PROJECT REPORT
On
Random Data Retrieval In MapReduce Paradigm Under Hadoop Platform

Prepared By:
Gazal Agarwal & Sapna Kumari
Penultimate year
BTECH(CSE)
Heritage Institute Of Technology, Kolkata

Under
Dr. Rajat K. De
Professor
Machine Intelligence Unit
Indian Statistical Institute
Kolkata

ACKNOWLEDGEMENT

We express our deep sense of gratitude to Prof.(Dr.) Dinabandhu Bhandari, Professor of CSE (HIT), PhD.(JU), M.Tech(ISI), B.Sc(CU) at Heritage Institute Of Technology, Kolkata for encouraging us to take training at Indian Statistical Institute, Kolkata .

We acknowledge with thanks the kind patronage, loving inspiration and timely guidance of our mentor Dr. Rajat K. De, Professor Machine Intelligence Unit, Indian Statistical Institute.

We are very thankful to our college mentor Professor Sandip Sammadar, Professor of CSE(HIT),M.Tech(WBUT), B.E.(BU) for his valuable guidance, keen interest and constructive criticism which has contributed immensely to the evolution of ideas during the project.

ABSTRACT:

In our project we have implemented an algorithm which aims to ease in the process of retrieving data randomly.
Hadoop is an open source distributed processing framework that manages data processing and storage for big data applications running in clustered systems. It is at the center of a growing ecosystem of big data technologies that are primarily used to support advanced analytics initiatives, including predictive analytics, data mining and machine learning applications. Hadoop can handle various forms of structured and unstructured data, giving users more flexibility for collecting, processing and analyzing data than relational databases and data warehouses provide.
However one limitation of Hadoop is that it only allows records to be accessed in a sequential fashion. For a very large file in which only a few records are to be fetched, this process is time consuming and slow. We have written mapreduce codes to allow selective data to be fetched in any order. This can allow the use of our algorithm in professional institutes where records of particular students or faculty is to accessed. We have encapsulated the functionality of query languages like SQL in our algorithm.

INTRODUCTION TO HADOOP

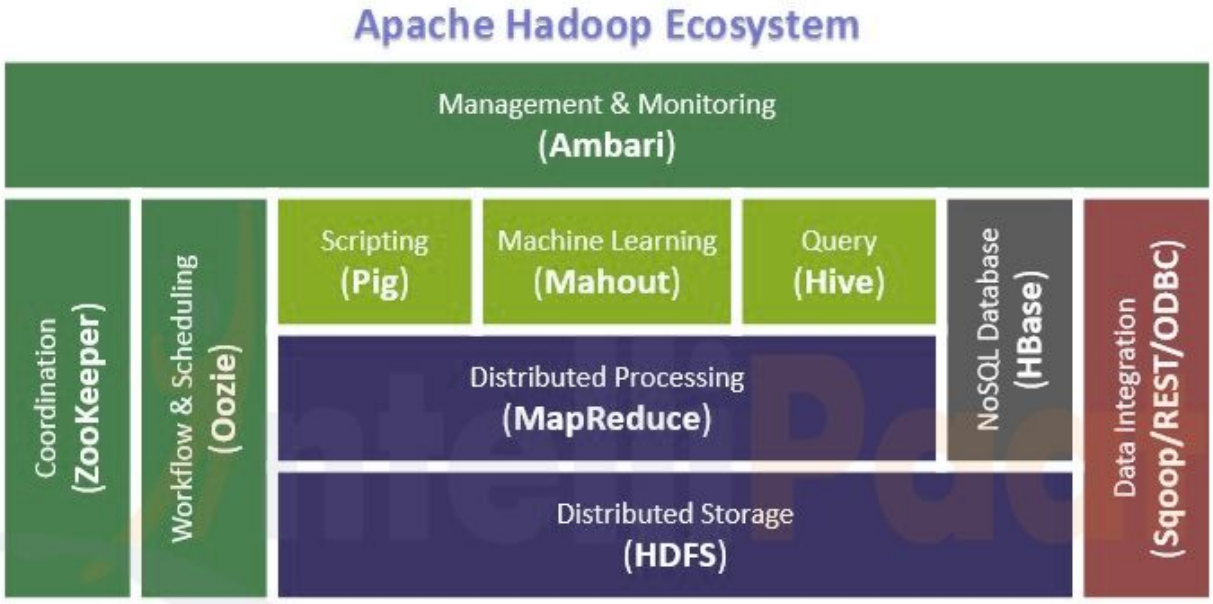
Apache Hadoop was born to enhance the usage and solve major issues of big data. The web media was generating loads of information on a daily basis, and it was becoming very difficult to manage the data of around one billion pages of content. In order of revolutionary, Google invented a new methodology of processing data popularly known as MapReduce. Later after a year Google published a white paper of Map Reducing framework where Doug Cutting and Mike Cafarella, inspired by the white paper and thus created Hadoop to apply these concepts to an open-source software framework which supported the Nutch search engine project. Considering the original case study, Hadoop was designed with a much simpler storage infrastructure facilities.

Apache Hadoop is the most important framework for working with Big Data. Hadoop biggest strength is scalability. It upgrades from working on a single node to thousands of nodes without any issue in a seamless manner.
The different domains of Big Data means we are able to manage the datas from videos, text medium, transactional data, sensor information, statistical data, social media conversations, search engine queries, ecommerce data, financial information, weather data, news updates, forum discussions, executive reports, and so on.
Google's **Doug Cutting** and his team members developed an Open Source Project namely known as HADOOP which allows you to handle the very large amount of data. Hadoop runs the applications on the basis of MapReduce where the data is processed in parallel and accomplish the entire statistical analysis on large amount of data.

It is a framework which is based on java programming. It is intended to work upon from a single server to thousands of machines each offering local computation and storage. It supports the large collection of data set in a distributed computing environment.
The Apache Hadoop software library based framework that gives permissions to distribute huge amount of data sets processing across clusters of computers using easy programming models.

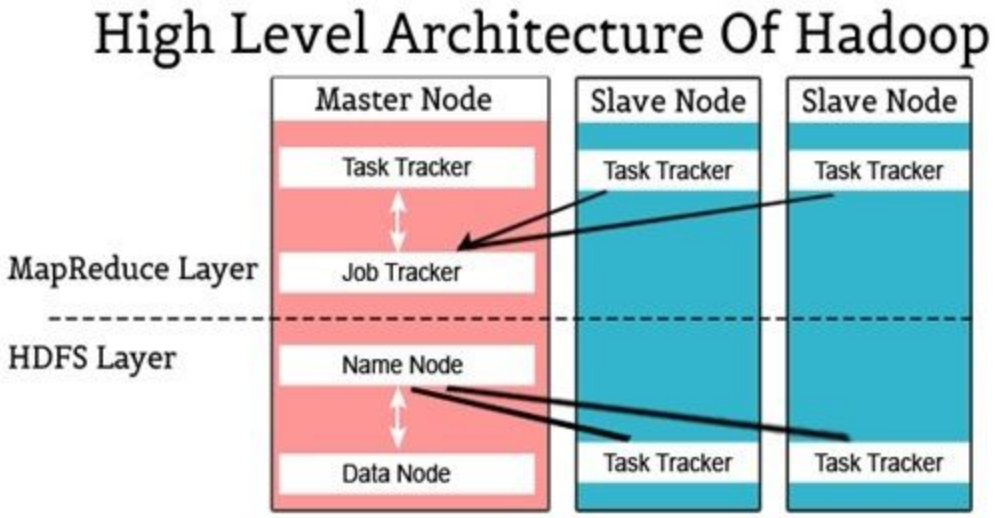
History of Hadoop

Hadoop was created by Doug Cutting and hence was the creator of Apache Lucene. It is the widely used text to search library. Hadoop has its origins in Apache Nutch which is an open source web search engine itself a part of the Lucene project.



The Hadoop High-level Architecture

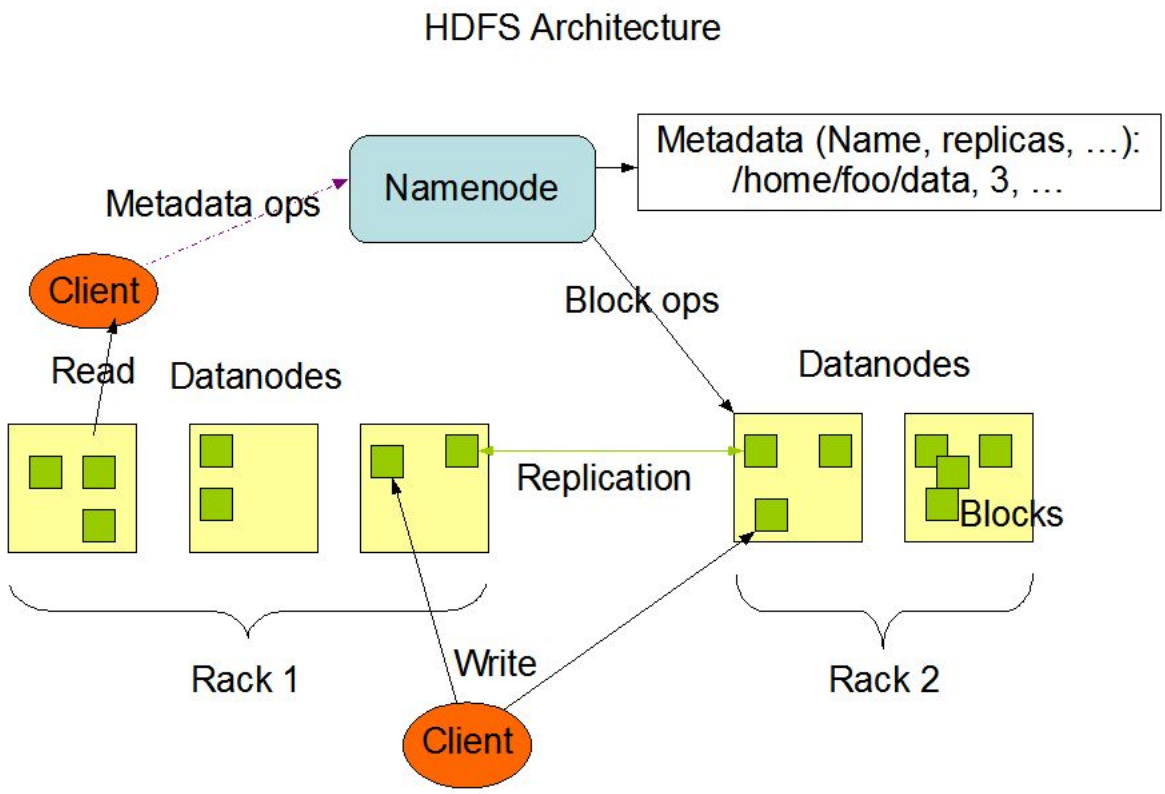
Hadoop Architecture based on the two main components namely MapReduce and HDFS



The Apache Hadoop Module

Hadoop Common: Includes the common utilities which supports the other Hadoop modules

HDFS: Hadoop Distributed File System provides unrestricted, high-speed access to the data application. It consists of a Master/Slave **architecture** in which Master is NameNode that stores meta-dta and Slave is DataNode that stores the actual data. **HDFS Architecture** consists of single NameNode and all the other nodes are DataNodes.



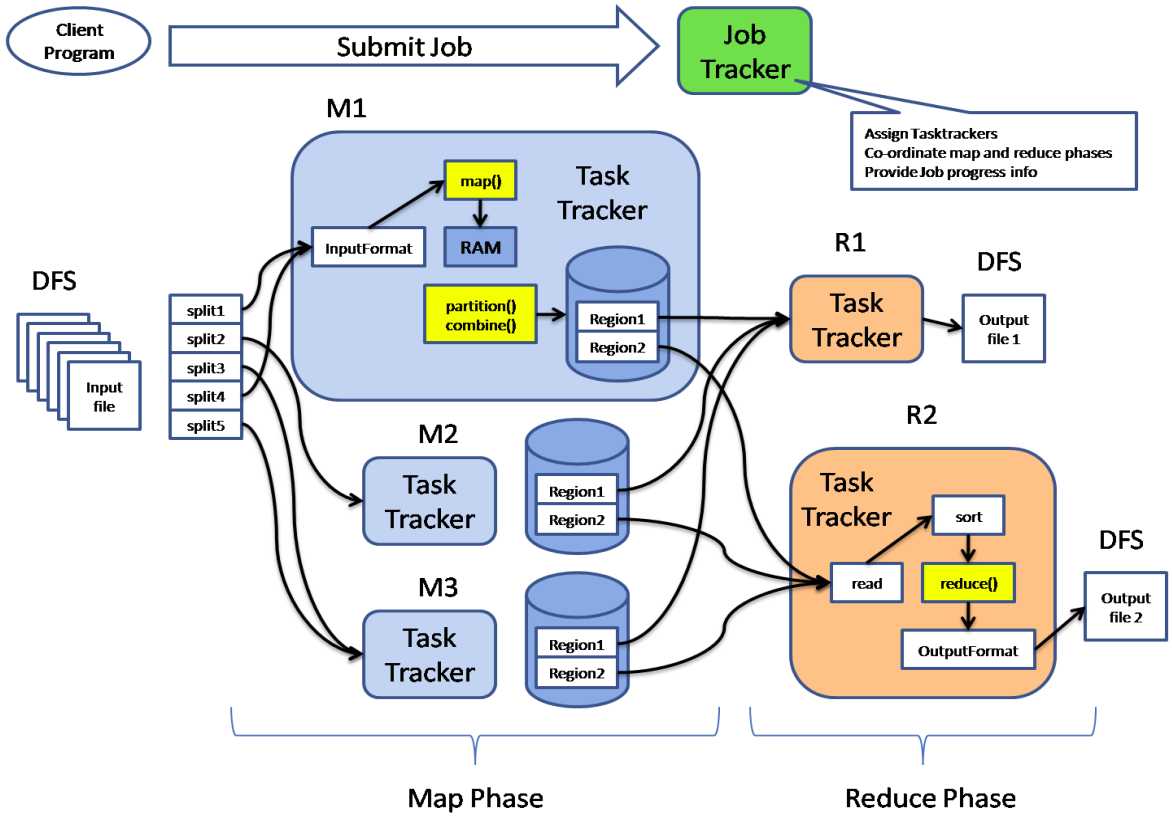
Hadoop YARN: This technology is basically used for scheduling of job and efficient management of the cluster resource.

MapReduce: Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.



Hadoop Multi-Node Cluster Set-up

Step 1: Set hostname (Master and Slave)

sudo apt-get update;

sudo apt-get install -y vim openssh-server

ifconfig [To check both IP Address]

sudo vi /etc/hosts [Delete 127.0.1.1 address and input these new address]
<IP Address> hadoop-master
<IP Address> hadoop-slave

sudo vi /etc/hostname
-> In Master [Replace default hostname]
hadoop-master
-> In Slave [Replace default hostname]
hadoop-slave
sudo reboot
ping hadoop-master [After reboot]
ping hadoop-slave

Step 2: Install Java (Master and Slave)
sudo apt-add-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer

Step 3: Create group and user (Master and Slave)
sudo addgroup hdgroup
sudo adduser -ingroup hdgroup hduser
sudo usermod -a -G sudo hduser

Step 4: Configure passwordless ssh (Master and Slave)
su - hduser
ssh-keygen -t rsa -P ""
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop-master
ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop-slave
ssh localhost [If no password asked, success. Exit afterward]
ssh hadoop-master [If no password asked, success. Exit afterward]
ssh hadoop-slave [If no password asked, success. Exit afterward]

Step 5: Install Hadoop (Master and Slave)
[Download hadoop packages using web browser in hadoop-master]
[Locate the downloaded package and copy to hduser home folder]
su - hduser
cp /home/user/packages/hadoop-2.7.1.tar.gz .
[Copy the downloaded package from hadoop-master to hadoop-slave]
scp ~/hadoop-2.7.1.tar.gz hduser@hadoop-slave:~/

Step 5: Cont. (Master and Slave)
tar xzfv hadoop-2.7.1.tar.gz [Extract the tar file]

sudo mkdir -p /usr/local/hadoop
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
sudo mv ~/hadoop-2.7.1/* /usr/local/hadoop/
ls - la /usr/local/hadoop
sudo chown -R hduser:hdgroup /usr/local/hadoop
sudo chown -R hduser:hdgroup /usr/local/hadoop_tmp

 # vi ~/.bashrc
export HADOOP_HOME=/usr/local/hadoop
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

 # source ~/.bashrc
hadoop version [The output should shows the version]

Step 6: Hadoop Configuration (Master)
[The configuration file in /usr/local/hadoop/etc/hadoop that must be edited]:

1. hadoop-env.sh
2. core-site.xml
3. mapred-site.xml
4. yarn-site.xml
5. hdfs-site.xml
6. masters
7. Slaves

 # vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh [1]
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

 # vi /usr/local/hadoop/etc/hadoop/core-site.xml [2]
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://hadoop-master:54310</value>
</property>

<property>
<name>hadoop.tmp.dir</name>
<value>/usr/local/hadoop_tmp</value>
</property>
</configuration>

cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml

 # vi /usr/local/hadoop/etc/hadoop/mapred-site.xml [3]
<configuration>
<property>
<name>mapreduce.jobtracker.address</name>
<value>hadoop-master:54311</value>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

 # vi /usr/local/hadoop/etc/hadoop/yarn-site.xml [4]
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>hadoop-master:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>hadoop-master:8032</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>hadoop-master:8088</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>hadoop-master:8031</value>

```
</property>
<property>
<name>yarn.resourcemanager.admin.address</name>
<value>hadoop-master:8033</value>
</property>
<property>
<name>yarn.nodemanager.vmem-check-enabled</name>
<value>false</value>
</property>
<property>
<name>yarn.log-aggregation-enable</name>
<value>true</value>
</property>
</configuration>
```

 # vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml [5]

```
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
<property>
<name>dfs.client.socket-timeout</name>
<value>3000000</value>
</property>
<property>
<name>dfs.datanode.socket.write.timeout</name>
<value>3000000</value>
</property>
</configuration>
```

 # vi /usr/local/hadoop/etc/hadoop/masters [6]

```
hadoop-master
```

 # vi /usr/local/hadoop/etc/hadoop/slaves [7]
hadoop-slave [Replace localhost]

[IMPORTANT - Execute the command only one time on Master]
hdfs namenode -format

[Copy all the edited files from Master to Slave]
cd /usr/local/hadoop/etc/hadoop
scp core-site.xml hadoop-env.sh hdfs-site.xml mapred-site.xml yarn-site.xml slaves masters hduser@hadoop-slave:/usr/local/hadoop/etc/hadoop

Step 7: Start Hadoop services (Master)
su - hduser [Must start all services as hduser]
start-dfs.sh
start-yarn.sh
jps [To verify installation and should see something like this]
xxxxx SecondaryNameNode
xxxxx Resource Manager
xxxxx NameNode
xxxxx Jps

```
hduser@hadoop-master:~$ su - hduser
Password:
hduser@hadoop-master:~$ start-dfs.sh
18/11/21 00:05:43 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Starting namenodes on [hadoop-master]
The authenticity of host 'hadoop-master (192.168.32.10)' can't be established.
ECDSA key fingerprint is SHA256:XKwKRQW9fQsJKhm+QErI5iQ71q2r6+lvus0O0R6NOkE.
Are you sure you want to continue connecting (yes/no)? yes
hadoop-master: Warning: Permanently added 'hadoop-master,192.168.32.10' (ECDSA) to the list of known hosts.
hduser@hadoop-master's password:
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-hadoop-master.out
hadoop-slave: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-hadoop-slave.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is SHA256:XKwKRQW9fQsJKhm+QErI5iQ71q2r6+lvus0O0R6NOkE.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
hduser@0.0.0.0's password:
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-hadoop-master.out
18/11/21 00:06:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
hduser@hadoop-master:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-hadoop-master.out
hadoop-slave: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-hadoop-slave.out
hduser@hadoop-master:~$ jps
3460 NameNode
3832 ResourceManager
3672 SecondaryNameNode
4092 Jps
hduser@hadoop-master:~$
```

Step 8: Start Hadoop history server (Slave)

su - hduser [Must start history server as hduser]
mr-jobhistory-daemon.sh start historyserver
jps [To verify installation and should see something like this]
xxxxx NodeManager
xxxxx Datanode
xxxxx JobHistoryServer
xxxxx Jps

```
lazag@hadoop-slave:~$ su - hduser
Password:
hduser@hadoop-slave:~$ sudo vi ~/.bashrc
[sudo] password for hduser:
hduser@hadoop-slave:~$ source ~/.bashrc
hduser@hadoop-slave:~$ hadoop version
Hadoop 2.7.1
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled by jenkins on 2015-06-29T06:04Z
Compiled with protoc 2.5.0
From source with checksum fc0a1a23fc1868e4d5ee7fa2b28a58a
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.1.jar
hduser@hadoop-slave:~$ mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /usr/local/hadoop/logs/mapred-hduser-historyserver-hadoop-slave.out
hduser@hadoop-slave:~$ jps
3141 NodeManager
3303 JobHistoryServer
3015 DataNode
3338 Jps
```

Step 9: Run sample job (Master)
cd /usr/local/hadoop/share/hadoop/mapreduce
hadoop jar hadoop-mapreduce-examples-2.7.1.jar pi 5 10
[Check job log at slave]
tail -f /usr/local/hadoop/logs/yarn-hduser-nodemanager-hadoop-slave.log

Step 10: Check Hadoop Health and Jobs using Web UI (Master and Slave)
1. Hadoop Resource Manager -> http://hadoop-master:8088
2. Hadoop Cluster Health -> http://hadoop-master:50070
3. Hadoop Job History -> http://hadoop-master:19888

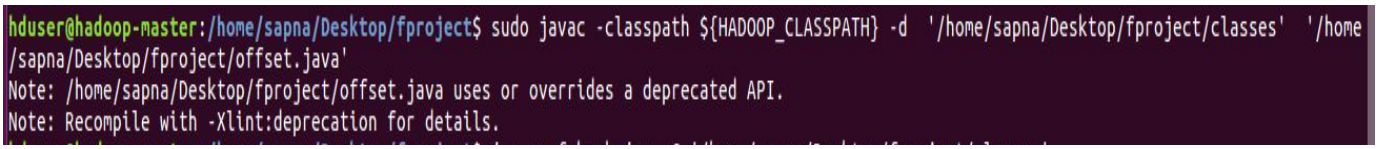
Steps for running the java MapReduce code on hadoop platform :

1.Set the hadoop classpath.

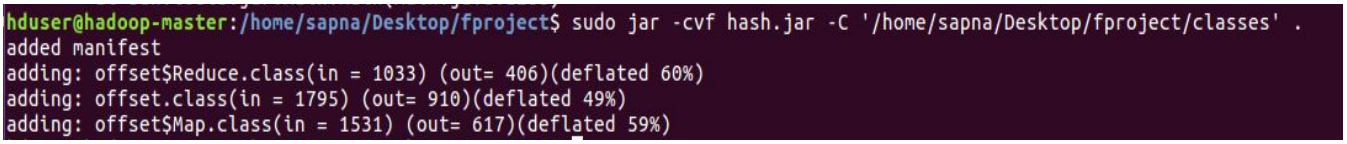
```
hduser@hadoop-master:~$ export HADOOP_CLASSPATH=$(hadoop classpath)
hduser@hadoop-master:~$ echo $HADOOP_CLASSPATH
/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/*:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/share/hadoop
/hdfs:/usr/local/hadoop/share/hadoop/hdfs/lib/*:/usr/local/hadoop/share/hadoop/hdfs/*:/usr/local/hadoop/share/hadoop/yarn/lib/*:/usr/local/hado
op/share/hadoop/yarn/*:/usr/local/hadoop/share/hadoop/mapreduce/lib/*:/usr/local/hadoop/share/hadoop/mapreduce/*:/usr/local/hadoop/contrib/capa
city-scheduler/*.jar
hduser@hadoop-master:~$ sudo ~/.bashrc
sudo: /home/hduser/.bashrc: command not found
hduser@hadoop-master:~$ sudo vi ~/.bashrc
hduser@hadoop-master:~$ echo $HADOOP_CLASSPATH
/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/*:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/share/hadoop
/hdfs:/usr/local/hadoop/share/hadoop/hdfs/lib/*:/usr/local/hadoop/share/hadoop/hdfs/*:/usr/local/hadoop/share/hadoop/yarn/lib/*:/usr/local/hado
op/share/hadoop/yarn/*:/usr/local/hadoop/share/hadoop/mapreduce/lib/*:/usr/local/hadoop/share/hadoop/mapreduce/*:/usr/local/hadoop/contrib/capa
city-scheduler/*.jar
```

2.Create a folder(input folder inside project directory) on hdfs using *mkdir* and upload the input file(grade.csv) using *put* in that folder.

3.Compile the code(offset.java) for generating the hash file using the offset value.



4.Create a jar file(hash.jar) for compiled classes of offset.java.



5.Run the code on hadoop platform by providing address of the input and the output files' directory.

```
hduser@hadoop-master:/home/sapna/Desktop/fproject$ hadoop jar hash.jar offset project/input project/hash
18/11/21 01:00:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
18/11/21 01:00:43 INFO client.RMProxy: Connecting to ResourceManager at hadoop-master/192.168.32.10:8032
18/11/21 01:00:43 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execut
e your application with ToolRunner to remedy this.
18/11/21 01:00:44 INFO input.FileInputFormat: Total input paths to process : 1
18/11/21 01:00:45 INFO mapreduce.JobSubmitter: number of splits:3
18/11/21 01:00:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1542738991474_0006
18/11/21 01:00:45 INFO impl.YarnClientImpl: Submitted application application_1542738991474_0006
18/11/21 01:00:45 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8088/proxy/application_1542738991474_0006/
18/11/21 01:00:45 INFO mapreduce.Job: Running job: job_1542738991474_0006
18/11/21 01:01:05 INFO mapreduce.Job: Job job_1542738991474_0006 running in uber mode : false
18/11/21 01:01:05 INFO mapreduce.Job:  map 0% reduce 0%
18/11/21 01:01:47 INFO mapreduce.Job:  map 100% reduce 0%
18/11/21 01:01:51 INFO mapreduce.Job: Job job_1542738991474_0006 completed successfully
18/11/21 01:01:51 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=346329
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=5752487
                HDFS: Number of bytes written=113374
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=6
        Job Counters
                Launched map tasks=3
                Data-local map tasks=3
                Total time spent by all maps in occupied slots (ms)=122601
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=122601
                Total vcore-seconds taken by all map tasks=122601
                Total megabyte-seconds taken by all map tasks=125543424
        Map-Reduce Framework
                Map input records=7594
                Map output records=7594
                Input split bytes=381
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=1716
                CPU time spent (ms)=10750
                Physical memory (bytes) snapshot=491900928
                Virtual memory (bytes) snapshot=5897224192
                Total committed heap usage (bytes)=337641472
        File Input Format Counters
                Bytes Read=5752106
        File Output Format Counters
                Bytes Written=113374
hduser@hadoop-master:/home/sapna/Desktop/fproject$
```

6.Now the output can be viewed by using -cat command followed by the output folder.
*$hdfs dfs -cat project/hash/**

Sample output is as follows:

```
45897307        5736563
45897308        5737076
45920401        5737586
47503301        5738051
47503302        5738545
47647001        5739044
47657701        5739519
47657702        5739985
47691101        5740451
48065701        5740943
48154401        5741445
48387801        5741945
48400201        5742450
48414501        5742928
48621101        5743406
hduser@hadoop-master:/home/sapna/Desktop/fproject$
```

7.Similarly we compile the java file(access.java) and then create the access.jar file and run the code on hadoop platform where the input file is the file containing the queries asked by the user. The output displays the required information.

```
hduser@hadoop-master:/home/sapna/Desktop/fproject$ sudo javac -classpath ${HADOOP_CLASSPATH} -d  '/home/sapna/Desktop/fproject/classes1'  '/hom
e/sapna/Desktop/fproject/access.java'
[sudo] password for hduser:
Note: /home/sapna/Desktop/fproject/access.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
hduser@hadoop-master:/home/sapna/Desktop/fproject$ sudo jar -cvf access.jar -C '/home/sapna/Desktop/fproject/classes1' .
added manifest
adding: access.class(in = 1603) (out= 860)(deflated 46%)
adding: access$Map.class(in = 2986) (out= 1346)(deflated 54%)
adding: access$Reduce.class(in = 1033) (out= 406)(deflated 60%)
hduser@hadoop-master:/home/sapna/Desktop/fproject$
```

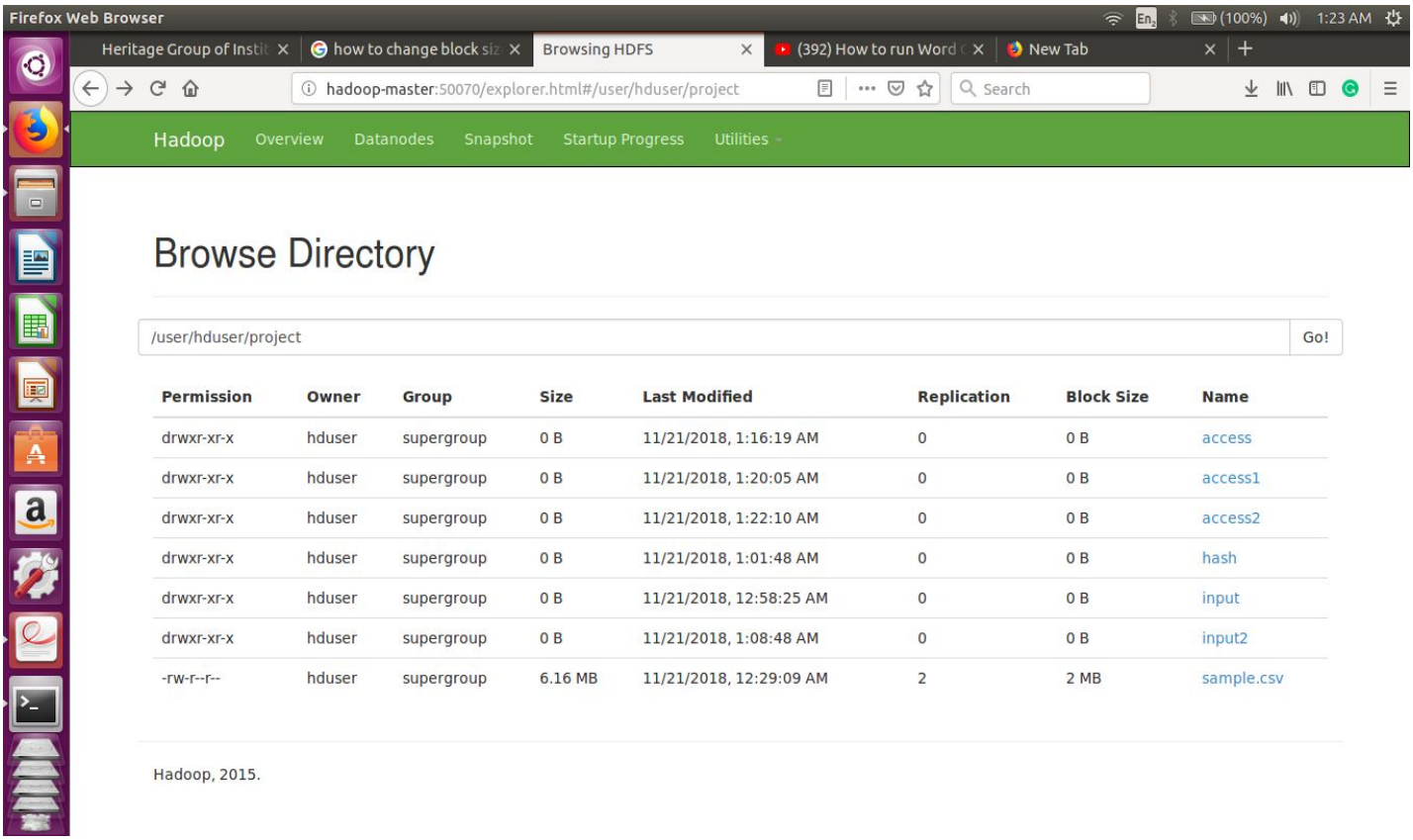
```
able
hduser@hadoop-master:/home/sapna/Desktop/fproject$ hadoop jar access.jar access project/input2 project/access
18/11/21 01:09:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
18/11/21 01:09:35 INFO client.RMProxy: Connecting to ResourceManager at hadoop-master/192.168.32.10:8032
18/11/21 01:09:35 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execut
e your application with ToolRunner to remedy this.
18/11/21 01:09:36 INFO input.FileInputFormat: Total input paths to process : 1
18/11/21 01:09:36 INFO mapreduce.JobSubmitter: number of splits:1
18/11/21 01:09:36 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1542738991474_0007
18/11/21 01:09:36 INFO impl.YarnClientImpl: Submitted application application_1542738991474_0007
18/11/21 01:09:36 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8088/proxy/application_1542738991474_0007/
18/11/21 01:09:36 INFO mapreduce.Job: Running job: job_1542738991474_0007
18/11/21 01:09:55 INFO mapreduce.Job: Job job_1542738991474_0007 running in uber mode : false
18/11/21 01:09:55 INFO mapreduce.Job:  map 0% reduce 0%
18/11/21 01:10:09 INFO mapreduce.Job:  map 100% reduce 0%
18/11/21 01:10:10 INFO mapreduce.Job: Job job_1542738991474_0007 completed successfully
18/11/21 01:10:10 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=115446
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=298
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=29
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=11237
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=11237
                Total vcore-seconds taken by all map tasks=11237
                Total megabyte-seconds taken by all map tasks=11506688
        Map-Reduce Framework
                Map input records=24
                Map output records=0
                Input split bytes=124
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=234
                CPU time spent (ms)=1880
                Physical memory (bytes) snapshot=163381248
                Virtual memory (bytes) snapshot=1957539840
                Total committed heap usage (bytes)=111673344
        File Input Format Counters
                Bytes Read=174
        File Output Format Counters
                Bytes Written=0
hduser@hadoop-master:/home/sapna/Desktop/fproject$
```

Final output is as follows:

```
hduser@hadoop-master:/home/sapna/Desktop/fproject$ hadoop dfs -cat project/access/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/11/21 01:17:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
1       100690,02503400,025034,Amridge University,NULL,0.82999998331069,70.8799972534179,22.5300006866455,1.28999996185302,6.94000005722046,13.
2299995422363,7.01000022888183,94.0899963378906,53683.7,10.6499996185302,3.59999990463256,10.8400001525878,57,168,46100,40100,6800,22900,59900,
79500,34900,61,75,32,PrivacySuppressed,PrivacySuppressed,159,80,88,0.696,PrivacySuppressed,46400,PrivacySuppressed,PrivacySuppressed,PrivacySup
pressed,PrivacySuppressed,37700,53700,37,146,40000,33200,7300,18500,50900,75400,33600,67,60,PrivacySuppressed,PrivacySuppressed,PrivacySuppress
ed,141,76,70,0.658,31300,PrivacySuppressed,PrivacySuppressed,PrivacySuppressed,PrivacySuppressed,PrivacySuppressed,28100,53000,41,172,38700,278
00,0.66279071569442,47,193,39200,34800,5800,18400,52200,78700,30500,0.668,41,174,42400,30500,0.7356321811676
2       100706,00105500,001055,University of Alabama in Huntsville,NULL,0.28999999165534,76.3799972534179,18.9799995422363,1.41999995708465,2.5
4999995231628,17.6700000762939,8.90999984741211,95.2699966430664,58688.62,9.36999988555908,3.64000010490417,10.9300003051757,188,1361,50500,456
00,12900,27300,70200,93300,32700,527,454,380,155,851,510,731,630,0.779,47000,53400,52100,46600,49800,51800,43800,58400,139,1258,39200,35500,910
0,21800,54300,70300,25800,503,376,379,187,814,444,619,639,0.685,37100,42900,38300,31800,37100,43000,34300,44000,137,1155,43400,27600,0.71861469
745635,168,1413,44400,40600,12400,23300,62100,80700,27600,0.73,152,1106,50200,37700,0.78119349479675
3       100724,00100500,001005,Alabama State University,NULL,0.10999999940395,42.6899986267089,52.3199996948242,1.40999996662139,4.090000152587
89,11.8100004196167,6.76000022888183,94.5299987792968,46065.2,16.9599990844726,4.80999994277954,10.6700000762939,285,1977,29500,26700,7100,1480
0,40800,55400,20500,1139,636,202,799,1657,320,1009,968,0.528,28400,30700,32400,27600,29100,31800,27800,31300,188,1594,23100,21000,4300,12200,31
600,39800,15100,1004,442,148,613,1352,242,829,765,0.393,21800,25300,25400,19600,22700,25200,22000,24300,251,1587,25200,31100,0.42722117900848,2
17,1621,26000,23900,5400,13800,35700,48400,17300,0.468,297,1810,29400,20000,0.52983427047729
4       100751,00105100,001051,The University of Alabama,NULL,0.10000000149011,75.3499984741211,21.0599994659423,1.20000004768371,2.41000008583
068,16.4799995422363,9.21000003814697,96.0800018310547,57928.41,10.0500001907348,3.25999999046325,10.8999996185302,409,3648,49900,42700,13100,2
8000,64300,85000,42600,1081,1065,1502,572,3060,588,1941,1707,0.786,46000,50100,52400,44900,50300,47600,42600,58100,320,4322,39000,34800,9800,21
900,48900,61600,41800,1089,1168,2065,583,3697,625,2247,2075,0.695,39700,37600,39500,34600,38800,40400,34400,44100,292,3088,41700,42100,0.718911
94581985,366,4165,43100,38400,11500,25100,53900,77400,39200,0.753,321,3536,47200,49300,0.77460408210754
hduser@hadoop-master:/home/sapna/Desktop/fproject$
```

View of HDFS:



CODE TO GENERATE HASH FILE

offset.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

public class offset
{
public static class Map extends Mapper<LongWritable,Text,Text,Text> {
private int i=0;
public void map(LongWritable key, Text value,Context context) throws
IOException,InterruptedException{
String line = value.toString();
String [] arr = line.split("\t");
context.write( new Text(arr[0]) , new Text(key.toString()));
}
}

public static class Reduce extends Reducer<Text,Text,Text,Text> {
public void reduce(Text key, Text values,Context context) throws
IOException,InterruptedException {

context.write(key, values);
}
}

public static void main(String[] args) throws Exception {

Configuration conf= new Configuration();
```

```
Job job = new Job(conf,"random access");
job.setJarByClass(offset.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setNumReduceTasks(0);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);
//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
//deleting the output path automatically from hdfs so that we don't have to delete it explicitly
outputPath.getFileSystem(conf).delete(outputPath);
//exiting the job only if the flag value becomes false
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

CODE TO ACCESS A FILE RANDOMLY

access.java

```
import java.io.IOException;
import java.lang.reflect.*;
import java.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.util.*;
import org.apache.hadoop.util.LineReader;
import org.apache.hadoop.mapreduce.InputSplit;
//import org.apache.hadoop.mapreduce.lib.input.TaggedInputSplit;
```

```
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;


public class access {

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

   FSDataInputStream fsin1,fsin2;
         private Text word = new Text();
   private int i=1;
        @Override
        public void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {

   try{
   Configuration conf = context.getConfiguration();
   Path p2= new Path("hdfs:///user/hduser/project/input/data123.txt");
   FileSystem fs2 = p2.getFileSystem(conf);
   fsin1 = fs2.open(p2);


   //Configuration conf = context.getConfiguration();

   Path p3= new Path("hdfs:///user/hduser/project/opt/part-m-00003");
  //p3 points to the hdfs location of the original file which the user wants to access.
  //FileSystem fs2 = p2.getFileSystem(conf);
   FileSystem fs3 = p3.getFileSystem(conf);
   fsin2 = fs3.open(p3);
   BufferedReader br=new BufferedReader(new InputStreamReader(fsin2));

   String line1;
   String line=value.toString();
   String line2=br.readLine();
   while(line2!=null)
   {
         String [] arr = line2.split("\t");
```

```
        if(arr[0].equalsIgnoreCase(line))
        {

        fsin1.seek(Long.parseLong(arr[1]));
        line1 = fsin1.readLine();
        word.set(line1);
        context.write(new Text(String.valueOf(i)), word);
        i++;
        line="";
        break;
        }
   line2=br.readLine();
   }

   }
   catch(IOException e){}

        }
        }

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


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

    context.write(key, values);
   }

        }

        public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();
        // conf.set("fs.defaultFS", "localhost:50070");
        Job job = new Job(conf);
        job.setJarByClass(access.class);
        job.setJobName("random access");
   job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
```

```
    job.setNumReduceTasks(0);
          job.setOutputKeyClass(Text.class);
          job.setOutputValueClass(Text.class);
          job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    Path outputPath = new Path(args[1]);
    //MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class, Map.class);
          //MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,
Map1.class)
     //MultipleInputs.addInputPath(job, new Path(args[2]),TextInputFormat.class, Map2.class);
          FileInputFormat.addInputPath(job, new Path(args[0]));
          FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);

          }

}
```

SUMMARY

Big data analytics refers to the tools and practices that can be used for transforming raw data into some meaningful information and hence processing a large amount of data in lesser amount of time as compared to the traditional data processing systems such as RDBMS. But there was no method to retrieve the data randomly from a huge file of more than 1 GB using map- reduce platform in hadoop paradigm. So we tried to reduce the access time of a file by developing a hash file in which we stored the byte offset value of each individual record of main file. The hash table is constructed just once and stored in hdfs. Now when a user tries to access a particular record he just have to give the set of values of the column which is a primary key in the table(for example roll number) in the form of a file, the values can be in any order. After receiving this input file from the user our program searches for the particular records in the main file using the its offset value stored in hash file and then stores it in a different file in hdfs. Since the hash file is created just once and stored in hdfs but we can use it as many times as we want unless it is updated due to insertion or deletion of records. Thus in this way we can save a large amount of time which is wasted in a sequential search of a record from a file.