

# Image Processing with Hadoop Framework

Presentation by

Priya Patel

225589

PPatel5@my.Harrisburgu.edu



# AGENDA OF PRESENTATION

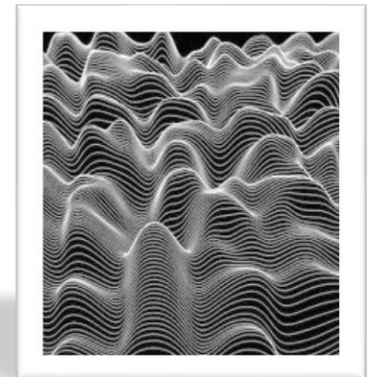
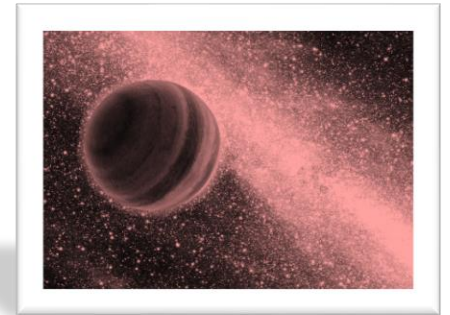
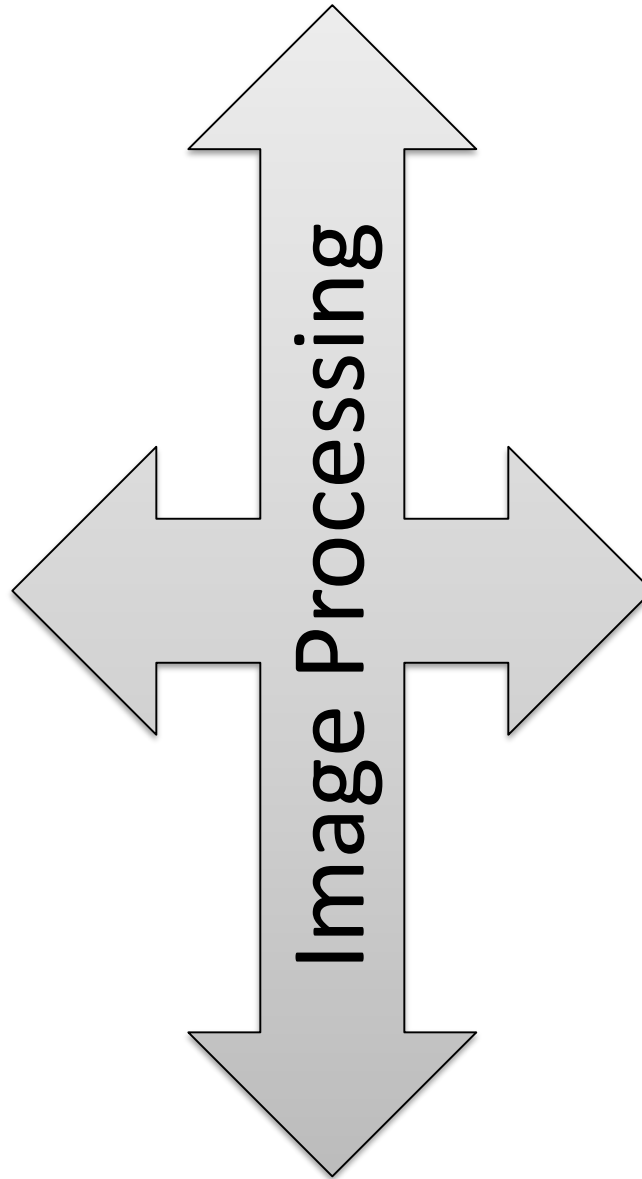
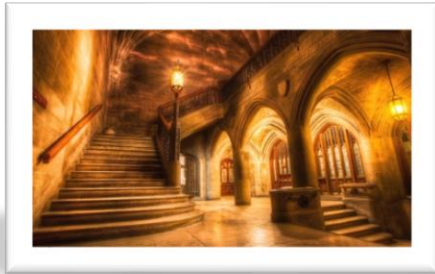
- ✓ What is Image Processing?
- ✓ How It works?
- ✓ Why we need to delete duplicate images?
- ✓ What is Hadoop? How it is related to Image Processing?
- ✓ What is Motivation behind?
- ✓ What are current issues?
- ✓ What has been done by others?
- ✓ Examples of related Technical Papers
- ✓ How it is related to Algorithm concepts?
- ✓ How we approach?
- ✓ Code Representation



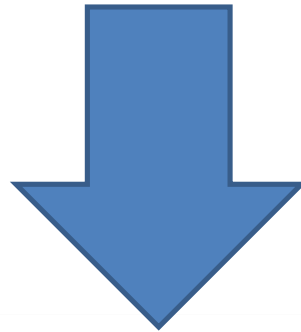
# WHAT IS IMAGE PROCESSING?

- ✓ In [imaging science](#), **image processing** is any form of [signal processing](#) for which the input is an image, such as a [photograph](#) or [video frame](#); the output of image processing may be either an image or a set of characteristics or [parameters](#) related to the image.[1]





# WHY WE NEED TO DELETE DUPLICATION?



# BIG DATA

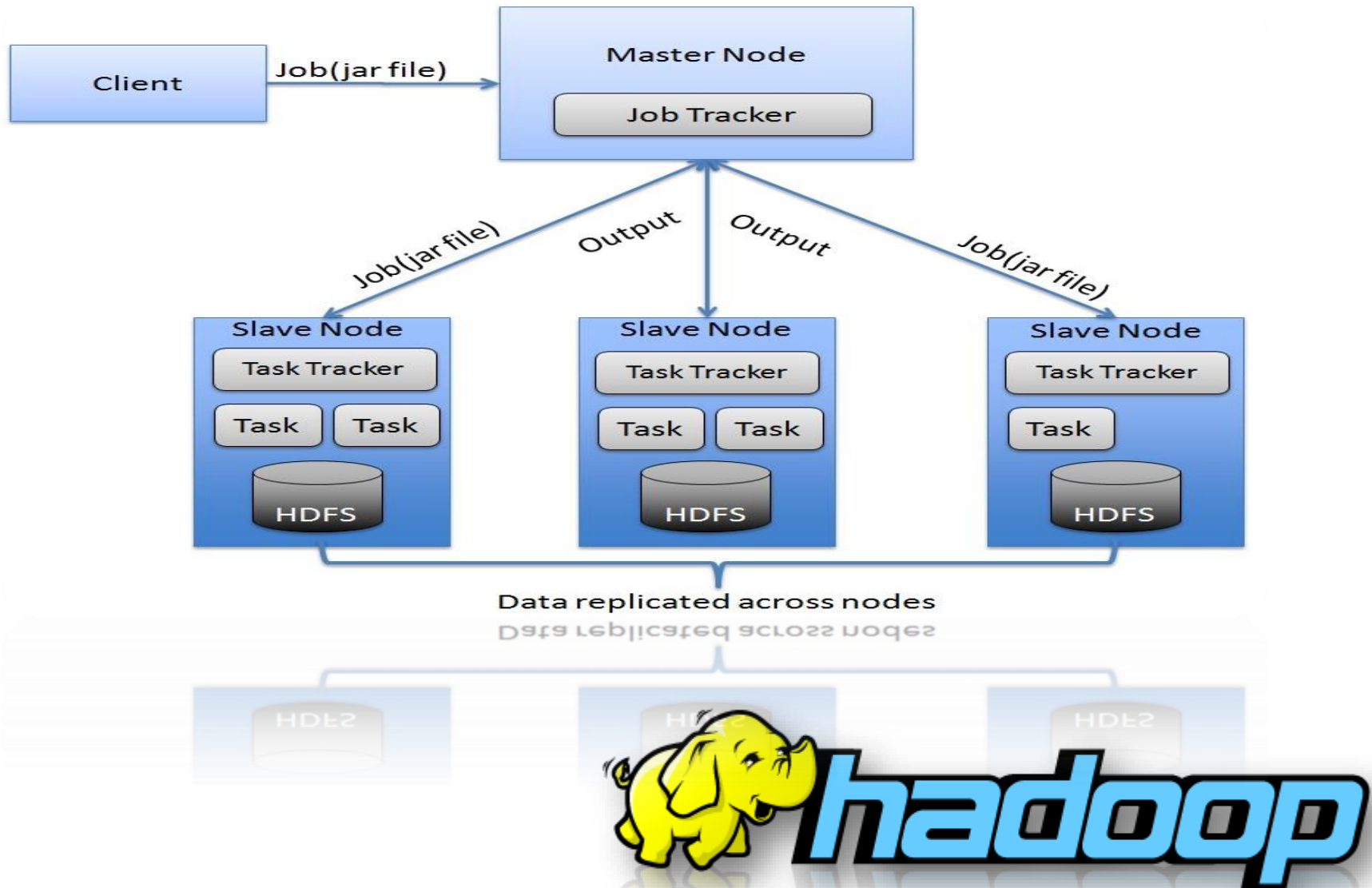
- ✓ **Big data** is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process them using traditional data processing applications.[2]

## Examples of Big Data:

- ✓ CCTV Footages
  - ✓ Social Media Files
  - ✓ Gmail Logs
  - ✓ Medical Data
  - ✓ Online Shopping Data
  - ✓ Airline Details
- 
- ✓ During 1990 – 64mb to 128 mb RAM size & Data Storage 1 GB to 20 GB
  - ✓ 2014 – 4 GB to 16 GB & Data Storage 1 TB
  - ✓ 90 % Data come within last 2 year.



# HADOOP ARCHITECTURE





# WHAT HAS BEEN DONE BY OTHERS?

- Multiple tools

Related changes

Upload file

Special pages

Permanent link

Page information

Wikidata item

Print/export

Create a book

Download as PDF

Printable version

Languages

فارسی

Türkçe

Edit links

► [neuroimaging software](#) (21 P)

Pages in category "Image processing software"

The following 62 pages are in this category, out of 62 total. This list may not reflect recent changes ([learn more](#)).

**3**

- [3DSlicer](#)

**A**

- [Amira \(software\)](#)
- [Analysis of Functional NeuroImages](#)
- [ANIMAL \(image processing\)](#)
- [AutoCollage 2008](#)
- [Avizo \(software\)](#)

**B**

- [Bitplane](#)
- [Bsoft](#)

**C**

- [CamFind](#)
- [CellCognition](#)
- [CellProfiler](#)
- [CoLocalizer Pro](#)
- [CONN \(functional connectivity toolbox\)](#)
- [CVPtools](#)

**E**

- [Endrov](#)

**F**

- [Fiji \(software\)](#)
- [FreeSurfer](#)

**G**

- [GemIdent](#)
- [Ginkgo CADx](#)
- [GNU Octave](#)

**H**

- [HDR PhotoStudio](#)
- [Huygens Software](#)

**I**

- [IDL \(programming language\)](#)
- [Ilastik](#)
- [Image Studio Lite](#)
- [Image SXM](#)
- [ImageApp](#)
- [ImageJ](#)
- [ImageNets](#)
- [Insight Segmentation and Registration Toolkit](#)
- [Integrated Software for Imagers and Spectrometers](#)
- [Integrating Vision Toolkit](#)
- [InVesalius](#)
- [ITK-SNAP](#)

**K**

- [Kaleidica](#)
- [KNIME](#)

**M**

- [Mango \(software\)](#)
- [Mathematica](#)
- [MATLAB](#)
- [Medical imaging](#)
- [MeVisLab](#)
- [MicroDicom](#)
- [Microscope image processing](#)
- [MountainsMap](#)

**N**

**O**

- [Netpbm](#)
- [Nrrd](#)

**P**

- [OpenCV](#)
- [Openlab](#)

**S**

- [PhotoModeler](#)
- [PurVIEW](#)

**T**

- [TomoPy](#)

**V**

- [Vaa3D](#)
- [VIGRA](#)
- [VIPS \(software\)](#)
- [VisualAp](#)
- [VXL](#)

**W**

- [Warpalizer](#)

**Z**

- [Zeroth \(software\)](#)



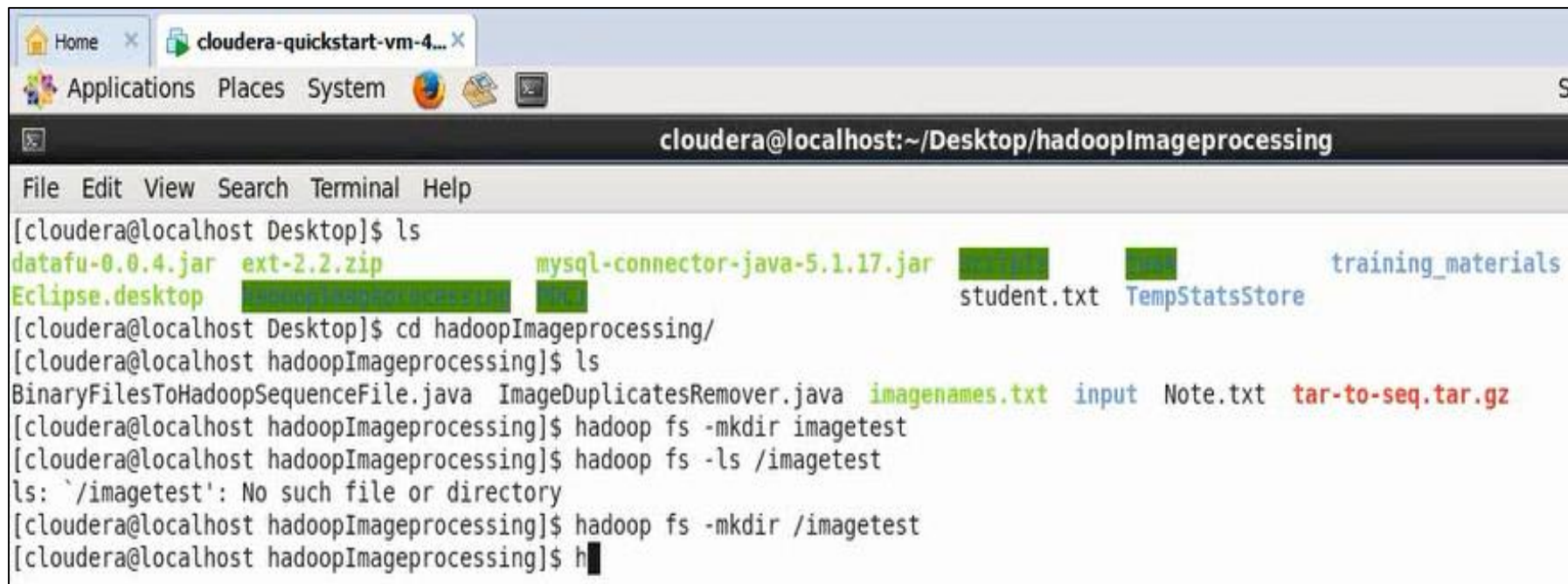
# HOW WE APPROACH ?

- 1) Image file as a Input (Store in local System).



```
*imagenames.txt (~/Desktop/hadoopImagepro)
File Edit View Search Tools Documents Help
Open Save Undo
*imagenames.txt
/imagetest/input/abc.jpg
/imagetest/input/Chrysanthemum.jpg
/imagetest/input/def.jpg
/imagetest/input/Desert.jpg
/imagetest/input/Hydrangeas.jpg
/imagetest/input/Jellyfish.jpg
/imagetest/input/Koala.jpg
/imagetest/input/Lighthouse.jpg
/imagetest/input/Penguins.jpg
/imagetest/input/Tulips.jpg
/imagetest/input/xyz.jpg
```

## 2) Import Image files from Local System to HDFS for Processing



```
cloudera@localhost:~/Desktop/hadoopImageprocessing
File Edit View Search Terminal Help
[cloudera@localhost Desktop]$ ls
datafu-0.0.4.jar  ext-2.2.zip  mysql-connector-java-5.1.17.jar  student.txt  training_materials
Eclipse.desktop  hadoopImageprocessing  Note.txt  TempStatsStore
[cloudera@localhost Desktop]$ cd hadoopImageprocessing/
[cloudera@localhost hadoopImageprocessing]$ ls
BinaryFilesToHadoopSequenceFile.java  ImageDuplicatesRemover.java  imagenames.txt  input  Note.txt  tar-to-seq.tar.gz
[cloudera@localhost hadoopImageprocessing]$ hadoop fs -mkdir imagetest
[cloudera@localhost hadoopImageprocessing]$ hadoop fs -ls /imagetest
ls: `/imagetest': No such file or directory
[cloudera@localhost hadoopImageprocessing]$ hadoop fs -mkdir /imagetest
[cloudera@localhost hadoopImageprocessing]$ h
```

```
/imagetest/input/Chrysanthemum.jpg
/imagetest/input/Desert.jpg
/imagetest/input/Hydrangeas.jpg
/imagetest/input/Jellyfish.jpg
/imagetest/input/Koala.jpg
/imagetest/input/Lighthouse.jpg
/imagetest/input/Penguins.jpg
/imagetest/input/Tulips.jpg
/imagetest/input/abc.jpg
/imagetest/input/def.jpg
/imagetest/input/xyz.jpg
```



### 3) Covert Image File into Sequence File

```
cloudera@localhost:~/Desktop
File Edit View Search Terminal Help

0I[Mr00@zj00D6[0:00L-000b*Wj0j0[0;MH?;0t072[R0t0k0[000a{0^0}C[000Zs[0v00[0S?00*P0;]0B[VM00[00000H0[0t][0qv00t0[00?0WR09[0c[0S[00000UM0I0z2]>00[9000I0
0[01700[0x00[0e[0040[0j[0[00X00?0000[0>0x[00000[0m000V00000[0E[0[000[0X20P0x#Z00[0s[0[00"0:[0].?9+0u-0z[000[0X[0Bc00<t[00000C00[0[0[00q0v0P000K[0J0
[0qJ00000000++0
pN00

1 0030)000[0MUT0K000[0i[0
z00
0^j00z
0z0l0dq0u!000s0zY0[0000
000ZMJ[0[00000#|0"ePR0[00RC&G0[00C00.Wi0F000000K30I0]200K-0[0?{
0[0YJ[000#00Z0.0m00[000[000[00gk0{[0h]00Qjw0200b0,0/0[0n?00q[0[0i*0[0d00t000[0
00~0000000F00@0s0000000j0A0D00MY[0X*0[000[00G00[0c*0[000
00pY<Rx0u000<l0u0[0[00D0[0[00p0[0z8f00000m000[0Y[0[0}00v[00f][0[0[00000HX!V*0[0
000N0(Uu[0[0+0Z050h:00[0K[00000mk0[0N0?000000Q0jji%0[0E020xTG[0000010
0[000[0
B0'00000i00[0Z00060[00E000l100$|G00>}D0000q000V00[00[0u000
B0[0SP[0X'0000s-00[006[0d0472)C0-00:0CS0>000:0000m000[00[0[0n0000[0j0r0[00[0000,000[00[00[00x0m0u0RQ0T
(0:H00A0iz[00([00Y00[0000000[0q[0[0M*0|F%00I=[00[0RST0[0Uil,
[00H004s0v00[0d[0000-0-[00N]M000#0uS0[0.M0k0"0'[0[0[0R[0S00z&0+d0vN0LHz00X0[00000j,o00[0j29[00[0S0
0[000000s0[000y0%0000[000t#{000f&0[0[0[0Y[0Y[0000[0[010R00[00E00h0,R[0040V00IV[00C0%N0[S0Su0B0L00v[00[00000P[0Z0}0[0c0r00[0T*0P40[0F000[0k[0Z[00XD000V[0000p00000VG0-0
sNi0K"Z00-hB0w700000'g'00P0[0[0?00000t[0[0[0000030D000000j~/000v0w$<[04vxX[0[0]?[0
4#[000000000[0Y0Ki0000[0[0"00v0bK[0[0[00gk0Z[000_0S70u0/s00}0[0
h0Uc0[0000t0P30X0ivg000[0705[0[000H2[0000+F0;I%0[00=
~000w0p0v
0000000000[00000.0000070H0[0[0[00+e0000:[0%20
e0jz[0[0G00LN/TZ"00SR>00U0)f0u2000[0m0[0{w00@000|00l00000^&=100Fj0000\T0~/00[00[0wbdce0'0012H-ie[0000_0i[00EPq00!0>B00Q)o00Fc#[00#0K6(0k0[0S00@0G0g0[0I0[00[000M0R0000[0
00f100[00[0000m00;000700[0000L000az[9J0*00F0
0[00k0U00[0
[0]>00>[00]0=cS0,0g0?000y:0Z0[09zn0[00[0000aN0t00I0*0[0J0[00[0a`?00[0hN[000*00C
0JNCF00H#[00[0[0=000N00P030)0i0cx000[0wGP0[0w0r00I000*ph00a_0[0]/0[0i000g-100J[0P[000L
j800C0[000b00[00"00000[0?K00=0w(0>0d00c0[00000v00
(0<[0[0[0[04[00c\i0700I0.Q00[010U|00[0I0~}0
q+[0QD0>_0g0h00c[0n00|&00
R0KUM $Ty000000E
0/50000[00E0:0[077n[00[000[00Gm0)V0$)2[0'r
5H0u[0000[00T0[03[000000<[0~00k000-000000v[0S00000[000G+000[0T40[000[0000000[00I000000MeT3[0000T0[0T0[0]0[000
```

# CODE : IMAGE FILE TO SEQUENCE FILE

## Main Method:

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: BinaryFilesToHadoopSequenceFile <in Path for url file> <out pat for sequence file>");
        System.exit(2);
    }

    Job job = new Job(conf, "BinaryFilesToHadoopSequenceFile"); //Job Object
    job.setJarByClass(BinaryFilesToHadoopSequenceFile.class); // Driver Class
    job.setMapperClass(BinaryFilesToHadoopSequenceFileMapper.class); //Mapper Class
    job.setOutputKeyClass(Text.class); //K2 File Name
    job.setOutputValueClass(BytesWritable.class); //V2 Content of File
    job.setInputFormatClass(TextInputFormat.class); //Input files in Text Format
    job.setOutputFormatClass(SequenceFileOutputFormat.class); // Output is a Sequence File format

    FileInputFormat.addInputPath(job, new Path(otherArgs[0])); //Input Path passing through Command Line
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1])); // Output path passing through Command Line
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```



# Mapper Code:

```
// Mapper Method
public static class BinaryFilesToHadoopSequenceFileMapper extends Mapper<Object, Text, Text, BytesWritable> {
    //Over ride Map Method K1 V1 value
    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {

        logger.info("map method called..");

        String uri = value.toString(); //Reading a file
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(URI.create(uri), conf); //Loading a file from HDFS
        FSDataInputStream in = null;
        try {
            in = fs.open(new Path(uri)); //Reading a Image File Content
            java.io.ByteArrayOutputStream bout = new ByteArrayOutputStream();
            byte buffer[] = new byte[1024 * 1024]; // Writing into ByteArray

            while( in.read(buffer, 0, buffer.length) >= 0 ) {
                bout.write(buffer);
            }
            context.write(value, new BytesWritable(bout.toByteArray())); //Writing a V2 which is in Sequence File Format
        } finally {
            IOUtils.closeStream(in);
        }
    }
}
```

# Duplicate Image Removal

```
/imagedtest/input/Chrysanthemum.jpg ████████  
/imagedtest/input/Desert.jpg  
/imagedtest/input/Hydrangeas.jpg ████████  
/imagedtest/input/Jellyfish.jpg  
/imagedtest/input/Koala.jpg ████████  
/imagedtest/input/Lighthouse.jpg  
/imagedtest/input/Penguins.jpg  
/imagedtest/input/Tulips.jpg  
/imagedtest/input/abc.jpg  
/imagedtest/input/def.jpg ████████  
/imagedtest/input/xyz.jpg
```



# Duplicate Image Removal

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

    //This is the line that makes the hadoop run locally
    //conf.set("mapred.job.tracker", "local");

    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "image dups remover");
    job.setJarByClass(ImageDuplicatesRemover.class);
    job.setInputFormatClass(SequenceFileInputFormat.class);
    job.setMapperClass(ImageMd5Mapper.class);
    job.setReducerClass(ImageDupsReducer.class);
    //job.setNumReduceTasks(2);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

---



# Mapper Code:

```
public static class ImageMd5Mapper extends Mapper<Text, BytesWritable, Text, Text>{

    public void map(Text key, BytesWritable value, Context context) throws IOException, InterruptedException {
        //get the md5 for this specific file
        String md5Str;
        try {
            md5Str = calculateMd5(value.getBytes());
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            context.setStatus("Internal error - can't find the algorithm for calculating the md5");
            return;
        }
        Text md5Text = new Text(md5Str);

        //put the file in the map where the md5 is the key, so duplicates will
        // be grouped together for the reduce function
        context.write(md5Text, key);
    }

    static String calculateMd5(byte[] imageData) throws NoSuchAlgorithmException {
        //get the md5 for this specific data
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(imageData);
        byte[] hash = md.digest();

        // Below code of converting Byte Array to hex
        String hexString = new String();
        for (int i=0; i < hash.length; i++) {
            hexString += Integer.toString( ( hash[i] & 0xff ) + 0x100, 16).substring( 1 );
        }
        return hexString;
    }
}
```

## Reducer Code

```
public static class ImageDupsReducer extends Reducer<Text,Text,Text,Text> {  
  
    public void reduce(Text key, Iterable<Text> values, Context context)  
        throws IOException, InterruptedException {  
        //Key here is the md5 hash while the values are all the image files that  
        // are associated with it. for each md5 value we need to take only  
        // one file (the first)  
        Text imageFilePath = null;  
        for (Text filePath : values) {  
            imageFilePath = filePath;  
            break;//only the first one  
        }  
        // In the result file the key will be again the image file path.  
        context.write(imageFilePath, key);  
    }  
}
```



# Final Output

```
14/12/28 17:26:05 INFO mapred.JobClient: HDFS: Number of bytes read=120975714
14/12/28 17:26:05 INFO mapred.JobClient: HDFS: Number of bytes written=565
14/12/28 17:26:05 INFO mapred.JobClient: HDFS: Number of read operations=3
14/12/28 17:26:05 INFO mapred.JobClient: HDFS: Number of large read operations=0
14/12/28 17:26:05 INFO mapred.JobClient: HDFS: Number of write operations=1
14/12/28 17:26:05 INFO mapred.JobClient: Job Counters
14/12/28 17:26:05 INFO mapred.JobClient:   Launched map tasks=1
14/12/28 17:26:05 INFO mapred.JobClient:   Launched reduce tasks=1
14/12/28 17:26:05 INFO mapred.JobClient:   Data-local map tasks=1
14/12/28 17:26:05 INFO mapred.JobClient:   Total time spent by all maps in occupied slots (ms)=19443
14/12/28 17:26:05 INFO mapred.JobClient:   Total time spent by all reduces in occupied slots (ms)=7525
14/12/28 17:26:05 INFO mapred.JobClient:   Total time spent by all maps waiting after reserving slots (ms)=0
14/12/28 17:26:05 INFO mapred.JobClient:   Total time spent by all reduces waiting after reserving slots (ms)=0
14/12/28 17:26:05 INFO mapred.JobClient: Map-Reduce Framework
14/12/28 17:26:05 INFO mapred.JobClient:   Map input records=11
14/12/28 17:26:05 INFO mapred.JobClient:   Map output records=11
14/12/28 17:26:05 INFO mapred.JobClient:   Map output bytes=681
14/12/28 17:26:05 INFO mapred.JobClient:   Input split bytes=129
14/12/28 17:26:05 INFO mapred.JobClient:   Combine input records=0
14/12/28 17:26:05 INFO mapred.JobClient:   Combine output records=0
14/12/28 17:26:05 INFO mapred.JobClient:   Reduce input groups=9
14/12/28 17:26:05 INFO mapred.JobClient:   Reduce shuffle bytes=534
14/12/28 17:26:05 INFO mapred.JobClient:   Reduce input records=11
14/12/28 17:26:05 INFO mapred.JobClient:   Reduce output records=9
14/12/28 17:26:05 INFO mapred.JobClient:   Spilled Records=22
14/12/28 17:26:05 INFO mapred.JobClient:   CPU time spent (ms)=2860
14/12/28 17:26:05 INFO mapred.JobClient:   Physical memory (bytes) snapshot=302854144
14/12/28 17:26:05 INFO mapred.JobClient:   Virtual memory (bytes) snapshot=1337724928
14/12/28 17:26:05 INFO mapred.JobClient:   Total committed heap usage (bytes)=202649600
[cloudera@localhost Desktop]$ hadoop fs -ls /imagetest/uniqueimages
```

# REFERENCES

1. <http://www.disi.unige.it/person/RovettaS/rad/image-processing-wikipedia-book.pdf>
2. [Data, data everywhere"](#). *The Economist*. 25 February 2010. Retrieved 9 December 2012.
3. <https://hadoop.apache.org/>
4. ["Hadoop Releases"](#). *apache.org*. Apache Software Foundation. Retrieved 2014-12-06.
5. [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB8QFjAA&url=http%3A%2F%2Ffunivagora.ro%2Fjour%2Findex.php%2Fijccc%2Farticle%2Fdownload%2F285%2Fpdf\\_142&ei=dRFJVdXJCs2eyATUs4HYBg&usg=AFQjCNHjQs0b8e6p9qa6eqXHXOTBovcnRA](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB8QFjAA&url=http%3A%2F%2Ffunivagora.ro%2Fjour%2Findex.php%2Fijccc%2Farticle%2Fdownload%2F285%2Fpdf_142&ei=dRFJVdXJCs2eyATUs4HYBg&usg=AFQjCNHjQs0b8e6p9qa6eqXHXOTBovcnRA)
6. <http://ijaeit.com/Paper-Pdf/Paper%208.pdf>
7. [http://ssg.astro.washington.edu/downloadable/PASP2011\\_AstronomyInTheCloud.pdf](http://ssg.astro.washington.edu/downloadable/PASP2011_AstronomyInTheCloud.pdf)

# Thank You



# Final Report

## Image Processing on Hadoop Platform

CISC 525-50- B-2019/Spring - Big Data Architectures

Priya Patel

225589

[PPatel5@my.harrisburgu.edu](mailto:PPatel5@my.harrisburgu.edu)

Mahek Patel

224609

[MPatel6@my.harrisburgu.edu](mailto:MPatel6@my.harrisburgu.edu)

### Introduction

Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of commodity computers using the simple programming model. Hadoop represents a Java-based distributed computing framework that is designed to support applications that are implemented via the MapReduce programming model. It contains two core components which are HDFS (Hadoop Distributed File System) and MapReduce. MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

In this group project, we are going to perform Image processing on Hadoop Cluster. Let's take one real time scenario to elaborate the definition of this Project. On Social Media web Sites like Facebook, Twitter etc. have a number of users who share and upload pictures and videos. Sometime Image file with same name, same format, same resolution and same size is shared by multiple users all these Images are stored in the Data Base even though it carries a same properties. This doesn't look good as duplicate files occupied needless space in File System.

### What are the problems of existing approaches?

Social media, today's biggest influence and usage. People used to upload images, post statues, upload videos and many more. Where are all these files being saved? The biggest problem or we can say crucial issue is to manage all these files and retrieve them when needed. There are hundreds of social media and millions of user accounts on it. Overcome this issue has given very fresh and jejune technologies to the computer world. One of them is Image Processing.

There are lots of duplicate images available on Internet and they are none of use. Remove them effectively and carefully can give us huge vacant space and speed in surfing.

The Photos application is one of Facebook's most popular features. Up to date, users have uploaded over 15 billion photos which makes Facebook the biggest photo sharing website. For each uploaded photo, Facebook generates and stores four images of different sizes, which translates to a total of 60 billion images and 1.5PB of storage. The current growth rate is 220 million new photos per week, which translates to 25TB of additional storage consumed weekly. At the peak there are 550,000 images served per second. These

numbers pose a significant challenge for the Facebook photo storage infrastructure.

### **Solution for this Problem**

To overcome the problem of storing a petabyte of data, one solution is removing the duplicate records/file/images/videos.

To remove duplicated records from a dataset, the main consideration is how to decide that two records are duplicate? We need to compare records to determine their degree of similarity, which implies that corresponding fields in the records has to be compared. The comparison of fields to determine whether or not two syntactic values are alternative representations of the same semantic entity.

In this proposed approach, we used different algorithms to convert image Binary files into Hadoop Sequence files and compare this Sequence file and rid of duplicate files i.e. use the Data Cleaning Concept.

Data cleaning is a technique for detecting missing and incorrect values and correcting them, as well as matching duplicate records in an integrated data warehouse or database table. It focuses on eliminating variations in data contents and reducing data redundancy, and is aimed at improving the overall data quality and consistency.

### **Related Work**

Nowadays, data cleaning is becoming so popular in research area and thus several researches have been done on that topic. Among them, the most common form of data cleaning is duplicate detection and elimination. A.E Monge and C.PElkan

proposed the priority queue algorithm for detecting duplicates in [1]. Mauricio A. Hernandez and Salvato J. Stolf proposed Sorted Neighbourhood Method (SNM) for duplicate detection in [10] and many researchers are still discovering the various duplicate detection algorithms. In doing duplicate detection, data pre-processing must be performed firstly. Data pre-processing stage refers to the processes that is prior to duplicate detection process. The goal of this process is to scrub the data into more consistent state to get more standardized data for the merge/purge process to achieve better results [9]. After the data pre-processing, duplicate detection and elimination process can be performed. The standard method for detecting exact duplicates in a database is to sort the database and then check if the neighbouring records are identical [10]. The most reliable way to detect approximate duplicates is to compare every record with every other recording the database

To manage a large sets of Images and video is become the biggest problem for most of Social Media Websites. There are different tools available for Process the large sets of Image and video files. They are using lots of algorithm for searching and filtering to find out similar image files from database and remove them to generate ample space to store petabytes of Image data.

### **Example of Current Systems**

There are multiple softwares and applications online available. Image Comparer, Digital Image detector, Image Processor etc.



According to our Project, let's take a real example. Facebook, the biggest social media website nowadays, stores users' posts, photos, and albums on it. As per the procedure, it uses to be stored on cloud and also on back-end server. So, a single photo on an average occupies 2 MB, and every day we need a huge free space to store files.

To overcome this issue, Facebook has a feature of removing duplicate images. Let's take, you and your friend uploaded the same image which has the same name, resolution, colors, effects, everything. So, there is no necessity of storing both files. We can simply store one and give reference to others. Facebook uses the same concepts and it removes duplicate ones.

To remove duplicated records from a dataset, the main consideration is how to decide that two records are duplicate? We need to compare records to determine their degree of similarity, which implies that corresponding fields in the records have to be compared. The comparison of fields to determine whether or not two syntactic values are alternative representations of the same semantic entity.

### **Work Completion**

In the final phase of this project, we build a Hadoop Cluster along with HDFS storage system to store and process image files using Java code.

After having a Hadoop Framework up and running, we use the ecosystem called Map Reduce for importing image files from Local File System to Hadoop File System.

As Hadoop Framework can process image file as Sequence File, we build a Map Reduce job that converts image binary file into Hadoop Sequence File format. All image

files are available in Hadoop File System and ready to process. Step 1: we took a certain number of input images and stored it into Hadoop File System. Here we used Java Map reduce code.

Step 2: All the images which we took, we converted those into sequence file.

Step 3: Comparison of sequence file with one another.

Step 4: Identified the duplicate images.

Step 5: Removed the duplicate images from Hadoop File System.

## Implementation code

### Code 1:

#### Image File to Hadoop Sequence file conversion

##### Language: Java

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.URI;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem; import
org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.IOUtils;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.log4j.Logger;

public class BinaryFilesToHadoopSequenceFile {

    private static Logger logger = Logger.getLogger(BinaryFilesToHadoopSequenceFile.class);

    public static class BinaryFilesToHadoopSequenceFileMapper extends Mapper<Object,
Text, Text, BytesWritable> {

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

            logger.info("map method called..");

            String uri = value.toString();
            Configuration conf = new Configuration();
```

```

FileSystem fs = FileSystem.get(URI.create(uri), conf);
FSDataInputStream in = null; try {

    in = fs.open(new Path(uri));
    java.io.ByteArrayOutputStream bout = new ByteArrayOutputStream();
    byte buffer[] = new byte[1024 * 1024];

    while( in.read(buffer, 0, buffer.length) >= 0 ) {
        bout.write(buffer);
    }
    context.write(value, new
BytesWritable(bout.toByteArray())); } finally {
    IOUtils.closeStream(in);
}
}
}

```

```

public static void main(String[] args) throws Exception
{ Configuration conf = new Configuration();
String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs(); if (otherArgs.length != 2) {
    System.err.println("Usage: BinaryFilesToHadoopSequenceFile <in Path for url
file> <out pat for sequence file>");
    System.exit(2);
}
}

```

```

Job job = new Job(conf, "BinaryFilesToHadoopSequenceFile");
job.setJarByClass(BinaryFilesToHadoopSequenceFile.class);
job.setMapperClass(BinaryFilesToHadoopSequenceFileMapper.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(BytesWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);

FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
} }

```

## **Code 2:**

### **Duplicate File Removal**

#### **Language: Java**

```
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.BytesWritable;
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.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.util.GenericOptionsParser;

public class ImageDuplicatesRemover {

    public static class ImageMd5Mapper extends Mapper<Text,
BytesWritable, Text, Text>{

        public void map(Text key, BytesWritable value, Context context)
throws IOException,InterruptedException {
            //get the md5 for this specific file
            String md5Str;
            try {
                md5Str = calculateMd5(value.getBytes());
            } catch (NoSuchAlgorithmException e)
            { e.printStackTrace();
                context.setStatus("Internal error - can't find the algorithm for
calculating the md5");
```

```

        return;
    }
    Text md5Text = new Text(md5Str);

    //put the file in the map where the md5 is the key, so duplicates
will
    // be grouped together for the reduce
    function context.write(md5Text, key);
}

```

```

    static String calculateMd5(byte[] imageData) throws
NoSuchAlgorithmException {
        //get the md5 for this specific data
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(imageData);
        byte[] hash = md.digest();

        // Below code of converting Byte Array to
        hex String hexString = new String();
        for (int i=0; i < hash.length; i++) {
            hexString += Integer.toString( ( hash[i] & 0xff ) + 0x100,
16).substring( 1 );
        }
        return hexString;
    }
}

```

```

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

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

```

```

that
        //Key here is the md5 hash while the values are all the image files
        // are associated with it. for each md5 value we need to take only
        // one file (the first)
        Text imageFilePath = null;
        for (Text filePath : values) {
            imageFilePath = filePath;
            break;//only the first one
        }
        // In the result file the key will be again the image file
        path. context.write(imageFilePath, key);
    }
}

```

```

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

    //This is the line that makes the hadoop run locally
    //conf.set("mapred.job.tracker", "local");

    String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "image dups remover");
    job.setJarByClass(ImageDuplicatesRemover.class);
    job.setInputFormatClass(SequenceFileInputFormat.class);
    job.setMapperClass(ImageMd5Mapper.class);
    job.setReducerClass(ImageDupsReducer.class);
    //job.setNumReduceTasks(2);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
}

```

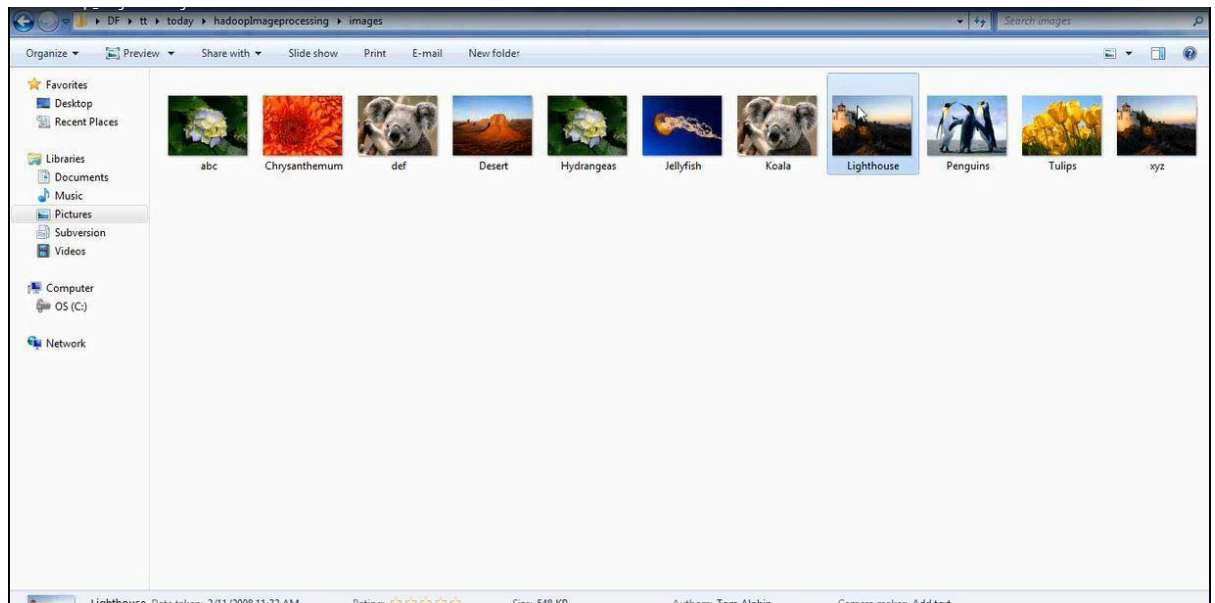
```

        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

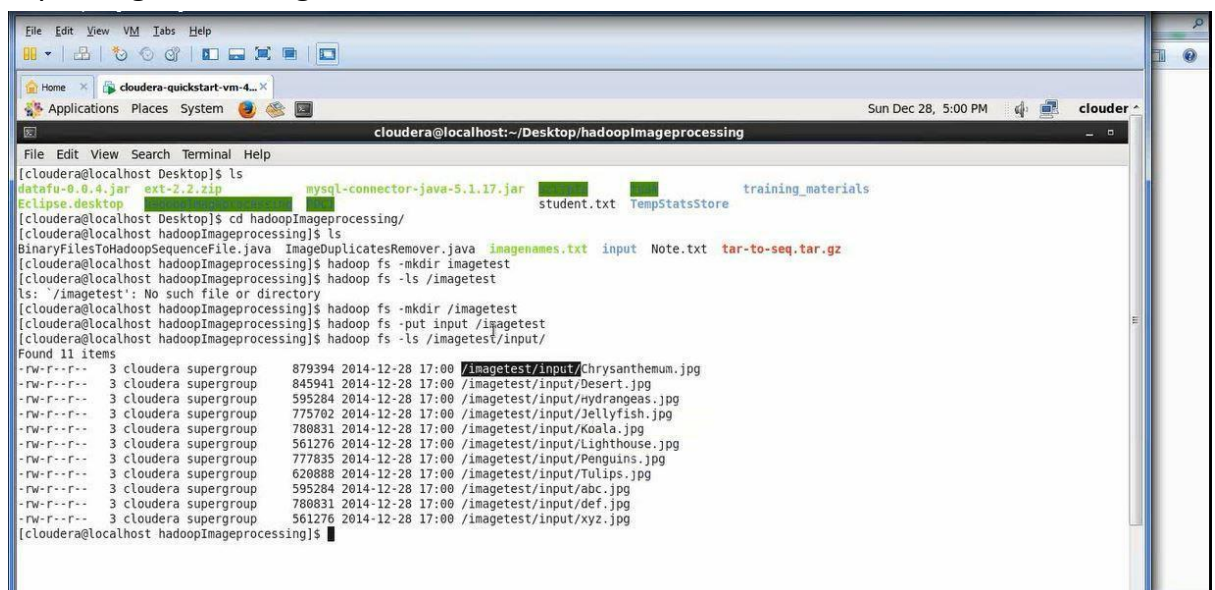
```

## Screenshot

### 1) Input of Images Files:



### 2) Importing Files using HDFS





### 3) Processing Imaging

```

File Edit View Search Terminal Help
14/12/28 17:17:49 INFO mapred.JobClient: map 100% reduce 100%
14/12/28 17:17:55 INFO mapred.JobClient: Job complete: job_201412101953_0033
14/12/28 17:17:55 INFO mapred.JobClient: Counters: 32
14/12/28 17:17:55 INFO mapred.JobClient: File System Counters
14/12/28 17:17:55 INFO mapred.JobClient: FILE: Number of bytes read=27446218
14/12/28 17:17:55 INFO mapred.JobClient: FILE: Number of bytes written=41498780
14/12/28 17:17:55 INFO mapred.JobClient: FILE: Number of read operations=0
14/12/28 17:17:55 INFO mapred.JobClient: FILE: Number of large read operations=0
14/12/28 17:17:55 INFO mapred.JobClient: FILE: Number of write operations=0
14/12/28 17:17:55 INFO mapred.JobClient: HDFS: Number of bytes read=7775000
14/12/28 17:17:55 INFO mapred.JobClient: HDFS: Number of bytes written=128975585
14/12/28 17:17:55 INFO mapred.JobClient: HDFS: Number of read operations=13
14/12/28 17:17:55 INFO mapred.JobClient: HDFS: Number of large read operations=0
14/12/28 17:17:55 INFO mapred.JobClient: HDFS: Number of write operations=1
14/12/28 17:17:55 INFO mapred.JobClient: Job Counters
14/12/28 17:17:55 INFO mapred.JobClient: Launched map tasks=1
14/12/28 17:17:55 INFO mapred.JobClient: Launched reduce tasks=1
14/12/28 17:17:55 INFO mapred.JobClient: Data-local map tasks=1
14/12/28 17:17:55 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=22160
14/12/28 17:17:55 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=13416
14/12/28 17:17:55 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
14/12/28 17:17:55 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
14/12/28 17:17:55 INFO mapred.JobClient: Map-Reduce Framework
14/12/28 17:17:55 INFO mapred.JobClient: Map input records=11
14/12/28 17:17:55 INFO mapred.JobClient: Map output records=11
14/12/28 17:17:55 INFO mapred.JobClient: Map output bytes=128975210
14/12/28 17:17:55 INFO mapred.JobClient: Input split bytes=130
14/12/28 17:17:55 INFO mapred.JobClient: Combine input records=0
14/12/28 17:17:55 INFO mapred.JobClient: Combine output records=0
14/12/28 17:17:55 INFO mapred.JobClient: Reduce input groups=11
14/12/28 17:17:55 INFO mapred.JobClient: Reduce shuffle bytes=13723093
14/12/28 17:17:55 INFO mapred.JobClient: Reduce input records=11
14/12/28 17:17:55 INFO mapred.JobClient: Reduce output records=11
14/12/28 17:17:55 INFO mapred.JobClient: Spilled Records=33
14/12/28 17:17:55 INFO mapred.JobClient: CPU time spent (ms)=7650
14/12/28 17:17:55 INFO mapred.JobClient: Physical memory (bytes) snapshot=446083072
14/12/28 17:17:55 INFO mapred.JobClient: Virtual memory (bytes) snapshot=134798288

```

#### 4) Conversation of Image File into Sequence file

[illegible]

## 5) Duplicate Image removing



## Conclusion

Hadoop is most recent framework used to handle and process Big amount of data which is mostly adopted by social media and financial industries. Using this Mapreduce job, we can reduce time of Image Processing and also remove duplicate Images in fraction of seconds from huge data sets.

## **References**

[http://www.academia.edu/4112213/Duplicate\\_Record\\_Elimination\\_using\\_Priority\\_Queue\\_Algorithm](http://www.academia.edu/4112213/Duplicate_Record_Elimination_using_Priority_Queue_Algorithm)

<http://www.datacenterknowledge.com/archives/2013/01/18/facebook-builds-new-data-centers-for-cold-storage/>

<http://blog.iconfinder.com/detecting-duplicate-images-using-python/>

<http://royal.pingdom.com/2010/06/18/the-software-behind-facebook/>

<http://www.computerworld.com/article/2690856/8-big-trends-in-big-data-analytics.html>

<http://dl.acm.org/citation.cfm?id=2185816>

<http://hipi.cs.virginia.edu/doc/api/index.html>

<http://www.academic-pub.org/ojs/index.php/ijecs/article/view/1092>