P.2 File Management in Hadoop and Benchmarking.

Ans → 1. File Management Task:-

- Adding files and directories.
- Retrieving files
- Deleting files.

2. Benchmarking & stress Testing an Apache Hadoop cluster. We will be performing these task on Hadoop running in Pseudo-distributed mode on a Windows environment, using WSL. Make sure Hadoop is already installed & configured.

Part(i): Implementing File Management Task in Hadoop

1. Adding Files and Directories to HDFS:-

To add files and directories to the Hadoop Distributed file System (HDFS), we use the hadoop fs command.

- Create a Directory in HDFS: To create a directory in HDFS, use the following command

- hadoop fs -mkdir /user/hadoop/new_directory

- Copy a local File to HDFS:- if you want to add file from your local file system to HDFS. Use the -copyfromLocal command

- hadoop fs -copyfromLocal /path/to/local/file.txt /user/hadoop/new_directory/

2. Retrieving files from HDFS:-
you can retrive files from HDFS back to your local file system using the -get or -copyTo Local command.

- Retrive a file from HDFS:-
  - hadoop fs -get /user/hadoop/new_directory/path /to/local/destination/

- List the contents of a directory in HDFS:-
To list the contents of a directory in HDFS, you can use the -ls command

  - hadoop fs -ls /user/hadoop/new_directory

3. Deleting Files & Directories from HDFS
To delete file and directories from HDFS, use the -rm and -r flags.

- Delete a file from HDFS:
  -> hadoop fs-rm /user/hadoop/new_directory/file.txt

- Delete a Directory & all its Contents:
  -> hadoop fs-rm -r /user/hadoop/new_directory

Part (ii): Benchmarking and stress Testing an Apache Hadoop Cluster

1. Benchmarking Hadoop Using the terasort Benchmark

-> Apache Hadoop comes with a built-in benchmark called Terasort, which is a sorting benchmark that tests the scalability and performance of your Hadoop cluster.

Step 1: Generate Data for the Benchmark

- You need to generate some test data before running the benchmark.
- first, create a directory to store the test data:
  - hadoop fs -mkdir /user/hadoop/terasort-input
  - hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar terasort 10000000 /user/hadoop/terasort-output

Step 2: Run the Benchmark

- The terasort command runs the sorting process and place the output in /user/hadoop/tera-output.

- hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar terasort /user/hadoop/terasort-input /user/hadoop/terasort-output

Step 3: Monitor the Benchmarking Job

1. Monitor via Web UI:
   - YARN Resource Manager: http://localhost:8088/ to monitor job progress & performance.
   - HDFS NameNode web UI: http://localhost:50070/ to monitor HDFS usage and health.

2. Job Completion:
   - Once the job completes, you will see the sorted output in the /user/hadpood/terasort_output directory.
   - You can verify the ouput running:
   - hadoop fs -ls /user/hadoop/terasort_output

2. Stress Testing Hadoop Cluster:

Stress testing is important to evaluate the stability and performance of the Hadoop cluster under heavy load. You can use multiple methods to stress Hadoop. such as running multiple MapReduce job. adding large datasets.

Step 1: Run Multiple Concurrent Jobs.

- Generate multiple datasets:

-> hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-example*.jar teragen 10000000 /user/hadoop/stress_input-1

-> hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-example*.jar teragen 10000000 /user/hadoop/stress_input-2

-> hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-example*.jar teragen 10000000 /user/hadoop/stress_input-3

- Run multiple sort operation on each dataset Concurrently.

-> hadoop jar $Hadoop_HOME/~~stress~~ share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar terasort /user/hadoop/stress_input-1 /user/hadoop/stress_output-1 &

-> hadoop jar $Hadoop_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar terasort /user/hadoop/stress_input-2 /user/hadoop/stress_output-2 &

-> hadoop jar $Hadoop_HOME/share/hadoop/mapreduce/hadoop-mapreduce-example-*.jar terasort /user/hadoop/stress_input-3 /user/hadoop/stress_output-3 &

→ Step 2: Monitor system performance

- Use the Yarn Resource Manager UI: http://localhost:8088/ to monitor the job status and resource consumption (CPU)
- check the HDFS NameNode UI: http://localhost:50070/ for information on HDFS health and block usage.

→ Step 3: Track the Hadoop cluster's Health & Performance

1. HDFS Web UI: check the usage, file system health, and block discription.
2. Resource Manager: View the status of running job, CPU, memory usage, and overall job completion statistics.

→ Step 4: Evaluate the Results:-

- After Completing the benchmark & stress test job, check the Hadoop logs to assess the overall performance & identify bottlenecks. Hadoop logs can be found in the following directory:
- /usr/local/hadoop/logs
- The logs can help you identify areas where the system may be underperforming or experiencing issue, such as slow disk I/O, network bottlenecks or insufficient memory.

→ Conclusion :-

By following the above steps, you have successfully performed the following tasks:

1. Added files and directories to HDFS.
2. Retrived and deleted files from HDFS.
3. Run benchmarking tasks using the Terasort example to assess Hadoop's performance.
4. Conducted a stress test by running multiple jobs concurrently and monitoring system performance.