

# Hadoop Image Processing Interface on Multinode Storage Set

D Siva Chidambaram<sup>1</sup>, Venkatesh D<sup>2</sup>, Vinothkumar V<sup>3</sup>, Kathirvel Raj S<sup>4</sup>

<sup>1</sup> Assistant Professor, Department of Computer science and Engineering, Sri Muthukumaran Institute of Technology, Chennai, India.

<sup>2, 3, 4</sup> Student, Department of Computer science and Engineering, Sri Muthukumaran Institute of Technology, Chennai, India.

**Abstract** – In the past few years the development in the computer science field and usage of internet led to generation of enormous data all over the world. The introduction of new technologies slowly decreased the cost of storage. But the data generation rate rising immensely. For analyzing textual data there are lot of tools in Hadoop. But the media files mostly occupied by the Image data. The main problem is not about the storage the images, its about processing and managing the data. Image data plays a major role in social media network. The amount of images being uploaded to the internet is rapidly increasing, with Facebook users uploading 2.5 billion new photos every month, however applications that make use of this data are severely lacking. Current computer vision applications use a small number of input images because of the difficulty is in acquiring computational resources and storage options for large amounts of data. The Hadoop Mapreduce platform provides a system for large and computationally intensive distributed processing, though use of Hadoops system is severely limited by the technical complexities. So we propose an open-source Hadoop Image Processing Interface (HIPI). In this paper we proposed how the image data was processed by using service of HDFS and Multinode Hadoop cluster.

**Index Terms** – Big data, Hadoop, Computer vision, Hadoop Image processing Interface (HIPI), Hadoop Distributed File System (HDFS), MapReduce.

## 1. INTRODUCTION

Hence Internet is an essential part for sharing images in social network. These images are stored and processed by using Hadoop ecosystem tools. Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data.

Sets using the MapReduce paradigm [1][2][4][13][14]. For storing the large set of data and for quicker process of data Hadoop framework is used. HDFS (Hadoop Distributed File System) [5][10], Mapreduce [12][14][26], HIPI (Hadoop Image processing Interface) [7][8] are the tools used to make the batch processing easier. Finally it saves the processing time.

In the first section and second section, a brief introduction of Big Data [3], Hadoop Services and the Hadoop-Ecosystem is given. In the third section processing and data flow of the MapReduce paradigm has been explained. In fourth section HIPI, its Image Bundle [19][21] formation and processing has

been described. Cluster [9] and its configurations as well as VMware configurations are explained in the fifth section. In the sixth section working of the paper has been explained and the conclusion is given in the last section.

## 2. BIG DATA

The defining characteristics of Big Data was described by below factors-

**Volume:** The term big data refers to very large quantities of data. While there isn't an exact size that qualifies a dataset for the big data label, most big data repositories are measured in terabytes or petabytes.

**Velocity:** Within most big data stores, new data is being created at a very rapid pace and needs to be processed very quickly. For example, the stream of data coming from social media feeds represents big data with a high velocity.

**Variety:** Big data comes from a wide variety of sources and resides in many different formats. A big data repository might include text files, images, video, audio files, presentations, spreadsheets, email messages and databases.

**Variability:** In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks. Is something trending in social media? Daily, seasonal and event-triggered peak data loads can be challenging to manage. Even more so with unstructured data.

**Complexity.** Today's data comes from multiple sources, which makes it difficult to link, match, cleanse and transform data across systems. However, it's necessary to connect and correlate relationships, hierarchies and multiple data linkages or your data can quickly spiral out of control.

### 2.1 Hadoop Architecture and services.

Hadoop is a generic processing framework designed to execute queries and other batch read operations on massive datasets that can scale from tens of terabytes to petabytes in size. HDFS and MapReduce together provide a reliable, fault-tolerant software framework for processing vast amounts of data in parallel on large clusters of commodity hardware (potentially scaling to

thousands of nodes) Hadoop meets the needs of many organizations for flexible data analysis capabilities with an unmatched price-performance curve. The flexibility in data analysis feature applies to data in a variety of formats, from unstructured data, such as raw text, to semi-structured data, such as logs, to structured data with a fixed schema.

HDFS is different from other distributed file systems in the sense that it uses a write-once-read-many model that relaxes concurrency requirements, provides simple data coherency, and enables high-throughput data access. HDFS prides on the principle and proves to be more efficient when the processing is done near the data rather than moving the data to the applications space. The data writes are restricted to one writer at a time. The bytes are appended to the end of the stream and are stored in the order written.

HDFS	Distributed file system Subject of this paper!
MapReduce	Distributed computation framework
HBase	Column-oriented table service
Pig	Dataflow language and parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	Distributed coordination service
Chukwa	System for collecting management data
Avro	Data serialization system

Table 2.1.1 Hadoop services

HDFS has many notable goals:

- Ensuring fault tolerance by detecting faults and applying quick recovery methods.
- MapReduce streaming for data access.
- Simple and robust coherency model.
- Processing is moved to the data, rather than data to processing.

The Hadoop ecosystem includes other tools to address particular needs:

**Common:** A set of components and interfaces for distributed filesystems and general I/O (serialization, Java RPC, persistent data structures).

**Pig:** A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters.

**Hive:** A distributed data warehouse. Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to MapReduce jobs) for querying the data.

**HBase:** A distributed, column-oriented database. HBase uses HDFS for its underlying storage, and supports both batch-style computations using MapReduce and point queries (random reads).

**ZooKeeper:** A distributed, highly available coordination service. ZooKeeper provides primitives such as distributed locks that can be used for building distributed applications.

**Sqoop:** A tool for efficiently moving data between relational databases and HDFS.

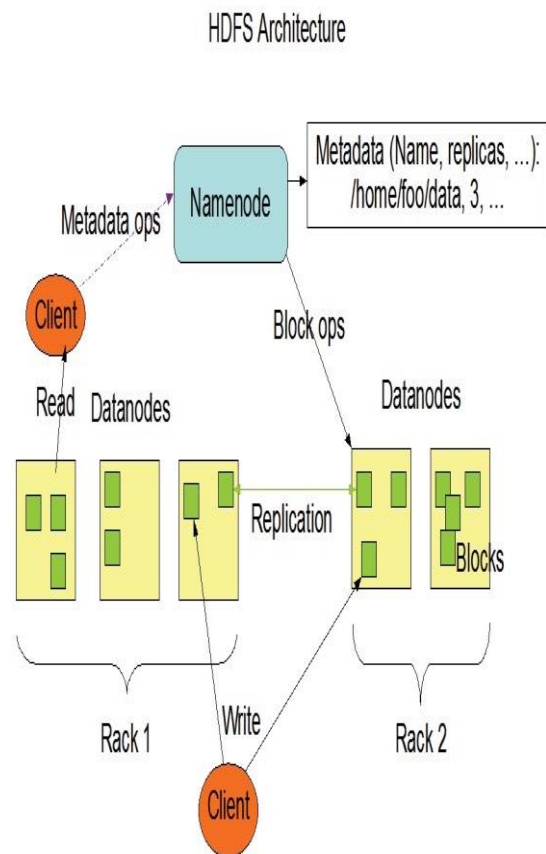


Figure 2.1.1 Hadoop Architecture

### 3. MAPREDUCE

Hadoop MapReduce is a simple software framework used to process vast amounts of data in the scale of higher terabytes in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job splits the dataset into independent chunks that are processed by many map tasks in parallel. The framework then sorts the output of this map which

are then input to the reduce task. The input and output of the job are stored on the file-system. The framework also takes care of scheduling, monitoring, and re-launching failed tasks. Typically, in Hadoop the storage and compute nodes are the same, the MapReduce framework and HDFS run on the same set of machines. This allows for the framework to schedule jobs on nodes where data are already present, resulting in high throughput across the cluster. The framework consists of one JobTracker on the master/Namenode and one TaskTracker per cluster node/Datanodes. The master is responsible for scheduling jobs, which are executed as tasks on the Datanodes. The Namenode is responsible for monitoring, re-executing failed tasks.

**List Processing:** MapReduce as paradigm transforms a list of input data into another set of output lists. The list processing idioms are map and reduce. These are principles of several functional programming languages like Scheme, LISP.

**Mapping Lists:** A list of items provided one at a time to a function called the mapper, which transforms these elements one at a time to an output data element. For example, a function like to Upper converts a string into its uppercase version, and is applied to all the items in the list; the input string is not modified but a new transformed string is created and returned.

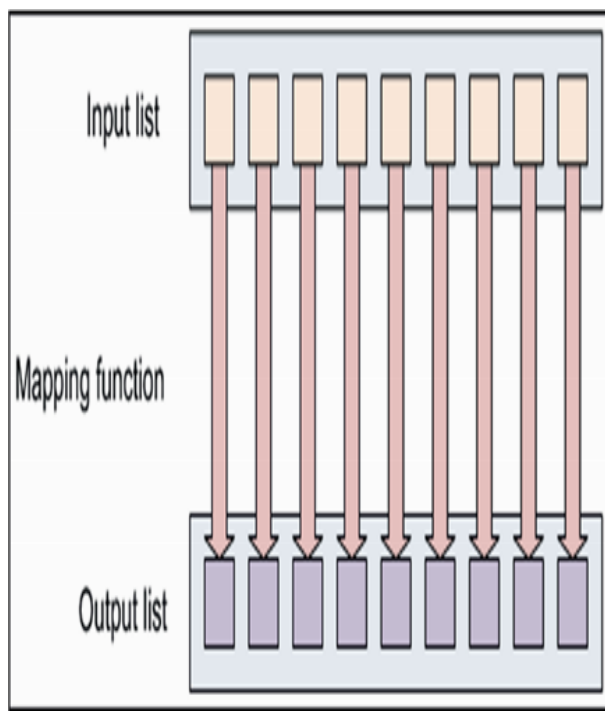


Figure 3.1 Mapping List

**Reducing Lists:** The intermediate outputs of the mappers are sent to a reducer function that receives an iterator over input values. It combines these values returning a single output.

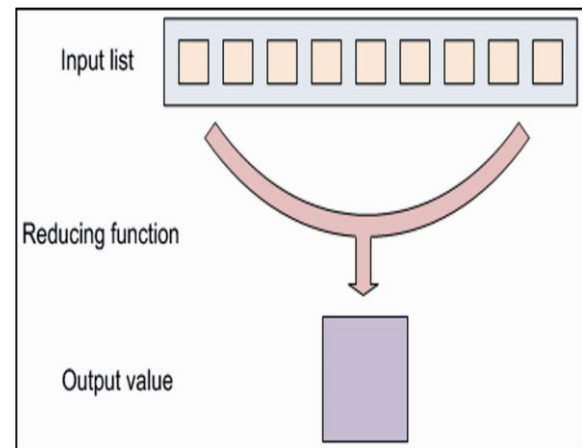


Figure 3.2 Reducing list

### Keys and Values:

Every value in an input for MapReduce program has a key associated to it. Keys identify related values. A collection of timestamped speedometer readings, the value, from different cars has the license plate number, the key, associated to it.

The MapReduce programming model, where a Hadoop application accesses Hadoop distributed filesystem (HDFS) in a cluster.

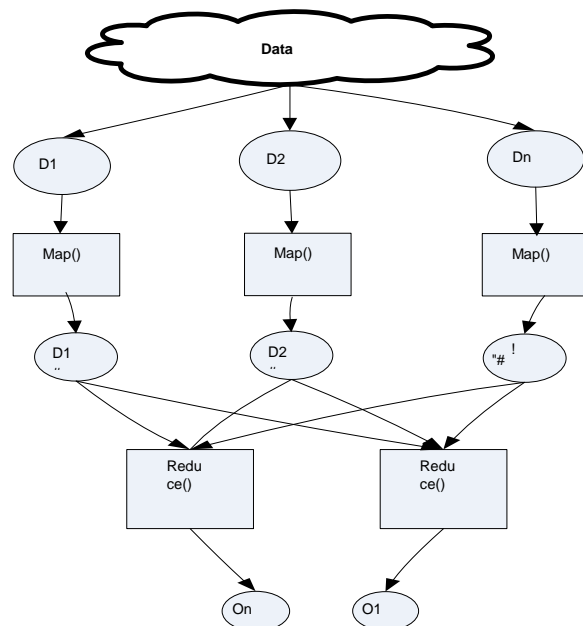


Figure 3.3 Mapreduce working

## 4. HADOOP IMAGE PROCESSING INTERFACE.

The Hadoop Image Processing Framework is intended to provide users with an accessible, easy-to-use tool for

developing large-scale image processing applications. The main goals of the Hadoop Image Processing Framework are:

- Provide an open source framework over Hadoop MapReduce for developing large-scale image applications.
- Provide the ability to flexibly store images in various Hadoop file formats
- Present users with an intuitive application programming interface for image-based operations which is highly parallelized and balanced, but which hides the technical details of Hadoop MapReduce.
- Allow interoperability between various image processing libraries.

#### Image-based MapReduce:

Using the HIPI Image Bundle data type as inputs, we have created an input specification that will distribute images in the HIPI Image Bundle across all map nodes.

As distribution is important in MapReduce, images should be processed in the same machine where the bundle block resides. In a generic MapReduce system, the user is responsible for creating InputFormat and RecordReader classes to specify the MapReduce job and distribute the input among nodes.

#### Processing image bundle using Mapreduce:

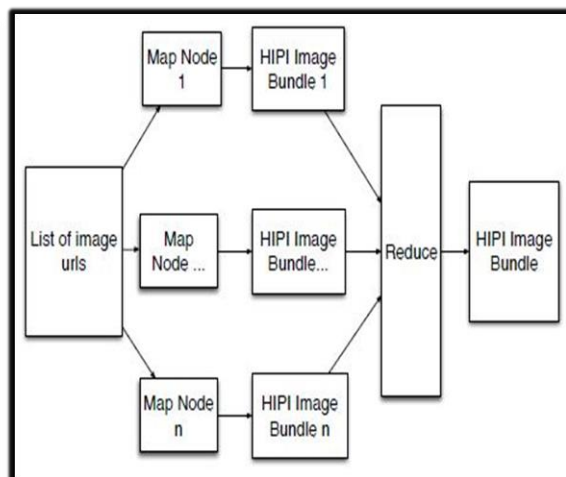


Figure 4.1 Hadoop Image bundle

Once a HIPI object is received, it can be added to the image bundle simply by passing the HIPI object to the appendImage method. Each map task generates a number of image bundles depending on the image list. In the reduce phase, all of these image bundles are merged into one large bundle. The HIPI class also provides interoperability between various image data types. These HIPIs are generated from the Inputstreams.

When importing these images into HDFS and converting them into bundles, two files are created. The first is a '.hib' file which contains the actual image data. The other is the '.hib.dat' file which contains the metadata of these images.

Images are distributed as various image data types and users have immediate access to pixel values. If pixel values are naively extracted from the image byte formats, valuable image header data (e.g. JPEG EXIF data or IHDR image headers) is lost. The framework holds the image data in a special HIPIHeader data type before converting the image bytes into pixel values. After processing pixel data, image headers are reattached to the processed results.

#### Extracting image bundles using MapReduce

In addition to creating and processing image bundles, the framework provides a method for extracting and viewing these images. Generally, Hadoop extracts images from an image bundle iteratively, inefficiently using a single node for the task.

The process of the Extractor module is explained below:

##### Step 1: Input the image bundle to be extracted:

- The image bundle specified for extraction is passed as a parameter to the Extractor module
- The image bundle is then distributed across the nodes as individual map tasks.
- Each map task will extract the requisite images onto the specified filesystem.

##### Step 2: Split Image bundle across nodes:

- The input image bundle is split across available nodes using the framework's custom input format and record reader classes for maximum throughput.

- Once a map task starts, HIPI objects are retrieved.

##### Step 3: Extract individual image.

- The image bytes obtained from each HIPI object are stored onto the filesystem in the appropriate file format based on its image type.
- The Extractor module lacks a reduce phase.

## 5. HADOOP CLUSTER

A Hadoop cluster is a special type of computational cluster designed specifically for storing and analyzing huge amounts of unstructured data in a distributed computing environment.

MultiNode cluster is that hadoop cluster which consists of a single master machine and one or more-than-one slave machines. Typically one machine in the cluster is designated as the NameNode and another machine as the JobTracker; these

are the masters. The rest of the machines in the cluster act as both DataNode and TaskTracker; these are the slaves.

#### VMware configuration:

VMware Workstation is used to setup Multinode hadoop cluster ,(i.e) 3 machines on 3 different systems, one master and two slaves. All the three machines used the Ubuntu operating system. The both master and slave machines must be able to reach each other over the network. The easiest is to put both machines in the same network with regard to hardware and software configuration is connect both machines via a single hub or switch and configure the network interfaces to use a common network such as 192.168.0.x/24. To make it simple, we will assign the IP address 192.168.0.1 to the master machine and 192.168.0.2 and 192.168.0.3 to the slave machines.

Configuration	Machines		
	Master	Slave1	Slave2
RAM	3GB	1GB	1GB
Processors allotted	2	2	2
Hard Disk	30GB	30GB	30GB

Table 5.1 Requirements for VMware

#### Cluster Configurations

After the machines have been successfully created and the connection has been established amongst them, following steps have to be performed in a specific order to set-up those created machines.

##### Step1: Add Entries in hosts file-

- Edit hosts file and add entries of master and slaves:

```
1. sudo nano /etc/hosts
2. MASTER-IP master
3. SLAVE01-IP slave01
4. SLAVE02-IP slave02
```

Figure 5.1 slaves files for master, slave01&slave02

##### Step2: Configure passwordless SSH-

- The ssh-key is required to be generated on the master node and that key is copied to the respective slaves in the cluster.

```
1. ssh slave01
2. ssh slave02
```

Figure 5.2 Passwordless slave machines

##### Step3: Hadoop multi-node cluster setup Configuration

The multimode cluster was created by editing several configuration files such as:

File name	Location
.bashrc	User home directory
hadoop-env.sh	HADOOP_HOME/etc/hadoop
core-site.xml	HADOOP_HOME/etc/hadoop
hdfs-site.xml	HADOOP_HOME/etc/hadoop
mapred-site.xml	HADOOP_HOME/etc/Hadoop

Table 5.1 Editing filenames with locations

The core-site.xml and hdfs-site.xml was edited by following entries :

```
1. <configuration>
2. <property>
3. <name>fs.defaultFS</name>
4. <value>hdfs://master:9000</value>
5. </property>
6. <property>
7. <name>hadoop.tmp.dir</name>
8. <value>/home/ubuntu/hdata</value>
9. </property>
10. </configuration>
```

Figure 5.3 core-site.xml for master.

```
1. <configuration>
2. <property>
3. <name>dfs.replication</name>
4. <value>2</value>
5. </property>
6. </configuration>
```

Figure 5.4 configurations for core-site.xml

- There is a file named yarn-site.xml. This file contains all the yarn related properties. This file is needed when the cluster runs Hadoop 2.x.x version.

A master file needs to be added on the machine of master (namenode) by the name 'masters'. This file will contain the name of the master machine. Another file named slaves is to be added on all the machines which need to have a slave-type of configuration. This file will contain the name of all the slave machines.

## 6. IMAGE PROCESSING

The Multinode cluster was created successfully and the Hadoop services are started by using the command "start-



dfs.sh" in terminal. After the service started the confirmation was given by "jps" command.

```
ubuntu@master:~$ jps
2768 ResourceManager
2530 SecondaryNameNode
2899 NodeManager
2200 NameNode
3193 Jps
2333 DataNode

ubuntu@slave2:~$ jps
6857 NodeManager
2875 DataNode
6989 Jps
```

Figure 6.1 jps command

### Processing :

After the services had been started, images were then imported from local disk to Hadoop Distributed File System (HDFS) and all those images were converted into a bundle for batch processing. Images from various sources were obtained and all of them were kept together in the same folder. These images were imported in the HDFS using the slave2 machine.

The following operations are carried out by master node only:

- The various attributes and parameters are stored by the image file such as the resolution, size, captured device.
- The required information are retrieved from the HIB files are stored in HDFS with help of MapReduce program.

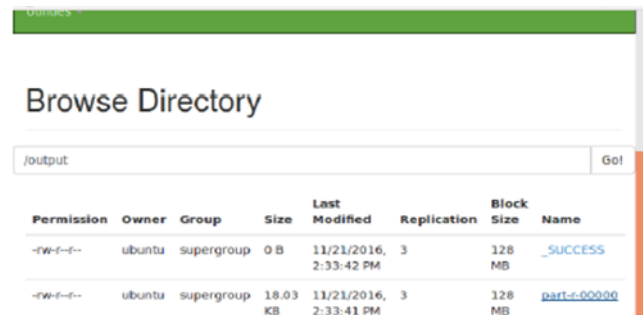
```
16/11/21 14:32:31 INFO mapreduce.Job: map 0% reduce 0%
16/11/21 14:32:45 INFO mapreduce.Job: map 26% reduce 0%
16/11/21 14:32:48 INFO mapreduce.Job: map 32% reduce 0%
16/11/21 14:32:52 INFO mapreduce.Job: map 34% reduce 0%
16/11/21 14:32:56 INFO mapreduce.Job: map 38% reduce 0%
16/11/21 14:32:59 INFO mapreduce.Job: map 40% reduce 0%
16/11/21 14:33:02 INFO mapreduce.Job: map 41% reduce 0%
16/11/21 14:33:05 INFO mapreduce.Job: map 44% reduce 0%
16/11/21 14:33:08 INFO mapreduce.Job: map 46% reduce 0%
16/11/21 14:33:11 INFO mapreduce.Job: map 48% reduce 0%
16/11/21 14:33:14 INFO mapreduce.Job: map 52% reduce 0%
16/11/21 14:33:17 INFO mapreduce.Job: map 58% reduce 0%
16/11/21 14:33:20 INFO mapreduce.Job: map 100% reduce 0%
16/11/21 14:33:43 INFO mapreduce.Job: map 100% reduce 100%
16/11/21 14:33:44 INFO mapreduce.Job: Job job_1479718792275_0001 completed successfully
16/11/21 14:33:44 INFO mapreduce.Job: Counters: 49
File System Counters
FILE: Number of bytes read=19580
FILE: Number of bytes written=273581
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
```

Figure 6.2 Mapreduce running the job

### Output:

If the MapReduce program ran successfully, then this directory will contain two files with the names '\_SUCCESS' and 'part-

r-000000'. The part-r-000000 file contains the output of the MapReduce task and it can be viewed easily. The output will contain three attributes: Resolution, Image Source, and Capture Device in a tab separated format respectively.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	ubuntu	supergroup	0 B	11/21/2016, 2:33:42 PM	3	128 MB	_SUCCESS
-rw-r--r--	ubuntu	supergroup	18.03 KB	11/21/2016, 2:33:41 PM	3	128 MB	part-r-000000

Figure 6.3 output directory in HDFS

## 7. CONCLUSION

In this paper, the unstructured data was successfully extracted HIB with the help of Hadoop framework and HIPI on multimode Hadoop cluster. The image Processing was performed on a MultiNode Hadoop Cluster and remarkable results were obtained. Around 100MB of images were collected from various sources such as camera devices, internet, and smartphones. The total processing time was just a one minute. The Hadoop MultiNode

Clusters make use of distributed and parallel architecture. MultiNode cluster and Map reduce in Hadoop enhances the performance of the networks for processing various operation. Hence, Multimode Clusters will surely aid in saving the time and efforts for the processing of ever-increasing data in the future.

## REFERENCES

- [1] F. N. Afrati and J. D. Ullman. Optimizing Joins in a Map-Reduce Environment. In EDBT, pages 99–110, 2010.
- [2] S. Babu. Towards automatic optimization of MapReduce programs. In SOCC, pages 137–142, 2010.
- [3] Manyika, James, et al. "Big data: The next frontier for innovation, competition, and productivity." (2011).
- [4] Dittrich, Jens, and Jorge-Arnulfo Quiané-Ruiz. "Efficient big data processing in Hadoop MapReduce." Proceedings of the VLDB Endowment 5.12 (2012): 2014-2015.
- [5] Hadoop, Apache. "Apache Hadoop." 2012-03-07]. <http://hadoop.apache.org> (2011).
- [6] Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, <http://labs.google.com/papers/mapreduce.html>.
- [7] Sweeney, Chris, et al. "HIPI: a Hadoop image processing interface for image-based mapreduce tasks." Chris. University of Virginia (2011)
- [8] Sridhar Vemula, Christopher Crick "Hadoop Image Processing Framework" Published in: Big Data (BigData Congress), IEEE International Congress on 2015
- [9] Cluster Setup - Apache™ Hadoop - The Apache Software Foundation [https://hadoop.apache.org/docs/r1.2.1/cluster\\_setup](https://hadoop.apache.org/docs/r1.2.1/cluster_setup)
- [10] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop distributed file system", MSST '10: Proc. of the 26th IEEE Symposium on Massive Storage Systems and Technologies 2010

- [11] Xie, Jiong, et al. "Improving mapreduce performance through data placement in heterogeneous hadoop clusters." *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 2010 IEEE International Symposium on. IEEE, 2010.
- [12] Hadoop, <http://hadoop.apache.org/mapreduce/>.
- [13] A. Floratou et al. Column-Oriented Storage Techniques for MapReduce. *PVLDB*, 4(7):419–429, 2011.
- [14] A. Gates et al. Building a High Level Dataflow System on Top of MapReduce: The Pig Experience. *PVLDB*, 2(2):1414–1425, 2009.
- [15] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *SOSP*, pages 29–43, 2003.
- [16] H. Herodotou and S. Babu. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. *PVLDB*, 4(11):1111–1122, 2011.
- [17] M. Isard et al. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In *EuroSys*, pages 59–72, 2007.
- [18] E. Jahani, M. J. Cafarella, and C. R'c. Automatic Optimization for MapReduce Programs. *PVLDB*, 4(6):385–396, 2011.
- [19] HANCOCK, P., BADDELEY, R., AND SMITH, L. 1992. The principal components of natural images. *Network: computation in neural systems* 3, 1, 61–70.
- [20] LI, Y., AND CRANDALL, D. 2009. Landmark classification in large-scale image collections. *Computer Vision* (Jan).
- [21] CONNER, J. 2009. Customizing input file formats for image processing in hadoop. Arizona State University. Online at: <http://hpc.asu.edu/node/97>.