Abdalla Targan,male,,SUD,athletics,0,0,0
Abdel Aziz Mehelba,male,,EGY,shooting,0,0,0
Abdelati El Guesse,male,,MAR,athletics,0,0,0
Abdelaziz Merzougui,male,,ESP,athletics,0,0,0
Abdelaziz Mohamed Ahmed,male,,SUD,aquatics,0,0,0
Abdelghani Demmou,male,,ALG,football,0,0,0
Abdelhafid Benchabla,male,,ALG,boxing,0,0,0
Abdelhakim Amokrane,male,,ALG,football,0,0,0
Abdelkader Chadi,male,,ALG,boxing,0,0,0
Abdelkadir Salhi,male,,ALG,football,0,0,0
Abdelkebir Ouaddar,male,,MAR,equestrian,0,0,0
Abdelkhalek Elbanna,male,,EGY,rowing,0,0,0
Abdellatif Mohamed Ahmed Mohamed,male,,EGY,wrestling,0,0,0
Abdelmajid El Hissouf,male,,MAR,athletics,0,0,0
Abdelmalik Lahoulou,male,,ALG,athletics,0,0,0
Abdelrahman Salah Orabi Abdelgawwad,male,,EGY,boxing,0,0,0
Abdelraouf Benguit,male,,ALG,football,0,0,0
Abderrahmane Benamadi,male,,ALG,judo,0,0,0
Abderrahmane Mansouri,male,,ALG,cycling,0,0,0
Abderrahmane Meziane,male,,ALG,football,0,0,0
Abdi Hakin Ulad,male,,DEN,athletics,0,0,0
Abdi Nageeye,male,,NED,athletics,0,0,0
Abdi Waiss Mouhyadin,male,,DJI,athletics,0,0,0
Abdoul Khadre Mbaye Niane,male,,SEN,aquatics,0,0,0
```

可以看到，使用 Hadoop 集群和 MapReduce 框架较好地完成了本次数据清洗的任务，且运行效率很高，

4 作业总结

在本次实验中，我主要完成了编写脚本自动化搭建 Hadoop 集群和基于 Hadoop 实现了一个简单的基于 MapReduce 的数据清洗任务，我的收获主要有以下几点。

- 对 Linux 命令有了更深的了解，在编写 shell 脚本的时候，我解决了很多遇到的问题。包括
  - 配置所有指定服务器集群之间的 SSH 互信
  - 使用 tar 命令将文件夹压缩和解压到不同的目录
  - 使用 ssh 命令在 slave 节点上远程执行 master 节点的脚本
  - 使用 echo 命令将配置文件写入 /etc/profile 和 /etc/hosts
  - 对每台服务器的 /etc/hosts 进行调整以防止外部无法访问。
  - 等等
- 对 Hadoop 的结构以及 MapReduce 的工作原理有了更加深刻的了解。通过一个给定的情景编写 java 程序，我体会到了 MapReduce 作为一个分布式计算软件框架的重要思想：将数据Map为一个键/值对的集合，然后对所有键/值对按照相同键值进行Reduce，这种分布式的思想在大数据分析中很重要。

5 参考资料

[1] How to Install Hadoop in Stand-Alone Mode on CentOS 7

[2] 自动配置SSH互信脚本

[3] There are 0 datanode(s) running and no node(s) are excluded in this operation

[4] Hadoop-Examples

[5] 解决Hadoop命令方式运行WordCount异常Class WordCount$XXXMapper not found