**Submission by:**

*Simran Dalvi (202318042)*

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:*

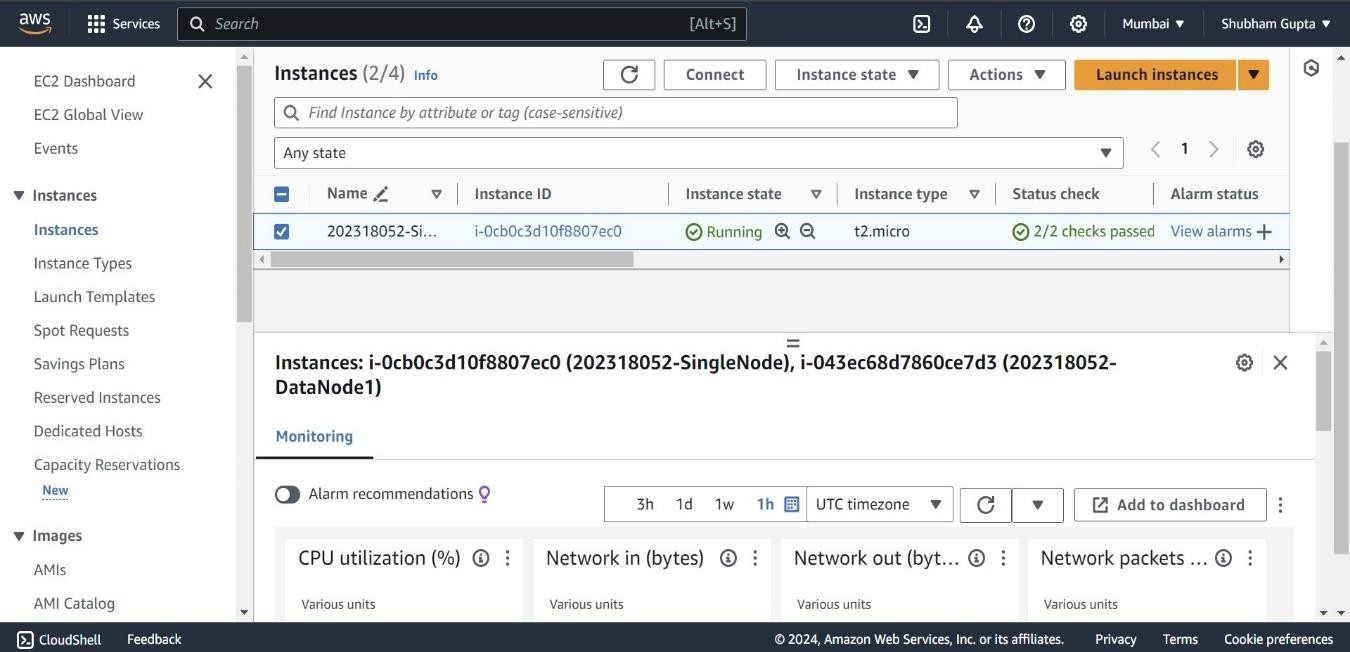
*print(current\_word,"\t",current\_count)*

# Compare Execution Time:

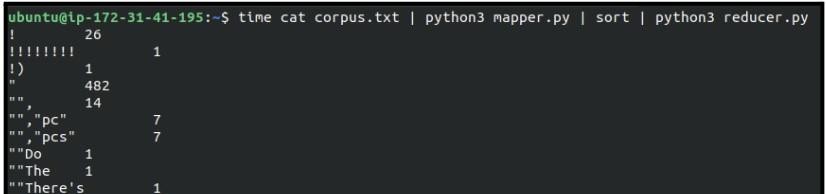
1. Record the start time before running the MapReduce job.
2. Execute the job on both single-node and multi-node clusters.
3. Note the completion time after each execution.
4. Calculate the execution time difference between the two setups.

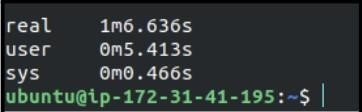
# Results and Screenshots:

1. **Single Node**

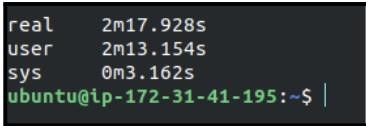
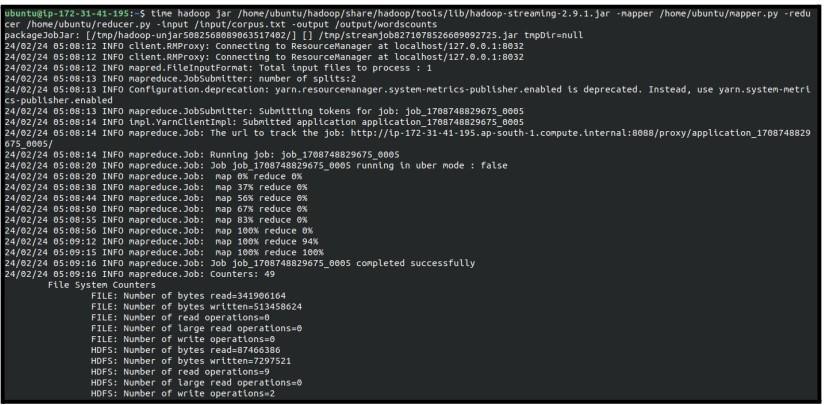


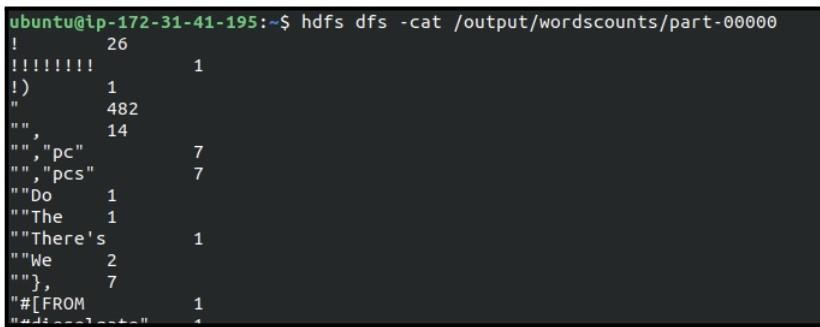
Without Hadoop



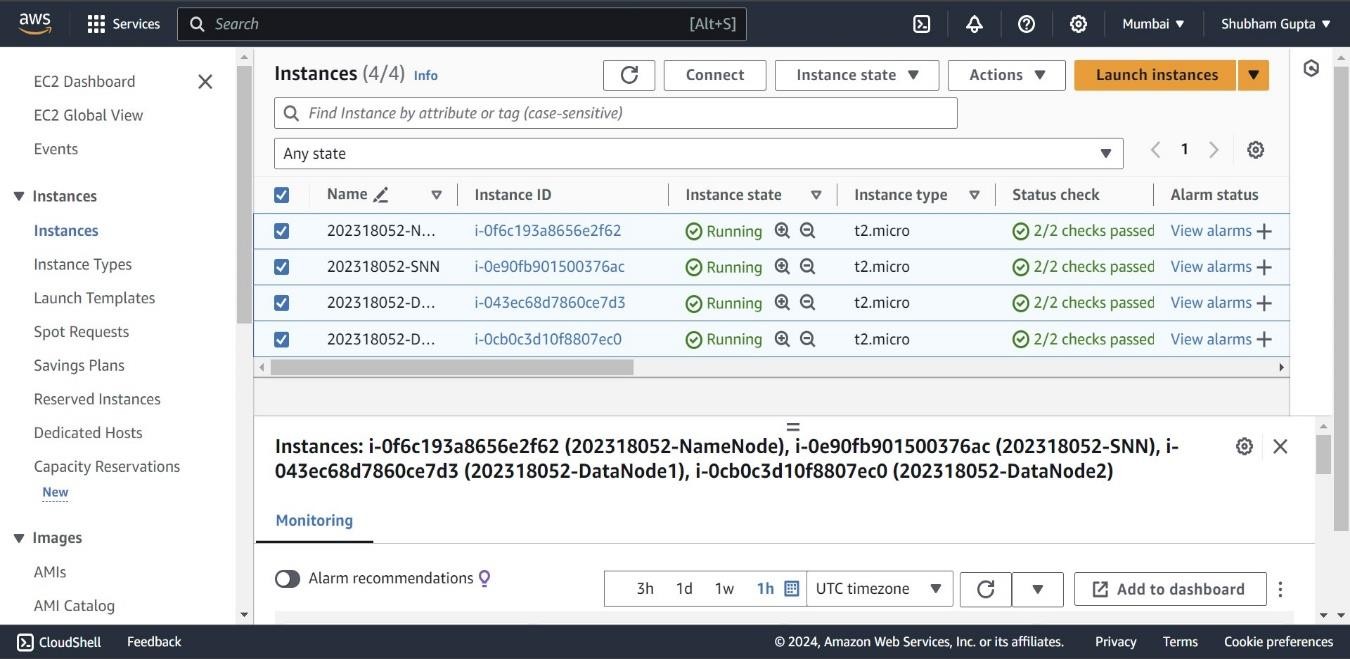


With Hadoop

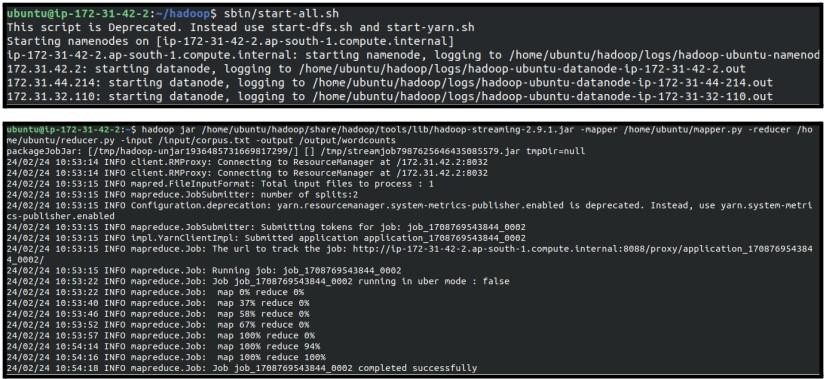


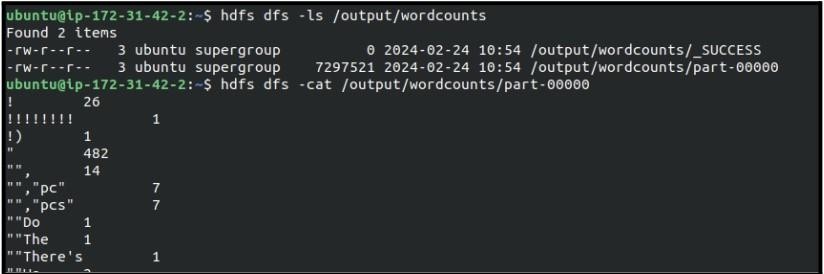
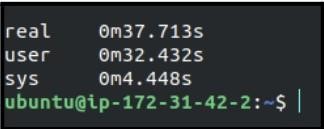


# Multi-Node cluster



With Hadoop multi-node cluster





# 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.