

PRACTICAL: 1

Aim: Study of various HDFS Commands:

1] ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

Some examples of the short options available:

-a, --all

do not ignore entries starting with.

-A, --almost-all

do not list implied. and ...

--author

with -l, print the author of each file

-b, --escape

print C-style escapes for nongraphic characters

EXAMPLE

ls command without option:

```
hp@hp-VirtualBox:~$ ls
```

```
Demo Documents examples.desktop Music Public user Desktop Downloads m1 Pictures  
Templates Videos
```

ls command with option:

```
hp@hp-VirtualBox:~$ ls -a
```

```
. Desktop m1 .sudo_as_admin_successful
.. Documents .mozilla Templates
.bash_history Downloads Music user
.bash_logout examples.desktop .oracle_jre_usage Videos

.bashrc .gconf Pictures .Xauthority
.cache .gnupg .profile .xsession-errors
.config .ICEauthority Public .xsession-errors.old
Demo .local .ssh
```

2] cd – change directory

SYNOPSIS

```
cd directory_name
```

DESCRIPTION

Changes the directory to the directory specified with the command.

EXAMPLE

```
hp@hp-VirtualBox:~$ cd Demo/
```

```
hp@hp-VirtualBox:~/Demo$
```

3] mv - move (rename) files

SYNOPSIS

```
mv [OPTION]... [-T] SOURCE DEST
```

```
mv [OPTION]... SOURCE... DIRECTORY
```

`mv [OPTION]... -t DIRECTORY SOURCE...`

DESCRIPTION

Rename **SOURCE** to **DEST**, or move **SOURCE(s)** to **DIRECTORY**. Mandatory arguments to long options are mandatory for short options too. `--backup[=CONTROL]`

make a backup of each existing destination file `-b` like `--backup` but does not accept an argument
`-f, --forcedo` not prompt before overwriting

`-i, --interactive`

prompt before overwrite

`-n, --no-clobber`

do not overwrite an existing file

EXAMPLE

`hp@hp-VirtualBox:~$ mv Demo1/Demo1.txt Demo` `hp@hp-VirtualBox:~$ ls Demo1 Demo1.txt`

4] man - an interface to the on-line reference manuals

SYNOPSIS

`man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension]`

`[-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation]`

`[--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page ...] ...`

`man -k [apropos options] regexp ...`

`man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...`

`man -f [whatis options] page ...`

`man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string]`

`[-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...`

`man -w|-W [-C file] [-d] [-D] page ...`

`man -c [-C file] [-d] [-D] page ...`

`man [-?V]`

DESCRIPTION

`man` is the system's manual pager. Each page argument given to `man` is normally the name of a program, utility or function. The

manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct `man` to look

only in that section of the manual.

The table below shows the section numbers of the manual followed by the types of pages they contain.

Executable programs or shell commands

System calls (functions provided by the kernel)

Library calls (functions within program libraries)

Special files (usually found in `/dev`)

File formats and conventions eg `/etc/passwd`

Games

Miscellaneous (including macro packages and conventions), e.g. `man(7)`, `groff(7)`

System administration commands (usually only for root)

Kernel routines [Non standard]

EXAMPLE

```
man ls
```

Display the manual page for the item (program) ls.

```
man -a intro
```

Display, in succession, all of the available intro manual pages contained within the manual. It is possible to quit between successive displays or skip any of them.

5] mkdir - make directories

SYNOPSIS

```
mkdir [OPTION]... DIRECTORY...
```

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

-m, --mode=MODE

set file mode (as in chmod), not a=rwx - umask -p, --parents

no error if existing, make parent directories as needed -v, --verbose

print a message for each created directory

-Z set SELinux security context of each created directory to the default type

EXAMPLE

```
hp@hp-VirtualBox:~$ mkdir m1
```

```
hp@hp-VirtualBox:~$ ls
```

Demo Documents examples.desktop Music Public user Desktop Downloads m1 Pictures
Templates Videos

6] `rmdir` - remove empty directories

SYNOPSIS

`rmdir` [OPTION]... DIRECTORY...

DESCRIPTION

Remove the DIRECTORY(ies), if they are empty.

`--ignore-fail-on-non-empty`

ignore each failure that is solely because a directory is non-empty

`-p, --parents`

remove DIRECTORY and its ancestors; e.g., '`rmdir -p a/b/c`' is similar to '`rmdir a/b/c a/b
a`'

`-v, --verbose`

output a diagnostic for every directory processed

EXAMPLE

```
hp@hp-VirtualBox:~$ ls
```

```
Demo Documents examples.desktop Music Public user Desktop Downloads m1  
Pictures Templates Videos
```

```
hp@hp-VirtualBox:~$ rmdir a1
```

```
hp@hp-VirtualBox:~$ ls
```

```
Demo Documents examples.desktop Pictures Templates Videos  
Desktop Downloads Music Public user
```

7] touch - change file timestamps

SYNOPSIS

touch [OPTION]... FILE...

DESCRIPTION

Update the access and modification times of each FILE to the current time.

A FILE argument that does not exist is created empty, unless -c or -h is supplied.

A FILE argument string of - is handled specially and causes touch to change the times of the file associated with standard output.

Mandatory arguments to long options are mandatory for short options too.

-a change only the access time

-c, --no-create

do not create any files

-d, --date=STRING

parse STRING and use it instead of current time

-f (ignored)

-h, --no-dereference

affect each symbolic link instead of any referenced file (useful only on systems that can change the timestamps of a sym-link)

-m change only the modification time

EXAMPLE

```
hp@hp-VirtualBox:~/Demo$ touch Demo.doc
```

```
hp@hp-VirtualBox:~$ ls Demo/
```

```
Demo1.txt Demo.doc Demo.txt
```

8] rm - remove files or directories

SYNOPSIS

rm [OPTION]... [FILE]...

DESCRIPTION

This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories.

If the `-I` or `--interactive=once` option is given, and there are more than three files or the `-r`, `-R`, or `--recursive` are given, then rm prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted

Otherwise, if a file is unwritable, standard input is a terminal, and the `-f` or `--force` option is not given, or the `-i` or `--inter-active=always` option is given, rm prompts the user for whether to remove the file. If the response is not affirmative, the file is skipped.

OPTIONS

Remove (unlink) the FILE(s).

`-f`, `--force`

ignore nonexistent files and arguments, never prompt `-i` prompt before every removal

`-I` prompt once before removing more than three files, or when removing recursively; less intrusive than `-i`, while still giving protection against most mistakes

`--interactive[=WHEN]`

prompt according to WHEN: never, once (`-I`), or always (`-i`); without WHEN, prompt

always

EXAMPLE

```
hp@hp-VirtualBox:~/Demo$ rm Demo.doc
```

```
hp@hp-VirtualBox:~/Demo$ ls
```

```
Demo1.txt Demo.txt
```


9] du - estimate file space usage

SYNOPSIS

du [OPTION]... [FILE]...

du [OPTION]... --files0-from=F

DESCRIPTION

Summarize disk usage of the set of FILES, recursively for directories.

Mandatory arguments to long options are mandatory for short options too.

-0, --null

end each output line with NUL, not newline

-a, --all write counts for all files, not just directories

--apparent-size

print apparent sizes, rather than disk usage; although the apparent size is usually smaller, it may be larger due to holes in ('sparse') files, internal fragmentation, indirect blocks, and the like

-B, --block-size=SIZE

scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

-b, --bytes

equivalent to '--apparent-size --block-size=1'

EXAMPLE

```
hp@hp-VirtualBox:~/Demo$ du Demo.txt
```

```
4      Demo.txt
```

10] clear - clear the terminal screen

SYNOPSIS

clear

DESCRIPTION

clear clears your screen if this is possible, including its scrollback buffer (if the extended "E3" capability is defined). clear

looks in the environment for the terminal type and then in the terminfo database to determine how to clear the screen.

clear ignores any command-line parameters that may be present.

EXAMPLE

```
hp@hp-VirtualBox:~$ clear
```

11] cat - concatenate files and print on the standard output

SYNOPSIS

```
cat [OPTION]... [FILE]...
```

DESCRIPTION

Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

-A, --show-all

equivalent to -vET

-b, --number-nonblank

number nonempty output lines, overrides -n

-e equivalent to -vE

-E, --show-ends

display \$ at end of each line

-n, --number

number all output lines

EXAMPLE

```
hp@hp-VirtualBox:~/Demo$ cat >Demo.txt
```

```
hello svit...
```

```
this is practical of cmd code with hadoop.^X^Z
```

```
[1]+ Stopped cat > Demo.txt
```

```
hp@hp-VirtualBox:~/Demo$ cat Demo.txt
```

PRACTICAL: 2

AIM: Study of Hadoop architecture and Installation on Hadoop single/multimode cluster.

Hadoop Architecture:

Apache Hadoop offers a scalable, flexible and reliable distributed computing big data framework for a cluster of systems with storage capacity and local computing power by leveraging commodity hardware. Hadoop follows a Master Slave architecture for the transformation and analysis of large datasets using Hadoop MapReduce paradigm. The 3 important hadoop components that play a vital role in the Hadoop architecture are -

Hadoop Distributed File System (HDFS) – Patterned after the UNIX file system

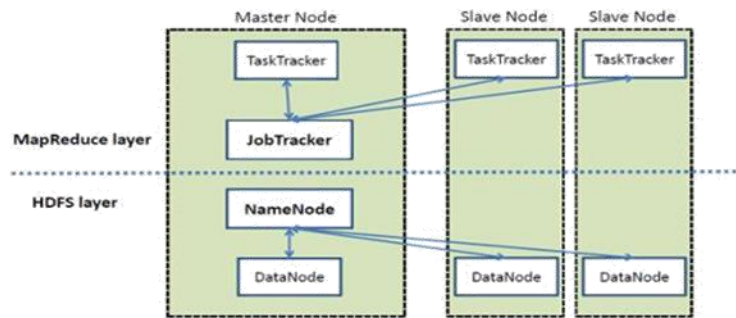
Hadoop MapReduce

Yet Another Resource Negotiator (YARN)

HadoopSkillset requires thoughtful knowledge of every layer in the hadoop stack right from understanding about the various components in the hadoop architecture, designing a hadoop cluster, performance tuning it and setting up the top chain responsible for data processing.

Hadoop follows a master slave architecture design for data storage and distributed data processing using HDFS and MapReduce respectively. The master node for data storage is hadoop HDFS is the NameNode and the master node for parallel processing of data using Hadoop MapReduce is the Job Tracker. The slave nodes in the hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a DataNode that synchronizes the processes with the Job Tracker and NameNode respectively. In Hadoop architectural implementation the master or slave systems can be setup in the cloud or on-premise.

High Level Architecture of Hadoop



Installation Steps (In Ubuntu):

Install Java and add the repository.

Command: `sudo add-apt-repository ppa:webupd8team/java`

It will ask for password, enter password.

Now update the System.

Command: `sudo apt-get update`

Now invoke the JAVA-8-installer

Command: `sudo apt-get install oracle-java8-installer` It will ask for accepting the license. Hit OK.

Confirm Java Version **Command:** `java -version`

Now install openssh server

Command: `sudo apt-get install openssh-server`

Generate ssh keys

Command: `ssh-keygen -t rsa -P ""`

Just press enter for password , don't enter any password by yourself.

Just enable ssh access

Command: `cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys`

Test ssh access

Command: `ssh localhost`

Now disable the IPv6

Open the sysctl.conf file in vi editor by following command:

Command: `sudo vi /etc/sysctl.conf`

Add the lines written below: `#disable ipv6 net.ipv6.conf.all.disable_ipv6 = 1`
`net.ipv6.conf.default.disable_ipv6 = 1 net.ipv6.conf.lo.disable_ipv6 = 1`

Reboot your System.

Once you restart the machine test if IPv6 is disabled, your output should be as below **Command:**
`cat /proc/sys/net/ipv6/conf/all/disable_ipv6`

Output: 1

Now we are ready for Hadoop Installation Download hadoop-2.7.3

Now extract the tar file

Command: `tar -xzvf hadoop-2.7.3.tar.gz`

You will see Hadoop folder

Open your .bashrc file as shown below:

Command: /hadoop-2.7.3/etc/hadoop\$ nano ~/.bashrc

Add following Lines:

GNU nano 2.2.6 File: /home/user/.bashrc

HADOOP VARIABLES START

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle export
HADOOP_INSTALL=/home/user/hadoop-2.7.3 export
PATH=$PATH:$HADOOP_INSTALL/bin export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL export
HADOOP_HDFS_HOME=$HADOOP_INSTALL export
YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native export
HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

You should now close the terminal and open a new terminal to check Hadoop and Java homes as shown below:

Command: echo \$JAVA_HOMEecho \$HADOOP_INSTALL

Now we will configure Hadoop.Go to hadoop-2.7.3/etc/hadoop in Hadoop folder,you should be able to see all the config files

Command: /hadoop-2.7.3/etc/hadoop\$ gedit hadoop-env.sh

Add the lines :

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
```

Open edit core-site.xml and add the following lines: **Command:** /hadoop-2.7.3/etc/hadoop\$ nano core-site.xml

Edit core-site.xml as followed and add the following lines:

GNU nano 2.2.6 File: core-site.xml

```
<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>
```

Open yarn-site.xml

Command: /hadoop-2.7.3/etc/hadoop\$ nano yarn-site.xml

Edit yarn-site.xml as follows and add the lines as shown <configuration>

```
<!--Site specific YARN configuration properties -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>
```



```
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
```

```
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
```

```
</property>
```

```
</configuration>
```

Open mapred-site.xml

Command: /hadoop-2.7.3/etc/hadoop\$ nano mapred-site.xml

Add the lines in mapred-site.xml file: <configuration>

```
<property>
```

```
<name>mapreduce.framework.name</name>
```

```
<value>yarn</value>
```

```
</property>
```

```
</configuration>
```

For hdfs – Create two directories for Hadoop storage **Command:** mkdir -p
hadoop_store/hdfs/namenode

```
mkdir -p hadoop_store/hdfs/datanode
```

Open hdfs-site.xml

Command: /hadoop-2.7.3/etc/hadoop\$ nano hdfs-site.xml

Add the following lines: <configuration> <property> <name>fs.replication</name>

```
<value>1</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.namenode.name.dir</name>
```

```
<value>file:/home/user/hadoop_store/hdfs/namenode</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.datanode.name.dir</name>
```

```
<value>file:/home/user/hadoop_store/hdfs/datanode</value>
```

```
</property>
```

```
</configuration>
```

Change folder permission

Command: sudo chown -R user:user /home/user/hadoop-2.7.3/etc/hadoop

sudo chown -R user:user /home/user/hadoop_store

Format namenode

Command: hdfs namenode -format

Start all the services of Hadoop as shown below: **Command:** start-all.sh

You can confirm the services by running jps command as shown below: **Command:** jps

3268 Namenode

3389 Datanod

3588 SecondaryNamenode

3800 ResourceManager

4247 Jps

3924 NodeManager

Type command to check the directory of hdfs

Command: hadoop fs -ls /

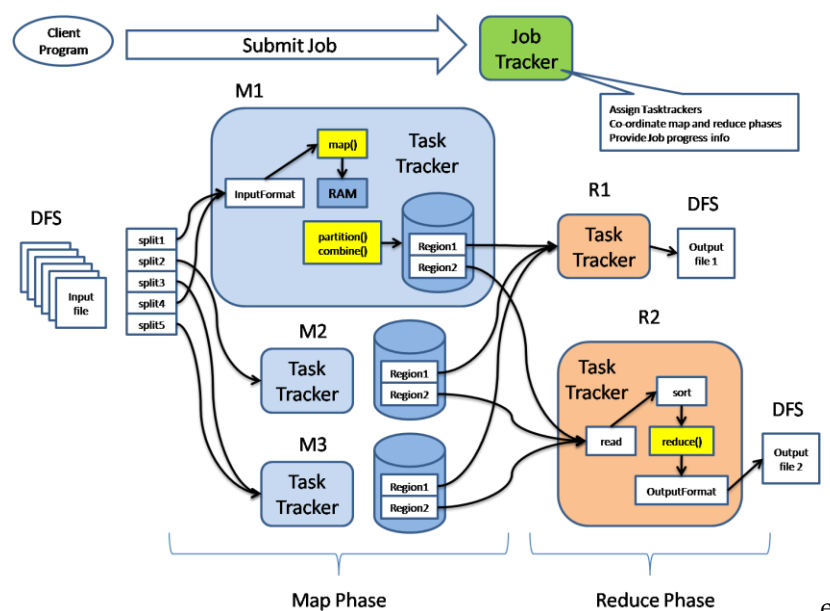
PRACTICAL: 3

Aim: To understand the overall programming architecture using Map Reduce API

MapReduce is mainly used for parallel processing of large sets of data stored in Hadoop cluster. Initially, it is a hypothesis specially designed by Google to provide parallelism, data distribution and fault-tolerance. MR processes data in the form of key-value pairs. A key-value (KV) pair is a mapping element between two linked data items - key and its value. The key (K) acts as an identifier to the value. An example of a key-value (KV) pair is a pair where the key is the node Id and the value is its properties including neighbor nodes, predecessor node, etc. MR API provides the following features like batch processing, parallel processing of huge amounts of data and high availability. For processing large sets of data MR comes into the picture. The programmers will write MR applications that could be suitable for their business scenarios. Programmers have to understand the MR working flow and according to the flow, applications will be developed and deployed across Hadoop clusters. Hadoop built on Java APIs and it provides some MR APIs that is going to deal with parallel computing across nodes.

The MR work flow undergoes different phases and the end result will be stored in hdfs with replications. Job tracker is going to take care of all MR jobs that are running on various nodes present in the Hadoop cluster. Job tracker plays vital role in scheduling jobs and it will keep track of the entire map and reduce jobs. Actual map and reduce tasks are performed by Task tracker.

Hadoop Map Reduce architecture



Map reduce architecture consists of mainly two processing stages. First one is the map stage and the second one is reduce stage. The actual MR process happens in task tracker. In between map and reduce stages, Intermediate process will take place. Intermediate process will do operations like shuffle and sorting of the mapper output data. The Intermediate data is going to get stored in local file system.

Mapper Phase

In Mapper Phase the input data is going to split into 2 components, Key and Value. The key is writable and comparable in the processing stage. Value is writable only during the processing stage. Suppose, client submits input data to Hadoop system, the Job tracker assigns tasks to task tracker. The input data that is going to get split into several input splits.

Input splits are the logical splits in nature. Record reader converts these input splits in Key-Value (KV) pair. This is the actual input data format for the mapped input for further processing of data inside Task tracker. The input format type varies from one type of application to another. So the programmer has to observe input data and to code according.

Suppose we take Text input format, the key is going to be byte offset and value will be the entire line. Partition and combiner logics come in to map coding logic only to perform special data operations. Data localization occurs only in mapper nodes.

Combiner is also called as mini reducer. The reducer code is placed in the mapper as a combiner. When mapper output is a huge amount of data, it will require high network bandwidth. To solve this bandwidth issue, we will place the reduced code in mapper as combiner for better performance. Default partition used in this process is Hash partition.

A partition module in Hadoop plays a very important role to partition the data received from either different mappers or combiners. Partitioner reduces the pressure that builds on reducer and gives more performance. There is a customized partition which can be performed on any relevant data on different basis or conditions.

Also, it has static and dynamic partitions which play a very important role in hadoop as well as hive. The partitioner would split the data into numbers of folders using reducers at the end of map reduce phase. According to the business requirement developer will design this partition code. This partitioner runs in between Mapper and Reducer. It is very efficient for query purpose.

Intermediate Process

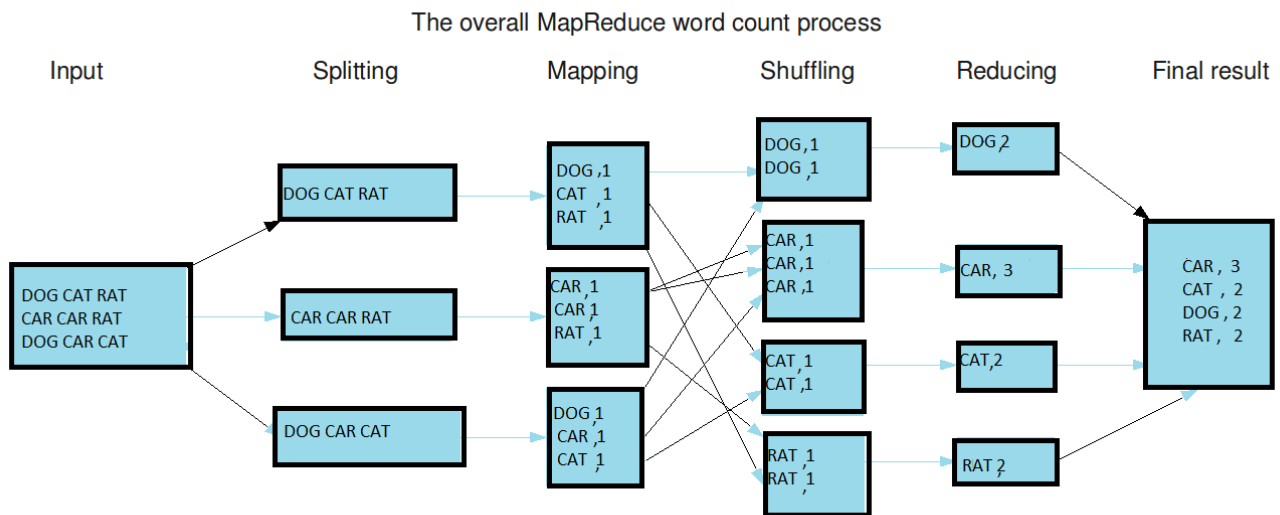
The mapper output data undergoes shuffle and sorting in intermediate process. The intermediate data is going to get stored in local file system without having replications in Hadoop nodes. This intermediate data is the data that is generated after some computations based on certain logics. Hadoop uses a Round-Robin algorithm to write the intermediate data to local disk. There are many other sorting factors to reach the conditions to write the data to local disks.

Reducer Phase

Shuffled and sorted data is going to pass as input to the reducer. In this phase, all incoming data is going to combine and same actual key value pairs is going to write into hdfs system. Record writer writes data from reducer to hdfs. The reducer is not so mandatory for searching and mapping purpose.

Reducer logic is mainly used to start the operations on mapper data which is sorted and finally it gives the reducer outputs like part-r-0001etc,. Options are provided to set the number of reducers for each job that the user wanted to run. In the configuration file mapred-site.xml, we have to set some properties which will enable to set the number of reducers for the particular task.

MapReduce word count Example



PRACTICAL: 4

Aim: Implement word count using map-reduce.

Step-1: Start Hadoop from it's directory

```
$ hadoop namenode-format
```

```
$ start-all.sh
```

Step-2: Write Mapper,Reducer and Controller programs and prepare.jar import
org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable;

```
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.mapred.*;
```

```
import org.apache.hadoop.util.*;
```

```
import org.apache.hadoop.conf.*;
```

```
import java.util.*;
```

```
import java.io.IOException;
```

```
public class WordCount{
```

```
//Mapper Class
```

```
public static class WordMapper extends MapReduceBase implements  
Mapper<LongWritable,Text,Text,IntWritable>{
```

```
//Mapper Function
```

```
@Override
```

```
public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,  
Reporter reporter) throws IOException {
```

```
String data = value.toString();
```

```
data = data.trim();
```

```
data = data.replaceAll(".",",")

String[] words = data.split("\\W+");

for(String word : words){

output.collect(new Text(word),new IntWritable(1));

}

//Reducer Class

public static class WordReducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable>{

//Reducer Function

@Override

public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable>
output, Reporter reporter) throws IOException{

int sum = 0;

while(values.hasNext()){

sum += (values.next()).get();

}

output.collect(key, new IntWritable(sum));

}

//Main Function

public static void main(String[] args)throws Exception{ JobConf wordJob = new
JobConf(Practical4.class); wordJob.setJobName("&quot;Word Count&quot;");
FileInputFormat.setInputPaths(wordJob, new Path(args[0]));
FileOutputFormat.setOutputPath(wordJob, new Path(args[1]));
wordJob.setMapperClass(WordMapper.class); wordJob.setCombinerClass(WordReducer.class);
wordJob.setReducerClass(WordReducer.class); wordJob.setOutputKeyClass(Text.class);
wordJob.setOutputValueClass(IntWritable.class);

JobClient.runJob(wordJob);

}

create local directory 'word'

$ mkdir word
```


compile java classes using hadoop core jar

```
$ javac -cp $classpath -d wordcounts WordCount.java
```

 compress the classes to jar

```
$ jar -jvf word.jar -C wordcounts/ .
```

Step-3: Create input directory in hdfs and store input data

```
$ hdfs dfs -mkdir /meghesh
```

```
$ hdfs dfs -put /home/svit/megesh.txt /meghesh/
```


Step-4: Start MapReduce process by deploying .jar

```
$ hadoop jar word.jar word meghesh output
```

Step-5: Display output

```
$ hdfs dfs -cat /output/part-00000
```

Input:



```
meghesh.txt (~/) - gedit
hello
performing practical 3
we are using hadoop
we are in svit
svit is locating in vasad
```

Output:



```
svit@svit-Vostro-3902: ~/wordcounts
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=137
File Output Format Counters
  Bytes Written=100
svit@svit-Vostro-3902:~/wordcounts$ hdfs dfs -cat /output/part-00000
3
are 2
hadoop 1
hello 1
in 2
is 1
locating 1
performing 1
practical 1
svit 2
using 1
vasad 1
we 2
svit@svit-Vostro-3902:~/wordcounts$
```

PRACTICAL: 5

Aim: Implementing Matrix Multiplication using Map Reduce.

Step-1: Start Hadoop from it's directory

```
$ hadoop namenode-format
```

```
$ start-all.sh
```

Step-2: Write Mapper,Reducer and Controller programs and prepare.jar

```
import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.mapred.*;
```

```
import org.apache.hadoop.util.*;
```

```
import org.apache.hadoop.conf.*;
```

```
import java.util.*;
```

```
import java.io.IOException;
```

```
public class Matrix {
```

```
    public static class Map extends Mapper<LongWritable, Text, Text, Text> {
```

```
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException{
```

```
            String line = value.toString();
```

```
            String[] indicesAndValue = line.split(",");
```

```
            Text outputKey = new Text();
```

```
            Text outputValue = new Text();
```

```
            if (indicesAndValue[0].equals("A")){
```

```
outputKey.set(indicesAndValue[2]);

outputValue.set("A," + indicesAndValue[1] + "," + indicesAndValue[3]);
context.write(outputKey, outputValue);

else {

outputKey.set(indicesAndValue[1]);

outputValue.set("B," + indicesAndValue[2] + "," + indicesAndValue[3]);
context.write(outputKey, outputValue);

}}}

public static class Reduce extends Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {

String[] value;

ArrayList<Entry<Integer, Float>> listA = new ArrayList<Entry<Integer, Float>>();
ArrayList<Entry<Integer, Float>> listB = new ArrayList<Entry<Integer, Float>>(); for (Text val
: values) {

value = val.toString().split(",");

if (value[0].equals("A")) {

    listA.add(new SimpleEntry<Integer, Float>(Integer.parseInt(value[1]),
Float.parseFloat(value[2])));

}

else

{

    listB.add(new SimpleEntry<Integer, Float>(Integer.parseInt(value[1]),
Float.parseFloat(value[2])));}

}

String i;

float a_ij;

String k;
```

```
float b_jk;

Text outputValue = new Text();

for (Entry<Integer, Float> a : listA) {

i = Integer.toString(a.getKey());

a_ij = a.getValue();

for (Entry<Integer, Float> b : listB) {

k = Integer.toString(b.getKey());

b_jk = b.getValue();

outputValue.set(i + "," + k + "," + Float.toString(a_ij*b_jk)); context.write(null, outputValue);
}}}}

public static void main(String[] args) throws Exception { Configuration conf = new
Configuration();

Job job = new Job(conf, "Matrix"); job.setJarByClass(Matrix.class);
job.setOutputKeyClass(Text.class); job.setOutputValueClass(Text.class);
job.setMapperClass(Map.class); job.setReducerClass(Reduce.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class); FileInputFormat.addInputPath(job, new
Path(args[0])); FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.waitForCompletion(true); }

create local directory 'matrix'

$ mkdir matrix

compile java classes using hadoop core jar

$ javac -cp $classpath -d matrix Matrix.java

compress the classes to jar

$ jar -jvf martix.jar -C matrix/ .

Step-3: Create input directory in hdfs and store input data $ hdfs dfs -mkdir /MatrixMultiply

$ hdfs dfs -put /home/hp/M1.txt /MatrixMultiply/ $ hdfs dfs -mkdir MatrixMultiply

$ hdfs dfs -put /home/hp/N1.txt /MatrixMultiply

Step-4: Start MapReduce process by deploying .jar $ hadoop jar matrix.jar matrix
MatrixMultiply output
```

Step-5: Display output

```
$ hdfs dfs -cat /output/part-00000
```

Input:

M1.txt	N1.txt
M, 0, 0, 63	N, 0, 0, 63
M, 0, 1, 45	N, 0, 1, 18
M, 0, 2, 93	N, 0, 2, 89
M, 0, 3, 32	N, 0, 3, 28
M, 0, 4, 49	N, 0, 4, 39
M, 1, 0, 33	N, 1, 0, 59
M, 1, 3, 26	N, 1, 1, 76
M, 1, 4, 95	N, 1, 2, 34
M, 1, 0, 25	N, 1, 3, 12
M, 1, 1, 11	N, 1, 4, 6

Output:

```
0,0 11878
0,1 14044
0,2 16031
0,3 5964
0,4 15874
1,0 4081
1,1 6914
1,2 8282 , 1,3 7479, 1,4 9647;
```

PRACTICAL: 6

Aim: Creating HDFS tables and loading them in Hive and learn joining of tables in Hive.

Starting hadoop from its local directory :

```
$ cd /usr/local/hadoop
```

```
$ /bin/hadoop namenode -format
```

```
$/bin/start-dfs.sh
```

Creating first table in Hive:

```
hive> create table if not exists customers (id int, name String, age int, address String, salary String) comment 'customer details' row format delimited fields terminated by ' ' lines terminated by '\n' stored as textfile;
OK
Time taken: 1.673 seconds
```

Creating second table in Hive:

```
hive> create table if not exists orders (oid int, odate String, id int, amount string) comment 'order details' row format delimited fields terminated by ' ' lines terminated by '\n' stored as textfile;
OK
Time taken: 2.278 seconds
```

Creating data text file in local directory:

customers.txt

1 Ramesh 32 Ahemdabad 2000.0

2 Jayesh 25 Delhi 1500.0

3 Jignesh 23 Kota 2000.0

4 Dharmesh 25 Mumbai 6500.0

5 Hardik 27 Bhopal 8500.

orders.txt

102 2009-10-08-00:00:00 3 3000

100 2009-10-08-00:00:00 3 1500

101 2009-11-20-00:00:00 2 1560

103 2008-05-20-00:00:00 4 2060

Loading data from local text files in Hive tables**Customers table:**

```
hive> load data local inpath '/home/training/Documents/customers.txt' overwrite
into table customers;
Copying data from file:/home/training/Documents/customers.txt
Copying file: file:/home/training/Documents/customers.txt
Loading data to table default.customers
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted /user/hive/warehouse/customers
OK
Time taken: 0.457 seconds
hive> select * from customers;
OK
1      Ramesh  32      Ahemdabad      2000.0
2      Jayesh  25      Delhi      1500.0
3      Jignesh  23      Kota      2000.0
4      Dharmesh      25      Mumbai      6500.0
5      Hardik  27      Bhopal      8500.0
Time taken: 0.229 seconds
```

Orders table:

```
hive> load data local inpath '/home/training/Documents/orders.txt' overwrite int
o table orders;
Copying data from file:/home/training/Documents/orders.txt
Copying file: file:/home/training/Documents/orders.txt
Loading data to table default.orders
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted /user/hive/warehouse/orders
OK
Time taken: 0.22 seconds
hive> select * from orders;
OK
102    2009-10-08-00:00:00      3      3000
100    2009-10-08-00:00:00      3      1500
101    2009-11-20-00:00:00      2      1560
103    2008-05-20-00:00:00      4      2060
Time taken: 0.118 seconds
```

Performing Inner Join :

```
hive> select c.id, c.name, c.age, o.amount from customers c join orders o on (c.
id = o.id);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201710080837_0001, Tracking URL = http://0.0.0.0:50030/jobdet
ails.jsp?jobid=job_201710080837_0001
Kill Command = /usr/lib/hadoop/bin/hadoop job -Dmapred.job.tracker=0.0.0.0:8021
-kill job_201710080837_0001
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2017-10-08 09:43:50,688 Stage-1 map = 0%, reduce = 0%
2017-10-08 09:43:54,775 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.13 se
c
2017-10-08 09:43:55,790 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.13 se
c
2017-10-08 09:43:56,811 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.87
sec
2017-10-08 09:43:57,825 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.87
sec
MapReduce Total cumulative CPU time: 1 seconds 870 msec
Ended Job = job_201710080837_0001
MapReduce Jobs Launched:
Job 0: Map: 2 Reduce: 1 Cumulative CPU: 1.87 sec HDFS Read: 0 HDFS Write: 0
SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 870 msec
OK
2      Jayesh      25      1560
3      Jignesh     23      3000
3      Jignesh     23      1500
4      Dharmesh    25      2060
Time taken: 11.297 seconds
```

Performing Right Outer Join:

```
SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 900 msec
OK
1      Ramesh      NULL      NULL
2      Jayesh      1560      2009-11-20-00:00:00
3      Jignesh     3000      2009-10-08-00:00:00
3      Jignesh     1500      2009-10-08-00:00:00
4      Dharmesh    2060      2008-05-20-00:00:00
5      Hardik      NULL      NULL
Time taken: 10.524 seconds
```


Performing Left Outer Join:

```
hive> select c.id, c.name, o.amount, o.odate from customers c left outer join orders o on (c.id = o.id);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201710080837_0002, Tracking URL = http://0.0.0.0:50030/jobdetails.jsp?jobid=job_201710080837_0002
Kill Command = /usr/lib/hadoop/bin/hadoop job -Dmapred.job.tracker=0.0.0.0:8021 -kill job_201710080837_0002
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2017-10-08 09:48:14,317 Stage-1 map = 0%, reduce = 0%
2017-10-08 09:48:18,335 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.1 sec
2017-10-08 09:48:19,343 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.1 sec
2017-10-08 09:48:20,355 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.84 sec
2017-10-08 09:48:21,365 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.84 sec
2017-10-08 09:48:22,374 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.84 sec
MapReduce Total cumulative CPU time: 1 seconds 840 msec
Ended Job = job_201710080837_0002
MapReduce Jobs Launched:
Job 0: Map: 2 Reduce: 1 Cumulative CPU: 1.84 sec HDFS Read: 0 HDFS Write: 0 SUCCESS
```

```
OK
1      Ramesh  32      Ahmedabad      2000.0
2      Jayesh  25      Delhi      1500.0
3      Jignesh 23      Kota      2000.0
4      Dharmesh 25      Mumbai  6500.0
5      Hardik  27      Bhopal   8500.0
Time taken: 10.62 seconds
```

Performing Full Outer Join:

```
hive> select c.id, c.name, o.amount, o.odate from customers c full outer join or
ders o on (c.id = o.id);
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201710080837_0004, Tracking URL = http://0.0.0.0:50030/jobdet
ails.jsp?jobid=job_201710080837_0004
Kill Command = /usr/lib/hadoop/bin/hadoop job -Dmapred.job.tracker=0.0.0.0:8021
-kill job_201710080837_0004
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2017-10-08 09:57:55,186 Stage-1 map = 0%, reduce = 0%
2017-10-08 09:57:59,203 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.14 se
c
2017-10-08 09:58:00,212 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.14 se
c
2017-10-08 09:58:01,222 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.9 s
ec
2017-10-08 09:58:02,227 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.9 s
ec
2017-10-08 09:58:03,234 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.9 s
ec
MapReduce Total cumulative CPU time: 1 seconds 900 msec
Ended Job = job_201710080837_0004
MapReduce Jobs Launched:
Job 0: Map: 2 Reduce: 1 Cumulative CPU: 1.9 sec HDFS Read: 0 HDFS Write: 0
SUCCESS
```

PRACTICAL: 7

Aim: Creating the HDFS tables and loading them in pig and learn joining of tables in Pig.

Start hadoop hdfs:

```
$cd <hadoop-directoy>/sbin
```

```
$/start-all.sh
```

Move text files to hdfs directory:

```
$hadoop fs -mkdir user1
```

```
$hadoop fs -mkdir user1/pig
```

```
$hadoop fs -put customers.txt user1/pig
```

```
$hadoop fs -put orders.txt user1/pi
```

Start pig latin in mapreduce mode:

```
$cd piglatin/bin
```

```
$pig -x mapreduce
```

Create customer and order tables:

```
grunt> customers = LOAD 'hdfs://localhost:8888/user1/pig/customers.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, address:chararray, salary:int);
```

```
> dump customers;
```

```
grunt> orders = LOAD 'hdfs://localhost:8888/user1/pig/orders.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int, amount:int);
```

```
(1,Ramesh,32,Ahmedabad,2000)
(2,Jayesh,25,Delhi,1500)
(3,Jignesh,23,Kota,2000)
(4,Dharmesh,25,Mumbai,6500)
(5,Hardik,27,Bhopal,8500)
(6,Komal,22,MP,4500)
(7,Muffy,24,Indore,10000)
```

dump orders;

```
(102,2009-10-08 00:00:00,3,3000)
(100,2009-10-08 00:00:00,3,1500)
(101,2009-11-20 00:00:00,2,1560)
(103,2008-05-20 00:00:00,4,2060)
```

Inner join

```
grunt> customer_orders = join customers by id, orders by customer_id;
grunt> dump customer_orders;
```

```
(2,Jayesh,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,Jignesh,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,Jignesh,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Dharmesh,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
```

Left outer join:

```
grunt> right_outer = join customers by id right outer, orders by customer_id;
grunt> dump right_outer;
```

```
grunt> full_outer = join customers by id full outer, orders by customer_id;
grunt> dump full_outer;
```

```
(2,Jayesh,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,Jignesh,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,Jignesh,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Dharmesh,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
```

Full outer join:

```
grunt> full_outer = join customers by id full outer, orders by customer_id;  
grunt> dump full_outer;
```

```
(1,Ramesh,32,Ahmedabad,2000,,,,)  
(2,Jayesh, 25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)  
(3,Jignesh ,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)  
(3,Jignesh ,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)  
(4,Dharmesh,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)  
(5,Hardik,27,Bhopal,8500,,,,)  
(6,Komal,22,MP,4500,,,,)  
(7,Muffy,24,Indore,10000,,,,)
```

PRACTICAL 8

Aim: Store the basic information about students such as roll no, name date of birth and address of student using various collection types such as list, set and map

Source Code:

```
import java.util.HashMap;

public class Student {
    private String enroll,name;

    public Student(String enroll, String name) {
        this.enroll = enroll;
        this.name = name;
    }

    public String getEnroll() {
        return enroll;
    }

    public void setEnroll(String enroll) {
        this.enroll = enroll;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
public static void main(String[] args){ java.util.HashMap<String,Student> list1 = new
HashMap<>(); Student tt=new Student("1001","Rahul");

Student temp1=new Student("1002","Vijay");

list1.put("1001",tt);

list1.put("1002",temp1);

System.err.println("Student List using Map");

Student t1=list1.get("1001");

System.err.println("Enroll::"+t1.getEnroll());

System.err.println("Name::"+t1.getName());

Student t2=list1.get("1002");

System.err.println("Enroll::"+t2.getEnroll());

System.err.println("Name::"+t2.getName());

java.util.ArrayList<Student> list_1=new java.util.ArrayList<>();

list_1.add(tt);

list_1.add(temp1);

System.err.println("\nStudent List using List");

for(int ii=0;ii<list_1.size();ii++){

Student te=list_1.get(ii);

System.err.println("Enroll::"+te.getEnroll());

System.err.println("Name::"+te.getName());

} java.util.Set<Student> set = new java.util.HashSet<>();

set.add(tt);

set.add(temp1);

System.err.println("\nStudent List using Set");

java.util.Iterator iterator = set.iterator();

while(iterator.hasNext()){
```

```
Student te= (Student) iterator.next();

System.err.println("Enroll::"+te.getEnroll());

System.err.println("Name::"+te.getName());

}

}

}
```

Output:

```
[BDAprac-Air :BDA Practical-8 $ javac Student.java
[BDAprac-Air :BDA Practical-8 $ java Student
Student List using Map
Enroll::1001
Name::Rahul
Enroll::1002
Name::Vijay

Student List using List
Enroll::1001
Name::Rahul
Enroll::1002
Name::Vijay

Student List using Set
Enroll::1001
Name::Rahul
Enroll::1002
Name::Vijay
```


PRACTICAL: 9

Aim: Basic CRUD operations in MongoDB

Starting MongoDB Server

Open terminal and launch mongod server

```
$cd /usr/local/mongodb
```

```
$/mongod
```

Open another terminal and launch the MongoDB shell

```
$cd /usr/local/mongodb
```

```
$/mongo
```

Create:

Creating a db and a Collection and inserting a document in db Inserting multiple document

```
> use Practicals
switched to db Practicals
> db.createCollection("Students",{autoIndexed:true})
{ "ok" : 1 }
> db.Students.insert({"name":"Bruce Wayne","DOB":new Date("1996-07-24"),"Blood":"B+"})
WriteResult({ "nInserted" : 1 })
```

```
> db.Students.insert([{"name":"Oliver Queen","DOB":new Date("1997-12-11"),"Blood":"B+"},{ "name":"Diana Prince","DOB":new Date("1998-06-20"),"Blood":"B+"}])
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 2,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
```

Read: Reading regular data

```
> db.Students.find()
{ "_id" : ObjectId("59d79031780f83df11832123"), "name" : "Bruce Wayne", "DOB" : ISODate("1996-07-24T00:00:00Z"), "Blood" : "B+" }
{ "_id" : ObjectId("59d792c9780f83df11832124"), "name" : "Clark Kent", "DOB" : ISODate("1996-09-11T00:00:00Z"), "Blood" : "A+" }
{ "_id" : ObjectId("59d8cdf072288e9b28656839"), "name" : "Barry Allen", "DOB" : ISODate("1996-12-22T00:00:00Z"), "Blood" : "A+" }
{ "_id" : ObjectId("59d8cf2b72288e9b2865683a"), "name" : "Oliver Queen", "DOB" : ISODate("1997-12-11T00:00:00Z"), "Blood" : "B+" }
{ "_id" : ObjectId("59d8cf2b72288e9b2865683b"), "name" : "Diana Prince", "DOB" : ISODate("1998-06-20T00:00:00Z"), "Blood" : "B+" }
```

Reading formatted Data

```
> db.Students.find().pretty()
{
  "_id" : ObjectId("59d79031780f83df11832123"),
  "name" : "Bruce Wayne",
  "DOB" : ISODate("1996-07-24T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d792c9780f83df11832124"),
  "name" : "Clark Kent",
  "DOB" : ISODate("1996-09-11T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cdf072288e9b28656839"),
  "name" : "Arthur Cury",
  "DOB" : ISODate("1996-12-22T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683a"),
  "name" : "Oliver Queen",
  "DOB" : ISODate("1997-12-11T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683b"),
  "name" : "Diana Prince",
  "DOB" : ISODate("1998-06-20T00:00:00Z"),
  "Blood" : "B+"
}
```

Queried output

```
> db.Students.find({"name":{"$ne":"Barry Allen"}},{"name":1,"_id":0,"Blood":1}).pretty()
{ "name" : "Bruce Wayne", "Blood" : "B+" }
{ "name" : "Clark Kent", "Blood" : "A+" }
{ "name" : "Oliver Queen", "Blood" : "B+" }
{ "name" : "Diana Prince" }
```

Update:

```
> db.Students.update({"name":"Barry Allen"},{$set:{'name':"Arthur Cury"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Delete:

```
> db.Students.find().pretty()
{
  "_id" : ObjectId("59d79031780f83df11832123"),
  "name" : "Bruce Wayne",
  "DOB" : ISODate("1996-07-24T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d792c9780f83df11832124"),
  "name" : "Clark Kent",
  "DOB" : ISODate("1996-09-11T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cdf072288e9b28656839"),
  "name" : "Arthur Cury",
  "DOB" : ISODate("1996-12-22T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683a"),
  "name" : "Oliver Queen",
  "DOB" : ISODate("1997-12-11T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683b"),
  "name" : "Diana Prince",
  "DOB" : ISODate("1998-06-20T00:00:00Z"),
  "Boold" : "B+"
}
```

```
> db.Students.remove({"name":"Clark Kent"})
WriteResult({ "nRemoved" : 1 })
> db.Students.find().pretty()
{
  "_id" : ObjectId("59d79031780f83df11832123"),
  "name" : "Bruce Wayne",
  "DOB" : ISODate("1996-07-24T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8cdf072288e9b28656839"),
  "name" : "Arthur Cury",
  "DOB" : ISODate("1996-12-22T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683a"),
  "name" : "Oliver Queen",
  "DOB" : ISODate("1997-12-11T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683b"),
  "name" : "Diana Prince",
  "DOB" : ISODate("1998-06-20T00:00:00Z"),
  "Boold" : "B+"
}
```

PRACTICAL: 10

Aim: To find and retrieve documents from Students collection

Starting MongoDB Server

Open terminal and launch mongod server

```
$cd /usr/local/mongodb
```

```
$/mongod
```

Open another terminal and launch the MongoDB shell

```
$cd /usr/local/mongodb
```

```
$/mongo
```

Find and retrieve using operational greater than (\$gt):

```
> db.Students.find({"name":{"$gt":"H"}}).pretty()
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683a"),
  "name" : "Oliver Queen",
  "DOB" : ISODate("1997-12-11T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8d9ad174184b56d775b18"),
  "name" : "Hal Jordan",
  "DOB" : ISODate("1994-07-14T00:00:00Z")
}
```

Find and retrieve using operational and (\$and) ,or (\$or) and operational less than (\$lt):

```
> db.Students.find({$and:[{"Blood":"B+"}, {"name":{"$lt":"F"}}]}).pretty()
{
  "_id" : ObjectId("59d79031780f83df11832123"),
  "name" : "Bruce Wayne",
  "DOB" : ISODate("1996-07-24T00:00:00Z"),
  "Blood" : "B+"
}
```

```
> db.Students.find({$or:[{"Blood":"B+"}, {"Blood":"AB+"}]}).pretty()
{
  "_id" : ObjectId("59d79031780f83df11832123"),
  "name" : "Bruce Wayne",
  "DOB" : ISODate("1996-07-24T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683a"),
  "name" : "Oliver Queen",
  "DOB" : ISODate("1997-12-11T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8d9ad174184b56d775b17"),
  "name" : "Clark Kent",
  "DOB" : ISODate("1997-05-05T00:00:00Z"),
  "Blood" : "AB+"
}
{
  "_id" : ObjectId("59d8d9ad174184b56d775b18"),
  "name" : "Hal Jordan",
  "DOB" : ISODate("1994-07-14T00:00:00Z"),
  "Blood" : "AB+"
}
```

Updating the data using operational query:

```
> db.Students.update({$and:[{"Blood":"B+"}, {"name":"Oliver Queen"}]}, {$set:{"Blood":"A+"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```

> db.Students.find().pretty()
{
  "_id" : ObjectId("59d79031780f83df11832123"),
  "name" : "Bruce Wayne",
  "DOB" : ISODate("1996-07-24T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8cdf072288e9b28656839"),
  "name" : "Arthur Cury",
  "DOB" : ISODate("1996-12-22T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683a"),
  "name" : "Oliver Queen",
  "DOB" : ISODate("1997-12-11T00:00:00Z"),
  "Blood" : "A+"
}
{
  "_id" : ObjectId("59d8cf2b72288e9b2865683b"),
  "name" : "Diana Prince",
  "DOB" : ISODate("1998-06-20T00:00:00Z"),
  "Blood" : "B+"
}
{
  "_id" : ObjectId("59d8d9ad174184b56d775b17"),
  "name" : "Clark Kent",
  "DOB" : ISODate("1997-05-05T00:00:00Z"),
  "Blood" : "AB+"
}
{
  "_id" : ObjectId("59d8d9ad174184b56d775b18"),
  "name" : "Hal Jordan",
  "DOB" : ISODate("1994-07-14T00:00:00Z"),
  "Blood" : "AB+"
}

```

OEP

Aim: Create system which can use of web search, web crawlers and web information retrieval.

Simple web crawler implementation in PHP which fetches basic information and images (if available) from entered URL.

Programs:

index.html

```

<!DOCTYPE html>

<html>

<head>

    <title> Web Crawl Sample </title>

</head>

<body>

```

```
<form method="post" action="crawl.php">

Enter URL:&nbsp;&nbsp; <input type="text" name="url" size="50" placeholder="e.g
http://www.google.com" />

&nbsp;&nbsp; <input type="submit" value="Submit" name="submit"
/> </form>

</body>

</html>
```

crawl.php

```
<?php

function preg_substr($start=0, $end, $str) // Regular expression
{
    $temp = preg_split($start, $str);
    $content = preg_split($end, $temp[1]);
    return $content[0];
}

function str_substr($start, $end, $str) // string
split {

    $temp = explode($start, $str, 2);

    $content = explode($end, $temp[1], 2);
    return $content[0];
}

function writelog($str)
{
    @unlink("log.txt");
    $open=fopen("log.txt","a" );
    fwrite($open,$str);
    fclose($open);
}
```

```
function getImage($url, $filename="", $dirName, $fileType, $type=0)
{
    if($url == ""){return false;}

    $defaultFileName = basename($url);
    $suffix = substr(strrchr($url,'.'), 1);

    if(!in_array($suffix, $fileType)){
        return false;
    }

    $filename = $filename == " ? time().rand(0,9).'.'.$suffix : $defaultFileName;

    if($type){
        $ch = curl_init();
        $timeout = 5;
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
        $file = curl_exec($ch);
        curl_close($ch);
    }else{
        ob_start();
        readfile($url);
        $file = ob_get_contents();
        ob_end_clean();
    }

    $dirName = $dirName.'/'.date('Y', time()).'.'.date('m',
time()).'.'.date('d',time()).'.'/; if(!file_exists($dirName)){
        mkdir($dirName, 0777, true);}
}
```



```

$res = fopen($dirName.$filename,'a');

fwrite($res,$file);

fclose($res);

return $dirName.$filename;

}

if($_SERVER['REQUEST_METHOD'] ==
"POST"){ $str = file_get_contents($_POST['url']);

$str = mb_convert_encoding($str, 'utf-8','iso-8859-1');

writelog($str);

//echo $str;

echo "Title:" . preg_substr('/<span id=
"btAsinTitle"[^>]*>/',/<Vspan>/', $str); echo '<br/>';

$imgurl= str_substr('var imageSrc = "',"$str);

echo '

```

Output: -

