Volume No:2, Issue No:9 (February-2017) ISSN No : 2454-423X (Online)

International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

Image Clustering Using Hadoop Image Processing Interface

Damarla Haritha
Department of Computer Science and Engineering
MVSR Engineering College,
Hyderabad, Telangana - 501510, India.

Abstract
Huge amounts of images are uploaded into the internet daily. As large image collections cannot be processed efficiently on one computer, image processing often requires distributed computing. Image processing can be very computationally demanding due to the large amount of images to process. Image Processing with Parallel computing is an alternative way to solve image processing problems that require large times of processing or handling large amounts of information in "acceptable time". The main idea of parallel image processing is to divide the problem into simple tasks and solve them concurrently, in such a way the total time can be divided between the total tasks (in the best case).

As there is no such distributed framework for processing of images, HIPI (Hadoop Image Processing Interface) allows image processing on distributed framework. Hadoop provides distributed computing framework for data but not for images. Images cannot be processed using hadoop, inorder to analyse large datasets of images, there is an interface for image processing i.e hadoop image processing interface(HIPI). HIPI is an interface over the hadoop file system which helps the users to run various image processing algorithms and analyse the images on a distributed framework. Hipi provides such a mechanism to process huge data sets of images. The Analysed images are then clustered using the em clustering algorithm

Key words: ***Image processing, Parallel Computing, Distributed Computing, HIPI***

INTRODUCTION

Distributed computing is a method of computer processing in which different parts of a program are run simultaneously on two or more computers that are communicating with each other over a network. Distributed computing is a type of segmented or parallel computing, but the latter term is most commonly used to refer to processing in which different parts of a program run simultaneously on two or more processors that are part of the same computer. While both types of processing require that a program be segmented divided into sections that can run simultaneously, distributed computing also requires that the division of the program take into account the different environments on which the different sections of the program will be running.

A Framework in which large problems can be divided into many small problems which are distributed to many computers. Later, the small results are reassembled into a larger solution. Distributed computing [1] is a natural result of using networks to enable computers to communicate efficiently.

Distributed Computing framework is used for creating and using compute clusters to execute computations in parallel across multiple processors in a single machine (SMP) [3-5], among many machines in a cluster, grid or cloud.

Distributed computing is the process of aggregating the power of several computing entities, which are logically distributed and may even be geologically distributed, to collaboratively run a single computational task in a transparent and coherent way, so that they appear as a single, centralized system [2]. Parallel computing is the simultaneous execution of the same task on multiple processors in order to obtain faster results. It is widely accepted that parallel computing is a branch of distributed computing, and puts the emphasis on generating large computing power by employing multiple processing entities simultaneously for a single computation task.

These multiple processing entities can be a multiprocessor system, which consists of multiple processors in a single machine connected by bus or switch networks, or a multicomputer system, which consists of several independent computers interconnected by telecommunication networks or computer networks. Besides in parallel computing, distributed computing has also gained significant development in enterprise computing. The main difference between enterprise distributed computing and parallel distributed computing [4] is that the former mainly targets on integration of distributed resources to collaboratively finish some task, while the later targets on utilizing multiple processors simultaneously to finish a task as fast as possible.

LITERATURE STUDY

Distributed Computing Framework

Distributed computing is a method of computer processing in which different parts of a program are run simultaneously on two or more computers that are communicating with each other over a network.

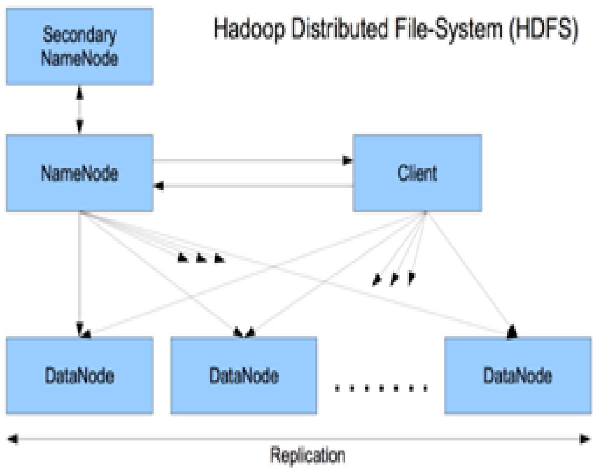
Distributed computing is a type of segmented or parallel computing, but the latter term is most commonly used to refer to processing in which different parts of a program run simultaneously on two or more processors that are part of the same computer. While both types of processing require that a program be segmented divided into sections that can run simultaneously, distributed computing also requires that the division of the program take into account the different environments on which the different sections of the program will be running.

A Framework in which large problems can be divided into many small problems that are distributed to many computers is called distributed framework. Later, the small results are reassembled into a larger solution. Distributed computing is a natural result of using networks to enable computers to communicate efficiently.

Distributed Computing framework is used for creating and using compute clusters to execute computations in parallel across multiple processors in a single machine (SMP), among many machines in a cluster, grid or cloud.

Apache Hadoop

The Apache Hadoop MapReduce project is a distributed computing framework that enables developers to write applications which run reliably on a large number of unreliable machines with the goal of processing Terabyte and larger data sets in paralleling clusters consisting of thousands of nodes [3]. Hadoop is an open source implementation of the MapReduce framework inspired by Google MapReduce, and the Google File System (GFS)[12], although the two systems are very different. The Hadoop Distributed File System, inspired by GFS from Google [2], is a distributed filesystem which runs on low cost commodity hardware in a fault tolerant manner to redundantly store Terabyte and larger data sets.



Hazelcast

Hazelcast[2] is an open source in-memory data grid based on Java. The Hazelcast[2] company is funded by venture capital. In a Hazelcast[2] grid, data is evenly distributed among the nodes of a computer cluster, allowing for horizontal scaling both in terms of available storage space and processing power.

JPPF[3] is a distributed parallel processing framework based on a master/worker architecture. A JPPF grid is made of 3 sorts of components that communicate with each other: clients which submit the work to the grid, nodes which execute the work, and servers which receive the work from clients and distribute it to the nodes in parallel.

Apache spark

Apache Spark[7] is an in-memory distributed data analysis platform that speeds up task processing. It provides high-level APIs in Java. It also supports a rich set of higher-level tools, including Shark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming [8], helping the development of parallel applications.

HIPI (HADOOP IMAGE PROCESSING INTERFACE)

The HIPI Framework:

HIPI[15] is an open-source Hadoop Image Processing Interface that aims to create an interface for Image Processing(IP) with Map Reduce technology. HIPI abstracts the highly technical details of Hadoop's system and is flexible enough to implement IP algorithms.

The following goals of HIPI are:

1. HIPI provides an open, extensible library for IP using Map Reduce technology.
2. With the help of HIPI, images are stored into the HDFS easily.
3. HIPI allows simple filtering of set of images.
4. Simple and unambiguous interface for IP in hadoop.
5. Enhances parallel processing of images.

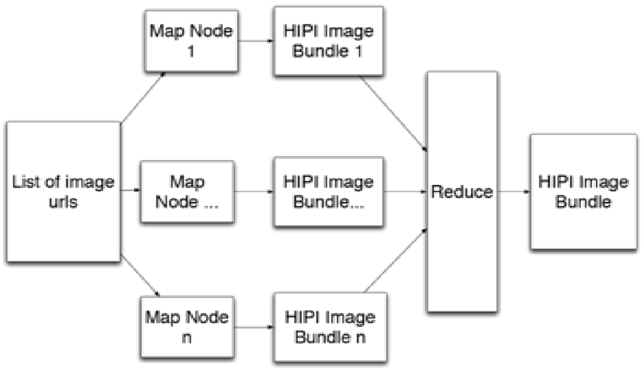


Fig 2.6: HIPI Map Reduce

The following work flow of HIPI is represented as:

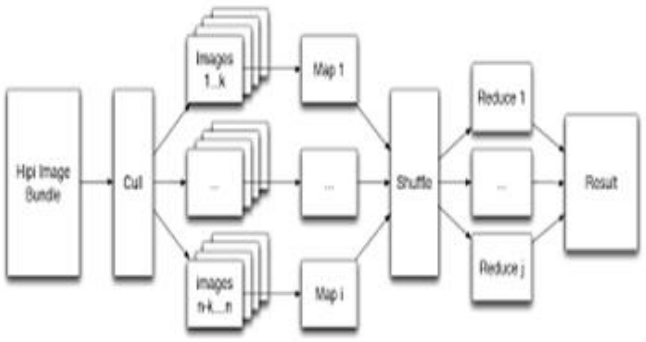


Fig 2.7: HIPI Framework

Data Storage:

A HIPI Image Bundle data type that stores many images in one large file so that MapReduce jobs can be performed more efficiently. A HIPI [4] Image Bundle consists of two files: a data file containing concatenated images and an index file containing information about the offsets of images in the data file as shown in the below fig.

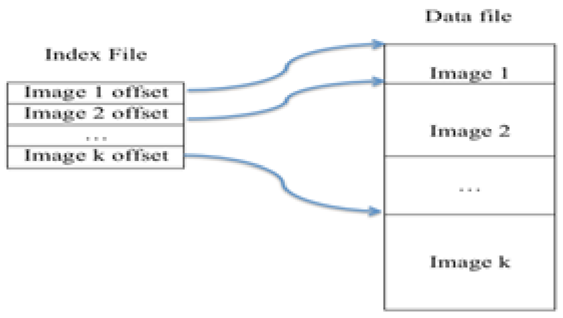


Fig 2.8: A depiction of the relationship between the index and data files in a HIPI Image Bundle

Downloading Images:

The following steps are used for downloading the images. They are:

1. The list of images should be stored by their image urls and should be present in a text file with exactly one image url per line.
2. For parallel execution, the input list of image urls are split and the number of nodes used to download these images are specified.
3. Download the images from the internet using the url's.
4. The downloaded images are stored in the image bundle (HIB).

XHAMI (Extended HDFS and Map Reduce Interface for Image Processing Applications)

Image processing applications deal with processing of pixels in parallel, for which Hadoop and MapReduce can be effectively used to obtain higher throughputs. XHAMI [4] offers extended library of HDFS [5] and MapReduce to process the single large scale images with high level of abstraction over writing and reading the images. XHAMI [4] has an API to implement Image processing that is two phase extensions to HDFS and MapReduce programming model.

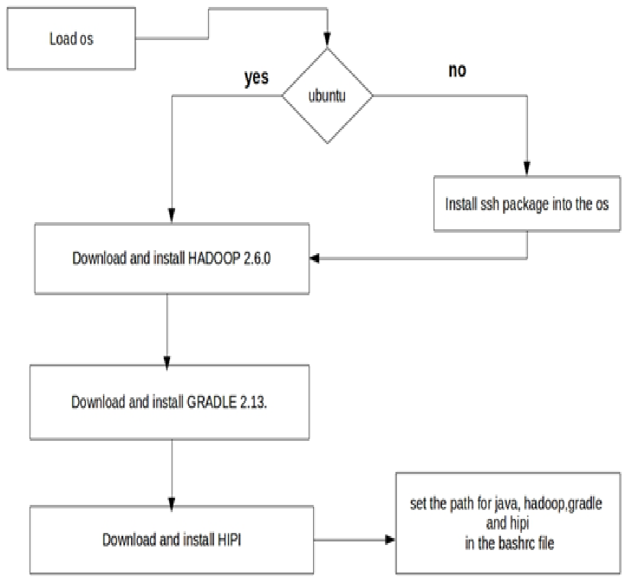

Load os
ubuntu
yes
no
Install ssh package into the os
Download and install HADOOP 2.6.0
Download and install GRADLE 2.13.
Download and install HIPI
set the path for java, hadoop,gradle and hipi in the bashrc file

Fig 3.1: Flowchart of environmental set up of Hadoop, HIPI,Gradle

ALGORITHM TO COMPUTE GREY LEVEL HISTOGRAM OF AN IMAGE

Algorithm:

Input: image, no. Of bins

Output: histogram values.

Steps:

1. Read the Input image .
2. Obtain the histogram of the input image by traversing image pixel data in raster-scan order and update running average in the bins.

Usage:

Haritha@ Haritha-Vostro-3549:~/work/hipi$ hadoop jar build/libs/helloWorld.jar tigers.hib hist1

ALGORITHM TO COMPUTE MEAN VALUES OF AN IMAGE

Algorithm:

Input: image

Output: average values.

Steps:

1. Read the Input image.
2. Obtain the mean values of the input image by averaging image pixel data and update running averages cumulatively.

Usage:

Haritha@ Haritha -Vostro-3549:~/work/hipi$ hadoop jar build/libs/helloWorld.jar tigers.hib average

EM CLUSTERING ALGORITHM

The EM algorithm was explained and given its name in a classic 1977 paper by Arthur Dempster, Nan Laird, and Donald Rubin. They pointed out that the method had been "proposed many times in special circumstances" by earlier authors. In particular, a very detailed treatment of the EM method [6] for exponential families was published by Rolf Sundberg in his thesis and several papers following his collaboration with Per Martin-Löf and Anders Martin-Löf.

Algorithm Overview

An expectation–maximization (EM) algorithm is an

Volume No:2, Issue No:9 (February-2017) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

iterative method for finding maximum likelihood or maximum a posteriori (MAP) [7] estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

EM Clustering

EM is an algorithm for maximizing a likelihood function when some of the variables in your model are unobserved (i.e. when you have latent variables).

To maximize a function, why don't we just use the existing machinery for maximizing a function. Well, if you try to maximize this by taking derivatives and setting them to zero, you find that in many cases the first-order conditions don't have a solution. There's a chicken-and-egg problem in that to solve for your model parameters you need to know the distribution of your unobserved data; but the distribution of your unobserved data is a function of your model parameters.

E-M tries to get around this by iteratively guessing a distribution for the unobserved data, then estimating the model parameters by maximizing something that is a lower bound on the actual likelihood function, and repeating until convergence - The EM algorithm Starts with guess for values of your model parameters.

E-step

For each data point that has missing values, use your model equation to solve for the distribution of the missing data given your current guess of the model parameters and given the observed data (note that you are solving for a distribution for each missing value, not for the expected value).

Now that we have a distribution for each missing value, we can calculate the expectation of the likelihood function with respect to the unobserved variables. If our guess for the model parameter was correct, this expected likelihood will be the actual likelihood of our observed data; if the parameters were not correct, it will just be a lower bound.

M-step

Now that we've got an expected likelihood function with no unobserved variables in it, maximize the function as you would in the fully observed case, to get a new estimate of your model parameters.
Repeat until convergence.

Applications

EM is frequently used for data clustering in machine learning and computer vision. In natural language processing, two prominent instances of the algorithm are the Baum-Welch algorithm [8] (also known as forward-backward) and the inside-outside algorithm for unsupervised induction of probabilistic context-free grammars.

In psychometrics, EM is almost indispensable for estimating item parameters and latent abilities of item response theory models.

With the ability to deal with missing data and observe unidentified variables, EM is becoming a useful tool to price and manage risk of a portfolio [9].

The EM algorithm (and its faster variant Ordered subset expectation maximization) is also widely used in medical image reconstruction, especially in positron emission tomography and single photon emission computed tomography. See below for other faster variants of EM.

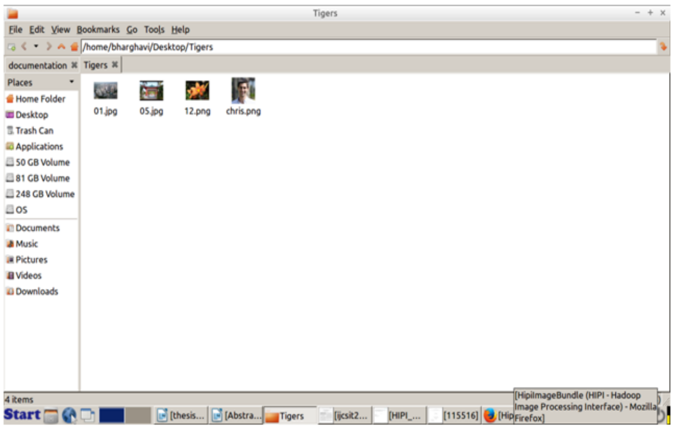
RESULTS

This chapter shows the results of an image dataset of 1000 images and then the images are clustered based on

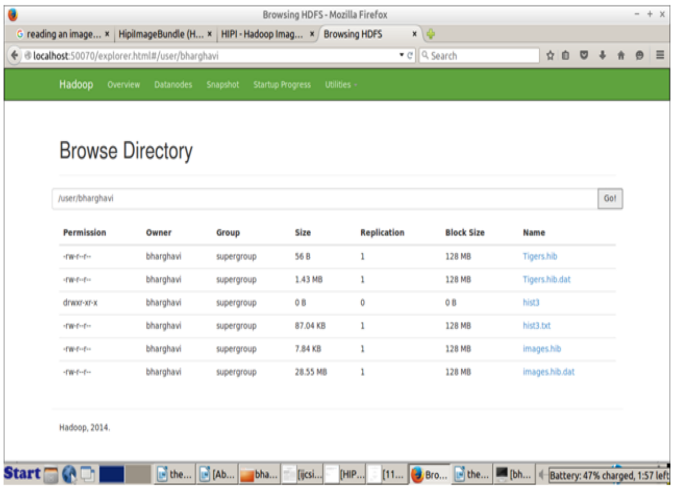
Volume No:2, Issue No:9 (February-2017) ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

their grey level histogram.

Input image folder (Tigers):



Input image in HDFS (Tigers.hib):


Browsing HDFS - Mozilla Firefox
localhost:50070/explorer.html#/user/bharghavi
Hadoop Overview Datanodes Snapshot Startup Progress Utilities
Go!
Start
Battery: 47% charged, 1:57 left

Browse Directory

/user/bharghavi

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	bharghavi	supergroup	56 B	1	128 MB	Tigers.hib
-rw-r--r--	bharghavi	supergroup	1.43 MB	1	128 MB	Tigers.hib.dat
drwxr-xr-x	bharghavi	supergroup	0 B	0	0 B	hist3
-rw-r--r--	bharghavi	supergroup	87.04 KB	1	128 MB	hist3.txt
-rw-r--r--	bharghavi	supergroup	7.84 KB	1	128 MB	images.hib
-rw-r--r--	bharghavi	supergroup	28.55 MB	1	128 MB	images.hib.dat

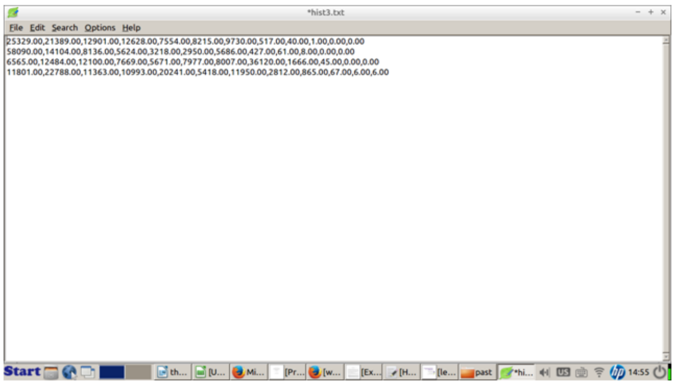
Hadoop, 2014.

Grey level Histogram Results:

The following shows the results of 4 images that are given as input to the histogram algorithm and the histogram values of the images are obtained respectively.

Table 6.1: List of Grey level Histogram values

image	Histogram values
1	25329.00,21389.00,12901.00,12628.00,7554.00,8215.00,9730.00,517.00,40.00,1.00,0.00,0.00
2	58090.00,14104.00,8136.00,5624.00,3218.00,2950.00,5686.00,427.00,61.00,8.00,0.00,0.00
3	6565.00,12484.00,12100.00,7669.00,5671.00,7977.00,8007.00,36120.00,1666.00,45.00,0.00,0.00
4	11801.00,22788.00,11363.00,10993.00,20241.00,5418.00,11950.00,2812.00,865.00,67.00,6.00,6.00



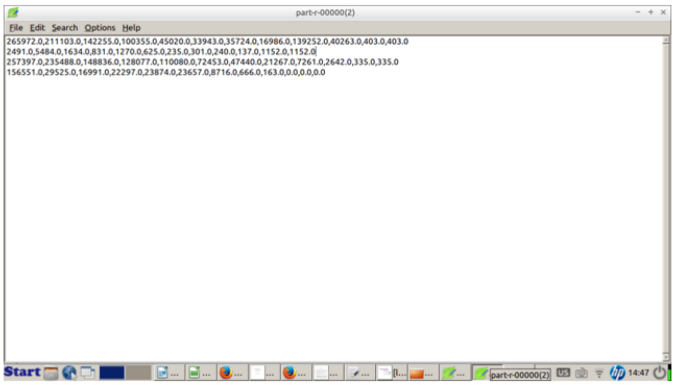
```
25329.00,21389.00,12901.00,12628.00,7554.00,8215.00,9730.00,517.00,40.00,1.00,0.00,0.00
58090.00,14104.00,8136.00,5624.00,3218.00,2950.00,5686.00,427.00,61.00,8.00,0.00,0.00
6565.00,12484.00,12100.00,7669.00,5671.00,7977.00,8007.00,36120.00,1666.00,45.00,0.00,0.00
11801.00,22788.00,11363.00,10993.00,20241.00,5418.00,11950.00,2812.00,865.00,67.00,6.00,6.00
```

Mean value Results:

The following shows the results of 4 images that are given as input to the mean algorithm and the mean values of the images are obtained respectively.

Table 6.2: List of Mean values

image	Mean values
1	265972.0,211103.0,142255.0,100355.0,45020.0,33943.0,35724.0,16986.0,139252.0,40263.0, 403.0,403.0
2	2491.0,5484.0,1634.0,831.0,1270.0,625.0,235.0,301.0,240.0,137.0,1152.0,1152.0
3	257397.0,235488.0,148836.0,128077.0,110080.0,72453.0,47440.0,21267.0,7261.0,2642.0, 335.0,335.0
4	156551.0,29525.0,16991.0,22297.0,23874.0,23657.0,8716.0,666.0,163.0,0.0,0.0,0.0



```
265972.0,211103.0,142255.0,100355.0,45020.0,33943.0,35724.0,16986.0,139252.0,40263.0,403.0,403.0
2491.0,5484.0,1634.0,831.0,1270.0,625.0,235.0,301.0,240.0,137.0,1152.0,1152.0
257397.0,235488.0,148836.0,128077.0,110080.0,72453.0,47440.0,21267.0,7261.0,2642.0,335.0,335.0
156551.0,29525.0,16991.0,22297.0,23874.0,23657.0,8716.0,666.0,163.0,0.0,0.0,0.0
```

EM Clustering

The following shows the results of Clustering 1000 images on 10 clusters using EM Clustering algorithm on a single node.

The 1000 images are put into a folder on the physical system

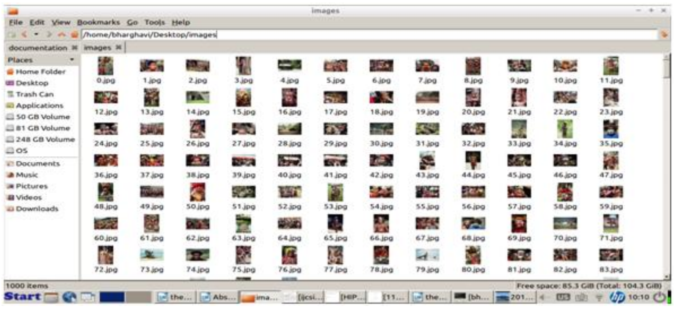
$mkdir /home/bharghavi/Desktop/images

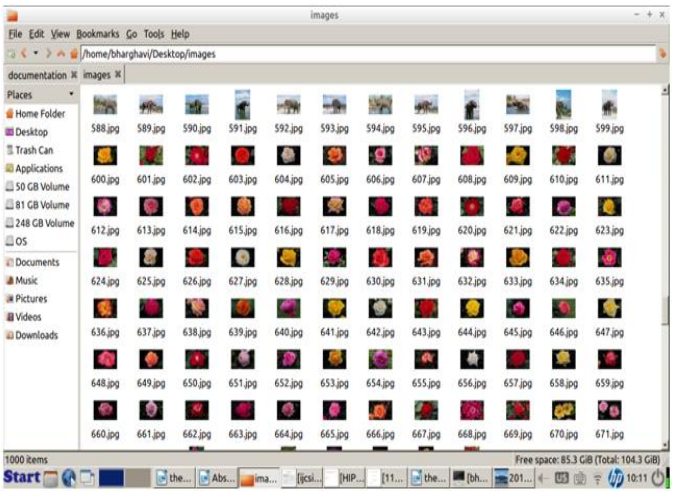
Volume No:2, Issue No:9 (February-2017) ISSN No : 2454-423X (Online)

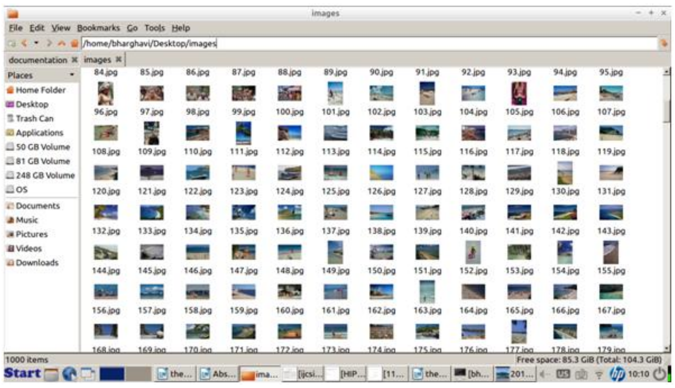
International Journal of Research in Advanced Computer Science Engineering

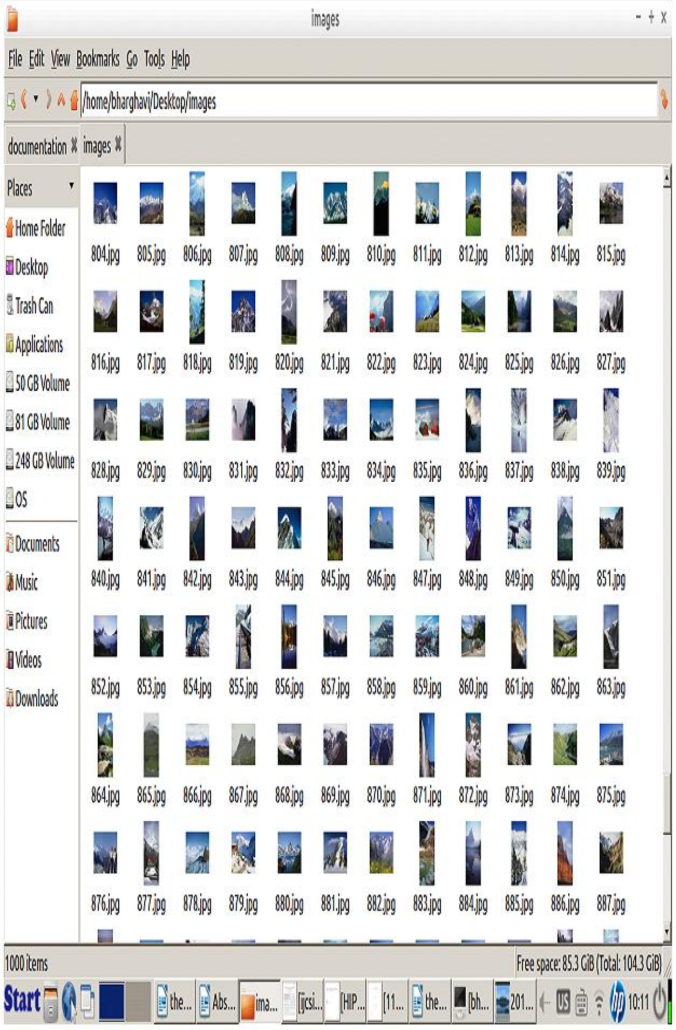
A Peer Reviewed Open Access International Journal

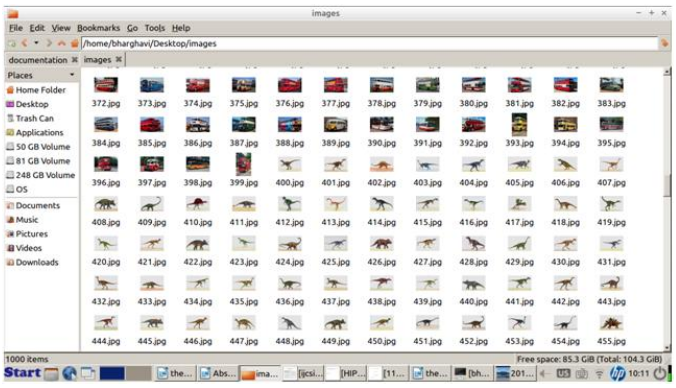
www.ijracse.com

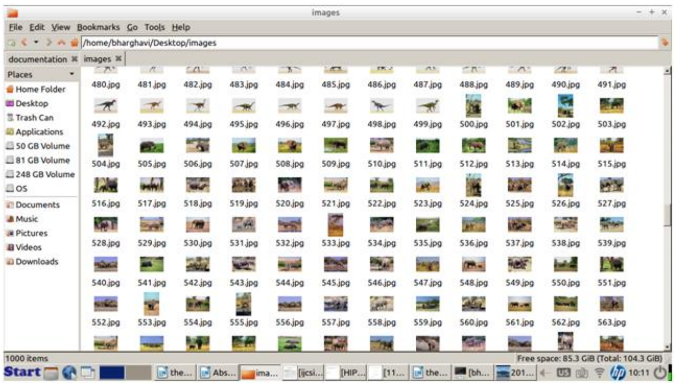



images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
documentation images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)
Start


images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
documentation images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)
Start


images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
documentation images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)
Start


images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
documentation images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)
Start

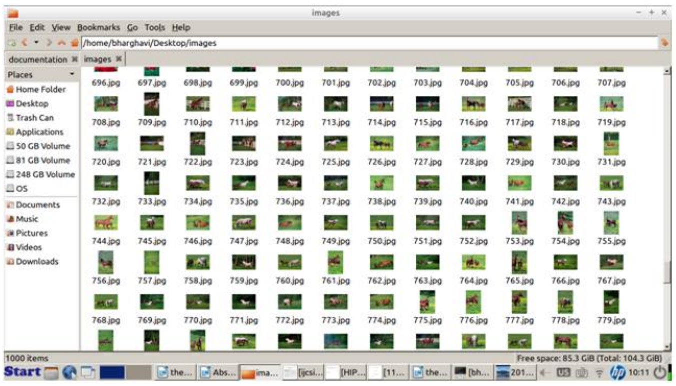

images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
documentation images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)
Start

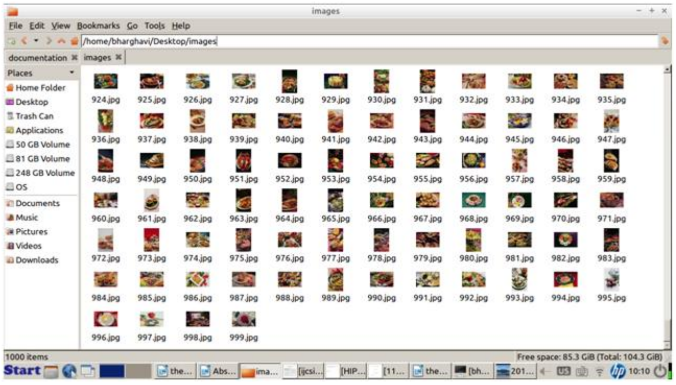
Volume No:2, Issue No:9 (February-2017) ISSN No : 2454-423X (Online)

International Journal of Research in Advanced Computer Science Engineering

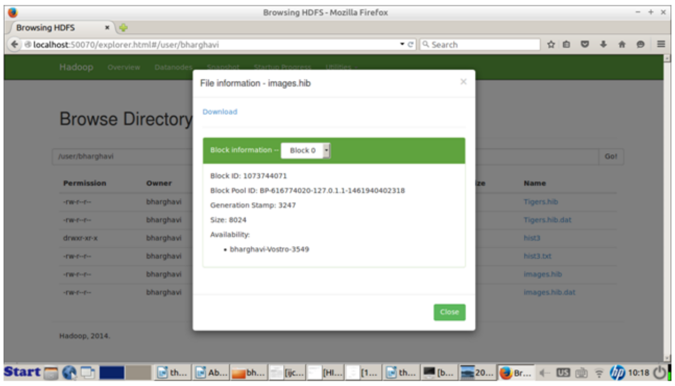
A Peer Reviewed Open Access International Journal

www.ijracse.com


images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)


images
File Edit View Bookmarks Go Tools Help
/home/bharghavi/Desktop/images
1000 items
Free space: 85.3 GiB (Total: 104.3 GiB)

These 1000 images are saved into the hdfs as images.hib. The screen shot is given below:


Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/user/bharghavi

Browse Directory

/user/bharghavi

Permission	Owner	Name
-rw-r--r--	bharghavi	Tigers.hib
-rw-r--r--	bharghavi	Tigers.hib.dat
drwxr-xr-x	bharghavi	hist3
-rw-r--r--	bharghavi	hist3.txt
-rw-r--r--	bharghavi	images.hib
-rw-r--r--	bharghavi	images.hib.dat

Hadoop, 2014.

File information - images.hib

Download

Block information -- Block 0

Block ID: 1073744071
Block Pool ID: BP-616774020-127.0.1.1-1461940402318
Generation Stamp: 3247
Size: 8024
Availability:
• bharghavi-Vostro-3549

Conclusion

HIPI is an image processing map-reduce framework that is designed to hide the complex details of Hadoop's powerful Map-reduce framework for processing the images. HIPI provides a format for storing images for efficient access within the Map Reduce pipeline, and simple methods for creating and storing image files of the float type and is stored in the form of HIB. HIPI interface brings about a new level of simplicity for creating large-scale vision applications that use the map-reduce framework for the processing of images. The features of the images are extracted by HIPI framework and given to the EM clustering algorithm in Hadoop for clustering.

Future Work

The future work includes implementation of more complex Image processing algorithms on the proposed system with the integration of OpenCV tool on a multinode setup of HIPI.

REFERENCES

[1] Shvachko, Konstantin, Hairong Kuang, Sanjay Radia, and Robert Chansler. "The hadoop distributed file system." In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, pp. 1-10. IEEE, 2010.

[2] Vemula, Sridhar, and Christopher Crick. "Hadoop Image Processing Framework." 2015 IEEE International Congress on Big Data. IEEE, 2015.

[3] Zaharia M, Das T, Li H, et al. (2012) Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. Proc. 4 th Edition.

[4] Apache Storm. https://storm.apache.org/. Accessed 27 Dec 2014.

[5] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. SIGOPS Oper. Syst. Rev., 37(5):29–43, 2003.

[6] Kune, Raghavendra, et al. "XHAMI--Extended HDFS and MapReduce Interface for Image Processing Applications." 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). IEEE, 2015.

[7] Sweeney, Chris, et al. "HIPI: a Hadoop image processing interface for image-based mapreduce tasks." Chris. University of Virginia (2011).

[8] Shvachko, Konstantin, Hairong Kuang, Sanjay Radia, and Robert Chansler. "The hadoop distributed file system." In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, pp. 1-10. IEEE, 2010.

[9] Tom White. Hadoop: The Definitive Guide. O’Reilly Media, Inc., 3rd edition, 2009.