Submission by:

Shubham Gupta (202318052)

Big Data Processing

(Assignment 3)

Title: Implementation of MapReduce Algorithm on Multi-Node Hadoop Cluster

Objective:

The objective of this project is to set up a multi-node EC2 instance on AWS, install Hadoop, implement the MapReduce algorithm in both single-node and multi-node clusters using the Hadoop streaming utility, and compare the execution time between the two setups.

Setup Multi-Node EC2 Instances:

1. Log in to the AWS Management Console.
2. Navigate to the EC2 dashboard.
3. Launch four EC2 instances in the Mumbai region and south availability zone.
4. Choose the Ubuntu OS during creation of EC2 instances.
5. Named the instances according to my student ID.
6. Assigned appropriate security groups and key pairs.
7. Noted down the public IP addresses or DNS names of each instance.

Install Hadoop:

1. SSH into each EC2 instance using the key pair.
2. Install Java Development Kit (JDK 8) if not already installed.
3. Download the Hadoop distribution 2.7.3 and extract it.
4. Configure Hadoop by editing configuration files (hadoop-env.sh, core-site.xml, hdfs-site.xml, mapred-site.xml, etc.).
5. Format the Hadoop file system using hadoop namenode -format.
6. Start the Hadoop daemons: start-dfs.sh and start-yarn.sh.
7. Verify the Hadoop installation through the web interface.

Implement MapReduce:

1. Write the mapper and reducer code in python.
2. Upload the code to a directory on the master node.
3. Ensure the input data is available and accessible to Hadoop.
4. Run the MapReduce job using the Hadoop streaming utility.
5. Monitor the job progress through the Hadoop web interface.

6. Check the output directory for the results upon completion.

```
#Mapper.py

#!/usr/bin/python3 -0
import sys

#Loop through each line in the input
for line in sys.stdin:

    # Remove leading and trailing whitespace
    line = line.strip()

    # Split the line into words
    words = line.split()

    # Emit key-value pairs of word and count of 1
    for word in words:
        print(word,"\t",1)
```

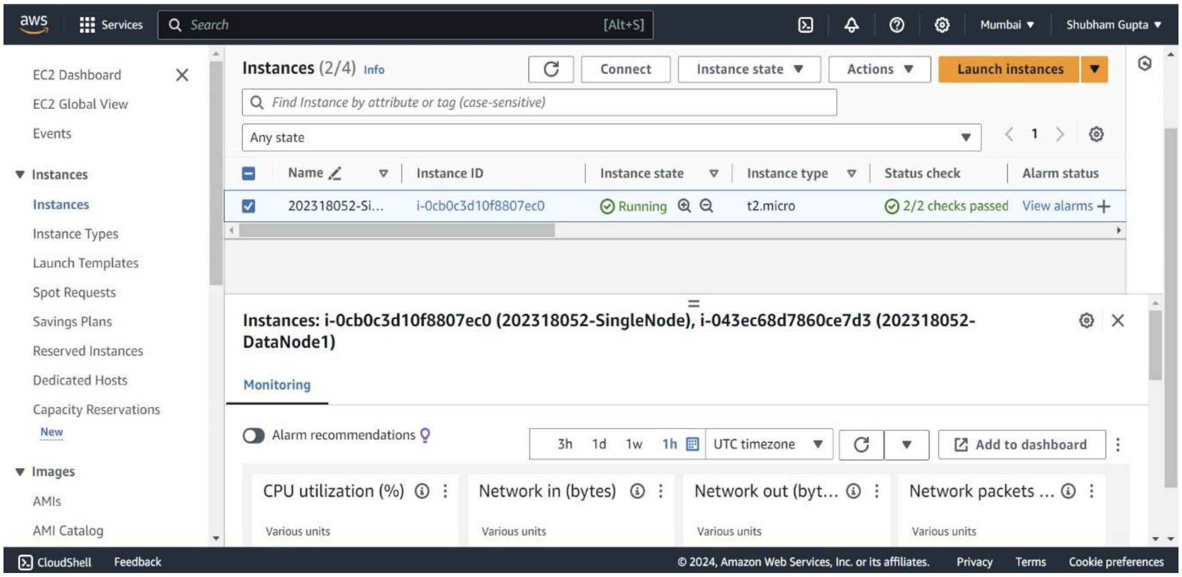
```
#Reducer.py
#!/usr/bin/python3 -0

import sys

#Initialize variables to keep track of current word and its count
current word = None
current_count = 0

#Loop through each line in the input
for line in sys.stdin:

    #Split the line into word and count, separated by tab
    word, count = line.strip().split('\t', 1)

    #Convert count to integer
    count = int(count)

    #If the word is the same as the current word, increment its count
    if word == current_word:
        current_count += count

    else:

        #If the word is different, print the current word and its count
        if current word:
            print(current_word,"\t",current_count)

            #Update current word and its count
            current word = word
            current count = count

#Print the last word and its count
if current word:
    print(current_word,"\t",current_count)
```

Compare Execution Time:

1. Record the start time before running the MapReduce job.
2. Execute the job on both single-node and multi-node clusters.
3. Note the completion time after each execution.
4. Calculate the execution time difference between the two setups.

Results and Screenshots:

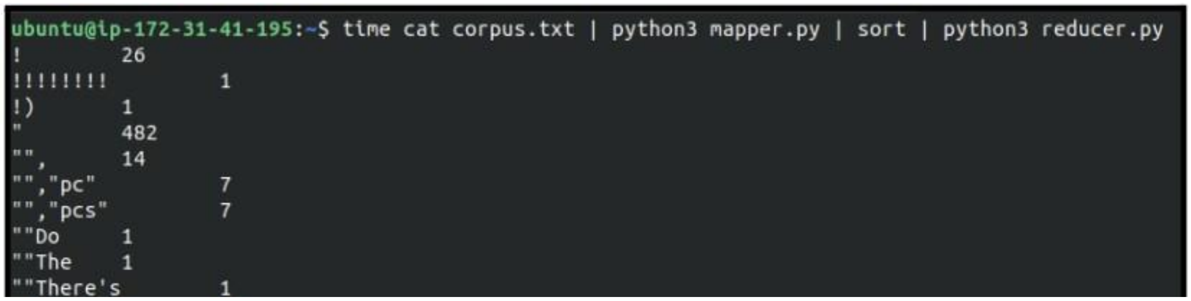
1) Single Node



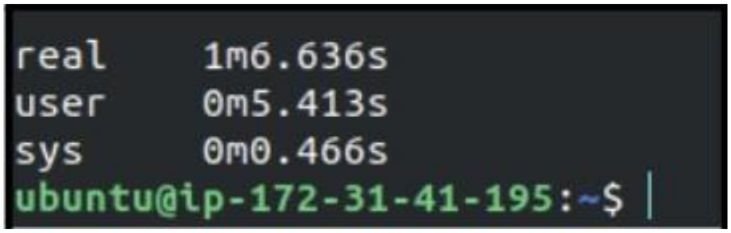
☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
☑	202318052-Si...	i-0cb0c3d10f8807ec0	Running	t2.micro	2/2 checks passed	View alarms +

Instances: i-0cb0c3d10f8807ec0 (202318052-SingleNode), i-043ec68d7860ce7d3 (202318052-DataNode1)

Without Hadoop

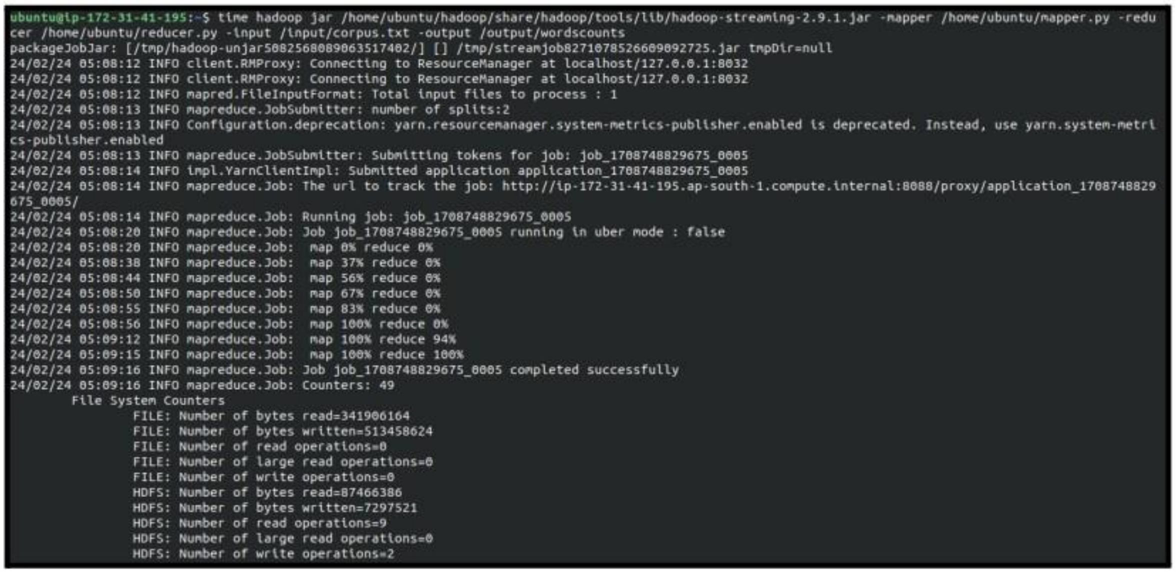


```
ubuntu@ip-172-31-41-195:~$ time cat corpus.txt | python3 mapper.py | sort | python3 reducer.py
!       26
!!!!!!!!        1
!)      1
"       482
"",     14
"","pc"         7
"","pcs"        7
""Do    1
""The   1
""There's       1
```

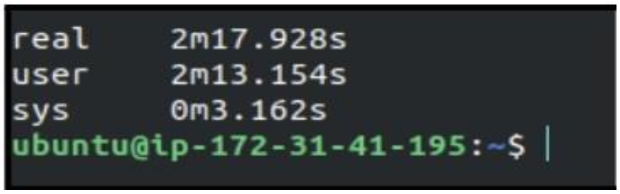


```
real    1m6.636s
user    0m5.413s
sys     0m0.466s
ubuntu@ip-172-31-41-195:~$ 
```

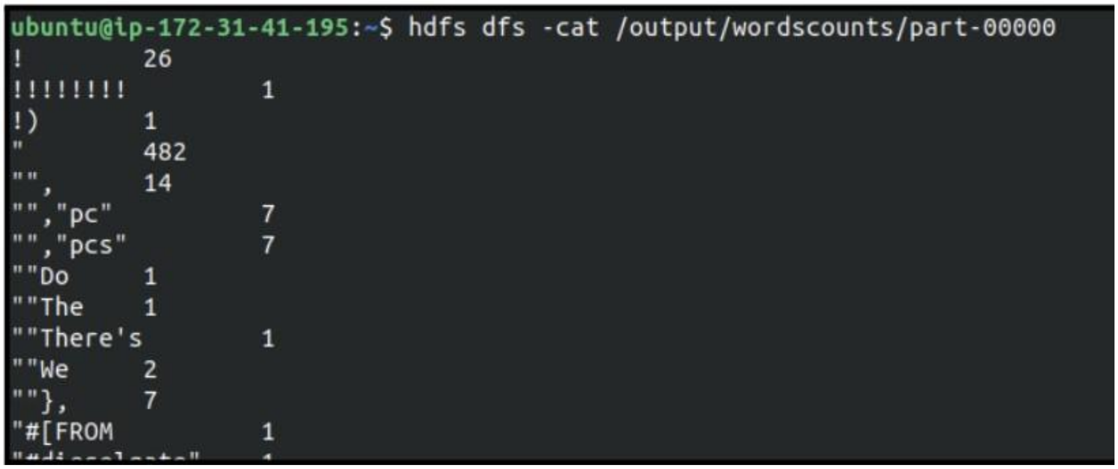
With Hadoop



```
ubuntu@ip-172-31-41-195:~$ time hadoop jar /home/ubuntu/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.1.jar -mapper /home/ubuntu/mapper.py -redu
cer /home/ubuntu/reducer.py -input /input/corpus.txt -output /output/wordscounts
packageJobJar: [/tmp/hadoop-unjar5082568089063517402/] [] /tmp/streamjob8271078526609092725.jar tmpDir=null
24/02/24 05:08:12 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
24/02/24 05:08:12 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
24/02/24 05:08:12 INFO mapred.FileInputFormat: Total input files to process : 1
24/02/24 05:08:13 INFO mapreduce.JobSubmitter: number of splits:2
24/02/24 05:08:13 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metri
cs-publisher.enabled
24/02/24 05:08:13 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1708748829675_0005
24/02/24 05:08:14 INFO impl.YarnClientImpl: Submitted application application_1708748829675_0005
24/02/24 05:08:14 INFO mapreduce.Job: The url to track the job: http://ip-172-31-41-195.ap-south-1.compute.internal:8088/proxy/application_1708748829
675_0005/
24/02/24 05:08:14 INFO mapreduce.Job: Running job: job_1708748829675_0005
24/02/24 05:08:20 INFO mapreduce.Job: Job job_1708748829675_0005 running in uber mode : false
24/02/24 05:08:20 INFO mapreduce.Job:  map 0% reduce 0%
24/02/24 05:08:38 INFO mapreduce.Job:  map 37% reduce 0%
24/02/24 05:08:44 INFO mapreduce.Job:  map 56% reduce 0%
24/02/24 05:08:50 INFO mapreduce.Job:  map 67% reduce 0%
24/02/24 05:08:55 INFO mapreduce.Job:  map 83% reduce 0%
24/02/24 05:08:56 INFO mapreduce.Job:  map 100% reduce 0%
24/02/24 05:09:12 INFO mapreduce.Job:  map 100% reduce 94%
24/02/24 05:09:15 INFO mapreduce.Job:  map 100% reduce 100%
24/02/24 05:09:16 INFO mapreduce.Job: Job job_1708748829675_0005 completed successfully
24/02/24 05:09:16 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=341906164
                FILE: Number of bytes written=513458624
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=87466386
                HDFS: Number of bytes written=7297521
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
```

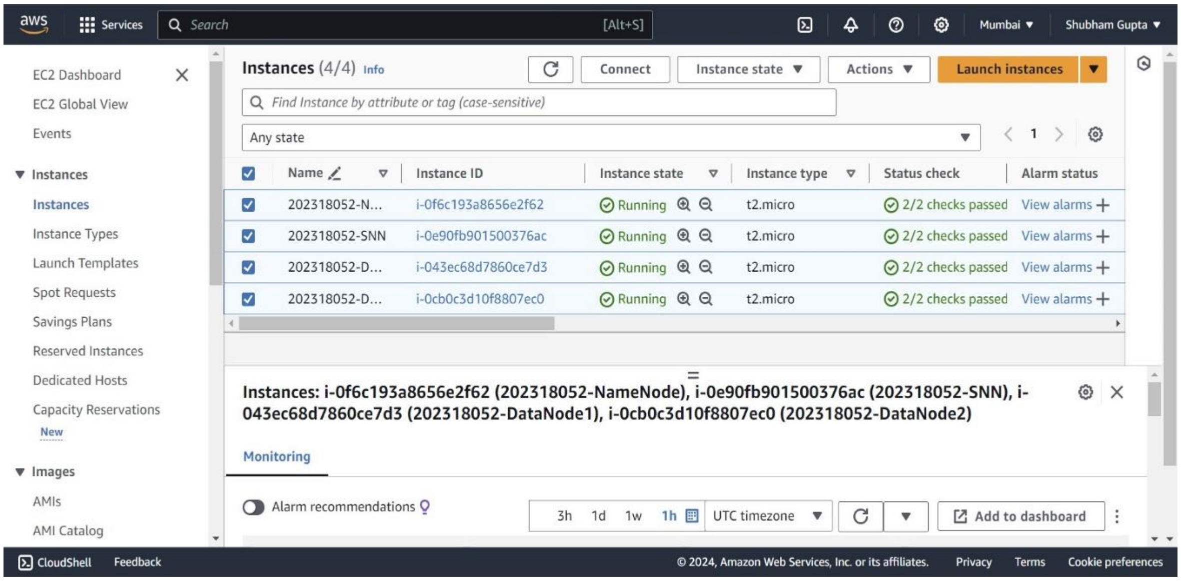


```
real    2m17.928s
user    2m13.154s
sys     0m3.162s
ubuntu@ip-172-31-41-195:~$ 
```



```
ubuntu@ip-172-31-41-195:~$ hdfs dfs -cat /output/wordscounts/part-00000
!               26
!!!!!!!!                1
!)              1
"               482
"",             14
"","pc"                 7
"","pcs"                7
""Do            1
""The           1
""There's               1
""We            2
""},            7
"#[FROM                 1
```

2) Multi-Node cluster



Instances (4/4) Info

Find Instance by attribute or tag (case-sensitive)

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
☑	202318052-N...	i-0f6c193a8656e2f62	Running	t2.micro	2/2 checks passed	View alarms +
☑	202318052-SNN	i-0e90fb901500376ac	Running	t2.micro	2/2 checks passed	View alarms +
☑	202318052-D...	i-043ec68d7860ce7d3	Running	t2.micro	2/2 checks passed	View alarms +
☑	202318052-D...	i-0cb0c3d10f8807ec0	Running	t2.micro	2/2 checks passed	View alarms +

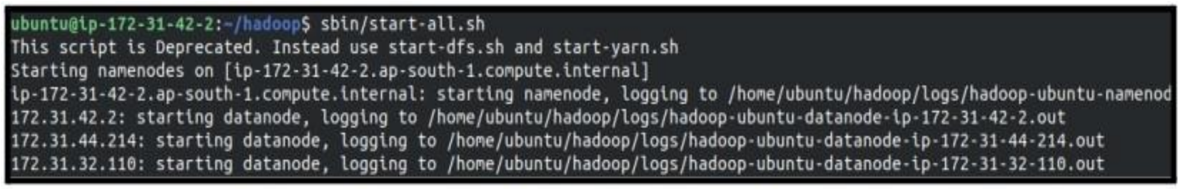
Instances: i-0f6c193a8656e2f62 (202318052-NameNode), i-0e90fb901500376ac (202318052-SNN), i-043ec68d7860ce7d3 (202318052-DataNode1), i-0cb0c3d10f8807ec0 (202318052-DataNode2)

Alarm recommendations

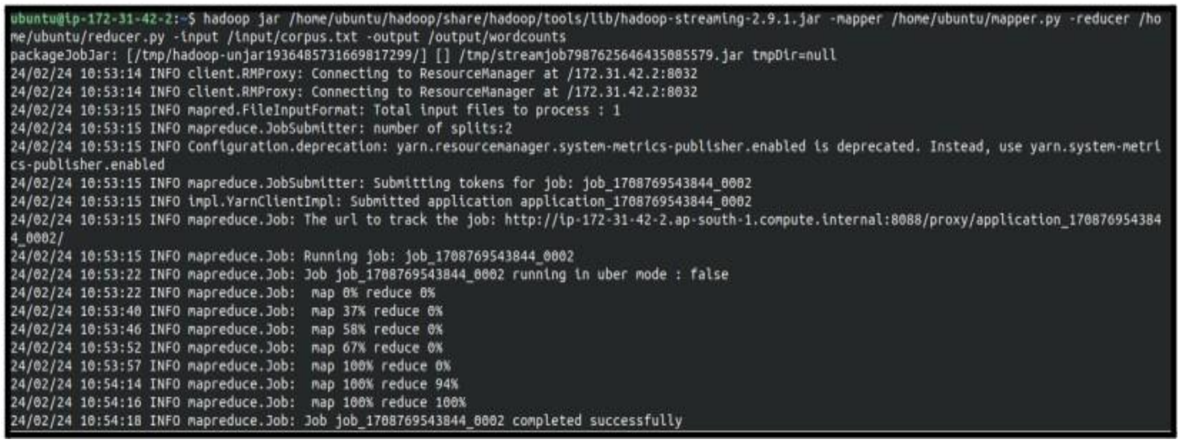
3h 1d 1w 1h UTC timezone

© 2024, Amazon Web Services, Inc. or its affiliates.

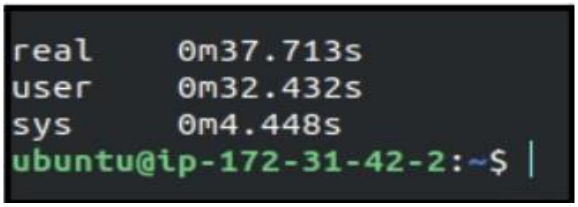
With Hadoop multi-node cluster



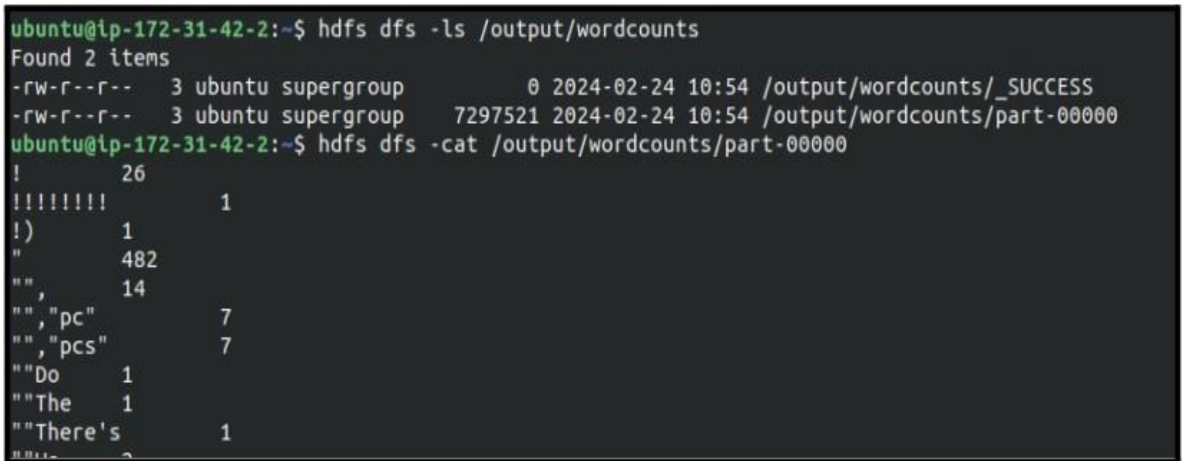
```
ubuntu@ip-172-31-42-2:~/hadoop$ sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [ip-172-31-42-2.ap-south-1.compute.internal]
ip-172-31-42-2.ap-south-1.compute.internal: starting namenode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-namenod
172.31.42.2: starting datanode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-42-2.out
172.31.44.214: starting datanode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-44-214.out
172.31.32.110: starting datanode, logging to /home/ubuntu/hadoop/logs/hadoop-ubuntu-datanode-ip-172-31-32-110.out
```



```
ubuntu@ip-172-31-42-2:~$ hadoop jar /home/ubuntu/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.1.jar -mapper /home/ubuntu/mapper.py -reducer /ho
me/ubuntu/reducer.py -input /input/corpus.txt -output /output/wordcounts
packageJobJar: [/tmp/hadoop-unjar1936485731669817299/] [] /tmp/streamjob7987625646435085579.jar tmpDir=null
24/02/24 10:53:14 INFO client.RMProxy: Connecting to ResourceManager at /172.31.42.2:8032
24/02/24 10:53:14 INFO client.RMProxy: Connecting to ResourceManager at /172.31.42.2:8032
24/02/24 10:53:15 INFO mapred.FileInputFormat: Total input files to process : 1
24/02/24 10:53:15 INFO mapreduce.JobSubmitter: number of splits:2
24/02/24 10:53:15 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metri
cs-publisher.enabled
24/02/24 10:53:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1708769543844_0002
24/02/24 10:53:15 INFO impl.YarnClientImpl: Submitted application application_1708769543844_0002
24/02/24 10:53:15 INFO mapreduce.Job: The url to track the job: http://ip-172-31-42-2.ap-south-1.compute.internal:8088/proxy/application_170876954384
4_0002/
24/02/24 10:53:15 INFO mapreduce.Job: Running job: job_1708769543844_0002
24/02/24 10:53:22 INFO mapreduce.Job: Job job_1708769543844_0002 running in uber mode : false
24/02/24 10:53:22 INFO mapreduce.Job:  map 0% reduce 0%
24/02/24 10:53:40 INFO mapreduce.Job:  map 37% reduce 0%
24/02/24 10:53:46 INFO mapreduce.Job:  map 58% reduce 0%
24/02/24 10:53:52 INFO mapreduce.Job:  map 67% reduce 0%
24/02/24 10:53:57 INFO mapreduce.Job:  map 100% reduce 0%
24/02/24 10:54:14 INFO mapreduce.Job:  map 100% reduce 94%
24/02/24 10:54:16 INFO mapreduce.Job:  map 100% reduce 100%
24/02/24 10:54:18 INFO mapreduce.Job: Job job_1708769543844_0002 completed successfully
```



```
real    0m37.713s
user    0m32.432s
sys     0m4.448s
ubuntu@ip-172-31-42-2:~$ |
```



```
ubuntu@ip-172-31-42-2:~$ hdfs dfs -ls /output/wordcounts
Found 2 items
-rw-r--r--   3 ubuntu supergroup          0 2024-02-24 10:54 /output/wordcounts/_SUCCESS
-rw-r--r--   3 ubuntu supergroup    7297521 2024-02-24 10:54 /output/wordcounts/part-00000
ubuntu@ip-172-31-42-2:~$ hdfs dfs -cat /output/wordcounts/part-00000
!       26
!!!!!!!!        1
!)      1
"       482
"",     14
"","pc"         7
"","pcs"        7
""Do    1
""The   1
""There's       1
```

Conclusion:

In conclusion, setting up a multi-node Hadoop cluster on AWS and implementing MapReduce algorithms allows for distributed data processing has significantly improve performance compared to single-node setups. The comparison of execution times shows efficiency of the cluster configuration.