Experiment No. 7

Title: Demonstrate the use of map and reduce tasks.

Objective: The objective of this experiment is to gain a comprehensive understanding of MapReduce tasks in cloud computing and to demonstrate their practical application.

Tools Used: Hadoop, Java

Prerequisite: Basic understanding of Java programming and Hadoop framework.

Theory: MapReduce is a programming model that allows for distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. The model is inspired by the map and reduce functions commonly used in functional programming.

MapReduce involves two important tasks - Map and Reduce.

Map Task: The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Essentially, the map function is applied in parallel to every pair in the input dataset. This produces a list of pairs for each call. After that, MapReduce sorts and groups the pairs based on the keys to produce a single key and a list of all the values that share that key.

Reduce Task: In the Reduce task, the reduce function is applied in parallel to each group, which in turn produces a collection of values in the same domain. Each reduce function processes the key and the corresponding list of values to yield output tuples.

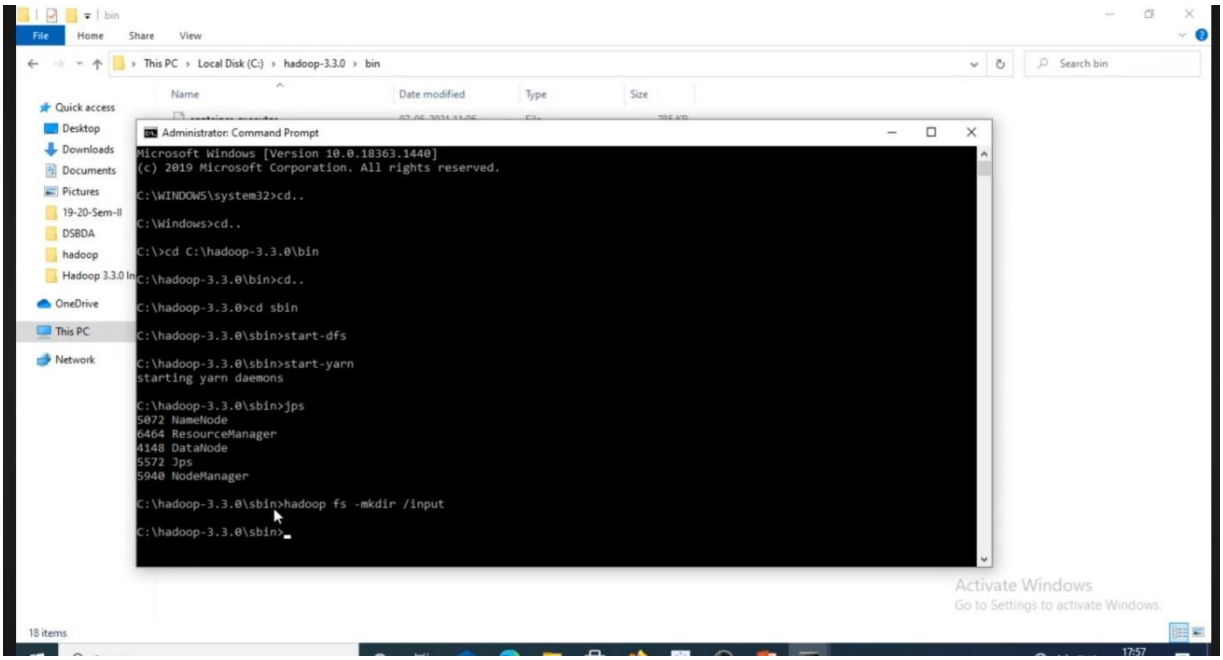
MapReduce programs are designed to compute large volumes of data in a parallel fashion. This requires dividing the dataset into independent chunks that can be processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

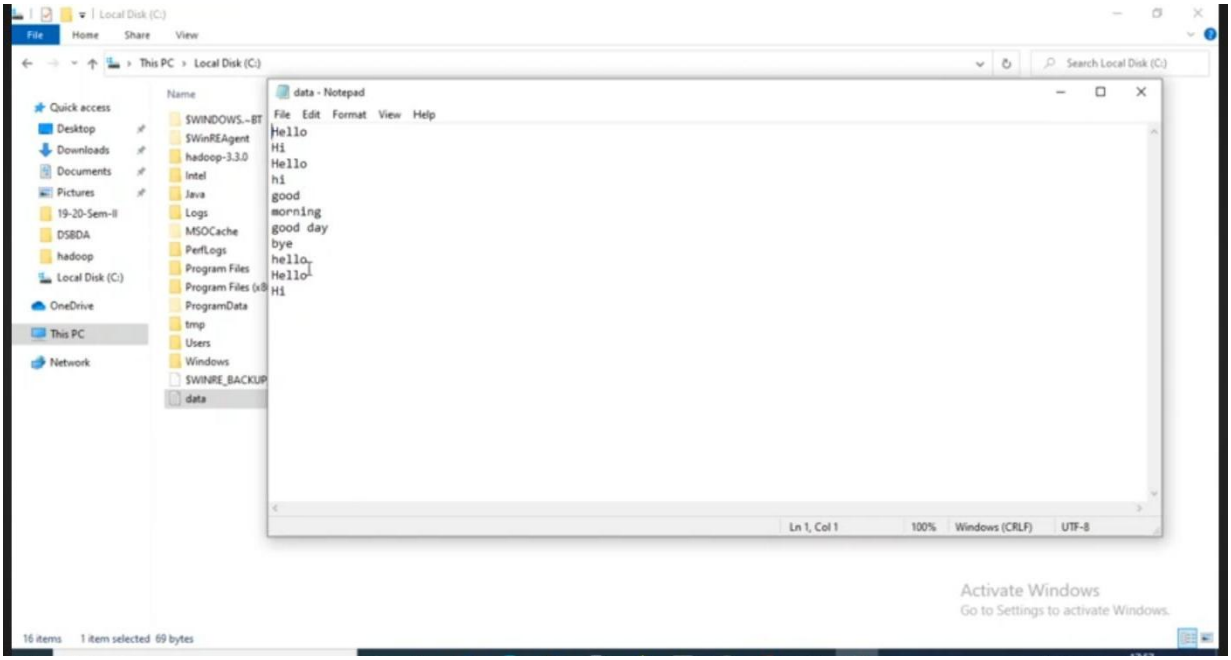
MapReduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel – though in practice this is limited by the number of independent data sources and/or the number of CPUs near each source. This applies to the reduce operations as well, and a minimal form of synchronization is required at the end, leading to substantial gains in speed. MapReduce is thus a good example of the divide and conquer algorithmic paradigm.

Steps to perform the experiment:

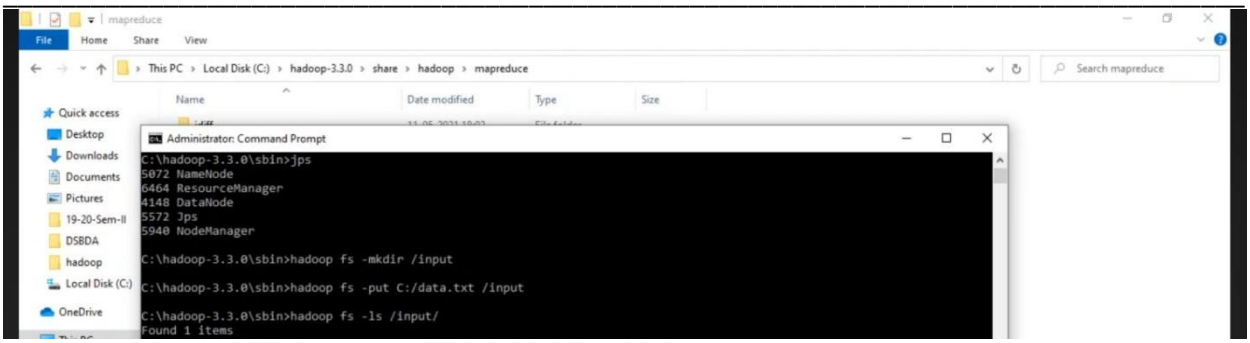
1. Start Hadoop Services.



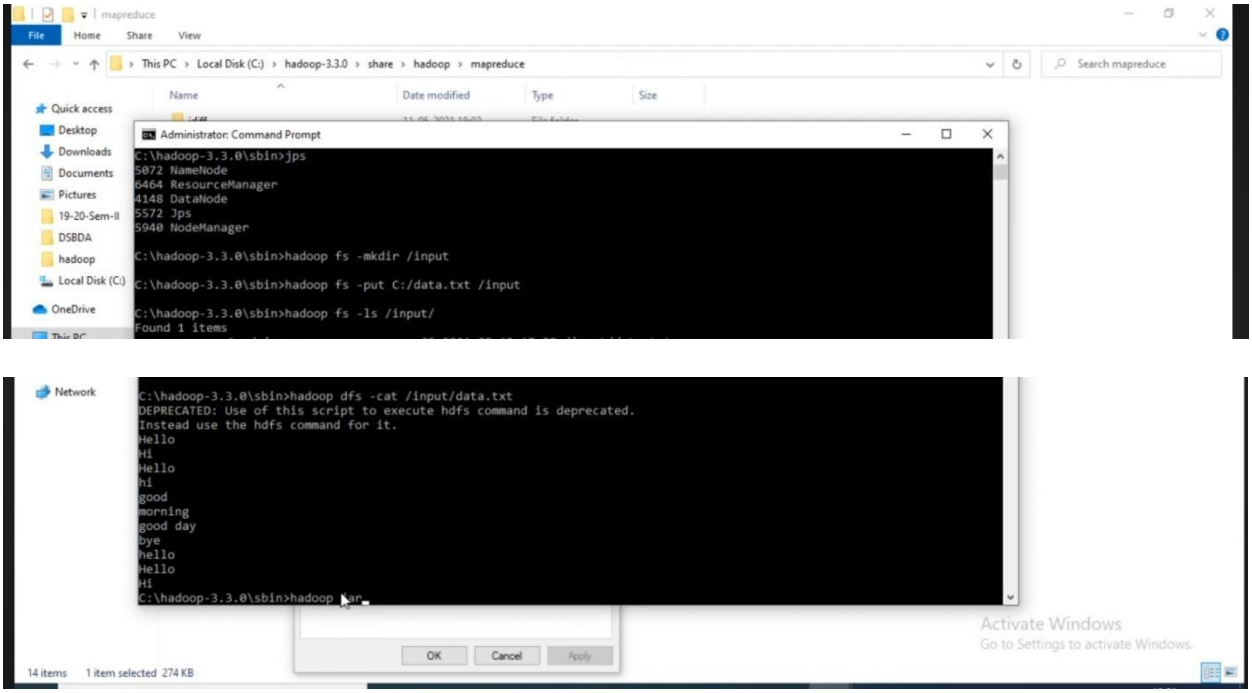
2. Prepare input file(s) for the Wordcount program.



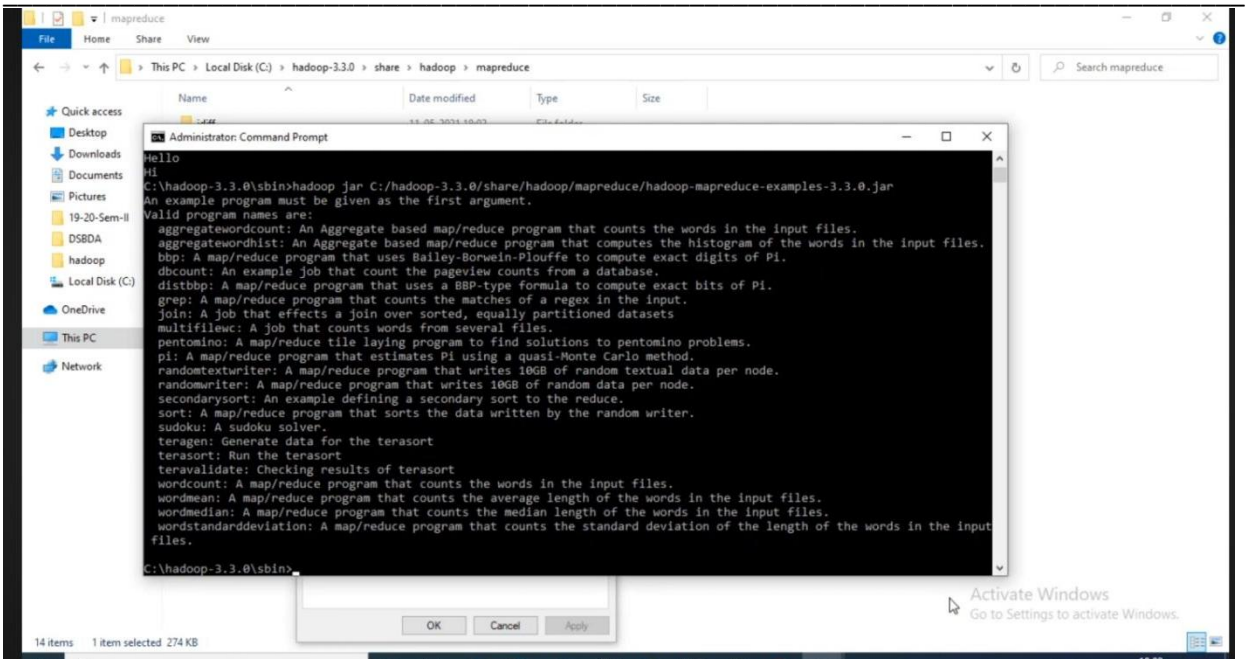
3. Interact with file(s) and directories in HDFS.



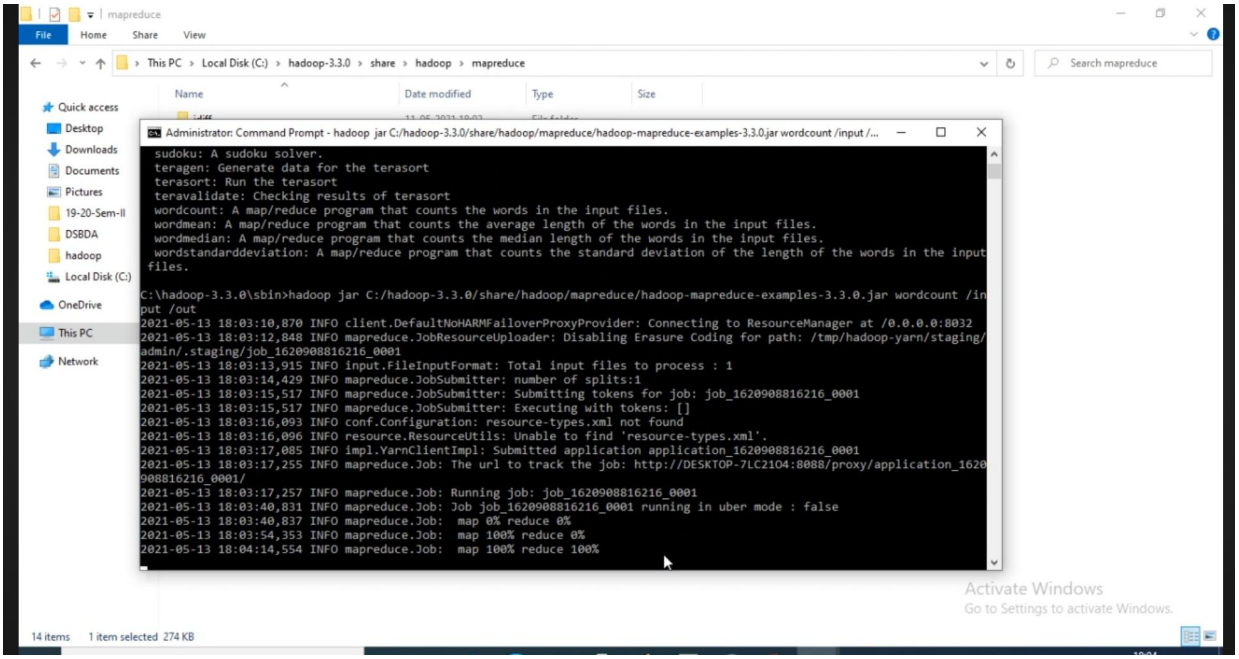
4. Read the Contents of a File in HDFS.

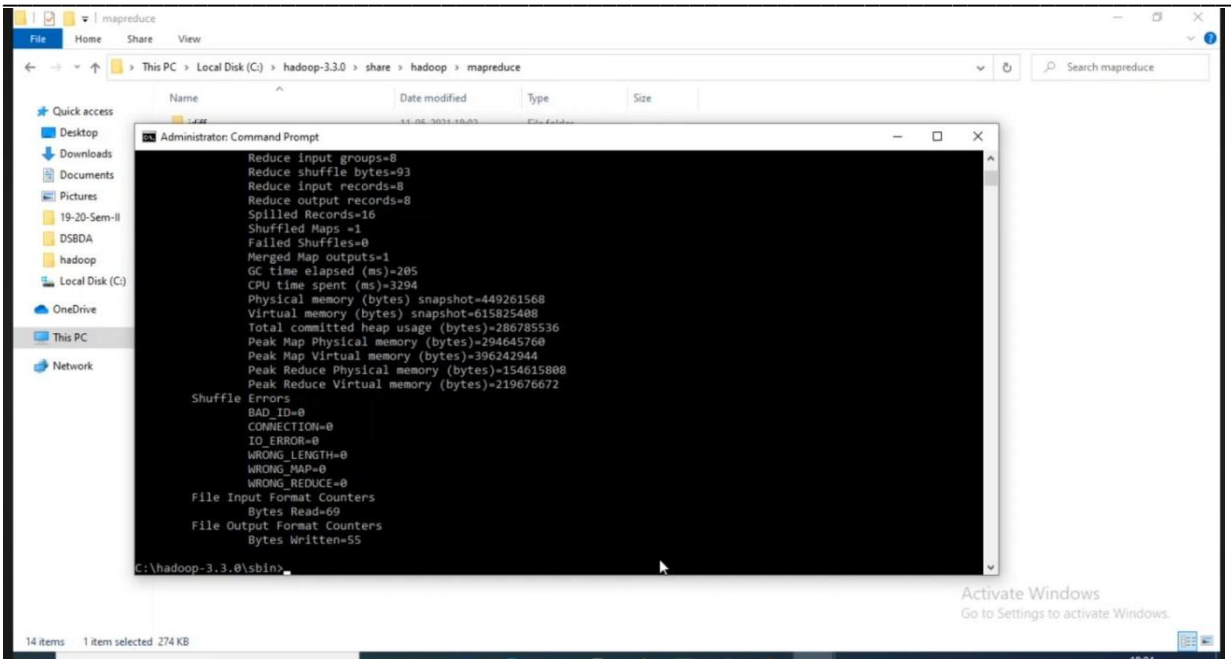


5. Run the the JAR file containing Hadoop example programs, including WordCount.

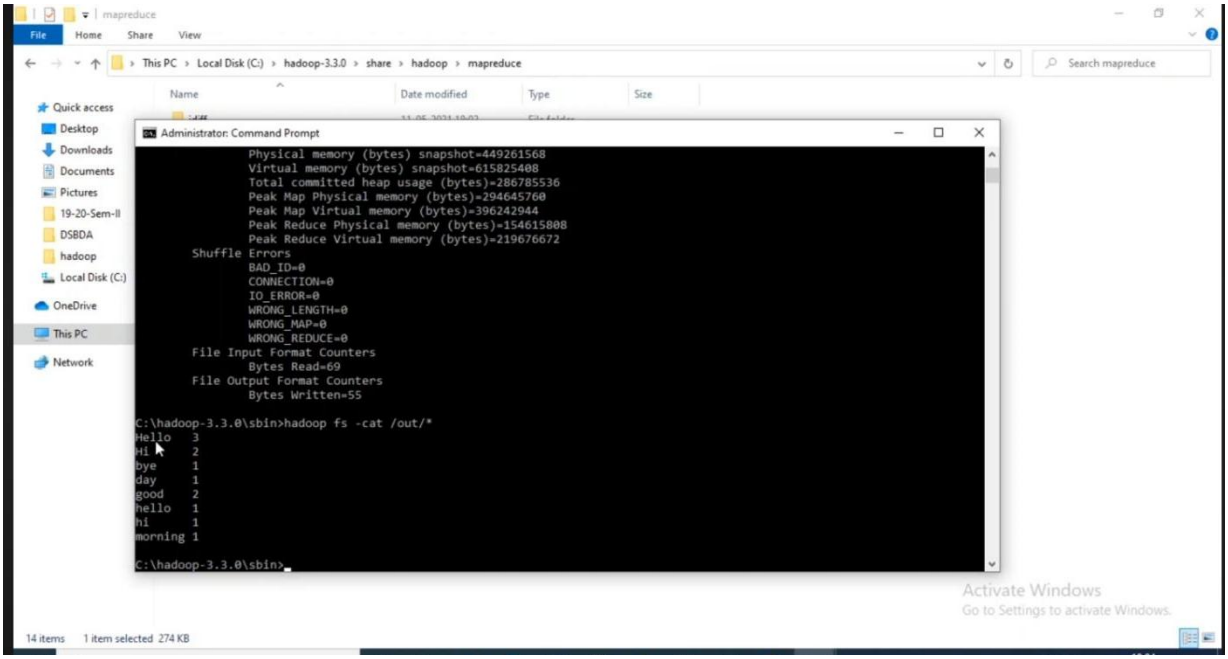


6. Submit the MapReduce job to the Hadoop cluster.





7. View the output of the Wordcount program.



Conclusion: Through this experiment, we have successfully demonstrated the use of Map and Reduce tasks in cloud computing using Hadoop. This experiment helps in understanding the fundamental operations of MapReduce and its application in real-world scenarios.