

BDA Experiment

Aim : Implement application for counting frequency of words in a text file using MapReduce.

Objective: To implement mapreduce.

Methodology:

What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

Generally MapReduce paradigm is based on sending the computer to where the data resides!

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Map stage – The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage – This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

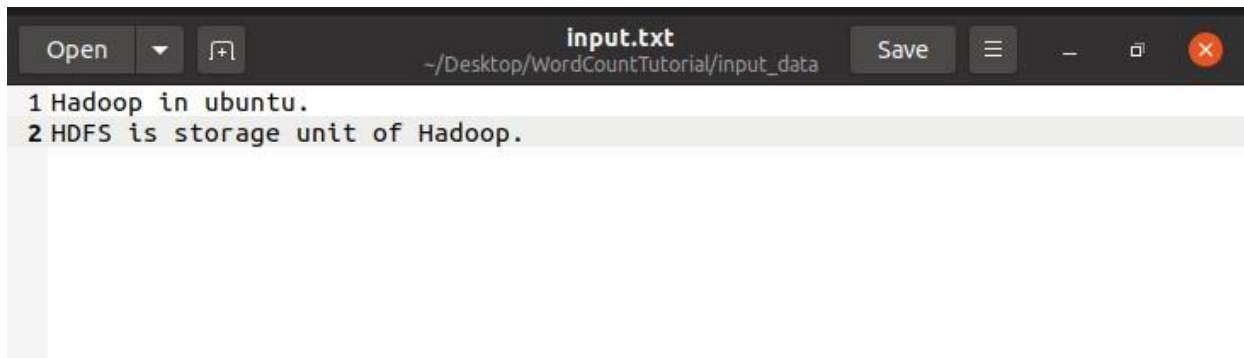
Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

Tools Used: Hadoop, text editor, java

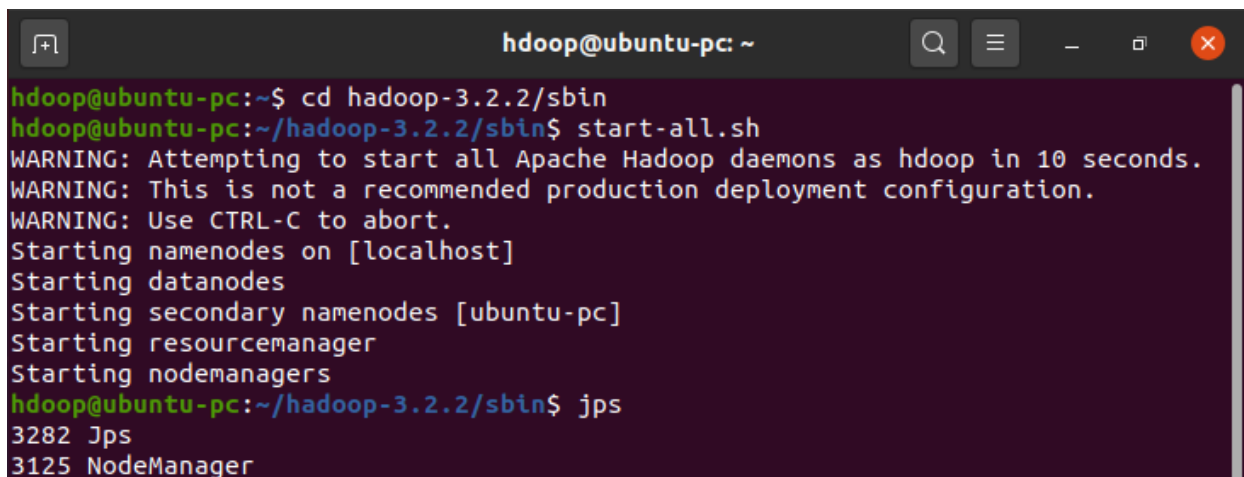
Actual Work Done:

Create a text file



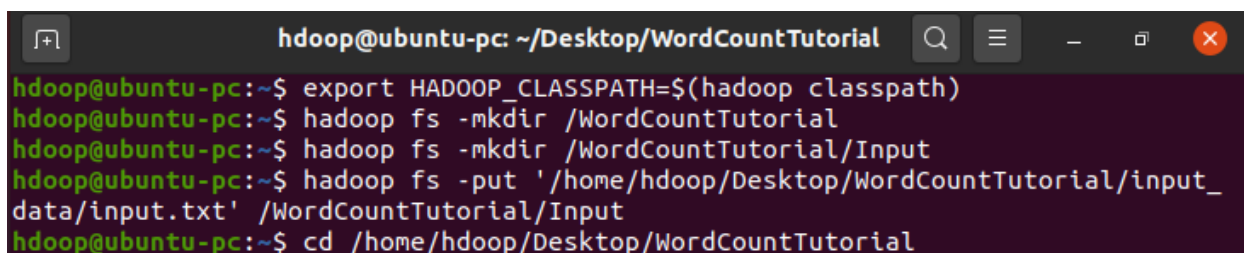
```
Open  input.txt  Save  ~/Desktop/WordCountTutorial/input_data
1 Hadoop in ubuntu.
2 HDFS is storage unit of Hadoop.
```

Start the hadoop cluster



```
hdoop@ubuntu-pc: ~$ cd hadoop-3.2.2/sbin
hdoop@ubuntu-pc:~/hadoop-3.2.2/sbin$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hdoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu-pc]
Starting resourcemanager
Starting nodemanagers
hdoop@ubuntu-pc:~/hadoop-3.2.2/sbin$ jps
3282 Jps
3125 NodeManager
```

Set the classpath, make the necessary directories and make a jar file



```
hdoop@ubuntu-pc: ~$ export HADOOP_CLASSPATH=$(hadoop classpath)
hdoop@ubuntu-pc:~$ hadoop fs -mkdir /WordCountTutorial
hdoop@ubuntu-pc:~$ hadoop fs -mkdir /WordCountTutorial/Input
hdoop@ubuntu-pc:~$ hadoop fs -put '/home/hdoop/Desktop/WordCountTutorial/input_data/input.txt' /WordCountTutorial/Input
hdoop@ubuntu-pc:~$ cd /home/hdoop/Desktop/WordCountTutorial
```

```

hadoop@ubuntu-pc:~/Desktop/WordCountTutorial$ javac -classpath ${HADOOP_CLASSPATH} -d '/home/hadoop/Desktop/WordCountTutorial/tutorial_classes' '/home/hadoop/Desktop/WordCountTutorial/WordCount.java'
hadoop@ubuntu-pc:~/Desktop/WordCountTutorial$ jar -cvf firstTutorial.jar -C tutorial_classes/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1739) (out= 739)(deflated 57%)
adding: WordCount.class(in = 1491) (out= 814)(deflated 45%)
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 754)(deflated 56%)
hadoop@ubuntu-pc:~/Desktop/WordCountTutorial$ hadoop jar '/home/hadoop/Desktop/WordCountTutorial/firstTutorial.jar' WordCount /WordCountTutorial/Input /WordCountTutorial/Output
2022-04-03 22:15:30,231 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-04-03 22:15:32,064 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2022-04-03 22:15:32,245 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1649002662853_0001
2022-04-03 22:15:33,014 INFO input.FileInputFormat: Total input files to process : 1
2022-04-03 22:15:33,958 INFO mapreduce.JobSubmitter: number of splits:1
2022-04-03 22:15:34,945 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1649002662853_0001
2022-04-03 22:15:34,947 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-04-03 22:15:35,677 INFO conf.Configuration: resource-types.xml not found
2022-04-03 22:15:35,678 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-03 22:15:36,621 INFO impl.YarnClientImpl: Submitted application application_1649002662853_0001
2022-04-03 22:16:26,717 INFO mapreduce.Job: Job job_1649002662853_0001 completed successfully
2022-04-03 22:16:26,942 INFO mapreduce.Job: Counters: 54
    File System Counters
        FILE: Number of bytes read=110
        FILE: Number of bytes written=468697
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=170
        HDFS: Number of bytes written=68
        HDFS: Number of read operations=8
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
        HDFS: Number of bytes read erasure-coded=0
    Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=8751
        Total time spent by all reducers in occupied slots (ms)=7006

```

Get the output

```
hadoop@ubuntu-pc: ~/Desktop/WordCountTutorial
Peak Map Physical memory (bytes)=219385856
Peak Map Virtual memory (bytes)=2479849472
Peak Reduce Physical memory (bytes)=116953088
Peak Reduce Virtual memory (bytes)=2488541184
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=50
File Output Format Counters
  Bytes Written=68
hadoop@ubuntu-pc:~/Desktop/WordCountTutorial$ hadoop dfs -cat /WordCountTutorial
/Output/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

HDFS      1
Hadoop    1
Hadoop.   1
in         1
is         1
of         1
storage   1
ubuntu.   1
unit       1
hadoop@ubuntu-pc:~/Desktop/WordCountTutorial$
```

Conclusion/Outcome : Thus I have carried out Mapreduce for word count successfully.