“Implementation of Distributed Processing using Hadoop”

A

Project Report

submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

Name	Roll No.
Ayush Vardhan	R610213014
Ishan Sharma	R610213018

under the guidance of

Mr. Rajeev Tiwari

Assistant Professor, CIT, UPES

Department of Computer Science & Engineering

Centre for Information Technology

University of Petroleum & Energy Studies

Bidholi, Via Prem Nagar, Dehradun, UK

May – 2016

The innovation driven
E-School

CANDIDATE'S DECLARATION

I/We hereby certify that the project work entitled **"Implementation of Distributed Processing using Hadoop"** in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (with specialization in Mainframe Technology) and submitted to the Department of Computer Science & Engineering at Center for Information Technology, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from **February**, **2016** to **May**, **2016** under the supervision of **Mr. Rajeev Tiwari, Assistant Professor, CIT, UPES**.

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

Ayush Vardhan **Ishan Sharma**
(R610213014) **(R610213018)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 17th May 2016

Mr. Rajeev Tiwari
Project Guide

Mr. Hanumat G. Sastry
Program Head – CSE with Specialization in mainframe technology
Center for Information Technology
University of Petroleum & Energy Studies
Dehradun – 248 001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. Rajeev Tiwari**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, **Mr. Hanumat G Sastry**, for his great support in doing our project in **Distributed Processing with Hadoop** at **CIT**.

We are also grateful to **Dr. Manish Prateek, Associate Dean** and **Dr. Kamal Bansal,** Dean CoES, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Name	**Ayush Vardhan**	**Ishan Sharma**
Roll No.	**R610213014**	**R610213018**

ABSTRACT

“Implementation of Distributed Processing using Hadoop” is a project which is going to implement Distributed processing and explore its applications over centralized computing that inculcates Dictatorship, administrative system, under-utilization of resources and higher initial setup.

Distributed processing, a phrase that includes parallel processing where single node utilizes one or more processors and resources of other nodes. Its is also known as distributed computing.

Distributed processing give more performance than single system. If any one node in this environment goes down another will take care of the application and services that is available to cater the business customers. There are no limitations on the resources, more resources can be added easily.

The goal of the project is to implement distributed processing using Hadoop and then to compare the results from distributed computing with that of result from centralized computing.

Hadoop is a Java based framework for running applications on large clusters of hardware. Hadoop has a famous component - “HDFS” i.e, “Hadoop Distributed File System”, is a highly fault-tolerant distributed system. Hadoop provides high throughput access to application data and is suitable for application that have large data sets.

Hadoop is one of many ways Distributed Computing concept can be implemented, which is used in this project. Hadoop environment is a trending technology for distributed storage and distributed data processing of very large datasets on computer cluster. And so it is solving a big problem of industry - “BIG DATA”.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 History

Computation demands were always higher than technical status. The use of concurrent processes that communicate by message-passing has its root in Operating System architectures studied in 1960s. First widespread distributed system were Local Area Networks. Earliest example of a large scale distributed application was Electronic Mail (Email), that was application of ARPANET. Today various hardware and software architectures are used for distributed processing.

1.2 Requirement Analysis

Requirement to set up a Hadoop cluster basically include following :

1.2.1 Software Requirements:

1.2.1.1 GNU/Linux is supported as development/production environment.
1.2.1.2 Java 6.0 or above, must be installed
1.2.1.3 SSH must be installed.

1.2.2 Hardware Requirements:

1.2.2.1 Minimum RAM of 4GB.

1.3 Objectives

Main objective is to set-up a Hadoop distributed processing environment in distributed mode. Then implementing Map Reduce program that performs following tasks:

1.3.1 Reading a large data set
1.3.2 Getting useful insights
1.3.3 Comparing the results with traditional system.

1.4 Pert Chart Legend

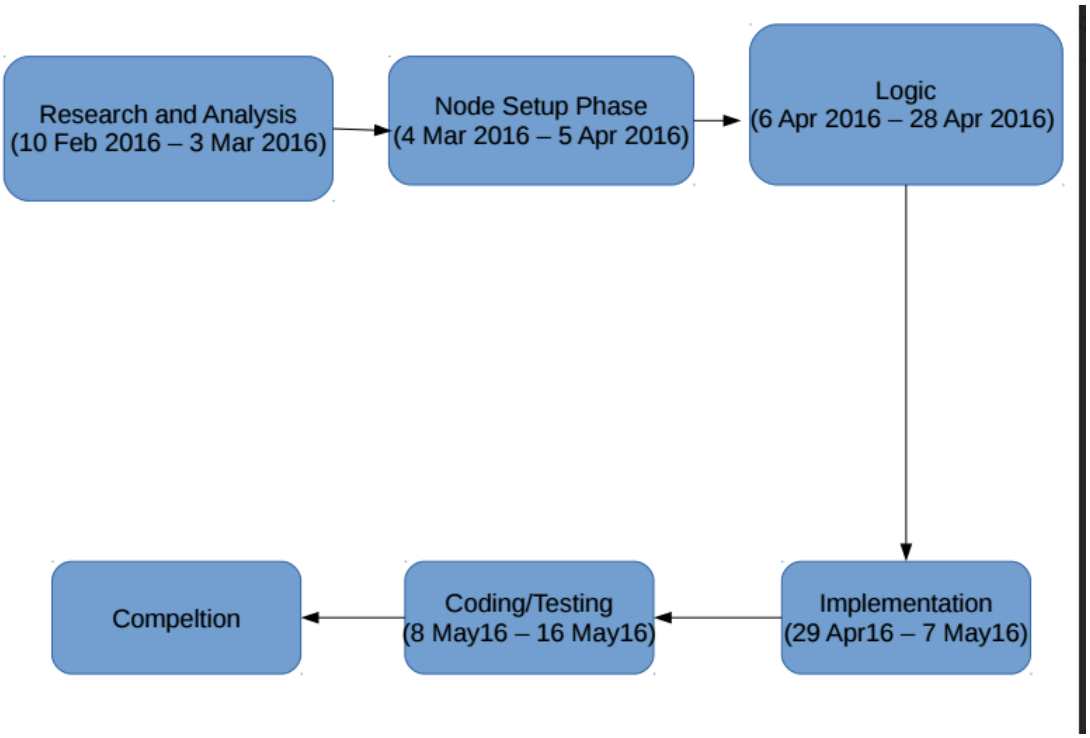


Fig: 1

2. SYSTEM ANALYSIS

2.1 Existing System

Centralized data processing is performed on a single computer/node or a cluster of coupled computers in a single location. All data processing is performed on central computer. In a centralized IT Infrastructure, a central computing facility is there where the entire data, irrespective of source, origin and type are located and processed. Some advantages include: I) Economies of scale in procurement of hardware. II) Easy maintenance and software facilities. Centralized infrastructure was popular at time of business on Mainframe Computers. These control systems were not very efficient and thus trend changed gradually.

2.2 Motivations

With the advancements in data communication technologies and availability of reliable data communication facilities at low costs, business enterprises are switching over from centralized data processing to other degrees of decentralized data processing or distributed data processing, where it has low cost, reliability, flexibility, improved performance and reduced processing time.

2.3 Proposed System

There are many technologies that are implementing distributed computing advancement. Hadoop is one of all those great technologies that solves Enterprise problems. Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. There are three modes in which Hadoop can be deployed. These are:

I. Stand-Alone mode
II. Pseudo-Distributed mode
III. Fully-Distributed mode

We are following fully distributed mode of Hadoop cluster where, ‘n’ number of machines forming a Hadoop cluster. Hadoop daemons run on a cluster of machines. There is one host onto which Name node is running and another host on which data node is running and then there are machines on which task tracker is running. We have separate masters and separate slaves in this distribution.

2.4 Modules

2.4.1 Setting up Nodes for Hadoop Cluster

Hadoop almost follows the UNIX pattern. Hadoop is installed in "/usr/lib/hadoop-0.20/" directory. Setting up nodes for Hadoop cluster includes :

I. Adding dedicated Hadoop system user.
II. Configuring SSH

Then, hadoop is installed on machine that covers configuring three major files. 1. core-site.xml 2. hdfs-site.xml 3. mapred-site.xml. These files are located in "conf/" subdirectory. After configuring these files, HDFS is formatted via node name. Then its the time to start your single cluster. Similarly, Set up other clusters which will be inter-connected that would work as slave and master node accordingly as per network configurations.

In hadoop environment, Name node is the one, where JobTracker runs and which accepts job requests from clients while JobTracker is the one that schedules jobs and tracks the assign jobs to Task tracker.

2.4.2 Developing and executing Map Reduce

MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. This is a programming technique and program model to provide distributed computing with Java. This algorithm works on "Divide and Conquer" algorithm. It performs two important tasks: Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

During a MR job, Hadoop sends the map and reduce tasks to appropriate servers in the clusters.

3. MODELLING

3.1 Activity Diagrams

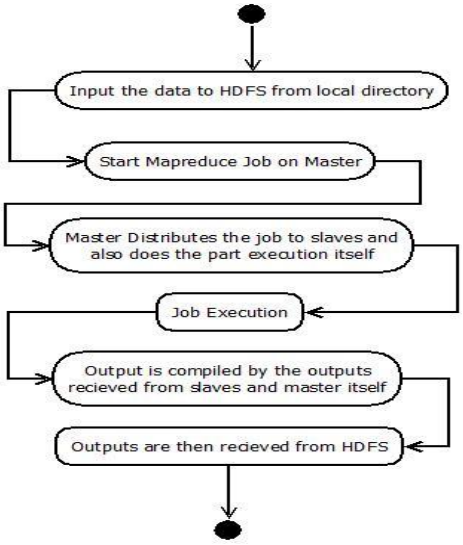


Fig. 2

3.2 Data Flow Diagrams

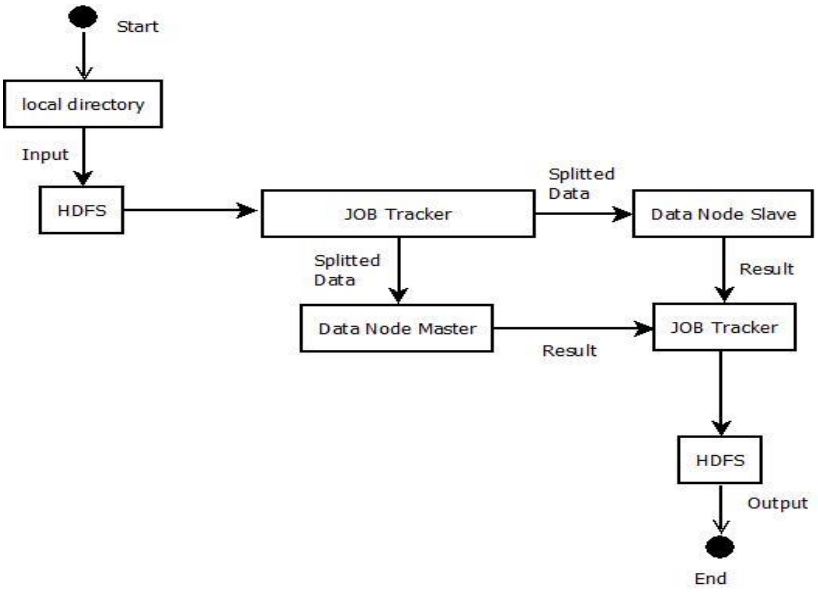


Fig. 3

3.3 Flow Chart

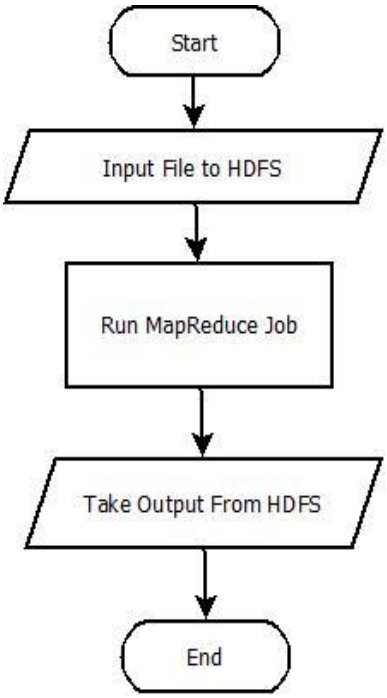


Fig. 4

4. IMPLEMENTATION OF DISTRIBUTED PROCESSING USING HADOOP

4.1 Distributed Processing

Distributed processing is a setup where multiple computer nodes work for same program utilizing their resources. Its a rough synonym for parallel processing, in which users string multiple computer together to achieve parallel processing. Here, a collection of computers appear as a single-system image. Arrangement of networked computers in which data processing capabilities are spread across the network. In DDP, specific jobs are performed by specialized computers which may be far removed from the user and/or from other such computers.

4.2 Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

4.2.1 Technical Specifications

Hadoop consists of MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper. MapReduce and Hadoop distributed file system (HDFS) are the main component of Hadoop. Hadoop cluster has 3 components: 1. Client, 2. Master, 3. Slave. Client is the one who loads data and submit map reduce jobs. Now, Name node, Secondary name node and Jobtracker combinely consist of Masters.

Name Node oversees the health of Data Node and coordinates access to the data stored in Data Node.

Job Tracker coordinates the parallel processing of data using MapReduce.

The job of Secondary Node is to contact NameNode in a periodic manner after certain time interval(by default 1 hour). NameNode which keeps all file-system metadata in RAM has no capability to process that metadata on to disk. So if NameNode crashes, you lose everything in RAM itself and you don't have any backup of file-system. What secondary node does is it contacts NameNode in an hour and pulls copy of metadata information out of NameNode. It shuffle and merge this information into clean file folder and sent to back again to NameNode, while keeping a copy for itself. Hence Secondary Node is not the backup rather it does job of housekeeping.

4.2.2 Features

Here are a few key features of Hadoop:

I. Flexibility in data processing.
II. Easily Scalable
III. Fault Tolerant
IV. Faster at data processing
V. Cost effective.

4.2.3 Applications

Apache Hadoop, the open source MapReduce framework, has dramatically lowered the cost barriers to processing and analyzing big data. Technical barriers remain, however, since Hadoop applications and technologies are highly complex and still foreign to most developers and data analysts. Talend, the open source integration company, makes the massive computing power of Hadoop truly accessible by making it easy to work with Hadoop applications and to incorporate Hadoop into enterprise data flows.

5. IMPLEMENTATION

5.1 Map Class

Maps are the individual tasks which transform input records into a intermediate records. The transformed intermediate records need not be of the same type as the input records. A given input pair may map to zero or many output pairs. Maps input key/value pairs to a set of intermediate key/value pairs.

5.2 Reduce Class

Reduces a set of intermediate values which share a key to a smaller set of values. Reducer has 3 primary phases:

-Shuffle

The Reducer copies the sorted output from each Mapper using HTTP across the network.

-Sort

The framework merge sorts Reducer inputs by keys (since different Mappers may have output the same key).

The shuffle and sort phases occur simultaneously i.e. while outputs are being fetched they are merged.

-Secondary Sort

To achieve a secondary sort on the values returned by the value iterator, the application should extend the key with the secondary key and define a grouping comparator. The keys will be sorted using the entire key, but will be grouped using the grouping comparator to decide which keys and values are sent in the same call to reduce.

5.3 Algorithms

1. Start
2. Import java and hadoop libraries.
3. Create class WordCountMids {
4. Create a “Map” class which extends “MapReduceBase” and implements “Mapper” interface <LongWritable, Text, Text, IntWritable> .
5. IntWritable one <- new IntWritable(1)
6. Text word <- new Text()

7. Create a “map” function with arguments (LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)
8. String line <- value.toString()
9. Scanner scanner <- new Scanner(line)
10. StringTokenizer tokenizer <- new StringTokenizer(line," ")
11. Check whether tokenizer has more value.
12. if yes,
word.set(tokenizer.nextToken())
output.collect(word, one);

// END OF map Method and “map” class.

13. Create a “Reduce” class which extends “MapReduceBase” and implements “Reducer” interface <Text, IntWritable, Text, IntWritable>.
14. Create a “reduce” function with arguments (Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter)
15. String search <- “apps";
16. int sum <- 0
17. Check for values (values.hasNext())
if yes :
18. sum <- sum + values.next().get();
19. if(search.equals(key.toString())){
output.collect(key, new IntWritable(sum));
}
//End of reduce method
//End of Reduce class

19. Create main() method.
20. JobConf conf <- new JobConf(WordCountMids.class);
21. conf.setJobName("wordcountmids");
22. conf.setOutputKeyClass(Text.class);
23. conf.setOutputValueClass(IntWritable.class);
24. conf.setMapperClass(Map.class);
25. conf.setReducerClass(Reduce.class);
26. FileInputFormat.setInputPaths(conf, new Path(args[0]));
27. FileOutputFormat.setOutputPath(conf, new Path(args[1]));
28. JobClient.runJob(conf);
29. End Main.
30. End Class WordCountMids.

6. OUTPUT SCREENS

Master Node:

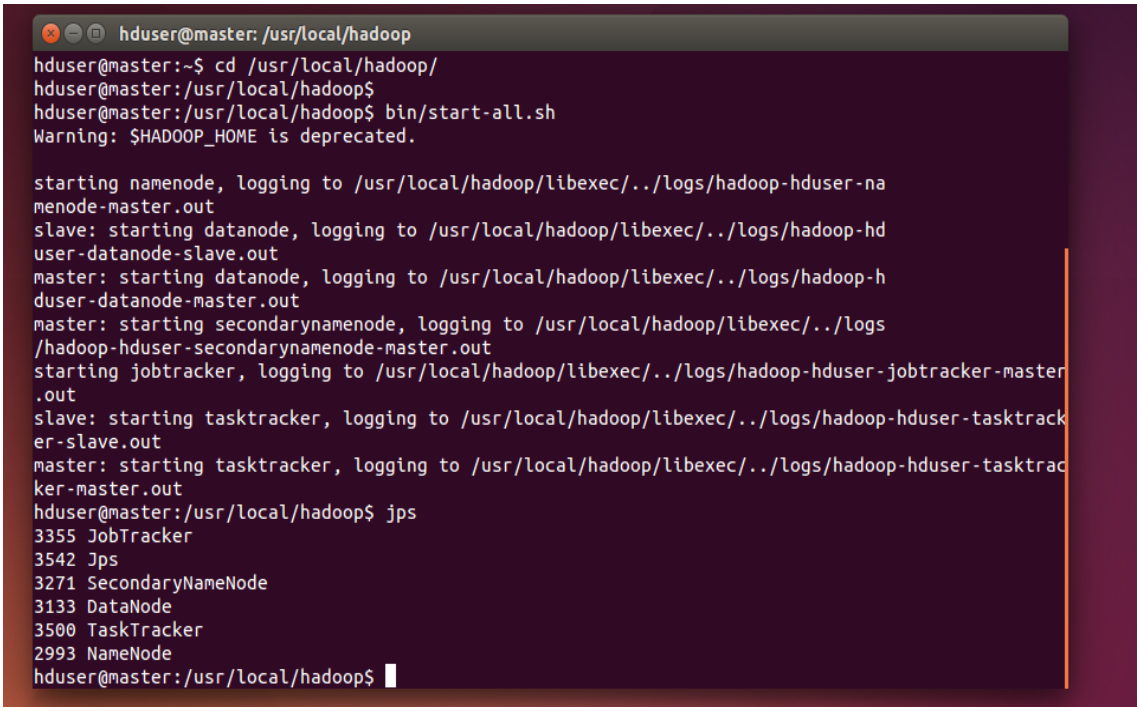


Fig. 5

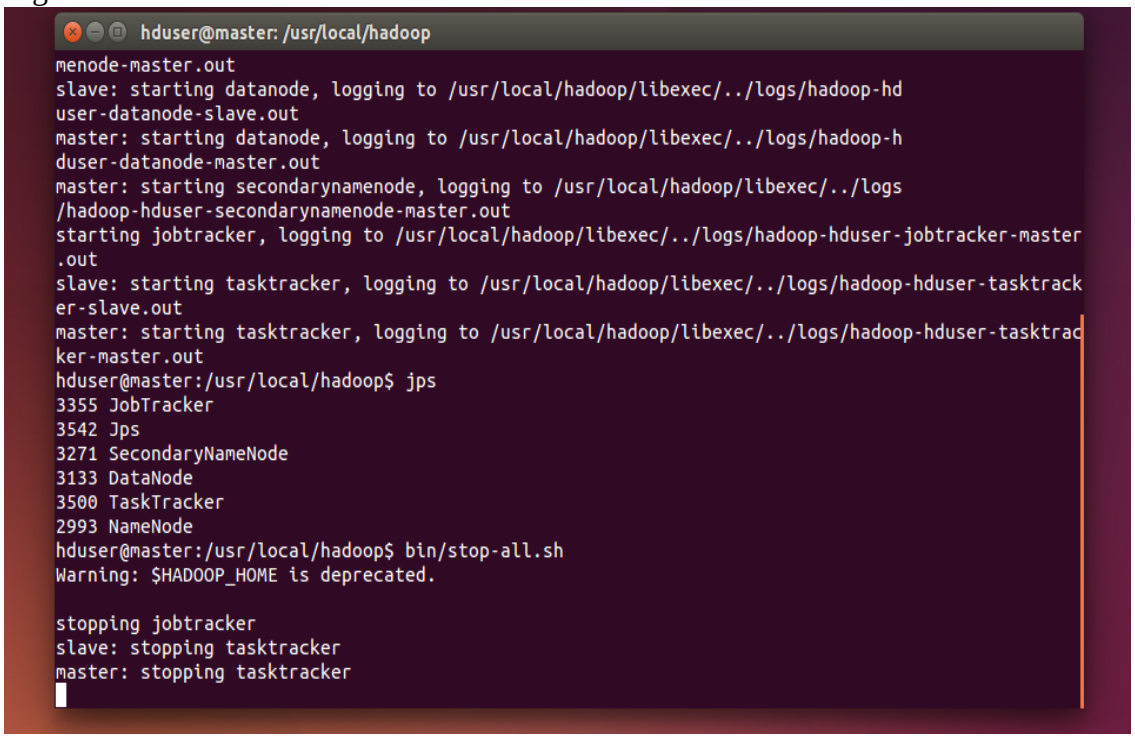


Fig. 6

Slave Node:

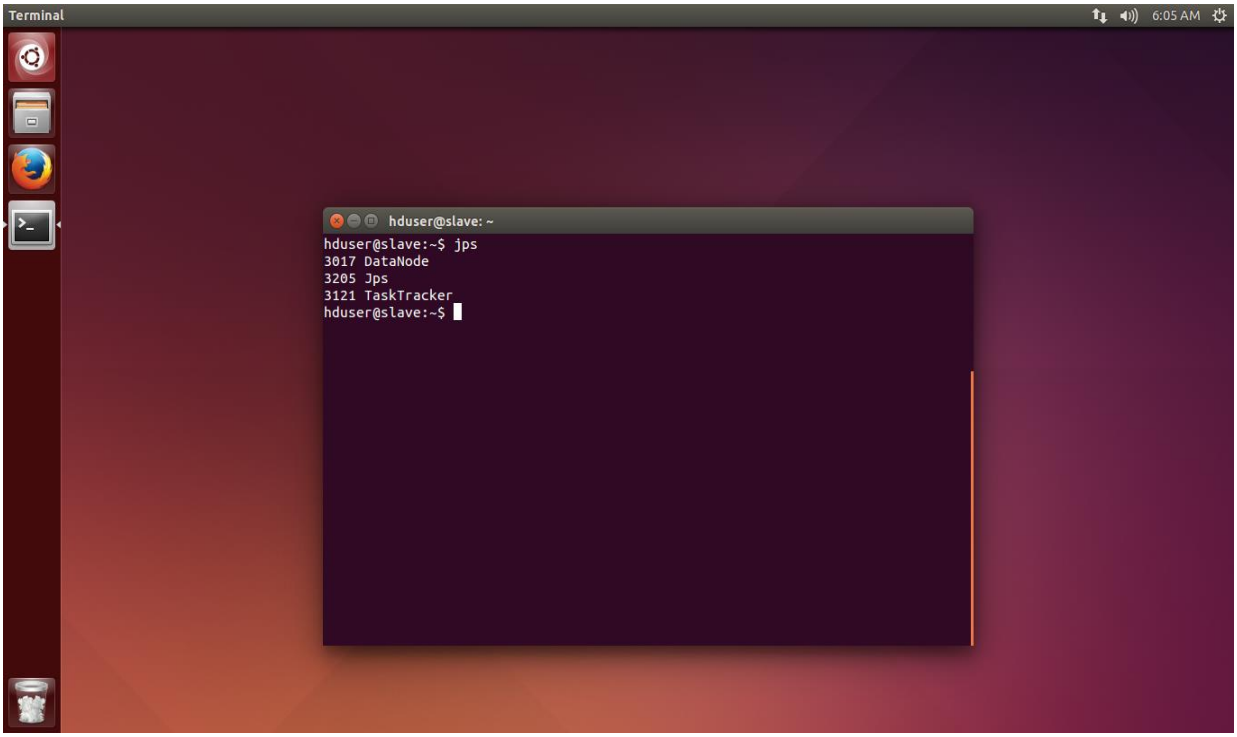


Fig. 7

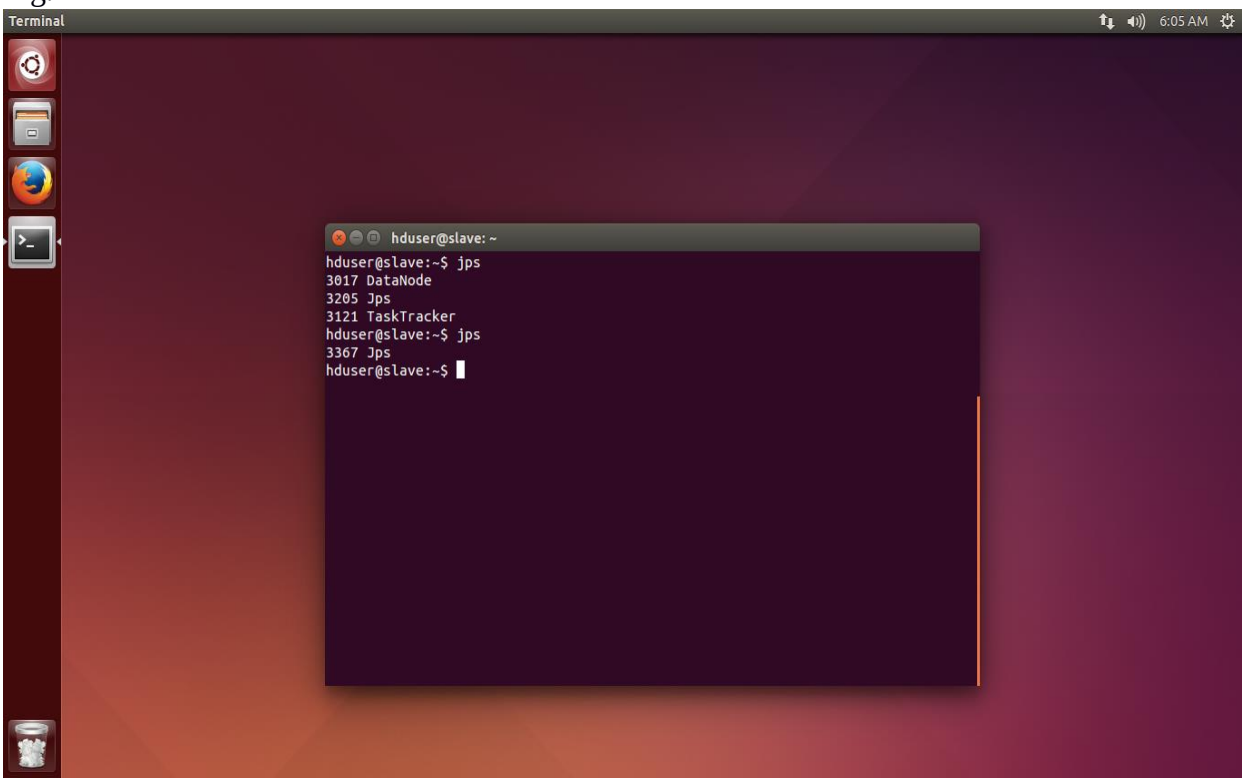


Fig. 8

Running a Map-Reduce Job

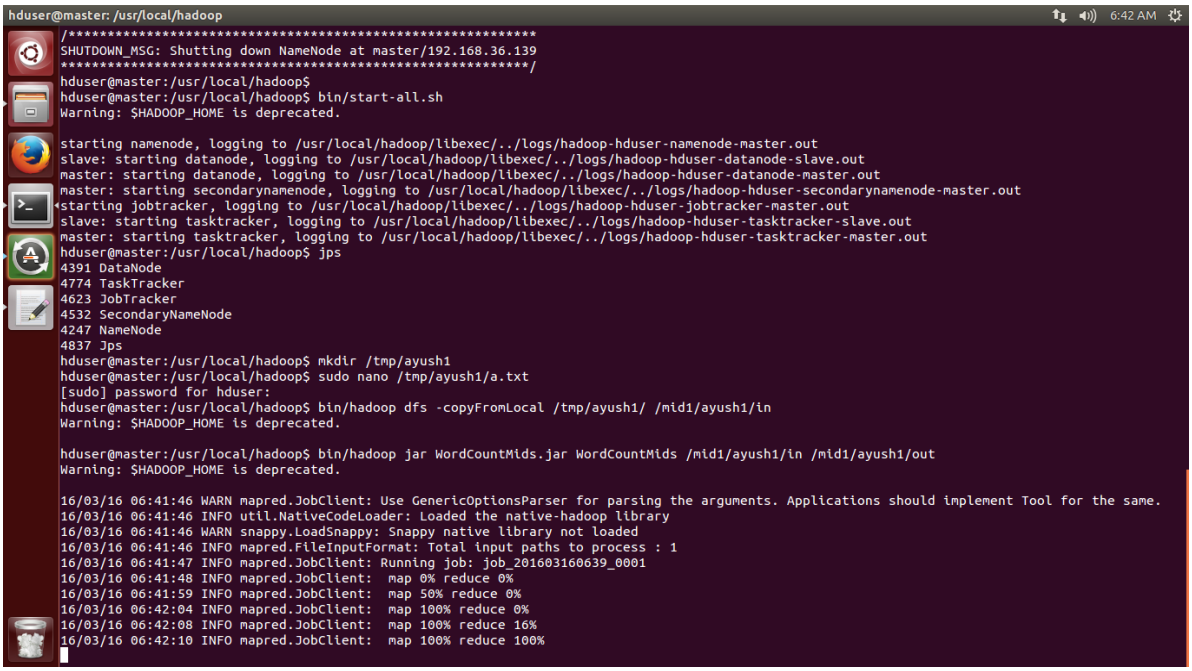
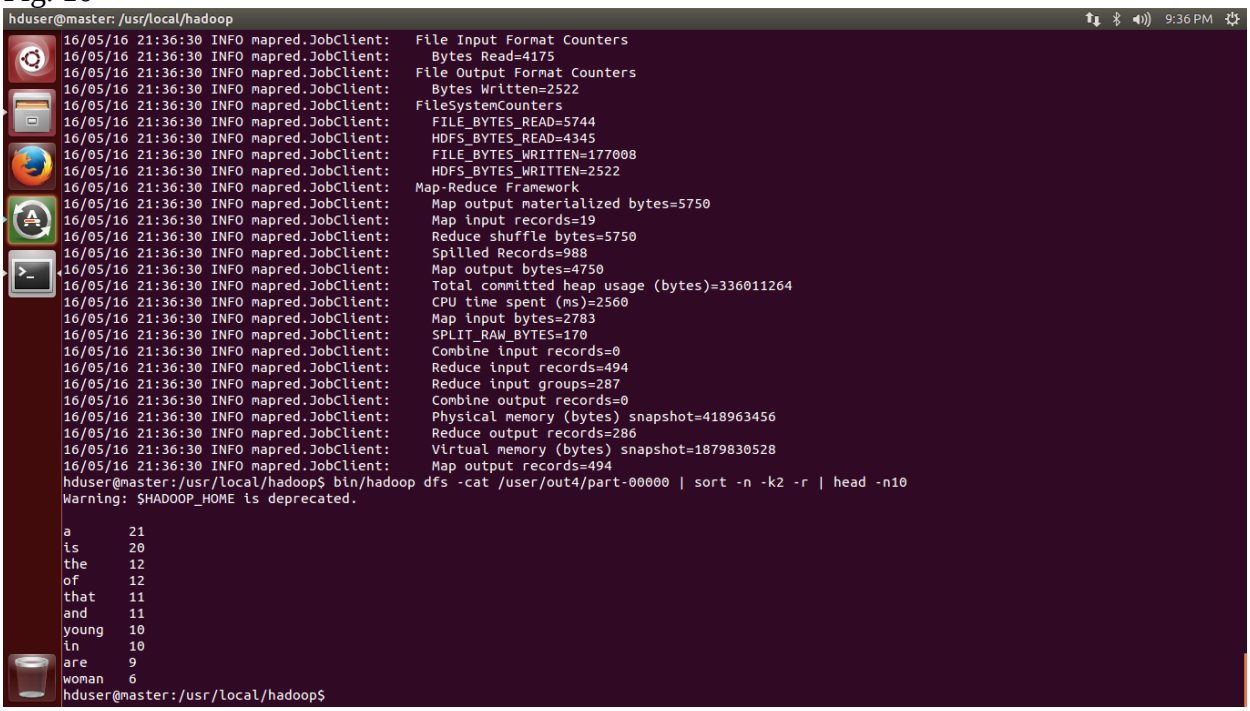


Fig. 9

Fig. 10



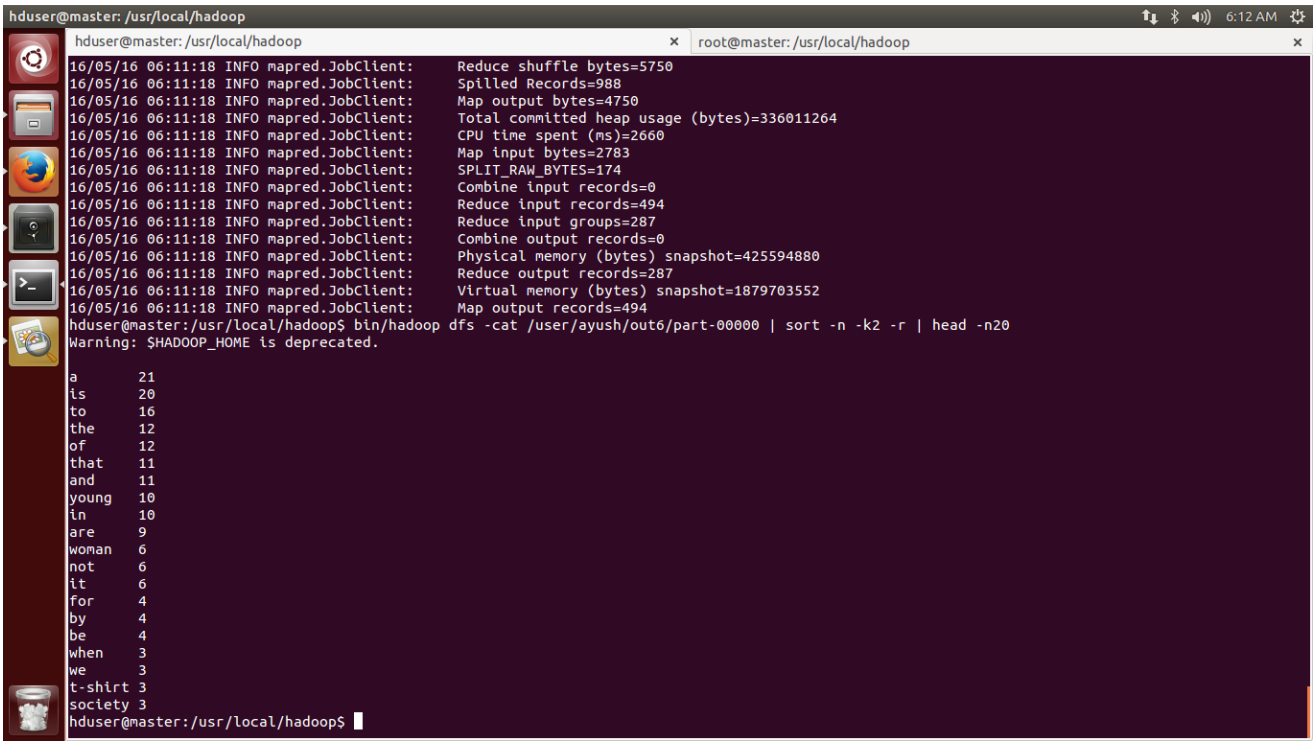


Fig. 11

7. LIMITATIONS AND FUTURE ENHANCEMENTS

Distributed Computing is one of the major implementation of the Hadoop Ecosystem. However despite of its overwhelming accomplishments there are certain key points regarding its efficiency which are lacking and have not been answered in years. Horizontal Scaling of the system proposes its own issues for the infrastructure and management. It also presents a lot of challenges regarding efficiency, but is better for enterprises with large data demands. This project can also be expanded with the vertical scaling of the infrastructure i.e., by increasing the number of cores, frequency of the CPU etc. higher efficiency can be achieved.

8. CONCLUSION

Distributed Computing is one of the major implementation. However despite of Hadoop's overwhelming accomplishments there are certain key points regarding its efficiency which are lacking and have not been answered in years.

Horizontal Scaling of the system proposes its own issues for the infrastructure, efficiency and management.
This project can also be expanded with the vertical scaling of the infrastructure i.e. by increasing the number of cores, frequency of the cpu etc. higher efficiency can be achieved.

9. REFERENCES

[1] Hadoop Tutorial. Ref: http://www.tutorialspoint.com/hadoop/, Cite: 2016.
[2] Jerry Zhao, Jelena Pjesivac-Grbovic, “MapReduce: The programming model and practice”, Research At Google, 2009.