CO515: Advances in Computer Networks: Selected Topics – Lab10
Big Data analysis with Hadoop: Extracting information from email records

E/19/446: Wijerathna I.M.K.D.I.

20/06/2024

Submission

1. Code Implementation.

Create a Java MapReduce program to extract useful information, such as the count of emails received by each sender.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
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 java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class EmailLogAnalysis {

    // Mapper class that processes each line of the input file
    public static class EmailMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text recipient = new Text();
        // Regular expression to extract the recipient email address
        private Pattern pattern = Pattern.compile("to=<(.+?)>");

        // The map method processes each line of the input file
        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
            Matcher matcher = pattern.matcher(value.toString());
            if (matcher.find()) {
                // Set the recipient email address as the key
                recipient.set(matcher.group(1));
                // Write the key-value pair to the context
                context.write(recipient, one);
            }
        }
    }
```

```java
 // Reducer class that aggregates the counts of emails for each recipient
    public static class EmailReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {
        private IntWritable result = new IntWritable();

        // The reduce method processes each group of values associated with a key
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
            int sum = 0;
            // Sum up the counts for each recipient
            for (IntWritable val : values) {
                sum += val.get();
            }
            // Set the result and write the key-value pair to the context
            result.set(sum);
            context.write(key, result);
        }
    }

    // Main method to configure and run the MapReduce job
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        // Create a new job instance
        Job job = Job.getInstance(conf, "email log analysis");
        // Set the jar file that contains the driver, mapper, and reducer classes
        job.setJarByClass(EmailLogAnalysis.class);
        // Set the mapper class
        job.setMapperClass(EmailMapper.class);
        // Set the combiner class, which is executed after the mapper but before the
reducer
        job.setCombinerClass(EmailReducer.class);
        // Set the reducer class
        job.setReducerClass(EmailReducer.class);
        // Set the output key and value types for the job
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        // Set the input and output paths for the job
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        // Exit the program based on the job result
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Explanation of the MapReduce program written.

This program implements a Hadoop MapReduce job that processes email log files to count the number of emails received by each recipient. The "EmailMapper" class extracts recipient email addresses from each log entry and outputs a key-value pair (recipient, 1). The "EmailReducer" class then aggregates these counts for each recipient, providing the total number of emails received by each one.

The behavior of this program is further explained in the following flowchart.

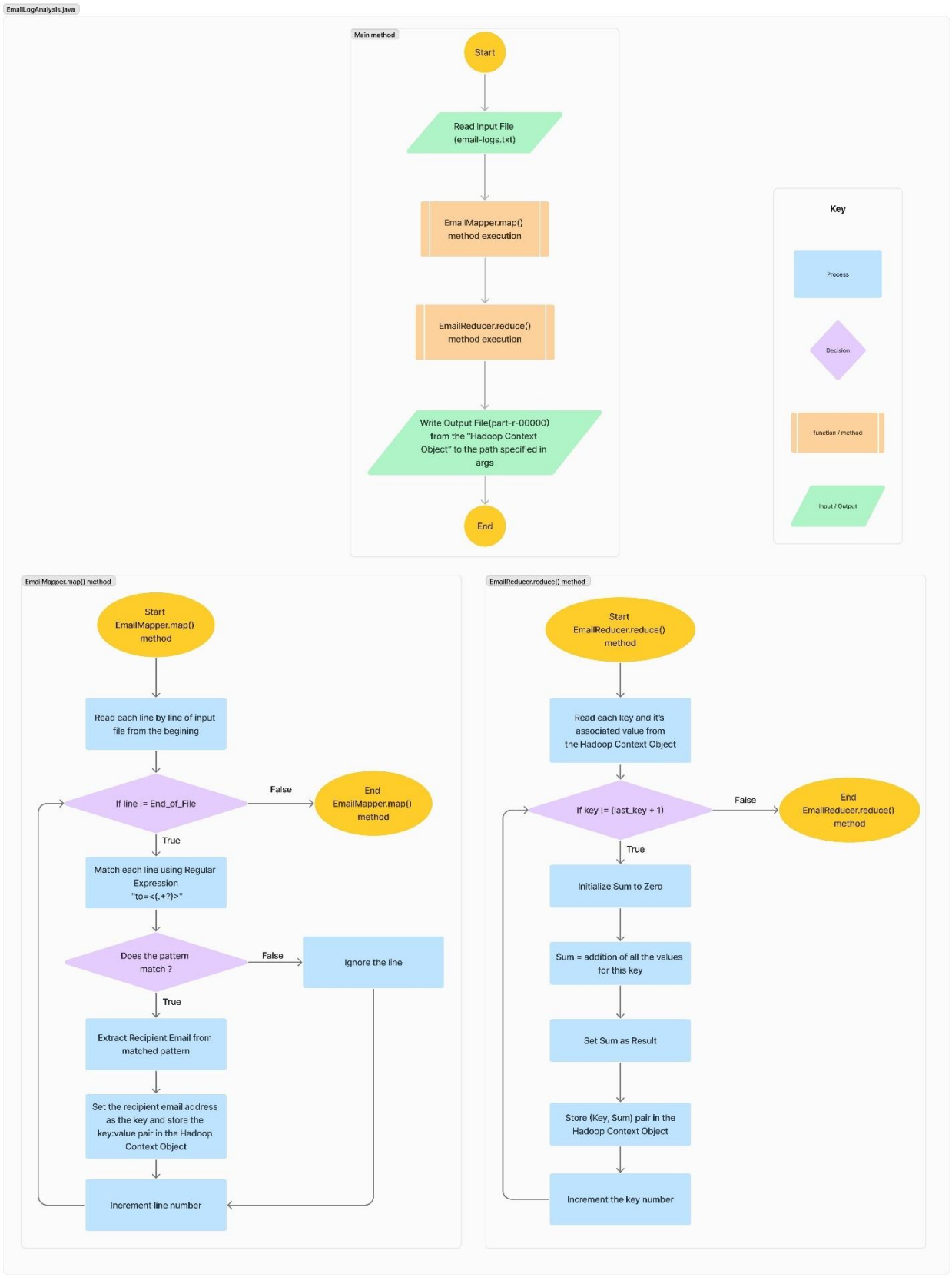

Figure 1: Flow chart explanation of the Map Reduce Job

2. Results: Output from the MapReduce job. Any observations or insights from the email log analysis

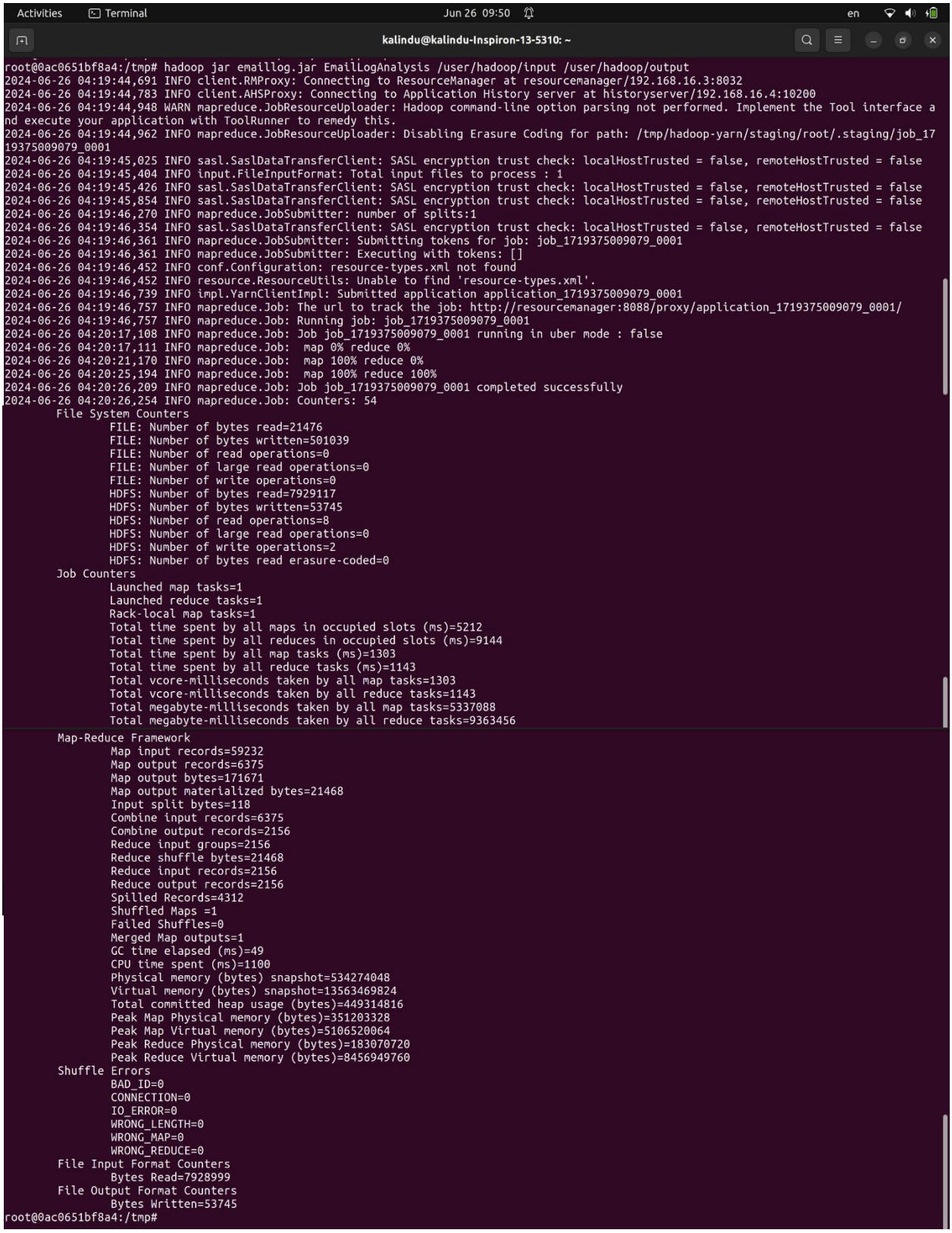

Figure 2: Results from running the Map Reduce Job

1. *Job Execution Summary:*
   - *The job ran successfully.*
   - *It involved both map and reduce tasks.*
   - *The total time spent by maps and reduces is provided.*
     - *Map tasks: 1100 milliseconds*
     - *Reduce tasks: 1143 milliseconds*
   - *Physical memory and virtual memory usage are reported.*
     - *Peak physical memory used by the entire job: approx.351 MB*
     - *Peak virtual memory used by the entire job: approx. 8,456 MB*

2. *MapReduce Counters:*
   - *Input records: 59,232*
   - *Output records: 6,375*
   - *Bytes read: 7,928,999*
   - *Bytes written: 53,745*

3. *File Input and Output Format Counters:*
   - *Bytes read from input files: 7,928,999*
   - *Bytes written to output files: 53,745*

Conclusions and Observations form the email log analysis:

To further analyze the results, I've used python to plot the output results.

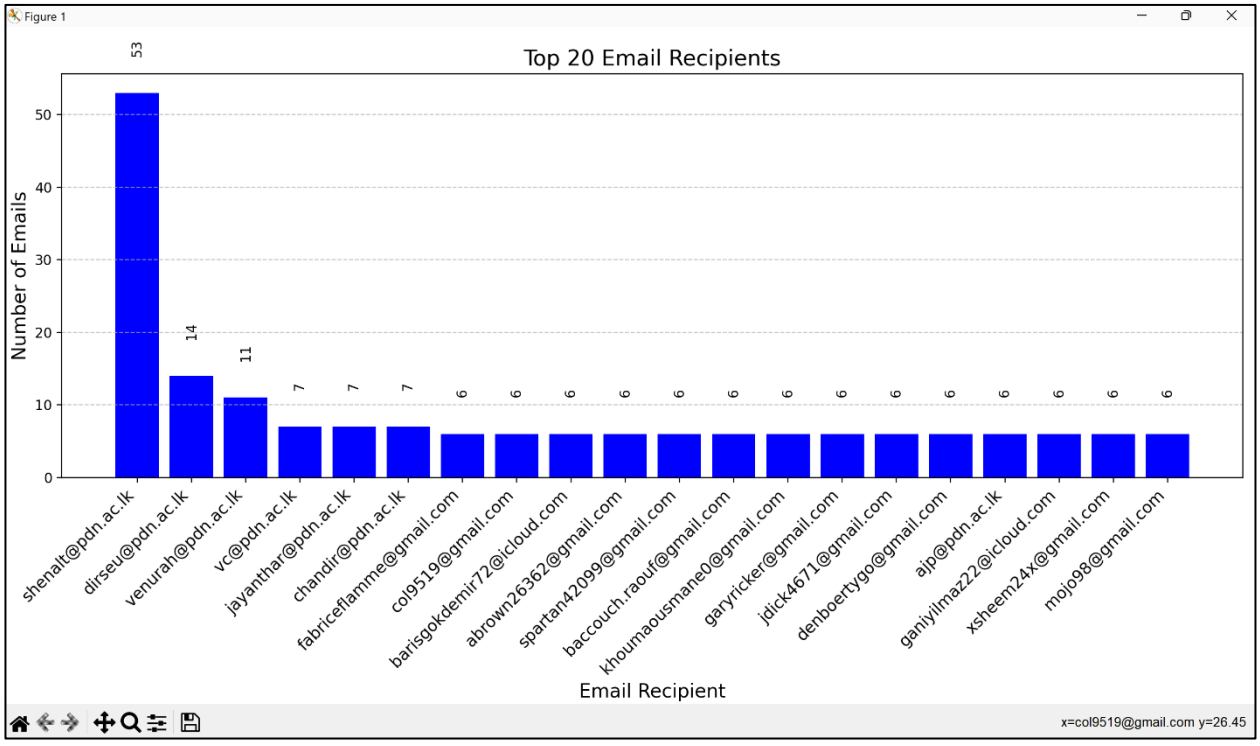

Figure 3: Top 20 Email Recipients

It can be observed that out of all the recorded 2156 distinct recipients, shenalt@pdn.ac.lk is the top recipient with a significant lead with 53 emails been received.

When further analyzing on the basis of grouping by email domain, "gmail.com", "pdn.ac.lk","hotmail.com" and "yahoo.com" were the highest out of the 90 distinct recorded email domains.

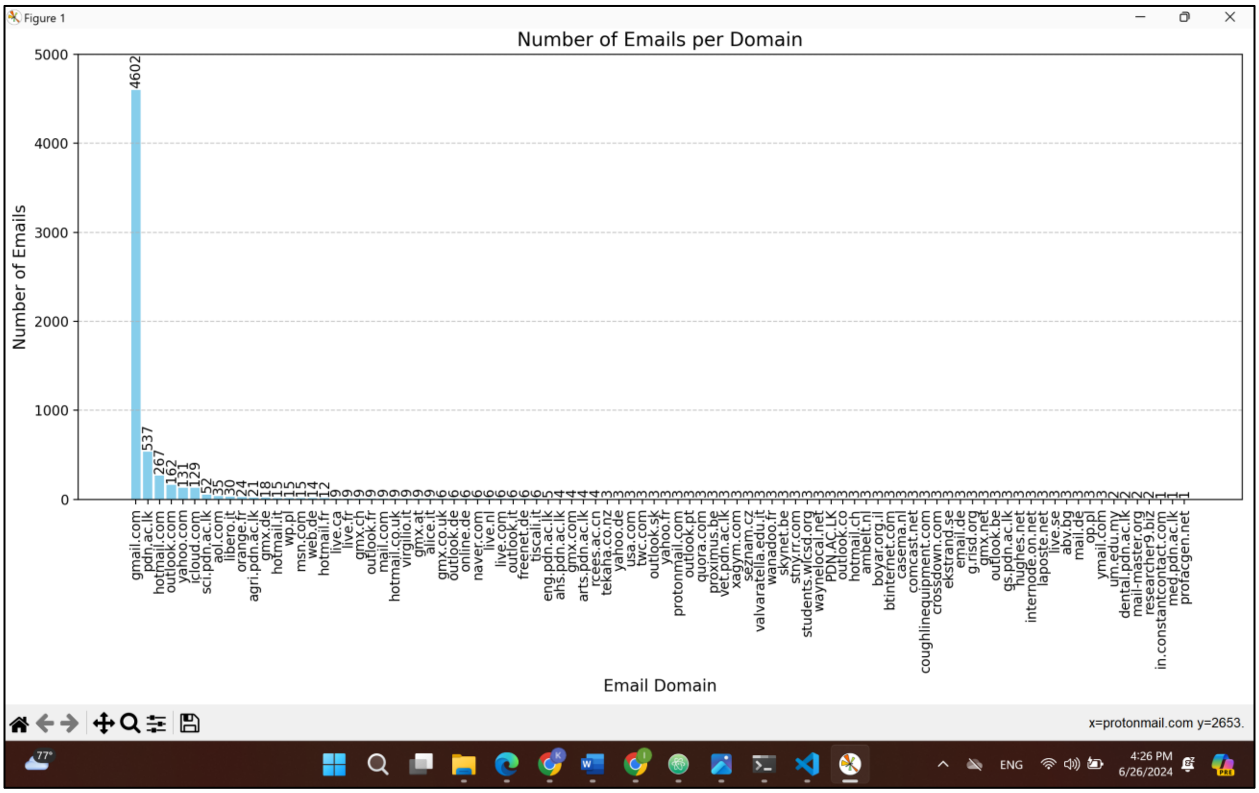

Figure 4: Plot Analisys on the basis of email domains

Therefore, in conclusion, Hadoop MapReduce framework demonstrated significant potential in processing and analyzing large-scale email log data efficiently. By leveraging the parallel processing capabilities of Hadoop, I was able to quickly extract and aggregate useful information from a substantial dataset. The results obtained from the MapReduce job provided valuable insights into email traffic patterns, identifying top recipients and dominant email domains. This showcases Hadoop's ability to handle and process big data, making it an invaluable tool for large-scale data analysis and deriving meaningful insights from complex datasets.

The python codes I've used for this plot analysis are attached in the appendix.

Lab Activities

Activity 1: Setting up Hadoop Environment

For this lab I used a docker compose environment using the official Hadoop docker images for namenode, datanode, resourcemanager, nodemanager, and historyserver.
(https://hub.docker.com/r/apache/hadoop)

My Hadoop Docker cluster setup:

```
/docker-hadoop-project-directory
├── docker-compose.yml
├── hadoop.env
├── namenode
│   ├── Dockerfile
│   └── email-logs.txt
└── volumes
    ├── hadoop_namenode
    ├── hadoop_datanode
    └── hadoop_historyserver
```

Figure 5: Hadoop docker cluster file structure

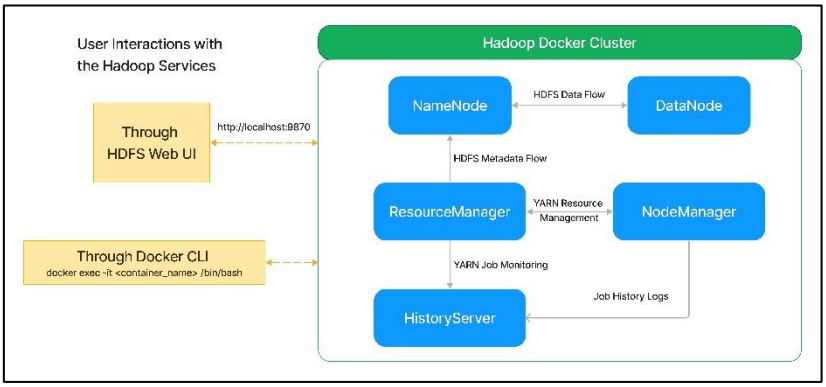

Figure 6: Hadoop docker cluster setup

The docker-compose.yml file, hadoop.env file and ./namenode/Dockerfile are attached in the appendix.

Starting the Hadoop services:

```
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab10/myhadoop/docker-hadoop$ docker compose up --build -d
[+] Running 5/6
 ⠿ Network docker-hadoop_default  Created
 ✔ Container namenode             Started
 ✔ Container nodemanager          Started
 ✔ Container datanode             Started
 ✔ Container resourcemanager      Started
 ✔ Container historyserver        Started
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab10/myhadoop/docker-hadoop$
```

Figure 7: Starting Hadoop cluster using docker compose

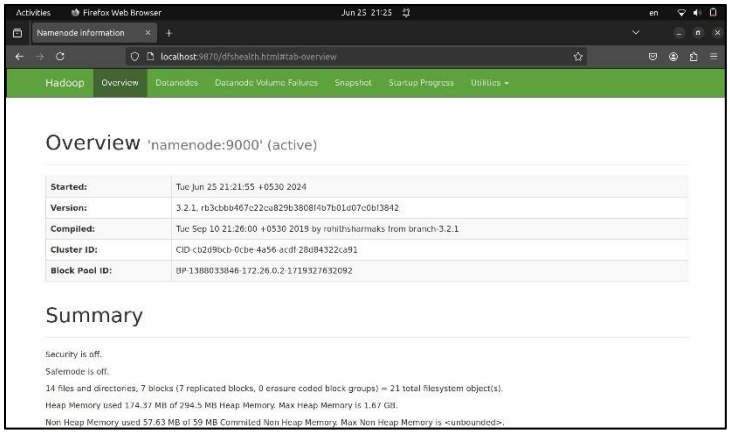

Figure 8: HDFS Web UI started

Activity 2: Loading Email Logs into HDFS

In the execution of Dockerfile for namenode container, I have already copied the provided email-logs.txt into /tmp/ folder of the namenode container. The following is the command implementation for uploading email-logs.txt from /tmp/ into the HDFS directory.

```
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO315/Lab10/myhadoop/docker-hadoop/namenode$ docker exec -it namenode /bin/bash
root@0ac0651bf8a4:~# hdfs dfs -put /tmp/email-logs.txt /user/hadoop/input
2024-06-25 16:19:46,883 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@0ac0651bf8a4:~#
```

Figure 9: Uploading email logs into HDFS

Data successfully uploaded, and visible form the HDFS web UI:

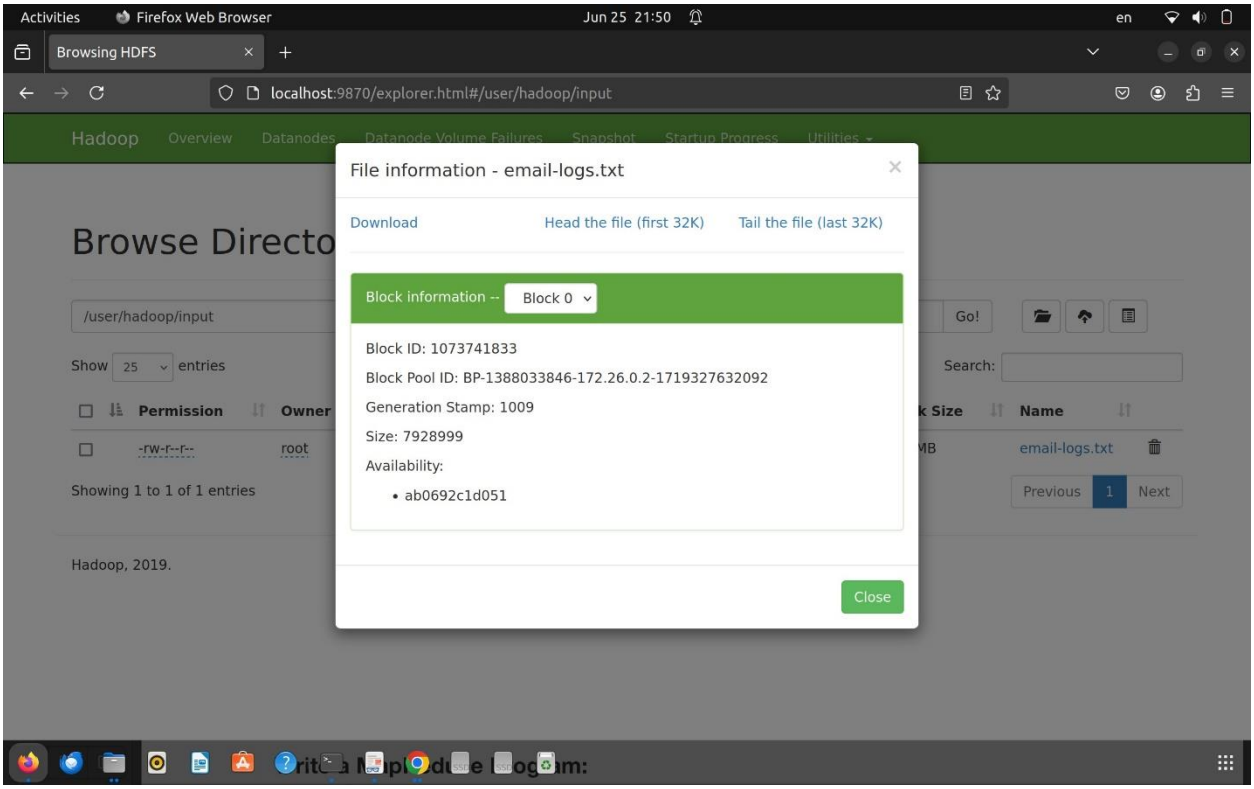

Figure 10: email-logs.txt is visible in the HDFS directory

Activity 3: Running a MapReduce Job to Extract Information

```
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab10/myhadoop/docker-hadoop/namenode$ docker cp ./EmailLogAnalysis.java namenode:/tmp/EmailL
ogAnalysis.java
Successfully copied 4.1kB to namenode:/tmp/EmailLogAnalysis.java
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab10/myhadoop/docker-hadoop/namenode$ docker exec -it namenode /bin/bash
root@0ac0651bf8a4:/# cd tmp
root@0ac0651bf8a4:/tmp# # Create a directory for compiled classes
root@0ac0651bf8a4:/tmp# mkdir -p emaillog_classes
root@0ac0651bf8a4:/tmp# javac -classpath `hadoop classpath` -d emaillog_classes EmailLogAnalysis.java
root@0ac0651bf8a4:/tmp# jar -cvf emaillog.jar -C emaillog_classes/ .
added manifest
adding: EmailLogAnalysis$EmailReducer.class(in = 1758) (out= 746)(deflated 57%)
adding: EmailLogAnalysis$EmailMapper.class(in = 2006) (out= 868)(deflated 56%)
adding: EmailLogAnalysis.class(in = 1517) (out= 815)(deflated 46%)
root@0ac0651bf8a4:/tmp#
```

Figure 11: Copying the java code into the namenode container and creating the jar file

```
Activities   Terminal                    Jun 26 09:50                    en
                                  kalindu@kalindu-Inspiron-13-5310: ~
root@0ac0651bf8a4:/tmp# hadoop jar emaillog.jar EmailLogAnalysis /user/hadoop/input /user/hadoop/output
2024-06-26 04:19:44,691 INFO client.RMProxy: Connecting to ResourceManager at resourcemanager/192.168.16.3:8032
2024-06-26 04:19:44,783 INFO client.AHSProxy: Connecting to Application History server at historyserver/192.168.16.4:10200
2024-06-26 04:19:44,948 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
2024-06-26 04:19:44,962 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_17
19375009079_0001
2024-06-26 04:19:45,025 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-06-26 04:19:45,404 INFO input.FileInputFormat: Total input files to process : 1
2024-06-26 04:19:45,426 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-06-26 04:19:45,854 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-06-26 04:19:46,270 INFO mapreduce.JobSubmitter: number of splits:1
2024-06-26 04:19:46,354 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2024-06-26 04:19:46,361 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1719375009079_0001
2024-06-26 04:19:46,361 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-06-26 04:19:46,452 INFO conf.Configuration: resource-types.xml not found
2024-06-26 04:19:46,452 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-06-26 04:19:46,739 INFO impl.YarnClientImpl: Submitted application application_1719375009079_0001
2024-06-26 04:19:46,757 INFO mapreduce.Job: The url to track the job: http://resourcemanager:8088/proxy/application_1719375009079_0001/
2024-06-26 04:19:46,757 INFO mapreduce.Job: Running job: job_1719375009079_0001
2024-06-26 04:20:17,108 INFO mapreduce.Job: Job job_1719375009079_0001 running in uber mode : false
2024-06-26 04:20:17,111 INFO mapreduce.Job:  map 0% reduce 0%
2024-06-26 04:20:21,170 INFO mapreduce.Job:  map 100% reduce 0%
2024-06-26 04:20:25,194 INFO mapreduce.Job:  map 100% reduce 100%
2024-06-26 04:20:26,209 INFO mapreduce.Job: Job job_1719375009079_0001 completed successfully
2024-06-26 04:20:26,254 INFO mapreduce.Job: Counters: 54
```

Figure 12: Running the Job

Full output from running the job is as provided in the Figure 2. (page No.4)

Activity 4: Analyzing Results

Output file from the job:

```
Activities   Terminal                    Jun 26 09:52                    en
                                  kalindu@kalindu-Inspiron-13-5310: ~
root@0ac0651bf8a4:~# hdfs dfs -cat /user/hadoop/output/part-r-00000
2024-06-26 04:22:01,212 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
015736582906ceco@gmail.com      3
07lehwald12@web.de      3
10pelopez10@gmail.com   3
1op1kledinggoedkoop@gmail.com   3
1ronnieleeallen1982@gmail.com   3
1uniquebird@gmail.com   3
2218.meis@gmail.com     3
23thomas.williams@gmail.com     3
2m1oe2@gmail.com        3
3akash2002@gmail.com    3
4kbuizyb94@gmail.com    6
63tate.d@gmail.com      3
7134lola@gmail.com      3
75gast@gmail.com        3
8647237707ms.ms3171@gmail.com   3
912braden@gmail.com     3
A8kXgdkK0Tr61iADh71QRpQ==_1114652663026_0r9n/LBCEeytQ/oWPsAWSg==@in.constantcontact.com 1
ALUWIHARE@PDN.AC.LK     1
KMW@PDN.AC.LK   2
a.kaufmann54@gmx.net    3
a.lyleevans@gmail.com   3
a.omdijkshoorn@gmail.com        3
aaip@pdn.ac.lk  4
aangforghostz@gmail.com 3
aarularasi@pdn.ac.lk    2
abbasesteban@gmail.com  3
abd746447@gmail.com     3
abdulhalimrushdi3131@gmail.com  3
abergleif404@gmail.com  3
abirferdous443@gmail.com        3
abrahamsniyaaz10@gmail.com      3
```

Figure 13: content in output/part-r-00000 file

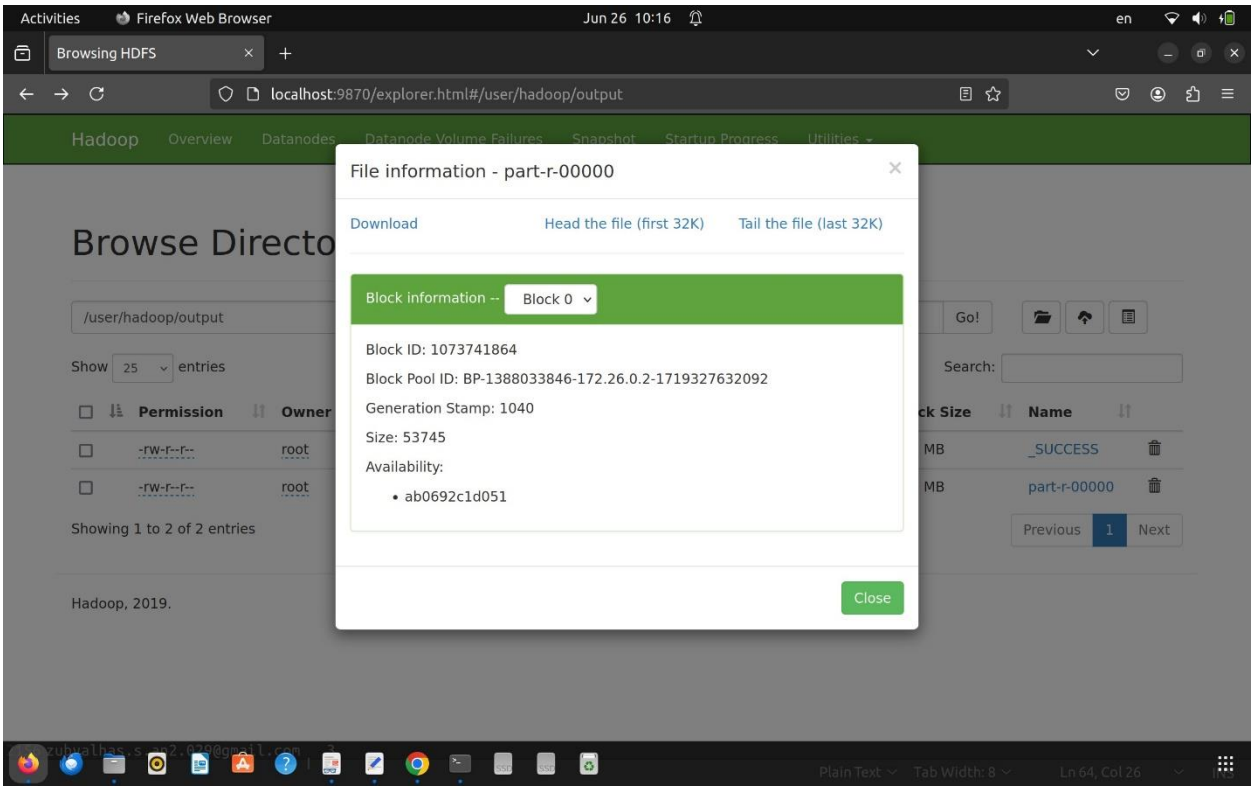

Figure 14: Output file is visible in the web UI as well

Appendix

docker-compose.yml

```yaml
version: "3"

services:
  namenode:
    build:
      context: ./namenode/
      dockerfile: Dockerfile
    container_name: namenode
    restart: always
    ports:
      - 9870:9870
      - 9000:9000
    volumes:
      - hadoop_namenode:/hadoop/dfs/name
    environment:
      - CLUSTER_NAME=test
    env_file:
      - ./hadoop.env

  datanode:
    image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
    container_name: datanode
    restart: always
    volumes:
      - hadoop_datanode:/hadoop/dfs/data
    environment:
      SERVICE_PRECONDITION: "namenode:9870"
    env_file:
      - ./hadoop.env

  resourcemanager:
    image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8
    container_name: resourcemanager
    restart: always
    environment:
      SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"
    env_file:
      - ./hadoop.env

  nodemanager1:
    image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8
    container_name: nodemanager
    restart: always
    environment:
      SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864
resourcemanager:8088"
    env_file:
```

```
      - ./hadoop.env

  historyserver:
    image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8
    container_name: historyserver
    restart: always
    environment:
      SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864
resourcemanager:8088"
    volumes:
      - hadoop_historyserver:/hadoop/yarn/timeline
    env_file:
      - ./hadoop.env

volumes:
  hadoop_namenode:
  hadoop_datanode:
  hadoop_historyserver:
```

./namenode/Dockerfile

```
FROM bde2020/hadoop-base:2.0.0-hadoop3.2.1-java8

MAINTAINER Ivan Ermilov <ivan.s.ermilov@gmail.com>

HEALTHCHECK CMD curl -f http://localhost:9870/ || exit 1

ENV HDFS_CONF_dfs_namenode_name_dir=file:///hadoop/dfs/name
RUN mkdir -p /hadoop/dfs/name
VOLUME /hadoop/dfs/name

# Copy the email-logs.txt file to the container
COPY email-logs.txt /tmp/email-logs.txt

ADD run.sh /run.sh
RUN chmod a+x /run.sh

EXPOSE 9870

CMD ["/run.sh"]
```

Codes that I used to plot the output results received from the MapReduce job

convertToCSV.py

```
with open('./part-r-00000.txt', 'r') as infile, open('./part-r-00000.csv', 'w') as
outfile:
    for line in infile:
        parts = line.strip().split('\t')  #  tab-separated data
        outfile.write(f'{parts[0]},{parts[1]}\n')
```

analyzeByRecepeint.py

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the output file into a pandas DataFrame
df = pd.read_csv('./part-r-00000.csv', sep=',', header=None, names=['Recipient',
'Count'])

# Sort the DataFrame by the 'Count' column in descending order
df_sorted = df.sort_values(by='Count', ascending=False)

# Select the top ten email recipients
df_top_ten = df_sorted.head(20)

# Plot the data
plt.figure(figsize=(15, 10))
bars = plt.bar(df_top_ten['Recipient'], df_top_ten['Count'], color='blue')

# Add labels and title
plt.xlabel('Email Recipient', fontsize=14)
plt.ylabel('Number of Emails', fontsize=14)
plt.title('Top 20 Email Recipients', fontsize=16)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, fontsize=12, ha='right')

# Add text annotations on top of each bar (vertically)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 5, int(yval), ha='center',
va='bottom', fontsize=10, rotation=90)

# Add gridlines
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
# Display the plot
plt.tight_layout()
plt.show()
```

groupByDomian.py

```
import pandas as pd
import matplotlib.pyplot as plt

# Function to extract domain from email
def extract_domain(email):
    return email.split('@')[-1]

# Read the output file into a pandas DataFrame
df = pd.read_csv('part-r-00000.csv', sep=',', header=None, names=['Recipient',
'Count'])

# Extract domain from each email recipient
df['Domain'] = df['Recipient'].apply(extract_domain)

# Group by domain and sum the counts
domain_counts = df.groupby('Domain')['Count'].sum().reset_index()

# Sort the DataFrame by the 'Count' column in descending order
domain_counts_sorted = domain_counts.sort_values(by='Count', ascending=False)

# Plot the data
plt.figure(figsize=(15, 10))
bars = plt.bar(domain_counts_sorted['Domain'], domain_counts_sorted['Count'],
color='skyblue')

# Add labels and title
plt.xlabel('Email Domain', fontsize=12)
plt.ylabel('Number of Emails', fontsize=12)
plt.title('Number of Emails per Domain', fontsize=14)

# Rotate x-axis labels for better readability
plt.xticks(rotation=90, fontsize=10)

# Set y-axis range from 0 to 5000
plt.ylim(0, 5000)

# Add gridlines
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
# Add text annotations on top of each bar (vertically)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 20, int(yval), ha='center',
va='bottom', fontsize=10, rotation=90)

# Display the plot
plt.tight_layout()
plt.show()
```