Analysis of Algorithms for detecting De-duplications in HDFS.

18MCS0036 Sudhir Sardhara 18MCS0078 Rutuja k. sajane

SCOPE, Vellore Institute of Technology, Vellore, Tamil-Nadu

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Abstract

Todays challenging task is to large volume of data in the real time. DFS is one of the strategy to handle this kind of situations. But DFS has some drawbacks. To overcome this drawback HDFS Is comes in picture. Hadoop Distributed File System support data duplication to achieve high data reliability. To implement De-duplication values are computed for files using MD5, SHA1, SHA256 & SHA512 algorithms. The generated hash value for a file is checked with the existing file to identify the presence of duplication. If duplication exists, the system will not allow the user to upload the duplicate copy to the HDFS. The main objective is that whole-file deduplication together with insufficiency is a highly efficient means of lowering storage consumption, even in a backup scenario.

Introduction

In big data storage, large amount of data is stored in the form of Giga Bytes (GB) and Tera Bytes (TB). The data stored in big data is in unstructured form means data without any format. This large amount of data is collected from different sources that were increases duplicity in data. To find duplicate data and remove it from big data storage is a difficult problem also managing unstructured data or converting it into structured form is difficult problem to solve. To solve above mentioned problems, lots of techniques exists named File-level chunking, Block-Level Chunking. Block-Level Chunking further divided into Fixed-Size Chunking and Variable Size Chunking to find duplicate data

Proposed Methodology

HADOOP provides five kinds of services HADOOP services are Name node (Master node), Secondary name node (Master node), Data node (Slave node), Job trackers (Master node) and Task tracker (Slave node).

I) Name node: The name node acts as master server. To manage the file system namespace, name node will executes file system operation such as renaming, closing, opening files and opening directory, Name node contains all the information of data nodes and information will be maintained as a tree structure format.

II) Data nodes: The data node act as slave server. To perform read an instruction.

III) Job tracker: Job tracker will schedule jobs and track the assigned jobs to task tracker

IV) Task tracker: Task tracker will track the task and report status to job tracker.

V) Secondary Namenode: Secondary Namenode performs some internal housekeeping for the namenode. Despite its name, the secondary namenode is not a backup for the namenode and performs a completely different function. Secondary Namenode whole purpose is to have a checkpoint in HDFS. It is just a helper node for namenode.

Design of HDFS

HDFS is a file system designed for storing very large files, files may be gigabyte to terabytes in size. Today Hadoop cluster run on petabytes of data , along with streaming data access patterns that is write once and run anywhere, running on clusters of commodity hardware Hadoop does not require expensive and high reliable hardware to run .

File level De-duplication

1. File level De-duplication:It eliminates duplicate copies of the same file. This is also called as Single instance storage (SIS). File-level de-duplication performs to identify the multiple copies of the same file, that stores as first copy, and then just links the other references to the first file.

Architecture:

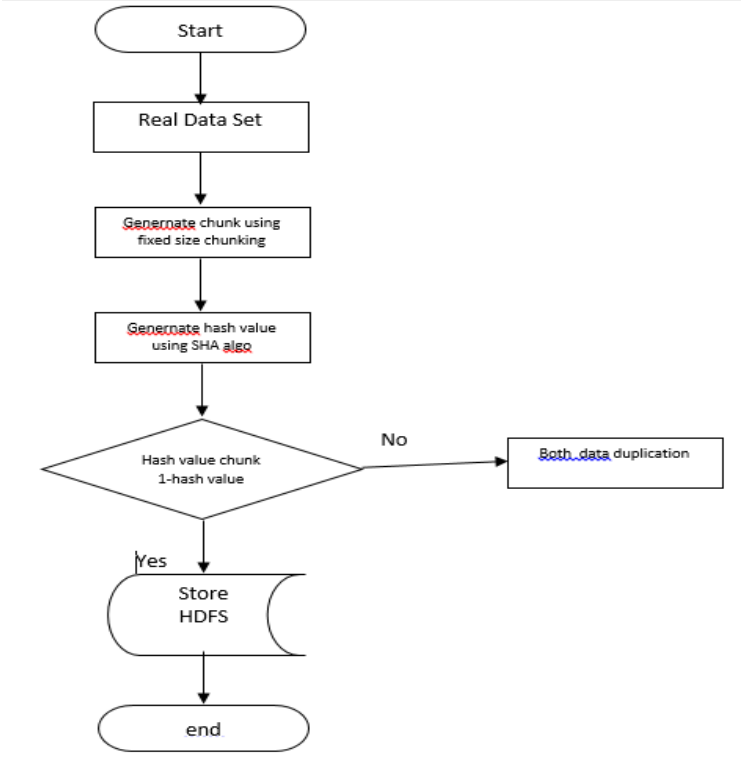
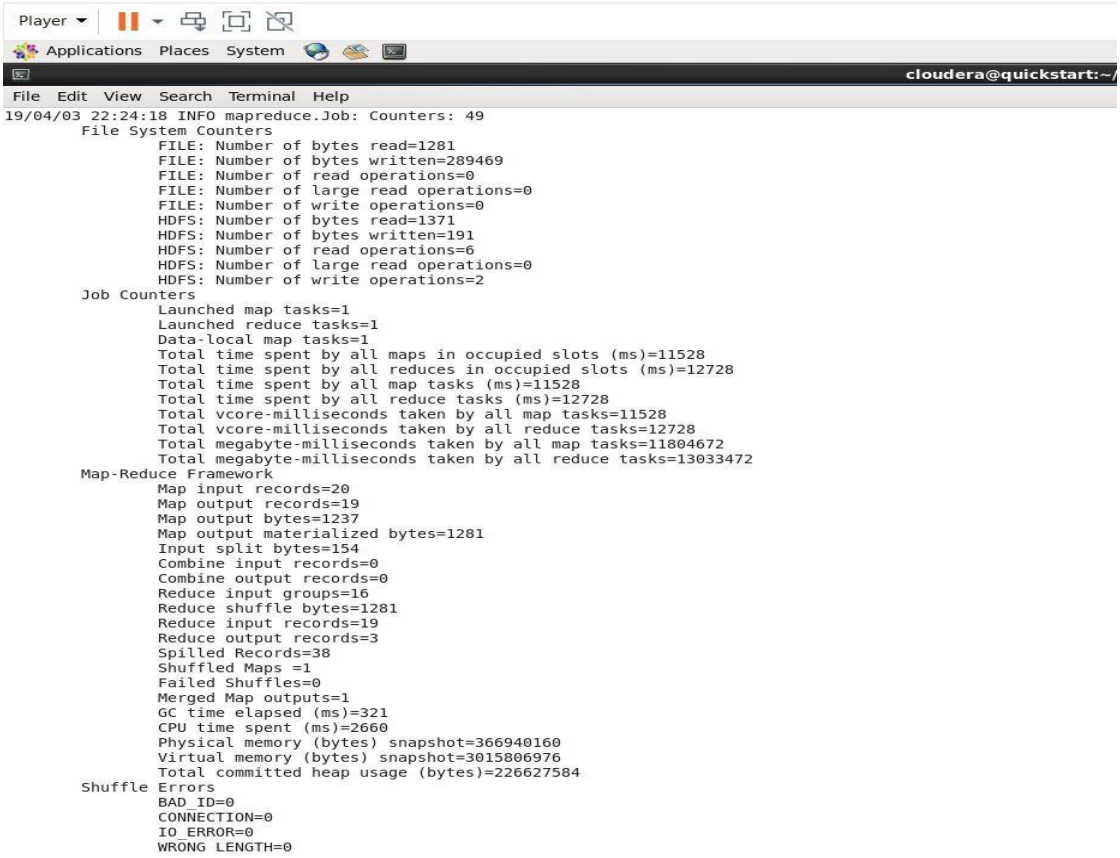
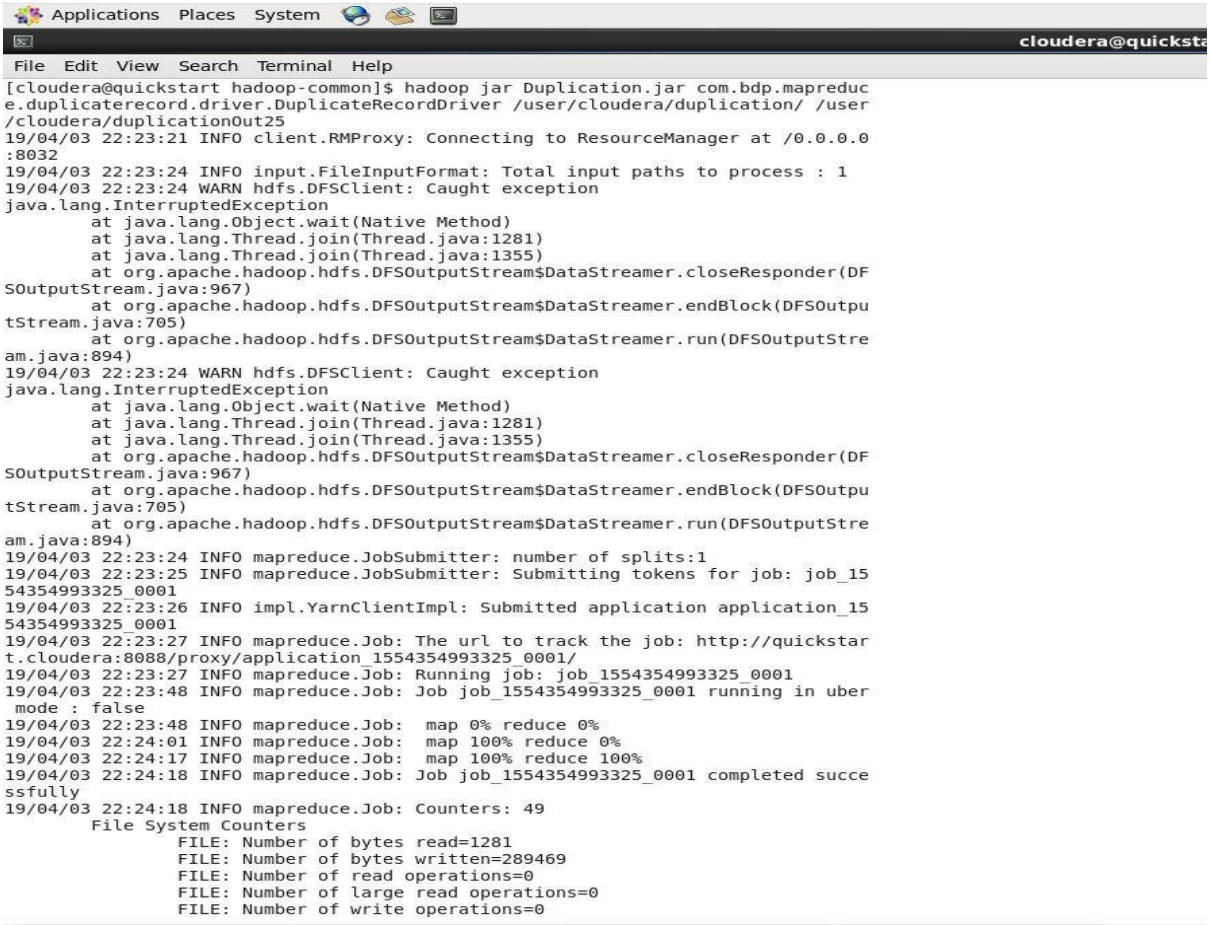


Fig:1

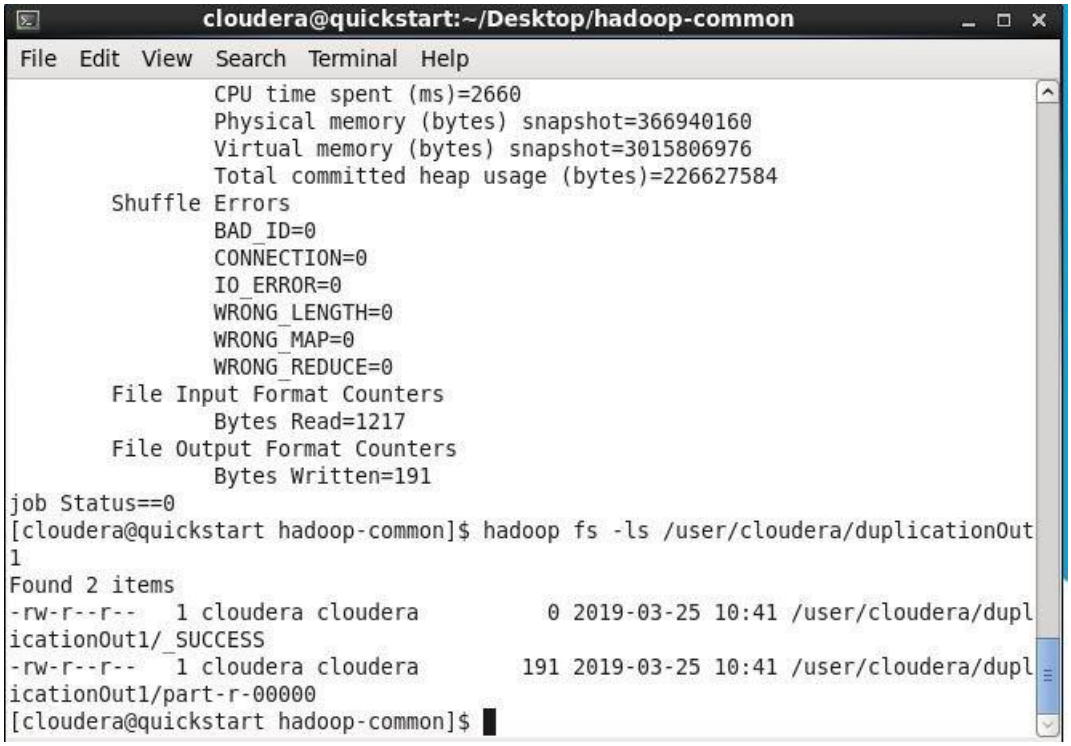
Related Work:

1. Creating folder in hadoop. And copy the dataset into that directory
2. Write map, reduce and driver program
3. Generate jar file

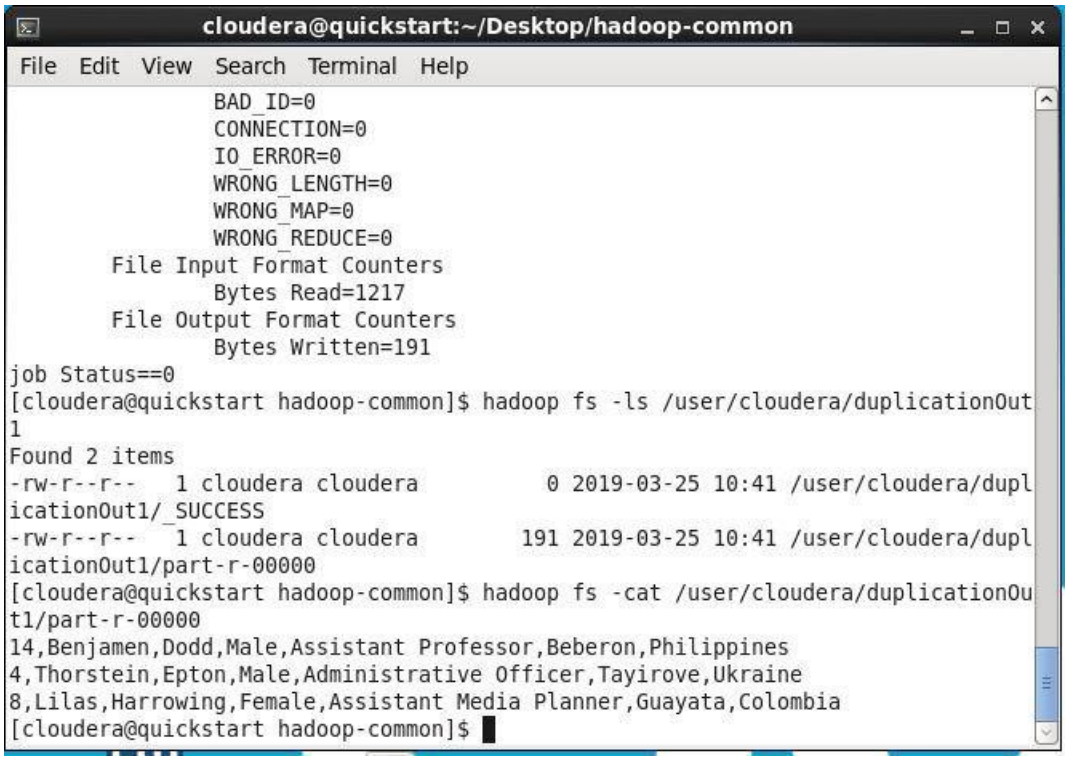

Player
Applications Places System
cloudera@quickstart:~/
File Edit View Search Terminal Help
19/04/03 22:24:18 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=1281
FILE: Number of bytes written=289469
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=1371
HDFS: Number of bytes written=191
HDFS: Number of read operations=6
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=11528
Total time spent by all reduces in occupied slots (ms)=12728
Total time spent by all map tasks (ms)=11528
Total time spent by all reduce tasks (ms)=12728
Total vcore-milliseconds taken by all map tasks=11528
Total vcore-milliseconds taken by all reduce tasks=12728
Total megabyte-milliseconds taken by all map tasks=11804672
Total megabyte-milliseconds taken by all reduce tasks=13033472
Map-Reduce Framework
Map input records=20
Map output records=19
Map output bytes=1237
Map output materialized bytes=1281
Input split bytes=154
Combine input records=0
Combine output records=0
Reduce input groups=16
Reduce shuffle bytes=1281
Reduce input records=19
Reduce output records=3
Spilled Records=38
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=321
CPU time spent (ms)=2660
Physical memory (bytes) snapshot=366940160
Virtual memory (bytes) snapshot=3015806976
Total committed heap usage (bytes)=226627584
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG LENGTH=0


Applications Places System
cloudera@quicksta
File Edit View Search Terminal Help
[cloudera@quickstart hadoop-common]$ hadoop jar Duplication.jar com.bdp.mapreduc
e.duplicaterecord.driver.DuplicateRecordDriver /user/cloudera/duplication/ /user
/cloudera/duplicationOut25
19/04/03 22:23:21 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
19/04/03 22:23:24 INFO input.FileInputFormat: Total input paths to process : 1
19/04/03 22:23:24 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
at java.lang.Object.wait(Native Method)
at java.lang.Thread.join(Thread.java:1281)
at java.lang.Thread.join(Thread.java:1355)
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DF
SOutputStream.java:967)
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutpu
tStream.java:705)
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStre
am.java:894)
19/04/03 22:23:24 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
at java.lang.Object.wait(Native Method)
at java.lang.Thread.join(Thread.java:1281)
at java.lang.Thread.join(Thread.java:1355)
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DF
SOutputStream.java:967)
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutpu
tStream.java:705)
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStre
am.java:894)
19/04/03 22:23:24 INFO mapreduce.JobSubmitter: number of splits:1
19/04/03 22:23:25 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
54354993325_0001
19/04/03 22:23:26 INFO impl.YarnClientImpl: Submitted application application_15
54354993325_0001
19/04/03 22:23:27 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1554354993325_0001/
19/04/03 22:23:27 INFO mapreduce.Job: Running job: job_1554354993325_0001
19/04/03 22:23:48 INFO mapreduce.Job: Job job_1554354993325_0001 running in uber
mode : false
19/04/03 22:23:48 INFO mapreduce.Job: map 0% reduce 0%
19/04/03 22:24:01 INFO mapreduce.Job: map 100% reduce 0%
19/04/03 22:24:17 INFO mapreduce.Job: map 100% reduce 100%
19/04/03 22:24:18 INFO mapreduce.Job: Job job_1554354993325_0001 completed succe
ssfully
19/04/03 22:24:18 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=1281
FILE: Number of bytes written=289469
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0

4. Run the program.

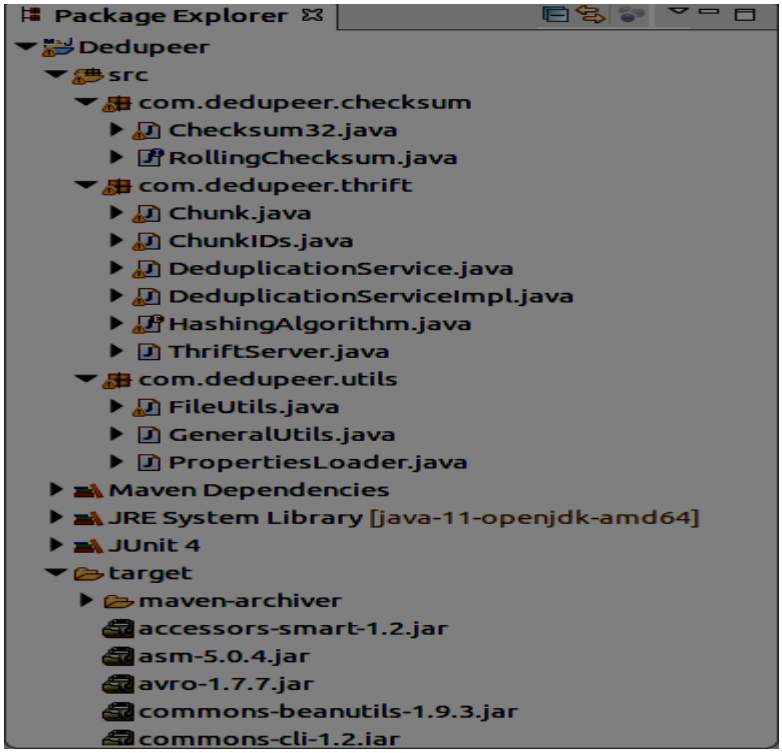


5. It will display duplicate records from file.

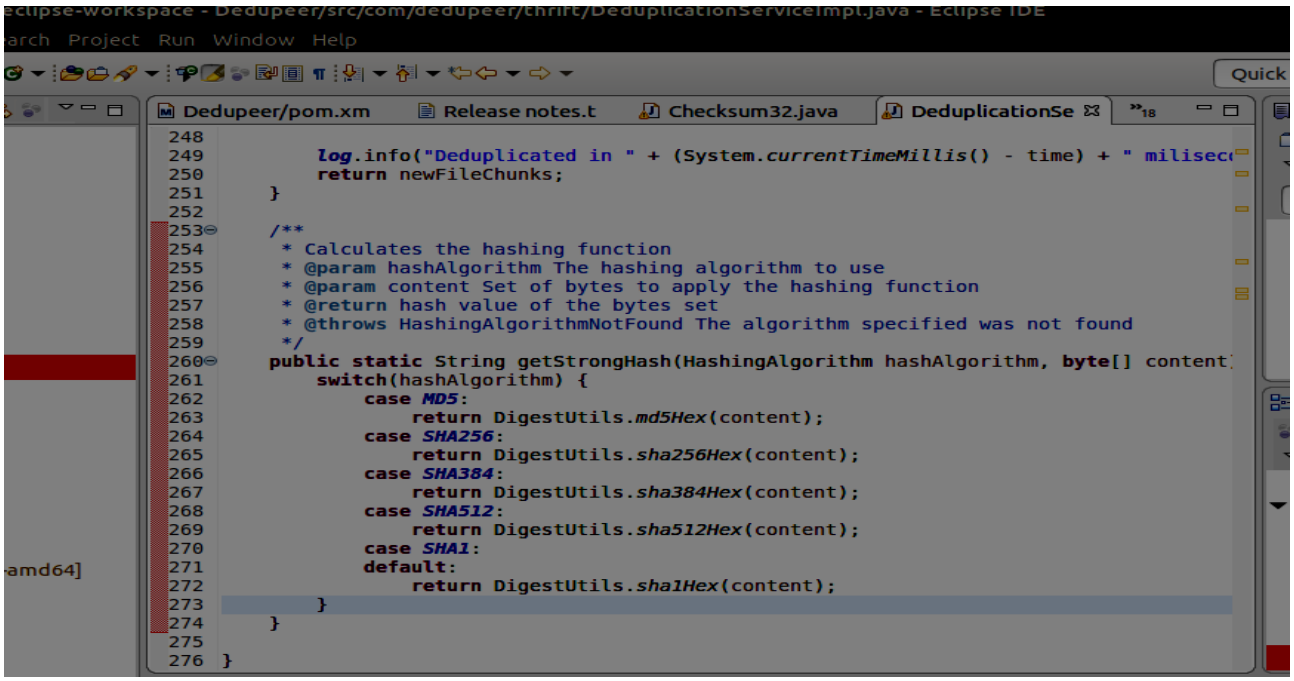


6. Now By using algorithms . To find which algorithm give better result

7.Create Maven project in eclipse



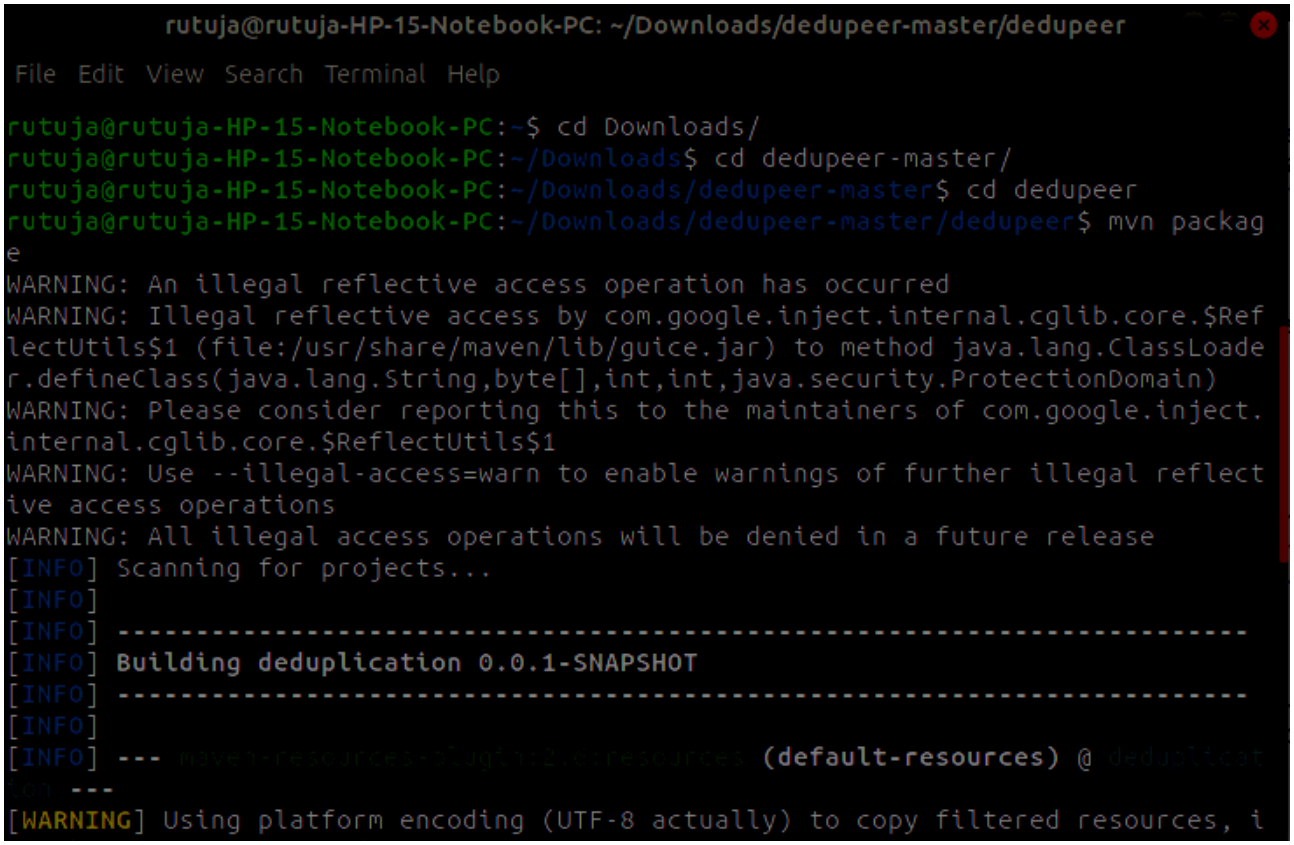
8.Code to find duplicate data using algorithm

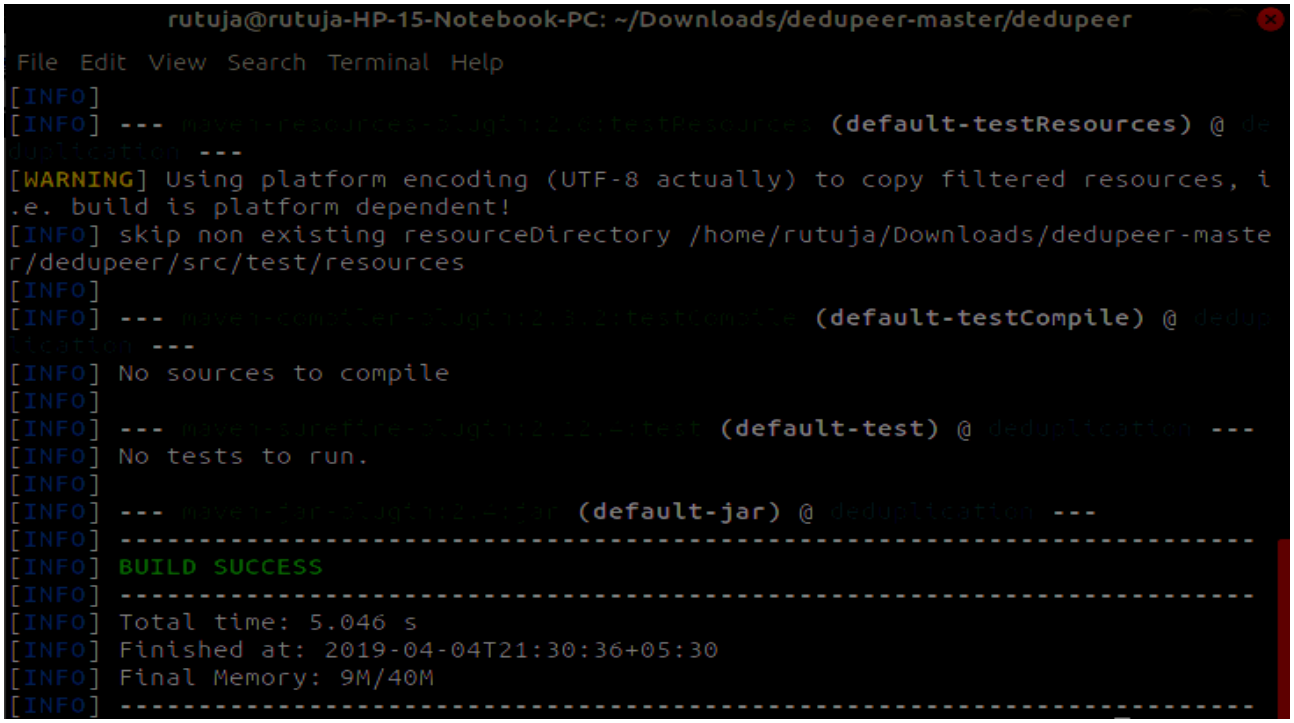


9.Generate unique hash value for each Chunk- File id divided into chunk. Chunk size is 1 megabyte. And each chuck assign a unique hash value by using is fixed chunking algorithm. Then output of chunk algorithm gives an input to the SHA algorithm. By using hash value it compare the data with datasets if duplication of data find the data will discard from file.

```
package com.dedupeer.thrift;


import java.util.Map;
import java.util.HashMap;
import org.apache.thrift.TEnum;

public enum HashingAlgorithm implements org.apache.thrift.TEnum {
  MD5(1),
  SHA1(2),
  SHA256(3),
  SHA384(4),
  SHA512(5);

  private final int value;

  private HashingAlgorithm(int value) {
    this.value = value;
  }

  public int getValue() {
    return value;
  }

  public static HashingAlgorithm findByValue(int value) {
    switch (value) {
      case 1:
        return MD5;
      case 2:
        return SHA1;
      case 3:
        return SHA256;
      case 4:
        return SHA384;
      case 5:
        return SHA512;
      default:
        return null;
    }
  }
```

10.Creating jar file for maven project. Jar file automatically create on target directory





11.Successfully created the jar file. Record store on hdfs .

```
hduser@rutuja-HP-15-Notebook-PC:/home/rutuja$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hduser in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
localhost: namenode is running as process 4523.  Stop it first.
Starting datanodes
localhost: datanode is running as process 4701.  Stop it first.
Starting secondary namenodes [rutuja-HP-15-Notebook-PC]
rutuja-HP-15-Notebook-PC: secondarynamenode is running as process 4946.  Stop it first.
2019-04-04 22:36:13,835 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
 applicable
Starting resourcemanager
Starting nodemanagers
hduser@rutuja-HP-15-Notebook-PC:/home/rutuja$ jps
8881 ResourceManager
4946 SecondaryNameNode
9061 NodeManager
9224 Jps
4523 NameNode
4701 DataNode
hduser@rutuja-HP-15-Notebook-PC:/home/rutuja$ hadoop fs -ls /record
2019-04-04 22:36:42,506 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... u
 applicable
Found 6 items
-rw-r--r--   1 hduser supergroup     453616 2019-03-25 13:38 /record/data.txt
-rw-r--r--   1 hduser supergroup       1217 2019-03-24 18:34 /record/duplicateRecordSampleData
-rw-r--r--   1 hduser supergroup       1217 2019-03-24 15:34 /record/duplicateRecordSampleData.txt
drwxr-xr-x   - hduser supergroup          0 2019-03-25 19:16 /record/employee
drwxr-xr-x   - hduser supergroup          0 2019-03-25 14:57 /record/out
-rw-r--r--   1 hduser supergroup       1195 2019-03-25 15:29 /record/sample.txt
hduser@rutuja-HP-15-Notebook-PC:/home/rutuja$
```

Conclusion:

Big data is the one of the emerging technology in today trends. Hadoop tool is used for running big data application. However, the arrival of Hadoop some difficult issues are considered. One of the issue is File duplication in the Hadoop System. File De-duplication is a useful technique for eliminating duplicate copies of file in the Hadoop system. By using map-reduce we succefully find the duplicate records from file

References:

[1] D. T. Meyer and W. J. Bolosky, “A study of practical deduplication,” ACM Trans. Storage, vol. 7, no. 4, pp. 1–20, 2012.

[2] A. Venish and K. S. Sankar, “Proceedings of the International Conference on Soft Computing Systems,” vol. 398, pp. 13–21, 2016.

[3] D. Kajaree and R. . Behera, “A Survey on Web Crawler Approaches,” Int. J. Innov. Res. Comput. Commun. Eng., vol. 5, no. 2, pp. 1302–1309, 2017.

[4] G. Shaikh and I. Technology, “A Survey on Deduplication Strategies and Storage Systems,” pp. 85–90, 2015.