

Exp.No: 2

Roll No.:210701301

Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm

AIM:

To run a basic Word Count Map Reduce program.

Procedure:

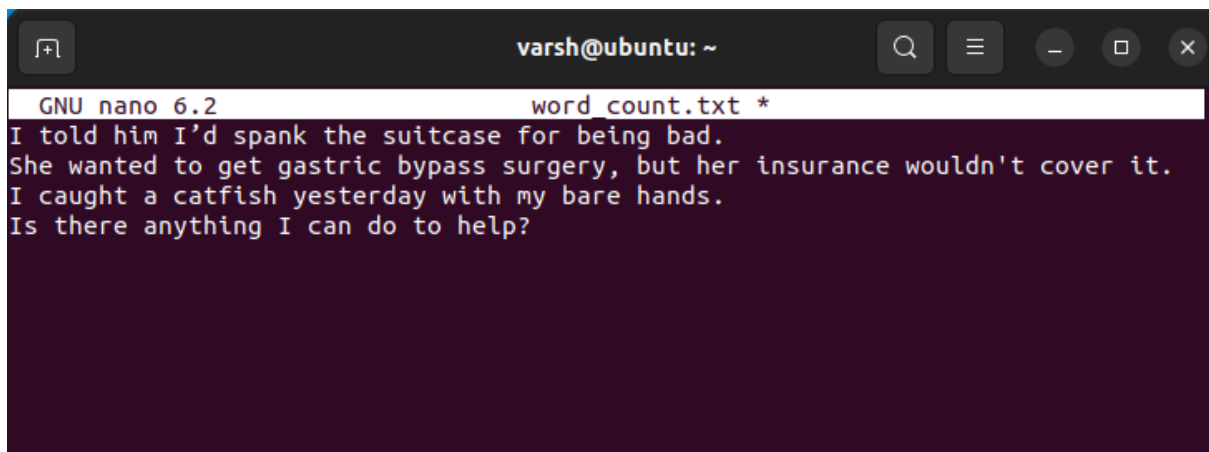
Step 1: Create Data File:

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyse.

Login with your hadoop user.

nano word_count.txt

Output: Type the below content in word_count.txt



```
varsh@ubuntu: ~  
GNU nano 6.2 word_count.txt *  
I told him I'd spank the suitcase for being bad.  
She wanted to get gastric bypass surgery, but her insurance wouldn't cover it.  
I caught a catfish yesterday with my bare hands.  
Is there anything I can do to help?
```

Step 2: Mapper Logic - mapper.py:

Create a file named "mapper.py" to implement the logic for the mapper. The mapper will read input data from STDIN, split lines into words, and output each word with its count.

nano mapper.py

Copy and paste the mapper.py code

```
#!/usr/bin/env python3
```

```
# import sys because we need to read and write data to STDIN and STDOUT
```

```
#!/usr/bin/python3
```

```
import sys
```

```
for line in sys.stdin:
```

```
    line = line.strip() # remove leading and trailing whitespace
```

```
    words = line.split() # split the line into words
```

```
    for word in words:
```

```
print( '%s\t%s' % (word, 1))
```

Step 3: Reducer Logic - reducer.py:

Create a file named "reducer.py" to implement the logic for the reducer. The reducer will aggregate the occurrences of each word and generate the final output.

nano reducer.py

Copy and paste the reducer.py code

reducer.py

```
#!/usr/bin/python3 from operator
```

```
import itemgetter
```

```
import sys
```

```
current_word = None
```

```
current_count = 0
```

```
word = None
```

```
for line in sys.stdin: line = line.strip()
```

```
word, count = line.split('\t', 1)
```

```
try:
```

```
count = int(count)
```

```
except ValueError:
```

```
continue
```

```
if current_word == word:
```

```
current_count += count else:
```

```
if current_word:
```

```
print( '%s\t%s' % (current_word, current_count))
```

```
current_count = count current_word = word if current_word == word:
```

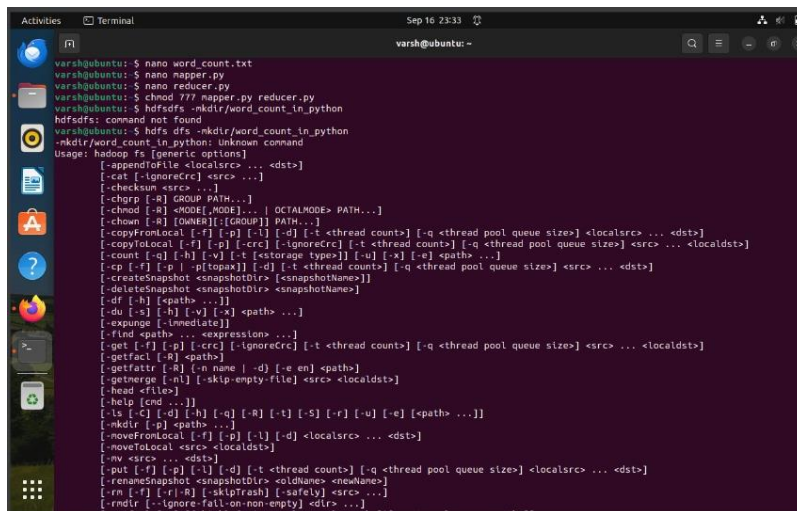
```
print( '%s\t%s' % (current_word, current_count))
```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

```
start-all.sh hdfsdfs -mkdir /word_count_in_python hdfsdfs -copyFromLocal
```

```
/path/to/word_count.txt/word_count_in_python
```



```
varsh@ubuntu:~$ nano word_count.txt
varsh@ubuntu:~$ nano mapper.py
varsh@ubuntu:~$ nano reducer.py
varsh@ubuntu:~$ chmod 777 mapper.py reducer.py
varsh@ubuntu:~$ hdfs dfs -mkdir/word_count_in_python
hdfsdfs: command not found
varsh@ubuntu:~$ hdfs dfs -mkdir/word_count_in_python
-mkdir/word_count_in_python: Unknown command
Usage: hadoop fs [generic options]
[-appendToFile <localsrc> ... <dst>]
[-cat [-ignorecrc] <src> ...]
[-checksum <src> ...]
[-chgrp [-R] <group> <path>...]
[-chmod [-R] <mode>[,<mode>]... | <OCTALMODE> <path>...]
[-chown [-R] [<owner>][:<group>] <path>...]
[-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] [-q <thread pool queue size>] <localsrc> ... <dst>]
[-copyToLocal [-f] [-p] [-crc] [-ignorecrc] [-t <thread count>] [-q <thread pool queue size>] <src> ... <localdst>]
[-count [-q] [-h] [-v] [-t <storage type>] [-u] [-x] [-e] <path> ...]
[-cp [-f] [-p] [-p[ropag]] [-d] [-t <thread count>] [-q <thread pool queue size>] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> [<snapshotName>]]
[-df [-h] [<paths> ...]]
[-du [-s] [-h] [-v] [-x] <path> ...]
[-expunge [-immediate]]
[-find <path> ... <expressions> ...]
[-get [-f] [-p] [-crc] [-ignorecrc] [-t <thread count>] [-q <thread pool queue size>] <src> ... <localdst>]
[-getfacl [-R] <path>]
[-getfatr [-R] [-n name | -d] [-e en] <path>]
[-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
[-head <file>]
[-help [cmd ...]]
[-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] [<paths> ...]]
[-mkdir [-p] <path>]
[-moveFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
[-moveToLocal <src> <localdst>]
[-mv <src> ... <dst>]
[-put [-f] [-p] [-l] [-d] [-t <thread count>] [-q <thread pool queue size>] <localsrc> ... <dst>]
[-renameSnapshot <snapshotDir> <oldName> <newName>]
[-rm [-f] [-r] [-R] [-skiptrash] [-safely] <src> ...]
[-rmr [-f] [-ignore-fail-on-non-empty] <dir> ...]
```

Step 6: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

chmod 777 mapper.py reducer.py

Step 7: Run Word Count using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

Then run the Word Count program using Hadoop Streaming.

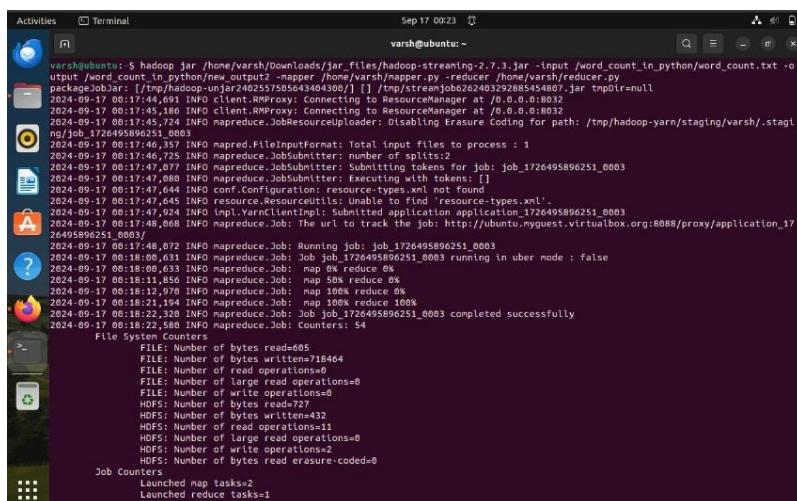
hadoop jar /path/to/hadoop-streaming-3.3.6.jar \ -input

**/word_count_in_python/word_count_data.txt **

**-output /word_count_in_python/new_output **

**-mapper /path/to/mapper.py **

-reducer /path/to/reducer.py



```
varsh@ubuntu:~$ hadoop jar /home/varsh/Downloads/jar_files/hadoop-streaming-2.7.3.jar -input /word_count_in_python/word_count.txt -o
output/word_count_in_python/new_output2 -mapper /home/varsh/mapper.py -reducer /home/varsh/reducer.py
packageJobJar: [/tmp/hadoop-unjar2402557505643404306/] [] /tmp/streamjob262403292885454807.jar tmpDir=null
2024-09-17 00:17:45,691 INFO client.RMProxy: Connecting to ResourceManager at /s.0.0.0:8032
2024-09-17 00:17:45,724 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/varsh/.stagi
ng/job_1726495896251_0003
2024-09-17 00:17:46,357 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-17 00:17:46,725 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-17 00:17:47,077 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1726495896251_0003
2024-09-17 00:17:47,088 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-17 00:17:47,644 INFO conf.Configuration: resource-types.xml not found
2024-09-17 00:17:47,645 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-17 00:17:47,924 INFO ImplVarClientImpl: Submitted application application_1726495896251_0003
2024-09-17 00:17:48,868 INFO mapreduce.Job: The url to track the job: http://ubuntu.myquest.virtualbox.org:8088/proxy/application_17
26495896251_0003/
2024-09-17 00:17:48,872 INFO mapreduce.Job: Running Job: job_1726495896251_0003
2024-09-17 00:18:00,631 INFO mapreduce.Job: Job job_1726495896251_0003 running in uber mode : false
2024-09-17 00:18:00,633 INFO mapreduce.Job: map 0% reduce 0%
2024-09-17 00:18:11,856 INFO mapreduce.Job: map 50% reduce 0%
2024-09-17 00:18:12,070 INFO mapreduce.Job: map 100% reduce 0%
2024-09-17 00:18:21,194 INFO mapreduce.Job: map 100% reduce 100%
2024-09-17 00:18:22,320 INFO mapreduce.Job: Job job_1726495896251_0003 completed successfully
2024-09-17 00:18:22,480 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=605
  FILE: Number of bytes written=718464
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=727
  HDFS: Number of bytes written=432
  HDFS: Number of read operations=11
  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=2
  Launched reduce tasks=1
Data-local map tasks=2
```

Step 8: Check Output:

Check the output of the Word Count program in the specified HDFS output directory.

hdfs dfs -cat /word_count_in_python/new_output/part-00000

```
Activities Terminal Sep 17 09:24 varsh@ubuntu: -
File Input Format Counters
  Bytes Read=569
File Output Format Counters
  Bytes Written=432
2024-09-17 00:18:22,581 INFO streaming.StreamJob: Output directory: /word_count_in_python/new_output2
varsh@ubuntu: $ hdfs dfs -cat /word_count_in_python/new_output2/part-00000
All 1
I 3
IS 1
I'd 1
Joe 1
She 1
Uncle 1
a 4
anything 1
are 1
bad 1
bare 1
being 1
bikes 1
but 1
bypass 1
can 1
catfish 1
caught 1
cover 1
do 1
either 1
for 1
gastric 1
gave 1
get 1
hands 1
help? 1
her 1
him 1
insurance 1
lt. 1
locked 1
me 1
```

Result:

Thus, the program for basic Word Count Map Reduce has been executed successfully