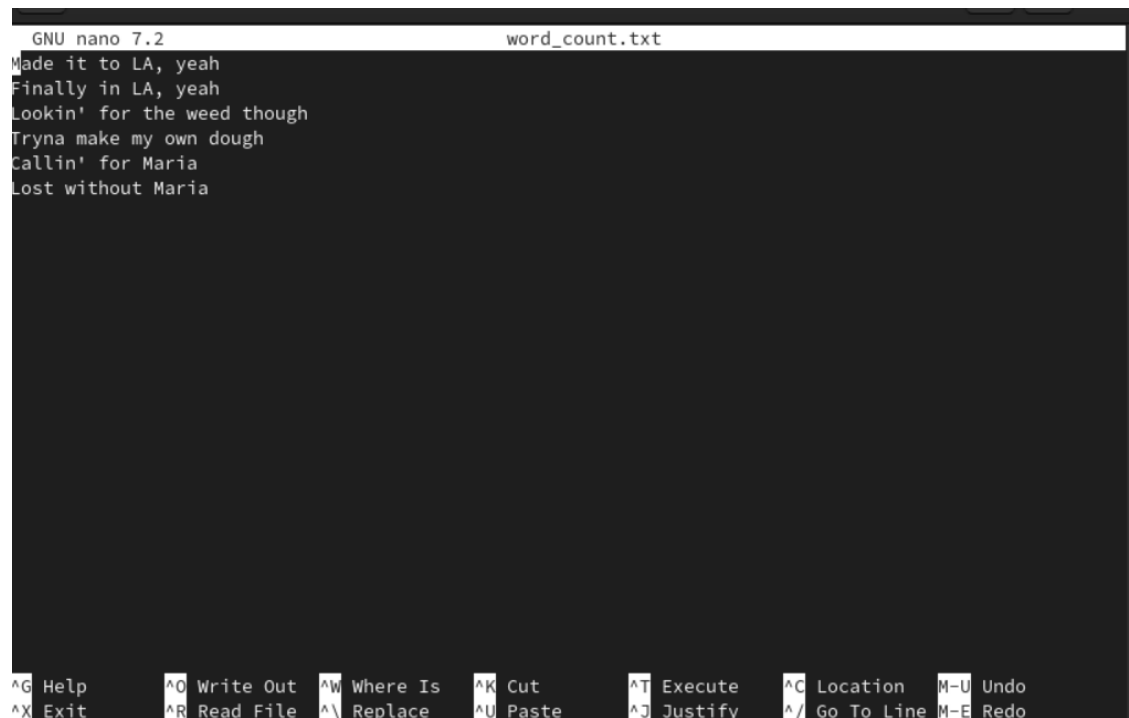


**Exp No: 2****Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm****Aim:**

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

```

```

Sep 11 2:50 AM
osboxes@fedora:~$
div_legacy      apply the offline fsimage viewer to a legacy fsimage
storagepolicies list/get/set/satisfystoragePolicy block storage policies

Client Commands:

classpath        prints the class path needed to get the hadoop jar and
                  the required libraries
dfs              run a filesystem command on the file system
envvars          display computed Hadoop environment variables
fetchdtc         fetch a delegation token from the NameNode
getconf          get config values from configuration
groups           get the groups which users belong to
lsnapsnapshot    list all snapshots for a snapshottable directory
lsSnapshottableDir list all snapshottable dirs owned by the current user
snapshotDiff     diff two snapshots of a directory or diff the current
                  directory contents with a snapshot
version          print the version

Daemon Commands:

balancer         run a cluster balancing utility
datanode         run a DFS datanode
dfsrouter        run the DFS router
dfsbalancer      Distributes data evenly among disks on a given node
https            run HTTPS server, the HDFS HTTP Gateway
journalnode      run the DFS journalnode
mover            run a utility to move block replicas across storage types
namenode         run the DFS namenode
nfs3             run an NFS version 3 gateway
portmap          run a portmap service
secondarynamenode run the DFS secondary namenode
qps             run external storagepolicysatisfier
zkfc            run the ZK Failover Controller daemon

SUBCOMMAND may print help when invoked w/o parameters or with -h.
osboxes@fedora:~$ jps
3298 ResourceManager
7738 Jps
4599 NodeManager
3848 NameNode
4030 DataNode
4271 SecondaryNameNode
osboxes@fedora:~$ hdfs dfs -ls/
ls/: Unknown command
Usage: hadoop fs [generic options]
       [-appendToFile [-n] <localsrc> ... <dst>]
       [-cat [-ignoreCrc] <src> ...]
       [-checksum [-v] <src> ...]

```

```

Sep 11 2:50 AM
osboxes@fedora:~$
-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] [-c] [-ignoreCrc] [-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] [-pTopaz]] [-q] [-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] [-c] [-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> ...]
       [-rmr [-i] [-ignore-fail-on-non-empty] <dir> ...]
       [-setfacl [-R] [-b] [-k] [-m] <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:~$
[-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> ...]
[-tail [-f] [-s <sleep interval>] <file>]
[-test [-defswrz] <path>]
[-text [-ignoreCrc] <src> ...]
[-touch [-a] [-m] [-t TIMESTAMP (yyyyMMdd:HHmmss) ] [-c] <path> ...]
[-touchz <path> ...]
[-truncate [-w] <length> <path> ...]
[-usage [cmd ...]]

Generic options supported are:
-conf <configuration file>      specify an application configuration file
-D <property=value>             define a value for a given property
-fs <file:///hdfs://namenode:port> specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.
-jt <local|resourceManager:port> specify a ResourceManager
-files <file1,...>              specify a comma-separated list of files to be copied to the map reduce cluster
-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 -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.