LAB EXERCISES 2

Aim:

(i) Implement the following file management tasks in Hadoop:

1. Adding files and directories
2. Retrieving files
3. Deleting files

To implement the three file management tasks in Hadoop such as Adding files and directories, retrieving files and Deleting files.

Procedure:

1. Utilize hadoop fs **-mkdir** command to add a directory.
   ```
   >>>hadoop fs -mkdir <hdfs_directory_path>
   ```
2. Utilize hadoop fs **-put** command to add a file, progress updates will be displayed as the file being copied to HDFS. If the file is successfully added, no error message will be shown.
   ```
   >>>hadoop fs -put <input_file_location> <directory_path>
   ```
3. Enter hadoop fs **-get** command to retrieve the file, progress updates will be displayed as the file is being retrieved from HDFS. If the file is successfully retrieved, no error message will be shown.
   ```
   >>>hadoop fs -get <input_file_location> <output_path_location>
   ```
4. Enter hadoop fs **-rm** command to delete a file, if there is an issue, an error message will be displayed.
   ```
   >>>hadoop fs -rm <file_path>
   ```

Commands:

- To create the directory

  ```
  >>> hadoop fs -mkdir /Priyanshu_exp2
  ```

- To copy the directory

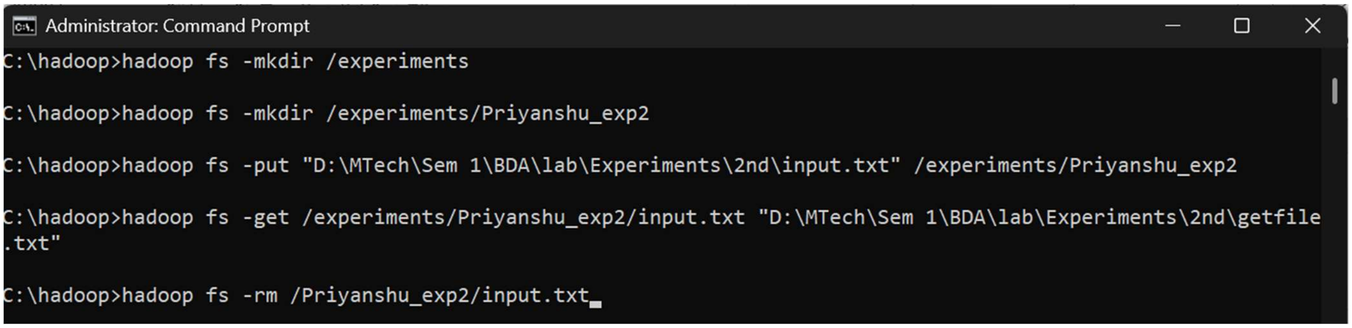
  ```
  >>> hadoop fs -put "D:\NIT-J\BDA Lab + Class\EXP 2\input.txt" /Priyanshu_exp2
  ```

- To retrieve the files and directory

  ```
  >>> hadoop fs -get /Priyanshu_exp2/input.txt "D:\NIT-J\BDA Lab + Class"
  ```

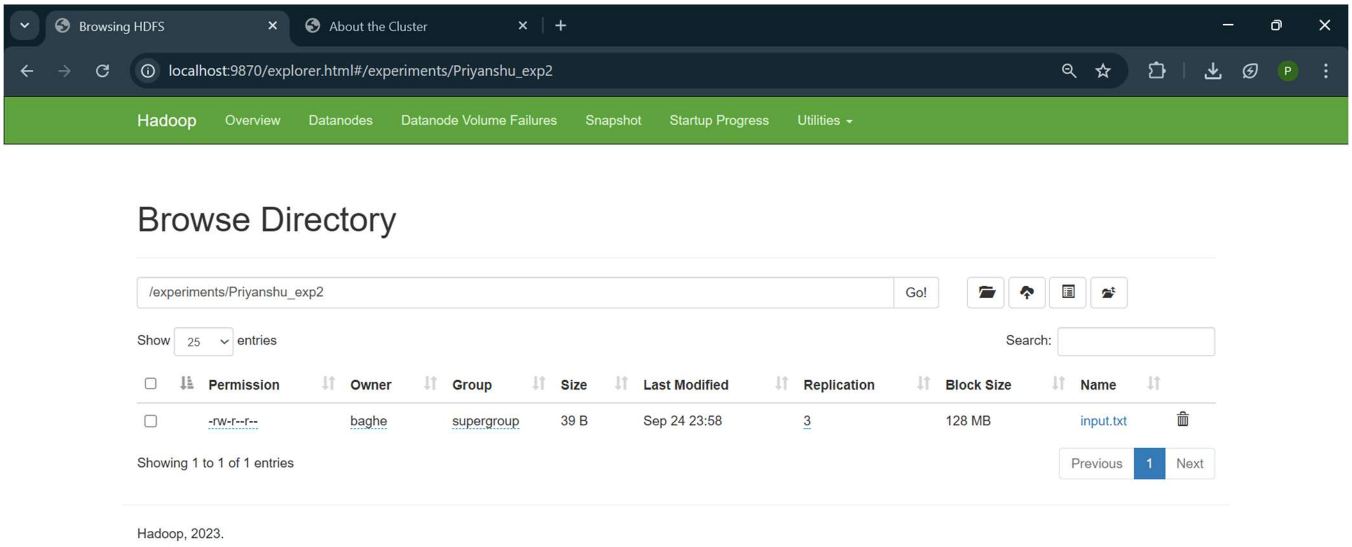
- To remove files and directory

  ```
  >>> hadoop fs -rm /Priyanshu_exp2/input.txt
  ```



```
C:\hadoop>hadoop fs -mkdir /experiments

C:\hadoop>hadoop fs -mkdir /experiments/Priyanshu_exp2

C:\hadoop>hadoop fs -put "D:\MTech\Sem 1\BDA\lab\Experiments\2nd\input.txt" /experiments/Priyanshu_exp2

C:\hadoop>hadoop fs -get /experiments/Priyanshu_exp2/input.txt "D:\MTech\Sem 1\BDA\lab\Experiments\2nd\getfile
.txt"

C:\hadoop>hadoop fs -rm /Priyanshu_exp2/input.txt
```

Snapshot No. 3 (Commands)



Browse Directory

/experiments/Priyanshu_exp2

☐	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	baghe	supergroup	39 B	Sep 24 23:58	3	128 MB	input.txt

Showing 1 to 1 of 1 entries

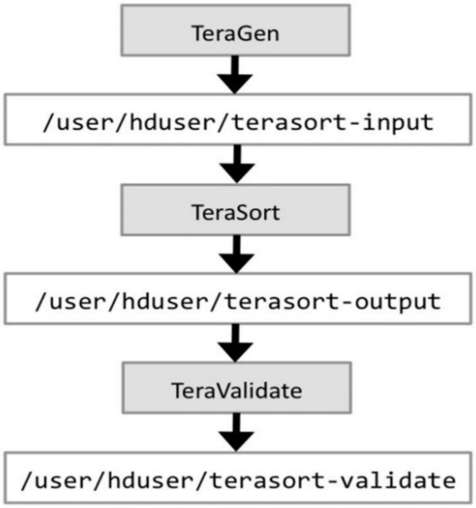
Hadoop, 2023.

Snapshot No. 4 (Browse HDFS at Result)

(ii) Benchmark and stress test an Apache Hadoop cluster

The benchmark measures the number of operations performed by the name-node per second. Specifically, for each operation tested, it reports the total running time in seconds (Elapsed Time), operation throughput (Ops per sec), and average time for the operations (Average Time). The higher, the better.

Steps to perform benchmark and test in Hadoop:


TeraGen
/user/hduser/terasort-input
TeraSort
/user/hduser/terasort-output
TeraValidate
/user/hduser/terasort-validate

Bench marking in Hadoop involves the systematic measurement and comparison of different aspects of the Hadoop system's performance against established standards or other systems. It helps in understanding how well the Hadoop cluster performs under various workloads and conditions. Bench marking typically involves running standardized tests or workloads on the Hadoop cluster and analysing metrics such as throughput, latency, resource utilization, and scalability. Bench marking in Hadoop includes:

Workload characterization: 1.Benchmark

selection 2.Execution

3.Measurement and analysis

4.Stress Testing

Stress testing, also known as load testing, involves evaluating the behaviour of a Hadoop cluster under extreme conditions to assess its resilience, stability, and scalability. The goal of stress testing is to push the system beyond its normal operating limits to identify failure points, bottlenecks, and weaknesses

To generate test code:

```
>>> hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce- examples-3.3.6.jar teragen <no._of_rows> <test_directory>
```

```
>>> hadoop  jar  hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.3.6.jar teragen 10000 /Test
```

To Sort generated code:

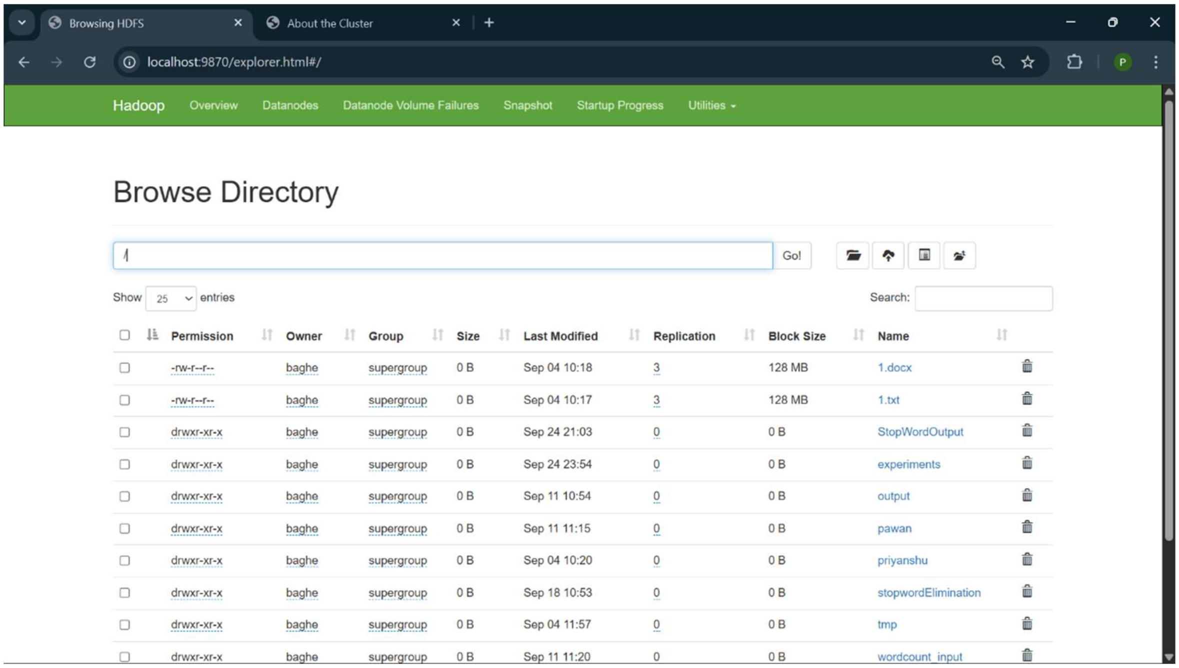
```
>>> hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce- examples-3.3.6.jar terasort <test_directory><sort_directory
```

```
>>> hadoop  jar  hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.3.6.jar terasort /Test /Sort
```

To validate the code:

```
>>> hadoop jar $HADOOP_HOME/hadoop-3.2.3/share/hadoop/
mapreduce/hadoop- mapreduce-examples-3.3.6.jar teravalidate <sort_directory>
<Validate_directory
```

```
>>> hadoop  jar  hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.3.6.jar teravalidate /Sort /val
```



Browse Directory

/

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	baghe	supergroup	0 B	Sep 04 10:18	3	128 MB	1.docx
□	-rw-r--r--	baghe	supergroup	0 B	Sep 04 10:17	3	128 MB	1.txt
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 24 21:03	0	0 B	StopWordOutput
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 24 23:54	0	0 B	experiments
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 11 10:54	0	0 B	output
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 11 11:15	0	0 B	pawan
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 04 10:20	0	0 B	priyanshu
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 18 10:53	0	0 B	stopwordElimination
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 04 11:57	0	0 B	tmp
□	drwxr-xr-x	baghe	supergroup	0 B	Sep 11 11:20	0	0 B	wordcount_input

Snapshot No. 5 (Browse HDFS at Result)

Result: Thus, the commands for file management, bench marking and stress test in Hadoop environment is written and executed successfully.