

Big Data Midterm Project: Big Data Processing with MapReduce on a Hadoop Cluster on AWS

Group size: 1 or 2 students.

Project Overview

In this project, you will gain hands-on experience working with **big data** by implementing a **MapReduce** job and running it on a fully distributed **Hadoop cluster** deployed on **AWS**. Students will be required to:

1. **Find a publicly available dataset** online.
 2. **Set up a Hadoop cluster** on AWS using four EC2 instances, and **Install** Java and Hadoop on your instances.
 3. **Write a MapReduce job** in Java to process the dataset.
 4. **Run and test the MapReduce job** on the cluster.
 5. **Analyze and interpret the output.**
-

Step 1: Finding a Dataset

You should explore online sources to find a dataset suitable for analysis. Some suggested sources include:

- **Kaggle** (<https://www.kaggle.com/datasets>)
- **Google Dataset Search** (<https://datasetsearch.research.google.com/>)
- **UCI Machine Learning Repository** (<https://archive.ics.uci.edu/ml/index.php>)
- **Data.gov** (<https://www.data.gov/>)

The dataset size should be at least **500MB+** to simulate big data processing.

Step 2: Setting up the Hadoop Cluster on AWS

You need to set up a **4-node Hadoop cluster** (1 Master, 3 Workers) using Amazon EC2 instances.

Instance Setup:

- Choose the **Ubuntu free tier version** as the OS.
- Configure **Security Groups** (allow SSH and Hadoop ports).
- Install **Java 8** and **Hadoop 3.x** on each instance.
- Set up **passwordless SSH** for communication between nodes.

- Configure **HDFS and YARN** to enable a fully distributed mode.

Hadoop Configuration Files to Modify:

- `core-site.xml`
 - `hdfs-site.xml`
 - `mapred-site.xml`
 - `yarn-site.xml`
 - `slaves`
-

Step 3: Implementing the MapReduce Job

Write a **Java-based MapReduce program** to process the dataset.

Example:

- **Word Count** (Basic Example).
-

Step 4: Running the MapReduce Job on Hadoop Cluster

(1) Copy dataset to HDFS:

```
hdfs dfs -put dataset.csv /input
```

(2) Run the MapReduce job:

```
hadoop jar mapreduce_project.jar MainClass /input /output
```

(3) Retrieve output from HDFS:

```
hdfs dfs -get /output output_folder
```

Step 5: Analyzing the Results

- Inspect and interpret the output data.
-

Project Submission Requirements

Submit a **Word file (.doc/.docx)** containing the following sections:

1. Title Page include

- Project Title
- Student Name/Names
- Course and Section Number
- NJIT UCID
- Email address

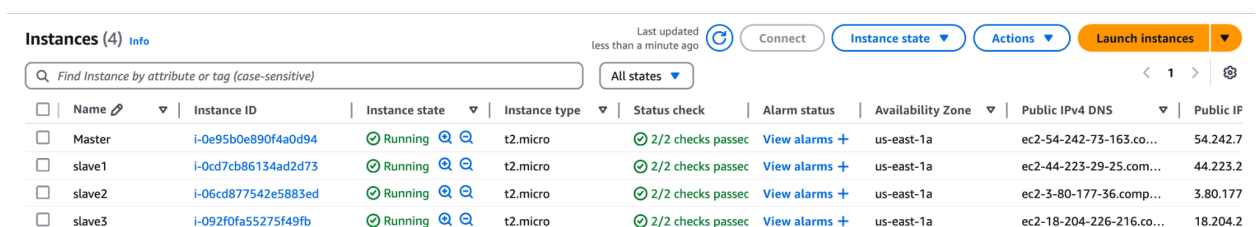
Otherwise, you will lose 5 points.

2. Dataset Introduction

- **Dataset Name & Source:** Provide the name and link to the dataset.
 - **Dataset Description:** Describe its content, fields, and structure.
 - **Size & Format:** Mention file size and format (CSV, JSON, TXT, etc.).
 - **Reason for Selection:** Explain why this dataset is useful for MapReduce processing.
-

3. Hadoop Cluster Setup Successful Screenshots

- (1) A screenshot of the Amazon instance management web interface showing the running state of each of your VM instances. Make sure your instance state is running, and the status check is passed like the screenshot below.



Instances (4) Info										
Find Instance by attribute or tag (case-sensitive)					All states		1			
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP	
<input type="checkbox"/>	Master	i-0e95b0e890f4a0d94	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-242-73-163.co...	54.242.7	
<input type="checkbox"/>	slave1	i-0cd7cb86134ad2d73	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-223-29-25.com...	44.223.2	
<input type="checkbox"/>	slave2	i-06cd877542e5883ed	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-3-80-177-36.comp...	3.80.177	
<input type="checkbox"/>	slave3	i-092f0fa55275f49fb	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-18-204-226-216.co...	18.204.2	

- (2) A screenshot of passphraseless SSH login from your Namenode/Master node instance to one of Datanodes/Slave nodes. The screenshot should be able to show a successful login message after you give the SSH login command.

Make sure this screenshot is like the one below, it should include:

- your Namenode/master node IP (ubuntu@ip-172-31-41-117 in green letters)

- ssh command (e.g, ssh slave1)
- a welcome message after successfully login to the slave instance (slave1 in this example), and the new instance's IP (slave1's IP in this example). This should work for all your Datanodes/slave nodes. Note that your instance IP will be different from the instances in my screenshots because every instance has a unique IP.

```
ubuntu@ip-172-31-41-117:~$ ssh slave1
The authenticity of host 'slave1 (172.31.33.210)' can't be established.
ED25519 key fingerprint is SHA256:gFhxJnXOTPt22yEyDSu5BMPaNwJYtUsBC0MWP8wMh+0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'slave1' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

System information as of Sun Sep 22 21:44:04 UTC 2024

System load:  0.0                Processes:           105
Usage of /:   23.0% of 6.71GB    Users logged in:    0
Memory usage: 19%                IPv4 address for enX0: 172.31.33.210
Swap usage:   0%

[
Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sun Sep 22 21:38:12 2024 from 87.249.138.210
ubuntu@ip-172-31-33-210:~$
logout
Connection to slave1 closed.
```

- (3) Provide two screenshots showing the output of the "jps" command on both the Namenode/Master node and Datanode/Slave nodes after you format the namenode and start Hadoop. Below is the correct output screenshots for your reference.

Namenode/Master node

```
ubuntu@ip-172-31-48-162:~$ jps
2240 SecondaryNameNode
2034 NameNode
2645 Jps
2391 ResourceManager
```

Datanode/Slave node

```
ubuntu@ip-172-31-59-36:~$ jps
1940 DataNode
2200 Jps
2074 NodeManager
```

4. MapReduce Code

- **Code Explanation:** Describe the Map, Reduce, and Driver classes.
- **Full Java Code** (properly formatted with comments).
- **Compilation & Execution Commands**

5. Code Execution & Output Interpretation

Successful Execution Screenshot:

- (1) Screen of running your code command and its result.

```

ubuntu@ip-172-31-82-103:~/hadoop-2.6.5/bin$ hadoop jar ~/MyWordCount.jar MyWordCount /input /output
24/10/06 14:24:26 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
24/10/06 14:24:26 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
24/10/06 14:24:26 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
tion with ToolRunner to remedy this.
24/10/06 14:24:26 INFO input.FileInputFormat: Total input paths to process : 1
24/10/06 14:24:26 INFO mapreduce.JobSubmitter: number of splits:1
24/10/06 14:24:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local310394148_0001
24/10/06 14:24:27 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
24/10/06 14:24:27 INFO mapreduce.Job: Running job: job_local310394148_0001
24/10/06 14:24:27 INFO mapred.LocalJobRunner: OutputCommitter set in config null
24/10/06 14:24:27 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitte
24/10/06 14:24:27 INFO mapred.LocalJobRunner: Waiting for map tasks
24/10/06 14:24:27 INFO mapred.LocalJobRunner: Starting task: attempt_local310394148_0001_m_000000_0
24/10/06 14:24:27 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
24/10/06 14:24:27 INFO mapred.MapTask: Processing split: hdfs://master:9000/input/crawling.txt:0+1227
24/10/06 14:24:27 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
24/10/06 14:24:27 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
24/10/06 14:24:27 INFO mapred.MapTask: soft limit at 83886080

```

- (2) Process of MapReduce without any error until the end.

```

Map output bytes=2095
Map output materialized bytes=2535
Input split bytes=102
Combine input records=0
Combine output records=0
Reduce input groups=89
Reduce shuffle bytes=2535
Reduce input records=217
Reduce output records=89
Spilled Records=434
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=39
CPU time spent (ms)=0
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=242360320

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=1227

File Output Format Counters
Bytes Written=747

ubuntu@ip-172-31-82-103:~/hadoop-2.6.5/bin$

```

- (3) Screen of getting the output command and its result.

```
ubuntu@ip-172-31-82-103:~/hadoop-2.6.5/bin$ hdfs dfs -cat /output/part-r-00000
(That      2
(Without    2
(confusing  2
Against    1
Confusing   3
Confusing,  1
Consuming   1
Consuming,  1
Controlling 2
Crawling    4
Discomfort, 1
Distracting, 1
```

- **Output Interpretation:**

Explain the output data.

6. Challenges & Troubleshooting

- **Challenges Faced:** Describe any technical or conceptual difficulties.
 - **Troubleshooting Steps:** Explain how issues were resolved.
 - **Performance Observations:** Mention bottlenecks and optimization strategies used.
-

7. Summary & Key Learnings

- **Project Reflection:** Discuss key takeaways from the project.
 - **Real-World Application:** How MapReduce can be applied in industry.
 - **Future Improvements:** Suggest potential optimizations or extensions.
-

Submission Format

- **File Format:** Word document (.doc or .docx).
- **File Name:**
 Lastname_Firstname_midtermproj.doc/Lastname_Firstname_midtermproj.docx (only your name is needed)
- Under your submission, leave a comment and include your group member's name if you have one.
- Your project will automatically lose 10 points if this submission rule is violated.

Late policy

A report is late if it is not submitted to Canvas before the deadline. If you turn in your report n days late, your total point will be deducted by $(50 \times n)$ points. For example, suppose you turn in your report 1 day late (if you turn in your work after the deadline on the due date, it is also considered as 1 day late). Then, you lose $(50 \times 1) = 50$ points automatically, and your total point is 50 points. Further, suppose you lose 10 points in documentation. Thus, you receive $(50 - 10) = 40$ points in total.

For all late submissions of the report, they must be emailed to TA and cc me in the email. The email subject and the file name in the email must be the same:

`Lastname_Firstname_midtermproj.doc/Lastname_Firstname_midtermproj.docx`

Note: Each student should submit one midterm project only. If the student has submitted his/her midterm project (even incomplete) in Canvas, the student is NOT allowed to send another midterm project to me. Your project will automatically lose 80 points if this rule is violated.