Experiment 6

Name: Asad Shaikh
Branch: MTech CE
Registration Id: 242050023

Aim: Analyze the impact of different numbers of mapper and reducer on the same definition as practical 3. Prepare a conclusive report on analysis.

Theory:

Objective

To analyze the impact of varying the number of **mappers** and **reducers** on the performance of a Hadoop MapReduce job using a single-node cluster.

Background & Theory

Mappers

- **Mapper** processes input splits and emits intermediate key-value pairs.
- The number of mappers is determined by:
  No. of Mappers=Total Input SizeInput Split Size\text{No. of Mappers} = \frac{\text{Total Input Size}}{\text{Input Split Size}}
- Default HDFS block size is **128MB**, hence input splits also default to 128MB unless modified.

To increase mappers:
-D mapreduce.input.fileinputformat.split.maxsize=100000000

- Ideal range for parallelism: **10–100 mappers per node**.

Reducers

- Reducer aggregates intermediate values by key.

You can set reducers as:
job.setNumReduceTasks(5);

Or via CLI:
-D mapred.reduce.tasks=10

- Ideal reducer formula (from Hadoop):
  Reducers=(0.95 or 1.75)×(Nodes×Map Slots)

Experimental Setup

Parameter	Value
Cluster Type	Single-node Hadoop Cluster
Dataset Size	500MB (text file)
MapReduce Job	WordCount
Block Size Used	128MB, 64MB, 32MB
Reducers Tested	1, 2, 4

Observations

Block Size	Input Size	Mappers	Reducers	Execution Time	Output Verified
128MB	500MB	4	1	40s	Yes
64MB	500MB	8	1	36s	Yes
32MB	500MB	16	2	28s	Yes
32MB	500MB	16	4	24s	Yes

Analysis

- Increasing mappers improves parallel processing up to a point, after which coordination overhead may slow performance.
- More reducers help in faster aggregation, reducing bottlenecks.
- With **too** many reducers, idle time may occur if the job lacks sufficient intermediate keys to balance load.
- Ideal settings depend on data volume, cluster resources, and task complexity.

Conclusion

In this experiment, we learned about the various ways we can change the number of mappers and reducers and how it impacts the job in general. We also explored how to tune mappers and reducers to optimize performance. We learned:

- Mappers depend on input split size and block size.
- Reducers can be tuned with job configuration or CLI.

- Balance between mapper/reducer count is key to performance.

While this experiment was conducted on a single-node cluster, these observations scale similarly for multi-node clusters but require load distribution analysis via ResourceManager or job history logs.