A MAPREDUCE BASED ALGORITHM FOR DATA MIGRATION IN A PRIVATE CLOUD ENVIRONMENT

Anurag Kumar Pandey, Ruppa K. Thulasiram, and A. Thavaneswaran*

Department of Computer Science, University of Manitoba, Winnipeg, Canada
*Department of Statistics, University of Manitoba, Winnipeg, Canada

ABSTRACT

When a resource in a data center reaches its end-of-life, instead of investing in upgrading, it is possibly the time to decommission such a resource and migrate workloads to other resources in the data center. Data migration between different cloud servers is risky due to the possibility of data loss. The current studies in the literature do not optimize the data before migration, which could avoid data loss. MapReduce is a software framework for distributed processing of large data sets with reduced overhead of migrating data. For this study, we design a MapReduce based algorithm and introduce a few metrics to test and evaluate our proposed framework. We deploy an architecture for creating an Apache Hadoop environment for our experiments. We show that our algorithm for data migration works efficiently for text, image, audio and video files with minimum data loss and scale well for large files as well.

KEYWORDS:

Cloud Computing, Private Cloud, Data Migration, MapReduce, Data Loss, Cost

1. INTRODUCTION

Cloud computing is an environment that enables resource sharing irrespective of the location of the user. 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 connects directly to that hardware and allows to split one system into separate, distinct, and secure environments known as virtual machines (VMs). Thus cloud computing is used to provisioning of various services like Infrastructure as a Service, (IaaS), Software as a Service (SaaS) , Platform as a Service (PaaS) or Anything as a Service (XaaS) being offered to an individual organization. The above classification is based on the type of service, while public, private, hybrid or community cloud are cloud classification based on its deployment model [1].

There are multiple benefits of cloud [2] such as Elasticity, Cost Saving, Accessibility and Reliability. The public cloud, for example, represents a set of standard resources of varying types that can be combined to build applications [3] and the services are offered to clients for different purposes such as storage of files, etc. Cloud enables users to get computing resources/services over the internet irrespective of the location from a remote network of servers [4].

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

Natarajan Meghanathan et al. (Eds) : CCSEA, CLOUD, SIPRO, AIFU, SEA, DKMP, NCOM - 2019
pp. 195-212, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90916

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 [5].

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.

There are multiple approaches discussed in literature as presented later in the related work section. 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.

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. We are 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 and contribution from this research is:

1) Migrating the data efficiently without any loss in the data or minimum loss of data.
2) Designing an algorithm to reduce the time taken to migrate the data between the servers over the cloud.
3) Studying the scale effect by migrating a large amount of data in a short span of time.

1.1 Data Migration

Data migration refers to the process of moving data, applications or other business elements from an organization's onsite systems to the cloud, or moving them from one cloud environment to another system. There are different categories of data migrations in an organization through cloud computing. One of the most common category of data migration is the transfer of data and applications from a client's server to the public cloud. Another common category of data migration is the data migration 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. The transfer may also be performed between two different platforms of a cloud and this is known as cloud to cloud migration. Data migration might also takes place from a cloud server to a local server or data center.

1.1.1. Types of Data Migration

There are various types of data migration that takes place over a cloud system. Most important types are briefly described below.

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 [6]. 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. 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 the process of migrating data from one database to another. This is performed at times where there may be a necessity to shift from one database vendors to another, upgrading the software of the database or move the database to the cloud [7]. 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.

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 [8]. It may also include migrating the applications' 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 might interact with other applications, and every migration has its own data model. Usually, applications are not migrated since tools in the management and configuration in the virtual machines might change between different environments and due to change in operating system. 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, which may lead to data theft or loss. Protection of sensitive data is important in cloud environment.

1.1.2. 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 [9]. With no capability to control and predict the migration time control, the management tasks will not be able to attain the expected performance. 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 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 [10]. 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 [11]. The bandwidth is one of the major implications among those. The authors [11] 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. 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. [12], 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.

1.2 Hadoop 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 [13]. The framework takes care of scheduling tasks, monitoring them and re-executing any unsuccessful tasks. According to the Apache Software Foundation [13], the primary objective of MapReduce is to split the input data set into independent chunks that are processed in a completely parallel manner.

From Figure 1, it can be seen that MapReduce contains two main functions known as Map and Reduce. The Map function converts the input data into intermediate Key / Value Pairs (KVP) 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).

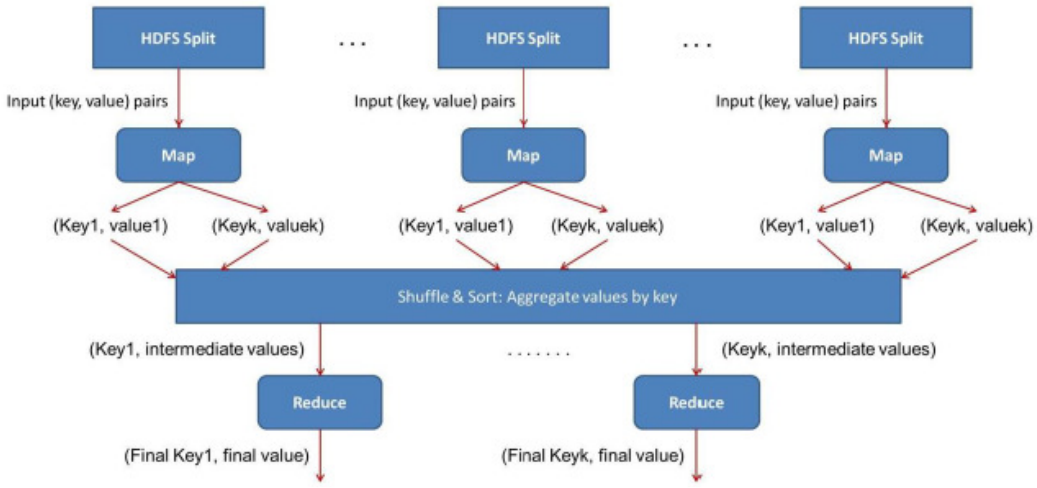

Figure 1: Basic functionality of MapReduce [14]

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) [15]. It assigns a map task for each split when functions related to the users are recorded for every split. Having a lot 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. 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 of. 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. When there are more than one reducer, 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 works well by utilizing a hash function which stores keys.

Rest of the paper is organized as follows: In Section 2 we discuss some of the closely related works. We describe our architecture and experimental set up in Section 3. In Section 4 we explain our algorithm and MapReduce implementation flow. We introduce three metrics for performance of our MapReduce alogithm and analyze our results in Section 5. We conclude our study in Section 6.

2. RELATED WORK

In this section, we discuss some of the approaches related to the data migration. The literature consists of various approaches such as DCTCP [16], D2TCP [17] and D3 [18] 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 [19] 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 volume of data.

Various technologies like elastic optical networks and DC networks have been discussed by Lu et al. [20] 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 [20] 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. We have discussed in this section briefly about Hadoop, geo distributed data centers and energy efficiency. Liu et al. [21] used Hadoop clusters to implement the MapReduce for cloud computing applications. According to these 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 [21].

HadoopDB: There has been a lot of research studying correlating (related) data into similar nodes. HadoopDB saves information in a localized database management system and hence interrupts the dynamic scheduling and fault tolerance of Hadoop. According to Dittrich et al. [22], the two input files are grouped in Hadoop 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 a gap in the performance among Hadoop and parallel databases. 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. [23]. It was discovered that indexing the map significantly enhanced Hadoop's execution.

GridBatch [24] 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 e.g. IBM DB2, TeraData, and Aster Data tables are co-divided, and the inquiry analyzer 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 [25]. 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 [26] and some MPI primitives [27] 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 in parallel and to give straightforward adaptation of the fault tolerance. The locality optimization draws its motivation from techniques such as active

disks [28], 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 [29]. The main weakness of a simple enthusiastic scheduling is that if a given task causes failures repeatedly, the whole processing fails to complete. Few instances of this problem have been fixed in this technique to skip the bad records. 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 [38]. The data sorting which is a part of the MapReduce library is similar to the operation of Now-Sort [30]. 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. [31] 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. [32]. Like MapReduce, it depends on re-execution as a system for actualizing adaptation to internal failure

Geo-distributed cloud services contain many data centers spread across different locations. They can provide larger capacities to the end users and they are mainly used for social media applications [33]. According to [33], there are challenges like storing and migrating the data over long distances. An efficient framework has been proposed by Microsoft Team [33], 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 [33], 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. [34] 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 [34]. The proposed algorithm by Li et al. [34] 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. [35] 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.

Each of these approaches 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. Our focus and objective is to build such a system framework that would efficiently migrate data using Map Reduce and avoid data loss.

3. EXPERIMENTAL SETUP AND ARCHITECTURE

We describe in this Section various modules of our architecture depicted in Figure 2.

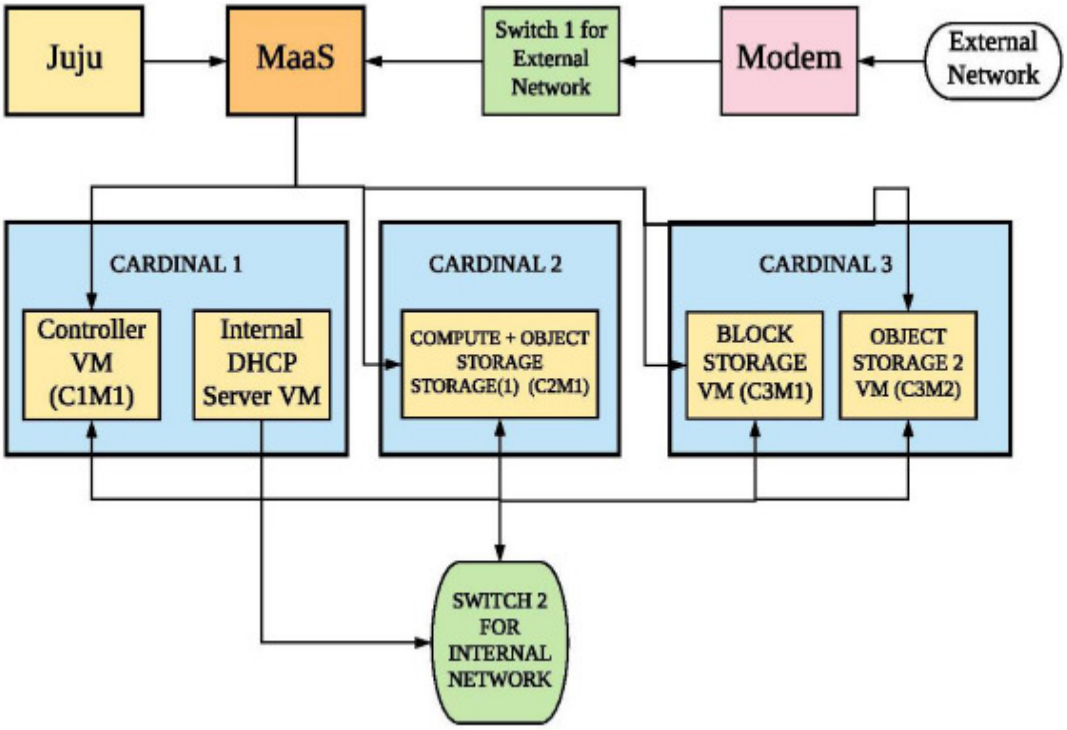

Figure 2. Cloud Architecture

3.1 OpenStack

The OpenStack project is an open source cloud computing platform for all types of clouds [36]. The purpose of using this open source software 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 IaaS solution through a group of associated services [36]. Each service provides an application programming interface (API) to facilitate 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 virtualized infrastructure as it provides multiple hypervisors such as Kernel Virtual Machine (KVM), Qemu and Hyperv. KVM is a Linux kernel module that allows a user space program to utilize the hardware virtualization features of various processors [37]. 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 [37].

Several components contribute in building an OpenStack based Cloud. For this experiment, we have installed Nova compute, Glace, Cinder, Swift, Horizon, Keystone and Neutron.

3.2 MAAS

Metal as a Service (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 services running on those machines.

Minimum Requirements for MAAS

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

Factors that influence hardware specifications include: a) the number of connecting clients (client activity); b) the manner in which services are distributed; c) whether high availability is used; d) whether load balancing is used; and e) the number of images that are stored.

3.3 JAAS

Juju as a Service (JAAS), is the best way to quickly model and deploy cloud-based applications. 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.

3.4 Building a Testbed

There are various ways of deploying the cloud. We have deployed a version of Openstack Pike using MAAS and JAAS as shown in Figure 2.

To build a private Cloud, we 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. We 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 created on Cardinal 1 which is working as internal DHCP server to allocate Provider IP’s. We have used OpenStack Pike to create a private Cloud environment for these machines. OpenStack is a Cloud software platform with a three node architecture [36] as shown in Figure 2. OpenStack should have minimum three nodes to implement Cloud but to get more resources more number of compute nodes can be added. There can be only one controller and network nodes each 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.

In this figure 2, (a) MaaS is acting as DHCP server for external network; (b) MaaS provides management IPs; (c) Switch 2 provides IPs for internal network; (d) all cardinals have Ubuntu 18.04 LTS; (e) All VM’s have Ubuntu 16.04 LTS.

3.5 Hadoop MapReduce Implementation

Data migration being a complex function, the data has to be optimized to make the process simpler. Hence, we use MapReduce, a model that optimizes the data, for my experiment. It is a programming model that processes big data sets. We have made two VMs on the compute nodes in our 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. We 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. With the help of data migration, we show that using MapReduce for migration, reduces data loss as well as improves the efficiency of the migration.

4. EXPERIMENTS

4.1 Implementation Flow

Figure 3 presents the implementation process, and is explained below.

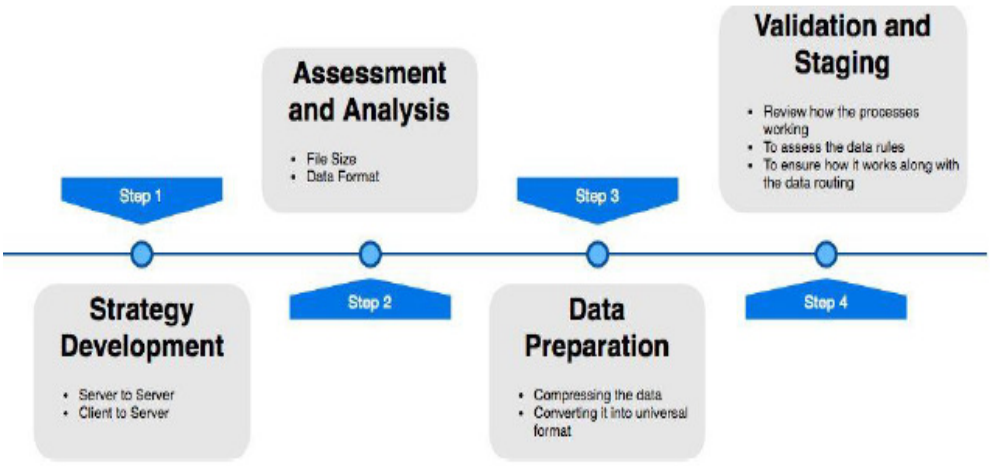

Figure 3. Implementation Flow

4.1.1.Strategy Development

The strategy development process for data migration can be chosen from different strategies based on the needs and available processing windows. The strategy depends on the following two criteria:

1) Data migration from server to server (with administrative permission): in this process, the data is migrated of 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.

2) Data migration from client to server (With administrative permission): in this process, the data will be migrated from one client platform to another (such as moving from client A to server B in Cloud environment), also our client as well as server will have root or administrator access, which will allow to have full control over the setup and configuration of the server.

4.1.2. Assessment and Analysis

Two important parameters for a file are its size and format. We assessed and analyzed the performance of our MapReduce algorithm based on these two input parameters. The file size helps in computing data loss. In order to assess the performance of the migration process, different files of sizes (in MB) were considered. The data format help us in identifying what type of data is to be transferred during migration. For our experiments, the data file format is in csv, excel, images, audio, or video.

4.1.3. Data Preparation

During data migration from one server to another, a large amount of data is transferred, in general. In our experiment, data is transferred from Server A to Server B. If any data gets corrupted during the transfer, it is difficult to identify the location and directory of the corrupted file. Hence, prior to migration the data must be optimized for easier transfer of data. For this optimization process, we used the MapReduce framework. The final output (.mat file) is converted into PDF or another universal format (.rar) to ensure the security and privacy level of data.

4.1.4 Validation

The migration process performance is validated for ensuring the requirements and customized settings function. The validation and performance analysis process covers the following features: (a) review the process flow; (b) assess the data rules; (c) to ensure proper working of the process along with the data routing. The following parameter setting were used to achieve these features.

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

The efficiency and total execution cost are computed based on the above parameters to determine the performance of the algorithm, which we discuss later in the results section.

4.2 MapReduce Implementation

Our 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 open source tool and easily available and hence, we use this for our study.

For data migration, we 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 three 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. These Keys that have similar data are combined by merging the data sizes in the Value and storing it in a single array. Since, the KVP might have more than one row or column, it will be stored as a 2D array.

Algorithm: MAPRDUCE DATA MIGRATION

Input: Image, Audio, Video, Excel files
Output: (Key, Value) Pairs
Initialization:
  Mapper (INP, INF_VL, IN_K_VL)
    IMV = Data Fragmentation Condition (INP)
    Add IMV to IN_K_VL
Return IN_K_VL
Reducer (KY_VL, INT_VLTR, OT_K_VL)
  While HASNEXT (INT_VLTR)
    OT= GETNEXT (INT_VLTR)
  End While
  Add OT to OT_K_VL
Return OT_K_VL
Migrate (Server 1 ⟶ 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);
End

The Map Reduce data migration algorithm takes different types of input such as audio, video, images and csv 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 IN_K_VL object.

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, map reduces provides a new key from intermediate Key Value store (IN_K_VL). 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. 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.

5. RESULTS AND DISCUSSIONS

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 that is now in the Server B. This new KVP is compared with the previous KVP to find if the values are same. Any mismatch would mean 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.

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.

Data Loss: D_L

Total amount of data before migration: D_{BM}
Total amount of data after migration: D_{AM}

$$D_L = D_{BM} - D_{AM}$$

We have performed the experiments and various performance evaluation 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. We evaluated the cost of migration for the proposed model and also evaluated the cost for migrating data without using MapReduce. Execution cost was calculated by adding the cost incurred during the idle time with the cost incurred to execute the work flow schema.

Total Execution Cost: E_C

Idle Time: I_T
Busy Time: B_T
Information that is stored after transfer: λ
Data that is transferred over the network: γ

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

Efficiency: E

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

Total Data Transferred: DT
Data Re-transferred: DRT

$$E = \frac{DT - DRT}{DT} * 100$$

TABLE 1 : INPUT FILE FORMAT: Audio and Video Files (.wav, .mp4)			
File Size (In MB)	Total Execution Cost (In second)	Data Loss (In MB)	Efficiency
10	9.789856	1.07	90.674%
100	41.099143	6.98	91.986%
400	90.371984	19.91	94.793%
700	159.12896	34.08	95.168%
1100	242.95312	41.09	96.023%

Table 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 big data sets, the algorithm performs better. The data loss also reduces with increasing size. This concludes that our algorithm performs efficiently in case of audio video migration for scaled data size.

Table 2 presents the performance of our algorithm on image data files. The Map and Reduce function for image migration uses 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 big data sets, the algorithm performs better. The image being static (still), performs better than audio video files. The data loss is low with increasing size. Hence it can be conclude that the algorithm performs well in case of image migration also with respect to scaling data sizes.

TABLE 2: INPUT FILE FORMAT: Image Files (.png, .jpg, .tif)			
File Size (In MB)	Total Execution Cost (In second)	Data Loss (In MB)	Efficiency
10	8.132984	0.99	92.147%
100	39.227144	5.20	93.997%
400	87.994872	17.80	17.80
700	146.297184	20.73	95.778%
1100	223.224165	28.07	96.814%

TABLE 3: INPUT FILE FORMAT: Excel Files (.csv, .xlsx)			
File Size (In MB)	Total Execution Cost (In second)	Data Loss (In MB)	Efficiency
10	7.193986	0.93	93.674%
100	33.938971	3.71	94.986%
400	71.392817	13.77	94.793%
700	138.029641	20.73	96.168%
1100	194.837194	28.07	97.023%

Table 3 presents the performance of our algorithm for textual data files in csv or xlsx format. The Map and Reduce function for document migration uses keywords as key values. This performs better than other data formats mentioned before. 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 big 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 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.Analyzing the results in Figure 4, 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 provide by Matlab is a benefit over other programming languages as it helped in optimizing the datasets.

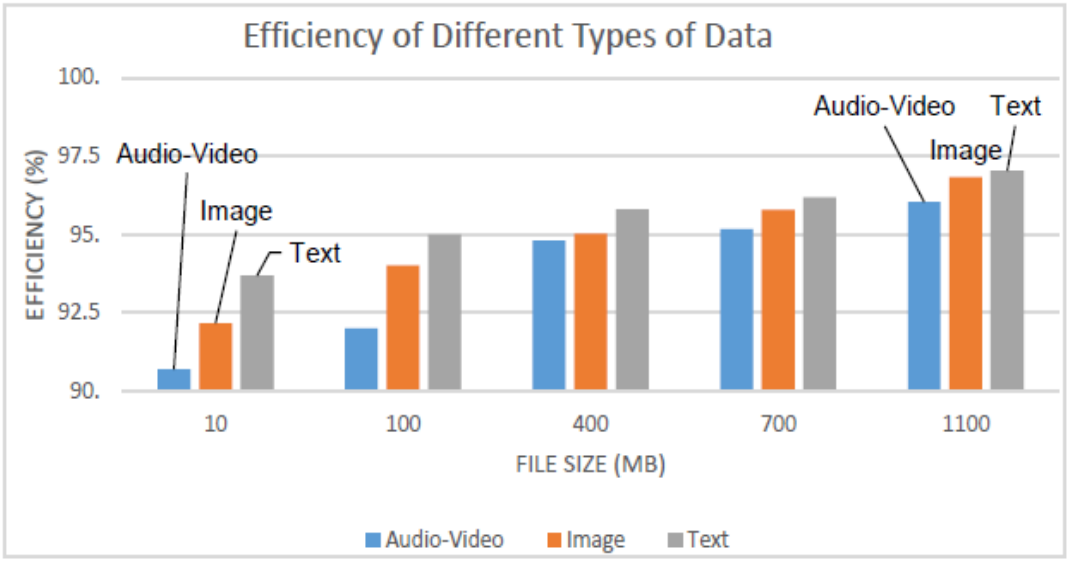

Figure 4. Efficiency Analysis

6. CONCLUSIONS

A migration technique has been discussed in this study that can efficiently transfer the data between the servers using MapReduce. The framework that has been developed for migrating the data has reduced the data loss. The MapReduce has efficiently optimized the data migration code when large data sets are considered for migration. The performance metrics introduced have been

validated by computing the efficiency and the cost for migration. From the results, it can be seen that the efficiency of the proposed algorithm increases with the increasing file size, that is, we established the scalability of the proposed algorithm. Also, the cost of the transfer is shown to be reduced.

The data migration technique used for the experiment performs faster than the previously available work in cloud computing. The work has been performed 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, internet interruptions, failure of a server, less bandwidth and less frequency have been the issues for data loss in the experimental results reported. With stable network connectivity, we expect that the data loss could be avoided.

Larger heterogeneous files (text, images, audio and videos) have been migrated by our algorithm for 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. By improving the configuration of the system architecture and physical servers we might expect to improve the performance further. The efficiency of our algorithm cannot be guaranteed for input files with complex data (features of the input files) and other file formats, hence, improving the algorithm for such input files is left as future work.

ACKNOWLEDGEMENT

The last two authors acknowledge the partial funding from the Faculty of Science Inter-disciplinary research collaboration initiation grant and discovery grant from Natural Sciences and Engineering Research Council (NSERC).

REFERENCES

[1] N. Subramanian and A. Jeyaraj, “Recent security challenges in cloud computing,” Computers and Electrical Engineering, Elsevier, vol. 71, pp. 28–42, 2018.
[2] 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
[3] 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
[4] 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.
[5] 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.
[6] 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
[7] 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
[8] 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
[9] U. Deshpande and K. Keahey, “Traffic-sensitive live migration of virtual machines,” Future Generation Computer Systems, vol. 72, pp. 118–128, 2017.
[10] T. Wood, “Improving data center resource management, deployment, and availability with virtualization,” Ph.D. dissertation, University of Massachusetts Amherst, 2011, open Access Dissertations.

[11] P. Teli, M. V. Thomas, and K. Chandrasekaran, "Big data migration between datacenters in online cloud environment," Procedia Technology, Elsevier, vol. 24, pp. 1558–1565, 2016.
[12] B. Thalheim and Q. Wang, "Data migration: A theoretical perspective," Data and Knowledge Engineering, Elsevier, vol. 87, pp. 260–278, 2013.
[13] Apache Software. (2013) Mapreduce tutorial. Accessed: 2018-08-23. [Online]. Available:https://hadoop.apache.org/docs/r1.2.1/mapredtutorial.html#MapReduce+-+User+Interfaces
[14] S.Kumar. (2018) What is Hadoop map reduce and how does it work? Accessed: 2018-08-22. [Online]. Available: https://qph.fs.quoracdn.net/main-qimg-82813242aa853b91a91b96134f0c5c13-c
[15] 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
[16] 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.
[17] 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.
[18] 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.
[19] 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, 2011, pp. 71–80.
[20] 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.
[21] 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.
[22] J. Dittrich, J.-A. Quian´e-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
[23] 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.
[24] H. Liu and D. Orban, "Gridbatch: Cloud computing for large-scale data-intensive batch applications," in Cluster Computing and the Grid, (CCGRID). 8th IEEE Intl. Symposium on. IEEE, 2008, pp. 295–305.
[25] R. Ladner and M. Fischer, "Parallel prefix computation," Journal of the ACM (JACM),vol. 27, no. 4, pp. 831–838, 1980.
[26] L. G. Valiant, "A bridging model for parallel computation," Communications of the ACM, vol. 33, no. 8, pp. 103–111, 1990.
[27] W. D. Gropp, E. Lusk, A. Skjellum, and A. Lusk, Using MPI: portable parallel programming with the message-passing interface. MIT press, 1999, vol. 1
[28] 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
[29] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
[30] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system." IEEE 26th Symposium on Mass Storage Systems and Technologies, 2010, pp. 1–10.
[31] 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 Networked System Design & Implementation, vol. 4, 2004, pp. 365–378
[32] 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.
[33] 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

[34] 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.
[35] 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 J. on Selected Areas in Communications, vol. 31, no. 12, pp. 2710–2721, 2013.
[36] OpenStack. (2018) Get started with openstack. Accessed: 2018-11-22. [Online].Available: https://docs.openstack.org/install-guide/get-started-with-openstack.html
[37] QEMU. (2017) Features/kvm Accessed: 2019-01-21. [Online]. Available: https://wiki.qemu.org/Features/KVM
[38] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the grid," Grid computing: Making the global infrastructure a reality, pp. 299–335, 2003.

AUTHORS

Mr. Anurag Pandey is a graduate student with the Department of Computer Science, University of Manitoba, Winnipeg, Manitoba. He is currently completing his MSc program. He has previously worked as a Software Engineer. His research interests include Cloud Computing, Amazon Web Services, and Big Data (Infrastructure).

Dr. Ruppa K. Thulasiram (Tulsi) is a Professor with the Department of Computer Science, University of Manitoba, Winnipeg, Manitoba. He received his Ph.D., from Indian Institute of Science, Bangalore, India and spent years at Concordia University, Montreal, Canada; Georgia Institute of Technology, Atlanta; and University of Delaware as Post-doc, Research Staff and Research Faculty respectively before taking up a position at University of Manitoba. Tulsi's current research interests include Computational Finance, Cloud Computing, Blockchain Technology for Financial Applications and related areas. He has written many papers in the areas of High Temperature Physics, Gas Dynamics, Computational Finance, Grid/Cloud computing, Computational Intelligence and Blockchain Applications research areas. He has supervised many MSc and PhD theses and graduated many students. He has also received many best paper awards in reputed conferences. He holds a patent for true random generators along with students and colleagues. Tulsi has developed a curriculum for cross-disciplinary computational finance area as well as on Cloud Computing at University of Manitoba for both graduate and senior undergraduate level and has been teaching for the past several years. Tulsi has organized many conferences and has been editor and guest editor with many journals. He is associated with many professional societies such as IEEE, ACM, ASAC etc.

Dr. Aerambamoorthy Thavaneswaran (Thava) is a Professor with the Department of Statistics, University of Manitoba, Winnipeg, Manitoba. He received his M.Math. and Ph.D.in statistics from University of Waterloo. Thava's research interests include Inference for stochastic processes, Estimating Functions, Nonlinear Time Series, Filtering, Smoothing, Empirical Financial Time series modelling, Mathematical Finance: option pricing/bond pricing, Fuzzy Methods in Finance and Financial Risk Forecasting. He has supervised many MSc and PhD theses and graduated many students at University of Manitoba, Temple University and University of Malaya. He has also published more than 100 papers having more than 1000 citations. Thava has been an associate editor and guest editor with many journals. He is associated with statistical society of Canada and International Statistical Institute (ISI) as an elected member.