



東南大學
SOUTHEAST UNIVERSITY

云计算技术及应用 课程实践报告

| | |
|-----|----------|
| 标 题 | 并行计算技术实践 |
| 学 号 | 232349 |
| 姓 名 | 李志成 |

东南大学计算机科学与工程学院

二零二四年三月

目录

| | |
|--------------------------------------|----|
| 实验二 并行计算技术实践 | 1 |
| 1 实验目的及要求 | 1 |
| 1.1 实验目的 | 1 |
| 1.2 实验要求 | 1 |
| 2 实验环境安装与配置 | 1 |
| 2.1 Hadoop 本体安装 | 1 |
| 2.1.1 安装 java | 1 |
| 2.1.2 创建 ssh 密钥并添加到自己的本地计算机 | 1 |
| 2.1.3 下载并解压 Hadoop | 2 |
| 2.2 Hadoop 配置 | 2 |
| 2.2.1 配置环境变量 | 2 |
| 2.2.2 配置 Hadoop 基础框架 | 2 |
| 2.2.3 核心变量配置 | 3 |
| 2.2.4 配置 HDFS (Hadoop 文件系统) | 3 |
| 2.2.5 配置 MapReduce (由 YARN 驱动) | 4 |
| 2.2.6 启动 Hadoop 集群 | 5 |
| 2.2.7 关闭 Hadoop 集群 | 6 |
| 3 WordCount 实验 | 6 |
| 3.1 安装 maven | 6 |
| 3.2 创建项目 | 7 |
| 3.3 编辑 pom.xml | 7 |
| 3.4 在 HDFS 中创建一个文本文件 | 10 |
| 3.5 使用 Hadoop 运行编译的 jar 文件 | 11 |
| 4 KMeans 实验 | 12 |
| 4.1 准备相关文件 | 12 |
| 4.2 构建项目 | 13 |
| 4.3 运行项目 | 14 |
| 4.4 检查结果 | 15 |
| 5 总结 | 15 |

1 实验目的及要求

1.1 实验目的

本实验旨在通过实践操作加深对 Hadoop 框架及其组件的理解，掌握 Hadoop 环境的搭建与配置、基于 Hadoop 的应用开发流程，以及在分布式环境下运行 MapReduce 作业的方法。通过完成 WordCount 和 KMeans 两个典型的 MapReduce 编程任务，进一步理解 MapReduce 编程模型的工作机制以及在大数据处理中的应用。

1.2 实验要求

- 1) 熟练掌握 Hadoop 的安装与配置过程，包括 Java 环境的配置、SSH 密钥的生成与配置、Hadoop 各基础组件的配置等。
- 2) 能够在配置好的 Hadoop 环境中，独立完成 MapReduce 程序的编写、打包、部署与运行。
- 3) 对 MapReduce 程序的运行过程有清晰的理解，包括输入数据的准备、作业的提交、作业运行的监控、输出结果的验证等。
- 4) 能够分析和解决在 Hadoop 环境搭建和 MapReduce 程序运行过程中遇到的常见问题。

2 实验环境安装与配置

2.1 Hadoop 本体安装

2.1.1 安装 java

Hadoop 基础组件及其外围项目大多是基于 Java 的，要运行 Hadoop，我们需要在系统中安装 Java 运行时环境（JRE）。有多种 JRE 可供选择，我们使用 openjdk-11 作为 JRE。Hadoop 官方文档中说明了对 Java 版本的支持情况。Hadoop 框架使用 SSH 协议与本地（或远程计算机）进行通信，我们需要在本地计算机中安装 SSH 服务器守护程序。

```
sudo apt install openjdk-11-jdk openssh-server -y
```

2.1.2 创建 ssh 密钥并添加到自己的本地计算机

Hadoop 通常运行于分布式环境中。在这种环境下，它使用 SSH 协议与自己（或其他服务器）进行通信，为了能够配置 Hadoop 更安全地与服务器进行通信，我们为本地环境 Hadoop 用户生成公-私密钥对，以支持 SSH 中的密钥对认证：

```
ssh-keygen -b 4096 -C "for hadoop use" # 为了方便这里可以直接一路回车  
ssh-copy-id -p 22 localhost # 输入 yes
```

上述命令生成带备注信息的长度为 4096 位的 RSA 密钥，并将密钥复制至本地计算机中，用于支持 SSH 密钥对认证方式。

这时候可以尝试使用 `ssh localhost`，应该可以无密码登录。

```
lzc@lzc-virtual-machine:~$ ssh -p '22' 'localhost'
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-25-generic x86_64)

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

扩展安全维护 (ESM) Applications 未启用。

0 更新可以立即应用。

启用 ESM Apps 来获取未来的额外安全更新
请参见 https://ubuntu.com/esm 或者运行: sudo pro status
```

2.1.3 下载并解压 Hadoop

Hadoop 的各种发行版本通常从 Hadoop 官方网站的下载页面(opens new window)下载。我们使用来自 Hadoop 官方网站的发行版本 3.3.6。我们准备将相关发行版本安装至文件系统的 /opt 目录下。

首先从下载页面获得 Hadoop 的发行版文件：（利用清华源的镜像，速度更快）

```
wget
"https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz"
```

下载好压缩包后，解压到/opt 目录。

```
sudo tar -zxvf hadoop-3.3.6.tar.gz -C /opt
```

2.2 Hadoop 配置

Hadoop 并非是单个软件，而是一系列大数据存储及处理的工具集合。所以，有必要针对常用的基础组件进行配置，以完成 Hadoop 环境的搭建。

2.2.1 配置环境变量

安装 Hadoop 后，我们需要正确设置以下环境变量。可以将这些添加环境变量的命令添加至 ~/.bashrc 文件末尾，以避免每次登录时都需要执行，注意以下命令需要一次性复制和运行，不可以分多次：

```
cat << 'EOF' >> ~/.bashrc
export HADOOP_HOME=/opt/hadoop-3.3.6
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$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 PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
EOF
```

然后执行 source ~/.bashrc 使配置生效或者直接重启 shell。

2.2.2 配置 Hadoop 基础框架

顺利运行 Hadoop 组件，我们有必要告诉 Hadoop 有关 Java 运行时环境（JRE）的位置信息。Hadoop 框架的基础配置文件位于 /etc/hadoop/hadoop-env.sh 中，我们只需要修改其中的 JAVA_HOME 行。如果你使用其他方式（比如非标准端口）通过 SSH 客户端连接到（本地）

服务器，则需要额外修改 HADOOP_SSH_OPTS 行。

首先我们接下来的操作都需要以/opt/hadoop-3.3.6 作为当前路径，也就是说需要先 cd 到 /opt/hadoop-3.3.6，另外注意以下命令同样需要一次性复制和运行。

(接下来会经常用到/etc/hadoop 这个目录，注意这个是相对于/opt/hadoop-3.3.6 的，而不是系统根目录下的/etc/hadoop，前者有一个.，表示相对路径。)

```
cat << 'EOF' >> ./etc/hadoop/hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_SSH_OPTS="-p 22"
EOF
```

2.2.3 核心变量配置

Hadoop 生态的大多数组件采用.xml(opens new window)文件的方式进行配置。它们采用了统一的格式：即将配置项填写在每个配置文件的<configuration>与</configuration>之间，每个配置项通过以下的方式呈现：

```
<property>
<name>配置项名称</name>
<value>配置值</value>
</property>
```

例如，一个正常的 hadoop 配置文件应该类似如下结构，下文所说的放在<configuration> </configuration> 之间都是如此：

```
<configuration>

<property>
<name>配置项名称 1</name>
<value>配置值 1</value>
</property>

<property>
<name>配置项名称 2</name>
<value>配置值 2</value>
</property>

</configuration>
```

首先需要修改./etc/hadoop/core-site.xml，我们需要配置 fs.defaultFS 为 hdfs://localhost/，也就是说，需要把以下内容添加到./etc/hadoop/core-site.xml 的<configuration> </configuration>之间。

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

2.2.4 配置 HDFS（Hadoop 文件系统）

为了支持 Hadoop 的运行，我们需要使用能支持分布式的文件系统，HDFS 就是为了 Hadoop

使用的。

在使用 Hadoop 文件系统（HDFS）之前，我们需要显式配置 HDFS，以指定 NameNode 与 DataNode 的存储位置。我们计划将 NameNode 与 DataNode 存储于本地文件系统上，即存放于~/hdfs 中：

```
mkdir -p ~/hdfs/namenode ~/hdfs/datanode
```

然后，修改 ./etc/hadoop/hdfs-site.xml，将以下内容放到该文件的 <configuration> </configuration> 之间。

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>

<property>
<name>dfs.name.dir</name>
<value>/home/lzc/hadoop/namenode</value>
</property>

<property>
<name>dfs.data.dir</name>
<value>/home/lzc/hadoop/datanode</value>
</property>
```

首次启动 Hadoop 环境之前，我们需要初始化 Namenode 节点数据，使用以下命令：

```
hdfs namenode -format
```

应该可以看到很多输出信息。

2.2.5 配置 MapReduce（由 YARN 驱动）

MapReduce 是一种简单的用于数据处理的编程模型，YARN(Yet Another Resource Negotiator) 是 Hadoop 的集群资源管理系统。

将以下内容放到 ./etc/hadoop/mapred-site.xml 的 <configuration> </configuration> 之间。

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

<property>
<name>yarn.app.mapreduce.am.env</name>
<value>HADOOP_MAPRED_HOME=/opt/hadoop-3.3.6</value>
</property>

<property>
<name>mapreduce.map.env</name>
<value>HADOOP_MAPRED_HOME=/opt/hadoop-3.3.6</value>
</property>
```

```
<property>
<name>mapreduce.reduce.env</name>
<value>HADOOP_MAPRED_HOME=/opt/hadoop-3.3.6/</value>
</property>
```

将以下内容放到./etc/hadoop/yarn-site.xml 的<configuration> </configuration>之间。

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

<property>
<name>yarn.resourcemanager.hostname</name>
<value>localhost</value>
</property>
```

2.2.6 启动 Hadoop 集群

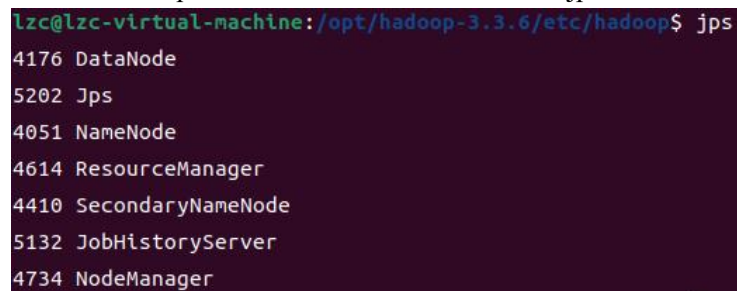
使用以下命令：

```
start-dfs.sh
start-yarn.sh
mr-jobhistory-daemon.sh start historyserver
```

可能会出现 ssh 的提示 Are you sure you want to continue connecting (yes/no/[fingerprint])?, 直接 yes 就可以。

这将启动以下守护进程：一个 namenode、一个辅助 namenode、一个 datanode（HDFS）、一个资源管理器、一个节点管理器（YARN）以及一个历史服务器（MapReduce）。

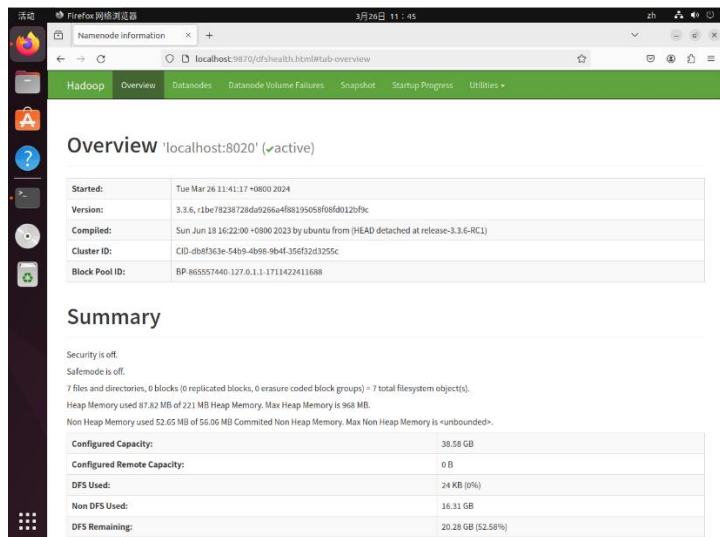
为验证 Hadoop 相关服务启动成功，可以使用 jps 命令。



```
lzc@lzc-virtual-machine:/opt/hadoop-3.3.6/etc/hadoop$ jps
4176 DataNode
5202 Jps
4051 NameNode
4614 ResourceManager
4410 SecondaryNameNode
5132 JobHistoryServer
4734 NodeManager
```

jps 命令显示的内容与上述预期一致。

可以使用浏览器访问该主机的 9870 端口，以查看 HFS 的相关情况：



此时 hadoop 基础的组件已经安装完成。

2.2.7 关闭 Hadoop 集群

使用以下命令：

```
stop-yarn.sh
stop-dfs.sh
mr-jobhistory-daemon.sh stop historyserver
```

```
lzc@lzc-virtual-machine:/opt/hadoop-3.3.6/etc/hadoop$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
lzc@lzc-virtual-machine:/opt/hadoop-3.3.6/etc/hadoop$ stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [lzc-virtual-machine]
lzc@lzc-virtual-machine:/opt/hadoop-3.3.6/etc/hadoop$ mr-jobhistory-daemon.sh stop historyserver
WARNING: Use of this script to stop the MR JobHistory daemon is deprecated.
WARNING: Attempting to execute replacement "mapred --daemon stop" instead.
lzc@lzc-virtual-machine:/opt/hadoop-3.3.6/etc/hadoop$ jps
6777 Jps
```

可以看到，我们成功关闭了 Hadoop 集群服务。

3 WordCount 实验

3.1 安装 maven

首先使用命令 `sudo apt-get install maven`。

将配置文件复制到 `~/m2` 目录：`mkdir ~/m2 && cp /etc/maven/settings.xml ~/m2`。

编辑 `~/m2/settings.xml` 文件，将以下内容添加到 `<mirrors></mirrors>` 中间。

```
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
  <url>https://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
```


3.2 创建项目

使用以下命令，这将在当前文件夹创建一个 workcount 文件夹。

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes
-DgroupId=cn.edu.seu.lzc -DartifactId=wordcount -DpackageName=cn.edu.seu.lzc
-Dversion=1.0-SNAPSHOT -DinteractiveMode=false
```

```
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: basedir, Value: /home/lzc
[INFO] Parameter: package, Value: cn.edu.seu.lzc
[INFO] Parameter: groupId, Value: cn.edu.seu.lzc
[INFO] Parameter: artifactId, Value: wordcount
[INFO] Parameter: packageName, Value: cn.edu.seu.lzc
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /home/lzc/wordcount
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 47.309 s
[INFO] Finished at: 2024-03-26T14:39:49+08:00
[INFO] -----
```

终端显示项目创建成功！

3.3 编辑 pom.xml

进入 workcount 文件夹，编辑 pom.xml，在<dependencies></dependencies>之间添加如下内容：

```
<dependency>
  <groupId>commons-cli</groupId>
  <artifactId>commons-cli</artifactId>
  <version>1.2</version>
</dependency>

<dependency>
  <groupId>commons-logging</groupId>
  <artifactId>commons-logging</artifactId>
  <version>1.2</version>
</dependency>

<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-mapreduce-client-jobclient</artifactId>
  <version>2.8.5</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-common</artifactId>
  <version>2.8.5</version>
  <scope>provided</scope>
```

```
</dependency>
```

在</dependencies>后面增加如下内容:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.7</source>
        <target>1.7</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

使用 mvn clean install -DskipTests 同步依赖。

```
[INFO] Installing /home/lzc/wordcount/target/wordcount-1.0-SNAPSHOT.jar to /home/lzc/.m2/repository/cn/edu/seu/lzc/wordcount/1.0-SNAPSHOT/wordcount-1.0-SNAPSHOT.jar
[INFO] Installing /home/lzc/wordcount/pom.xml to /home/lzc/.m2/repository/cn/edu/seu/lzc/wordcount/1.0-SNAPSHOT/wordcount-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:02 min
[INFO] Finished at: 2024-03-26T15:12:48+08:00
[INFO] -----
```

依赖同步成功!

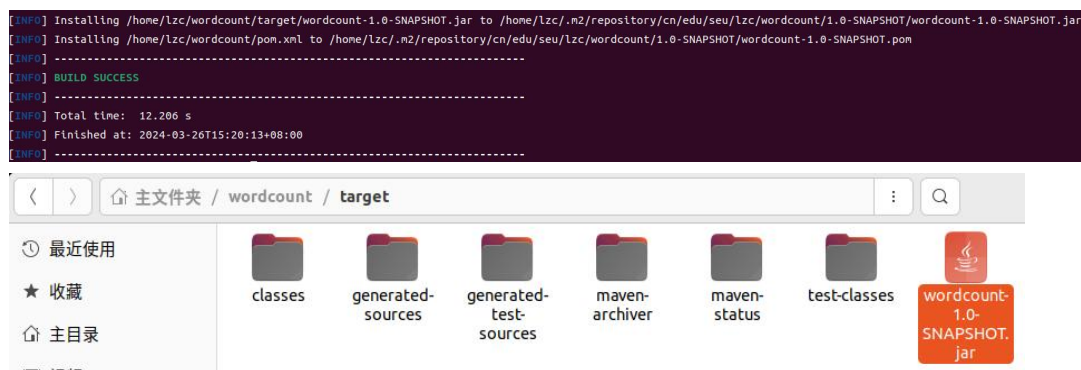
新建 src/main/java/cn/edu/seu/lzc/WordCount.java, 内容如下:

```
1. package cn.edu.seu.lzc;
2.
3. import java.io.IOException;
4. import java.util.StringTokenizer;
5.
6. import org.apache.hadoop.conf.Configuration;
7. import org.apache.hadoop.fs.Path;
8. import org.apache.hadoop.io.IntWritable;
9. import org.apache.hadoop.io.Text;
10. import org.apache.hadoop.mapreduce.Job;
11. import org.apache.hadoop.mapreduce.Mapper;
12. import org.apache.hadoop.mapreduce.Reducer;
13. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15. import org.apache.hadoop.util.GenericOptionsParser;
16.
17. public class WordCount {
18.     public static class TokenizerMapper
19.         extends Mapper<Object, Text, Text, IntWritable>{
20.
21.         private final static IntWritable one = new IntWritable(1);
22.         private Text word = new Text();
```

```
23.
24.     public void map(Object key, Text value, Context context
25.     ) throws IOException, InterruptedException {
26.         StringTokenizer itr = new StringTokenizer(value.toString()
27.         );
28.         while (itr.hasMoreTokens()) {
29.             word.set(itr.nextToken());
30.             context.write(word, one);
31.         }
32.     }
33.
34.     public static class IntSumReducer
35.         extends Reducer<Text,IntWritable,Text,IntWritable> {
36.         private IntWritable result = new IntWritable();
37.         public void reduce(Text key, Iterable<IntWritable> values,
38.             Context context
39.         ) throws IOException, InterruptedException {
40.             int sum = 0;
41.             for (IntWritable val : values) {
42.                 sum += val.get();
43.             }
44.             result.set(sum);
45.             context.write(key, result);
46.         }
47.     }
48.
49.     public static void main(String[] args) throws Exception {
50.         Configuration conf = new Configuration();
51.         String[] otherArgs = new GenericOptionsParser(conf, args).get
52.         RemainingArgs();
53.         if (otherArgs.length < 2) {
54.             System.err.println("Usage: wordcount <in> [<in>...] <out>
55.             ");
56.             System.exit(2);
57.         }
58.         Job job = Job.getInstance(conf, "word count");
59.         job.setJarByClass(WordCount.class);
60.         job.setMapperClass(TokenizerMapper.class);
61.         job.setCombinerClass(IntSumReducer.class);
62.         job.setReducerClass(IntSumReducer.class);
63.         job.setOutputKeyClass(Text.class);
64.         job.setOutputValueClass(IntWritable.class);
65.         for (int i = 0; i < otherArgs.length - 1; ++i) {
```

```
64.         FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
65.     }
66.     FileOutputFormat.setOutputPath(job,
67.         new Path(otherArgs[otherArgs.length - 1]));
68.     System.exit(job.waitForCompletion(true) ? 0 : 1);
69. }
70. }
```

然后使用命令 `mvn clean install -DskipTests` 打包，可以看到当前目录的 `target` 文件夹下多了一个 `wordcount-1.0-SNAPSHOT.jar` 文件。



3.4 在 HDFS 中创建一个文本文件

```
hdfs dfs -mkdir /miao # 创建文件夹
hdfs dfs -ls /
```

可以在终端看到:

```
lzc@lzc-virtual-machine:~$ hdfs dfs -mkdir /miao
lzc@lzc-virtual-machine:~$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x - lzc supergroup      0 2024-03-26 21:24 /miao
drwxrwx--- - lzc supergroup      0 2024-03-26 11:42 /tmp
```

在本地创建一个 `miao.txt`，随便写一些内容，实验手册中给出的文本内容如下：

Long long ago, there is one cat that is expected at using the cat utility on Linux. One day, it saw a spring in the spring in spring. He is astonished and laughing to death.

Years later, people mark the 9 May as the Dying Cat Linux Day to celebrate the people who is struggling at using Linux.

Written by a people who loves cat.

Please add your own student id and name here:

1845** Zhang Shan

然后把这个文件放到 `hdfs` 中去：

```
hdfs dfs -put miao.txt /miao/  
hdfs dfs -ls /miao
```

可以在终端看到:

```
lzc@lzc-virtual-machine:~$ hdfs dfs -put miao.txt /miao/  
lzc@lzc-virtual-machine:~$ hdfs dfs -ls /miao  
Found 1 items  
-rw-r--r--  1 lzc supergroup      396 2024-03-26 21:28 /miao/miao.txt
```

3.5 使用 Hadoop 运行编译的 jar 文件

在终端输入命令:

```
hadoop jar ~/wordcount/target/wordcount-1.0-SNAPSHOT.jar cn.edu.seu.lzc.WordCount  
/miao/miao.txt /miao/output/
```

终端显示:

```
lzc@lzc-virtual-machine:~$ hadoop jar ~/wordcount/target/wordcount-1.0-SNAPSHOT.jar cn.edu.seu.lzc.WordCount /miao/miao.txt /miao/output/  
2024-03-26 21:33:28,423 INFO client.DefaultNoHARMFalloverProxyProvider: Connecting to ResourceManager at localhost/127.0.0.1:8032  
2024-03-26 21:33:28,718 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/lzc/.staging/job_1711459400592_0001  
2024-03-26 21:33:28,941 INFO input.FileInputFormat: Total input files to process : 1  
2024-03-26 21:33:29,013 INFO mapreduce.JobSubmitter: number of splits:1  
2024-03-26 21:33:29,174 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1711459400592_0001  
2024-03-26 21:33:29,174 INFO mapreduce.JobSubmitter: Executing with tokens: []  
2024-03-26 21:33:29,303 INFO conf.Configuration: resource-types.xml not found  
2024-03-26 21:33:29,303 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.  
2024-03-26 21:33:29,467 INFO impl.YarnClientImpl: Submitted application application_1711459400592_0001  
2024-03-26 21:33:29,503 INFO mapreduce.Job: The url to track the job: http://lzc-virtual-machine:8088/proxy/application_1711459400592_0001/  
2024-03-26 21:33:29,504 INFO mapreduce.Job: Running job: job_1711459400592_0001  
2024-03-26 21:33:37,622 INFO mapreduce.Job: Job job_1711459400592_0001 running in uber mode : false  
2024-03-26 21:33:37,625 INFO mapreduce.Job:  map 0% reduce 0%  
2024-03-26 21:33:42,674 INFO mapreduce.Job:  map 100% reduce 0%  
2024-03-26 21:33:48,703 INFO mapreduce.Job:  map 100% reduce 100%  
2024-03-26 21:33:48,709 INFO mapreduce.Job: Job job_1711459400592_0001 completed successfully  
2024-03-26 21:33:48,787 INFO mapreduce.Job: Counters: 54
```

```
File System Counters  
  FILE: Number of bytes read=665  
  FILE: Number of bytes written=553755  
  FILE: Number of read operations=0  
  FILE: Number of large read operations=0  
  FILE: Number of write operations=0  
  HDFS: Number of bytes read=491  
  HDFS: Number of bytes written=427  
  HDFS: Number of read operations=8  
  HDFS: Number of large read operations=0  
  HDFS: Number of write operations=2  
  HDFS: Number of bytes read erasure-coded=0  
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)=2964  
  Total time spent by all reduces in occupied slots (ms)=2874  
  Total time spent by all map tasks (ms)=2964  
  Total time spent by all reduce tasks (ms)=2874  
  Total vcore-milliseconds taken by all map tasks=2964  
  Total vcore-milliseconds taken by all reduce tasks=2874  
  Total megabyte-milliseconds taken by all map tasks=3035136  
  Total megabyte-milliseconds taken by all reduce tasks=2942976
```

```
Map-Reduce Framework
  Map input records=8
  Map output records=77
  Map output bytes=701
  Map output materialized bytes=665
  Input split bytes=95
  Combine input records=77
  Combine output records=58
  Reduce input groups=58
  Reduce shuffle bytes=665
  Reduce input records=58
  Reduce output records=58
  Spilled Records=116
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=42
  CPU time spent (ms)=1460
  Physical memory (bytes) snapshot=475193344
  Virtual memory (bytes) snapshot=5515620352
  Total committed heap usage (bytes)=298455552
  Peak Map Physical memory (bytes)=274767072
  Peak Map Virtual memory (bytes)=2750410752
  Peak Reduce Physical memory (bytes)=200425472
  Peak Reduce Virtual memory (bytes)=2765289600
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=396
File Output Format Counters
  Bytes Written=427
```

等待大概 1 分钟之后运行完毕,使用 `hdfs dfs -ls /miao/output` 命令可以看到有一个 `_SUCCESS` 文件和一个 `part-r-00000` 文件。

```
lzc@lzc-virtual-machine:~$ hdfs dfs -ls /miao/output
Found 2 items
-rw-r--r--  1 lzc supergroup          0 2024-03-26 21:33 /miao/output/_SUCCESS
-rw-r--r--  1 lzc supergroup       427 2024-03-26 21:33 /miao/output/part-r-00000
```

使用命令 `hdfs dfs -cat /miao/output/part-r-00000` 可以看到运行的结果:

```
lzc@lzc-virtual-machine:~$ hdfs dfs -cat /miao/output/part-r-00000
1845**  1
9       1
Cat     1
Day     1
Dying   1
He      1
Linux   1
Linux.  2
Long    1
May     1
```

与预期效果一致, 成功完成 WordCount 实验。

4 KMeans 实验

4.1 准备相关文件

首先解压 KMeans.zip:

```
unzip KMeans.zip
cd KMeans/
```

使用 `java -jar ProcessCorpus.jar` 将文本文件转化成词向量, 该程序会要求输入一些信息, 可以按照以下内容输入:

Enter the directory where the corpus is located: 20_newsgroups

Enter the name of the file to write the result to: vectors

Enter the max number of docs to use in each subdirectory: 100

How many of those words do you want to use to process the docs? 10000

```
lzc@lzc-virtual-machine:~/KMeans$ java -jar ProcessCorpus.jar
Enter the directory where the corpus is located: 20_newsgroups
Enter the name of the file to write the result to: vectors
Enter the max number of docs to use in each subdirectory: 100
20_newsgroups
Counting the number of occurs of each word in the corpus...Found 39987 unique words in the corpus.
How many of those words do you want to use to process the docs? 10000
Done creating the dictionary.
Converting the corpus to a list of vectors...Done vectorizing all of the docs!
```

使用 java -jar GetCentroids.jar 初始化集群起始点，该程序会要求输入一些信息，可以按照以下内容输入：

Enter the data file to select the clusters from: vectors

Enter the name of the file to write the result to: clusters

Enter the number of clusters to select: 20

```
lzc@lzc-virtual-machine:~/KMeans$ java -jar GetCentroids.jar
Enter the data file to select the clusters from: vectors
Enter the name of the file to write the result to: clusters
Enter the number of clusters to select: 20
..Done selecting centroids.
```

然后把刚刚创建的文件复制到 HDFS 中：

```
hdfs dfs -mkdir /data
hdfs dfs -mkdir /clusters
hdfs dfs -copyFromLocal vectors /data
hdfs dfs -copyFromLocal clusters /clusters
```

4.2 构建项目

首先使用如下命令创建项目：

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes
-DgroupId=cn.edu.seu.lzc -DartifactId=kmeans -DpackageName=cn.edu.seu.lzc
-Dversion=1.0-SNAPSHOT -DinteractiveMode=false
```

终端显示：

```

lzc@lzc-virtual-machine:~$ mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DgroupId=cn.edu.seu.lzc
-DartifactId=kmeans -DpackageName=cn.edu.seu.lzc -Dversion=1.0-SNAPSHOT -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO]
[INFO] >>> maven-archetype-plugin:3.2.1:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.2.1:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:3.2.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[WARNING] No archetype found in remote catalog. Defaulting to internal catalog
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven-archetype-quickstart:1.0)
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: basedir, Value: /home/lzc
[INFO] Parameter: package, Value: cn.edu.seu.lzc
[INFO] Parameter: groupId, Value: cn.edu.seu.lzc
[INFO] Parameter: artifactId, Value: kmeans
[INFO] Parameter: packageName, Value: cn.edu.seu.lzc
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /home/lzc/kmeans
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.385 s
[INFO] Finished at: 2024-03-26T22:21:13+08:00
[INFO] -----

```

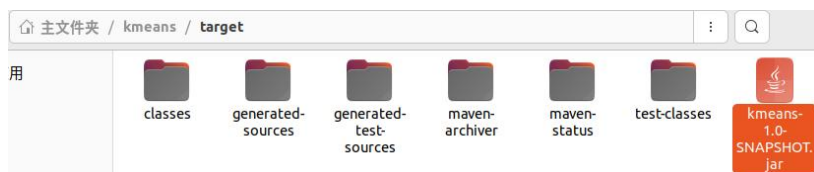
项目构建成功！

然后编辑 pom.xml，操作方法同 WordCount 实验中的 pom.xml。

再然后，把 KMeans 文件夹中，MapRedKMeans 下的所有文件复制到 src/main/java/cn/edu/seu/lzc/：

```
cp MapRedKMeans/* ../kmeans/src/main/java/cn/edu/seu/lzc/
```

然后使用 mvn clean install -DskipTests 构建项目，应该可以看到当前目录的 target 文件夹下多了一个 kmeans-1.0-SNAPSHOT.jar 文件。



4.3 运行项目

使用命令提交到 Hadoop 运行：

```
hadoop jar target/kmeans-1.0-SNAPSHOT.jar KMeans /data /clusters 3
```

其中 3 是 KMeans 的迭代次数，可以自己定义，次数越多需要的时间可能也越久。

执行完毕后，使用命令将结果从 HDFS 拷贝回本地，注意需要先 cd 到解压出来的 KMeans 文件夹，方便后续检查结果：

```
cd KMeans
```



```
hdfs dfs -copyToLocal /clusters3/part-r-00000
```

4.4 检查结果

使用命令 `java -jar GetDistribution.jar` 运行检查结果的 jar 文件。按照如下信息输入：

Enter the file with the data vectors: vectors

Enter the name of the file where the clusters are loaded: part-r-00000

终端显示信息：

```
lzc@lzc-virtual-machine:~/KMean$ java -jar GetDistribution.jar
Enter the file with the data vectors: vectors
Enter the name of the file where the clusters are loaded: part-r-00000
..Done with pass thru data.
***** cluster0 *****

***** cluster1 ***** rec.sport.hockey: 10; comp.os.ms-windows.misc: 7; comp.sys.mac.hardware: 6; comp.sys.ibm.pc.hardware: 5; sci.electronics: 5; rec.sport.baseball: 3; rec.motorcycles: 3; talk.religion.misc: 2; sci.med: 2; talk.politics.mideast: 1; comp.graphics: 1; sci.crypt: 1; misc.forsale: 1; sci.space: 1; talk.politics.guns: 1;

***** cluster10 ***** misc.forsale: 15; comp.sys.mac.hardware: 6; talk.politics.mideast: 5; comp.windows.x: 4; rec.sport.baseball: 4; comp.graphics: 4; sci.electronics: 4; talk.religion.misc: 2; rec.motorcycles: 2; talk.politics.guns: 2; rec.autos: 1; comp.sys.ibm.pc.hardware: 1; sci.med: 1; sci.crypt: 1; rec.sport.hockey: 1;
```

成功完成了 KMeans 实验！

5 总结

通过本学期的《云计算技术及应用》课程，我深刻体会到并行计算技术在处理大规模数据集时的强大能力。Hadoop 作为一个开源的分布式存储与计算平台，不仅提供了一个可靠、可扩展的基础架构，也通过 MapReduce 等编程模型简化了并行计算的复杂性。

在实验过程中，我从零开始，亲手搭建了 Hadoop 环境，并通过完成 WordCount 和 KMeans 两个典型案例，加深了对 MapReduce 编程模型的理解，体验了从数据准备、作业编写、调试到最终执行的全过程。这些经历不仅让我掌握了并行计算技术的应用，也激发了我探索更多大数据处理技术的兴趣。

最重要的是，这段经历让我明白，在面对海量数据的处理时，传统的单机计算方式已远远不能满足需求，而分布式计算、并行处理将是解决这一挑战的关键。未来，我希望能将在本课程中学到的知识和技能，应用到更广阔的领域中，解决更多实际问题。同时，我也将继续学习和探索云计算和大数据技术的最新发展，以保持自己在这领域的竞争力。