

Experiment No. 7

Title: Demonstrate the Use of Map and Reduce Tasks

Objective:

To showcase the functionality and implementation of map and reduce tasks in a distributed computing environment using Hadoop MapReduce framework.

Tools used:

- Hadoop MapReduce framework
- Terminal or Command Prompt

Prerequisite:

- Installed Hadoop framework
- Basic understanding of Java programming
- Sample dataset for processing

Theory:

MapReduce is a programming model used to process and generate large datasets in a distributed computing environment. It divides tasks into two phases: the map phase, which processes input data and produces intermediate key-value pairs, and the reduce phase, which aggregates and processes intermediate results to generate the final output.



NUTAN COLLEGE OF ENGINEERING & RESEARCH (NCER)
Department of Computer Science & Engineering (CSE)

Steps to Demonstrate Map and Reduce Tasks:

Absolutely! To demonstrate the MapReduce process using Java with Hadoop, here's an example for word count:

Sure, to set up an Amazon AWS cloud instance for running Hadoop and executing MapReduce jobs, you'll need to follow these steps:

Step 1: Launch an EC2 Instance:

- Log in to your AWS Management Console and navigate to EC2.
- Launch a new instance, choosing an Amazon Machine Image (AMI) based on Linux (Amazon Linux or Ubuntu) that fits your requirements. Ensure it has enough resources (RAM, CPU, storage) for your Hadoop setup.

Instance: i-0566cf1ce2748c806 (Linux)

Details

Status and alarms New

Monitoring

Security

Networking

Storage

Tags

▼ Instance summary [Info](#)

Instance ID

i-0566cf1ce2748c806 (Linux)

IPv6 address

–

Hostname type

IP name: ip-172-31-93-56.ec2.internal

Answer private resource DNS name

–

Auto-assigned IP address

3.85.135.34 [Public IP]

Public IPv4 address

3.85.135.34 [open address](#)

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-93-56.ec2.internal

Instance type

t2.micro

VPC ID

vpc-0770b8c5ba3517c82

Private IPv4 addresses

172.31.93.56

Public IPv4 DNS

ec2-3-85-135-34.compute-1.amazonaws.com [open address](#)

Elastic IP addresses


–

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer for recommendations.

Step 2. Connect to Your EC2 Instance:

- Once the instance is running, connect to it via SSH using a terminal or an SSH client like PuTTY (for Windows). Use the key pair you generated during instance creation for authentication.

 Services [Option+S]

EC2 > Instances > i-0566cf1ce2748c806 > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-0566cf1ce2748c806 (Linux) using any of these options


EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID


 i-0566cf1ce2748c806 (Linux)

Connection Type

☒ **Connect using EC2 Instance Connect**
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.


☐ **Connect using EC2 Instance Connect Endpoint**
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address

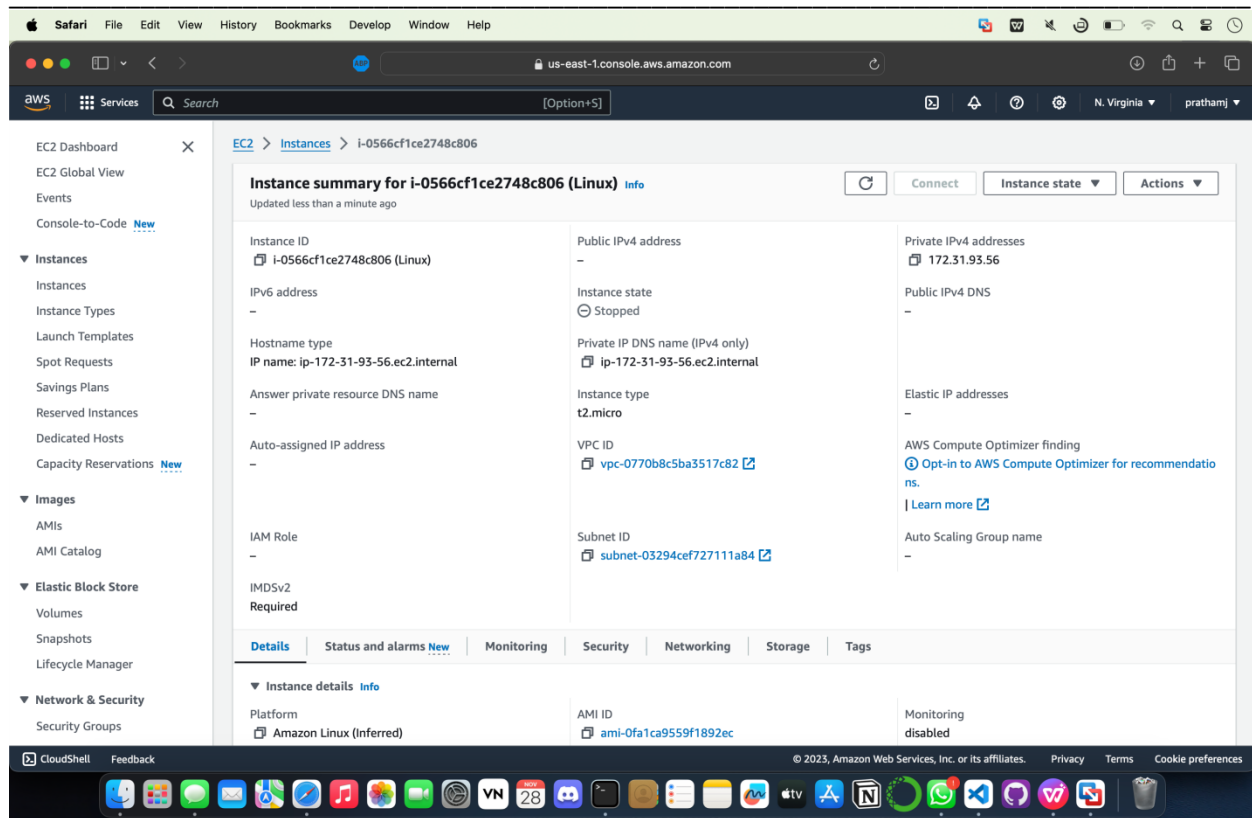
 3.85.135.34

User name

Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.

 **Note:** In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel **Connect**



The screenshot shows the AWS Management Console interface. The left sidebar contains navigation links for EC2 Dashboard, EC2 Global View, Events, Console-to-Code, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, and Security Groups. The main content area displays the 'Instance summary for i-0566cf1ce2748c806 (Linux)'. The instance is in a 'Stopped' state. Key details include: Instance ID: i-0566cf1ce2748c806 (Linux), Public IPv4 address: -, Private IPv4 addresses: 172.31.93.56, Public IPv4 DNS: -, Private IP DNS name (IPv4 only): ip-172-31-93-56.ec2.internal, Instance type: t2.micro, VPC ID: vpc-0770b8c5ba3517c82, Subnet ID: subnet-03294cef727111a84, IAM Role: -, IMDSv2: Required. Below the summary, there are tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Instance details' section shows Platform: Amazon Linux (Inferred), AMI ID: ami-0fa1ca9559f1892ec, and Monitoring: disabled. The bottom of the screen shows a macOS dock with various application icons.

Step 3. Update and Install Necessary Packages:

- Update the system: `sudo yum update` (for Amazon Linux) or `sudo apt update && sudo apt upgrade` (for Ubuntu).

NUTAN COLLEGE OF ENGINEERING & RESEARCH (NCER)
Department of Computer Science & Engineering (CSE)

Step 4. Download and Install Hadoop:

- Download Hadoop from the Apache Hadoop website or use `wget` directly on the EC2 instance.

```
Complete!
[ec2-user@ip-172-31-93-56 ~]$ wget https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
--2023-11-28 05:53:30-- https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.95.219, 2a01:4f8:10a:201a::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 605187279 (577M) [application/x-gzip]
Saving to: 'hadoop-3.3.1.tar.gz'

41% [=====] 251,043,840 14.1MB/s eta 26s

i-0566cf1ce2748c806 (Linux)
PublicIPs: 3.85.135.34 PrivateIPs: 172.31.93.56

CloudShell Feedback
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```

- Extract the downloaded Hadoop tarball: `tar -xzf hadoop-3.x.x.tar.gz` (replace `3.x.x` with your downloaded version).

```
100%[=====]
2023-11-28 05:54:12 (13.7 MB/s) - 'hadoop-3.3.1.tar.gz' saved [605187279/605187279]

[ec2-user@ip-172-31-93-56 ~]$ ls
hadoop-3.3.1.tar.gz
[ec2-user@ip-172-31-93-56 ~]$

i-0566cf1ce2748c806 (Linux)
PublicIPs: 3.85.135.34 PrivateIPs: 172.31.93.56

CloudShell Feedback
```

- Set up Hadoop environment variables like `HADOOP_HOME`, `JAVA_HOME`, and add them to the `.bashrc` or `.bash_profile` file.

Setting up environment variables like `HADOOP_HOME` and `JAVA_HOME` involves editing the `.bashrc` or `.bash_profile` file in your user's home directory on the EC2 instance. Here's a step-by-step guide to add these variables:

1. Find the Paths for Hadoop and Java:

You'll need the paths where Hadoop and Java are installed. If you've downloaded Hadoop as discussed earlier, your Hadoop path will be the directory where you extracted Hadoop. To find the Java path, you can use the ``which java`` command:

```
```bash
which java
```
```

2. Edit ``.bashrc`` or ``.bash_profile``:

Use a text editor like ``nano`` or ``vi`` to open the ``.bashrc`` or ``.bash_profile`` file:

```
```bash
nano ~/.bashrc # or nano ~/.bash_profile
```
```

3. Add Environment Variables:

Scroll to the bottom of the file and add the following lines:

```
```bash
export JAVA_HOME=/path/to/your/java # Replace '/path/to/your/java' with the Java
path you obtained

export HADOOP_HOME=/path/to/your/hadoop # Replace '/path/to/your/hadoop' with
the Hadoop path

export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/
sbin
```
```

Ensure you replace `/path/to/your/java` and `/path/to/your/hadoop` with the actual paths on your EC2 instance.

4. Save and Apply Changes:

Press `Ctrl + X`, then `Y` to save the changes in nano editor.

5. Load the Updated Configuration:

To apply the changes without logging out, use the following command:

```
```bash
source ~/.bashrc # or source ~/.bash_profile
```
```

6. Verify the Environment Variables:

To confirm that the environment variables are set correctly, you can use `echo`:

```
```bash
echo $JAVA_HOME

echo $HADOOP_HOME

```
```

```
hadoop-3.3.1.tar.gz
[ec2-user@ip-172-31-93-56 ~]$ nano ~/.bashrc      # or nano ~/.bash_profile
[ec2-user@ip-172-31-93-56 ~]$ nano ~/.bashrc      # or nano ~/.bash_profile
[ec2-user@ip-172-31-93-56 ~]$ nano ~/.bashrc      # or nano ~/.bash_profile
[ec2-user@ip-172-31-93-56 ~]$ echo $JAVA_HOME

[ec2-user@ip-172-31-93-56 ~]$ echo $HADOOP_HOME

[ec2-user@ip-172-31-93-56 ~]$ echo $JAVA_HOME

[ec2-user@ip-172-31-93-56 ~]$ echo $HADOOP_HOME

[ec2-user@ip-172-31-93-56 ~]$
```


NUTAN COLLEGE OF ENGINEERING & RESEARCH (NCER)
Department of Computer Science & Engineering (CSE)

Setting these environment variables ensures that Hadoop and Java commands can be executed from any directory in the terminal without needing to provide their full paths.

After setting these variables, you should be able to run Hadoop commands like `hadoop`, `hdfs`, etc., from any directory within your EC2 instance's terminal session.

Step 5. Configuration:

- Configure Hadoop files such as `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, and `yarn-site.xml` located in the Hadoop configuration directory (`\$HADOOP_HOME/etc/hadoop/`). Modify these files according to your cluster setup and specifications.

```
hduser@hadoop:~$ ls
'WordCount$IntSumReducer.class'
'WordCount$TokenizerMapper.class'
WordCount.class
WordCount.java
hduser@hadoop:~$ jar cf wc.jar WordCount*.class
hduser@hadoop:~$ hdfs dfs -mkdir -p input_dir
mkdir: Call From hadoop.us-central1-c.c.crow-383317.internal/10.128.0.2 to localhost:54310 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
hduser@hadoop:~$ start-all.sh
This script is deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/hadoop-2.10.2/logs/hadoop-hduser-namenode-hadoop.out
localhost: starting datanode, logging to /usr/local/hadoop/hadoop-2.10.2/logs/hadoop-hduser-datanode-hadoop.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/hadoop-2.10.2/logs/hadoop-hduser-secondarynamenode-hadoop.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/hadoop-2.10.2/logs/yarn-hduser-resourcemanager-hadoop.out
localhost: starting nodemanager, logging to /usr/local/hadoop/hadoop-2.10.2/logs/yarn-hduser-nodemanager-hadoop.out
hduser@hadoop:~$ hdfs dfs -mkdir -p input_dir
hduser@hadoop:~$ hdfs dfs -put /home/bhaskar/Desktop/test.txt input_dir
put: '/home/bhaskar/Desktop': No such file or directory
put: '/test.txt': No such file or directory
hduser@hadoop:~$ touch test.txt
hduser@hadoop:~$ nano test.txt
hduser@hadoop:~$ pwd
/home/hduser
hduser@hadoop:~$ hdfs dfs -put /home/hduser/test.txt input_dir
hduser@hadoop:~$ hadoop fs -ls input_dir
Found 1 items
-rw-r--r-- 1 hduser supergroup 68 2023-06-06 11:45 input_dir/test.txt
hduser@hadoop:~$ hadoop jar /home/hduser/wc.jar WordCount input_dir output
hduser@hadoop:~$
```

Step 6. Upload Your Code and Data:

- Transfer your Java MapReduce code (`WordCountMapper.java`, `WordCountReducer.java`, and the driver) and the sample input data file (`input.txt`) to your EC2 instance using `scp` or `sftp` from your local machine.

NUTAN COLLEGE OF ENGINEERING & RESEARCH (NCER)
Department of Computer Science & Engineering (CSE)

```

1 import java.io.IOException;
2 import java.util.StringTokenizer;
3
4 import org.apache.hadoop.conf.Configuration;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapreduce.Job;
9 import org.apache.hadoop.mapreduce.Mapper;
10 import org.apache.hadoop.mapreduce.Reducer;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13
14 /* The following class has the implementation of Mapper,Combiner,Reducer and the main method. For simplicity,
15 we will use the same class for Reducer and Combiner.
16 */
17 public class WordCount {
18
19 /* Extend the custom class with Mapper<IpKey, IpValue, OpKey, OpValue >
20 IpKey ->InputKey to mapper class (Object in most cases)
21 IpValue ->InputValue to mapper class (Text in most cases)
22 OpKey ->OutputKey from mapper class (Will be the InputKey for Reducer)
23 OpValue ->OutputValue from mapper class (Will be InputValue for Reducer)
24 */
25 public static class TokenizerMapper
26     extends Mapper<Object, Text, Text, IntWritable>{
27
28 //Create new IntWritable object
29 private final static IntWritable one = new IntWritable(1);
30
31 //Create new Text object
32 private Text word = new Text();
33
34 /*User logic is placed inside map function.
35 Output (Key,value) pair is written to context in every stage,
36 context facilitates the data movement from stage to another stage.
37 Eg: from mapper class to reducer class
38 */
39

```

Step 7. Compile Your MapReduce Code:

- Compile your Java files (`WordCountMapper.java`, `WordCountReducer.java`, and the driver) using `javac`. Ensure Hadoop libraries are included in the classpath during compilation.

Step 8. Run Your MapReduce Job:

- Execute your MapReduce job using the Hadoop command-line interface (`hadoop jar` command). Ensure to provide the correct input and output paths.

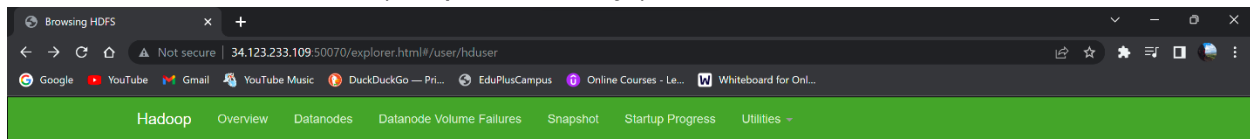
Step 9. Monitor Job Execution:

NUTAN COLLEGE OF ENGINEERING & RESEARCH (NCER)
Department of Computer Science & Engineering (CSE)

- Use Hadoop's command-line tools (`yarn` or `mapred` commands) or access the Hadoop web interface to monitor the job's progress and status.

Step 10. Review Output:

- Once the job completes, check the output directory specified in your Hadoop job submission for the results (`output_directory`).



Browse Directory

/user/hduser

Go!

Show

25

entries

Search:

| <input type="checkbox"/> | <div>Permission</div> | <div>Owner</div> | <div>Group</div> | <div>Size</div> | <div>Last Modified</div> | <div>Replication</div> | <div>Block Size</div> | <div>Name</div> | <div></div> |
|--------------------------|-----------------------|------------------|------------------|-----------------|--------------------------|------------------------|-----------------------|-----------------|-------------|
| <input type="checkbox"/> | drwxr-xr-x | hduser | supergroup | 0 B | Jun 06 17:15 | 0 | 0 B | input_dir | |
| <input type="checkbox"/> | drwxr-xr-x | hduser | supergroup | 0 B | Jun 06 17:17 | 0 | 0 B | output | |

Showing 1 to 2 of 2 entries

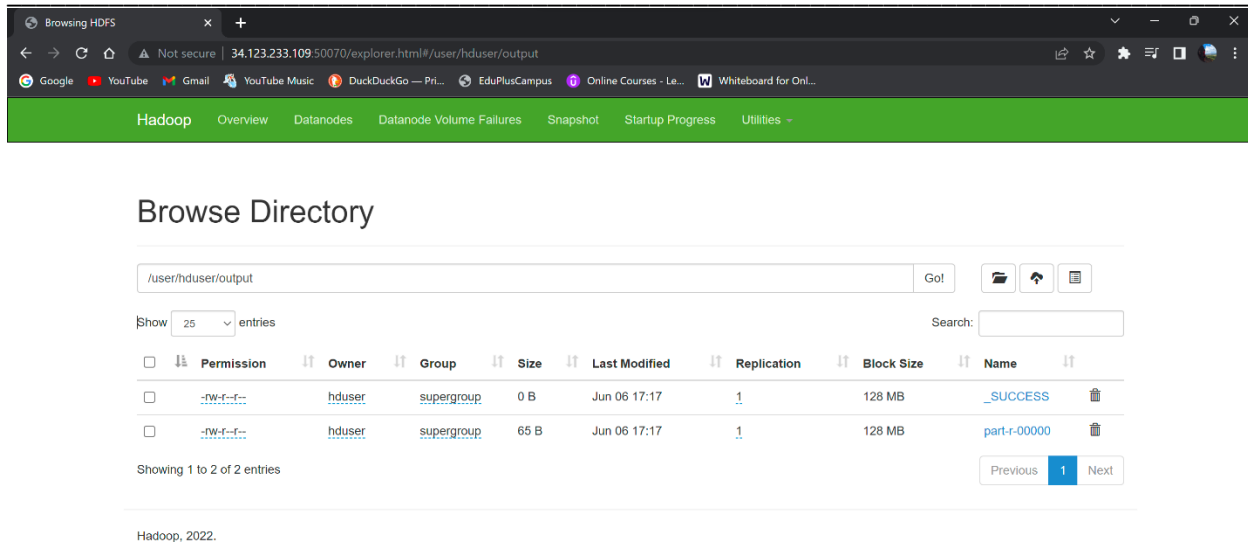
Previous

1

Next

Hadoop, 2022.

Success



Browse Directory

/user/hduser/output

Go!

Show 25 entries

Search:

| Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
|------------|--------|------------|------|---------------|-------------|------------|------------------------------|
| -rw-r--r-- | hduser | supergroup | 0 B | Jun 06 17:17 | 1 | 128 MB | _SUCCESS |
| -rw-r--r-- | hduser | supergroup | 65 B | Jun 06 17:17 | 1 | 128 MB | part-r-00000 |

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2022.

Remember, configuring Hadoop on AWS EC2 involves setting up security groups, configuring ports, and potentially additional settings for AWS-specific networking and storage services. Also, manage your EC2 instance security by restricting access through security groups and using key pairs for SSH access.

Be cautious with AWS resources to avoid unnecessary charges, especially when running instances or services. Adjust the instance type and configurations based on your processing needs and budget.

Conclusion:

Successfully demonstrated the use of map and reduce tasks in a distributed computing environment using the Hadoop MapReduce framework. This experiment highlighted the process of dividing computational tasks into smaller map and reduce phases, showcasing how Hadoop efficiently processes large datasets by leveraging parallel processing and distributed computing techniques.