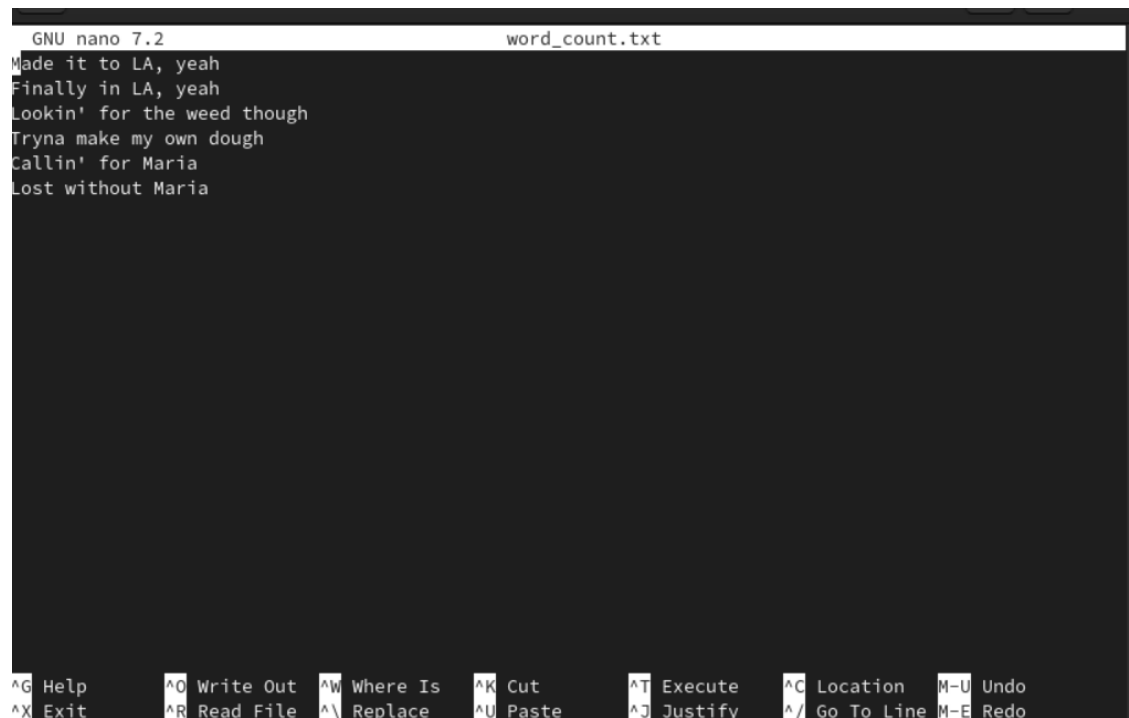


Exp No: 2**Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm****Aim:**

To Run a basic Word Count MapReduce program to understand Map Reduce Paradigm.

Procedure:**Step 1: Create Data File:**

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyze. Login with your Hadoop user.



```
GNU nano 7.2 word_count.txt
Made it to LA, yeah
Finally in LA, yeah
Lookin' for the weed though
Tryna make my own dough
Callin' for Maria
Lost without Maria

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo
```

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:
```

```
    nano word_count.txt 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
```

Step 5: Make Python Files Executable:

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

```
chmod 777 mapper.py reducer.py
```

Step 6: 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

```

```

-libjars <jar1,...>          specify a comma-separated list of jar files to be included in the classpath
-archives <archive1,...>    specify a comma-separated list of archives to be unarchived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]

osboxes@fedora:~$ hdfs dfs -ls
ls: './': No such file or directory
osboxes@fedora:~$ hdfs dfs cat /output/part-r-00000
cat: Unknown command
Did you mean -cat? This command begins with a dash.
Usage: hadoop fs [generic options]
[-appendToFile [-n] <localsrc> ... <dst>]
[-cat [-ignoreCrc] <src> ...]
[-checksum [-v] <src> ...]
[-chgrp [-R] GROUP PATH...]
[-chmod [-R] <MODE>[,<MODE>]... | OCTALMODE] PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-concat <target path> <src path> <src path> ...]
[-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] [-q <thread pool queue size>] <localsrc> ... <dst>]
[-copyToLocal [-f] [-p] [-l] [-d] [-t <thread count>] [-q <thread pool queue size>] <src> ... <localdst>]
[-count [-q] [-h] [-v] [-t <storage type>] [-u] [-x] [-e] [-s] <path> ...]
[-cp [-f] [-p] [-p[<topaz>]] [-d] [-t <thread count>] [-q <thread pool queue size>] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> [<snapshotName>]]
[-df [-h] [<path> ...]]
[-du [-s] [-h] [-v] [-x] <path> ...]
[-expunge [-immediate] [-fs <path>]]
[-find <path> ... <expression> ...]
[-get [-f] [-p] [-rc] [-ignoreCrc] [-t <thread count>] [-q <thread pool queue size>] <src> ... <localdst>]
[-getfacl [-R] <path>]
[-getfattr [-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] [<path> ...]]
[-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> ...]
[-rmdir [--ignore-fail-on-non-empty] <dir> ...]
[-setfacl [-R] [[-b|-k] [-m|-x <acl_spec>] <path>][[--set <acl_spec>] <path>]]
[-setfattr [-n name [-v value] | -x name] <path>]
[-setrep [-R] [-w] <rep> <path> ...]
[-stat [format] <path> ...]

```

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
```

```

osboxes@fedora:~$ hdfs dfs -cat /output/part-r-00000
all 1
and 3
at 2
daylight 1
drank 1
from 2
hate 2
hiding 1
i 5
it 4
love 2
of 1
oh 2
our 1
poison 1
same 3
sins 1
the 5
time 2
wine 1
you 1
osboxes@fedora:~$

```

Result:

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