Building and Evaluating an OpenStack Based Private Cloud for Studying Data Migration

by

Anurag Kumar Pandey

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
August 2019

© Copyright 2019 by Anurag Kumar Pandey

Thesis advisor
Dr. Ruppa K. Thulasiram

Author
Anurag Kumar Pandey

Building and Evaluating an OpenStack Based Private Cloud for Studying Data Migration

Abstract

Cloud computing is essentially a data center with an arrangement of multiple system resources connected to each other for better storage and management of data. This concept of Cloud computing has grown quickly and is now a technology that is widely used. Large data centers are created to serve large number of clients' workload from various walks of life. When a resource in a data center reaches its end-of-life, it may be realized that instead of investing in upgrading, replacing the resource or legacy applications running on that resource, it is time to possibly decommission such a resource and migrate workloads to the cloud and to other resources in the data center. Data migration between different Cloud resources and the servers of the same private Cloud is risky due to the possibility of data loss and also due to high temporal cost.

For my thesis, I have proposed a novel architecture and algorithm to study this phenomenon. I have used MapReduce data processing software framework within a private Cloud environment to determine the data loss. I have proposed metrics such as efficiency, speed, computation time and the cost of data migration and formulae for these metrics to test and evaluate my proposed framework.

The data migration technique I have introduced in my experiments shows better performance than the previously available work in cloud computing literature. I have done

experiments using image files like png, jpeg and tiff, audio files like wav, video files like mp4 and documents like xls and csv. Whatever small amount of data loss occurred during my experiments could be avoided with a stable network connectivity. I have used large heterogeneous files (text, images, audio and videos) for my algorithm to compute execution time and efficiency.

With the limited resources available for the current study, the size of files migrated were bounded due to system limitations and also data loss could not be avoided completely. By improving the configuration of the system architecture and physical servers I might expect to improve the performance further.

Contents

Abstract . . . ii
Table of Contents . . . v
List of Figures . . . vi
List of Tables . . . viii
Acknowledgments . . . x
Dedication . . . xi

1 Introduction **3**
1.1 Cloud Computing . . . 3
1.1.1 Cloud Classification . . . 5
1.2 Data Migration . . . 7
1.2.1 Types of Migration . . . 7
1.2.2 Lack of Migration Progress Management . . . 9
1.3 Hadoop Cluster . . . 10
1.3.1 MapReduce . . . 11
1.4 Thesis Organization . . . 13

2 Literature Review **14**
2.1 Hadoop MapReduce based approaches . . . 15
2.2 Problem Statement . . . 19

3 Private Cloud **20**
3.1 OpenStack . . . 20
3.1.1 OpenStack Components . . . 21
Compute (Nova) . . . 22
Image Service (Glance) . . . 23
Object Storage (Swift) . . . 23
Dashboard (Horizon) . . . 23
Identity Service (Keystone) . . . 24
Networking (Neutron) . . . 24
Block Storage (Cinder) . . . 25
3.2 Architecture of Cloud Deployed . . . 25

3.2.1 MAAS . . . 25
Minimum Requirements for MAAS . . . 26
3.2.2 JAAS . . . 27
Why use Juju? . . . 27
3.2.3 Building a Testbed . . . 27
Controller Node . . . 29
Compute Node . . . 29
Block Storage . . . 29
Object Storage . . . 29
Networking . . . 30
3.3 Hadoop MapReduce Implementation . . . 30

4 Solution Methodology **31**
4.1 Introduction . . . 31
4.2 Why MapReduce migration is preferred? . . . 32
4.3 Matlab-MapReduce . . . 33

5 Experiments, Results and Evaluation **38**
5.1 Experiment . . . 38
5.1.1 Flow of Implementation Process . . . 39
Strategy Development . . . 39
Assessment and Analysis . . . 41
Data Preparation . . . 41
Validation and Staging . . . 42
5.2 Results . . . 43
5.2.1 Data Migration for audio and video files . . . 43
5.2.2 Data Migration for image files . . . 44
5.2.3 Data Migration for documents . . . 45
5.3 Evaluation . . . 47

6 Conclusion **53**

7 Key Contributions **55**

A Source Code **56**
A.1 Install MAAS [1] . . . 56
A.2 Install JAAS [2] . . . 59
A.3 Install Openstack [3] . . . 60
A.4 Install Hadoop MapReduce [4] . . . 63

Bibliography **75**

List of Figures

1.1 Cloud Deployment Model 4
1.2 Basic functionality of MapReduce[5] 11

3.1 OpenStack Components Installed 22
3.2 Private Cloud Architecture 26
3.3 Use of Juju [2] 28

4.1 Normal Migration vs MapReduce Migration 32
4.2 MapReduce Algorithm Phases [6] 34

5.1 Work Flow Schema 40
5.2 Loading Audio Video Files 43
5.3 MapReduce for Audio Video Files 44
5.4 Loading Image Files 45
5.5 MapReduce for Image Files 45
5.6 Loading Document Files 46
5.7 MapReduce for Document Files 47
5.8 Execution Cost 49
5.9 Data Loss 49
5.10 Execution Cost 50
5.11 Data Loss 50
5.12 Execution Cost 51
5.13 Data Loss 51
5.14 Efficiency Analysis 52

A.1 MAAS Login 57
A.2 DHCP enabled 58
A.3 Images used in Deployment 58
A.4 List of Working Machines 59
A.5 After Installation 60
A.6 Hadoop Installation – Setting Environment Variable 64
A.7 Hadoop Installation – Configuring core-site.xml 64

A.8 Hadoop Installation – Configuring hdfs-site.xml 65
A.9 Hadoop Installation – Configuring mapred-site.xml 65
A.10 Hadoop Installation – Configuring yarn-site.xml 66
A.11 Hadoop Installation – Configuring hadoop-env.sh 66

List of Tables

5.1 Audio and Video Migration 49
5.2 Image Migration 50
5.3 Text Migration) 51

List of Algorithms

1 MAPRDUCE DATA MIGRATION 36

Acknowledgments

There is no one I am indebted to more than my advisor Dr. Ruppa Thulasiram, for guiding me, encouraging me, extending his intelligence, patience to my work's completion, and for being there always at the time of crisis where my composure was weak.

I am very grateful to my committee members, Dr. Noman Mohammed and Dr. Saumen Mandal for their valuable comments, time and guidance.

I am extremely thankful to my parents Arun Kumar Pandey and Sudha Pandey for having an utmost belief in my commitment and efforts. Special thanks to Ajit Pandey and Rani Pandey for being there with me in every phase of life. They all have encouraged me since the beginning of my education to put my knowledge towards meaningful work. It would have been impossible without them to accomplish this work. Many thanks to my sister Ankita Pandey for being strict but at the same time being my best support to encourage me to keep going.

I would like to thank Sheenam for pushing me to pursue this work and motivating me to make it happen. My special thanks to my lab-mates Navdeep, Manmohit, Moosa and Muskan for their help in my stressful times and for all cheerful memories that we celebrated in our lab. Thanks to all the students of computer science department and technical staff for their help and encouragement in my work. I would also like to thank Ankit Sharma and Ritu Patwal Sharma who were also responsible for my 50% of happiness over the past two years.

Last but not the least, I would like to thank that extreme power of God which infused me with self motivation to complete this work.

This thesis is dedicated to my late grandparents Asth Bhuja Nath Pandey and Sarojini Pandey. Although, they are not with me, but I can feel their blessing with me always.

"Man needs his difficulties because they are necessary to enjoy success." - Dr. A.P.J. Abdul Kalam

Glossary

Cheetah A high performance custom data warehouse built on top of MapReduce

Data Node the HDFS node, which is used for storing the data.

Hadoop Cluster It is a type of cluster used for storing and analyzing the large amount of unstructured data.

HDFSIt is Hadoop Distributed File System used for storing data for Hadoop applications

Hive A data warehouse project built on top of Hadoop having a SQL like interface to query various databases in a Hadoop file systems.

Hybrid Cloud The cloud computing environment which is a combination of private and public cloud.

Hypervisor A software to run virtual machines

JaaS Juju as a service is used to configure, scale and deploy the softwares related to cloud, big data etc.

KVM Kernel Virtual Machine or KVM is an open source hypervisor

MaaS Metal-as-a-service used for server provisioning

Mapper The first phase which processes the input record and generates intermediate key value pairs.

MapReduce A programming framework used for processing huge data sets with parallel and distributed algorithm on a hadoop cluster.

Name Node The HDFS node used to store the metadata, also called as master node.

OpenStack The open-source software platform used for cloud computing, which is deployed as infrastructure-as-a-service, to provide virtual servers and other resources to the customers.

Private Cloud The type of cloud which is similar to public cloud and providing services

through a proprietary architecture

Public Cloud The computing services proivded over the internet by third-party providers.

Reducer Tha last phase in which intermediate key value pars are reduced to a smaller set of values based on common keys.

Virtual machines The simulation of computer system providing the functionaity similar to a physical computer through implementation of software, hardware or their combination.

Virtualization It is the process of creating virtual version of a system which can be hardware platform, storage systems and computer network resources

Chapter 1

Introduction

The usage of solid state drives have increased over the years for storing data for commercial and domestic usage. This has brought in a high demand for multiple tier data storage system and making efficient usage of these expensive drives. It is known that the Solid state drives can be used efficiently in the organizations since it can bring in lots of performance benefits in their applications that need high processing capacities, batch processing, transaction processing, analysis of query or decision support [7]. Cloud computing is the functionality where the computing resources are provided as a service through the internet or a network.

1.1 Cloud Computing

Cloud computing is an environment that works in a network and makes way for resource sharing irrespective of the location of the user. The data is stored in a place and may have multiple cached files and copies in multiple locations [8]. Cloud computing includes online commercial servers, data centres, private servers, etc. The cloud computing architecture

consists of front-end devices, back-end platforms, cloud based delivery and networking. Front end devices are those that are used by clients and customers to access the files using a browser or other applications. Back end platforms are computers, virtual machines, servers, etc that store the data and are the backbone of cloud computing [9]. Virtualization is the key to cloud computing, since it allows us to create multiple simulated environments or dedicated resources from a single, physical hardware system. A hypervisor is a software that runs on this hardware and enables splitting a single physical hardware system into separate, distinct, and secure environments known as virtual machines (VMs). Main purpose of Cloud computing is the provisioning of various services like IaaS (Infrastructure as a Service), SaaS (Software as a Service), PaaS (Platform as a Service) or XaaS (Anything as a Service) offered to an individual or an organization. The above service classification is provided by deploying cloud as public, private, hybrid or community cloud based on the service model deployed [8].

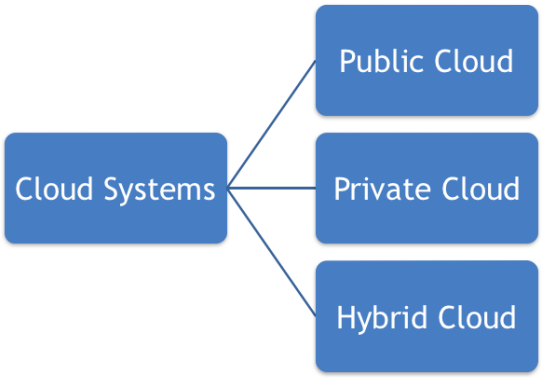


Figure 1.1: Cloud Deployment Model

1.1.1 Cloud Classification

Public clouds can be used by any individual or organization as a pay-per-use service. Some services are also available at free of cost to some extent. The customers have a choice of keeping the files public or keep them private. These cloud systems can be accessed through the internet by users and hence has a high accessibility. The infrastructure is owned and operated by service providers like Google, Microsoft and others.

Private clouds are used by enterprises and organizations and the servers are exclusively dedicated to that organization. The data cannot be shared with any other individual or organization. The servers may either be managed by the same company or by a third party.

The other type of cloud system is the hybrid cloud system where multiple cloud systems are used together and may offer the advantage of public and private cloud systems. For sensitive applications, organization may use private clouds whereas they may use a public cloud for non-sensitive applications. There are multiple benefits of cloud [10] such as

- Elasticity: Resources are scaled as per demand in a dynamic way.
- Cost Saving: Reduces capital infrastructure.
- Accessibility: Users can access data from any location and any time.
- Reliability: Cloud storage is generally reliable with reduced risk of data loss.

The public cloud, for example, represents a set of standard resources of varying types that can be combined to build applications [11]. In a public cloud, the services are offered to clients for different purposes suc as storage of files, payroll etc. With a public cloud, an organization does not need to spend time and money for maintaining infrastructure. Instead, they can pay a nominal fee to the cloud service provider (CSP) and focus on their core business.

Community clouds are different because it is an infrastructure that is shared by several organizations with similar interests. Private clouds are intended to be used by a single organization and is usually managed by the internal administrators for the use by employees of the organization.

Currently, cloud computing plays a major role in technology and undertakes huge changes. The technology development in cloud computing through the years has led to multiple innovations. It enables users to get computing resources/services over the internet irrespective of the location from a remote network of servers [12]. Presently, Cloud computing refers to multiple types of services and applications used over the internet. There is no need of any special tools for using the resources in the cloud computing. In social network applications, cloud based functionalities enables instant messaging, video communication, etc.

For my thesis, I plan to setup a private cloud. The significance of a private cloud over the public cloud is that important data can be stored with the minimum fear of it getting leaked over the internet. A private cloud can be maintained anytime if organization(s) require it without depending on the cloud providers. However, there is a disadvantage in using a private cloud. If even a small portion of a server gets corrupted, it may lead to the data loss [13].

To address this problem, additional servers need to be installed in the private cloud to keep multiple copies of the same data, which can be used for data recovery. The additional server should function continuously without any hindrance and should always contain the up to date copies of the files present in the original server. Hence, any file that is added to the original server in the private cloud should be copied to additional servers. During server maintenance, the data may have to be migrated to different sets of servers. The data should also be deleted from the server initiating the migration because the data may be sensitive

and should be avoided falling into wrong hands within the organization.

1.2 Data Migration

Data migration refers to the process of moving data, applications or other business elements from an organization's onsite computers to the cloud, or moving them from one cloud environment to another. There are different types of data migrations in an organization through cloud computing. One of the most common types of data transfer is the transfer of data and applications from an on-premise server to the public cloud. Other types of transfer is the cloud migration is between two servers of the same organization located in different locations. They can be located even in different continents but are transferred over the internet [14]. The transfer may also be performed between two different platforms over the cloud and this is known as cloud to cloud migration. There is also a type, where the migration takes place from a cloud server to a local server or data centre.

1.2.1 Types of Migration

There are various types of data migration that takes place over a cloud system. They are classified depending on the type of data transferred over the network.

Storage migration is the process of migrating data from existing drives and locations into state-of-the-art drives elsewhere. This will provide more significant and faster performance, providing more scaling with more cost effectiveness [15]. This requires data management characteristics like cloning, backup and disaster recovery, snapshots, etc. The process takes time to perform validation and optimization of data and to identify outdated or corrupted data. It also involves migrating blocks of files and storage from a system storage

to another irrespective of whether it is drive, disk or in the cloud [16]. There are multiple storage migration techniques and tools which helps in smoother transition of the process. It also increases the chance of modernizing the storages and stop inefficient drives.

Database migration is another type of migration. It is the process of migrating data (files with different format like text, audio, video etc) from one system to another. This is performed at times where there may be a necessity to shift from one vendors to another, upgrading the software or shift the database to the cloud [17]. In this type of migration, the basic data may change, that may affect the application layer when there is a shift in protocols or data. This technique deals with modifying the data without altering the structure. A few key tasks include calculating the size of database for determining the amount of storage required, testing applications and making sure that the data will be confidential. There may be compatibility problems that may occur at the time of migration process, hence it is necessary to test the process first [13].

Application migration is the process of migrating an application from an environment or storage to another. This may include migrating the whole application or a part of it from a storage to cloud or between different clouds [18]. It may also include migrating the application's main data to a newer form of application that is used by another provider. It is mostly used when an organization switches to another vendor platform or application. There are complexities associated with the process since the applications interact with other applications, and every migration has its own data model. Usually, applications are not migrated [19] because the operating systems, tools in the management and configuration in the virtual machines can change between different environments. Migration of applications may need other middle ware tools to bridge the gap in technology.

There are challenges associated with migration in the cloud. A lot of enterprises do not

have the technical experience required for transferring between cloud systems or between servers over the cloud, which causes lots of disadvantages. A solution would be to outsource the work to other companies which may lead to data theft or loss. Protection of sensitive data is important.

1.2.2 Lack of Migration Progress Management

Live data migration in cloud computing has uncovered major weaknesses in existing solutions that lacks progress management in the migration, the ability to predict and control the time of migration [20]. With no capability to control and predict the migration time control, the management tasks will not be able to attain the expected performance [21].

If a system administrator requires to take down a physical machine for maintenance or for migrating the contents of the system to the cloud, the time management cannot be guaranteed and may disrupt the process and may lead to disruption of productive time in the business. The failure and prediction systems that are applied may not detect the abnormal activities in the servers during the data migration. The migration is also be performed to balance the load [22]. These scenarios reveal the weaknesses in current live migration. Hence, the system administrator has to analyze and predict the time taken to complete the migration and ensure that the migration process is managed efficiently.

In general, data migration seems simple and hence, managers, do not pay much attention on it, care less about the migration and maintenance of servers. However, there are huge implications [23]. The bandwidth is one of the major implications among those. The authors [23] also discuss reducing the cost of processing the geographically distributed big data.

The data that is transferred between the servers is usually huge and entire data may not reach the other server. A small corrupted block in the server (original/additional) may lead to

a big failure [24]. Addressing the problem of migration is complex and a separate industry has been booming and growing at a rapid pace. According to the reports by Thalheim et al. [25], in the past only 16% of the data migration projects had been completed successfully without any error. These authors have also mentioned that since the migration takes significant time, only 64% of the data migration project had a timely delivery. Data migration being a complex function, the data has to be optimized to make the process simpler. Hence, I use MapReduce, a model that optimizes the data, in my thesis. It is a programming model that processes big data sets.

1.3 Hadoop Cluster

A Hadoop cluster is a hardware cluster used to facilitate utilization of open-source Hadoop technology for data handling [26]. The cluster consists of a group of nodes, which are processes running on either a physical machine or virtual machine. The Hadoop cluster works in coordination with distributed storage HDFS - Hadoop Distributed File System - and distributed processing framework (MapReduce) to deal with unstructured data.

The Hadoop cluster works on a master-slave model [26]. A node called ***NameNode*** [1] works as the Hadoop master. This node communicates with various ***DataNode*** [2]nodes in the cluster to support concurrent tasks. Hadoop cluster also uses other Apache open-source technologies like Apache MapReduce and Apache Yarn. Hadoop clusters are used for all sorts of predictive analytics, product and service development, customer relationship management and much more.

[1]It is a master server for managing the file system namespace and controlling the access to files by clients.
[2]This node stores the actual data.

1.3.1 MapReduce

According to Apache Hadoop project, Hadoop MapReduce is a software framework for distributed processing of large data sets on compute clusters of commodity hardware [27]. The framework takes care of scheduling tasks, monitoring them and re-executing any unsuccessful tasks.

According to the Apache Software Foundation [27], the primary objective of MapReduce is to split the input data set into independent chunks that are processed in a completely parallel manner.

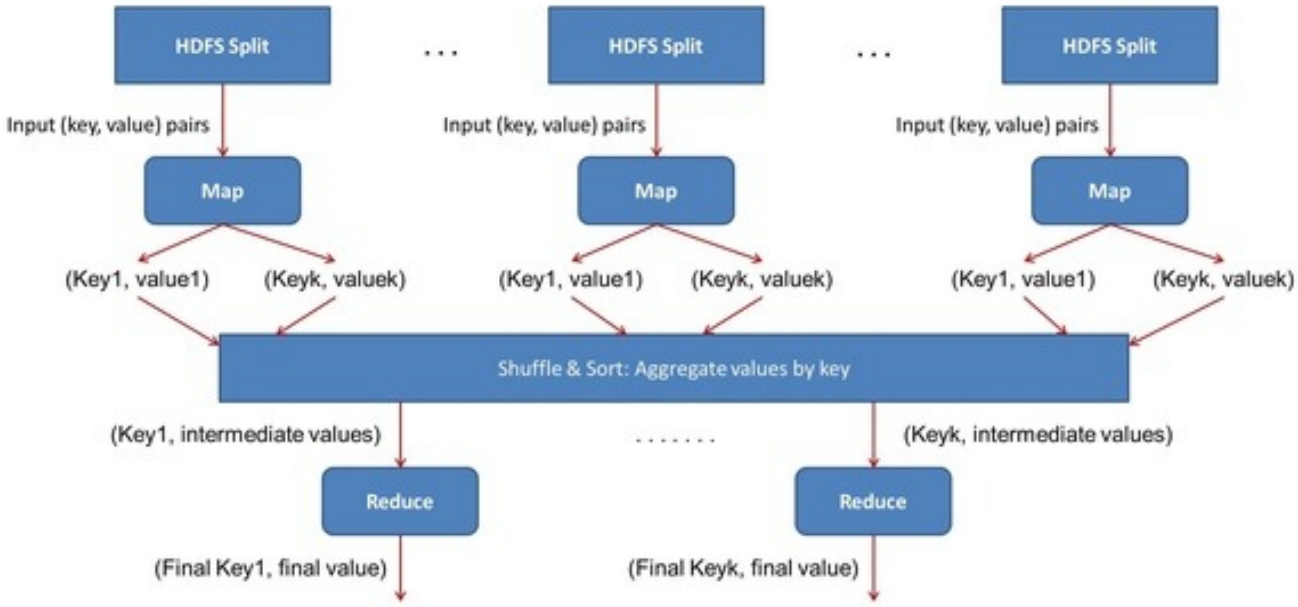


Figure 1.2: Basic functionality of MapReduce[5]

From Figure 1.2, it can be seen that MapReduce contains two main functions known as Map and Reduce. The Map function converts the input data into intermediate KVP (Key / Value Pairs) format by grouping the data. A KVP contains data of two linked items which is a Key and a Value. The Key assigns a unique identity for the group of data, whereas the Value contains a pointer that points to the location of the data. The Map now has data in

a structured manner along with the Key and Value assigned to it. This output is used as an input to the Reduce task. In the reduce task, it obtains the structured data i.e. intermediate KVP's and converts them into smaller structures. The KVP data for each group is stored in the Hadoop distributed file system(HDFS).

The Apache Hadoop MapReduce is a framework that allows coders to create applications that can process large amount of processing parallelly in multiple nodes. It is an open source modified version of MapReduce based on Google MapReduce and GFS (Google File System) [28] even though both of these are not similar. A MapReduce functionality is a type of work that consists of input data, MapReduce functionalities and the details of the configuration. Hadoop works by dividing the job into tasks as map tasks or reduce tasks. Two different types of nodes that control the job execution are a tracker node and multiple task trackers. These task trackers run the tasks allocated to them and send reports to the Job Tracker since it preserves the whole progress of every task.

The input to the MapReduce task is divided into fixed size pieces known as chunks or splits (64 MB chunks) [4]. It assigns a map task for each split when functions related to the users are recorded for every split. Having lots of split means that the time taken to process every split is smaller while comparing to the time taken to process whole input at once. Hence, if the splits are processed in parallel, it will be faster when the splits are small, since system can perform the processing more quickly. Even though the machines are identical, failed processes or other tasks that run simultaneously makes load balancing desirable, and the nature of the load balancing increases as the chunks become more fine grained [29]. On the contrary, if the chunks are too small, the overhead of dealing with the chunks and of map tasks creation starts to dominate the execution time of the overall tasks. For most tasks, a good chunk size will in general be the measure of a HDFS block, which is 64 MB as a

standard.

Map jobs compile their output to the localized disk, not to HDFS. This is on the grounds that the output of the map is the intermediate output. It is handled by reduce function to deliver the final output and once the activity is finished the output from the map can be disposed off. Hence, storing the map output in HDFS with copies would be unnecessary. For each HDFS block from the output of the reduce, the primary copy is saved on a localized node, with different copies being saved on other data nodes. In this way, writing the output of the reduce task consumes bandwidth of the network.

The number of reduce jobs is not controlled by the input size, but it is mentioned in the code. When there are more than one reducers, mapper partition's the output, each making one section for each reduce task. There can be multiple keys per partition, however the records for any given key are all in a single partition. The partitioning may be controlled by a partitioning function as defined by the user, however the default partitioner that stores keys by utilizing a hash function works extremely well.

1.4 Thesis Organization

A literature review along with problem statement is presented in Chapter 2. The cloud architecture with OpenStack services, MaaS and Juju is explained in Chapter 3. In Chapter 4, I have explained the data migration with the help of MapReduce based algorithm that I have designed. In order to evaluate the architecture model and algorithm, I came up with new metrics which is introduced and discussed in Chapter 5. I conclude my thesis study in Chapter 6 by summarizing my work and listed some possible direction for future work in Chapter 7.

Chapter 2

Literature Review

In this section, I discuss some of the approaches related to the data migration. The literature consist of various approaches such as DCTCP [30], D2TCP [31] and D3 [32], that are used for minimizing the cost of data movement inside the data center. These approaches focus on data transfer within a single geographical location. The paper by Cho and Gupta [33] presents a system named Pandora, that gives optimal cost solution for transferring a significant amount of data from one data center to another data center located around the globe. This approach finds the optimal cost using the physical shipment of disks as well as online data transfer. The problem with this approach is the conventional physical shipment is not an efficient solution to transfer large volumes of data.

Various technologies like elastic optical networks and DC networks have been discussed by Lu et al. [34] for migrating data efficiently and creating backups for use in big data. The authors have described the impacts of applications of big data on the existing network. After this, authors have made a model for the data migration over the network. They have proposed efficient algorithms with respect to BL-Anycast-KSP-Single-DC algorithm.

A joint resource defragmentation has been discussed in [34] for improving the performance of the network and a mutual backup model has also been proposed for better data backup. However, the efficiency of data migration is very less for elastic optical inter-DC networks and it is difficult to control and manage the network.

2.1 Hadoop MapReduce based approaches

Efficient migration of data has been studied under different contexts. I discuss in this section briefly about Hadoop, geo distributed data centers and energy efficiency. Liu et al. [35] used Hadoop clusters to implement the MapReduce for cloud computing applications. According to authors, when the data size grows, the performance of MapReduce is reduced. They introduced a performance rating scheme to analyze this phenomenon. Principle Component Analysis method was used to fill out the critical Hadoop configuration metrics that strongly impact the workload performance from excessive configuration items [35].

HadoopDB: There has been lots of research that correlates related data into similar nodes. HadoopDB [36] saves information in a localized database management system and hence interrupts the dynamic scheduling and fault tolerance of Hadoop. According to Dittrich et al. [37], the two input files are grouped in Hadoop [37] by creating a unique file with the specifications of a Trojan Index. Trojan Index is a solution to integrate indexing capability into Hadoop to provide index, that can help in executing the MapReduce jobs. Despite the fact that this methodology does not require an alteration of Hadoop, it is a static solution that expects users to rearrange their input data. Newer benchmarks have distinguished an gap in the performance among Hadoop and parallel databases [38; 39]. There has been considerable interest in advancing Hadoop with methods from other databases, while

retaining the flexibility of Hadoop. A serious analytical benchmark study of different parts of the process pipeline of Hadoop was been led by Jiang et al. [40]. It was discovered that indexing the map significantly enhanced Hadoop's execution.

Hadoop is not very intrusive, like indices are saved as "Trojans" inside HDFS chunks and splits; there is no required change to Hadoop itself. In contrast, data placement in Hadoop is done at load time. Also, colocation [41] in case of Hadoop can help to improve efficiency in joins and operations. *"Cheetah"* and *"Hive"* are two information warehousing arrangements on Hadoop similar to parallel databases.

GridBatch [42] is another expansion to Hadoop with a few new administrators, and in addition another record type, which is divided by a partitioning function as defined by the user. It enables applications to determine documents that should be co-put too. Their answer intermixes the partitions at the record framework level, though this strategy decouples them with the goal that diverse applications can utilize distinctive strategies to characterize related documents. In further developed apportioning highlights of parallel database frameworks [43]. Eg. IBM DB2, TeraData, and Aster Data tables are co-divided, and the inquiry analyser abuses this reality to create proficient question designs. This methodology adjusts these plans to the MapReduce framework, while holding Hadoop's dynamicity and adaptability. To accomplish this, proposed approach varies from parallel databases in that proposed framework performs co-position at the record framework level and in a best-exertion way: When constraints in the space or failures prevent co-situation, high accessibility and adaptation to internal failure are given higher need.

Programming Models: There have been multiple programming models that has provided restricted programming and utilizes restrictions for parallel computation automatically. An associated functionality can be used for the prefixes using parallel prefix computations

[44; 45]. These models can be simplified using MapReduce based on real world computations. An implementation that is tolerant on fault that scales to thousands of processors has been provided. Conversely, a large portion of the parallel preparing frameworks have just been executed on little scales and leave the points of interest of taking care of machine failures to the developer. Higher levels of abstraction is provided by bulk synchronous programming [46] and some MPI primitives [47] that make it easier for programmers to code simultaneous programs. A prime distinction between these frameworks and MapReduce is that MapReduce misuses a limited programming model to use the client program parallelly and to give straightforward adaptation of the fault tolerance. The locality optimisation draws its motivation from techniques such as active disks [48; 49], where computation is pushed onto processing elements that are close to local disks, to reduce the amount of data sent across I/O subsystems or the network.

Scheduling: Commodity processors are utilized where a small amount of disks are directly associated instead of running directly on disk controller processors, but the general methodology is similar. The backup task techniques are like the eager scheduling techniques used in the Charlotte System [50]. The main weakness of a simple enthusiastic scheduling is that if a given task causes failures repeatedly, the whole processing fails to complete. The MapReduce execution depends on an in-house cluster management framework which is responsible for distributing and running user tasks on a large number of common systems. Even though it is not the focus of this work, the cluster management technique is similar to other techniques like *Condor* [51]. The data sorting which is a part of the MapReduce library is similar to the operation of *Now-Sort* [52] . The source machines segment the information to be arranged and sends it to the reduce tasks. The reduce task arranges the information in a local storage. It is known that *Now-Sort* is not very user friendly and cannot be defined

by the user.

BAD-FS is an altogether different programming model from MapReduce that has been proposed by Bent et al. [53] for targeting the tasks across a wide area network. However, there are two main similarities: (1) Both frameworks utilize excess execution to recuperate from data losses that is caused by failures; (2) Both utilize a similar type of planning to diminish the amount of information sent through dense networks. TACC framework has been designed for simplifying the creation of services within a network is given by Fox et al.[54]. Like MapReduce, it depends on re-execution as a system for actualizing adaptation to internal failure.

Geo-distributed cloud services contain many data centres spread across different locations. They can provide larger capacities to the end users and they are mainly used for social media applications [55]. According to authors, there are challenges like storing and migrating the data over long distances. An efficient framework has been proposed by Microsoft Team [55], which provides a solution to data placement. This solution helps to reduce the data movement between geo-distributed data centers. The effectiveness of the proposed framework has been verified by comparing to offline transfers. However, in this model [55], storage limits are not considered for every cloud location, and only the predicted data is sent.

An energy efficient tool has been developed in Li et al. [56] for migrating data in a virtual machine. It is an emulator where it provides functionality of an actual computer. A double threshold model with multiple resource utilization has been designed to migrate in the virtual machine [56]. The proposed algorithm by Li et al. [56] has shown better energy efficiency in cloud data center. To transfer data over the cloud efficiently, a cost-effective data migration technique has been proposed by Zhang et al. [15] using a framework

similar to MapReduce. Online lazy migration (OLM) and randomized fixed horizon control (RFHC) algorithms have been proposed by these authors to transfer the data efficiently. The performance of the online algorithms has been shown to improve when compared to optimal offline algorithm such as Smith Waterman alignment algorithm [57].

All these approaches focus on various aspects and issues of data migration. The major problem with these approaches is not having a generic solution to data migration problem. Each approach is best suited for a specific scenario or a particular data set. There is a need for building a framework that can efficiently migrate the data and calculate the data loss as well.

2.2 Problem Statement

There might be data loses happening during migration. The few data migration approaches discussed above, do not compute the data loss accurately or may not even consider such loss. These existing approaches migrate data without any optimizing tools like MapReduce. This makes it difficult to compute the data loss during the transfer. Hence, there is a need for creating a novel framework that can efficiently migrate data without any data loss or minimal data loss. I am building such a framework that can efficiently migrate the data using MapReduce and also help in computing the data loss, if any.

The overall objective of this research is to:

- To migrate the data efficiently with minimum or no loss of data.
- To reduce the time taken to migrate the data between the servers over the cloud.

Chapter 3

Private Cloud

3.1 OpenStack

Originally OpenStack was developed as a collaborative project between NASA and Rackspace [58]. In 2010, OpenStack was released as Austin. Austin had very limited features, for example, initially it only supported object storage. Realizing the potential for virtualization market, companies started contributing to OpenStack project.

The OpenStack project is an open source cloud computing platform for all types of clouds [59]. The purpose of using is that it is simple to implement, highly scalable, and feature rich. It is one of the widely used cloud computing platforms among developers and cloud computing technologists.

OpenStack basically provide Infrastructure-as-a-Service (IaaS) solution through a group of associated services [59]. Each service provides an application programming interface (API) to facilitates this integration. According to the needs, one can install the required services. OpenStack has gained a lot of popularity due to its flexibility and ability to provide a virtu-

alized infrastructure as it provides multiple hypervisors such as KVM, Qemu and Hyperv.

KVM (Kernel Virtual Machine) is a Linux kernel module that allows a user space program to utilize the hardware virtualization features of various processors [60]. Today, it supports recent Intel and AMD processors (x86 and x86/64). QEMU can make use of KVM when running a target architecture that is the same as the host architecture [60].

OpenStack comes with practically all of the benefits of cloud computing. OpenStack's orchestration and self-service capabilities offer developers and IT staff with faster and better access to IT resources. Faster deployment of IT resources also means end users and business units no longer have to wait days or weeks to start using the network services and applications they need. OpenStack enables the construction of private clouds as well as it can help in regulatory compliance endeavors. If your cloud is in your own data center, you will have more control of access privileges, security measures, and security policies. Another reason Openstack is advantageous is that as an open source project it does not require any subscription or an annual fee to use.

An important feature that makes OpenStack advantageous compared to CloudStack [61], Nebula [62] and others is that OpenStack supports small scale deployment. OpenStack can be tested using the development version called as Devstack [63] that supports deployment onto a single, local machine for rapid application development and testing with minimal required effort in setting up. For these reasons, I selected OpenStack in my thesis work as the Cloud computing software to build a private Cloud using Qemu-KVM.

3.1.1 OpenStack Components

Several components contribute in building an Openstack based Cloud as shown in Figure 3.1. As OpenStack is an open-source software, it is made up of several other components.

The OpenStack community has recognized these components as the core components. I describe these components briefly in this subsection.

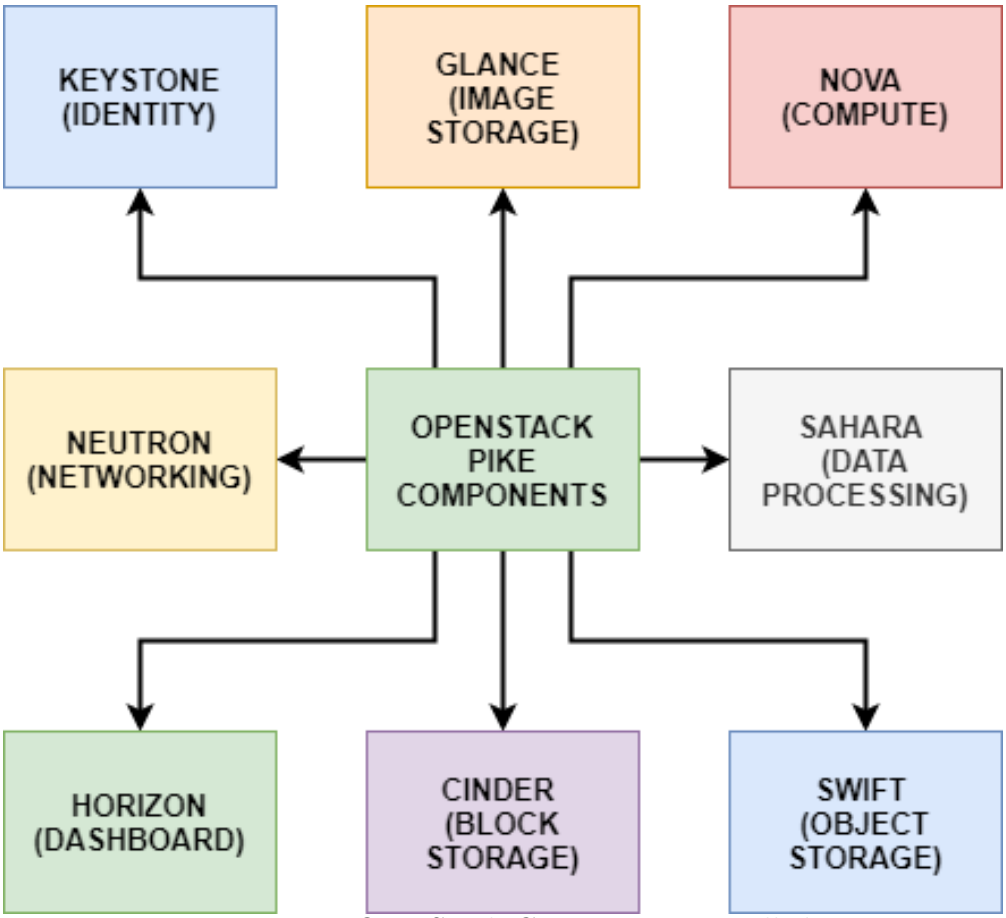


Figure 3.1: OpenStack Components Installed

Compute (Nova)

This OpenStack component works as a cloud computing controller. It is used to manage pools of computer resources working in a virtualization environment and having high computing configurations. Very low hardware requirement and no proprietary software, Nova''s architecture provides high flexibility to design the cloud. Nova also has the ability to integrate the legacy systems and third-party products.

Nova compute can be deployed using different types of hypervisor softwares such as KVM, VMware, LXC etc. It also manages virtual machines as well as instances that handle various computing tasks.

Image Service (Glance)

OpenStack image service offers to discover, register, and restore virtual machine images. Glance works on client-server architecture and delivers a user REST API, which allows querying of virtual machine image metadata as well as retrieval of the actual image. Glance uses the stored images as templates while deploying new virtual machine instances.

OpenStack Glance supports different format of virtual machine images such as Raw, VirtualBox (VDI), VMWare (VMDK, OVF), Hyper-V (VHD), and Qemu/KVM (qcow2).

Object Storage (Swift)

OpenStack Swift creates scalable data storage to store petabytes of accessible data. The stored data in swift can be leveraged, retrieved and updated. Swift has a distributed architecture with no central point of control. It provides greater redundancy, scalability, and performance.

Swift is a extremely available, shared, eventually consistent object store. Data replication and distribution over various devices is an important feature of swift, which makes it ideal for cost-effective, scale-out storage.

Dashboard (Horizon)

Horizon is the only graphical interface to automate cloud-based resources. It is the authorized implementation of OpenStack's Dashboard. It supports with third-party services

such as monitoring, billing, and other management tools to service providers and other commercial vendors.

Identity Service (Keystone)

OpenStack Identity Service provides a central list of users, mapped against all the OpenStack services, which they can access. It integrates with existing backend services such as LDAP while acting as a common authentication system across the cloud computing system.

Keystone supports various forms of authentication like standard username password credentials, AWS-style (Amazon Web Services) logins and token-based systems. Additionally, the catalog provides an endpoint registry with a queryable list of the services deployed in an OpenStack cloud.

Networking (Neutron)

Neutron provides networking capability like managing networks and IP addresses for OpenStack. It ensures that the network is not a limiting factor in a cloud deployment and offers users with self-service ability over network configurations. OpenStack networking allows users to create their own networks and connect devices and servers to one or more networks. Developers can use SDN technology to support great levels of multi-tenancy and massive scale.

Neutron also offers an extension framework, which supports deploying and managing of other network services such as virtual private networks (VPN), firewalls, load balancing, and intrusion detection system (IDS).

Block Storage (Cinder)

OpenStack Cinder delivers determined block-level storage devices for application with OpenStack compute instances. A cloud user can manage their storage needs by integrating block storage volumes with Dashboard and Nova.

Cinder can use storage platforms such as Linux server, EMC (ScaleIO, VMAX, and VNX), Ceph, Coraid, CloudByte, IBM, Hitachi data systems, SAN volume controller, etc. It is appropriate for expandable file systems and database storage.

3.2 Architecture of Cloud Deployed

There are various ways of deploying the cloud. I have deployed the minimal version of Openstack Pike using Metal as a Service (MAAS) and Juju as a service (JAAS) as shown in Figure 3.2.

3.2.1 MAAS

Metal As A Service or MAAS, treats physical servers like virtual machines in the cloud. It turns bare metal into an elastic cloud-like resource so we don't have to manage each server individually.

Machines can be quickly provisioned using MAAS. MAAS can also destroy instances easily as similar to instances in a public cloud like Amazon AWS, Google GCE, and Microsoft Azure, among others.

MAAS can act as a standalone PXE service. It can also be integrated with other technologies. It is basically designed to integrate well with Juju, the service and model management service. It's a perfect combinations as MAAS manages the machines and Juju manages the

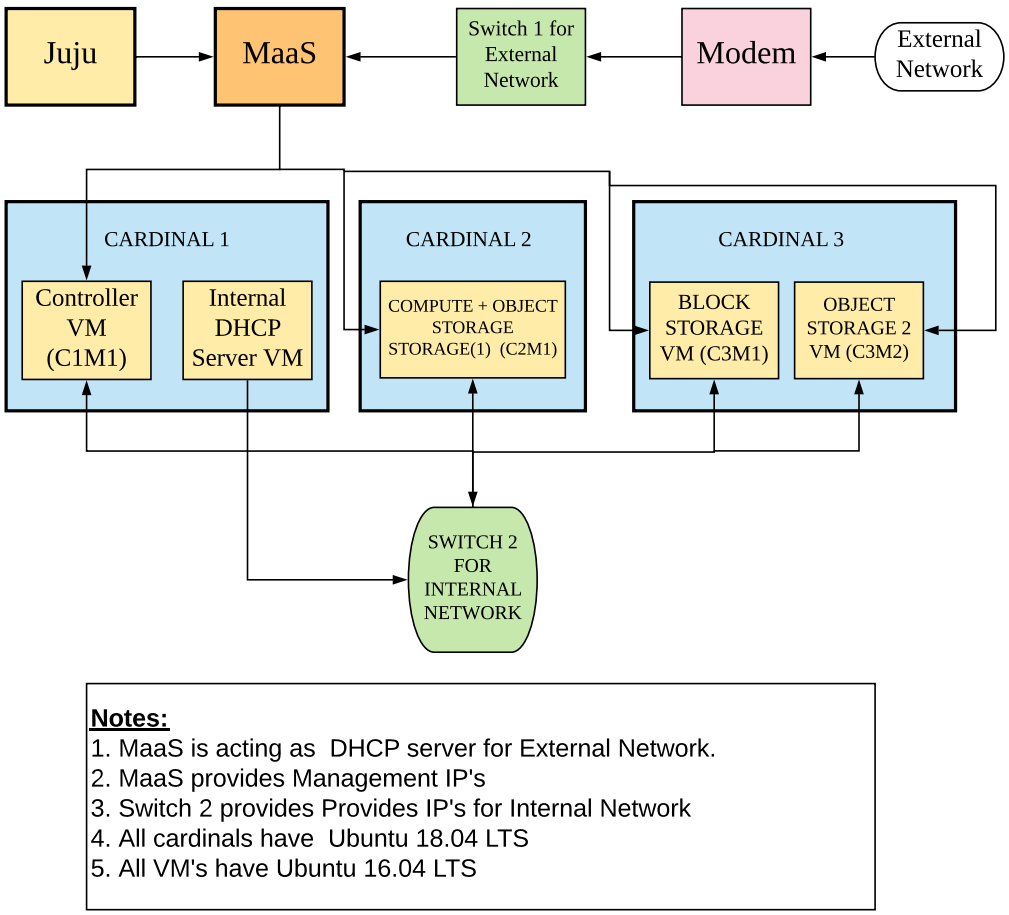


Figure 3.2: Private Cloud Architecture

services running on those machines.

Minimum Requirements for MAAS

The minimum requirements for the machines that run MAAS vary widely depending on local implementation and usage.

Factors that influence hardware specifications include:

1. The number of connecting clients (client activity).

2. The manner in which services are distributed.

3. Whether high availability is used.

4. Whether load balancing is used.

5. The number of images that are stored.

3.2.2 JAAS

Juju as a service or JAAS, is the best way to to quickly model and deploy cloud-based applications.

Why use Juju?

Juju is used to operate software on bare-metal servers by using Canonical's Metal as a Service (MAAS), in containers using LXD, and more. The models in Juju provide an abstraction which allows the operations know-how to be cloud agnostic. This means that Charms and Bundles in Juju can help in operating the same software with the same tool on a public cloud, private cloud, or a local laptop [2]. Figure 3.3 explains why Juju is used and how it is helpful to us.

3.2.3 Building a Testbed

To build a private Cloud, I used three Dell R420 servers with multiple Ethernet ports. These servers are named as Cardinal 1, Cardinal 2 and Cardinal 3. All servers have 20 GB RAM and 8 Intel Xeon processors on each of them. Ubuntu 18.04 LTS (Desktop Version) was used as the operating system running on each of them. I have two different desktops with 8 GB RAM and Ubuntu 18.04 LTS version to install MAAS and JAAS separately. MAAS is also acting as DHCP server for external network providing Management IP's. A VM is

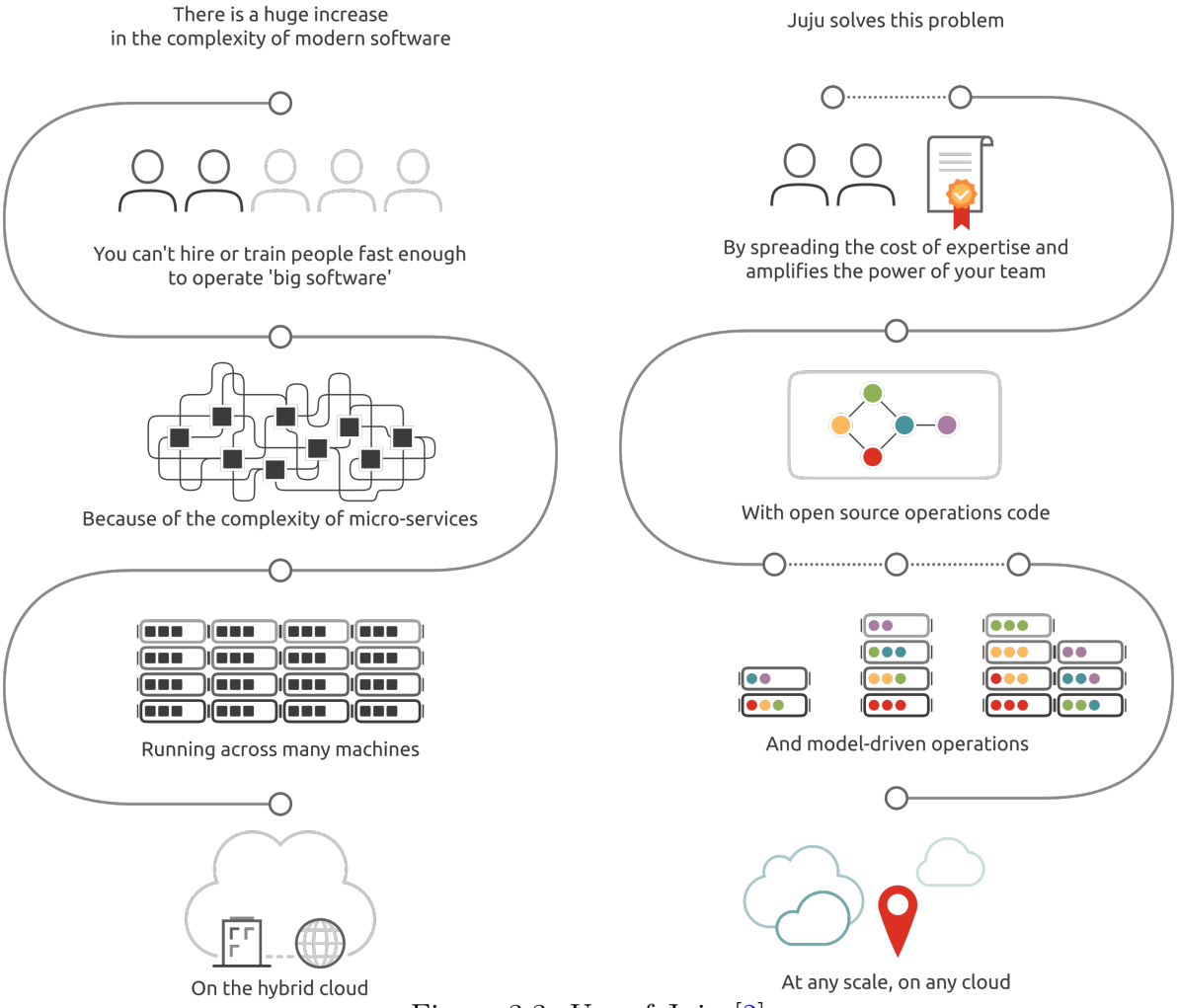


Figure 3.3: Use of Juju [2]

created on Cardinal 1 which is working as internal DHCP server to allocate Provider IP's. I have used OpenStack Pike (minimal installation) to create a private Cloud environment for these machines. OpenStack is a Cloud software platform with a three node architecture [59] as shown in Figure 3.2. OpenStack should have minimum three nodes to implement Cloud. There can be only one controller and network nodes in OpenStack setup. These three node are setup on three Dell servers using Qemu-KVM. These nodes are created as VMs on those servers to support the networking required while setting up OpenStack. The three node types in OpenStack are explained next.

Controller Node

The controller node runs various services like Identity service, Image service, management portions of Compute, management portion of Networking and the Dashboard.

I have deployed Controller node on Cardinal 1 as a VM using Qemu-KVM. Controller Node VM has Ubuntu 16.04 LTS (Server Edition) as operating system installed on it.

Compute Node

The hypervisor portion of compute that operates instances is run by compute node. By default it uses the KVM hypervisor. The compute node also runs a Networking service agent that connects instances to virtual networks. We can deploy more than one compute node.

For my experiment I have deployed one Compute Node on Cardinal 2 as a VM using Qemu-KVM. Compute Node has Ubuntu 16.04 LTS (Server Edition) as operating system installed on it.

Block Storage

The Block Storage node contains the disks that the Block Storage and Shared File System services provision for instances. "It provides persistent block storage for running instances" [59]. For my experiment, I have deployed one Block Storage Node on Cardinal 3 as a VM using Qemu-KVM. This node has Ubuntu 16.04 LTS (Server Edition) as operating system installed on it.

Object Storage

The Object Storage node contain the disks that the Object Storage service uses for storing accounts, containers, and objects. For my experiment, I have deployed two Object Storage

Node on Cardinal 2 and Cardinal 3 respectively as a VM using Qemu-KVM. These nodes have Ubuntu 16.04 LTS (Server Edition) as operating system installed on it.

Networking

Some nodes use two networks like internal and external network. They are called as Provider Network and Management Network respectively. The provider network bridges virtual networks to physical networks and relies on physical network infrastructure for layer-3 (routing) services. Additionally, a DHCP service setup on Cardinal 1 provides IP address information to instances. Similarly, the Management Network provides the physical network infrastructure for routing services.

3.3 Hadoop MapReduce Implementation

I have made two VMs on the compute nodes in my testbed. These VMs are used for demonstrating the migration for different types of files like csv, image, pdf and audio files. The data migration is done based on IP of these VMs. I have made this environment to run the MapReduce code because MapReduce requires the HDFS for running and executing the jobs. These codes of MapReduce are written using MATLAB environment. The Mapper and Reducer functions are implemented separately for different types of files. The setup of Hadoop MapReduce environment is explained in detail in Appendix A (A.2). With the help of data migration, I show that (as explained in next Chapters) using MapReduce for migration, prevents data loss as well as improves the efficiency of the migration.

Chapter 4

Solution Methodology

4.1 Introduction

My aim is to migrate data between two servers and compute if there is any data loss during the transfer. The proposed framework combines data migration and MapReduce. Apache Hadoop is the most commonly used MapReduce tool since it is an open source tool and easily available and hence, I have used this for my study.

Data migration involves transferring a large amount of data from one server to another, in general. If any data gets corrupted during the transfer, it is difficult to identify the location of the corrupted file. Hence, prior to migration the data must be optimized for easier transfer. For this optimization process, I have used the MapReduce framework.

After MapReduce step, the data is transferred from Server A to Server B in the private cloud environment. The KVP is obtained from the data which is now in the Server B. This new KVP is compared with the previous KVP to find if the values are same. If there is any mismatch in the KVP, it means that there is some loss in the transferred data. If the

matching of data takes place well without any error, it means that there is no loss in data.

4.2 Why MapReduce migration is preferred?

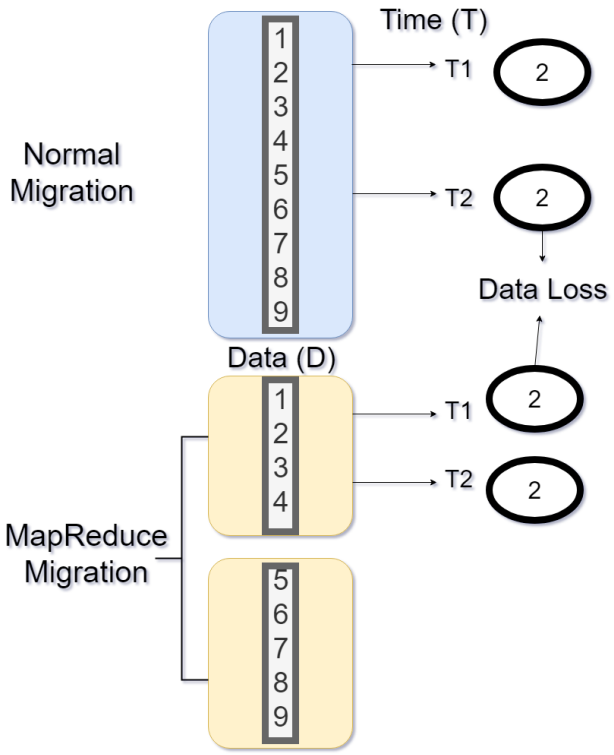


Figure 4.1: Normal Migration vs MapReduce Migration

As seen in Figure 4.1, during normal migration if the data is migrated as a single chunk at time T1, a particular item from the data set may be lost (for example, item 2 in this figure). To regain that data, the migration needs to be re-initiated fresh. Since the chunk is large, in general, the cost of redoing the migration becomes exorbitant. If there is loss of same data again during re-transmission, then migration needs to be repeated. This results in huge amount of traffic as well as execution cost increases since the whole data set is migrated repeatedly.

On the other hand, if MapReduce migration is preferred, where the whole data set is split

into smaller chunks, we can see that there is data loss from one chunk only, in this example. The other chunk is transferred without any loss. To regain the lost data, only the chunk where from data loss happened needs to be migrated rather than migrating entire data set. If there is loss of same data again, then migration needs to be repeated for that particular chunk only. This results in lowering the huge amount of traffic as well as execution cost decreases by nearly 50% for the above example. If the data size increases, the chunk size and number of chunk would also increase. Therefore, MapReduce based migration only needs to transfer the chunk from which data is lost. Hence, with increasing data size, the cost of execution and network traffic could be reduced.

4.3 Matlab-MapReduce

It is possible to effectively build a Hadoop MapReduce cluster by utilizing the MATLAB Distributed Computing Server in order to perform the data migration. It is necessary for both the systems to run the same MATLAB version to run this work. Mathworks has released a custom implementation of the MapReduce since 2014b version. This can be accessed by using the MapReduce function. However, this work has not used the function and attempted to perform data migration using MapReduce without the use of functions.

As shown in figure 4.2, the input data into the Map is stored in an object known as *datastore* that handles the data distribution and partitioning into small parts. Each of these parts is processed by a different map function, where this result is stored in another object called as *KeyValueStore*. These outputs are grouped by the different keys and each group of elements is encountered by a reduce function. The final output is stored in an output object from where it can be accessed. It has to be ensured that the data does not fall in the wrong

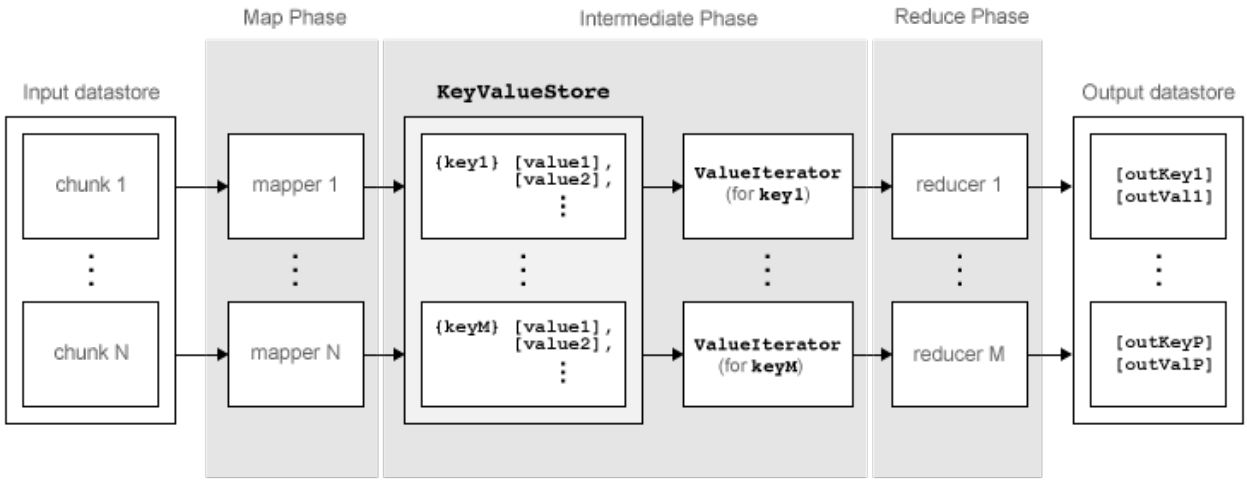


Figure 4.2: MapReduce Algorithm Phases [6]

hands. The data owner must have access to the documentation when the data is transferred and must also be able to control who can access the data.

For data migration, I have written a script in Matlab that replicates the data from Server A to Server B and will delete the data in Server A after MapReduce is performed. MapReduce model comprises of two major stages, the map stage and the reduce stage. The Map function optimizes the data and converts it into structured data. Mapping is performed in parallel on multiple nodes or groups of data.

After completion of Map stage, intermediate KVPs are sent to reduce function where the different mapping steps are combined. The reducer takes all the values associated with a single key k and outputs any number of KVPs. The data will be saved in the KVP, where the *Key* is an integer data assigned to each group of data. The *Value* in KVP is a floating-point type and contains the corresponding data. The *Key* and *Value* are stored as an array for each group of data. There might be more than one data with the same *Key*, however, the *Value* will be different. Since, the KVP might have more than one row or column, it is stored as a 2D array.

The MapReduce data migration algorithm (Algorithm 1) takes different types of input such as audio, video, images and text files. The first step is to create datastore for the data set. This datastore works as an input for MapReduce allowing MapReduce to process data in chunks. The input to the map function is data (INP), information (INF_VL) and intermediate Key Value store (IN_K_VL). The INP and INF_VL are the result of the call function made to the datastore. The map function adds the KVPs to the $\mathrm{IN}_{KV}Lobject$.

The inputs to the reduce function is intermediate key (KY_VL), value iterator (INT_VLTR) and final key value store (OT_K_VL). The KY_VL is the active key added by map function. Whenever there is a call made to reduce function, a new key from intermediate Key Value store (IN_K_VL) is provided. The INT_VLTR objects contains all the values associated with KY_VL. The HASNEXT and GETNEXT functions are used to scroll through the values. OT_K_VL is the final key value store where the reducer functions has added the KVPs. MapReduce takes all the KVPs from OT_K_VL and returns to the output datastore.

After MapReduce function, the migration takes place. The migration process has an important condition that both the servers should have the same version of Matlab environment(it is deployed on the servers). The migration is done based on the IP address of the sender and receiver. The sender defines the address of receiver in TCPIP function and the port. Similarly the receiver defines the IP address and port of the sender.

The size of the groups where the data loss has taken place is used to compute the total data loss during the transfer. This will be done by computing the difference between the total amount of data before the migration and total amount of data after the migration.

Algorithm 1 MAPRDUCE DATA MIGRATION

Input : Image, Audio, Video, Excel files
Output : (Key, Value) Pairs

Initialization
Function `Mapper(`*INP, INF_VL, IN_K_VL*`):`
| IMV = Data Fragmentation Condition (INP)
| Add IMV to IN_K_VL
return IN_K_VL
Function `Reducer(`*KY_VL, INT_VLTR, OT_K_VL*`):`
| **while** *HASNEXT (INT_VLTR)* **do**
| | OT= GETNEXT (INT_VLTR)
| **end**
| Add OT to OT_K_VL
return OT_K_VL
Function `Migrate(`*Server 1 to Server 2*`):`
| **Server 1**:
| MIG_D_V= TCPIP (Server 2 IP Address, Port, Client)
| Mapping Rule:
| Set (MIG_D_V, Output Buffer Size, Output Bytes)
| Fopen (MIG_D_V)
| Data Recovery:
| Fwrite (MIG_D_V, Input)
| Fclose (MIG_D_V)
|
| **Server 2**:
| SVR_END= TCPIP (Server 1 IP Address, Port, SERVER)
| Set (SVR_END, InputBufferSize, Input Bytes)
| Set (SVR_END, Timeout, 30)
| Fopen (SVR_END)
| Act_D = fread (SVR_END, INPUT PORT)
| Fclose (SVR_END)
return

Data Loss: D_L

Total amount of data before migration: D_{BM}

Total amount of data after migration: D_{AM}

$$D_L = D_{BM} - D_{AM} \tag{4.1}$$

If there is any loss in data, it is necessary to get back the lost data. Since the data is still available in Server A, KVP is used to identify the location of the lost data. The group where the lost data is present will be subjected to MapReduce procedure to optimize it again and then the data is transferred again to Server B. This is done until all the data has been transferred successfully. Once this has been confirmed, the data in Server A will be deleted to complete the last step of data migration.

Chapter 5

Experiments, Results and Evaluation

5.1 Experiment

A cloud environment is initially setup with the help of multiple servers. The three identical main servers used in this research have the following configuration.

DELL POWEREDGE R420 RACK SERVER	
Cores	12
Processors	Intel Xeon processor E5-2400 product family
RAM	20 GB
Hard drives	500 GB SATA disk
Operating System	Ubuntu 16.04 LTS (Xenial Xerus)

In this work, VidTIMIT dataset has been used. This dataset comprises of images, audio and video files. The simulation results are discussed in the Evaluation Section below. Different file formats have been used to transfer the data between the two servers. The

execution cost, file size, data loss and efficiency are calculated for different file formats and are tabulated. The videos and images in the dataset contain typical outdoor/indoor visuals that takes place commonly. Some of these images have also different challenges like different weather condition, noisy video and/or low frame rate. It can be seen that the efficiency is the highest for transferring images with very low data loss in VidTIMIT dataset. The efficiency can be seem best in case of simple textual data in comparison to VidTIMIT dataset.

5.1.1 Flow of Implementation Process

Figure 5.1.1 explains the implementation process. The following steps are involved in the implementation process:

1. Strategy Development
2. Assessment and Analysis
3. Data Preparation
4. Validating and Staging

Strategy Development

The strategy development process will consider the style of migration that is most suitable for user needs. It can be chosen from different strategies based on the needs and available processing windows. The strategy depends on the following criteria:

1. Data migration from server to server (no administrative permission): In this process, the data will be migrated from one server platform to another (such as moving from

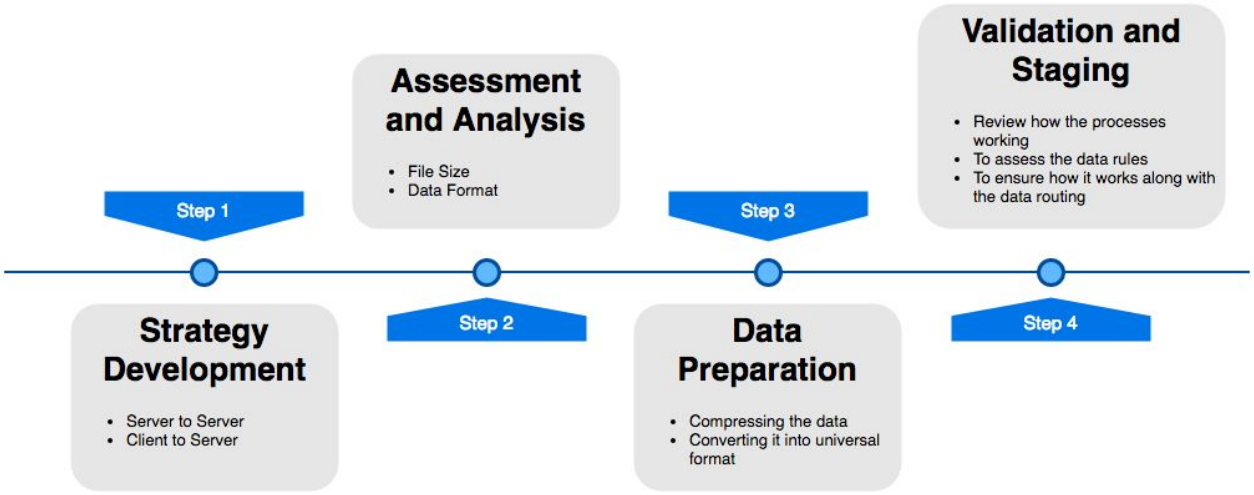


Figure 5.1: Work Flow Schema

server A to server B in Cloud environment), but do not have administrator access hence there is no control over the setup and configuration of the server. When the data is transferred between the servers it cannot be accessed by anyone else since they will not have administrator level access. This can provide more security to the data.

2. Data migration from server to server (with administrative permission): In this process, the data will be migrated from one server platform to another (such as moving from server A to server B in Cloud environment), also our servers will have root or administrator access, which will allow to have full control over the setup and configuration of the server. Since the users can access the data during the migration, they will have full control of their data. However, the security features will be reduced.

3. Data migration from client to server (no administrative permission): In this process, the data will be migrated from a client (such as Web browser) to the server in the Cloud, but do not have administrator access hence there is no control over the setup and configuration of the server.

4. Data migration from client to server (with administrative permission): In this process, the data will be migrated from a client (such as web browser) to the server in the Cloud, also our servers will have root or administrator access, which will allow to have full control over the setup and configuration of the server.

For my thesis, I have studied server to server migration.

Assessment and Analysis

I have assessed and analyzed data migration based on two important parameters. They are *file size* and *data format*. The file size will help in computing data loss. In order to assess the performance of the migration process, different files sizes like KB, MB, and GB will be considered. The data format will help us in identifying what type of data is to be transferred during migration. During the migration process, the data file format will be csv, excel,txt etc. By breaking the data into smaller chunks, I have also migrated audio and video files, which are generally large.

Data Preparation

The data preparation involves two steps:

1. Compressing the data: This is achieved using MapReduce. The process is explained in subsection 5.2 (MapReduce Implementation Details)

2. Converting the final output into PDF or another universal format like RAR to ensure the security and privacy level of data.

Validation and Staging

The migration process performance will be validated for ensuring the requirements and customized settings function. The validation and performance analysis process will cover the below features:

1. review the process flow.

2. assess the data rules.

3. to ensure proper working of the process along with the data routing.

The above mentioned features will be achieved using various parameters tabulated below.

Parameter	**Description**
Schedule ID	Migration Schedule ID
Server	Primary file system's server
Files Migrated	The number of files that were migrated
Status	Migration completion status
Start Time	Date and time when the migration began
End Time	Date and time when the migration ended
Rules used	Rules used by the policy
Pre-Migration File System Space Used	File system size, total used space before the migration
Post-Migration File System Space Used	File system size and the total used space after the migration
File System Capacity	File system's total capacity

These parameters are contained in the migration reports. At last, the migration reports can be downloaded in CSV format, then imported into a spreadsheet and processed, saved, or printed.

5.2 Results

5.2.1 Data Migration for audio and video files

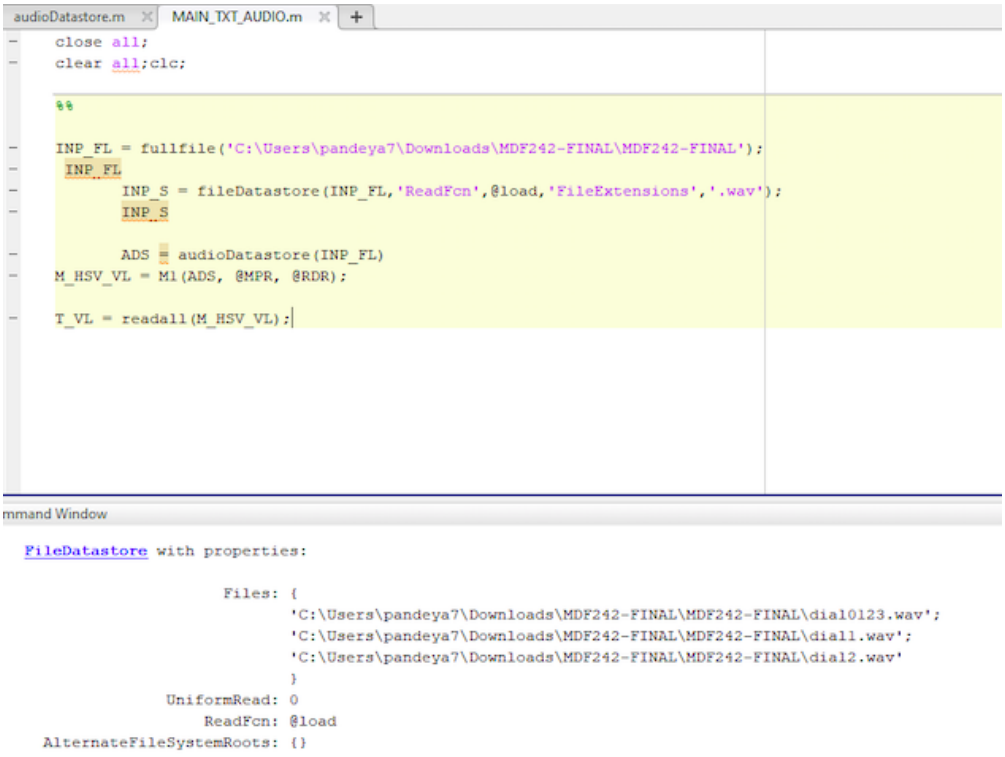


Figure 5.2: Loading Audio Video Files

The files get parsed into small chunks and then get transferred. The data is then migrated to targeted location. Audio and Video files are migrated using MapReduce as shown in Figure 5.2. Different files with the extension .wav, .mp4 are transferred between two servers. Initially mapping is done in parts and takes place in steps of 20% as shown in Figure 5.3. Once

the mapping function takes place and gets completed, the reduce function will get initiated. Similarly, video files with the extension .mp4 are also migrated using this approach.

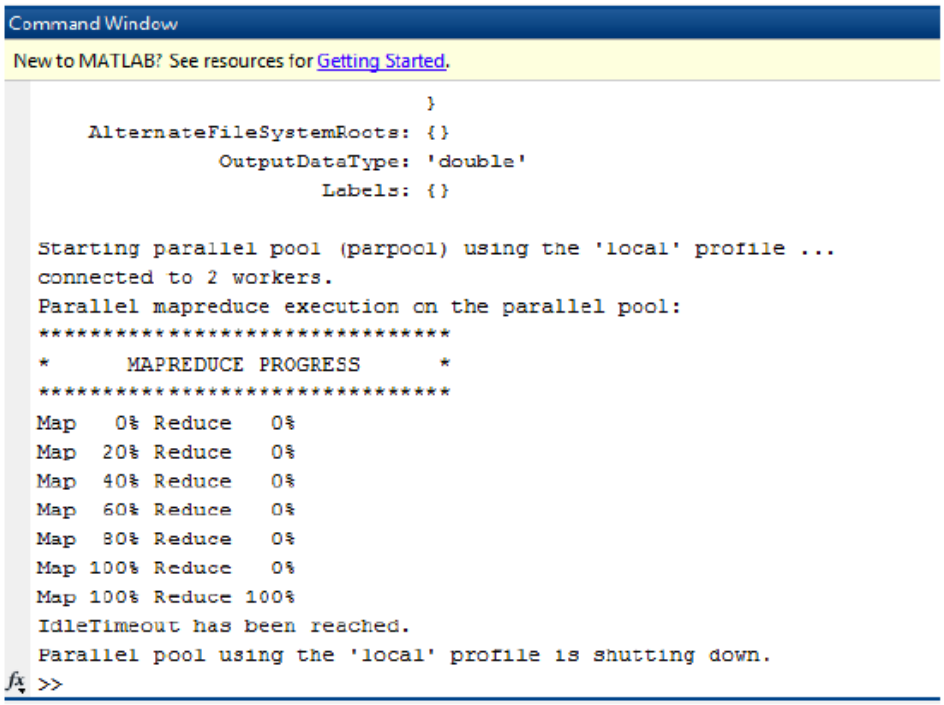


Figure 5.3: MapReduce for Audio Video Files

5.2.2 Data Migration for image files

Image files are also migrated using MapReduce in this work as shown in Figure 5.4. Data migration is performed for the image with extension .jpg, .tiff, and .png. The dimensions of the image files are read with each pixel value considered to be rows and columns of a matrix. After the dimensional pixels are taken into the matrix, mapping is performed as seen in Figure 5.5. The length and breadth of the image in terms of pixels will be equal to the dimensions of the matrix. The mapping takes place in chunks of 33%. After individually mapping the pixels and gets completed, the reduce function will take place.

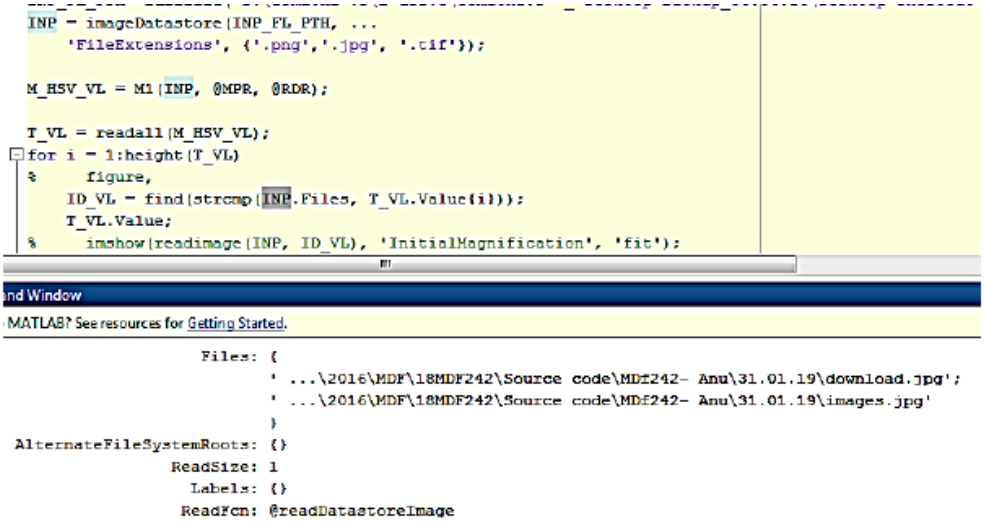


Figure 5.4: Loading Image Files

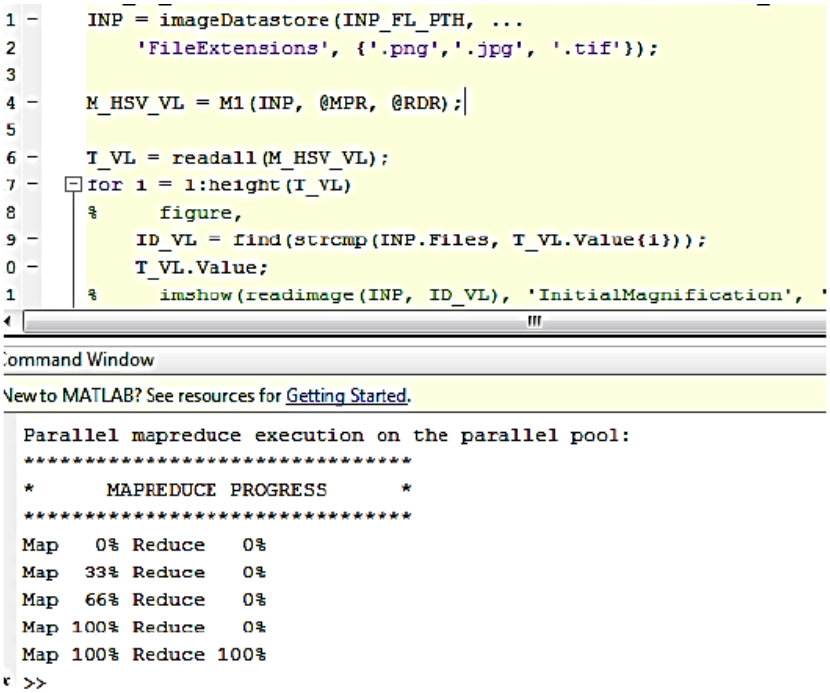


Figure 5.5: MapReduce for Image Files

5.2.3 Data Migration for documents

Experiments with document files for data migration were done for my thesis study as wells as shown in Figure 5.6. The data Migration is performed for spreadsheet files with

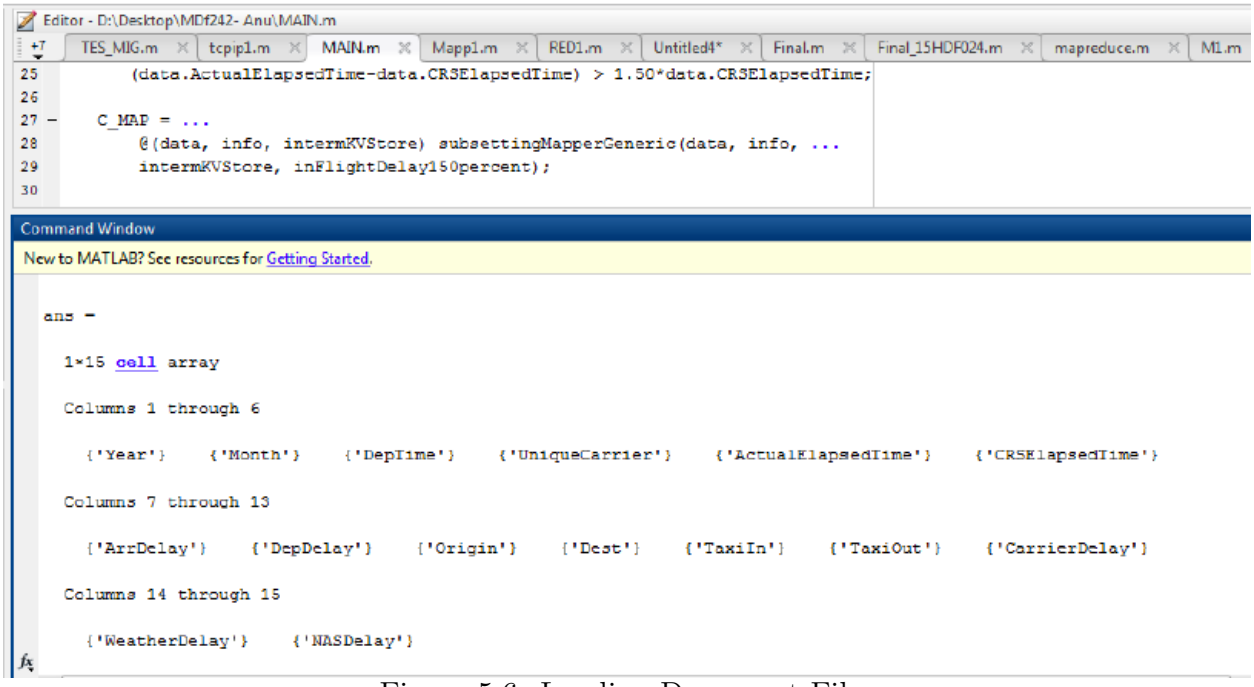


Figure 5.6: Loading Document Files

extension .xsl and .csv. Since spreadsheets store data in multiple rows and columns, each cell contains some data, hence the data in these files are read from every row and column. The contents of each cell are taken as a constituent of a matrix since all the contents of the matrix are considered with the size of excel. Therefore the number of rows and columns in the resulting matrix will be equal to the rows and columns of the spreadsheet. After the cell contents are taken into the matrix, mapping is done as seen in Figure 5.7. It takes place in increments of 50% and when mapping is completed, the reduce function takes over. The spreadsheet gets parsed into chunks and then transferred. Once this takes place, the data is then migrated to targeted location.

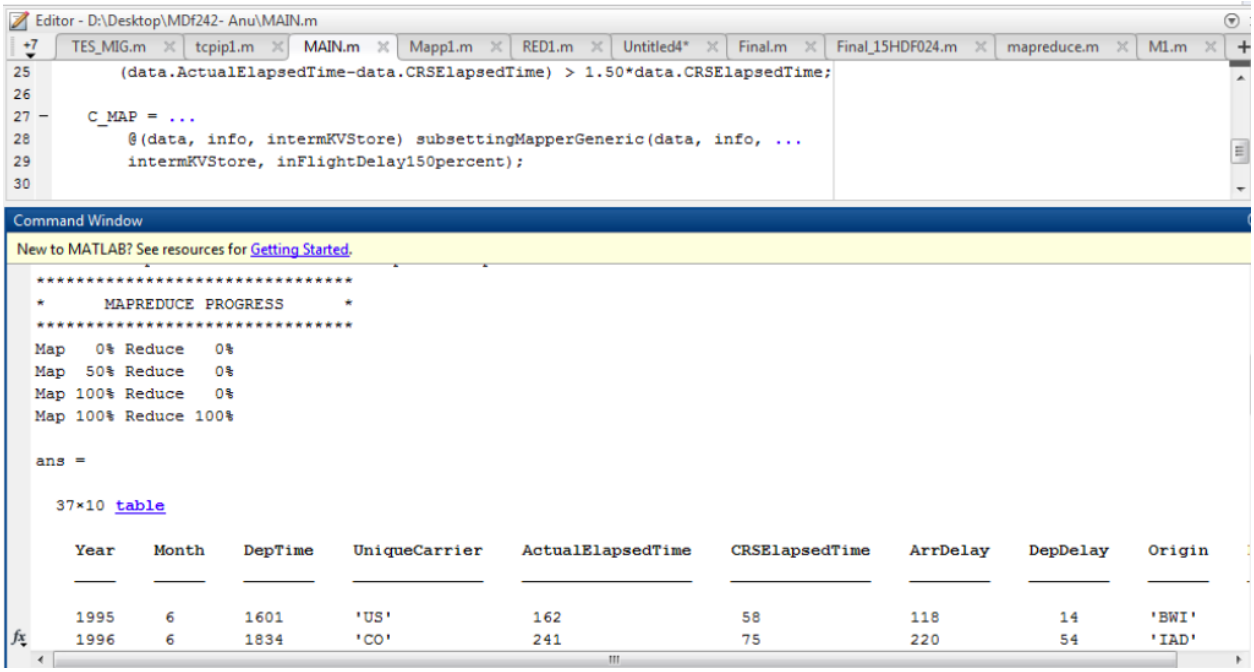


Figure 5.7: MapReduce for Document Files

5.3 Evaluation

I have performed experiments and evaluated performance of the data migration with various measures to check the effectiveness of the proposed method that combines the data migration method using MapReduce with input parameters such as number of files, file size, output parameters such as accuracy and the cost of the transfer.

The energy consumption of the servers is generally high, which accounts for the high data migration cost. The cost of migration has to be low in any framework in order to be efficient. I will evaluate the cost of migration for the proposed model and will also evaluate the cost for migrating data without using MapReduce. Execution cost will be calculated by adding the cost incurred during the idle time with the cost incurred to execute the work flow schema.

Execution cost: E_c

Idle Time: I_t

Busy Time: B_t

In my study, the cost factors mainly considers the cloud services and partially the cloud users. The cost factors are:

Information that is stored after transfer: λ

Data that is transferred over the network : γ

$$E_c = \frac{(\lambda * I_t) + (\gamma * B_t)}{\lambda + \gamma} \tag{5.1}$$

Here, λ, γ are cost factors wherein $\lambda <= \gamma$

Speed of Migration: V

Amount of data migration (i.e. data per unit): A_{data}

Total Time: T_t

$$V = \frac{A_{data}}{T_t} \tag{5.2}$$

Efficiency: E

Efficiency is another important metric for performance evaluation. It can be defined as the percentage of data that is transferred without any data loss.

Total Data Transferred: D_T

File Size (MB)	Total Execution Cost (Seconds)	Data Loss (MB)	Efficiency (%)	Speed of Migration (MB/s)
10	9.789856	1.07	90.674	1.02
100	41.099143	6.98	91.986	2.43
400	90.371984	19.91	94.793	4.43
700	159.12896	34.08	95.168	4.41
1100	242.95312	41.09	96.023	4.56

Table 5.1: Audio and Video Migration

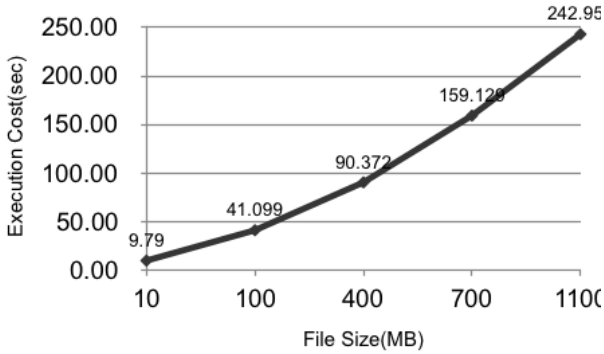


Figure 5.8: Execution Cost

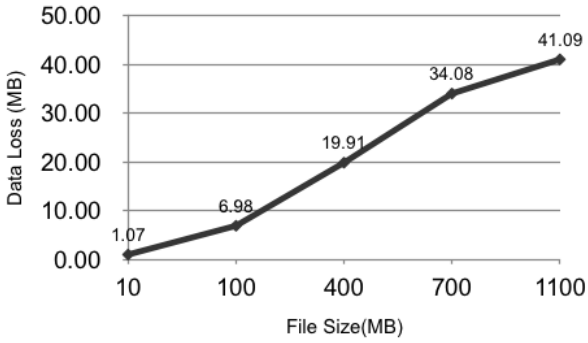


Figure 5.9: Data Loss

Data Re-transferred: D_{RT}

$$E = \frac{D_T - D_{RT}}{D_T} * 100\% \tag{5.3}$$

Table 5.1 captures various results for the performance of data migration of audio video data files. The algorithm working behind these files need to run an additional function as audio datastore. Due to this there is a delay, which leads to addition of few seconds to the total execution cost. The total execution cost keeps decreasing with respect to the size of data transfer. The efficiency improves as it can be seen in the Table showing that for larger data sets, the algorithm performs better. The data loss [1] also reduces with increasing size.

[1]This is during the first migration and is recovered in subsequent migration

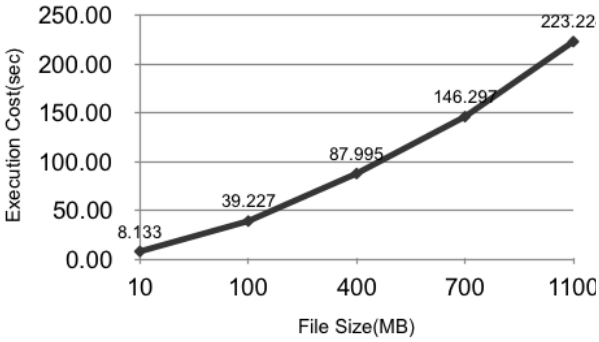


Figure 5.10: Execution Cost

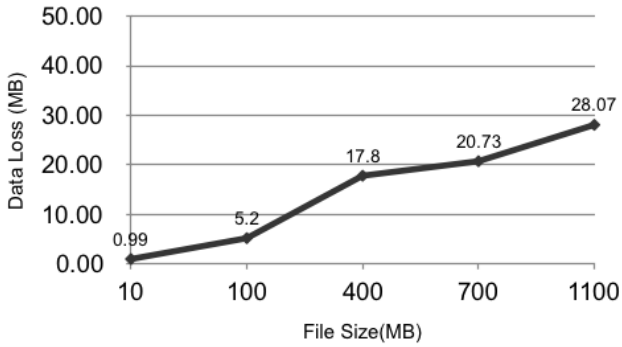


Figure 5.11: Data Loss

This concludes that our algorithm performs efficiently in case of audio video migration for scaled data size.

File Size (MB)	Total Execution Cost (Seconds)	Data Loss (MB)	Efficiency (%)	Speed of Migration (MB/s)
10	9.789856	1.07	90.674	1.02
100	41.099143	6.98	91.986	2.43
400	90.371984	19.91	94.793	4.41
700	159.12896	34.08	95.168	4.40
1100	242.95312	41.09	96.023	4.53

Table 5.2: Image Migration

Table 5.2 presents the performance of our algorithm on image data files. The Map and Reduce function, for image migration use contrast and saturation as key values. The total execution cost keeps decreasing with respect to the size of image transferred. The efficiency improves as it can be seen in the Table showing that for larger data sets, the algorithm performs better. The image being static (still), performs better than audio-video files. The data loss [2] is low with increasing size. Hence, it can be concluded that the algorithm performs well in case of image migration also with respect to scaling data sizes.

Table 5.3 presents the performance of our algorithm for textual data files in csv or xlsx

[2]As mentioned before, this is during the first migration and is recovered in subsequent migration

File Size (MB)	Total Execution Cost (Seconds)	Data Loss (MB)	Efficiency (%)	Speed of Migration (MB/s)
10	7.193986	0.93	93.664	1.39
100	33.938971	3.71	94.986	2.95
400	71.392817	13.77	94.793	5.61
700	138.029641	20.73	96.168	5.07
1100	194.837194	28.07	97.023	5.64

Table 5.3: Text Migration)

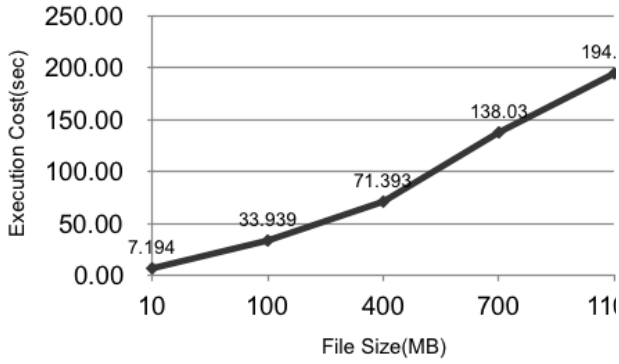


Figure 5.12: Execution Cost

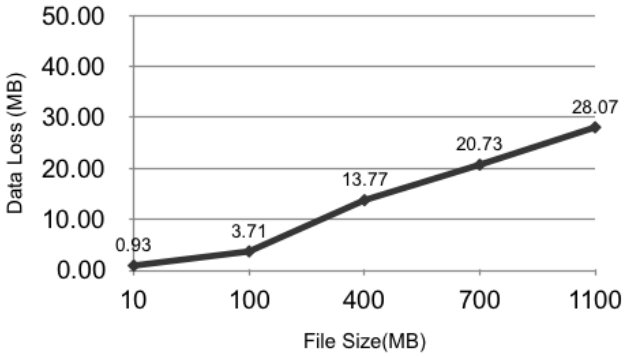


Figure 5.13: Data Loss

format. The Map and Reduce function for document migration uses keywords as key values. This performs better than other data formats (such as image, audio and video). The total execution cost is very low in comparison to other data format with same amount of data size. The total execution cost keeps decreasing with respect to the size of documents transferred.

The efficiency improves as it can be seen in the Table showing that for larger data sets, the algorithm performs well. Due to data being present in row-column format, we can separate the data easily based on key values. Hence, the performance improves with respect to other file format such as image, audio and video files. The data loss [3] is low with increasing size. With this we can conclude that our algorithm performs better in case of document migration with respect to increasing size of file.

[3]As mentioned before, this is during the first migration and is recovered in subsequent migration

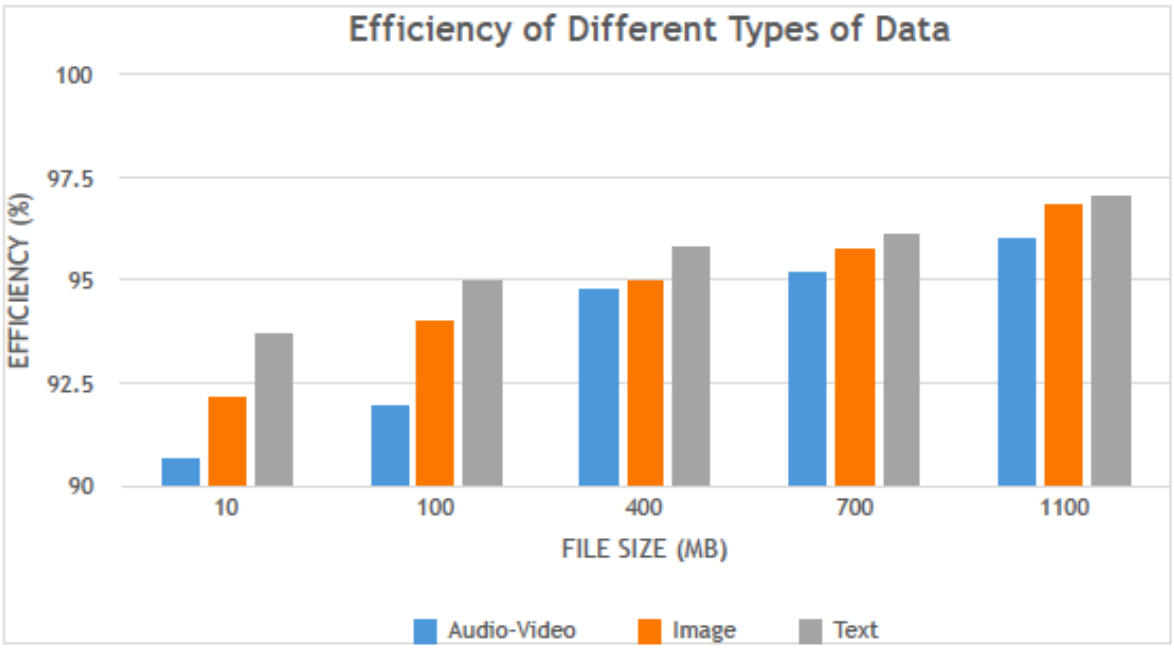


Figure 5.14: Efficiency Analysis

Analyzing the results in Figure 5.14, we can observe that the efficiency is best for textual data in csv or xlsx format. It can be seen that with increasing size of data, the efficiency increases irrespective of the data format. This shows that algorithm works better providing a high efficiency for bigger data sets. The goal was to minimize the data loss and it decreases with respect to the size of the data migrated. The datastore functionality provided by Matlab is a benefit over other programming languages as it helped in optimizing the datasets.

Chapter 6

Conclusion

I have proposed a migration technique in this thesis, with which I have shown that data transfer can be done efficiently. I have used MapReduce to achieving this goal of efficient data migration. The architectural model and algorithm I have developed for migrating the data has also helped reducing the data loss during transfer. The MapReduce has efficiently optimized the data migration code when large data sets are considered for migration. I have introduced new performance metrics using which the efficiency and the cost for migration have been computed. I have shown that the efficiency of the proposed algorithm increases with the increasing file size, that is, I have established the scalability of the proposed algorithm. Also, I have shown that can be reduced.

The data migration technique I have introduced in my experiments shows better performance than the previously available work in cloud computing literature. I have done experiments using image files like png, jpeg and tiff, audio files like wav, video files like mp4 and documents like xls and csv. While performing the simulation, I have faced issues such as internet interruptions, failure of a server, less bandwidth and less frequency resulting

in small amount of data loss in the experimental results reported. With a stable network connectivity, I expect that the data loss could be avoided.

I have used large heterogeneous files (text, images, audio and videos) for my algorithm to compute execution time and efficiency. The execution cost can further be reduced if the files formats were homogeneous while also decreasing the data loss.

With the limited resources used for the current study, the size of files migrated were bounded due to system limitations and also data loss could not be avoided completely. By improving the configuration of the system architecture and physical servers we might expect to improve the performance further.

The efficiency of my algorithm can only be guaranteed for input files with regular data. For complex data (features of the input files) and other file formats, improving the efficiency of the algorithm is left as future work.

Chapter 7

Key Contributions

For my thesis I have [64]

1. made an architectural model for setting up private cloud. I have used physical server for setting up the private cloud. The private cloud is setup using latest technologies like MAAS and JUJU which makes cloud highly available, fault tolerant and elastic;

2. designed an algorithm for migrating data in private cloud environment;

3. introduced new metrics for evaluating the overall performance of the private cloud and data migration algorithm.

Appendix A

Source Code

A.1 Install MAAS [1]

One small server for MAAS is needed. I have a system with 8GB RAM and Core i5 as processor. It is recommended to have the MAAS server provide DHCP and DNS on a network the managed machines are connected to. This is done in further steps.

1. Install Ubuntu server 18.04 LTS (Desktop version)

2. Run the following commands on terminal window.

   COMMAND: sudo apt update

   COMMAND: sudo apt install maas

3. Create your admin credentials

   COMMAND: sudo maas init

4. Login using the credentials

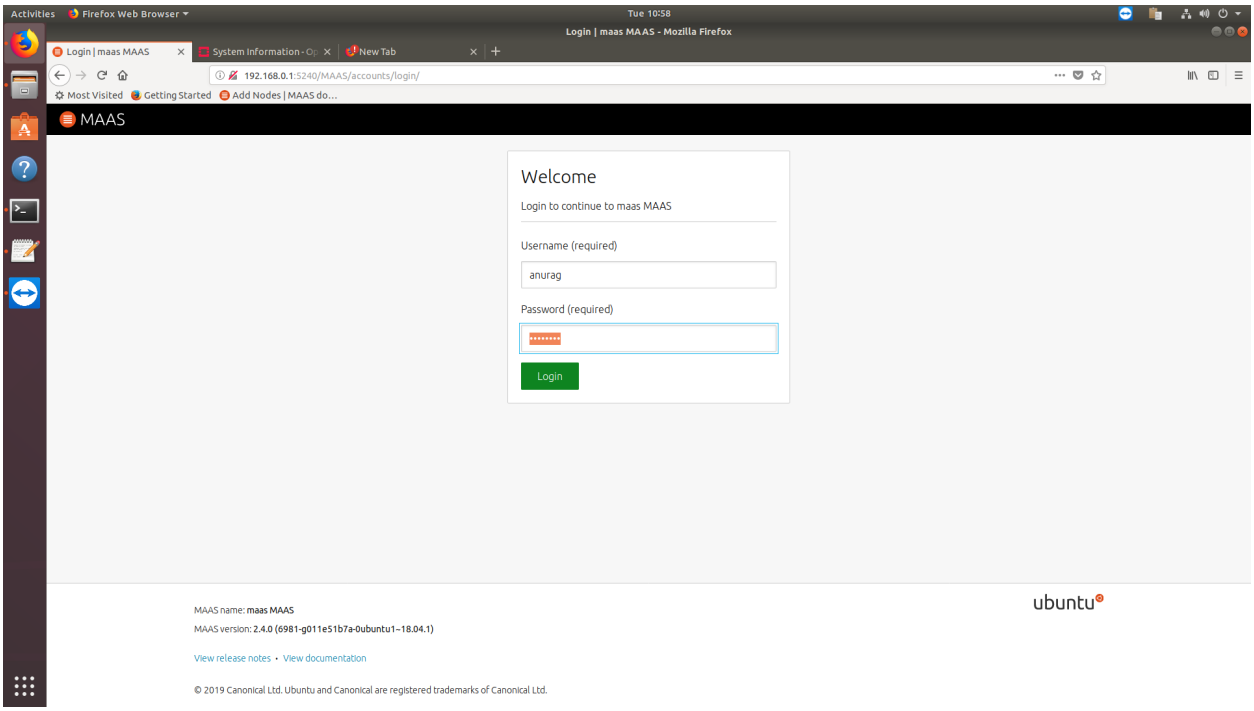


Figure A.1: MAAS Login

5. Set the DHCP properties (Figure A.2)

   Select the subnet where to create the DHCP Dynamic range on.

   Fill in the details for the dynamic range.

6. Enlist and commission servers (Figure A.3 and A.4)

   (a) Set all the servers to PXE boot

   (b) Boot each machine once. You should see these machines appear in MAAS

   (c) Select all the machines and “Commission” them using the “Take action” button

   (d) When machines have a “Ready” status you can start deploying

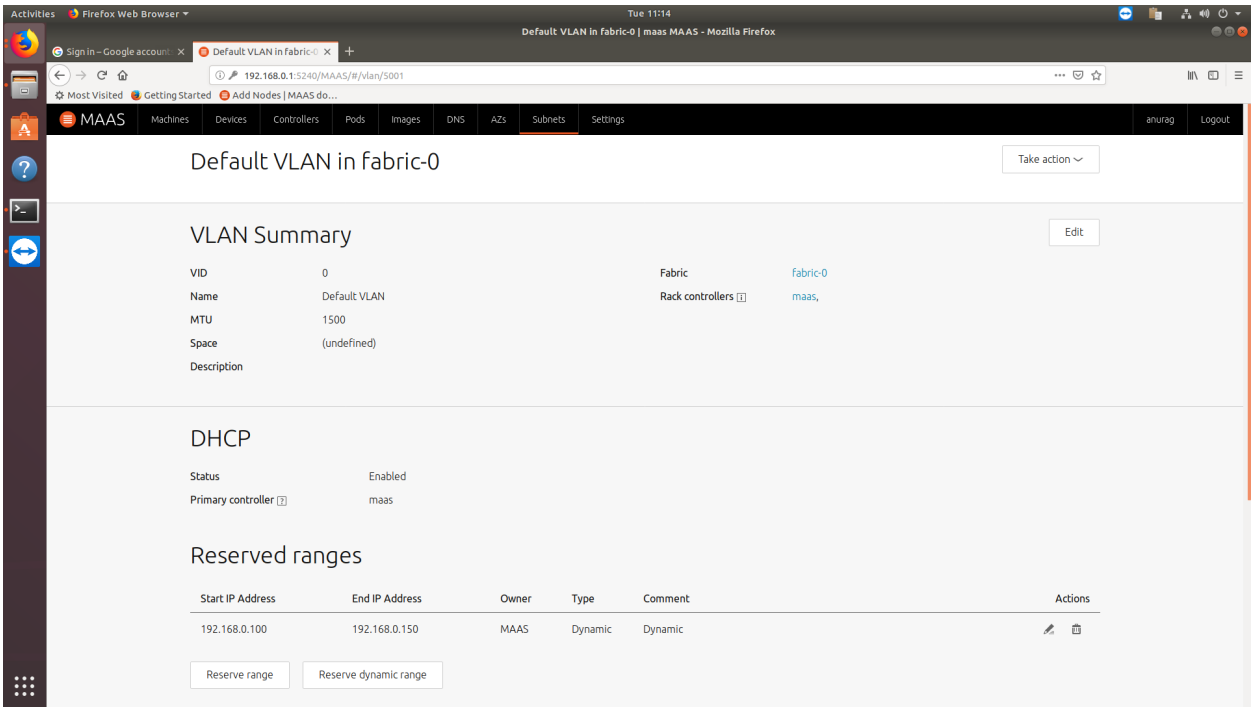


Figure A.2: DHCP enabled

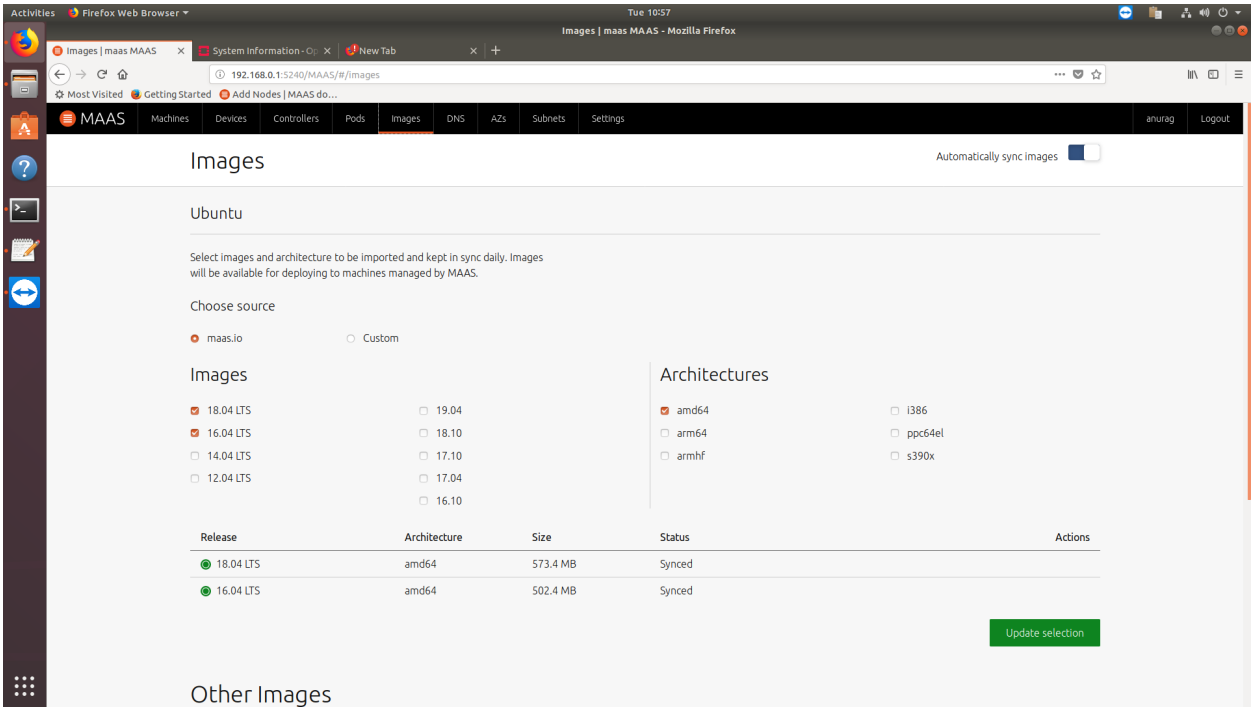


Figure A.3: Images used in Deployment

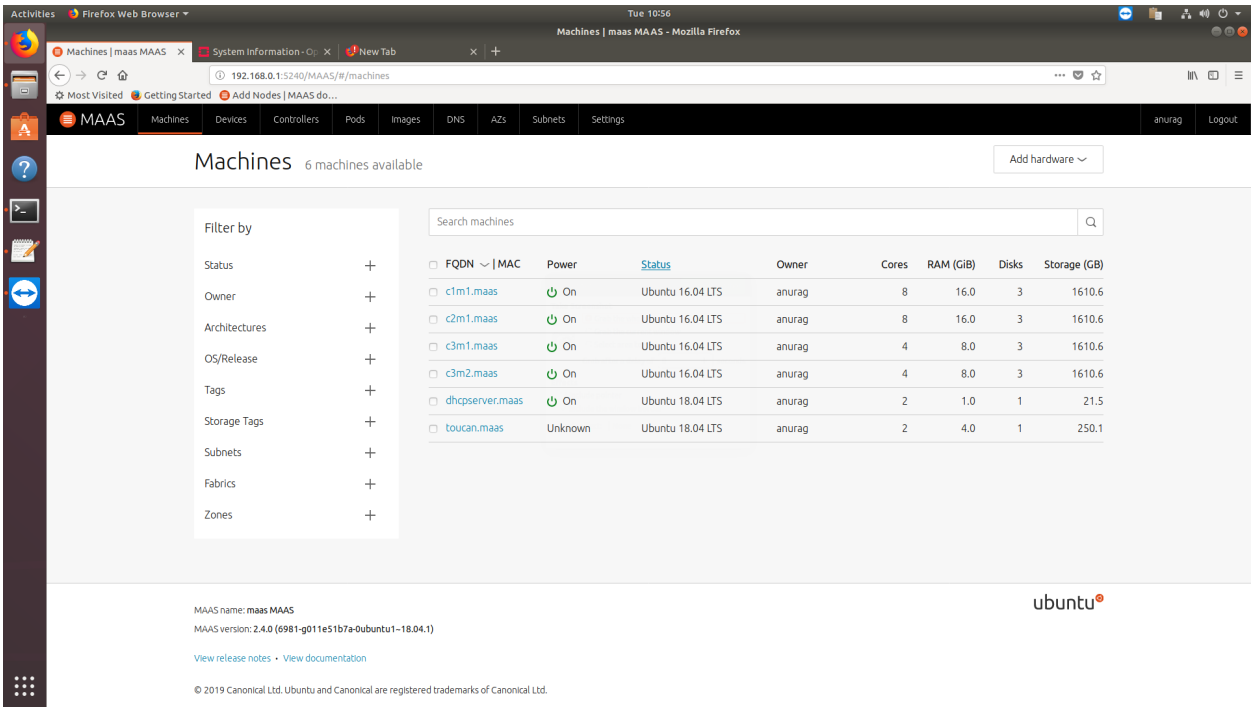


Figure A.4: List of Working Machines

A.2 Install JAAS [2]

The most recent stable version of Juju is 2.5.1. The recommended way to install Juju is with snaps. On Ubuntu 16.04 LTS (and greater) snapd is installed by default. I have made a separate machine for installation of JUJU. It is also added in MAAS.

1. Juju can be installed using this command

   COMMAND: sudo snap install juju –classic

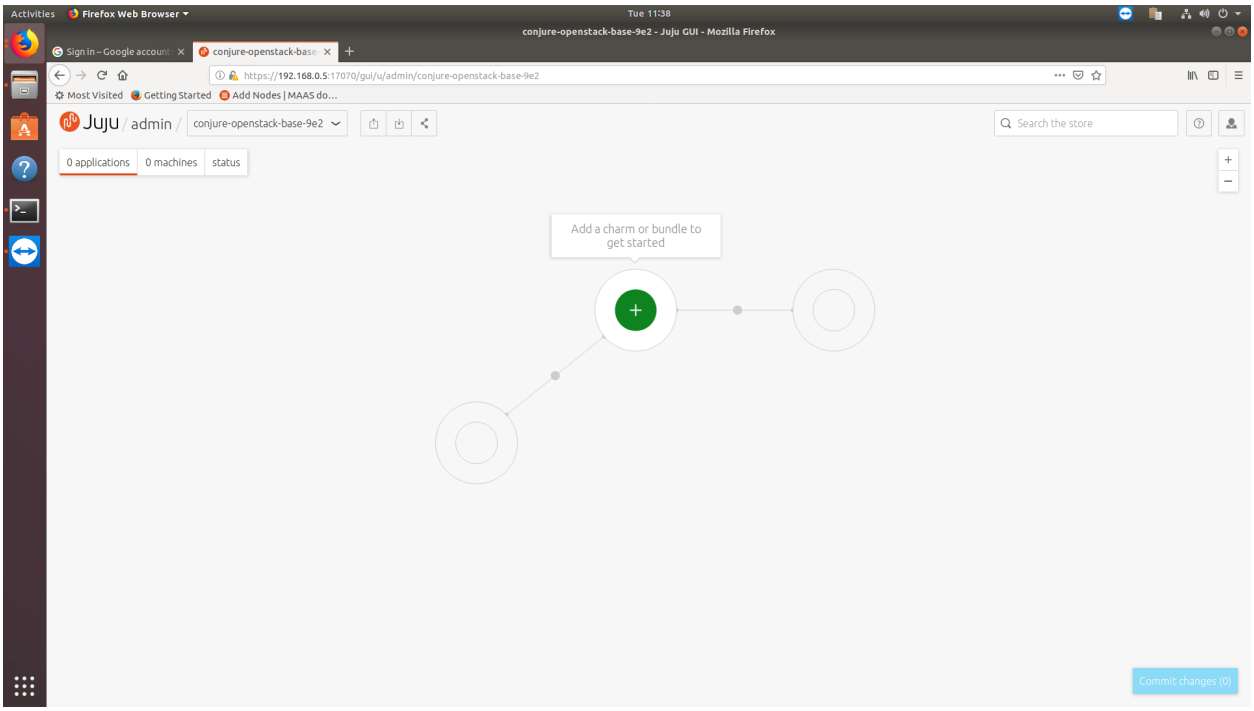


Figure A.5: After Installation

A.3 Install Openstack [3]

I have installed OpenStack (Pike version) minimal version. For minimal deployment few components need to be deployed.

1. Identity Service (Keystone)

   (a) Install latest version of python-pyasn1

   (b) Connect to the database server as the root user

   COMMAND: mysql -u root -p

   (c) Create the keystone database

   COMMAND: CREATE DATABASE keystone

   (d) Grant proper access to the keystone database

(e) Install the packages

COMMAND: zypper install openstack-keystone apache2-mod_wsgi

(f) Populate the Identity service database:

COMMAND: su -s /bin/sh -c "keystone-manage db_sync" keystone

(g) Bootstrap the Identity service

(h) Edit the /etc/sysconfig/apache2 file and configure the APACHE_SERVERNAME option to reference the controller node

2. Image Service (Glance)

(a) connect to the database server as the root user

COMMAND: mysql -u root -p

(b) Create the glance database

COMMAND: CREATE DATABASE glance

(c) Create the glance user:

COMMAND:openstack user create –domain default –password-prompt glance

(d) Create the glance service entity:

COMMAND: openstack service create –name glance –description "OpenStack Image" image

(e) Create the Image service API endpoints:

COMMAND: openstack endpoint create –region RegionOne image public http://controller:9292

(f) Install the packages

COMMAND: apt install glance

(g) Populate the Image service database

COMMAND: su -s /bin/sh -c "glance-manage db_sync" glance

3. Compute Service (Nova)

(a) Create the nova_api, nova, and nova_cell0 databases as root user

(b) Create the nova user

COMMAND: openstack user create –domain default –password-prompt nova

(c) Create the nova service entity

COMMAND: openstack service create –name nova –description "OpenStack Compute" compute

(d) Create the Compute API service endpoints

COMMAND: openstack endpoint create –region RegionOne

(e) Install the packages

COMMAND: apt install nova-api nova-conductor nova-consoleauth nova-novncproxy nova-scheduler nova-placement-api

4. Networking Service (Neutron)

(a) Create the neutron user

COMMAND: openstack user create –domain default –password-prompt neutron

(b) Create the neutron service entity

COMMAND: openstack service create –name neutron –description "OpenStack Networking" network

(c) Restart the Compute API service and start the Networking services and configure it to start when the system boots.

5. Dashboard (Horizon)

   (a) Install the packages

   COMMAND: apt install openstack-dashboard

   (b) Edit the /etc/openstack-dashboard/local_settings.py file

   COMMAND: Configure the dashboard to use OpenStack services on the controller node: OPENSTACK_HOST = "controller"

   (c) Reload the web server configuration

   COMMAND: service apache2 reload

A.4 Install Hadoop MapReduce [4]

1. Download latest Java package

2. Extract the Java Tar File.

   COMMAND: tar -xvf jdk-8u101-linux-i586.tar.gz

3. Download the Hadoop 2.7.3 Package.

   COMMAND: wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz

4. Extract the Hadoop tar File.

   COMMAND: tar -xvf hadoop-2.7.3.tar.gz

5. Add the Hadoop and Java paths in the bash file (.bashrc). Open. bashrc file and add Hadoop and Java Path.

   COMMAND: vi .bashrc

```
# User specific aliases and functions

export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

# Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

Figure A.6: Hadoop Installation – Setting Environment Variable

   Then, save the bash file and close it.

6. For applying all these changes to the current Terminal, execute the source command.

   COMMAND: source .bashrc

7. Edit the Hadoop Configuration files

   COMMAND: cd hadoop-2.7.3/etc/hadoop/

8. Open core-site.xml and edit the property mentioned below inside configuration tag:

   COMMAND: vi core-site.xml

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Figure A.7: Hadoop Installation – Configuring core-site.xml

9. Edit hdfs-site.xml and edit the property mentioned below inside configuration tag:

   COMMAND: vi hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
<value>false</value>
</property>
```

Figure A.8: Hadoop Installation – Configuring hdfs-site.xml

10. Edit the mapred-site.xml file and edit the property mentioned below inside configuration tag:

    COMMAND: cp mapred-site.xml.template mapred-site.xml

    COMMAND: vi mapred-site.xml.

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

Figure A.9: Hadoop Installation – Configuring mapred-site.xml

11. Edit yarn-site.xml and edit the property mentioned below inside configuration tag:

    COMMAND: vi yarn-site.xml

12. Edit hadoop-env.sh and add the Java Path as mentioned below:

    COMMAND: vi hadoop–env.sh

13. Go to Hadoop home directory and format the NameNode.

    COMMAND: bin/hadoop namenode -format

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Figure A.10: Hadoop Installation – Configuring yarn-site.xml

```
# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

Figure A.11: Hadoop Installation – Configuring hadoop-env.sh

14. Once the NameNode is formatted, go to hadoop-2.7.3/sbin directory and start all the daemons.

    COMMAND: cd hadoop-2.7.3/sbin

    COMMAND: ./start-all.sh

    The above command is a combination of ***start-dfs.sh, start-yarn.sh*** & ***mr-jobhistory-daemon.sh***

15. Hadoop cluster is installed.

Bibliography

[1] Canonical Ltd. (2018) Install maas. Accessed: 2019-02-07. [Online]. Available: https://maas.io/install

[2] Canonical Ltd. (2018) Juju. Accessed: 2019-02-07. [Online]. Available: https://jujucharms.com/

[3] Rackspace Cloud Computing. (2018) Openstack installation guide. Accessed: 2018-12-17. [Online]. Available: https://docs.openstack.org/install-guide/

[4] Apache Software Foundation. (2018) Hadoop cluster. Accessed: 2018-12-17. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

[5] S. Kumar. (2018) What is hadoop mapreduce and how does it work? Accessed: 2018-08-22. [Online]. Available: https://qph.fs.quoracdn.net/main-qimg-82813242aa853b91a91b96134f0c5c13-c

[6] I. The MathWorks. (2019) Getting started with mapreduce. Accessed: 2019-07-01. [Online]. Available: https://www.mathworks.com/help/matlab/import_export/getting-started-with-mapreduce.html

[7] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, "Design tradeoffs for ssd performance." in *USENIX Annual Technical Conference*, vol. 57, 2008.

[8] N. Subramanian and A. Jeyaraj, "Recent security challenges in cloud computing," *Computers & Electrical Engineering, Elsevier*, vol. 71, pp. 28–42, 2018.

[9] H. Li, C. Yang, and J. Liu, "A novel security media cloud framework," *Computers & Electrical Engineering*, 2018.

[10] M. Younas, D. N. Jawawi, I. Ghani, T. Fries, and R. Kazmi, "Agile development in the cloud computing environment: A systematic review," *Information and Software Technology*, 2018.

[11] P. Hofmann and D. Woods, "Cloud computing: the limits of public clouds for business applications," *IEEE Internet Computing*, vol. 14, no. 6, pp. 90–93, 2010.

[12] C. Rong, S. Nguyen, and M. Jaatun, "Beyond lightning: A survey on security challenges in cloud computing," *Computers and Electrical Engineering*, vol. 39, no. 1, pp. 47–54, 2013.

[13] T. Laszewski and P. Nauduri, "Migrating applications to the cloud," in *Migrating to the cloud: Oracle client/server modernization.* Elsevier, 2011, ch. 8, pp. 181–208.

[14] L. F. Bittencourt, A. Goldman, E. R. Madeira, N. L. da Fonseca, and R. Sakellariou, "Scheduling in distributed systems: A cloud computing perspective," *Computer Science Review*, vol. 30, pp. 31–54, 2018.

[15] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[16] W. A. Bhat, "Bridging data-capacity gap in big data storage," *Future Generation Computer Systems*, 2018.

[17] T. Laszewski and P. Nauduri, "Migrating applications to the cloud," in *Migrating to the cloud: Oracle client/server modernization.* Elsevier, 2011, ch. 7, pp. 155–179.

[18] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, and R. Buyya, "Network-centric performance analysis of runtime application migration in mobile cloud computing," *Simulation Modelling Practice and Theory*, vol. 50, pp. 42–56, 2015.

[19] I. Cores, M. Rodríguez, P. González, and M. J. Martín, "Reducing the overhead of an mpi application-level migration approach," *Parallel Computing*, vol. 54, pp. 72–82, 2016.

[20] U. Deshpande and K. Keahey, "Traffic-sensitive live migration of virtual machines," *Future Generation Computer Systems*, vol. 72, pp. 118–128, 2017.

[21] A. Yazidi, F. Ung, H. Haugerud, and K. Begnum, "Effective live migration of virtual machines using partitioning and affinity aware-scheduling," *Computers & Electrical Engineering*, vol. 69, pp. 240–255, 2018.

[22] T. Wood, "Improving data center resource management, deployment, and availability with virtualization," Ph.D. dissertation, University of Massachusetts Amherst, 2011, open Access Dissertations.

[23] P. Teli, M. V. Thomas, and K. Chandrasekaran, "Big data migration between data centers in online cloud environment," *Procedia Technology, Elsevier*, vol. 24, pp. 1558–1565, 2016.

[24] C. Infotech. (2017) 9 best practices for successful data migration projects. Accessed: 2018-08-23. [Online]. Available: https://www.cygnet-infotech.com/blog/9-best-practices-for-successful-data-migration-projects

[25] B. Thalheim and Q. Wang, "Data migration: A theoretical perspective," *Data & Knowledge Engineering, Elsevier*, vol. 87, pp. 260–278, 2013.

[26] Apache Software Foundation. (2013) Hadoop cluster. Accessed: 2018-08-24. [Online]. Available: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html

[27] Apache Software. (2013) Mapreduce tutorial. Accessed: 2018-08-23. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#MapReduce+-+User+Interfaces

[28] S. Ghemawat, H. Gobioff, and S.-T. Leung, *The Google file system*. ACM, 2003, vol. 37, no. 5.

[29] P. GARG, "Performance enhancement of big data processing in hadoop map/reduce," Ph.D. dissertation, INDIAN INSTITUTE OF TECHNOLOGY BOMBAY MUMBAI, 2014.

[30] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 63–74, 2011.

[31] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[32] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 50–61, 2011.

[33] B. Cho and I. Gupta, "Budget-constrained bulk data transfer via internet and shipping networks," in *Proceedings of the 8th ACM international conference on Autonomic computing*. ACM, 2011, pp. 71–80.

[34] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," *IEEE Network*, vol. 29, no. 5, pp. 36–42, 2015.

[35] F.-H. Liu, Y.-R. Liou, H.-F. Lo, K.-C. Chang, and W.-T. Lee, "The comprehensive performance rating for hadoop clusters on cloud computing platform," *International Journal of Information and Electronics Engineering*, vol. 4, no. 6, p. 480, 2014.

[36] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 922–933, 2009.

[37] J. Dittrich, J.-A. Quiané-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, "Hadoop++: making a yellow elephant run like a cheetah (without it even noticing)," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 515–529, 2010.

[38] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[39] F. N. Afrati and J. D. Ullman, "Optimizing joins in a map-reduce environment," in *Proceedings of the 13th International Conference on Extending Database Technology*. ACM, 2010, pp. 99–110.

[40] D. Jiang, A. K. Tung, and G. Chen, "Map-join-reduce: Toward scalable and efficient data analysis on large clusters," *IEEE transactions on knowledge and data engineering*, vol. 23, no. 9, pp. 1299–1311, 2011.

[41] M. Y. Eltabakh, Y. Tian, F. Özcan, R. Gemulla, A. Krettek, and J. McPherson, "Cohadoop: flexible data placement and its exploitation in hadoop," *Proceedings of the VLDB Endowment*, vol. 4, no. 9, pp. 575–585, 2011.

[42] H. Liu and D. Orban, "Gridbatch: Cloud computing for large-scale data-intensive batch applications," in *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*. IEEE, 2008, pp. 295–305.

[43] D. Flanagan, *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.

[44] G. E. Blelloch, "Scans as primitive parallel operations," *IEEE Transactions on computers*, vol. 38, no. 11, pp. 1526–1538, 1989.

[45] R. Ladner and M. Fischer, "Parallel prefix computation," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 831–838, 1980.

[46] L. G. Valiant, "A bridging model for parallel computation," *Communications of the ACM*, vol. 33, no. 8, pp. 103–111, 1990.

[47] W. D. Gropp, W. Gropp, E. Lusk, A. Skjellum, and A. D. F. E. E. Lusk, *Using MPI:*

portable parallel programming with the message-passing interface. MIT press, 1999, vol. 1.

[48] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. R. Ganger, E. Riedel, and A. Ailamaki, "Diamond: A storage architecture for early discard in interactive search." in *FAST*, vol. 4, 2004, pp. 73–86.

[49] E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle, "Active disks for large-scale data processing," *Computer*, vol. 34, no. 6, pp. 68–74, 2001.

[50] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[51] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the grid," *Grid computing: Making the global infrastructure a reality*, pp. 299–335, 2003.

[52] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system." IEEE Publishing, 2010, pp. 1–10.

[53] J. Bent, D. Thain, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny, "Explicit control in the batch-aware distributed file system." in *NSDI*, vol. 4, 2004, pp. 365–378.

[54] A. Fox, S. Gribble, Y. Chawathe, E. Brewer, and P. Gauthier, "Cluster-based scalable network services," in *Proceedings of the sixteenth ACM symposium on operating systems principles*, ser. SOSP '97. ACM, 1997, pp. 78–91.

[55] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services," *Proceedings (CD-ROM) of the 7th USENIX conference on Networked systems design and implementation*, 2010.

[56] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, and C. He, "Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing," *Computing, Springer*, vol. 98, no. 3, pp. 303–317, 2016.

[57] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, and G. P. Nolan, "Computational solutions to large-scale data management and analysis," *Nature Reviews Genetics*, pp. 647–657, 2010.

[58] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[59] OpenStack. (2018) Get started with openstack. Accessed: 2018-11-22. [Online]. Available: https://docs.openstack.org/install-guide/get-started-with-openstack.html

[60] QEMU. (2017) Features/kvm. Accessed: 2019-01-21. [Online]. Available: https://wiki.qemu.org/Features/KVM

[61] The Apache Software Foundation. (2017) Apache cloudstack. Accessed: 2019-03-13. [Online]. Available: https://cloudstack.apache.org/

[62] OpenNebula. (2019) Opennebula. Accessed: 2019-03-13. [Online]. Available: https://opennebula.org/

[63] Rackspace Cloud Computing. (2018) Devstack. Accessed: 2019-03-13. [Online]. Available: https://docs.openstack.org/devstack/latest/

[64] A. Pandey, R. K. Thulasiram, and A. Thavaneswaran, "A mapreduce based algorithm

for data migration in a private cloud environment," *8th International Conference on Cloud Computing: Services and Architecture (CLOUD 2019)*, pp. 195–212, 2019.