A Network-aware Virtual Machine Placement approach for Data-intensive Applications in a Cloud environment

By

Yasser Alharbi

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

2017

Dedicated to my dear parents for their love and blessing. To my beloved wife (Shadeah) for her support and encouragement and to my loving daughter (Ilan) for making my life beautiful.

Acknowledgements

Completing this PhD thesis has been a journey. The experiences have been valuable and educational in academic and personal ways. It is my pleasure to thank those who have made this thesis possible and the journey interesting.

First and foremost, I would like to thank Allah, who has given me the strength and patience to carry out this work.

I would like to thank my supervisor, Professor Stuart D. Walker, for his time, effort, guidance, support and feedback throughout my PhD. For this I am truly grateful. I am truly glad to have had you by my side.

Many thanks to my colleagues and friends in Colchester for all the interesting (academic and non-academic) discussions and interactions we hadin particular, Dr Ibrahim Kabir, Dr Malik Alrashidi, Dr Abdullah Alzahrani, Dr Mohammed Alrshoodi, Dr Sami Alharthi and Dr Abdullah Alzharani. All this time would not have been so enjoyable without your company.

Special thanks also goes to Jing Piao, who helped me to build and configure my system in the CloudSim simulator.

Finally, I am grateful to my parents, brothers and sisters, who have constantly encouraged me to pursue my goals. I am also grateful to my wife for everything. With her, this PhD journey was a sweet dream and a pleasant experience.

Abstract

Cloud computing provides beneficial services to users, enabling them to share large amounts of information, employ Storage Nodes (SN), utilise Computing Nodes (CN) and gather knowledge for research. Virtual Machines (VMs) usually host data-intensive applications, which submit thousands of jobs that access subsets of the petabytes of data distributed over Clouds Datacentres (DCs). The VMs scheduling allocation decisions in cloud environments are based on different parameters, such as cost, resource utilisation, performance, time and resource availability. In the case of application performance, the decisions are often made on the basis of jobs being either data intensive or computation intensive. In data-intensive situations, jobs may be pushed to the data; in computation-intensive situations, data may be pulled to the jobs. This kind of scheduling, in which there is no consideration of network characteristics, can lead to performance degradation in a cloud environment and may result in large processing queues and job execution delays due to site overloads.

This thesis proposes a novel service framework, the **n**etwork- **a**ware VM placement approach for **d**ata- **i**ntensive applications (NADI), to address the need for improved application performance . NADI takes into account a jobs time cost based on a mechanism that maps VMs against the resources when making scheduling decisions across multiple DCs. So, it not only allocates the best available resources to a VM to minimise the time needed to complete its jobs but also checks the global state of jobs and resources so that the output of the whole cloud is maximised.

The thesis begins with a statement of the problem addressed and the objectives of the research. The methodology adopted for the research is described subsequently, and the outline of the thesis is presented. This is followed by a brief introduction highlighting the current approaches in VM placement and migration in cloud computing. Next, this thesis presents a framework for the proposed NADI with a description of its various components and enabling functionalities, which are required to realise this framework. Multi-objective strategies suitable for the problems in NADI are presented.

Novel algorithms for managing applications and their data are proposed; they aim to improve each jobs performance and minimise the traffic between the application and its related data. The results indicate that there are considerable performance improvements and that the completion time is reduced by 25% to 51%, which can be gained by adopting the NADI scheduling approach.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AWS Amazon Web Services

BW Balanced Workloads

CIW Computationally Intensive Workloads

CN Computing Nodes

CSS Cloud Scheduler Service

DCs Datacentres

DLS Data Location Service

DMM Data movement Manager

DMS Data Management Service

DS Discovery Service

DSC Data Scheduler

DVS distributed virtual switch

EC2 elastic compute cloud for amazon

FF First Fit

FFD First Fit decreasing

GH Greedy Heuristic

GIS Geographic Information System

HPC high performance Computing application

IaaS Infrastructure-as-a-Service

IS Information Service

IT information technology

ITU International telecommunications union

JSCS Job Submission and Control Services

LDS Logical Dataset

LDM Local Disk Manager

LFN Logical File Name

LAN local area network

NIST National Institute for Standards and Technology

NMS Network Management Service

NPC NP-Complete

NPH NP-Hard

OS Operating System

PC Personal Computer

PIP Physical Infrastructure Providers

SLA Service Level Agreement

SN Storage Nodes

SOA Service Oriented Architecture

VI virtual infrastructure

VIP Virtual Infrastructure Provider

vLink virtual link

VM virtual machine

WMS Workload Management System

List of Publications

Published papers

1. **Y. Alharbi** and K. Yang, "Optimizing jobs' completion time in cloud systems during Virtual Machine Placement," in *3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, IEEE,pp. 1-6, Muscat, Oman. 2016.

2. **Y. Alharbi** and S. Walker, Data-Intensive, Computing and Network Aware (DCN) cloud VMs Scheduling Algorithm, in *Future Technologies Conference (FTC2016)*. IEEE, pp. 1257-1264. San Francisco, USA. 2016.

3. **Y. Alharbi** and S. Walker, Virtual machine placement using pre-fetching data transfer, *Computing Conference 2017*, IEEE, 2017, pp. 16. London, UK.

1

Introduction

The main objective of this thesis was to design a framework for managing cloud resources in order to ensure an optimised delivery for data-intensive applications in a cloud computing model. For that purpose, the NADI framework was created; it takes the network, size and location of the data and computes the information from various information resources, helping to select the optimum site for job execution. It also includes each CN's CPU loads, and the CN with the least cost is selected. To this end, the novel NADI framework is proposed. Algorithms to evaluate the elements of the proposed framework are presented as well as mathematical models to validate the algorithms.

This chapter gives an overview of the research context to be presented in this thesis and the motivation for developing the proposed model. It discusses a description of the

problem as well as the objectives of the thesis. It then presents the primary contributions of this thesis. The chapter ends with a presentation of the organisation of the rest of this thesis.

1.1 Motivation

The cloud computing model has been very popular due to its flexibility in providing resources efficiently and quickly. In Infrastructure-as-a-Service (IaaS), the VMs serve the resource requests according to the specifications of the underlying infrastructure. If it takes a long time to place the VMs in the underlying infrastructure or if the requests that are accepted cannot get served , then the cloud computing paradigm loses its flexibility. When there is on-demand access to the services in cloud computing, the available infrastructure serves the requested resource within a short time span. One cannot predict the number of resource requests within a certain interval of time for an on-demand request unlike in spot-market access. VMs run on a single host; therefore, when there is on-demand access to serve more requests, the available resources should be used optimally [7].

VMs have arisen as tools that test, manage and consolidate servers as well as support services that can scale from one to as many VMs as required in both public and private DCs [8]. Currently, the size of modern DCs is growing, which provides a good reason to optimise the efficiency algorithms, placing VMs that can reduce operating expenses, defer upgrades, cloud tenants and applications performance . All these options will attract thousands of cloud customers, who compete for the same infrastructure resources; this is because a large data centre costs several hundred million dollars to build and operate. Nowadays, data-intensive applications for processing big data are being hosted in the VMs. Data-intensive workflows may take advantage of infrastructure, which is offered by the surroundings (such as the data cloud) [9, 10].

The data cloud offers services like data replication and low latency transport protocols, which dispense applications that are data intensive and that require the accessing, processing and transferring of huge datasets that are stored in the distributed repositories. The main aim of executing the data-intensive applications on the distributed resources is to optimise the overall time needed for completing the applications workflow. Workflow completion time refers to the overall time taken or required to finish all

the jobs in the workflow. Apart from the completion time, execution cost is another objective function of scheduling the applications . Communication time taken in staging both input and output files as well as the computation time required to execute them determines the completion time. The two objectives affect each other; therefore, when the tasks are scheduled to achieve only one objective, there will be a sub-optimal result [11].

Most of traditional planning algorithms were based on a pull paradigm, whereby the algorithm focused on staging the data into the computation resources. They ignored the data's location, which in turn resulted in a high cost of bandwidth utilisation [12, 13]. Therefore, a dissimilar paradigm is required where the choice of both the compute host and the data host should be considered first instead of selecting the compute host first and then the data host (or vice versa). The cloud setting offers the virtualised resources necessary for computation, and the data-intensive applications need to communicate between the nodes that are computing. Therefore, the data location as well as the VM placement affects the whole computation time. Most of the research that has been reported in the literature review prefers choosing the physical nodes to place data as well as viewing VMs as autonomous problems [14].

1.2 Problem Description and Objectives

Traditionally, the workflow decisions during data-intensive jobs have been made by the transmission of executables to the data while the computation-intensive jobs are transmitting data to the executables . Data-intensive applications, through which several workloads are submitted, allow the analysis of a large dataset that is replicated to globally distributed systems. Where there is no replication of the data to the computation site, the data is generated from the remote sites. The transfer of data from the other site slows the performance of the job [15]. The cloud has a large number of SNs and CNs (each with an unique ID), which supports the fact that if a computing activity runs at a remote site, the output data that is produced is supposed to be transmitted for the analysis of its results remotely by the user. Each data-intensive job has its

own distinct method of data production and consumption. To ensure high performance during job execution and maximisation of the cloud throughput, the computation input and output data should be aligned and scheduled concurrently [16].

These processes communicate effectively, which allows for a minimal data transfer cost and a reduced execution duration of the application (which consists of large dataset). Depending on the characteristics and the potential of the networks, the computing resources and the storage resources, It can be settled on sending data and the executables to the third location . It becomes less challenging when viewed from a local resource management perspective but much more challenging based on the scheduling perspective since it is supposed to generally increase the output of the cloud throughput. Scheduling is considered since it is a process that allows the user to organise a task across several sites. Increasing the complexity to the cloud, by increasing the number of tasks and predominantly geographically dispersed nodes, results in the data transfer service and the schedulers making a contradictory decision compared to the one that is reached from a local scheduling point of view. Therefore, the scheduling technology was invented earlier under the assumption that all systems that work together used the same local area network (LAN) [17]. It can be clearly seen that the limitations of the global decision-making system by extending the scheduling system and the local resource management system over a long distance.

To optimise the tasks performed through a scheduling algorithm, the algorithm is needed and it should consider various factors, such as the number of processors, data transfer, length of the queue and various network relations. Basically, the algorithm needs to allocate various weight values upon computation for each capable location targeted that represents the available network characteristics, queue length, processing cycle, output and input (in which the one with minimal cost is considered). Thus, the scheduling algorithm should then consider the cloud since it is composed of combined active network fundamentals, choosing the network as the first condition in the matrix of the scheduling decision. Based on the network and the strategic view of the whole cloud system, the process of moving data from one point to another is executed as

required by scheduling resources and supervision to ensure job completion.

Thus, optimisation of work distribution is performed, and the job scheduler is supposed to make the right decision by taking into consideration the changes in the clouds system, the total number of tasks, the location and size of the data, and the pool of the processing sequence. Consequently, the result should bring cloud applications and the network together , which will assist in understanding the importance of each other by bringing the applications closer to one another though they were developed independently [15, 18].

Allocating the VMs (data-intensive applications) in the right place can significantly improve jobs performance especially when taking into consideration a combination of the network, data and compute costs for these jobs. This thesis will address the following objectives:

- To investigate the requirements and specifications of building a framework for managing cloud resources.
- To investigate a cloud architectural framework and the functionalities suitable for deploying VMs in a virtual infrastructure.
- To propose multi-objective VM polices suitable for resources organised in a holistic manner.
- To investigate the effectiveness of applications' performance when the VM allocation algorithm includes the network, data and compute costs in scheduling decisions.
- To demonstrate the best benefits to the jobs' performance by having multi-option decisions, which are moving data towards the applications or where both the data and applications are moved toward a third location.

1.3 Thesis Contribution

This thesis proposes an architectural framework for the delivery of next-generation cloud computing services. The model presented is built on the concept of a holistic resource view. It presents provisioning and automation algorithms to validate the proposed framework. The multi-objective optimisation suitable for the model is presented and tested in a simulation environment in the thesis. The main contributions of this thesis can be summarised as follows:

- Proposed and described NADIs various components, actors and roles in the virtual infrastructure. All the architectural components presented are modelled based on the existing matured cloud architectures and standards. Possible market opportunities in the model are identified and related to developments in cloud economics and business models.

- Formulated multi-objective polices suitable for allocating VMs in the Virtual infrastructure. These objectives are aimed at minimising the jobs completion time by calculating the cost time, which is divided into two costs. These costs are the computational cost and the data and network cost.

- Proposed the VM allocation algorithms using multiple objective optimisation that tried to enhance jobs' performance. The proposed algorithms take into consideration the datas location, network status, the CPUs load, and attributes

    - **Y. Alharbi and K. Yang**, Optimizing jobs' completion time in cloud systems during Virtual Machine Placement. **3rd MEC International Conference on Big Data and Smart City (ICBDSC)-2016**. Muscat, Oman. **Published**

- Developed the allocation algorithms by considering the replication service in order to enhance the jobs performance and minimise the traffic between the application and its related data. The replication of data can help improve the scheduling and execution optimisation by reducing the frequency of remote data access.

  - ◇ **Y. Alharbi and S. Walker**, Data-Intensive, Computing and Network Aware (DCN) cloud VMs Scheduling Algorithm, **Future Technologies Conference (FTC2016)**. IEEE, 2016, pp. 18. San Francisco, USA. **Published**

- Proposed an adaptive scheduling algorithm that takes into consideration both the data requirements and the computation requirements of the tasks when making scheduling decisions. In this algorithm, the transfer of data is viewed in par with computation and explicitly considered when scheduling. The tasks are distributed to the optimal sites in terms of computation and transfer time. Additionally, the algorithm overlaps the transfer time of the job with its own queuing time as well as other tasks' computation time.

- Enhanced the jobs execution time by staging the data in and out in advance, which is before it starts to execute.

  - ◇ **Y. Alharbi and S. Walker**, Virtual machine placement using pre-fetching data transfer, **Computing Conference 2017**. IEEE, 2017, pp. 16. London, UK. **Published**

1.4 Thesis Methodology

The approach chosen for this research involves a review of the relevant literature, NADI specifications, design of the cloud framework for NADI, simulation and implementation. The phases involved in the methodology adopted are shown in Figure 1.1. The main reasons why this methodology was used are outlined in this section.

1.4.1 Review of Relevant Literature

Extensive literature relevant to the chosen topic of research was reviewed and classified under cloud computing, related works, DCs networking, virtualisation, closely related technologies, modelling and cloud development. The literature review starts by presenting the key issues and a general overview of networks, cloud computing, service oriented architectures, virtualisation, optimisation problem-solving approaches and related projects. Conference papers as well as journal articles were found using keywords. The relevant papers were categorised into general concepts, modelling and simulation, and enabling technologies. Such classifications of the research papers led to the contributions of prior knowledge . The variables and significant tools relevant to the development of cloud environments were identified in the literature review from various cloud computing projects and publications. The service specifications of the environment (problem-solving) that was studied led the research to identify the requirements of enabling NADI with a heterogeneous network and multi-objective design optimisation. It also enabled the identification of gaps in the Virtual requests of cloud computing requirements. The literature review fulfils the first objective of the research.

1.4.2 NADI Specifications and Design

The literature review enabled the discovery of the specifications as well as the design stage process. The design describes service specifications for the computation, network and storage service as shown in Figure 1.1. The specifications identified the components that are needed to design a framework for the Virtual infrastructure container

using the graphical input interfaces parameters which define a request. Given the relatively new features of cloud computing and its application, the relevant scenarios that can be activated by the chosen research area have been identified in terms of their respective requirements. Part of the description of the scenarios and their requirements have been inspired by relevant literature materials, which were published by the International telecommunications union (ITU) [19] and the National Institute for Standards and Technology (NIST) [20].

1.4.3 Developing a Cloud Framework for NADI

A novel NADI framework was developed to conform to existing cloud computing architectures. The components of the framework were envisioned to provide all the proposed functionalities desired in a Virtual resource request. The specifications developed in sec-

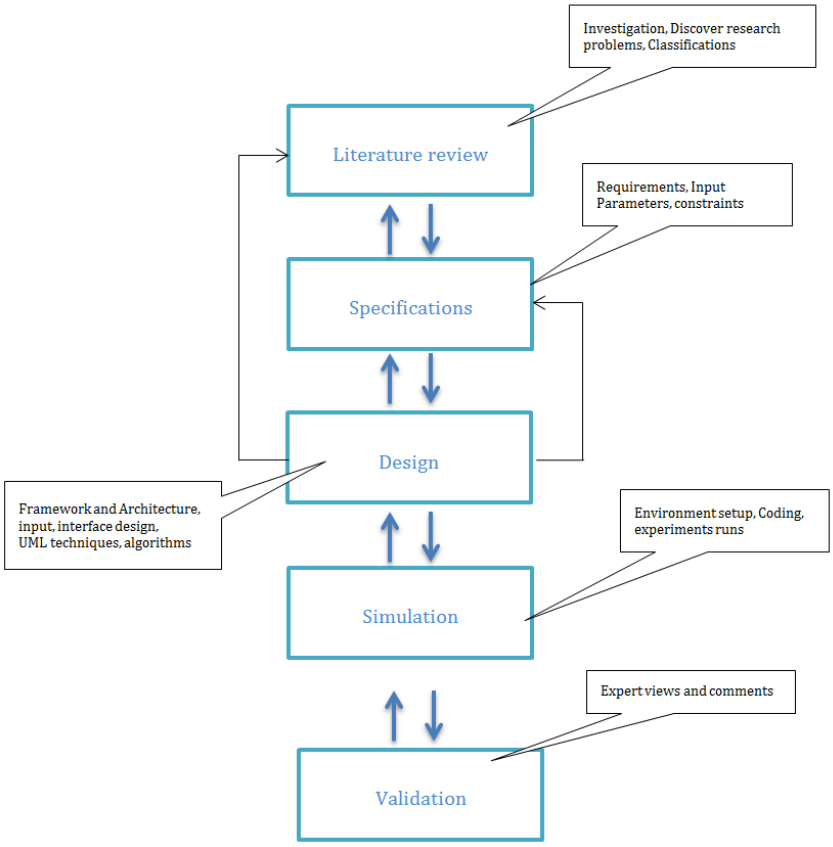


Figure 1.1: Methodology phases

tion 3.2 of chapter three illustrated the building blocks needed to convey a framework to support the specified services for NADI. The framework was developed to provide services and implement the services practically using the modules that are mentioned in section 3.3 to support NADI applications. For the simulation, various experiments have been conducted to demonstrate the ideas proposed in the thesis. Cloud provisioning and allocation, optimisation strategies, and DC allocation scheduling algorithms have been tested in a simulation environment. The concepts proposed in this thesis were applied in solving a problem that requires a large-scale data analysis.

1.5 Thesis Outline

Chapter 2 introduces the cloud computing and various enabling technologies, including networks and virtualisation. Current trends in the design of cloud DCs architecture are presented and explored in more detail. Additionally, the chapter presents the facilities needed to develop a cloud service using modelling and simulation. It also identifies research efforts in VM placement and migration in cloud management. The chapter concludes with a summary of the research efforts in the cloud computing field and identifies the specific areas that the research has focused on.

Chapter 3 outlines the mathematical and theoretical explanation of the NADI scheduling framework and describes the various components, actors and roles in the NADI framework. It has been demonstrated that, with the aid of mathematical equations, a matrix of various scheduling costs can substantially enhance the process of scheduling when every task is submitted and executed after considering particular related costs. In addition, this chapter offers a general overview of the requirements of NADI scheduling and names the salient characteristics of such a system. Of those characteristics, it was noted that the location of the data is the most important consideration for an optimum cloud; therefore, it should be one of the considerations when making scheduling decisions. It was also concluded that the best network path for storage and computing should be acknowledged. Then, the scheduler should incorporate and calculate the measurements of the network when planning to submit jobs.

Chapter 4 proposes the first version of a network-aware VM allocation algorithm for applications that are data intensive in a cloud setting (NADI). Then, it shows the relationships between NADI design components, including the broker, data location service, network monitoring service, information service and discovery service. The algorithms used in this contribution were also presented and explored in greater detail, and simulation tests were performed to examine the NADI algorithm. Through these tests, it was demonstrated that data location, network latency and computational resources can significantly affect application performance. Thus, the VM allocation scheduler should be capable of optimising the data-intensive scheduling process. It is important to control a VM's location so that the applications that are hosted by the VM will have a shorter access time when obtaining the data. It was also demonstrated that the keys to cloud optimisation include network-managed services and a suitable selection of the network links between DC locations before making scheduling decisions. Also, the overall queue and execution times can be significantly reduced if job data requirements and completion times are taken into account. Based on simulation results on CloudSim 3.0, it was also concluded that NADI is better in scalability and consistency compared to other contemporary scheduling approaches.

Chapter 5 describes the second version of NADI by considering computation power and its load, less network traffic and replicas locations. This approach is offered to obtain an optimised completion time of jobs in each VM. The NADI scheduling optimisation approach has been developed to generate the time cost matrix for the overall time needed for jobs completion and optimised among these values. Then, it allocates the VMs to the optimal site with the smallest time cost. If there are any performance enhancements for jobs using the replication service, the NADI uses the decision to replicate the required replica to the chosen location. Otherwise, it optimises the best replica . Additionally, the changeable component in NADI was in the Data Management Service (DMS), which deals with the data and the replicas information. The DMS's main role is to manage replicas, produce new replicas (if needed), and provide the access and location for these replicas. The algorithms used in this chapter are explained in

depth. The results show a significant improvement on the average completion time by adopting the NADI scheduling approach.

Chapter 6 proposes a final version of the NADI scheduling that considers both the computation and the data requirements in order to make the allocation decisions of VMs. Additionally, NADIv3 overlaps between staging the data and execution for the jobs. So, it transfers the data in advance of a task that has its own queuing time while other tasks are being executed. This technique significantly improves the average turnaround time of jobs. This version is considered to be the final version; it considers the data location, computation power and load, replication service, available bandwidth, and pre-fetching the data in advance . The design of this approach has been changed slightly in the DMS's components. These components respond to managing, tracking and transferring data between the source and the execution node. Then, the algorithms that are used in this chapter are explained in depth. The results obtained from the simulation experiment indicate considerable improvement of jobs performance. When compared with the previous approach, it can be observed that the jobs execution has been enhanced. Sequentially, the overall completion time is significantly improved.

Chapter 7 draws the conclusions of this thesis and gives a critical and concise summary of the work that has been carried out. The chapter also describes the application areas and limitations of the thesis, and it suggests potential future extensions for the work.

1.6 Chapter summary

The chapter presented the motivation for the research and stated the objectives of the research, which include developing specifications and a framework for the proposed model and the study of optimisation strategies for the proposed framework. The methodology chosen was modelling and simulation. The modelling stage ensured that the proposed ideas could be verified against well-established mathematical principles, making the research easily replicable by any researcher.

2

Background and Literature Survey

This chapter gives a brief summary of cloud computing. Then, it investigates recent advances in cloud computing. Basic enablers for the new paradigm are investigated and relevant scholarship on the topic reviewed. The main aim of this chapter is properly to identify the chosen research in the light of developments in computing as a whole. The chapter investigates definitions, concepts, and architectures proposed for the emerging paradigm of cloud computing. It then reviews various advances in network and virtualisation, and concludes with a presentation of relevant efforts to facilitate the design and implementation of a cloud system.

2.1 Introduction

Cloud computing is a novel revolution in the information technology field. This revolution has rapidly increased in both industrial and academic areas. Therefore, these two areas have been establishing projects within the area of cloud computing. Cloud computing is a combination of applications, platforms, and infrastructures to provide certain services to customers via the internet. Cloud computing has recently turned to employ many Datacentres (DCs) rather than several Personal Computer (PC), laptops, and mobile computers [21].

The cloud computing concept is not easily defined because it has been applied to different architectures and many technologies. The NIST [22] has, however, defined Cloud Computing as:

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service-provider interaction.

2.1.1 Related technologies:

2.1.1.1 Grid Computing

Grid computing is a federation of computer resources from different locations that coordinates networked resources to achieve the common computational objective [23]. Technical problems usually require several computer processing cycles or access to large amounts of data. Cloud computing has a similar function to that of Grid computing, and it is also concerned with how to achieve application-level objectives in distributed resources. However, Cloud computing differs from Grid computing in virtualization technologies at multiple levels, such as hardware and application platforms that are realizing resource sharing and dynamic provisioning for these resources [3] abd [24].

2.1.1.2 Utility Computing

Utility computing is a provisioning model that provides resources such as computing or storage with infrastructure management. So, it is available for a customer who will only pay for their usage instead of a flat rate, known as pay-per-use. Utility computing is similar to any type of on-demand computing, e.g. Grid computing and Cloud Computing. Furthermore, the Utility model attempts to achieve the maximum efficiency of using resources and minimizing operating costs [3].

2.1.1.3 Virtualization

Virtualization is the creation of a virtual Operating System (OS) that can be used as a CN (commonly known as a virtual machine (VM), storage device, network resources, or even as an OS). Virtualization software is a layer between a physical CNs OS and the VMs OS. It also allows any VMs OS to access and use the physical CN hardware such as the RAM, CPU, and bandwidth, which is similar to what the physical CNs OS does [25]. There are two information technology (IT) areas that use virtualization: Autonomic computing and Cloud computing architecture.

2.1.1.4 Autonomic Computing

Nowadays, advances in the field of computing, e.g. networking, computing technology, and software tools have produced a rapid growth in information services and applications. These services and applications have become complex, heterogeneous, and aggregate a huge number of computing communication and data storage resources. Autonomic computing in the Cloud aims to manage these resources with a high-level guidance and reduce the operation cost without human intervention. Then, automation manages the system status when any changing or unpredictable condition happens [3] and [26].

2.1.2 Cloud Computing Architecture

This section shows all different Cloud Computing Architectures such layered model, Business model, Cloud Computing Classification and Network Architecture.

2.1.2.1 Layered model

Basically, the cloud computing architecture as shown in fig 1 can be divided to first, the core stack and second, the management. In the core stack, there are three layers Resources, Platforms and Applications. The resources are all infrastructure which are composed of physical and virtualized computing, storage and networking resources [1]. This layer consist of all hardware resources including physical servers, routers and switches and Virtualization layer that partition the hardware resources and the creates a virtual pools of computing and storage resources [3]. The famous virtualization technologies are Xen [27], KVM [28] and VMware [29].

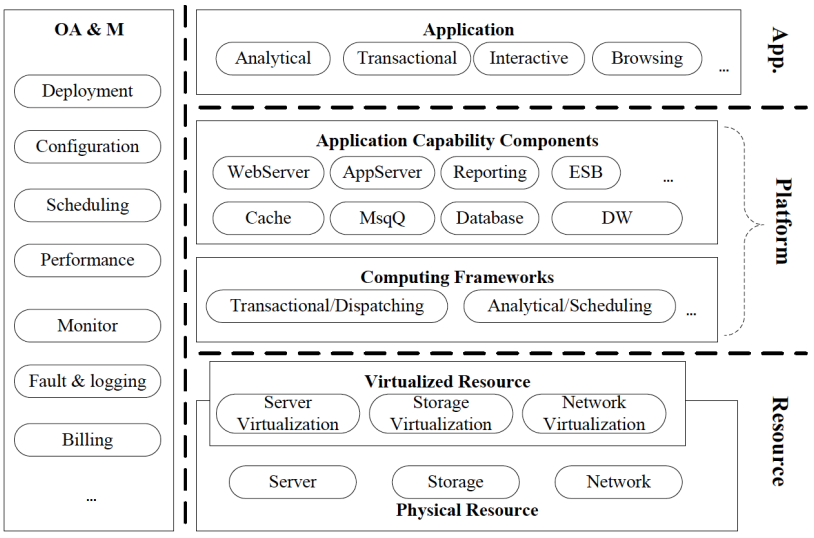


Figure 2.1: Cloud Computing Architecture [1]

The platform layer is built on top of the resources layer that consists of two layers for operating systems and apps frameworks. This layer has created to reduce the applications deployment burden into VM containers. E.g. Google App Engine is operated in this layer to provide API that supports the storage implementation, database and

business logic in web applications [3].

The application layer is the highest level of this model that consists of the cloud applications. This layer supports the distributed transactions and management for massive data volumes. Cloud applications can achieve high performance, availability and in same time reducing the operating cost by using the automatic scaling feature that often cannot find in traditional applications [3] and [1]. Some examples for these layers in Commercial Cloud Systems are shown in table 2.1.

Table 2.1: Examples of Commercial Cloud Systems [6]

Cloud Layer	**Examples of Commercial Cloud Systems**
Cloud Application	Google Apps And Salesforce Customer Relation Management(CRM) system
Cloud Platform Environment	Google App Engine and Salesforce Apex System
Cloud resources Infrastructure	**Computational Resources:** Amazon's,EC2, Enomalism Elastic Cloud.
	Storage: Amazon S3, EMC Storage Managed Service.
	Communication: Microsoft Connected Service Framework

2.1.2.2 Business model

Cloud computing services in business can be divided into three models, and every model can be defined as different layer. Cloud services models are Software as a Service SaaS, Platform as a Service PaaS, and Infrastructure as a Service IaaS [30]. Basically SaaS, is the application that delivered to customer who uses it via the internet. PaaS generally, is all resources that needed to build services and applications which customs use them via the internet without install or download them. IaaS is the computer foundations hardwares that use to deliver certain services to customer. Usually consist of Servers, Datacentres, storages and networking technologies. Also, it includes operating system and some technologies that use to manage these resources [30, 31].

2.1.3 Cloud Computing Classification

Moving the enterprise applications to Cloud arises many issues to be considered. E.g. some of cloud providers are interested in how to reduce operation cost, however; other

gets the highest reliability and security. From these concerned, it shows the differences between cloud classifications that are Private cloud, Public Cloud and Hybrid Clouds.

Private cloud is owned by a company or organization that has to control and manage the all Apps run in the datacentre. Public Cloud Usually runs by third parties e.g. Amazon that provides all resources as services to the general public. These resources expose to user cloud via the internet and uses pay-as-you-go to Cloud Users. Hybrid Cloud is combination of both Private and Public Cloud that allow a company runs apps on internal Cloud infrastructure or in a Public Cloud infrastructure [22].

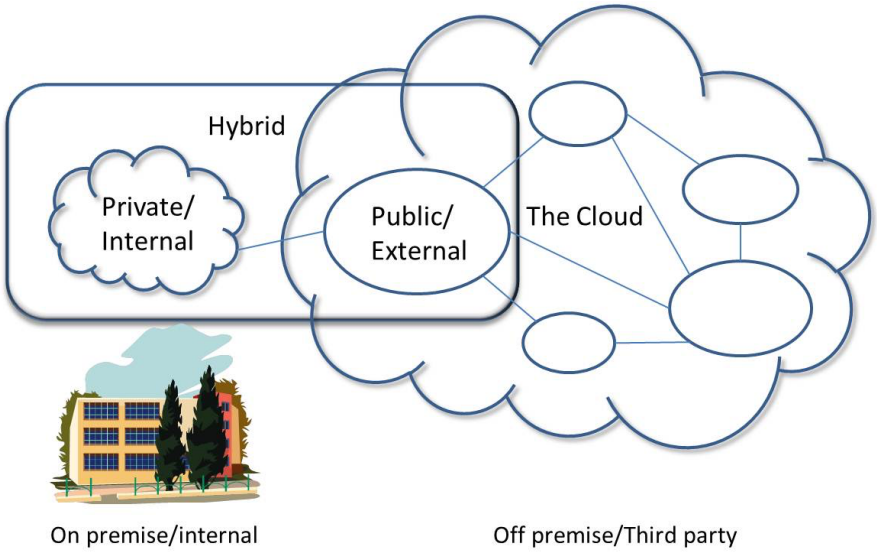


Figure 2.2: Cloud Computing types [2]

2.1.4 Datacentre Network Architecture

Network topologies in DCs usually share a three-tier architecture [4], as shown in Figure 2.3. At the bottom level in this figure is the access tier that organizes CNs in racks. In addition, each CN in the rack is connected to one or two Top of Rack (ToR) switches. Each ToR switch is also linked to one or two switches at the aggregation time. The top level in the figure is the aggregation switch, which is connected with multiple switches at the top level, which is known as the core tier [32]. Examples of this architecture are Tree, Fat-Tree, and VL2.

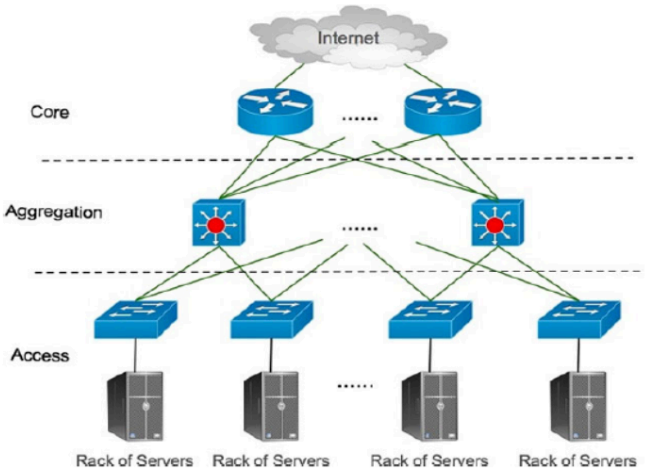


Figure 2.3: DC topology [3]

2.1.4.1 Tree:

The DC network architecture is the original three-tier architecture. This three-tier architecture is also labelled as a Tree. The physical tree architecture topology is a multirooted forest topology. Usually, the Tree architecture is rooted at one of the core switches [4].

2.1.4.2 Fat-Tree:

The Fat-Tree is considered as the extended version of the three-tier topology that is based on a complete binary tree between nodes. The Fat-Tree tries to solve the failure of nodes that is appear on the Tree topology. The Fat-Tree has two forms:

2.1.4.2.1 VL2:

VL2 is one form of Fat-Tree topology. It introduces a new routing schema called Valiant Load Balancing. VL2 has some similar features to Fat-Tree architecture [33]. However, the unique difference that distinguishes VL2 is the complete bipartite graph between core and aggregation switches.

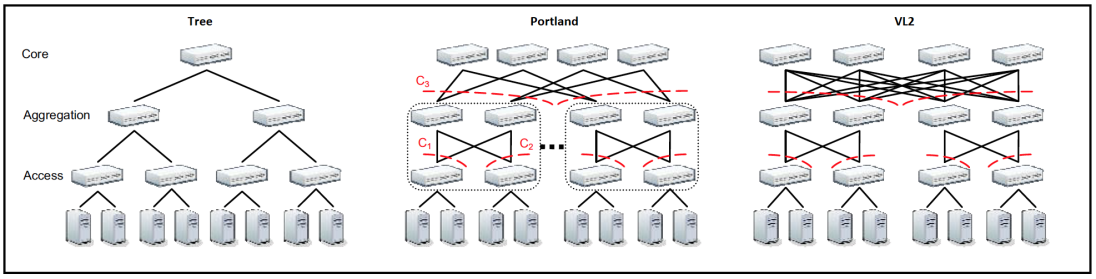


Figure 2.4: Common Data Centre Network Topologies [4]

2.1.4.2.2 Portland:

The Portland is also one of Fat-Tree topology form that is based on bipartite graphs [34]. The main difference distinguishing this topology is the block building, which is called a Pod. Each Pod is a combination of access and aggregation switches connected in a complete bipartite graph (see dotted areas in Figure 2.4). Each Pod has links between the aggregation switches and is connected to all core switches [4].

2.1.4.3 DCell:

DCell [35] is a network infrastructure to interconnect CNs. Each CN is connected to a different level of DCells via its multiple links, but all the CNs act equally. DCell incurs significant cabling complexity that may prevent large deployments. DCell uses only miniswitches to scale out, and it scales doubly exponentially with the CN node degree. Therefore, a DCell with a small CN node degree can support up to several million CNs without using core routers. BCube [36] builds on DCell, incorporating switches for faster processing and active probing for load spreading as shown in figure 2.5.

2.2 Cloud Simulation

2.2.1 CloudSim

CloudSim [37], a framework built by the GRIDS laboratory (University of Melbourne), enables simulation, modelling and practising design of cloud computing infrastructures. CloudSim is a self-contained platform used in modelling DCs, allocation services, and

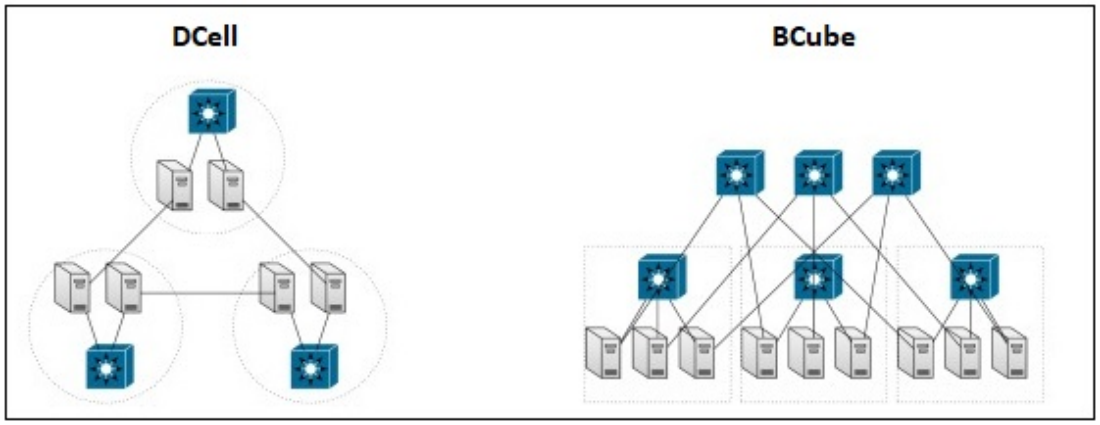


Figure 2.5: Advanced Topologies [5]

service brokers and scheduling of high scaled cloud podium. At the DC, it offers a virtualization engine that has extensive characteristics for modelling the creation of virtual engines life cycle management. CloudSim framework is created on top of the GridSim [38] framework built by the GRIDS laboratory.

2.2.2 Greenloud

Greencloud [39] can be defined as a erudite packet-level simulator used by the energy aware cloud computing DCs that emphasizes communications of the cloud. It provides a fine-grained modelling that is detailed on matters regarding the energy consumed by the DCs IT equipment such as communication links, network switches and CNs. Greencloud can be very useful in developing unique solutions in resource allocation, optimization of network infrastructures and communication protocols, workload scheduling and monitoring. It is an extension of NS2 network simulator published under General Public License Agreement.

2.2.3 Virtual Machine VM Placement

VM placement is defined as the process of mapping a virtual machine to a physical machine. In other words, the placement of the VM process is the selection of the most suitable host for the virtual machine. This method includes categorization of VMs and hardware resource requirements and the anticipated use of resources and placement

target. The placement aims either to maximize the use of available resources or to save power through the ability to shut down some CNs. Autonomous virtual machine placement algorithms have been designed, taking into account the above-mentioned objectives [40, 41]. In cloud computing environments, there are some parameters that should be considered to make the right decision, i.e. when and where the VMs should be placed or reallocated as described in the following paragraphs.

- Performance: this aims to improve the utilization in DCs that have a large number of applications running on CN platforms. VM placement approaches are also applied to achieve the highest performance, which could make a potential difference [41].
- Cost: cloud markets show that dynamic pricing is being increased so that some of the approaches try to modify VM sizes without any impact on service performance [42].
- Locality: VMs should be located close to users because it is the optimal solution for legal issues and security reasons [43].
- Reliability and availability: Because the main objectives of VM placement are reliability and availability, VMs in some cases may be placed, replicated, or migrated across multiple DCs that are spread over geographical zones [43].

2.2.4 Virtual Machine Migration

Migration of a VM is simply moving the VM that is running on a CN to another CN (which is normally called the destination node). The trick to do this while the VM is running on the source node and without an interruption is to move all active network connections even after the VM to the destination node. It is live as the original VM is running while the migration is in progress.

The great advantage is that the live migration has extremely little downtime, in the order of a few seconds. Thus, the main reason for using live migration is for resource management in cloud computing. For example, cloud computing providers, such as

Amazon elastic computing cloud for Amazon elastic compute cloud for amazon (EC2), have thousands of VMs running in their DCs. To save energy costs and balance load, those providers can move VMs using live migration without their customer applications in the VMs. Moving VMs from the source node to the destination node needs to consider these contents: CPU state, storage capacity, and network connections [40].

The benefits of using VM migration are to achieve one of these scenarios as described in the following paragraphs:

- Load balancing: this is to adjust VM placement to achieve critical business goals, such as high throughput [44].
- IT maintenance: Administrators move VMs to free certain hosts and shut down for maintenance [45].
- Power management: this is to consolidate VMs via live migration on an optimal number of CNs and selectively switch off [46].

2.3 Cloud Applications and Datacentre Traffic Patterns

Recently, different cloud services such as Microsoft Azure and Amazon Web Services (AWS) have become extremely popular because of instant payment options, widespread presence, dependability, and ability to provide storage facilities and various software options. Many different sectors are using the cloud, for example the healthcare sector, social networking, search websites, video transmission in compressed form, internet surfing, and many applications that provide information and many others as well [47–49]. Different types of VMs supported by a large quantity of data form the backbone of these applications that combine a large number of functions. With many information hubs utilizing applications dependent on the exchange of information, the data flowing through the intertraffic network has risen by leaps and bounds. The amount of traffic on a cloud application in the context of information exchange and computing is the basis of categorization of applications into three different varieties [50]:

2.3.1 Data-Intensive Workloads:

In these workloads, a large amount of data is exchanged but very few computations are required. An example is sharing of video files in which every single user initiates an individual video streaming procedure. In this scenario, the main obstacle lies in the interconnecting network whereas the ability to perform computing is not important. All tools used in the circuit (e.g. switches) have to work within the framework of a feedback system with the placement manager and work scheduler present in the central position so that the application is able to generate high efficiency and work in accordance with the Service Level Agreement (SLA). After receiving the feedback, the scheduler distributes the amount of work depending on the time taken to complete a workload and the size of incoming traffic in the interconnecting network. So, the application tries to function via links with low traffic, although a CN with high traffic can still handle the additional demand by virtue of its computational ability. So, the traffic can be diverted in a way that all links receive equal traffic and the time taken to complete the functions and minimum time required to initiate the process are reduced.

2.3.2 Computationally Intensive Workloads

The Computationally Intensive Workloads (CIW) present as a high performance Computing application (HPC) and their main purpose is to find solutions to complicated problems requiring a high degree of expertise and a large amount of computational power capacity. Such applications do not need a transfer of data via the communication link but an ability to handle a large number of computations and functions is required. So, all these applications can be compiled and can function via very few CNs using the method of VM consolidation so that they become energy efficient. Because the VMs have a very small amount of data flow, networks are generally free and a large number of network switches can be set to power saving mode (such as sleep mode), thus reducing the amount of power used by the DC.

2.3.3 Balanced Workloads

The Balanced Workloads (BW) stand for those applications that require the exchange of a large amount of data via VMs and the ability to perform extensive computing as well. An example of this variety is the Geographic Information System (GIS) in which information in graphical form has to be transferred and much computational power is needed to analyse this information. In this situation, the size of data flowing through the intercommunicating link determines the load on the CN. For the purpose of these applications, the scheduling strategies and VM placement have to consider both the CPU's loads and the amount of traffic in DCs links to avoid any congestion or heavy loads on the CN. As long as the DCs have heterogeneous on cloud platforms, CNs' power, applications, services and different traffic patterns variation between nodes (CNs and SNs), The network usages in DCs usually are found to be as the following points: [32, 51, 52].

1. There is only a weak interrelationship between the total cost incurred in communicating through the entire extent of VMs and the flow of traffic through paired VMs.

2. The VMs have different traffic rates.

3. It is seen that for the VMs where traffic rate was high the trend was a constant one and the same was true for VMs with low traffic rates.

2.4 Network-Aware VM Placement and Migration

VM placement and migration scheduling approaches are considered among the most challenging problems in Cloud management and recently have been categorized as a hot topic because by considering these approaches, the DCs will get improved resource utility and minimize the energy consumption depending on the objectives that are chosen [53, 54]. This has resulted in substantial research with different motives. So, this chapter consider the VMs network-aware algorithms in both allocation and migration

scheduling. In the next four sections, the most credible and latest scientific research will be reviewed, analysed, and commented on as shown in Figure 2.6.

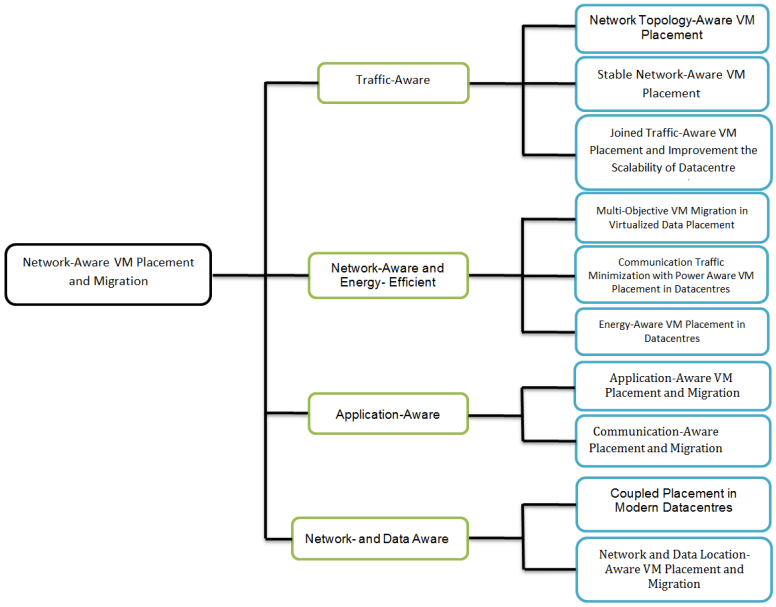


Figure 2.6: Categorization of network-aware VM placement and migration approaches

2.4.1 Traffic-Aware VM Approaches

2.4.1.1 Network Topology-Aware VM Placement

Georgiou et al. [55] conducted a study on the advantages of using in advance the input for the bandwidth requirements and the communication between VMs in the stage where decisions were being taken regarding VM allocation. According to the authors, it would be best to use two off-line VM cluster algorithms so that the least number of networks is used in the physical dimension while ensuring that the entire process can be completed within the limits of CNs capacity and its constraints. The arrangement of VMs is in a manner that it follows the principles of virtual infrastructure (VI) keeping in consideration the guidelines laid down for the resource setup and facilities (storage capacity, memory, core of CPU), bandwidth requirement of inter-VM communication working with reference to VI and set up as virtual link (vLink) and taking into account the limitations dealing with ensuring that the system does not fail for VM pairs. The

setup of physical parts is a collection of CNs that are arranged in accordance with Portland topology [56]. According to the authors, it has been seen that a tree-like network configuration that has been in use most commonly can lead to overuse and competition for network connection most commonly at the highest level, which ultimately hinders and increases the time taken to access the services. The use of Portland structural arrangement can help cloud centres use their available data resources efficiently and handle bandwidth requirements in a better manner.

This approach has two-layered structural arrangement, which is a physical infrastructure made up of homogeneous CNs arranged in accordance with Portland topology and a layer of bridging middle-ware placed above the infrastructure. Deployer and planner are the two main parts that form the middle-ware. The planner receives the requirement for VM specification and discovers the available resources information in the DCs infrastructure. After, the running of the VM placement algorithms to determine the mapping for both VM-to-CN and vLink-to-physical, it sends this information to the Deployed to make the decision. The VM deployment on physical layer components can be handled by the deployer, which may have been considered as a third-party. The main objective of this approach is to maximize the available bandwidth in DCs links in the stage-in which decision for VI deployment is being taken. Thus, there are two algorithms to provide this objective. The first algorithm is Virtual Infrastructure Opportunistic fit (VIO), which is used to allocate the communicated VMs side by side to minimize the traffic in the DCs.

Vicinity-Based Search (VIBES) is the second algorithm and is structured on the basis of Portland network principles, this algorithm tries to search for the best possible Portland area in which VMs and Virtual links can be accommodated in each request. In the next step, VIO is then tailored according to this area. VIBES uses the concept of pods (clusters of CNs that are under the same level of switches) to detect where the Vls can be fitted. The researchers also proposed that all neighbourhoods should be listed by keeping in consideration the CN resources and bandwidth capacity. The first pod that is picked up by VIBES has the highest number of resources and then VIO is

activated. If VIO does not accept the request more pods are approached in descending order of availability of resources. This process continues till a neighbourhood that has the required resources is found or when the area of search exceeds the maximum limit, in this scenario the demand for VI placement is not accepted.

Various simulation models have been used to assess the efficiency of VIBES and VIO and the results compared with networks of similar structures using the First Fit decreasing (FFD) algorithm. Three different data-flow structures are used to mimic removal and activation of VI, namely Epigenomics, Data Aggregation, and Pipeline [57]. According to the results, the algorithm gave better results than the FFD as far as utilization of network is concerned. There is an approximately 75% reduction in the network traffic travelling through the Portland architectures uppermost layer and use of VIBES leads to 20% more reduction. It is the view of the authors that further studies are needed to evaluate methods used to decreases the load on networks and more efficient use of power in network switches evaluating VIBES and VIOs performance in Bcube [36] and VL2 [33].

2.4.1.2 Stable Network-Aware VM Placement

Biran et al. [4] have proposed a VM placement algorithm to reduce the highest amount of bandwidth requirement and volume in all networks while simultaneously tapping into unused network links that can handle unexpected increase in the traffic, the study utilized the different communication trends, changes in amount of traffic in cloud platforms and non-trivial datacentres structural arrangement. They have pointed out many important factors related to the structural arrangement and traffic:

1. There is a large variation in the amount of traffic at different times by virtue of the impact of several factors like the time of day and occasional sudden increase in traffic.

2. The majority of DCs architectures use non-trivial form such as Fat-tree [34] and VL2 [33] or a different modification of the multipath routing process.

Taking into account the previous two points, Biran et al. [4] have designed two VM allocation algorithms to predict the communication requirements and deal with the changeable amount of network traffic. The Min Cut Ratio-aware VM placement (MCRVMP) concept was given by the study and the Integer Quadratic Programming model was used to set it up taking into account the limitations of the CN and network resource constraints using the dynamic routing process and complicated network configuration. They have also put forward a graph transformation method to enable MCRVMP application on different network configurations overriding the limitations of applicability of MCRVMP on tree configuration alone. Because MCRVMP belongs to the category of NP-Hard (NPH) problems, two different algorithms have been put forward by researchers to provide a solution to the problem of placement and they also drew up a comparison between random and optimal placements.

The principles of connected components (CCs) of functioning algorithms were used in setting up the heuristic algorithms. The VMs that interchange data among each other and with external output (VMs made of multilayer approach) are grouped together and it becomes easier to find a workable solution. The first algorithm is called 2-Phase Connected Component-based Recursive Split (2PCCRS) and follows the principles of recursive integer programming and works on a tree network configuration to find solutions to minor problems on a single-level tree. Because the 2PCCR uses a method comprised of two parts, the CCs are set up in the network and then the CCs are increased in scope and then the VMs are fixed on the CNs. So, by use of 2PCCRS the complex MCRVMP problem is divided into smaller parts and then solutions are found using the mixed integer programming solver in the two phases. The Greedy Heuristic (GH) is the second algorithm and in it there are no mathematical computations and every VM is allocated individually. The GH works in two stages. The first stage of GH takes into account the traffic congestion and arranges CCs in descending order in the first phase depending on the total traffic requirement in VM set up in a CC. The second stage handles the requirement of traffic by allocating every VM on CN that has the lowest load. During the second stage of the evaluation, the consistency of

MCRVMP-based placements in accordance with changing traffic patterns was proven by use of simulations framed on the basis of NS-2 whose primary focus was on the average time taken to deliver packets and the percentage of dropped packets. According to the results obtained, GH and 2PCCRS are able to handle three times higher traffic than nominal value if there are no dropped packets.

2.4.1.3 Joined Traffic-Aware and Networks Scalability Improvement

The work of Meng et al. [32] has focused on the problem of scalability being faced by the DCs and worked out techniques to solve the problem by best possible placement of VMs on CNs. The researchers adopted a new approach, which did not include altering the structural array of the network or the routing methods and put forward the hypothesis that an improvement in scalability can be brought by decreasing the distance between interlinked VMs. The trend of traffic in functional DCs was studied via a quantitative study and the conclusions brought three some important points:

1. There is only a weak interrelationship between the total cost incurred in communicating through the entire extent of VMs and the flow of traffic through paired VMs.

2. The VMs have different traffic rates.

3. It is seen that for the VMs where traffic rate was high the trend was a constant one and the same was true for VMs with low traffic rates.

As per the work in [32], the Traffic-aware VM Placement Problem (TVMPP) is a member of the Quadratic Assignment Problem group and deals with the concept of optimization [58] and is also considered complex enough to be granted the categorization of NPH. The communication costs matrix deals with the communication between the CNs whereas the traffic matrix stores the amount of traffic between VMs. These values in the matrices are considered to be the input components for TVMPP and the entire exercise is aimed at deriving the VM placement that would reduce the average traffic at the level of individual switches. The standard definition of communicating cost in

relation to two interlinked VMs envisages it as the total number of hops or switches that are present on the best possible path between the two interlinked VMs. The definition of slots has also been put forward by the researchers as a memory or CPU that is designated as the actual CN and a CN can have many such slots and any VM can take up any slot.

The present solutions in the context of TVMPP, which is an NPH problem, cannot be extrapolated to modern data hubs; to solve this problem researchers have used the concept of design to formulate an approximate algorithm with two layers, namely Cluster-and-Cut.

1. The similar concept is used both in VM-PM mapping and solving TVMPP because in both situations the two PMs that are attached to the physical hardware that is easy to use are arranged in proximity to two VMs with high network traffic.

2. Utilizing the concept of divide and rule.

The Slot Clustering and VMMinKcut are the two main parts of the Cluster-and-Cut heuristic. The number of hops that are present between the slots is the basis of division of n slots into k clusters by slot clustering. After this process of division is completed, the slots come out to be arranged in descending order of total costs (includes costs of both directions). The basis of decision of n slots in k clusters by using VMMinKcut is the amount of traffic in VM pairs, the division is done in a manner that the traffic flow in between the cluster is reduced to the minimum and the two VMs that are placed in one cluster have high exchange of traffic between each other. The minimum k-cut graph algorithm is used for this purpose [59] and the size of the k clusters is the same as the size of the k-slot clusters. In the next step, the Cluster-and-Cut maps individual VM clusters to slot clusters and as far as the combination of VM clusters is concerned, a solution is derived for a smaller TVMPP problem and then VMs are mapped to slots. In addition, the value of mathematical ability has been derived by the researchers for Slot Clustering and VMMinKcut and the values are $O(nk)$ and $O(n^4)$, respectively, with total value coming out to be $O(n^4)$.

The Cluster-and-Cut heuristic is assessed via simulation using trace-driven principles that take into account the hybrid traffic and the traffic flowing between VMs (traffic travelling in both directions is taken into account), all the data are collected from DCs that deal with day-to-day activities. According to the findings of the study, the Cluster-and-Cut leads to an objective function with a value that is almost 10% less than its counterparts in various network configurations and the time taken to solve the problem is decreased by 50%.

The weak point of the research is that some of the theoretical arguments do not hold good when used in practical situations in DCs. The limitations of the interconnecting network are not taken into account by TVMPP so the placement of VMs can be done in networks with high traffic [4]. It should also be remembered that a single VM is placed in one CN by the Cluster-and-Cut algorithm, thus causing wastage of the DCs resources. The concepts proposed are also based on the presumption that both static layers 2 and 3 are functioning in the DCs. It is also important to note that the migration costs due to VM shifts off-line are also not taken into consideration.

The closest related work to TVMPP [32] is the Integer Non-linear Program (INLP) approach [60]. The INLP is aimed to keep the reliability of communications between VMs. So, in this case, the distance between VMs should be not far and provide more than a path in case of a links failure report between them. This approach helps to increase the DCs reliability and minimize the communication delay between communicated VMs. As a precautionary procedure, this work creates backup VMs replicas at near CNs to guarantee the reliability and availability among VMs. The limitations of this approach are, first, they considered communication only between CNs while the SNs have been ignored. In case of the like between the data and application has a low rate of throughput, degrades the performance for both jobs and application. Second, creating many replicas for the VMs could cause network burdens and squandering of the DCs resources.

2.4.2 Network Aware and Energy-Efficient Approaches

2.4.2.1 Multi-Objective VMs Migration

The situation of excessive congestion in VMs of DCs whose functioning is dependent on interlinks between CN has been proposed by Huang et al. [61]. So far, the majority of research in the field of VM migration has dealt with the limitations of CN resources and aimed at finding a method to set up the VM in a manner that utilizes the least possible number of CNs, which will make the process more energy efficient and optimize the use of resources.

Huang et al. [61] have declared that most VMs that are in use at present share an interrelationship via exchange of information and traffic. So, it is essential that the technique used for online VM migration is two pronged and aims at optimization of resources along with decreasing the load of traffic on DCs by providing a multi-objective algorithm. These researchers have worked on the same concept as Huang et al. [48] and put forward three steps that can help in creating a structure in which resources are utilized to the best of their capacity.

1. The configuration where the first optimization takes place aims to optimize the CN resources by decreasing the number of CNs, thus reducing the energy consumption; the principles used here are the maximum minimum fairness mode and sharing of resources at hand.

2. In the second step of optimization, an attempt is made to decrease the costs of data exchange between VMs once the VM migration has taken place. Here, the traffic flow between VMs and the functional use of the application is taken into consideration.

3. The third step is an amalgamation of the above two steps, provided that the optimization is performed within the limits of the CN and the value of the total load of communication in the CN does not exceed the total bandwidth volume.

A further solution to the defined optimization problem has been put forward by

researchers using a GH algorithm composed of two stages, namely the Extension and Base algorithms. The input values of the base algorithm involve VMs groups, CNs groups, and heavily congested VMs along with the majority of resources utilized by user CNs. In the next step, the heavily congested VMs are arranged in descending order of the amount of resources used prior to completion of migration. The extension algorithm tries to pick the best option as the CN for placement, which has the least resource utilization and the lowest amount of load; this process on initiation takes into account the interrelationships and links between VMs. The VM migration effect is defined as the effect of both distance and cost of traffic between VMs. The total communication weight for each congested VM is calculated as the total of all VM communication weights and then the congested VM is transferred to the CN that shows the minimum effect of migration.

The study has also dealt with assessment of the concept of multi-objective VM allocation by comparing with the application-aware VM migration approach (AppAware [62]). The topology configuration of DCs can be set up in four different forms: Tree , Fat-Tree, VL2 , and BCube. The capacity of the CN, the demand resources for VMs, and the amount of traffic in VM are calculated artificially using the normal distribution with a change in the value of the mean. According to the study, the Bcube configuration resulted in more reduction of traffic volume than the tree configuration. The traffic load is also decreased to a higher extent in DC that has been put forward in this paper in comparison to AppAware as it produces migrations, which decrease the amount of traffic volume by 82.6% (this is true for a few VMs). The capacity of CN and the effect of migration are inversely proportional. The cause of this has been postulated to be the incorporation of the maximum value of resources utilized in the multiplier factor, which is reduced once the migration is completed. The increased load on VM resources is directly proportional to the effect of migration. The cause for this has been attributed to the effect of increased load on the intercommunication between VMs, especially in applications with many levels. The weight of communication affects the traffic load between network switches, leading to an increase in the effect of migration and increase

in the inter-VM weights.

2.4.2.2 Power-Aware and Communication Traffic Minimization

The static greedy placement of resources to VMs has been studied by [63] and did not take into account the effect of utilization of resources by CNs and VMs. According to the authors, all the VMs that have a high traffic load can be grouped together to utilize the least possible number of CNs, which will lead to a decrease in the hosts' external traffic as VMs that are in close proximity can transfer information to each other. The work puts VM dynamic placement in the position of an optimization problem so as to reduce the traffic load in the DCs. Mapping of CNs and VMs can provide a solution to this problem and a minimum k-cut problem [64] seems to be NPH. Around one-third of the total energy is saved if a CN is in an idle mode [65]. The research proposed that the second objective of VM placement would be to decrease the energy usage.

A theoretical simulation was put forward using computations to reduce the traffic load in the DCs while satisfying the resource limits of the CN. The solution to this problem is grouping the VMs with high traffic load together so that one CN can host all of them. So, they have proposed using the K-means clustering algorithm [64], which would help in the mapping of VM on CNs. The authors put forward a GH titled K-Means Clustering for VM consolidation, which from its start used the CN as a cluster. There are many advantages associated with this approach:

1. The detrimental effects of randomization are reduced by bringing about changes in the initial clusters and their number K.

2. The maximum limit for each cluster is equal to the highest possible capacity of the CN.

3. A decrease in the number of migrations can be obtained by fixed clusters.

Every round of K-Means clustering in the VM consolidation algorithm is analysed to calculate the distance between a CN and VM. Then, the VM is allocated to the CN that is closest to it. The entire process is redone till all VMs are placed on their

respective CNs. According to the research, the value of polynomial complex of the greedy algorithm is $O_{t,m,n}$ and the number of rounds is denoted by t, n stands for the total number of VMs and the total number of CNs in DCs is denoted by m. It has also proposed algorithms that can calculate the VM cluster distance and can help GHs deal with the VMs requests.

An assessment of the performance of three algorithms with the proposed algorithm was carried out in terms of calculating the amount of load on DCs both via simulation and synthetic methods. The three algorithms that were compared were:

1. First Fit (FF) heuristic
2. Random placement
3. Method of using simple greedy technique (the VM is placed on the CN that has a communication with the VM)

Exchange of information between VMs does not play a role in the FF heuristic and random placement. The two characteristics that were evaluated were

1. The total number of CNs needed (or energy used) after consolidation
2. The total traffic in DCs and the greedy algorithm shows the best reduction in both the above-mentioned parameters.

As far as the online VM deployment is concerned, the greedy and cluster algorithms give similar results even though the greedy algorithm has to function with a higher number of migrations and in spite of that, it can handle a request for new VMs quickly with no impact on other nodes.

2.4.2.3 Energy-Aware VM Placement

Nowadays, the DCs that are providing a platform for communication applications have been plagued by the problem of power consumption used by CNs and network devices, as stated by [48]. In this work, many important points related to DCs have been put

forward to stress the significance of VM placement that can fulfil the multi-objectives approach.

1. Using complicated applications that are made up of many intercommunicating VMs.
2. Increase in the size of DCs.
3. The methods for VM placement in use at present cannot perform many simultaneous functions efficiently.
4. Increase in the price of electricity.

To get an in-depth understanding of the above-mentioned points, the researchers tried to compare the energy used by the CNs themselves and the energy consumed in DCs by switching networks and data transferring. The authors proposed a multi-objective VMs allocation approach that has three phases. The first framework approach takes into consideration the resources of the CN (storage, RAM, and CPU) and the VMs capacities of required resources to maximize the CNs resources and decrease the use of energy by decreasing the number of CNs. This framework used the principles of proper fairness and in a condition where no significance is attached to the exchange of information between VMs. The second framework focuses on the pattern of traffic between VMs and the constraints set up by the CNs bandwidth capacity. It is designed as a convex programming problem whose aim is to reduce the communication costs between VMs. Lastly, the VM placement in the energy-aware approach is designed using a fuzzy logic concept with a balance achieved between the two objectives, which in combination can negate each other. A further prototype technique is to join the VMs allocation with each VM being fitted with a local controller and the DC fitted with a global controller that decides the placement of VMs on the CNs resources.

Two different concepts have been proposed to find the optimum solution, namely VMGrouping and SlotGrouping. The VMGrouping functions in a manner where the VM and CN links are set up in such a way that VM pairs that have high load are connected

to those CNs whose set up in physical form entails less cost. This setup is simulated in the form of a Balanced Minimum k-cut Problem [59] and a k-cut with minimum weight. The amount of k-cut that has the lowest possible value of weight is also fixed to divide the VMs to k disjoint subsets that are unequal in size. Then, the SlotGrouping connects the VM to the CN located in proximity keeping in consideration the limits of the CN. Proof of the efficiency of multi-objective VM placement techniques has been provided by the authors by using simulation experiments conducted under different values of VM loads, traffic, and physical CNs under a normal distribution for different values of mean, topology configuration, and architectures (e.g. Tree, VL2 , Fat-tree, and BCube.

Taking the formulation of total traffic volume of DC according to performance metrics and objective function into consideration, a comparative study was conducted between the joint policy of VM placement and arbitrary placement along with the policies of FFD, which is based on heuristics. The outcomes bring in notably decreased traffic frequency (50- 81%) and greater objective values in the joint VM placement that are further resolved into reduced resource wastage and transport cost. Viewed from the point of reduction of energy consumption, the performance is evaluated by comparing the proposed approach of placement with arbitrary placement, Grouping Genetic Algorithm (GGA) [66], optimal placement with the consideration of the number of used PMs as a performance metric, FFD, and a two-stage heuristic algorithm [67]. The results indicate that in terms of performance the method of energy-aware joint placement is ahead of the two-stage heuristic algorithm, arbitrary placement and GGA and lags behind optimal placement and FFD. Various objectives (meaning a simultaneous reduction of both traffic volume and resource wastage) trade off with each other to rationalize that pattern of performance, which is to be achieved by the joint policy of VM placement.

The focus of this study was the contemporary issue of bringing equilibrium between network awareness and energy during decision-making in VM placement. Nevertheless, the effect of required live migrations of VM and reconfigured performance of the hosted application and network links has not been accounted for in the work. This can result

in a malignant impact on network performance and SLAs with the consideration of the necessity of large VM migrations while deciding on new VM placement.

2.4.3 Application-Aware Approaches

2.4.3.1 Communication-Aware Approaches

The issues in scheduling VMs were first highlighted by [68], which are related to HPC applications and shared networks (at the time of positioning into different CNs) and a shared memory bus (at the time of positioning into the same CN) is used for communication. The researchers have observed a few drawbacks in the existing migration approaches and VM placement in relation to HPC and similar applications:

1. The approaches of VM placement effectively using the resources of the CN itself (such as memory and CPU) do not have any cognizance regarding the patterns of inter-VM communication and result in lower efficiency from the viewpoints of network utilization and final application performance.

2. The focus of the present network-aware VM placement is the optimal primary VM placement and the patterns of real-time communication as well as traffic demands are overlooked and therefore do not produce any reaction to changes.

The work in [68] looked into those setbacks through the proposal of the scheduling technique of energy-efficient VM and communication-aware that concentrate on similar applications using different models of programming in regard to inter-VM communication (such as Message Passing Interface (MPI) and OpenMP). The communication patterns and the duration of inter-VM bandwidth needs are ascertained by the proposed technique after inefficacious placement, rescheduling of VM placement by VM live migrations being recognized.

The researchers have accounted for the system framework in a brief note for managing VM migration requests encompassing peerVM information (such as VMs with mutual communication) and a central Migration Manager (MM). The execution of HPC jobs is carried out in individual VMs and every VM is enlisted with its peer-VM at the

time of the run. The responsibility of determining the communication pattern of the entire application lies upon MM. The physical CN is moreover expected to consist of adequate free resources (10-20% CPU) for controlling prospective VM migration. A frequently used greedy algorithm, called the Peer VMs Aggregation (PVA), has been offered by the researchers, which is to be driven by MM as VMs send requests for migration. The PVA algorithm principally aims at grouping the communication VMs along with mutual traffic within the same CN for their communication through a shared memory bus and also the flow of inter-VM traffic in the network can be reduced. The communication halts in VMs due to mutual communication reliance (and therefore results in elevated performance of application) would be reduced to a minimum and the traffic (and therefore yield decreasing the usage of the network) would be localised by it. The four parts below form the PVA algorithm:

1. Sort: On the basis of input/output traffic flows, the VMs requesting migration are arranged into a descending order and the requests of VMs allocated to the same CN are ignored.

2. Select: the VM with the highest rank is selected by the MM to be migrated to the destination CN with its peers.

3. Check: The credibility of VM migrations to the destination CN are assessed by the MM having been considered as a CN resource (such as network I/O, memory, and CPU).

4. Migrate: First, MM looks into the suitability of the CN regarding the VM migration and in finding alright the selected VM is directly migrated to that CN; else, a VM from destination CN is attempted to be migrated by the MM for obtaining adequate resources to place the selected VM into the same CN as its peer-VMs (but they should be suitable for migration in that VM). Nevertheless, upon the destination CN not hosting any VM enables the MM to allocate the selected VM to a CN comprising the same edge switch where the CN of its co-VMs is incorporated.

It is observed that total traffic of the DC is significantly decreased in the PVA approach as the traffic of network usage goes down by 25%. The researchers have reported the implementation of memory sub-systems and network topology with the well-known simulation toolkit CloudSim [69] and the NAS Parallel Benchmarks (NPB) in place of HPC application being segmented into two sections, namely pseudo-applications and kernel benchmarks [70]. At the point of being weighed up with the random placement algorithm based on CPU utilization, the implication indicates PVA to unify each VM in relation to an application within the same CN and hence yield an exact VM placement as the traffic pattern of interacting VMs gets decided. In addition, as far as the reduction of network usage is concerned through relocating inter-VM communication between the shared network and shared memory by unifying communicating VMs, the proposed approach seemed to have surpassed the placement based on CPU. In addition, the computation performance degradation of the application is carried out and weighed up with the perfect time for executing the individual jobs. According to observation, use of PVA causes 18% performance degradation of VMs, becoming 20% while using placements based on CPU.

The location of VM migration is prompted by the PVA approach but it does not clarify the time of migrating request by the VM. Furthermore, the related overhead of VM migration has not been included. Besides, unifying each VM with a HPC or parallel application into one CN would not always be possible. Ultimately, the aspect of energy efficiency from the proposed approach is not shown in the assessment, although the proposed algorithm tries to reduce the energy consumption as one of the considerations in this work.

2.4.3.2 Application-Aware VM Approaches

A problem in an application-aware VM placement has been pointed out by Song et al. [71] that focuses on scalability and energy efficiency of advanced DCs. Many factors involved in the management of modern DCs have been propounded by the researchers:

1. In the DCs, an excessive application of the services of large-scale data processing

is utilized.

2. Considering the increasing demands of inter-VM bandwidth in modern applications, many studies of network architecture scalability have been initiated so that the level of network connectivity can be increased through scheming dynamic routing to reduce the costs of the DC network.

3. Some mechanisms have also been proposed in many studies accounting for the minimisation of power and energy consumption so that utilization of CN resources could be enhanced and inactive CNs could be turned into lower states of power to reduce energy consumption.

4. Novell Plate Spin [72] and VMware Capacity Planner [73] are examples of the existing tools for VM placement. These tools do not consider the traffic patterns between nodes (CN or SN) and therefore can determine the placement of extensively communicating VMs into physical CNs having communication in a long-distance network.

Similarly, [48] and [71] proposed a problem of VM placement on the basis of convex optimization and proportional fairness for pointing the problem which combines to minimize the amount of DC traffic and DC power consumption so that scalability can be improved. Both demands of application-level inter-VM traffic and the constraints of CN resource capacity have been taken into account at the time of problem formulation. Moreover, evaluation based on simulation is shown and it is reported that VM placement algorithms based on FFD and random are outperformed by the aggregated algorithm of VM placement.

The problem of load balancing in the DCs through overloaded VMs migration to underloaded CNs and the migration being aware of the network has been acknowledged by Shrivastava et al. [62]. As proclaimed by these researchers, auxiliary network overhead can be brought in during the migration of VMs (part of multi-tiers applications) for eliminating hotspots as the inherent pairing of VMs is the basis of communication, specifically after being shifted to CNs with long network distance for the VM. The VM

migration has been formulated by the researchers in terms of an optimization problem aiming at detecting destination CNs regarding overloaded VMs resulting in reduced network traffic following the migration and ultimately a GH of network topology-awareness has been offered by them.

The name of the optimization problem in the proposal is application-aware as the migration decision regards the entire context of application moving above the overloaded VM. An outlook of VM interconnections with multi-tier application is constructed upon a dependency graph with VMs and VM communications as vertices and edges, respectively, of the graph. The function of network cost has been designed by the researchers in terms of traffic demand in the distance of network and edge for concerned host CNs defining that network distance as delay, latency, or number of hops in between any two CNs. Besides, the problem formulation also accounts for the capacity limitations of CNs resource.

AppAware, a greedy probable solution has been offered to optimize the problem, which is defined as NP-Complete (NPC). It tries to minimize the time cost at every deciding step of migration with the consideration of the network topology underly and inter-VM dependencies of the application level. There are four stages in AppAware:

1. Base Algorithm: Computation of the entire communication weight is conducted regarding all overloaded VMs of the system and it arranges all the VMs in descending order. This is further followed by the computation of a factor of migration effect regarding every chosen destination.

2. Incorporation of Application Dependency: The cost of VM migration to a destination CN is entirely computed in this stage of AppAware in terms of the summed corresponding individual cost for every peer VM to VM communication.

3. Topology information and CN Loads: Contiguous CN loads and network topology are accounted for in this stage of AppAware when migration is decided due to a CN being nearer (as per topological distance) to different underloaded CNs that would be preferable as a destination respective of VM for its ability to accommodate

dependent VMs into closer CNs.

4. Iterative Refinements: two extensions are further infused for enhancing AppAware so that DC traffic can be reduced. Several values for the migration effect across several iterations in the base algorithm of AppAware are computed in the first extension and the previous extension is further filtered in the second extension as the anticipated migration of future mapping in other VMs are regarded in respect of a provided postulated destination CN for every iteration.

The researchers have evaluated the performance of AppAware on the basis of numerical simulations through contrasting with a grey-box migration scheme, Sandpiper black-box, and the optimal solution [51]. The normal distribution is used to generate residual resources capacity of run-time CN (such as storage, RAM, and CPU) and usual, consistent, and increasing distribution including variance and fluctuating mean are used to generate inter-VM communication dependencies. The sole purpose of juxtaposing Sandpiper and AppAware against decisions of optimal migration is AppAware and small-scale centres of data (having 10 CNs) being claimed to provide solutions in an approximation of optimal solutions. AppAware and Sandpiper are compared with each other in regard to large DCs (having 100 CNs) and AppAware is claimed consistently to outperform Sandpiper through the generation of migration decisions decreasing traffic volume as the transportation of network by up to 81%. Furthermore, the suitability of AppAware is evaluated in respect of several network topologies by juxtaposing with decisions of optimal placement regarding VL2 and Tree network topologies. As reported, AppAware stays very near to optimal placement of Tree topology while performing but it distances itself from VL2. The capacity constraints of CNs resources are counted by AppAware in VM migration whereas capacity limitations of physical link bandwidth are disregarded. This can result in congested network links of low distance caused by VM migrations.

2.4.4 Network-Aware and Data-Aware Approaches

2.4.4.1 Coupled Placement in Modern Datacentres

The issue regarding positioning of data components and computation of applications into the SNs and CNs of virtualized DCs has been indicated by Korupolu et al. [74]. Various factors have been put forward heralding the introduction of heterogeneous advanced DCs and leading to non-trivial placing for each pair of compute-data in the optimization problem [74]:

1. There is an evolution over time in the DCs of enterprise, and performance can vary in separate parts of DCs (such as one network can be switched more recently and there can be higher I/O throughput and lower latency than with others).

2. Popular multi-purpose hardwired equipment (such as storage equipment having built-in resources).

3. The I/O rates between SNs and CNs of modern applications vary greatly.

The Coupled Placement Problem (CPP) can be officially categorized under optimization problems that aim at reducing overall cost related to each application only with the accomplishment of constraints of CN capacity after accounting for the factors mentioned above. A function defined by any user can be a cost function and conceptually the network cost is captured through it and positioning the components of the application (such as the VM) makes it occur in particular data components and CNs (such as file system or data block) of some SNs. Three separate heuristic algorithms were offered below in [74] for the initially provided CPP in terms of the NPH problem:

1. Individual Greedy Placement (INDV-GR) followed a greedy approach for positioning storage of application data arranged on the basis of their I/O rate for each data storage unit having SNs sorted according to their minimum distances to any CN that is connected. Therefore, as it has the nearest CNs, SNs are positioned with much greater throughput applications by the INDV-GR algorithm.

2. The affinities of CN-SN are regarded by Pairwise Greedy Placement (PAIR-GR), which is another greedy algorithm and puts effort so that data components and computers in every application can be positioned at the same time through the allocation of applications that are arranged according to the I/O rate being stabilized with the requirements of data storage and CPU for pairs of CN-SN being ordered through the in-between network gap of node pairs.

3. Lastly, a Coupled Placement Algorithm (CPA) with CPP was offered being represented bearing identical properties as the Stable-Marriage Problem [75] and the Knapsack Problem [76] to overlook the results of the greedy nature in the initial two algorithms that are early decisions of sub-optimal placement. After finding solutions for the Stable-Marriage Problem and the Knapsack Problem, placement decisions are frequently chosen by the CPA algorithm so that CPP problem can be mended in three stages:

    - CPA-Stg phase, which decides data storage placement.
    - CPA-Compute phase, which decides the placement of computation components on the basis of present storage placements.
    - CPA-Swap phase, which finds out application pairs so that their pairs of CNSN can be alleviated in terms of performing the swap and cost function.

Optimal solutions were experimented on the basis of simulation to compare the performance of CPA, PAIR-GR, and INDV-GR. The researchers for the larger problem and minor issues have applied the MINOS solver on the basis of LP relaxation and CPLEX ILP solver, respectively. The times of placement computation and values of cost function play the function of performance benchmarks and occurrence of the experiment includes the spheres of four dimensions:

1. The level of complication or size of problem by various simulated size of DC,

2. Integrity of fit by various data demands and mean application computation,

3. Difference in data demands and application compute, and The factor of link distance of physical network.

The proposition of the CPA algorithm is realized by deeply analyzing the outcomes and discussion so that it can be evaluated in terms of both placement computation time and optimization quality and also various workload features can be incorporated to make it stronger. On average, CPA is observed to generate placements with only 4% optimal lower limits acquired through LP formulations.

Nevertheless, the limitations of resource capacity and application models are perceived very simply in the optimization framework. First of all, in CPP there are one computer and one data storage component and modern applications are already infused with a synthesis of several computer components communicating with each other and also with numerous data storage components. Next, the CPU solely accounts for the demand of computing resources while the problem becomes multidimensional with memory and other characteristics dependent on the OS [77]. Moreover, the cost involved in reconfiguration or overhead is not assumed because of decisions regarding new placement, where the factors of data movement and VM migrations would be prevalent. Lastly, CPP formulation does not consider any capacity limits of network connection.

2.4.4.2 Network and Data Location-Aware Approaches

Several works on Virtual Machine placement have considered the physical node resources, e.g. CPU and memory with different objectives [78] placement of the VMs that have similar page content in one physical to reduce the aggregate memory footprint, [79] minimize the number of VMs in hosts to improve the effectiveness of dynamic placement, and other approaches focus on how to minimize CNs resources cost usage [80] [81]. None of these approaches consider data access time, which may impact on the application performance.

2.4.4.2.1 Fetching Data remotely

An issue is related to obtaining and sustaining the desired level of performance in the

applications of the data-intensive cloud where data need to be frequently transmitted from storage blocks [82]. The focus of this research area is modern centres of cloud data, including storage clouds (e.g. Amazon S3, EMC Storage Managed Service, iCloud, Dropbox and Google Drive) and compute clouds (e.g. Amazon EC2, Enomalism Elastic Cloud, Google Compute Engine and Microsoft Azure) where the related data can be accessed by hosted applications through intranet or internet using logical or physical communication paths. Furthermore, according to the researchers, random storage of data is possible and can be disseminated through many storage clouds or a single storage cloud. In addition, time of data access is not considered while assigning the applications by the brokers. It can result in such decisions of placement to be allowed to access data going a redundant distance.

Two algorithms were aimed at after searching extensively for solving the problem mentioned above: the VM migration approach and VM placement approach. Models of each application for the solutions bear a set of data blocks that are disseminated throughout various physical storage nodes having different distances (either of physical and logical) from the nodes of physical computing. A Speed(s, Δt) function has been used in modelling network speed between SN and CN on the basis of packet transfer time slot Δt and size of data s. Ultimately, data access time is calculated corresponding to every CN in terms of product sum of the size of every data block and the inverse value of commensurate network speed. All the new requests for application deployment are operated by the VM placement algorithm, which also searches extensively through each credible CN so that one with the lesser data access time can be found in regard to commensurate data blocks of submitted VM, which depends on the fulfilment of the resource capacity limits of the compute node. The boundary specified with SLA being exceeded by the application execution time activates the VM migration algorithm. At this point, one having less data access time commensurate with data blocks of migrating VM, can be found through another extensive search into credible CNs, which also depends on the fulfilment of capacity limits of the computer node resource.

The simulation toolkit of CloudSim confirms the effectiveness of the proposition of

the algorithm [37]. The centre of this assessment is the average time taken for task completion and the proposition of the algorithm is verified through the default policy of VM placement of CloudSim 2.0, such as on the most rarely used host abiding a load balancing approach to the VM is assigned by the VMAllocationPolicySimple. The small scale of DC consisting of three CNs having fixed capacities of resource, three data blocks, three VMs, and two SNs construct the simulation. The proposed approaches seemed to require less average time for task completion and it is signified by the optimized location of hosted VMs. Changes are brought into the network status matrix for activating the proposed algorithm of VM migration, which consequently migrates a few VMs to hosts out of which a more minimized average time of task completion emerged.

Apart from focusing on a much less complicated perception of allied DCs of the cloud, migration algorithms and the idea of VM placement carries out an extensive exploration where a larger DC may not be measured. Furthermore, the performance of the experimental evaluation is limited in a very small scale compared with the completely network-agnostic VM placement. In addition, the solution strategy or problem formulation does not regard reconfiguration overhead or VM migration. In addition, there are one computer and one data storage components and modern applications are already infused with a synthesis of several computer components communicating with each other and also with numerous data storage components.

The work of Hallett et al. [83] designs an optimized algorithm for VMs placement that considers large-scale data in distributed cloud computing infrastructure. Their approach takes into account some network factors, e.g. bandwidth availability and latency between source and destination. The case used for The paper is healthcare, so the consultant can view and process images (e.g. X-ray or ultrasound) to make a diagnosis through a web browser using HTTP protocol and at the same time download all patient records, which include images to his workstation using FTP protocol. If the workstation does not have enough resources, it migrates all data to fulfilment DCs. [84] proposes a job allocation approach that is based on Particle Swarm Optimization. The approach is used to minimize the application total completion time for the jobs execution

time and data transferring time among nodes on DC. This approach does not use VM placement and assumes they are placed statically where NADI approach places VMs dynamically based on network condition and CNs resources availability.

Work by Karimi et al. [85] investigates VM placement and migration that consider about bandwidth availability using CloudSim simulator 2.0. This algorithm chooses the optimal CN that has the smallest data transfer time between CN and SN for the required data and allocates the VM on it. In this respect, the algorithm is moving only the placement for a VM based on lowest data transfer time, whereas, the data are still resident in the same SNs. Similarly, the work done by Chang et al. [86] has the same idea and results as [85]; however, the network concerned is a wireless network for datacentres and applied for mobile applications only. The wireless network is only the concern in this approach and applied to mobile applications. The limitations of these approaches are not about the CPU attributes, how CPUs speed to finalise applications jobs, and how long jobs need to wait in the queue. These may lead to the degradation of applications performance, especially if jobs have to wait much longer in the processor queues.

The above works [82, 86] are the recent works most closely related to NADI, considering data location and attempting to find the optimal CN with the minimal data transfer time. However, these works only consider one file for every VM, while the work here considers more than one file and may be distributed on different SNs. Another difference is the computing, CPU power and loads needed for the job. Every applications jobs need data and computing resources, and these jobs have different processing times, based on the CN processor and different starting times and depending on the processor queues. Unlike previous works, which focus only on data transfer time for completion time, this paper provides a general solution by considering both the computing and data transfer times, aiming to minimise the average completion time and maximise the application performance.

The works [85, 86] are the closest and most recent related works to the work here, which considers data location and tries to find the optimal CN that has the minimal data

transferring time. These works only consider one file for every VM; however, this work considers more than one file and may be distributed on different SNs. In addition, the second difference is the job's computing needs and CPU's attributes. Every application's jobs need data and computing resources. These jobs have different processing time based on CN's processor and different starting times depend on processors queues. Unlike previous works that focus on data transferring time only for completion time, this approach provides a general solution by considering both computing time and data transfer time that aims to minimize the average completion time and maximize the application performance.

Work by Sato et al. [87] proposes VM migrations based on I/O intensive cloud applications. This work focuses on minimizing access time for files by migrating the VM that has application to a proper CN. The objective is to enhance applications performance during VMs reallocation, where target files are still resident in the same SNs. This approach has a very weak assumption to improve network performance by migrating VMs between CNs. In addition, it should consider the VM placement first to avoid unnecessary migration that may affect network performance. Meng et al. [32] have investigated traffic awareness for VMs to improve the network scalability by allocating VMs with heavy traffic at the CNs that have the minimum of total communication cost. The total communication load in their work is defined as the total number of hops between CNs; however, CNs with a small number of hops usually do not mean the communication cost and internal traffic are also small. In addition, their work does not consider data location to minimize data transfer time, network bandwidth and processor time for jobs, which may lead to the degradation of application performance.

In [32], the authors have investigated traffic-aware VM placement in DC networks. The main motivation for this approach is to improve the network scalability by placing the VMs with heavy traffic at the servers that have the minimum of total communication cost. The communication cost is defined as the number of hops between host machines. However, the hosts with a small number of hops usually do not mean the communication cost and internal traffic is also small. In [88], the authors consider minimizing the

routing cost between VMs for the long term. This approach has optimized the VMs placement to reduce the cross traffic among VMs and routing over multiple paths to increase efficiency in link utilization. The limitation in this approach does not consider data location to minimize data transfer time and network bandwidth and CPU utilities, which may lead to the degradation of application performance.

2.4.4.2.2 Data Replication

Replication technology is very beneficial for distributed systems that enhance reliability and availability. Replication helps in minimising the time of user waiting in cloud computing, alleviating data availability and reducing bandwidth consumption of cloud system through providing several replicas of a particular service at various nodes in the context of cloud computing. For example, if a node failed to carry out the user's request, The scheduler will replicate the request to another node to ensure that the application progress is not affected [89]. Two categories can be deduced out of data replication: dynamic replication algorithms and static replication optimisation. The replicas according to their numbers and locations are predetermined in a static replication. On the contrary, replicas are dynamically developed and omitted in dynamic replication following the change in the conditions of environmental load [90, 91].

The proposal in [87] consisted of a model-based algorithm on an approach centred on data where VM-based migrations are used on the ground of I/O cloud applications. This research mainly dealt with finding ways to diminish the file access time with the use of a reallocation algorithm to bring out the most effective strategy to transfer a VM to the location of target file within the storage while minimising the file access time. The location and size of the target file are provided, which allows for estimation of the network throughput between the sites; this offers a weak assumption for migrating a VM from one host to another to elevate the network performance. Thus, unnecessary migration should be avoided in the VM placement in the present work to preserve network performance.

Bandwidth hierarchy-based replication (BHR) was constituted by Park et al. [92] where data access time gets reduced as network-level locality gets increased by circum-

venting network congestion. The sites have been segmented into multiple regions and here network bandwidth of a region is more than the bandwidth between regions. Thus, the file will be fetched faster if the necessary file is dropped into the same region. There are two demerits of BHR. First, it aborts if a region is found with an existing replica and second, instead of the appropriate sites each requested site gets incorporated with the replicated files. The performance of the BHR strategy is good only with a small capacity of the storage elements. The BHR strategy [92] is extended in the modified BHR [93], which carries out replication of mostly accessed files and it may be applicable in the coming days as well.

This approach comprises a scheduling algorithm and a replication algorithm for a three-level hierarchy [94]. A hierarchical network structure with three levels came to their mind. The criterion for selection from the candidate replicas was one with the highest bandwidth for the requested file. Likewise, it deleted files using the same method. Comparatively, this enhances the performance over the least recently used (LRU) method. The most effective LAN, site, and region are chosen by them to develop a competent scheduling. A region having the highest number of requested files is defined as the best region (LAN and site).

A dynamic hierarchical replication (DHR) strategy was developed by Mansouri and Dastghaibyfard [95] that reserves replicas within the appropriate sites that access a specific file most compared with file storage in several sites. Here, the access latency is also reduced as the best replica is selected during replicas being held in various sites. The most useful replica location is chosen by the proposed strategy of replica selection for conducting the jobs of users and the replica requests that wait at data transfer time and storage are accounted for as well. According to the simulation outcomes, its job execution time is briefer than other strategies with a considerably smaller storage size of grid sites.

A system for improving the authenticity of other QoS is offered by Hussain and Mousa [96]. A schedule broker handles the replication process and all information regarding the number of replicas and their location spread into several DCs are found

in it. The following data file to be asked for is predicted by the current data file request pattern. The access frequency of data due to its low computation time is assumed in prior by using the linear series technique called Holts Linear and Exponential Smoothing (HLES). Replication is initiated through the less replication factor of those popular files than threshold. A file is no longer requested for data access if it loses its popularity and in this case replicas of those files should necessarily be erased. This work only considers replication for job completion time, where our approach has two mechanisms for accessing data fetching or replication based on shorter completion time plus considering the queuing and processing time for every job.

2.4.4.2.3 Pre-Fetching Data transferring time

The effective running and management of scientific workflows is made possible by distributed computing platforms. Some of the well-known systems for management of workflows include DAGMan [97], Pegasus [98], Taverna [99], and makeflow [100]. The Pegasus system is mainly used for parallel job scheduling [101], where it provides a task clustering technique for grouping parallel jobs according to a selected and fixed parameter such as the number of groups. However, this technique has many limitations because significant properties such as job properties may be ignored. Nevertheless, the dynamic task clustering approach has been developed to curb this limitation. One such approach is the Falkon Project [102]. In the Falkon approach, workflows are generated dynamically and tasks are clustered for computer resource sites. The provision of in advance network path reservation utilizing OSCARS by networks that are connection oriented, for example ESnet [103] and Internet2 [104] and the expectation of rapid growth of the amount of data in the future, the importance of data movement and placement become more significant factors in work scheduling algorithms. As such, the latest work scheduling programs pay attention to the optimal computation of resource allocation and efficient movement of data [105].

Compared to NADI, these approaches do not take into account the data location or the time cost for transferring these data, especially when a number of jobs require fetching data from remote sites or even sending the data output back. In such a case,

the overall performance can be improved significantly by mapping data to a common site, overlapping it, and transferring it in a pipelined manner.

Based on multiple simultaneous requests, Subramani et al. [106] proposed a greedy meta-scheduling algorithm. Their meta-scheduler that had the capacity to identify the sites that had the ability to start the job first. However, this approach is only suited for resources that are homogeneous besides disregarding data requirements. Nevertheless, authors such as Wldrich et al. [107] proposed a meta-scheduler with the ability to co-allocate resources that are of the random type. They wanted to realize a sequential allocation of multiple resources having particular QoS requirements while considering heterogeneity and site policies that are different. However, the work did not pay particular attention to data-intensive applications.

Other researchers, such as Ranganathan and Foster [108], came up with a data-intensive application based on data movement algorithms and a family of job scheduling. The application improves performance by use of data replication. Data duplication or data scheduling strategies and computation scheduling in the proposal of [108] are independent of each other. The two are not incorporated to ensure the job is done using the best resources. For the data grid environment, Chameleon [109], a resource broker was developed. Based on the job turnaround time, the broker proposed a family of cost models. These are used in making decisions on how a job should be scheduled when it is submitted. To obtain the most suitable scheduling performance, the data or the application code can be shifted. Bent et al. [110] provided a discussion on the scheduling of a jobs collection with data requirements. They presented an extended version of the Condor ClassAd mechanism that allowed the worker node to include information on the files that are present on the node. However, their work was mainly designed for cluster environments and thus unsuitable for grid computing, which is more distributed. Besides, their objective was throughput maximization and the reduction of data movement costs, whereas the objective here is the improvement of turnaround time of the job.

Jung et al. [111] are the closest and most recent related works to this work that

considers the overlapping data transfers during the computation phase. The mechanism is to execute multiple jobs across multiple computation nodes CNs. Basically, a job has a sequence of three phases; stage-in, executing, and stage-out. In the stage-in, the input data for the computation are moved to the CN and computation on the data follows. Then, the output data of the computation is sent out to a Storage node SN where data are resident. This overlapping mechanism improves the jobs execution times in distributed systems.

In comparison with the contribution here, we both design an efficient parallel execution model that aims to minimize the wasting time for data stage-in and -out during executing jobs. However, our approach has additional features, which are minimizing the queuing time and data transferring time. So, our scheduling decisions are made based on both computation requirements and data requirements. It provides a general solution by considering computing time, data transferring time and wasting time for data stage-in and -out that aims to minimize the average completion time and maximize the application performance.

2.5 A summary of the Network-aware algorithms evaluation approaches

The following Table 2.2 shows the existing VM placement and migration approches proposed by both academia and industry consider various system assumptions, problem modeling techniques and the features of the datacenters and applications, as well as different solution and evaluation approaches.

Approach	Allocation type	Modelling	Constrains	Objectives	Solution algorithms	Platform evaluation	workloads	Performance evaluation
[2]	Placement	NP-hard Integer Quadratic Programming	CNs resources capacity. CNs links bandwidth capacity.	Min of the max ratio of the demand and capacity across all network cuts.	Greedy Heuristic	NS2 simulation based using IBM ILOG CPLEX mixed integer Mathematical solver	Gaussian distribution based inter-VM and VMgatway traffic demands. CNs resources capacity and VMs resources request.	Worst case and average network cut load ratio based on average packet delivery delay
[23]	Placement and Migration	NP-hard combinatorial optimisation problem. Instance of Quadratic Assignment Problem (QAP)	Max placement of one VM per CN.	Min the aggregated traffic rates at each network switch. Improvement DCs scalability	Greedy Conquer strategy Min k-cut graph	Trace-driven simulation using global with classical gravity model.	Inter-VM traffic rates collected from production DC	Reduce the traffic that collected from traces collection production in DCs by 20%
[35]	Migration	Mathematical optimisation framework	CNs resources capacity. Inter-VM bandwidth requirements.	Min power cost. Min data transmission	Greedy	Simulation based on synthetic DC and load characteristics	Normal distribution based on CNs and VMs characteristics. Inter-VM traffic demands.	Reduce traffic rate. Min number of used CNs.
[43]	Placement	Graph search	CNs resources capacity. VM Anti-colocation	Min network utilisation Low decision time	Greedy Recursive backtracking	Simulation (JgraphT lib)	Workflow structures: pipeline data aggregation epigenomics	Network utilisation
[49]	Placement	Optimisation framework that uses max-min fair model	CNs resources capacity. Inter-VM bandwidth requirements.	Max CN utilization. Min network traffic in DCs	Two-staged Greedy Heuristic	Simulation based on synthetic DC and load characteristics	Normal distribution based on CNs and VMs characteristics. Inter-VM traffic demands.	Reduce network traffic by 82% average the impact of migration
[50]	Migration	Mathematical optimisation multiple Knapsack problem.	CNs resources capacity.	Min network overhead caused by VM migration	Greedy Heuristic	Simulation based on synthetic DC and load characteristics	Normal distribution based on CNs and VMs demand. Normal exponential, and uniform distributed based Inter-VM traffic demands.	Reduce in DC traffic based on objective function value

Table 2.2: A summary of the selected evaluation approaches

Approach	Allocation type	Modelling	Constrains	Objectives	Solution algorithms	Platform evaluation	workloads	Performance evaluation
[51]	Placement	NP-hard Integer Quadratic Programming	CNs resources capacity. CNs links bandwidth capacity.	Min of the max ratio of the demand and capacity across all network cuts.	Greedy Heuristic	NS2 simulation based using IBM ILOG CPLEX mixed integer Mathematical solver	Gaussian distribution based inter-VM and VMgatway traffic demands. CNs resources capacity and VMs resources request.	Worst case and average network cut load ratio based on average packet delivery delay
[52]	Placement and Migration	Mathematical optimisation	CNs resources capacity.	Min communication traffic Min power cost	Greedy Heuristic	Simulation based on synthetic DC and load characteristics	Workload traces from production DCs	Reduce the overall traffic. Reduce the number of active CNs.
[58]	Placement and Migration	Simple peer based Inter-VM communication pattern	CNs resources capacity. Inter-VM bandwidth requirements.	Min energy consumption. Network components. Average network utilisation.	Greedy VMs based on the number of I/O traffic flow	CloudSim simulation based on network and memory subsystem implementation	NPB parallel application benchmark used as HPC application	Uniformity of VM placement on CNs, Average utilisation of network links and Application performance degradation
[61]	Placement	Proportional fairness Convex optimisation	CNs resources capacity. Inter-VM bandwidth requirements.	Reduce data transmission Min Energy consumption	N/A	Simulation based on synthetic DC and load characteristics	Normal distribution based on CNs and SNs resource demand. Network I/O rate. Inter-VM traffic demands.	Reduce rate of traffic volume based on objective function value
[64]	Placement	Knapsack and stable marriage problem	CNs CPU capacity. SNs capacity	Min the total network cost across all application communication links	Two-staged Greedy Heuristic	Heterogeneous SAN. Synthetic workload-based simulation	Normal distribution based on CNs and SNs. Resources demands and network I/O rates.	Use network cost function Placement computation time.

Table 2.2: Continued

Approach	Allocation type	Modelling	Constrains	Objectives	Solution algorithms	Platform evaluation	workloads	Performance evaluation
[72]	Placement and Migration	Analytical optimisation	CNs resources capacity.	Min response time between CNs and SNs	Exhaustive search	Fixed simulation scenario. Implemented in CloudSim 2.0.	Small scale and fixed valued workload data.	Average jobs completion time reduced between 12s-100s.
[77]	Placement	Analytical optimisation	CNs resources capacity.	Min response time between CNs and SNs	Exhaustive search	Fixed simulation scenario. Applied of Mobile Cloud computing platform. Implemented in CloudSim 2.0.	Small scale and fixed valued workload data.	Average jobs completion time.

Table 2.2: Continued

2.6 Chapter summary

This chapter reviews existing literature in the research area and identifies current trends shaping cloud computing. It introduces the concept of cloud computing providing an overview of the current state and the opinions of experts regarding the new paradigm. Then, it explores the motivation and background knowledge related to the network-aware VM placement and migration in DCs. The chapter forms a foundation upon which the rest of the thesis investigates the chosen area. In this regard, grid computing and telecommunications technology have already solved similar problems faced in cloud computing and are widely considered in various studies in the development of the cloud computing paradigm. A comprehensive comparative analysis highlighting the significant features, benefits, and limitations of the techniques has been put forward.

3

A Framework for Virtual Machine Placement

The chapter covered previously provided an insight regarding the state of art and the Network-aware Virtual Machines (VMs) and their schedules in the cloud environment. There was a detailed discussion about the accomplishments of previous and current research. The existing literature was analysed carefully and helped in identifying the gap where further or more research is required. It was concluded that there is no prevailing scheduling algorithm or schedule to co-relate large amounts of data as well as their locations with network capabilities and computations when making the scheduling decisions. It was found out that no present VM scheduling system takes scheduling decisions by including the three parameters into account. In this chapter, there is a description of NADI scheduling algorithm together with the key scheduling parameters and a demonstration of ways to which they influence the scheduling optimization. Sec-

tion 3.2 illustrates scenarios of relevant usage in a concise way. Section 3.4 describes the set of requirements to be employed in the proposed NADI scheduling strategy. This chapter comes to an end describing the scheduling problem which is split into various scheduling costs which are; data transfer, network, and compute time cost. They are derived using appropriate mathematical formulae and a brief explanation is provided in section 3.5.

3.1 Introduction

Cloud computing together with cloud scheduling adheres to the Service Oriented Architecture (SOA) model [7] as shown in figure 3.1. Every component acts as a service, has a behaviour and performs autonomously. These characteristics require a strategy that is adaptive, descriptive and iterative to enable every service or component to be modeled in autonomous manner. However, every component should work together homogeneously and cohesively in its entirety. Scheduling theory in cloud computing is gaining priority every day hike in cloud popularity [112]. Scheduling refers to the mapping process which takes the tasks to the resources which are available depending on the tasks requirements as well as the characteristics. It is an imperative feature in ensuring effective functioning of the cloud since different task parameters require attention to ensure suitable scheduling. The available resources need to be utilized effectively without interfering with the facility factors of the cloud. There are three major phases in the scheduling process. Phase one involves discovering the available resources. The second phase is concerned with collecting information about the available resources and selecting the best so as to match the requirements of the application (usually known as matchmaking phase). This where this thesis revolves making the major contribution. Phase three is concerned with executing the job which also includes cleanup and file staging.

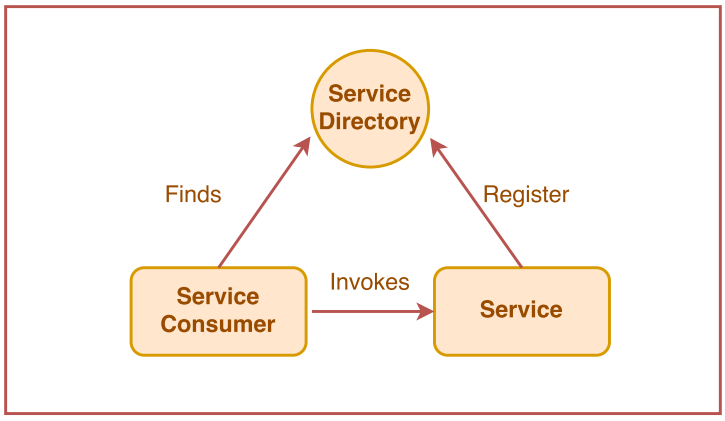


Figure 3.1: Service Oriented Architecture

In this chapter, the research approaches and issues in optimizing the matchmaking stage are described in detail. In this phase, the appropriate match between tasks and resources is the main concern. There have been several heuristics proposed to achieve the optimal match as argued in [113]. However, in data intensive scheduling, there is the need to include network characteristics in the scheduling decision the as opposed to the existing matchmaking approach which does not include network parameters. To improve the utilization and efficiency of the cloud system, network parameters should be embedded in the scheduling algorithm. The main aim is to reduce the time required in computing in applications which have large-scale data. In task scheduling, there are two goals which are: high performance computing and high throughput computing. The former aims at reducing overall execution time for every application and it is best used in parallel computing. The latter aims at scheduling autonomous tasks to boost the processing capability of the system over time interval. Here a high throughput strategy is discussed because many data intensive applications consume a lot of time data transfer and data execution operations. Chapter 2 reviews pointed out challenges and prospects of cloud computing as an incipient paradigm. From the information contained there in, it is clear that to enable a proper cloud model, new approaches should be investigated so as to address the challenges posed by the growing popular paradigm. As the cloud model continues growing, challenges such as resource, management of very large DCs increases which raises an emergence of newer cloud applications [34, 114–116].The cloud services need a resource manager which is responsive and has the capability of decreasing manual intervention in the cloud life cycle. The resource managers should have the capability of activating strength in the target applications as well as endowing the superfluous cloud computing visions.

This chapter will discuss the Network-aware and Data-intensive (NADI) VM Scheduling strategy which takes into consideration processing power, data and network characteristics while making the scheduling decisions among multiple sites to overcome the challenges. The NADI strategy looks at the cloud as a group of active network elements [117] (the network acts as the component of cloud which provides a guaranteed

and reliable service levels) and therefore, gives the network characteristics first priority in the scheduling decision matrix with data computations. The scheduler makes intelligent decisions by considering the network changing states, size of data and the processing pool cycles. There is a description of the NADI scheduling algorithm as well as constituents of the main scheduling parameters in chapter 2 and a demonstration of how they influence scheduling optimization. The scheduling problem is split into various scheduling costs which are: data transfer, network, and compute time cost. They are derived using appropriate mathematical formulae and are explained. In the following chapter, various system aspects are demonstrated from requirement point of view and different features of the strategy are employed in order to capture the requirements.

3.2 Usage Scenario

This part presents a classic use case scenario. This thesis studies three application cases for the usage scenario.

1. Physical Infrastructure Providers (PIP) - Is an entity which owns the IT resources and network and makes them available to the rest of entities on a pay per use basis.

2. Virtual Infrastructure Provider (VIP) -Is an entity which has access the physical resources of PIP and then makes them available to the market place after they have been abstracted, sliced and aggregated. VIP commonly acts and performs PIP functions.

3. A broker is responsible for aggregating the service from various VIPs and maps the requests of different cloud users to the services. The broker acts as an interface which organizes and coordinates the interaction among other actors. To cater for the needs of a user request, the broker consults VIPs resources and analyses the prevailing resource pool to consider the summative satisfaction for the succumbed request. Users submit requests for the particular composition of IT and network resources to the VIP and use the available resources to run the applications or computing any other job.

3.2.1 Single-Source Data

Every VM needs particular amount or number of computing memory, disk resources and power to operate. Additionally, every VM needs right to use particular data records that are warehoused in a disseminated storage hubs through the cloud. The archives consists of data and executable archives which needs to be moved to VMs prior to starting their process. In the current VM placement approaches, the data locations have been ignored due to the large size of files needed, the delay of data transfer is noteworthy and increases considerably in the required time by VMs to finish up their jobs. This is an imperative aspect when initialization takes place where there are huge files which need to be moved to the virtual machine. The steps followed in the scenario here are similar to the one used by Piao and Yan [82]. They have suggested a placement algorithm which allocates VMs on CNs which has the least data transfer delay. They have made an assumption that only a solitary VM can request the transfer of a file at a particular time.

3.2.2 Distributed Data Storages

The approach used previously makes an assumption that a single VM can only request transfer of a file at a time, hence it fails to address the issues of optimal rate distribution. According to Amazon EC2 who is among the well-known computing providers compared to the rest of resources, rates of data are not dedicated but are distributed and shared by the VMs Amazon EC2. Here the thesis makes one of the contributions as it will be discussed later in chapter 4.

3.2.3 Large-Scale DC Traffic

In the IT industry, cloud computing has been given much attention whereby the computing infrastructures, software application services and platforms are provided at a fair cost from high scale DCs which are accessed over the network. The cloud has a number of benefits to the users such as scalability, reduction of application management as well as overall management reduction and hardware costs. It also provides an opportunity to

leverage the prevailing DC infrastructure and grabs the advantage of economies of scale which is available to the ones who purchase huge volumes of network and hardware capacity. This scenario uses the large scale traffic and huge data volume required in order to show how NADI provides the guarantee levels of performance and the Applications efficiency.

3.3 Proposed NADI framework

The proposed algorithm takes into account not only computation resource requirements, but also data requirements (storage space, network condition, etc.) when making scheduling decisions.

3.3.1 Main components of NADI framework

The NADI framework has the four main layers which are Delivery, Service, Virtualization and Control and Physical Layers. This section gives a brief explanation of every layer. IBM's open cloud architecture [118] and NIST [22] have been considered.

3.3.1.1 Delivery layer

The Delivery layer enhances submission of requests from the user and offers to enable space for demands from the user such as specific VMs configurations, delay, number of needed VMs, response time, and communication protocols are properly identified. They are then submitted by use of a suitable description language or using set business objectives such as cost, auditing, deadline, and accounting. The NADI delivery layer is composed of four major components as illustrated in figure 3.2 which interact between them to recognize the functionalities. The consequent parts give the functionalities and responsibilities of the Delivery layers components.

3.3.1.1.1 The Service console: relates to user requests as accept or update and comes up with specifications of the workload whether complex or simple. In order to support the workload, the description of the user requests should support:

1. Defining application timeline for the job execution (this parameter is helpful in co-scheduling of the resources)
2. Resource group requests for example storage, network, and computation
3. The depiction function which is assigned to the resource and should have the flexibility to capture future and current capabilities.
4. The narration of the relationships which exist between the resources. The topology of the network which depicts particular virtual links and nodes, intra-communication, and node-node should be apprehended.
5. The depiction of the tools and applications required for every component for example programming tools and operating systems.

3.3.1.1.2 Report interface: is responsible for providing monitoring capabilities to report (to the NADI manager) virtual resource status, jobs, and the activities. This module ensures all the activities in the database are maintained and updated in all component per every request regardless of whether it is multi or single. It keeps a record of finishing times and offers time series data to NADI manager which is used in prediction of upcoming resource status.

3.3.1.1.3 NADI manager: organizes and coordinates all the dynamic characteristics of the user's VMs. This module has privileged functionalities to enable it to perform the service functions and resource management at this level. This efficiently decreases the space for management hence making it correct and small. NADI manager controls all the incoming as well as the outgoing traffic. The major or central function offered by this module is the template-based mechanism which controls and organizes component activities in the user request. Every user request is recognized by the template that is used by the manager to build the components, reconfigure prevailing components and achieve the interactions in the fundamental infrastructure together with other users VMs. The simulated switch which connects the component created newly (whether

storage or computing node) is added the list of switches known as v which are controlled by CM. There are two basic communications which take place in NADI, namely inter and intra user request communications.

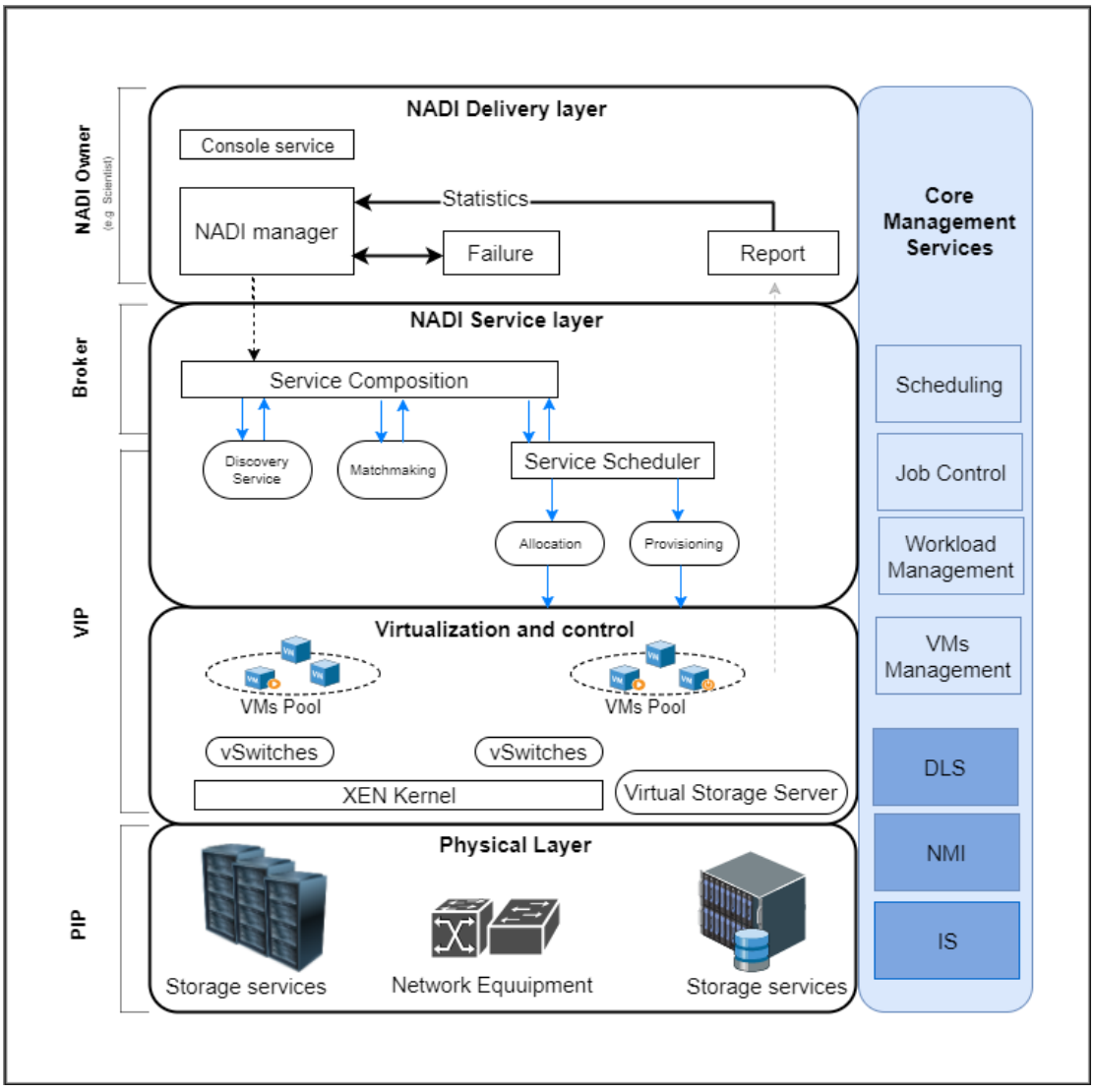


Figure 3.2: The interaction of various components in the proposed NADI Framework

3.3.1.2 Service layer

The service layer facilitates the creation of functionalities of container based cloud service. It performs a number of functions which are;

1. Providing the initial resources (they include network bandwidth, storage, CPU time and memory).

2. Interrelating with the underlying layer
3. Performing extra universal scheduling
4. Offering functionalities for analyzing job requests which are acquiesced by the owner.
5. Arranging privileged functionalities for example job allocation and resource provisioning for NADI users VMs when it is created.
6. Intermediating among all the NADI users storage and VMs in terms of usage and subscription.

3.3.1.3 Virtualization and control layer

In the suggested NADI model, there will be the creation of an adaptive virtual interface (VIF) for every node (VM). A VM is connected by a VIF to a distributed virtual switch (DVS) and forms a virtual link (vLink). Combining the connected virtual nodes and the virtual links 3.2 establishes the description of virtual network topology (VNT).This thesis uses an adaptive vSwitch which is based on traffic shaping approach for NADI. Every VM in a NADI users VMs is assigned to a DVS port which has a starting bandwidth as per the request. It is allowed to expand the capacity according to the unused bandwidth in the user's properties. In this way, VMs are able to adapt dynamically to different traffic conditions being controlled by DVS enhanced with the control protocols flow for example OpenFlow. The traffic seclusion mechanism is applied to DVS which takes control of VM activity and protects the service level agreements of every container. A database of entries for network policies is maintained by the DVS. This module provides the functionalities which are under the control of virtual infrastructure provider. In the seclusion approach for NADI, every VM interrelates with the NADI manager together with the other VMs, spoke arrangement and in the hub to finish on work flow jobs. For a VM which experiences high traffic (for example storage VM), it consumes all the bandwidth which is available for NADI users resources. However, other VMs may need less bandwidth. A mutable super port which has the same capacity of

bandwidth requested by NADI users resources is assigned when instantiating a NADI users resources. The super port can be looked at as the accretion of immovable capacity ports. The available bandwidth for whatever VM is calculated as follows:

$$AB_m = BW_{n,m} + \sum_{n=1}^{v} UB_{n,m} \tag{3.1}$$

where the AB_m the available bandwidth on the CN number m, $BW_{n,m}$ is the required bandwidth for VM number n and $UB_{n,m}$ is the total bandwidth that is used by VMs on CN number m.

3.3.1.4 Physical layer

Physical layer comprises of heterogeneous IT resources which are physically interconnected by the optical and electrical network in a topology which is hierarchical to enhance cloud service. PIP installs the Network Management Services (NMS) while the VIP allows the access.

3.3.1.5 Physical Infrastructure Providers

This entity offers infrastructure for other entities. The huge physical resources such as storage, servers, and networks which are accommodated in data centers are unfilled to enable hosting of the cloud application. This is vital as it supports isolation, abstraction, and aggregation. The datacenter which is arrayed by the PIP is ready and can span various locations as well as administrative control.

3.3.1.6 Virtual Infrastructure Provider

Virtual infrastructure provider abstracts and isolates the PIP physical infrastructure to sustenance the cloud computing model. It uses virtualisation the users provide virtual infrastructure. VIP roles may be found to exist in the PIP domain whereby the abstraction and ownership are done by a lone entity. Amazon is an example of such actor and it provides AWS. AWS is built from the physical infrastructure which is possessed by Amazon.

3.3.1.7 Broker

The broker is responsible for functions such aggregation, arbitrage and intermediation to build an inter-cloud service and also execute the integration of services which were existing to fulfill a request. The broker which is found in NADI is the main element of the market place which enables the IAAS-Users, PAAS developers and providers to build a relationship and come up with economic value. To cater for the needs of a user request, the broker analyses and chooses the offer and then assimilates different services, where they are necessary and then passes them to the user. Other similar roles are in [119] and [120]. The feature which distinguishes broker in NADI is its responsibility of maintenance and ensuring PAAS developers is available to the real world info database of IAAS users request as it offers intermediation provision to the PAAS developer also. With, self-organization, real-time information exchange and service policy creation, transaction traffic in the market place is estimated to raise the cumulative value. It enables adaptability and integration from PAAS and VIP users as solution developers and providers correspondingly.

3.3.1.8 Cloud User

Cloud user employs the service console or any suitable interface to initiate a request for a user. The actors make specifications concerning the set of resources required as well as the linked workflow needs for the user. The request made could be for example VM images set with a particular operating system as well as specified application domain which are installed on every VM.

3.3.2 The individual job's perspective scheme

In the figure 3.4, the scheduler is not aware of data in a traditional execution model. The job stands in the queue as it is only executed upon obtaining adequate resources. With the commencement, input data are pulled through the job from a distant location and after computing the output data are delivered to a distant location as well. The job carries out stage-out due to data stage-in and allocation of computational resources

into job is even done in the middle of data movement as well. Therefore, the course of data transfer consumed computational resources fully.

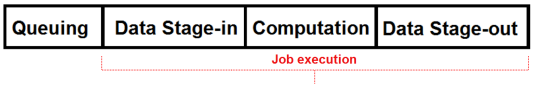


Figure 3.3: no data-intensive and no CPU utilities

Figure 3.5 shows the first proposed [121] and its conventional execution model, considering both data locality and computing resources. The scheduler calculates the computing time and data transmission time for all jobs and chooses the optimal computing node. In this model, the input data is fetched from a remote location but the chosen computing node will be that one that has the latest data transmission time, queuing time and execution time. This model can save time by considering data locality and network status.

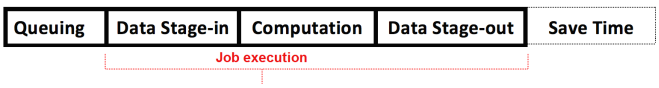


Figure 3.4: data-intensive, Network awareness and CPU utilities

Figure 3.6 illustrates data staging prior to job submission through scheduler and after the job completion scheduler stages it out. Compared to the previous one [121], resource utilization can be enhanced more with this solution [122]. In this model, the data will be replicated to the chosen DC which job will be executed there. The replication stage is before the job starts execution or even queuing.

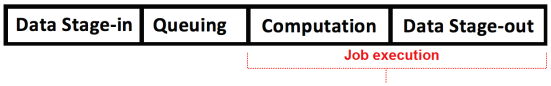


Figure 3.5: data-intensive and no CPU utilities using data Replication

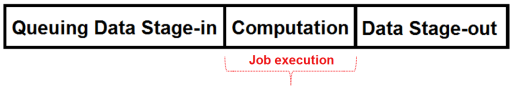


Figure 3.6: data-intensive and no CPU utilities using (Overlapping)

The assumption is that a job is composed of three stages: data transferring-in, ex-

ecution and data transferring-out, as shown in figure 3.7. In the first stage, the input data for the job is transferred from the precedent job CNs. In the execution stage, the job runs with the input data and produces output data for descendent jobs. In the data transferring-out stage, the output data are transferred to the descendent job or are submitted to the last job in the workflow. The scheduler calculates the computing time and data transmission time for all jobs and chooses the optimal computing node. In this model, the input data is fetched from a remote location in advance to the chosen computing node, which has the latest data transmission time, queuing time and execution time. Then, every job's data will be staged in advance during the computation time of other jobs (overlapping). This approach attempts to reduce job turnaround time by decoupling the SI and SO from the job execution and utilizes the computational resources. Compared to the previous approach, this solution enhances resource utilization. The execution procedure for all stages is executed as in figure 3.8.

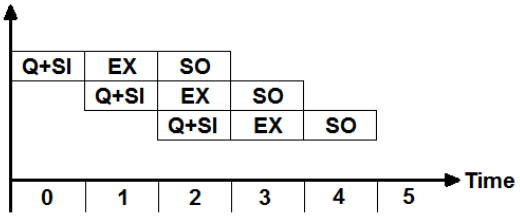


Figure 3.7: Classification of management techniques for data intensive application workflows

3.4 NADI Requirements

The eleven points below, summarise the major requirements for the NADI scheduler. Then, the following sections analyse these requirements that have a direct bearing on the performance needs of applications and these are described here as Compute-Related, Data-Related, and Network-Related requirements.

R_1 A consideration of Data Location is highly recommended when scheduling decisions are being made. The choice of data location is determined by delay and bandwidth of a location.

R_2 Scheduling Data Intensive Jobs should include the best replica of dataset. As a result, data transfer time can be reduced.

R_3 Assigning of jobs to resources requires to be done under the needed data set, as long as it considers cost, effectively.

R_4 If it is possible to schedule and execute Data and Jobs, they require to be transferred to a third location.

R_5 Consideration is needed to be taken before making a decision of a job or data movement concerning computation cost, data transfer, and network.

R_6 Jobs should be done within the user stated time limit by a scheduler through supporting of time constrained mechanism. Actual time scheduling that is applied in multimedia applications is not time constrained and deadline scheduling.

R_7 Scheduling of all jobs should be done in a location that contain the needed dataset, whose access is possible for all the jobs.

R_8 The job may need to recover portions of the data from a remote dataset.

R_9 A best network link in terms of the latest location-location calculations between storage and computing component should be searched by scheduler.

R_{10} It is required for the scheduler to make sure that the requirements software is accessible for the job implementation on CN that was selected.

R_{11} The scheduling system should calculate and incorporate the network measurements before planning a job submission.

3.4.1 Analysis of the requirements

Thus, brief analysis concerning the main requirements for the NADI scheduling process is presented. The specific requirements connected to NADI scheduling are expanded and explained in this section. At least three main requirements areas for the Cloud Meta-Scheduling System, which contain a direct bearing on the performance applications

needs are identified. Additionally, they are termed as compute-related, data-related, and network-related requirements.

3.4.1.1 Compute-Related Requirements

In considering to the specific requirements of the applications and those of the end-users, scheduling and resource management services need, which can rapidly and optimally locate high-performance computational resources is driven by compute-intensive applications. Not only the compute connected needs for an application including load, available computing capacity, and data location [123], but also the main factors [124] including application time for running, the cost and availability of the resource (R5), and application effectiveness on specific resources, should be considered by resource management services when locating and scheduling of jobs on the resources. Also, machine failure rate (R11), the network characteristics for moving results to mass storage (as shown in section 3.4.2) or the user quota and privileges on the projected resource together with any firewall security policies that could prevent the implementation of remote resources are other factors that needs to be considered. A method to range the jobs for offering quality of service to particular jobs is needed within the limited time as users can submit any amount of jobs freely (R5-R6). When the job load is high, jobs should be exported to the least loaded locations by the scheduler through self-organization (R11). As a result, jobs will be saved from starvation, to an extent of assisting the optimization of scheduling. Knowledge of steering and moving of the jobs to the locations that have fine resources is needed from the scheduler (R10). Not only consideration of these requirements is needed but also dictation for the need for applications to adequately define the needs.

3.4.1.2 Data-Related Requirements

Constraints concerning how the scheduling of computational resources is done in a Cloud environment are placed by most applications as they contain various basic file management and data placement needs [124]. A collection of input datasets is used

a base by applications, where this leads to production of output datasets that require analyzation or processing by a second application. The aim here is to visualize the data that is produced. Bearing on some applications scheduling is identified on input and output datasets size and the amount of the entire storage space selected for a user [125]. Consideration of the time needed to transfer the datasets when scheduling computer resources for the particular application is encouraged if the input datasets need to be recovered from a remote location (R8).

The scheduling performance is identified to have various methodological issues. one of these issues is Data-aware scheduling, which involves scheduling of the tasks closer to the data is needed. [126, 127] (R3). Firstly, consideration of the amount (and speed) of data transfer, which is needed in a task execution leads to attainment of considerable performance advancements [128]. Secondly, as datasets may be transferred amongst various database management systems, the contents of files can be accessed effectively. Thirdly, the scheduler should have provision to transfer data towards the computation location. Though, this would be achieved if there is guarantee of considerable performance gains by the data transfer and placement. In this case, the high available bandwidth in modern optical networks is enhancing its achievement (R7). Section 3.4.3 has more information. A key to the Data Intensive scheduling is data location, where scheduling optimization is impossible if data location is unknown (R1). The scheduling optimization can benefit from data location and probable decision to replicate the particular dataset (R2). Likewise, other significant parameters that should be combined in the Data Intensive scheduling process include data transfer and access costs. This process sometimes include replication, where the new dataset is used by the jobs when scheduled at a specific location. The purpose is to advance the scheduling optimization. Additional, a single location, if its cost is affordable, can be used to schedule all the jobs in a job burst (R3).

Various tertiary storage systems are used to store read data by Data Intensive jobs [129]. These tertiary systems can be accessed either within seconds or hours [130]. This happens when data exists in unmounted tape. It might turn out to be more cost effective

for the data intensive jobs to access and transfer data over the internet from the remote locations compared to accessing the tertiary storage in the local environment due to latest trends in the network performance (R5). Thus, consideration of tape access and related latencies during data intensive jobs scheduling is needed as their size cannot be compatible with that of the network cost as theirs can be larger.

3.4.1.3 Network-Related Requirements

A significant role in scheduling optimization is played by the fundamental network between submission and implementation locations together with the connection of the location storing when a dataset replica for Data Intensive scheduling is being selected. Thus, a finest network path between storage and computing elements should be considered by the scheduler (R9). Priority in replica selection that contain effective and reliable networks is required to be given to those locations (R4). A primary element in the Data Intensive scheduling is the network, where calculation and integration of the network measurements whilst planning for job submission need to be done by the scheduling system. As a result, Network Aware services and Network Aware scheduling decisions are enhanced. Thus, there should be consideration of computation cost, data transfer and network before any decision in job scheduling is made.

Reliable networks are simply the required factor by some applications whilst high-speed networks for large data transfer might be needed by communication-intensive applications. Networks conditions on the applications to be run needs to be discovered and monitored, at least. Thus, monitoring and predicting of services for network conditions, which can give the capacity to query the characteristics of networks connected to the target computing resources to applications, resource management services, and also users (R11).

3.4.2 Input Parameters and Objectives

The scheduling optimization connected decisions will be based on the parameters that follow section 3.5 which gives reasons for including these parameters. The final chap-

ter of this thesis gives the measurement and experimental process. Direct importance for Data Intensive and Network Aware Scheduling optimization is identified in these parameters. This section will later include their discussion.

- Bandwidths of the network links.
- Computing cycles accessible
- Computing node (Server) loads and respective job queues
- The application executables sizes
- Data file (input and output) sizes
- The data file locations

Maximization of CPU usage and throughput and minimization turnaround time, waiting and response time are the focus of the recommended scheduling algorithm. The measured parameters, where a significant element of the matchmaking process in the Cloud Scheduler is used as a base for scheduling algorithm creation. The targeted metrics within the scheduling process, where the success and optimization level of the scheduling system requires to be measured, are as follows:

- Queue and waiting time
- Processing and execution time
- Input data transfer time
- Executable transfer time
- Results transfer time

The sum of the times will include the total time of job performance in a Cloud environment.

3.5 Time Estimators

There are three major cost estimates which need to be calculated for the scheduling algorithm: the network, computation and data transfer costs and scheduling optimization will be based on these three estimates. before that, the essential variables in the formulation are defined in this section.Then the next sections show the parameters

There is a number x of datacentres (DCs) $\{dc_1, ...,dc_x\}$ and every DC contains of all storages nodes SNs $\{s_1, ...,s_l\}$ that store all Ds data subsets and computing nodes CNs $\{c_1, ...,c_m\}$. Also, there is a number b of data blocks $\{d_1, ..., d_b\}$ that distributed in several cloud SNs with logical or physical distances. Suppose there are n virtual machines $\{v_1, ...,v_n\}$ that need to be placed on m CNs. Every v has given k numbers of jobs $\{j_1, ..., j_k\}$.

3.5.1 Data Transferring time estimation

The first most important cost aspect in data intensive is the Data Transfer Time. A poor network status will lead to a longer data transfer time and similarly a larger data size will take more time to transfer. So we have designed two solutions to access dataset.

3.5.1.1 Fetching remote data

It is assumed that every job needs to access remotely certain data d_j that is stored on storage nodes S. So for all such transfers, the sum of data transferred should equal to the data required by the job, which is d_j

$$d_j = \sum_{s=1}^{l} (d_{s,j} \times a_{s,j}) \tag{3.2}$$

Indicator variable $a_{s,j}$ characterises where each job that has related data.

$$a_{s,j} = \begin{cases} 1, & \text{if job } j \text{ has related data on SN } s \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The equation ensures that the total data for all the jobs are bounded by the sum of data required by each job.

$$\sum_{j=1}^{k} d_j = \sum_{j=1}^{k} \sum_{s=1}^{l} (d_{s,j} \times a_{s,j}) \tag{3.4}$$

The total data for every VM n that donates as D_n should satisfy the following relationship:

$$D_n = \sum_{j=1}^{k} \sum_{s=1}^{l} (d_{s,j}^{n} \times a_{s,j}) \tag{3.5}$$

Now it is possible to calculate the data transfer time for all data required by job j on CN m as follows:

$$JT_{j,m} = \sum_{s=1}^{l} \left(\frac{d_{s,j}}{BW_{s,m}} \times a_{s,j} \right) \tag{3.6}$$

where $BW_{s,m}$ donates the available bandwidth between CN m and SN s.

The total data transfer time DTT for every job is changeable based on the network status between SN and CN, so now the total data access time DTT can be calculated these values stored in the DTT matrix = $[dtt_{m,n}]$ such that $dtt_{m,n}$ denotes the the total data access time for V_n jobs on CN m.

$$DTT_{m,n} = \begin{pmatrix} dtt_{1,1} & dtt_{1,2} & ... & dtt_{1,n} \\ dtt_{2,1} & dtt_{2,2} & ... & dtt_{2,n} \\ ... & ... & ... & ... \\ dtt_{m,1} & dtt_{m,2} & ... & dtt_{m,n} \end{pmatrix}$$

This matrix should satisfy the following relationship:

$$DTT_{n,m} = \sum_{j=1}^{k} JT_{j,m}^{n} \tag{3.7}$$

3.5.1.2 File Replication

There is a number of files $\{f_1, ..., f_e\}$ that are distributed in several cloud storage nodes $\{s_1, ..., s_l\}$and every file could be consist of a number of data blocks $\{b_1, ..., b_d\}$, that could be stored with logical/physical distances. Each file has different size Z_e and should satisfy the following relationship:

$$Z_e = \sum_{s=1}^{l} \sum_{b=1}^{d} d_{b,s}^{e} \tag{3.8}$$

Now the replication matrix $R = [r_{u,e}]$ is defined where $r_{u,e}$ indicates whether a replica of file e is assigned to DC u and $u \in [1, x]$

$$r_{u,e} = \begin{cases} 1, & \text{if any SN in datacentre } u \text{ stores a replica of file } e \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

where is $u \in [1, x]$

Let nr donate the number of replicas for file e that are stored in dc_x.

$$nr_e = \sum_{u=1}^{x} r_{u,e} \tag{3.10}$$

Now every replica size and location among DCs is defined using matrix $FA = [fa_{e,u}]$ where $fa_{e,u}$ indicates that the size for file e is stored in DC u and $u \in [1, x]$

$$FA_{e,u} = Z_e \times r_{u,e} \tag{3.11}$$

Now unreplicated datacentres matrix $UR = [ur_{u,e}]$ is defined where $ur_{u,e}$ indicates whether that the datacerntre u does not have a replica of file e.

$$ur_{u,e} = \begin{cases} 1, & \text{if datacentre } u \text{ does not have a replica of file } e \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

The matrix UR should satisfy with following equation.

$$UR = 1 - R \tag{3.13}$$

The replication time for any file to any DC can be calculated using the following equation.

$$RT_{v,e} = \frac{FA_{e,u}}{BW_{u,v}} \times ur_{e,u} \tag{3.14}$$

where is $u, v \in [1, x]$, $u \neq v$ and $BW_{u,v}$ donates the available bandwidth between datacentres u and v .

The second factor to be considered in file replication is storage writing time WT for all files that need to replicated to dc_u as follows:

$$WT_{u,s} = \frac{\sum_{f=1}^{e} FA_{e,u}}{W_{s,u}} \tag{3.15}$$

where $W_{s,u}$ indicates writing speed rate for storage node s in dc u.

The third factor to be considered also in file replication is the total intra data transferring time LDT inside the chosen datacentre to replicated files for every job between SN and CN. To calculate LDT, it is first necessary to calculate the data transferring time for all data that is required by job j on CN m as following equation.

$$LJT_{j,m} = \sum_{s=1}^{l} \left(\frac{d_{s,j}}{BW_{s,m}} \times x_{s,j} \right) \tag{3.16}$$

Where $BW_{s,m}$ donates that the available bandwidth between CN m and SN s.

Now LDT denotes the the total intra data transferring time for V_n jobs on CN m as following relationship:

$$LDT_{n,m} = \sum_{j=1}^{k} LJT_{j,m}^{n} \tag{3.17}$$

Finally, it is possible to calculate the total transfer time TVM for every VM which includes the replication time for the required files, storage writing time and the total intra data transferring time for all jobs as the following relationship:

$$TMV_{n,m}^{v} = \left(\sum_{f=1}^{e} RT_{v,e}^{n} \right) + WT_{u,s} + LDT_{n,m} \tag{3.18}$$

3.5.2 Computing time estimation

In data intensive situations jobs may be pushed to the data and in computation intensive situations data may be pulled to the jobs. This kind of scheduling approach can lead to performance degradation in a cloud environment and may result in large processing queues and job execution delays due to computing node overloads.

So it is assumed that the processing time for any job j on CN m for a certain size of input is known and calculating as the following.

$$P_{j,m} = \frac{ni_j}{capacity_m \times core_j} \tag{3.19}$$

where ni_j is the total number of instructions that the job will need to execute on a processor, $cores_j$ is the number of cores required by the job and $capacity_m$ is the processing strength of individual elements.

$$capacity_m = \sum_{i=1}^{np} cap_{i,m} \tag{3.20}$$

where $cap_{i,m}$ is the processing strength of individual elements, np processing elements (PEs) and $i \in [1, np]$.

Another factor that affects jobs scheduling is queuing time witch, is the minimum time for a job that has to wait on processor queue until it starts execution. So it should be part of the scheduling algorithm. Now the start time st_j for every job depends on the position in the execution queue on CM m. from previous equation (3.19), the processing time for any job on any CN can be deduced.

$$P_{j,m} = st_j + \frac{ni_j}{capacity_m \times core_j} \tag{3.21}$$

Now the computing time CT can be calculated which considers both processing time and queuing time on every computing node and stores these values in the computing time matrix $CT = [ct_{n,m}]$ such that $ct_{n,m}$ denotes the total computing time for VM n jobs on Computing node m.

$$CT_{n,m} = \begin{pmatrix} ct_{1,1} & ct_{1,2} & ... & ct_{1,m} \\ ct_{2,1} & ct_{2,2} & ... & ct_{2,m} \\ ... & ... & ... & ... \\ ct_{n,1} & ct_{n,2} & ... & ct_{n,m} \end{pmatrix}$$

This matrix should satisfy the following relationship:

$$CT_{n,m} = \sum_{j=1}^{k} P_{j,m}^{n} \tag{3.22}$$

3.5.2.1 Overlapping Data Transfer With Job Execution

This section will give a brief summar about the overlapping technique. Figure 3.7 illustrates the new scheme which has been discussed in section 3.3.3. The broker calculates the computing time and data transmission time for all jobs and chooses the optimal computing node. In this model, the input data is fetched from a remote location in advance to the chosen computing node, which has the latest data transmission time, queuing time and execution time. Then, every job's data will be staged in advance during the computation time of other jobs (overlapping). This approach attempts to reduce job turnaround time by decoupling the SI and SO from the job execution and utilizes the computational resources. Compared to the previous approach, this solution enhances resource utilization. When the overlap technique is applied to the computing time equation, the final form for this will be as the following.

$$LJT_{j,m} = Q_{j,m} + \sum_{s=1}^{l} \left(\frac{d_{s,j}}{BW_{s,m}} \times x_{s,j} \right) - P_{j-1,m} \tag{3.23}$$

where $Q_{s,m}$ donates the queuing time for job j on CN m, $BW_{s,m}$ donates the available bandwidth between CN m and SNs and $P_{j-1,m}$ is the processing time for the previous job $j-1$ on CN m.

3.5.3 The total Completion time estimation

In this section, it is necessary to calculate the total time needed for every VM request which includes Data Transferring time, Computing time and files replication if it is needed . So it depends on which algorithm will be used as in the following sections.

3.5.3.1 Fetching remote data

The Completion time TCT matrix stores the total execution time for every V_n on C_m which includes **queuing time**, **data access time** and **processing time** for each job j .

$$TCT_{m,n} = \begin{pmatrix} tct_{1,1} & tct_{1,2} & ... & tct_{1,n} \\ tct_{2,1} & tct_{2,2} & ... & tct_{2,n} \\ ... & ... & ... & ... \\ tct_{m,1} & tct_{m,2} & ... & tct_{m,n} \end{pmatrix}$$

This matrix should satisfy the following relationship:

$$TCT_{m,n} = DTT_{m,n} + CT_{j,m} \tag{3.24}$$

3.5.3.2 File Replication

In this algorithm, there are several of factors that has been considered. These factors are the replication time for the required files, storage writing time and the total intra data transferring time for all jobs .

$$RCT_{m,n} = \begin{pmatrix} rct_{1,1} & rct_{1,2} & ... & rct_{1,n} \\ rct_{2,1} & rct_{2,2} & ... & rct_{2,n} \\ ... & ... & ... & ... \\ rct_{m,1} & rct_{m,2} & ... & rct_{m,n} \end{pmatrix}$$

This matrix should satisfy the following relationship:

$$RCT_{m,n} = TMV^{v}_{n,m} + CT_{j,m} \tag{3.25}$$

3.5.3.3 Overlapping

$$RCT_{m,n} = TMV^{v}_{n,m} + LJT_{j,m} \tag{3.26}$$

3.6 Optimisation solution

Find a feasible computing node C_m that has the minimal total completion time for all jobs j {1, ..., k} in the workflow in every matrix RCT and TCT and chooses the optimal Computing node even if it's needed to replicate files. Then the broker allocates the Virtual Machine V_n on it as long as equation below is fulfilled.

$$Load(V_n) \leq Cap(C_m) \tag{3.27}$$

The above equation is to ensures that The V_n load on the C_m cannot exceed its CPU, Bandwidth and Memory capacity.

3.7 Chapter Summary

In this chapter, there has been an outline of the mathematical and theoretical explanation of the scheduling algorithm in NADI as well as bulk job scheduling . It has been showed with the aid of mathematical equations, a matrix of various scheduling costs can substantially enhance the process of scheduling when every task is submitted and executed after considering particular related costs. The key elements for enhancing scheduling and execution are data transfer time, results transfer time, server load and queue time, executable movement time and processing time. The above elements were included in the NADI scheduling algorithm. The major costs which required to be calculated were acknowledged as network, compute cost, and transfer cost and they were expressed in terms of mathematical notations.

4

Virtual Machine Placement using Data Transfer time

In the previous chapter, the general NADI framework for VM placement was presented, describing the components of each layer and the relationships between them as well as the actors and functionalities. A key component of the described framework was the allocation of VMs and their related data. The chapter also provided an overview and analysis of NADI scheduling requirements. All major requirements were classified as either computation-, data- or network-related requirements, so the scheduling problem was divided into different scheduling costs, which included the network, data transfer and computing costs; these were derived through suitable mathematical formulae and were explained. The chapter also explained how these costs are vital when dealing with

a large frequency of data-intensive jobs and when optimising scheduling decisions.

This chapter explains in detail the components related to the first revised version, NADIv1. Amongst these components, the data location is central to the working of an optimised cloud; hence, this location should be part of all scheduling decisions. It also establishes that the best or most optimal network path to computing and storage nodes should be identified and that the scheduling system should calculate and incorporate network measurements while planning job submission. The proposed system will implement the NADIv1 scheduling algorithms and will also help us test the system through carefully created simulations.

4.1 Introduction

Cloud computing entails the use of computing resources (both hardware and software) delivered to customers as a service over a network (typically the Internet). It allows users to share large-scale equipment and resources for computation, storage, information and knowledge for scientific research [131]. Most popular cloud applications are data-intensive, so different types of applications exist, designed for different purposes. Hadoop [132] is a platform for both storage and processing that processes a large number of data sets on computer nodes. The Hadoop service deals with data throughout the storing, accessing, processing, governing, securing and operating stages. Hadoop consists of two main models: Hadoop Distributed File System (HDFS) and Hadoop MapReduce. HDFS is the file system for stored data in distributed environment, whereas MapReduce is aprogramming model that sorts, processes and generates data sets by using a large number of computer nodes. For instance, applications which use the MapReduce frameworks process many petabytes in a single day [133]. Data nodes are used to store large data sets in existing cloud environments; the two main devices for this are probably SAN or NAS. Cloud applications process these data sets using a large number of virtual machines (VMs).

These applications deal with many jobs which must be executed using the available resources to achieve the best performance, the minimal total time for completion, the shortest response time, the efficient usage of resources etc. The most important objectives are to minimise the total completion time and produce the shortest response time. These objectives are affected by factors including the load on the individual servers, bottlenecks in communication data access and the scheduling mechanism [134]. One of the major factors affecting completion time is access latency when transferring data from computer nodes to data nodes. The best solution to this problem is to continue local processing of all data to minimise access latency; however, this is not possible due to the size of data sets. Constraints in computing node capacity also prevent VMs from being placed in their ideal locations [134, 135].

Hadoop used a system partition in combination with data computations using many servers in addition to dealing with the storage nodes in the cloud system. This resulted in positive impacts on the completion process, as it sped up the completion time of the application. While spreading data over a large number of nodes, placing an emphasis on the importance of completion time implies that the only crucial problem to solve is the placement of computation data in such a manner that time limitation goals are realised. But placing a VM without also considering network status may lead to large data access latencies, which in fact may increase the completion times [18]. For example, placing the computation VMs and corresponding data nodes on different racks and enabling the typical oversubscription of the aggregation layer to link to the datacentre contributes to bottlenecked networks.

This research addresses the above issues and proposes a VM placement algorithm called NADI which considers both the network I/O requirement between data node and computing node to minimise the response time between them and the process utilisation which minimises the total execution time for all jobs while making scheduling decisions across multiple datacentres.

4.2 NADIv1 Proposed Design for VMs Placement Scheduling

The algorithm proposed has take into account not only computation resource requirements but also data requirements (storage space, network condition etc.) when making scheduling decisions. The overall architecture of the NADIv1 scheduler is shown in Figure 3.2; this is the first version of the algorithm. At this stage, we only consider local or remote access to data. Later, in chapter 5, data movement is introduced as another solution to discover whether it helps improve application performance. The NADI scheduler includes a matchmaking layer responsible for selecting the best resources for job execution. The three main phases are as follows.

- Phase one is that of resource discovery, which generates a list of potential resources.
- Phase two involves gathering information about those resources and choosing the best set to match the application requirements (the so-called matchmaking phase). This is where this thesis makes its main contribution.
- Phase three places the VM on the chosen resource and executes the VM job execution phase, including file staging in and out and clean-up.

The general NADIv1 scheduler architecture is illustrated in Figure 3.2 in the previous chapter. It comprises a matchmaking layer accountable for choosing the appropriate resources to execute the VMs jobs. The matchmaker is one of the most important components in the NADIv1 scheduler. Basically, a cloud customer first specifies the needed requirements to run the application. These requirements can relate to the VM type, the workload pattern, the job details and the related data location. The matchmaker parses these requirements with other information, such as information about available resources and network status, to make the optimal decision about the appropriate CNs. The next important task is the second component, the allocation scheduler, whose role is to map these VMs onto the physical resources which run the applications. The third significant component in NADIv1 is the Job Submission and Control Services (JSCS), which submits the jobs to the chosen CNs, obtains updating reports during the job execution and keeps these parameters within the database for the users later access, in order to enhance the application performance. The NADI scheduler components have been discussed in detail in chapter 3, section 3.3; however, the contributions of some components change in their contributions to the thesis, as follows.

4.2.1 Network Management Service

Network Management Service Network Management Service (NMS) within a cloud environment is imperative in establishing the basis of performance difficulties or adjusting the system for enhanced performance. Network performance data is therefore vital to cloud scheduling, particularly within a data-intensive scheduling situation. Precise

state-of-the-art network information leads the scheduler to considerable optimisation of the process of job execution. The following parameters are related to network monitoring information (NMI) [136], which were described in the previous chapter but which are, at the same time, significant to the data-intensive scheduling algorithm decision.

- Delay: the time spent for data to go to the destination resource and back. This is known as the round-trip time.
- Network bandwidth: the data volume likely to be sent between two locations in distributed DCs per second.
- The route between CNs and SNs which is assumed by the data across the domains of the network.
- Flow monitoring: the capability to trace the flows of data traffic emanating from a particular resource. Consequences of both the traffic flow and the route taken are jitter and delay sources within a network. For instance, flow monitoring can assist the verification of peering. Peering requires the updating and exchange of router data between the peered networks or ISPs, characteristically employing the border gateway protocol (BGP) which does not just decrease delays but can also considerably impact the transfer time of the data.

4.2.2 Data Location Service

The data-intensive scheduling stage considers big data volumes and data transmission. To optimise the scheduling procedure wherein large data amounts are engaged, the data location, size and other associated factors should be considered. To deal with this matter, a Data Location Service (DLS) is formed which returns the information for a specified logical dataset just within this phase. (In chapter 5, this is expanded it and the replicas and replication service is considered in detail.) The DLS offers the way to locate data within the circulated computing system. The DLS is indexed through the datasets Logical File Name (LFN), and it directs datasets to SNs wherein they are positioned. Every LFN is connected to a unique identifier (UID) which is exceptional across the

whole cloud. Against every LFN, the DLS finds the UIDs which consecutively look up UID to PFN mappings for specified UIDs. The DLS offers the name of the location of DCs which are hosting the data as well as the physical locations in the SNs of the constituent files or the datasets at these locations . This information is fed into the DLS for the specific locations where the data is hosted, either through the scheduling system following job execution or by the data transmission system following transfer. The local manager at any DC operations might lead to adjustments to the DLS, for example in a situation of data loss or removal. To steer jobs to the information, the DLS is contacted to choose the locations hosting the necessary data. This transforms into an overt necessity for the cloud scheduler to allocate jobs at any possible set of locations.

4.2.3 Information Service

The primary stage of the scheduling procedure addresses discovery of the available resources distributed over the DCs. To provide dynamic resources and offer user expediency, a cloud resources scheduler should be capable of discovering and choosing the best and most accessible resources available to satisfy cloud users. This service is addressed in the design here by an Information Service (IS), as demonstrated in figure 4.1 The subsequent stage in the scheduling procedure is to select the resources, which is the broker's role, given a collection of possible resources that can meet the cloud users minimum requirements. These requirements include the VM, its jobs and the available CN resources (RAM, CPU, disk and bandwidth) in scheduling the VM with its related job. Generally, this scheduling occurs in two stages: information collecting, which is the role of the IS, and choice making, which is administered by the broker in the scheduling service .

4.2.4 Discovery Service

The Discovery Service (DS) is considered an entry point for the cloud scheduler and thus is fundamental to cloud scheduling choices. The DS facilitates the cloud customs

to discover the resources and services for VM scheduling and job execution. This is the initial stage within the lifecycle of the job execution and thus is an admission point for the cloud. A competent DS essentially raises the performance, reliability and choice-making capability of the scheduling system, specifically if the intricacy is extensive, as is the situation with the cloud environment. The DS permits the service data published from any location to be retrieved from any site within the DC network. This functionality renders it appropriate for application as a worldwide information index for decisions on scheduling. The DS provides the network and resource data for both static and dynamic scheduling. Once a cloud application needs to submit its jobs, the scheduling procedure is based on some parameters. These may entail the local DC policy, specific software and hardware needs or other considerations [137]. The DS is applied to find cloud resources and network limitations to enable the matchmaking procedure while the jobs need to be scheduled. Note that a push method to timetable the jobs is pursued, and jobs are scheduled as soon they are submitted to a scheduler. There is no specific interval or regularity with respect to the process of matchmaking and scheduling.

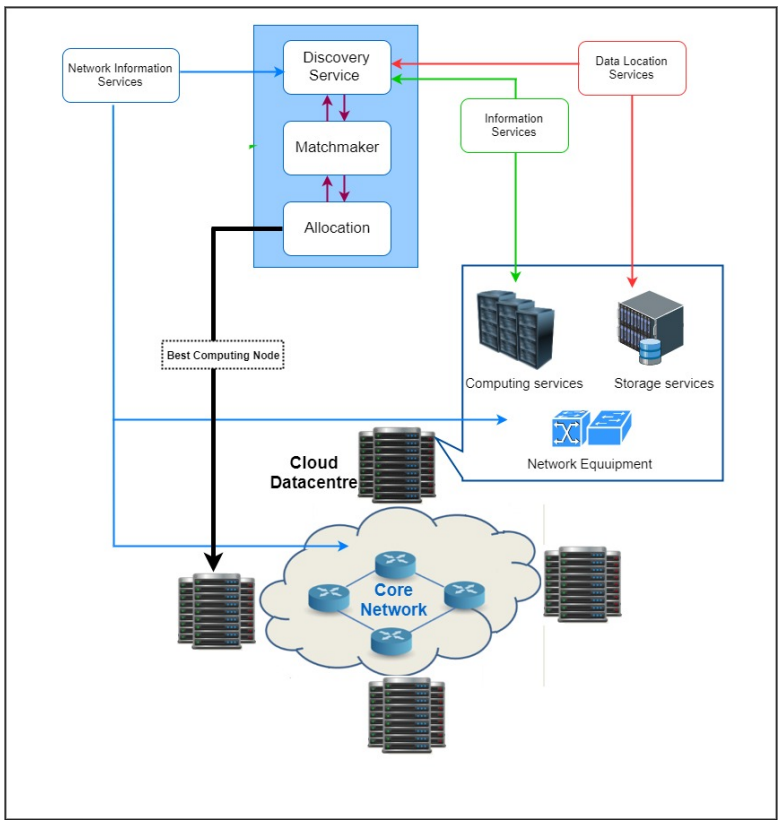


Figure 4.1: Generic VM allocation architecture with the NADI scheduler and the data

4.2.5 Workload Management System

Workload Management System (WMS) is a procedure which establishes the appropriate distributions of the workload to offer optimal performance for cloud users applications. It offers an organisation with the capability to micromanage or control; every work application is run to capitalise on workload throughput and to improve application performance by ensuring that no single CN is overburdened while others are underutilised. The workload management system (WMS) consists of the following three components:

4.2.5.1 Workload manager

The workload manager (WM) is the primary component of the WMS which offers a valid request. The WM takes the appropriate actions to satisfy this request and is responsible for carrying out the real job management operations (job submission or removal). The WM also fulfils user needs by managing workload accessibility to DC resources such as

the CPU, RAM disk I/O and memory. Several workloads need additional I/O operations and less CPU power, while others need the reverse, founded entirely on the workload type. The WM also ensures that workloads are distributed in the right way among available CNs.

4.2.5.2 Matchmaker

Matchmaker is one of the components which supports the WM and scheduler service. It collects the network status information provided by the NMI, collects data related to the application requested by the DLS and collects DC resource information from the IS to make optimal scheduling decisions . (The optimal scheduling decisions based on the total cost matrix have been discussed in chapter 3, section 3.5 with respect to choosing suitable resources for the VMs job execution.) Thus, the matchmaker is considered the most important task in the scheduling service. The NADI scheduler uses the matchmaking engine, IS, NMI and DLS to make the best choice of site for the execution. The data-related information, such as size and location, is provided by the DLS. Then the NADI scheduler allocates the VMs to the chosen CNs and submits the jobs to them.

4.2.5.3 Job submission and control services

The Job submission and control services (JSCS) is employed to submit and execute the jobs to the chosen CN. The goal is to enable the direct interfacing of appliances to the NADI scheduler and enable the incorporation of appliance programs. Job monitoring, submission, control and retrieval of the completed job status are managed by JSCS. It is important to consider the data placement activities and computational jobs in a distributed computing system, because they have the same importance, and data-intensive jobs need to be queued, scheduled, monitored and managed [138]. So JSCS uses its capabilities to compute the related jobs, and it recommends the best choice for dealing with the data placement activities during the scheduling of data-intensive jobs.

4.2.6 Cloud Scheduler Service

The Cloud Scheduler Service (CSS) is defined as the whole management procedure which deploys every VM and its application on cloud infrastructures. This management includes VM creation, allocation, migration and deallocation. It comprises two key stages:

4.2.6.1 VM provisioning

This entails the instantiation of one or more VMs which correspond to the particular hardware features and software requirements for the applications deployed on the VMs. The majority of cloud suppliers provide a collection of general-purpose VM classes having resource configurations and generic software in advance. For instance, Amazon EC2 supports 11 different types of VMs, which have different RAM, CPU and I/O disk options depending on the cloud users demands.

4.2.6.2 VM allocation

This is the process of scheduling and mapping the VMs onto physical CNs in the cloud environment. Presently, the majority of IaaS suppliers do not offer the providers of applications any control over resource provisioning. VMs are mapped to physical CNs, so the VM mapping is entirely concealed from the application providers [139]. This is where this thesis makes its main contribution.

4.3 Algorithm

The main steps of the proposed algorithm are as follows. It first checks the CPU, bandwidth and memory capacity for every available CN, as shown in line 3, below (and as shown in equation 3.12 in the previous chapter) . If this condition is fulfilled with the required VM, then as in line 5, it sends the VM jobs, related data location, chosen CN and storages to the function named Calculates (JT) (job data transfer) time, shown in algorithm 4.2 and using equations 3.9 and 3.10. This function in algorithm 4.2, from

lines 2 to 5, calculates (JT) for all jobs on the first CN, retunes the value to algorithm 4.1 and stores it in the data transfer time (DTT) matrix, as shown in line 7. After that, as shown in line 8, it calculates PT, computing time, as also shown in equations 3.9 and 3.10, by sending the VM jobs and chosen CN to the function in algorithm 3. Line 3 in this function calculates the processing time and when this job will start for every job, and it then returns the value to algorithm 4.1. The value is stored in the (CT) matrix, as shown in line 12. At the end of this processing, we will have the total completion time (TCT) matrix, with all computing time and data access time for the first CN, as shown in equation 3.11.

Algorithm 4.1 works in the following manner. First, it obtains information about the available peers from the discovery service. Then it communicates with each peer and collects the peers queue length and the total cost. (The architectural details and all three costs are discussed in chapter 3, sections 3.2 and 3.5.) Then it determines the CN with the minimum queue length and the total time cost. In principle, to calculate the total time cost on any individual CN, we must generate all job completion time matrices by computing the data transfer and computing time of all CNs with available resources. If the number of jobs and the total time cost of the remote CN is higher than the local cost which contains the data , then the VM is scheduled to the local CN. Otherwise, the VM, jobs and data are moved to the remote CN, subject to a cost-time mechanism.

Now the steps for the proposed VM scheduling are described, as shown in algorithm below .

Algorithm 4.1 VM Placement

Input: V VM List, D Required Data, C Computing node list S Storage node list and J job List

Output: V List of VMs assigned to C list of computing node

```
1: for v i to V do
2:    MinTCT ← MAX
3:    AllocatedHost ← NULL
4:    for c = 1 to C do
5:       if Load(v) ≤ Cap(c) then
6:          for j =1 to J  do
7:             DTT = DTT + JT(j, D, S, c)
8:             CT = CT + PT(j, c)
9:          end for
10:         TCT = DTT + CT
11:         if TCT < MinTCT then
12:            AllocatedHost ← c
13:            MinTCT ← TCT
14:         end if
15:      end if
16:   end for
17:   if AllocatedHost ≠ NULL then
18:      Allocation.add(v,AllocatedHost)
19:      Updating: Cap(AllocatedHost) = Cap(AllocatedHost) - Load(v)
20:   end if
21: end for
22: Return allocation
```

- (Input): the NADI scheduler first receives the users request details (application jobs, related data location and required processing time) and then collects the resource information from discovery services such as the VM request list, CN list and SN list. This information deals with the algorithm as input parameters.
- (Output) All VMs in the VM list will be assigned to CNs and will update the CN capacities (Ram, CPU and bandwidth).
- (Line 1): This is a loop to place VMs one by one from the VM request list.
- (Lines 2-3): These are variables which decide the optimal CN AllocatedHost which has the minimal completion time (MinTCT).
- (Lines 4-5): This is a loop to check whether the CN has sufficient residual capacity Cap(cn) to accommodate the requested VM capacity load(v).

- (Lines 6-9): This is a loop to calculate both the computing and the data transfer time for all VMs jobs one by one. The variables DTT and CT are the stored values, as shown in equations 3.6 and in section 3.21 in chapter 3. As can be seen, there are two functions which calculate the data transfer time and computing time for every job, as described in the following:

  ◇ In line 7, there is a call for the function JT, which leads to algorithm 4.2 . This algorithm checks the existence of the jobs related data in each s in the storage list. So if it exists, it calculates the transfer time for certain data between this storage and the chosen computing node.

Algorithm 4.2 Calculates Data Access Time

Input: j, D , S and c
Output: Calculates data transfer time for j

```
1: for s = 1 to S do
2:   for d = 1 to D do
3:     if s stores d then
4:       JT = d / BW(s, c)
5:     end if
6:   end for
7: end for
```

  ◇ Similarly, line 8 is called a CT function, as shown in algorithm 4.3, and it calculates the computing time for the sent job . It is important to know, for every computing node, the processing information such as the start time for the job, the processor strength and how much time the execution will take.

Algorithm 4.3 Calculates Computing Time

Input: j, c
Output: calculates processing time and waiting time

1: $PT = st(j) + \frac{ni(j)}{cap(c) \times core(j)}$
2: **return** PT

- (Line 10): Now the completion time for all VM job can be calculated, which is the sum of the computing and data transfer time, as shown in equation 3.23. Repeating lines 68 computes the data transfer and computing time for all jobs and stores the total time in the completion time matrix (TCT) for the chosen CN.

- (Lines 11 -16): By completing these lines, the TCT matrix stores all values for all CNs. Then there is a search for the CN with the optimal value in the TCT matrix; when found, the chosen CN will be the host for the VM.
- (Lines 17-19): Now the scheduler ensures that the value of the chosen CN is not empty; otherwise the VM will be allocated to it .
- (Line 20): This line is reached when all VMs in the VM list have been allocated successfully .

4.4 Simulation results

In this section, the investigation aimed at evaluating the proposed adaptive cloud placement algorithm are presented. CloudSim [69] discrete-event cloud simulation was used to model the cloud environment. The proposed NADI algorithm will be compared with Piao's algorithm [82] and CloudSim's default algorithm [37]. Piao's algorithm only considers the data location and chooses the best CN to minimise the data transfer time, while CloudSim's algorithm chooses the first CN with CPU and RAM availability . The system will collect all needed parameters and apply them using all compared algorithms equally. Next, both the simulation set-up and the workloads used for the evaluation are described.

4.4.1 Simulation Set-Up

The simulated model comprises one cloud datacentre containing 10 CNs. Each CN has two quad-core processors and 16 GB of RAM. The CN hosts the user application instances and the mechanism. The datacentre has one broker, which receives users' requests and places the VMs on the appropriate CNs based on the allocation algorithm used. There will be 25 VMs, requiring one core and 2 GB of RAM. A total of 25 files are distributed over the SNs in the datacentres, of 0.1- 3.0 GB sizes (generated randomly). Every VM has certain numbers of jobs, for example 10 jobs, gradually increasing to 50, ..., 250 jobs. Every job needs to access certain data (e.g. file1) stored on an SN (chosen

randomly). Table 4.1 shows all parameters used.

Parameters	**Values**
Number of CNs	10
Number of VMs	25
Number of jobs	50 - 250
Number of files	1
File Size / job	[100 - 3000] MB

Table 4.1: Scenario one Parameters

4.4.2 Simulation Evaluation

After the broker receives all VM requests needing allocation, its first calculates the total job completion time on each computing node, including the data (file) access time between sites that store the files and each server (file size/available bandwidth), the waiting time for each job before it processes and the processing time. Then, taking all these times into account, it chooses the CN with the shortest total time and allocates a VM to it. Then it updates the datacentre information (e.g. availability on the servers).

In this simulation, a one-source data retrieval technique is use. Looking first at Figure 4.2.a , the data transfer time for Piao's algorithm produces better results than the approach, because this algorithm only considers the data transfer time to minimise the total completion time and improve the application performance. Meanwhile, the CloudSim algorithm produces the worst results, due to its ignoring of the network state between the nodes.

Similarly, in Figure 4.2.b, Piaos algorithm has better results than CloudSim in total computing time, even though it does not consider CPU loads and less queuing. The CloudSim allocation policy aims to allocate many VMs on one CN to minimise energy consumption and maximise the CNs utilities, which puts a high load onto the CNs resources (CPU, memory, bandwidth and disk). This leads to an increase in the CPU load, resulting in increased total execution time, whereas the most important factor in application performance is minimising the total completion time, as we did. The NADIv1 algorithm shows the optimal results in computing time, because completion time is one of the considerations in its allocation policy.

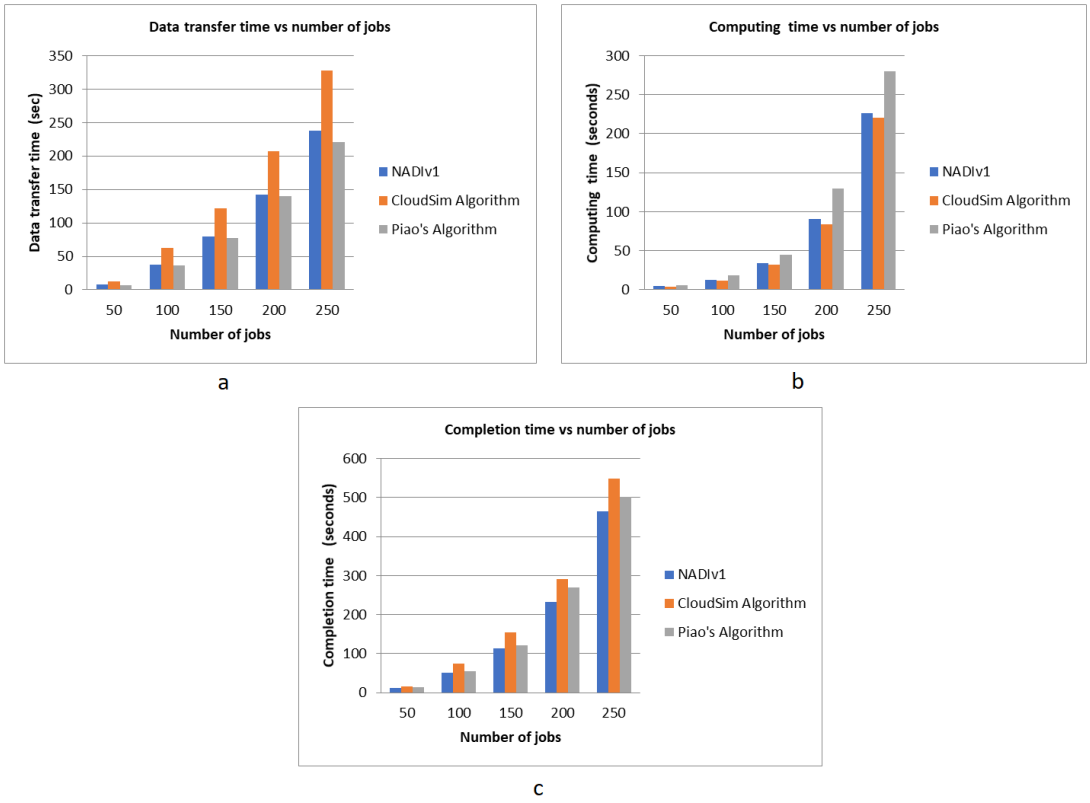


Figure 4.2: Completion time for all jobs

The above test demonstrates that the proposed allocation mechanism is useful in improving the completion time of tasks. Since the NADI algorithm tries to schedule independent tasks to optimise minimum completion time, it produces better estimates for task execution. Consequently, it results in a better completion time rate than either the Piao or CloudSim algorithms.

Clearly, the proposed algorithm has the shortest job completion time in every round, with increased job numbers, as shown in Figure 4.2.c. In fact, the decline of the average task completion time is caused by the optimised location of the hosted VMs, compared with the other two VM allocation policies. The results prove that the proposed network-aware VM allocation policy can lead to improved task completion time. Due to differences in network status and different data distributions, the improvement varies from 200 s to 32 s, as shown in Figure 4.2.c.

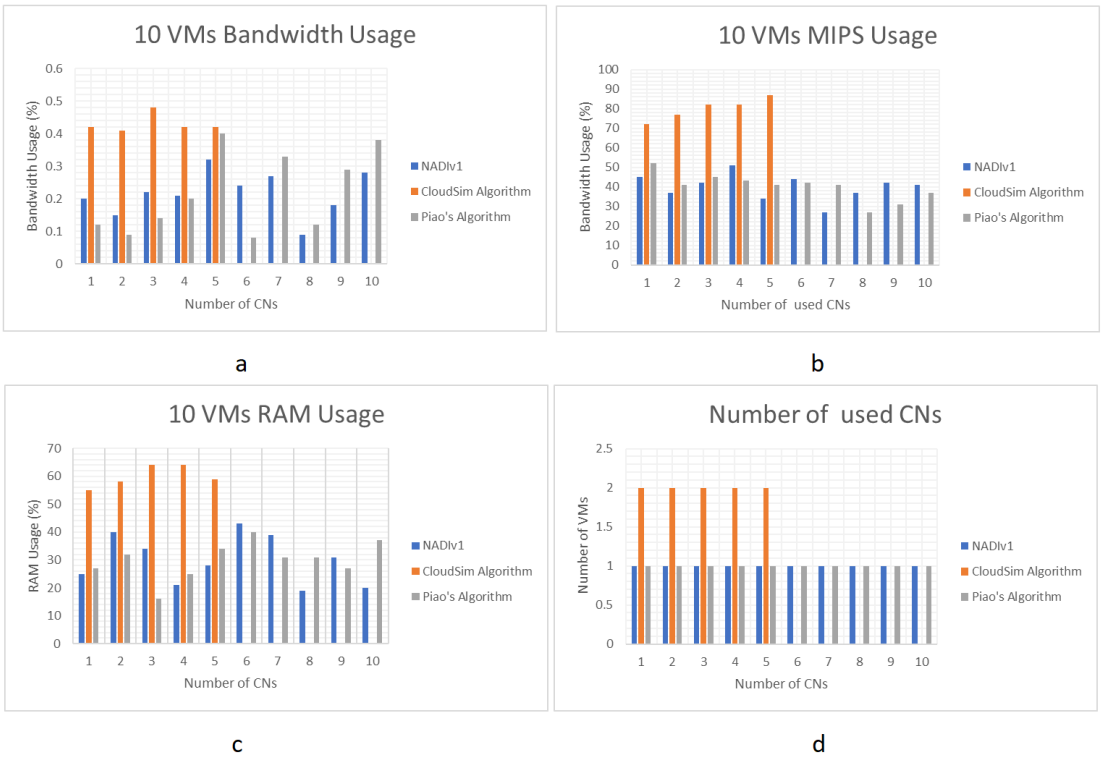


Figure 4.3: Resource usage for 10 CNs and 10 VMs

Figure 4.3 shows the usage rate for bandwidth, CPU, RAM and CNs, utilising graphs for 10 CNs and 10 VMs. First, the bandwidth usage in the Piao approach is less than NADI's, as the Piao algorithm only focuses on this one parameter to enhance application performance. However, the NADI algorithm considers the available bandwidth and CPU usage to achieve the best results for the application jobs, and this algorithm has the advantage in CPU usage, as shown in Figures 4.3.a and 4.3.b.

The CloudSim approach produces the worst results in both CPU and bandwidth, because of the high load on every CN, which results from the total number of VMs on each CN. Secondly, RAM usage is high in the CloudSim algorithm, because half of the CNs are in sleep mode, whereas the NADI algorithm and the Piao algorithm reciprocate the leverage consecutively, depending on the workload , as shown in Figure 4.3.c.

Figure 4.3.d demonstrates the number of CNs utilised by the scheduler to place the VMs. Here the new proposed algorithm places the VM on the CN with the minimal completion time, while Piao's algorithm only considers network bandwidth. The number

of CNs used by the proposed algorithm is maximised, when compared to the CloudSim algorithm, to achieve high performance. Although the number of CNs used by CloudSim is smaller than for NADI, the latter proposed algorithm is comparatively stable, as shown in Figure 4.3. The resource utilisation rate of the new algorithm appreciably outperforms that of the other algorithms.

Figure 4.4 shows that the new placement algorithm minimises resource usage, thus enhancing application performance compared to the Piao and CloudSim placement algorithms. The results also demonstrate that the CloudSim algorithm uses fewer CNs than does the NADI approach or the Piao algorithm. CloudSim reduces energy consumption, whereas in this thesis, the algorithm does not consider this factor, only focusing on how to improve the application performance.

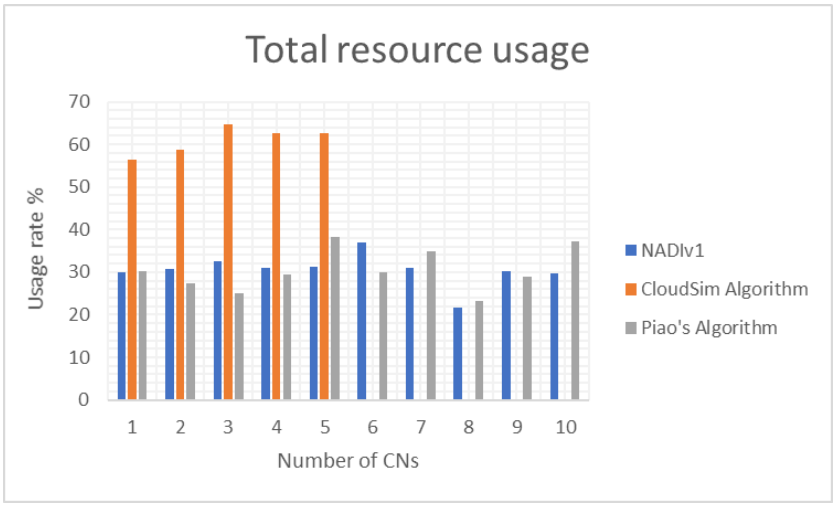


Figure 4.4: Comparison of the overall resource utilisation rates

In this second scenario, the total number of VMs remains at 25 , but the number of files requested by each VM is increased to 3, as in Table 4.2 below. Other parameters remain the same as in the first scenario, above. One of the advantages of the NADI algorithm is that it can consider more than one file, stored in different SNs. From Figure 4.5.c , it is clear that the technique used here significantly decreased the queue time of the jobs. This was because only those CNs were selected for VMs and job placement which had fewest jobs in the queue and which were most likely to quickly execute the jobs once they were scheduled on those CNs. The graph in Figure 4.5.c is based on the

average completion times for a varying number of jobs, as mentioned in the previous scenario.

Parameters	**Values**
Number of CNs	10
Number of VMs	25
Number of jobs	50 -250
Number of files	1
File size / job	[100 - 3000] MB

Table 4.2: Scenario Two Parameters

An increase in network cost can also affect the overall performance of the distributed system, especially in terms of transfer and communication time; therefore, it is an important consideration for any scheduling decision. A lower bandwidth also results in higher network costs. Figure 4.5.a demonstrates that in the new algorithm, through an increase in network bandwidth capacity, the data transfer time decreases (compared with the approaches of the other algorithms), and hence, results in a better completion time. The NADI algorithm produces better results in data transfer time than do the other two algorithms, because it considers the distributed files over the SNs; the Piao algorithm only considers the bandwidth between CN and SN. The main difference is the total number of files considered, where Piao only considers one file, which leads to degradation of the overall performance.

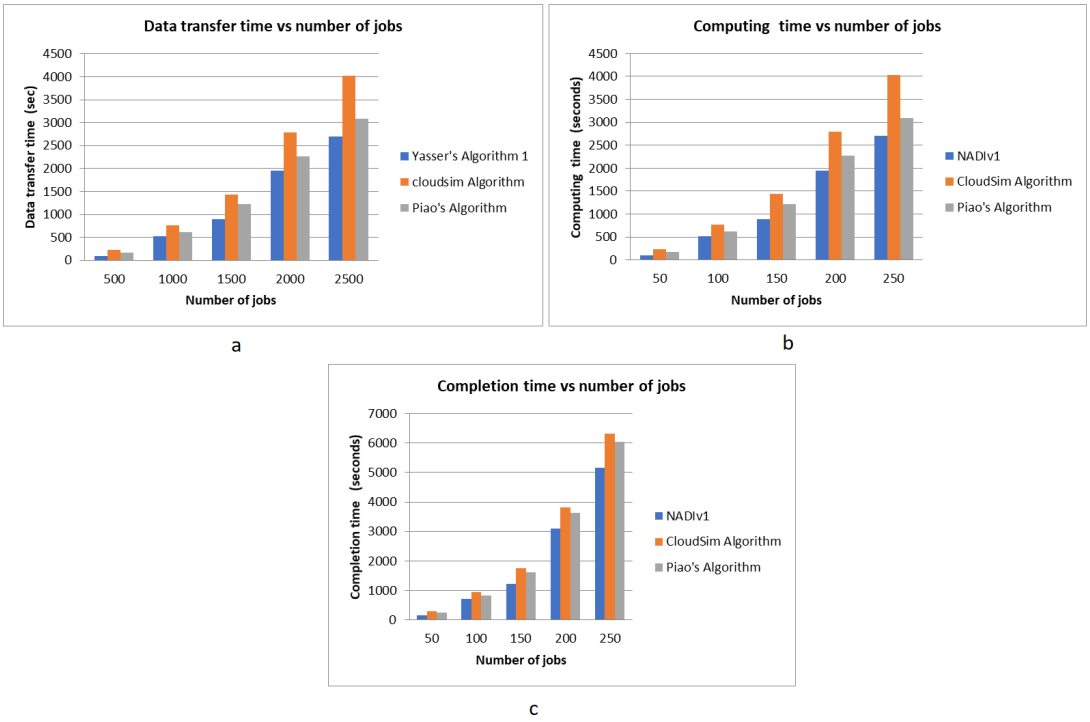


Figure 4.5: Completion time for all jobs

The execution time is normally longer than the processor time for the process, because the CPU is doing other things besides just running the process, including running other user jobs and operating system processes or waiting for disk or network I/O. Nor does execution time include queue time. When the number of jobs increases, it is evident from Figure 4.5.b that the average time to execute a job is also increased. More competing jobs clearly result in more time required for a specific job to complete. Figure 4.5.b shows that the new algorithm scheduling approach improves the execution times of jobs.

Figure 4.6 demonstrates the resource usage for 10 CNs and 25 VMs. In Figure 4.6.a, bandwidth eventually deteriorates in Piao's approach, due to the cost of file transfer time between the application and the SNs . The NADI algorithm produces better results, because the scheduler considers the total file transfer time for each CN. Again, the CPU usage in Figure 4.6.b shows that the new algorithm retains the advantage over the other two approaches, due to its consideration of how busy the CPU is.

The RAM usage graph in Figure 4.6.c shows that the CloudSim algorithm uses con-

siderable RAM in five CNs, whereas the other CNs are in sleep mode. This will exhaust the RAM and consequently will significantly affect the overall application performance of these CNs. At this stage, the NADI algorithm surpasses Piao's algorithm, because it checks the significance of the CPU load, which correlates with minimised RAM usage, as shown in Figure 4.6.c.

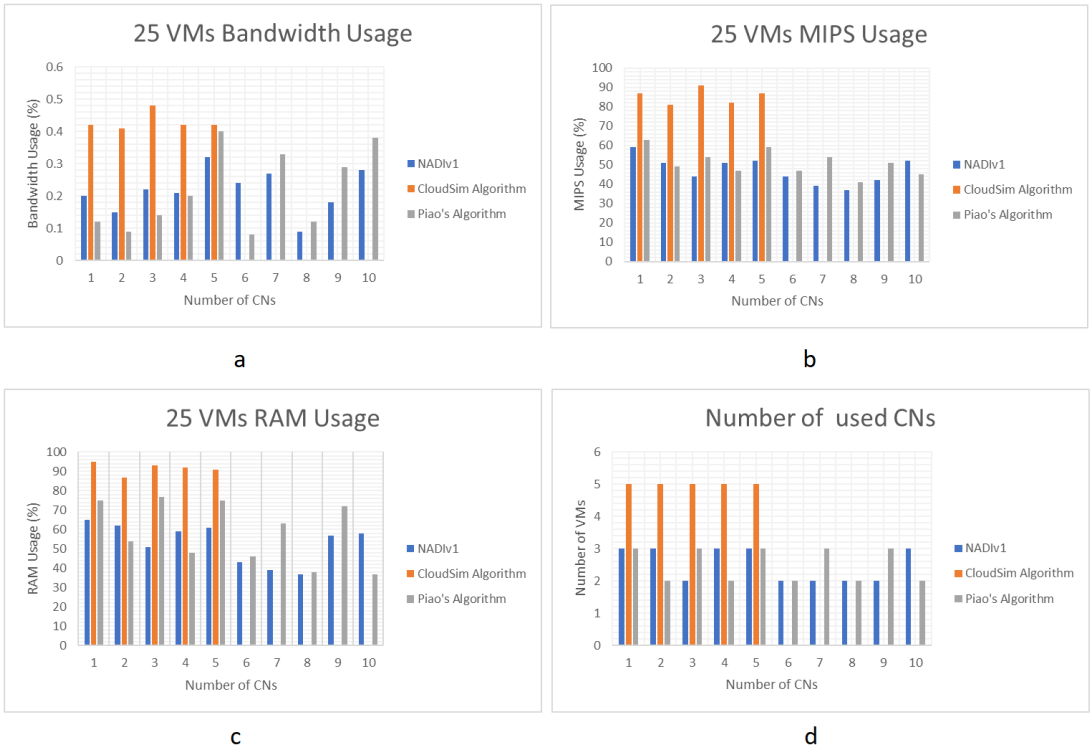


Figure 4.6: Comparison of the overall resource utilisation rate

Figure 4.6.d demonstrates the number of CNs utilised by the scheduler to place the VMs. Here the proposed algorithm places the VM on the CN with the minimal completion time, whereas the Piao algorithm only considers network bandwidth, and the CloudSim algorithm tries to maximise the CNs utilities. The proposed algorithm produces the optimal values and appreciably outperforms the other algorithms with respect to the resource utilisation rates.

Figure 4.7 shows that the new placement algorithm minimises resource usage, thus enhancing the application performance, compared to the Piao and CloudSim placement algorithms. The results also demonstrate that CloudSim uses fewer CNs than either

the NADI approach or that of the Piao algorithm; this reduces energy consumption. However, in this thesis, the new algorithm does not consider energy consumption; this chapter only focuses on how to improve the application performance.

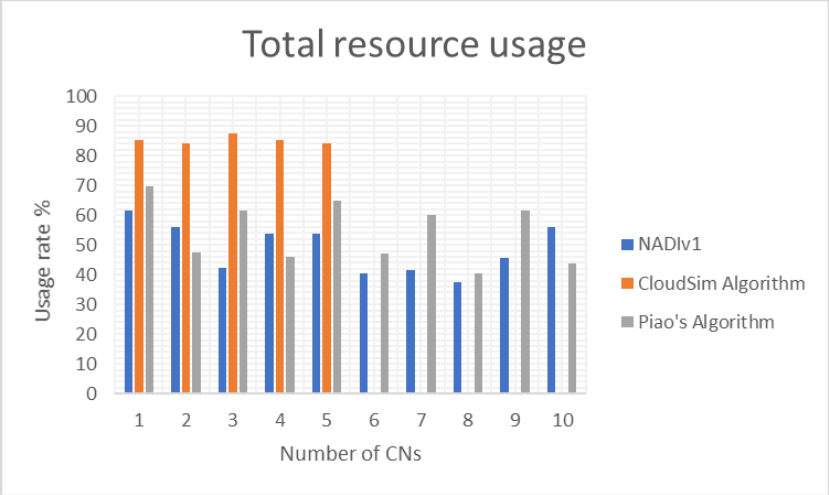


Figure 4.7: Comparison of the overall resource utilisation rates

In the following figures, some of the parameters have been changed, as shown in Table 4.3 as has been discussed in the third scenario, section 3.2.3. In this third scenario, large-scale evaluation to make it more complicated is used. The main aim of this scenario to simulate the network traffic in the real DCs. Thus, the total number of VMs was increased , and each VM requested a number of files. There were three DCs, and every DC involved 350 CNs. The simulation ran for 10 minutes, and in every minute, there were requests for VM placement. These VMs were gradually increased each minute, from 150 to 1500. Each VM requested 35 files, spread over the distributed SNs. The jobs also increased gradually from 500 to 5000 in each round. The same comparison points were also used: Piao's algorithm [82] and CloudSim's default algorithm [37].

Parameters	**Values**
Number of DCs	3
Number of Users	500-5000
Number of CNs	1000
Number of jobs	50 - 250
Number of files	3- 5
File size / job	[1 - 30] GB

Table 4.3: Scenario Three Parameters

The sub-figures in Figure 4.8 show the data transfer, computing and total completion times . Both the data transfer and computing times follow similar trends, primarily because NADI preferentially selected those CNs for job execution which could quickly execute the jobs (i.e. those that had short local queues with low latency). The trends in Figures 4.8.a and 4.8.b show that data transfer time is almost proportional to computation time, because if a job is running and taking more time on the processor, the waiting time for the new job will also increase. Thus, the related data for that job will arrive late, which increases the total completion time.

Also, whenever the burden on CNs is high, the CN resources (CPU and bandwidth) will be affected. As mentioned earlier, an increased network cost can also affect the overall performance of the distributed system, especially in terms of transfer and communication time, and a lower bandwidth also results in higher network cost. Therefore, network cost, as always, is an important consideration for any scheduling decisions. Figure 4.8.b demonstrates that whenever network capacity increases, it leads to reduced transfer times and, hence, better job execution times. Increasing the number of jobs in the queue can also influence the overall job completion times (i.e. the scheduling, queuing, execution and data transfer times). It can be ascertained from the graphs that as the number of jobs increases, NADI has a more profound impact on the scheduling optimisation and execution of the jobs. Since most of these jobs need to communicate data from remote locations, it is assumed that execution and transfer times will further decrease when thousands of jobs take actual input data from optimal locations, as demonstrated in Figure 4.8.

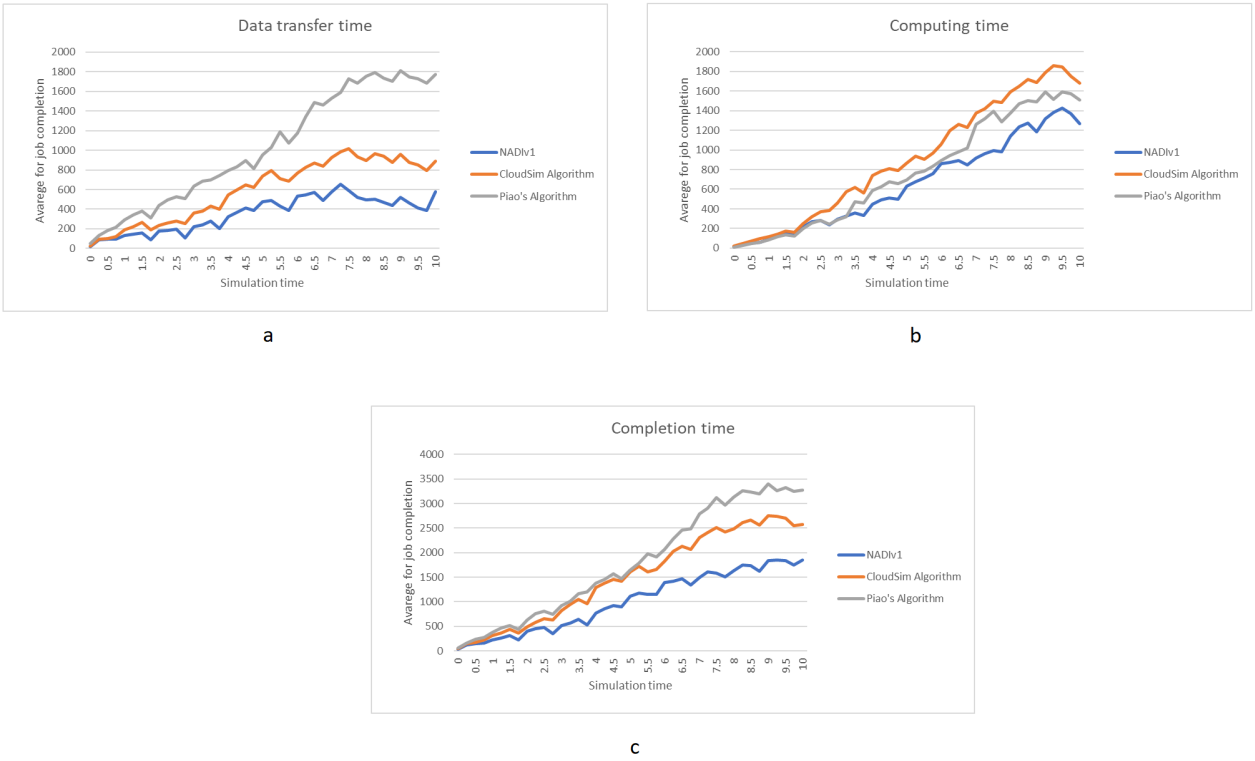


Figure 4.8: Completion time for all jobs

Only one job is executed on a CPU at a time, and jobs cannot run in parallel on that CPU, since a non-pre-emptive scheduling model is followed. When the number of jobs increases, it is evident from Figure 4.8.b that the average time to execute a job also increases. More competing jobs clearly result in more time required for a specific job to complete. The data transfer delay can be significant and can substantially increase the time required by the VM to complete the job execution.

Figure 4.8.c clearly shows that the NADI scheduling approach has improved the completion time of the jobs, an improvement ranging between 40% and 65%. This completion time will increase dramatically whenever the DC load, data size and number of jobs increases. NADI improves the execution times of jobs, since it selects only those CNs for job execution which have the required data, have less of a load and have fewer jobs in the queue. All of this contributes to fulfilling the execution optimisation, as demonstrated in equation 3.23 in chapter 3, section 3.5.3. Otherwise, CNs which have a higher number of jobs already running or CNs which are heavily loaded can worsen

execution times.

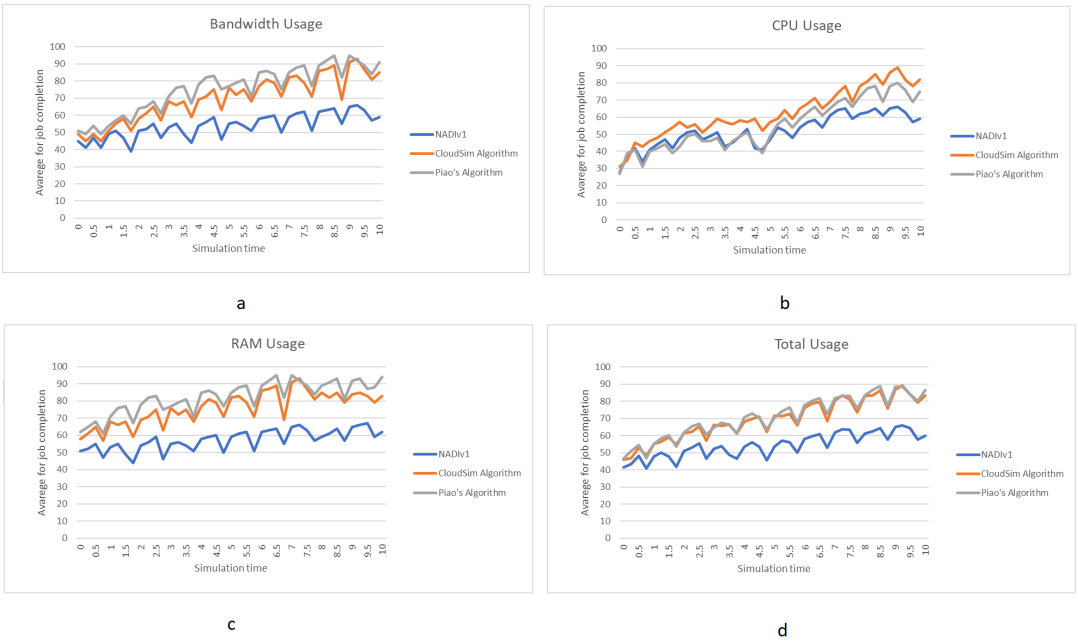
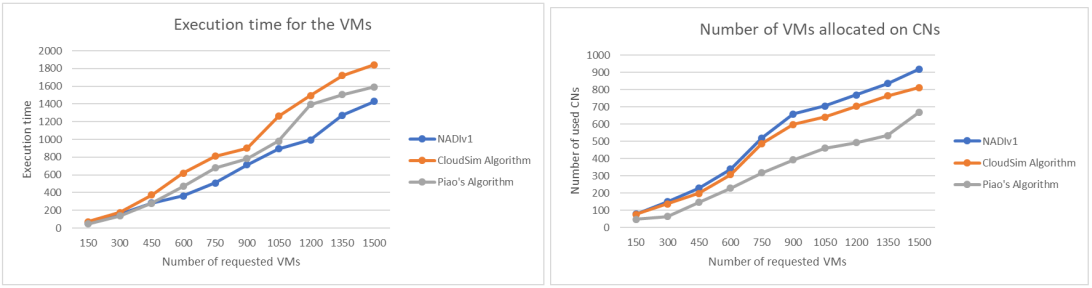


Figure 4.9: Comparison of the overall resource utilisation rate

The number of CNs allocated in the new algorithm are be compared with those of the other algorithms [37, 82], comparing bandwidth, CPU and total resource usages. Figure 4.9.a shows that the bandwidth usage for the NADI algorithm is better than that of the others, because it considers the available bandwidth in each link in the DCs network and chooses the optimal one. Piao's algorithm considers the best link between the CN and SN which stores the first file. It may be observed that the usage percentage gets better continuously whenever the workloads and VM requests increase.

Figure 4.9.b illustrates CPU usage. The CloudSim algorithm is the worst of the widely available resources in this respect. However, the resource availability can be significantly increased if the VM placement is decided wisely. Figure 4.9.c shows that considering the availability options for the bandwidth and CPU enhances RAM usage from 10% to 25%, thus resulting in the application's improved performance. NADI can decrease CN usage and minimise the total execution time, as shown in Figures 4.8 and 4.9, meaning that the overall application performance is enhanced.



a b

Figure 4.10: Number of VMs Allocated on CNs

Figure 4.10.a shows that the execution time for jobs can be reduced to as much as 47% of the total completion time if the NADI algorithm is used. Since these VMs are hosting data-intensive applications, data will be transferred between VMs while execution is in progress. Data transfer between VMs takes time, depending on the distance between these VMs and the amount of data transferred. The number of CNs allocated in the new algorithm with those of previous approaches has been compared. Figure 4.10.b compares the number of CNs selected between these algorithms; NADI decreases CN usage and minimise the total execution time, as shown in Figures 4.8, 4.9 and 4.10.a. Thus, the overall application performance is enhanced.

4.5 Chapter summary

While the data-intensive applications interact with distributed data in a cloud computing environment, the network status between the application and the data could significantly influence the performance of the application. This chapter proposed the first version of a network-aware VM placement algorithm for data-intensive applications in a cloud environment (NADI). Then it showed the relationships between NADI design components, including the broker, data location service, network monitoring service, information service and discovery service. The algorithms used in this contribution were also presented and explored in greater detail, and simulation tests were performed to examine the NADI algorithm. Through these tests, it was demonstrated that data location, network latency and computational resources can significantly affect application

performance. Thus, the VM allocation scheduler should be capable of optimising the data-intensive scheduling process. It is necessary to control the location of VMs, so the applications hosted by a VM can obtain a shorter data access time. It was also demonstrated that the keys to cloud optimisation include network-managed services and a suitable selection of the network links between DC locations before making scheduling decisions. Also, overall queue and execution times can be significantly reduced if job data requirements and completion times are taken into account. It was also concluded from the simulation results on CloudSim 3.0 that NADI is better in scalability and consistency when compared to other contemporary scheduling approaches. In the next chapter, the possibility of enabling the replication service to improve application performance will be discussed. This service allows one to choose one of the best replicas distributed over the DCs in order to enhance application efficiency, maximise the availability of the bandwidth and optimise data-intensive scheduling.

5

Virtual Machine Placement using Data Replication

The first version of NADI (NADIv1), was introduced in Chapter 4 . Some NADI components are changeable depending on the version of the approach, therefore, they were presented in detail. NADI's main objective is to choose the optimal locations for the VMs , which can minimise the computing and response times between the storage and computing nodes, thus enhancing the application performance and minimising the total job completion time. Therefore, data are accessed by fetching their either remotely or locally, depending on the locations of the VMs and the data. The summary of relevant works, as well as the current design and algorithms were explained in detail. Moreover, the results obtained in the simulation experiment indicated considerable improvement

of the application performance and minimisation of total job completion time.

This chapter presents the second version of the NADI approach, NADIv2, which enables the replication service. In this version, the Data Management Service (DMS) is the only component that has been changed from the previous version. The DMS consists of three major components: the Replica Location Service (RLS), the Replicator and the Replica Catalogue. The RLS is launched to obtain the most effective physical replica of the dataset and provide access to information. The Replicator is used to replicate the required data to the chosen location. The Replica Catalogue stores information in a database that maps between the different file names and their replicas . Generally, NADI chooses the best replica among those available or replicates the required data to a destination that does not have any in order to minimise the total completion time. The ideal situation, on one hand, is to access the data locally, however, this is not always possible due to the sizes of data sets and CN capacity constraints, which might prevent VMs from being allocated to them. On the other hand, in cases where the very spot of execution of a job obtains no data replication, then data are transferred from a distant or remote site, which deteriorates the overall execution of the job. The decisive factors in the performance of these applications are workload and workload types, network status between SNs and CNs, data location and CN CPU attributes. Thus, the completion time differs according to the application jobs in the workload based on the retrieval of vast amounts of data and VM placement decisions. NADI optimises the available computing resources and replica locations and chooses to replicate the data in order to allocate the VMs that improve application performance. When compared to the NADIv1 proposal for VM placement, the NADIv2 results reinforce the algorithms with the ability to significantly maximise the throughput of cloud links, increase performance and minimise the completion time of average jobs.

5.1 Introduction

Cloud computing can be defined as the implementation of computing resources (such as software and hardware) which the customer receives in the form of a service via a network (commonly the internet). It aims to disseminate large-scale components and resources that are required for storage, knowledge , computation and information in scientific research [131]. Over time, cloud applications have come to depend on networks in the areas of interactivity or data access and their requirement demands are continually increasing. Computation is requested by particular tasks cited as jobs and determined by computation, network capacity and storage. Cloud applications use multiple VMs in the processing of large volumes of data . Therefore, such applications manage a number of jobs that are carried out by available resources so that the best outcomes, briefest response times, shortest completion periods and the most effective application of resources can be obtained [133]. Compressing the time frame for response and completion is the key objective in this study. A number of factors, such as communication data access bottlenecks, the scheduling mechanism and load on a sole server, can influence this objective and others [134]. Access latency in data transfer between data nodes and computer nodes is an important factor impacting completion times. Continuous processing of all data locally is the most effective way to diminish access latency; however, data set sizes and computing node capacity barriers that restrict the proper placement of VMs sometimes prevent local data processing [134, 135]. The cloud broker is responsible for selecting an accurate resource followed by its management and job execution. The available resources and submission requests are matched with each other through a process that finally determines the resource to be used. This matchmaking process requires certain compatible scheduling mechanisms having the right heuristics and the ability to consider the features of the network to activate effective scheduling of data intensive jobs in practical computing resources.

Different quantities of data are produced by each data intensive application. The data (both input and output) and the computation should be first aligned and co-

scheduled to diminish the overall cost of data transfer and computation, further contributing to an improved overall job completion time and elevated cloud throughput. In fact, the decision to send both data and the VM to a third location can be made based on the features and capabilities of storage resources, the network and computing requirements . Perceived from the point of local resource management, this might appear to be a minor issue, however, global distributed scheduling should be managed seriously in case the overall cloud throughput requires optimisation at any particular time.

In this chapter, the second version of the NADI approach for VM scheduling is presented, which considers data and computation as well as network features during the allocation of VMs and scheduling of single or multiple jobs. Thus, this study focuses on the issue mentioned above by offering a VM placement algorithm that accounts for the network's I/O requirement and process utilization for diminishing the response time between the computing node and the data node and for reducing the total completion time for each job, while taking scheduling decisions over several data centres.

5.2 NADIv2 Design for VMs Placement Scheduling

Basically, the proposed algorithm in NADIv2 takes into account not only computation resource requirements but also data requirements (storage space, network conditions, etc.) when making scheduling decisions. The general NADI scheduler architecture is illustrated in Figure 3.2 in Chapter 3. It comprises a matchmaking layer that is accountable for choosing the appropriate resources to execute the VM's jobs. The matchmaker is an important component in the NADI scheduler. First, a cloud customer specifies the requirements necessary to run his application. These requirements can be related to VM type, workload pattern, task details or the related data location. Then the matchmaker parses these requirements with other information, such as available resources and network status, to make the optimal decision about the appropriate CNs. The second important task is performed by the allocation scheduler, whose role is to map these VMs on a physical resource that runs the applications. The third significant component

in NADI is the JSCS, which submits the jobs to the chosen CNs, obtains the updating reports during job execution and maintains these parameters within the database so that users can later access them in order to enhance the application performance. These components were discussed in detail in Section 3.3, however, here, a slight modification to the DLS component makes it a comprehensive Data Management Service. This component is considered the main contribution of the current chapter.

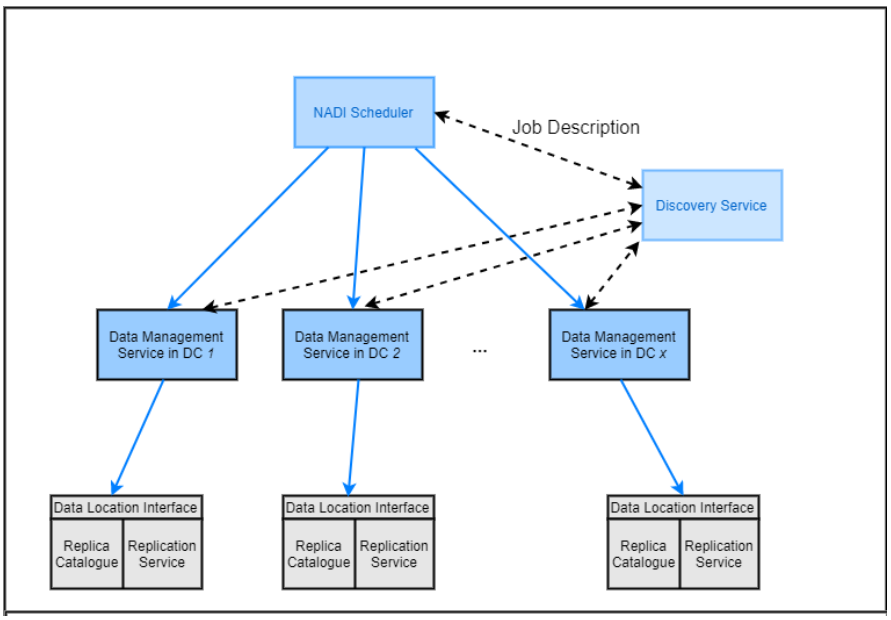


Figure 5.1: Interaction of the NADI Scheduler with the DMS

5.2.1 Data Management Service

As shown below , intensive data scheduling accounts for an enormous amount of data, as well as data replication and data transfer. To optimise the scheduling processes where there is involvement of huge amounts of data, it is necessary to take into account size, data location and other related aspects . The DMS is created to solve this issue as it returns the optimal replication of a specified logical dataset, making use of the Data Management Interface. A DMS offers a means of locating data replicas in a distributed computing system. Indexing of DMS is done by logical filename (LFN) and it atlases the dataset to an SN in their locations. Every LFN is connected to a Unique Identifier (UID), which is exclusive in the entire cloud . The DMS positions the UID against

every LFN, which in response, looks up the UID for mapping the Physical File Name (PFN) for a certain UID. Figure 5.1 illustrates DMS architecture; it shows three sub-DMS service instances that are distributed over DCs in the cloud. When the DS queries about the location of a certain replica, the sub-DMS that stores the replica will return the information about its physical location. If the NADI scheduler, which is now the client, requires some information concerning the dataset which is housed in the Storage Nodes (SNs) as well as registered in the Replica Catalogue, the scheduler will have to inform the DS, which offers a list of every location in which the DMS is operating. The client can now query the DMS by fleeting a rational dataset name whose location is of interest. The outcome of this query is that all the physical replicas will be returned to the scheduler . In the end, it has a reference number of the CN where the task will be executed.

If the NADI scheduler, which is now the client, requires some information concerning the dataset which is housed in the Storage Nodes (SNs) as well as registered in the Replica Catalogue, the scheduler will have to inform the DS , which offers a list of every location in which the DMS is operating. The client can now query the DMS by fleeting a rational dataset name whose location is of interest. The outcome of this query is that all the physical replicas will be returned to the scheduler . In the end, it has a reference number of the CN where the task will be executed.

5.2.1.1 Data Location Interface (DLI)

The Data Location Interface (DLI) is used by the DMS to access replicas of datasets from different catalogues. The DLI does not have direct interaction with the DS. Instead, the DMS operates between the DS and the DLI. Example of this is the content and location of DLI are not distributed to the Discovery Service. Rather, the DMS queries the catalogues via the DLI and then prints the information in the dataset to the DS. In Figure 5.2 the DLI is called the Logical Dataset (LDS) and processes it as an input which later returns the physical locations of datasets. The LDS refers to a data entity or a file collection. To simplify, we assume that every physical file that belongs to the dataset

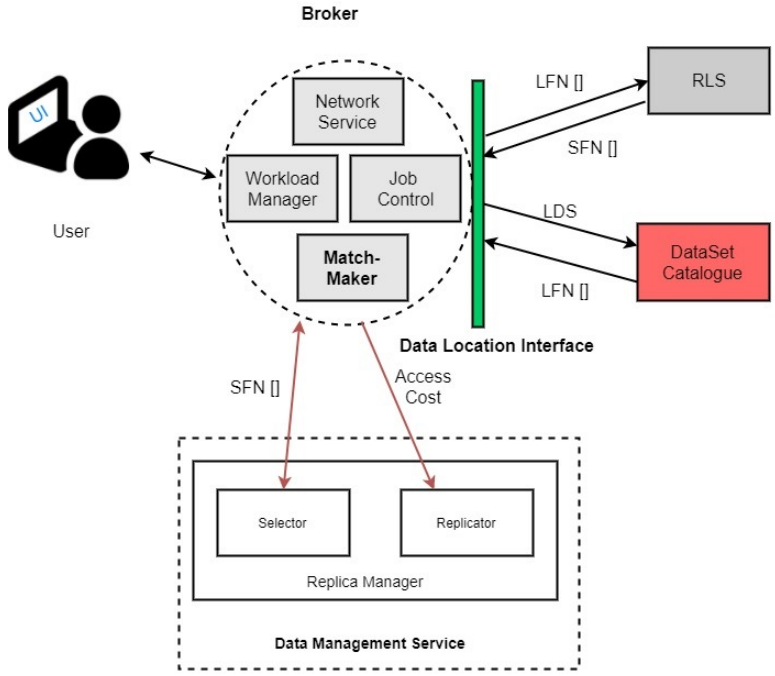


Figure 5.2: Data Location Interface

is housed in the same storage element and can always be accessed through the same protocol. If the NADI scheduler requires every physical location for a certain LDS, the DMS sends a query in order to fetch all potential physical locations where the dataset can be accessed. The outcome of this process is hence mapped by the service internally to obtain the corresponding hostnames of the SN. Abstractly, the NADI scheduler can treat the LDS and logical file names in the same way. It is imperative to know the locations of data type LDS and LFN replicas. The DMS returns replica locations of LFNs. LDSs require such a service to initialise the query; this is achieved by the DLI, which functions as the major interface between the catalogue for a particular data type and the NADI scheduler, thus allowing the scheduler to interact with any catalogue (which provides data locations) that leaks into the web service interface.

The DMS carries all the replica information in the network performance characteristics and investigates attempts to create another replica using the replication service in the DLI where the DC does not carry any replicas for the purposed data, as demonstrated in Figure 2 . This is aided by key factors such as SN space availability, replica size, bandwidth between SNs and CNs and lower average response time, thereby en-

hancing the application performance. The data location process permits applications to select a replica from different Replica Catalogues on the basis of data access characteristics and performance. A framework is provided by the RLS, which tracks the physical locations of the replicated data. When the scheduler calls for the LDS, the DMS makes use of the Replica Catalogue to recognise every replica location that contains instances of the physical dataset of the LDS, and it has to select the best instance for retrieval. The Replica Catalogue, on the other hand, builds and stores database mappings of site filenames and logical filenames for replicas.

The logical filename (LFN) refers to a name which describes the full set of file replicas. It is more efficient to look up all replicas in a Replica Catalogue using the LFN . The site filename (SFN) describes a file that is used by the storage management system. It can be a physical filename of a record which is stored in a disk or an LFN that is significant in a mass storage system.

5.2.1.2 Replica Manager

The Replica Manager consists of two sub-components: the Replicator and the Selector. The Replicator replicates or stores the chosen replica at the optimal location, which will be identified by the NADI algorithm. The Selector is responsible for choosing the replica that has the least access time cost. After the broker has received the FLNs and LFNs for the required replicas, they are sent to the Selector along with network status information in order to find the optimal replica . The chosen replica will be presented to the broker.

5.2.2 Algorithm

Basically, the algorithms detailed below perform the same as the previous algorithms in Chapter 4, Section 4.3, but they enable the replication service to enhance the application's performance . Therefore, these algorithms increase the chances of improving the efficiency of the application. In this section, the functionality of these algorithms will be explained. First, the broker receives all required resources from the users, including

VM numbers, application details and data location. Then, the broker communicates with the DS to gather all needed information such as network status, available resources and numbers of replicas. Subsequently, the broker calculates the completion time cost as mentioned in Chapter 3, Sections 3.5.1 to 3.5.3; it could use the replication service if necessary. The calculation of completion time is divided into three types of jobs: processing time, wait time in the CPU queues and data transfer time, as explained in detail in Chapter 3, Section 3.5. The main difference from the previous NADIv1 algorithm in Chapter 4 is in the processes for reading and writing files, which could be either remotely or locally, particularly if there are replicas or the replication services are being used. As one of the available options, the broker studies the possibility of file replication to DCs that do not have any replication. The replication service is typically used when there are heavy workloads; these files are used periodically. This helps to improve scheduling and optimise execution by reducing the frequency of remote data access.

In principle, to calculate the total time cost on any individual CN, a completion time matrix must be generated by computing the data transfer time and the time among all available CNs in each DC. This matrix consists of all three options mentioned above (local, remote or replication). If the total cost time of the remote CN is higher than the local cost for the remote data or the replica, the VM is scheduled to the local DC . Otherwise, the VM, jobs and data are moved (replicated) to the third location to obtain execution efficiency. Finally, the broker chooses the optimal solution from the various options. The steps for the proposed VM scheduling, as shown in Algorithm 5.1 below, will now be described .

Algorithm 5.1 VM placement algorithm

Input: V VM List, D Required Data, C Computing node list S Storage node list and J job List

Output: V List of VMs assigned to C list of computing node

```
1: for v = 1 TO V do
2:    N = 0 ; uN = 0
3:    for dc = 1 to Dc do
4:       for s = 1 to S do
5:          Check DC has a replica R
6:          if R ≠ 0 then
7:             N = N + 1
8:          else
9:             uN = uN + 1
10:         end if
11:      end for
12:   end for
13:   Chosen_c ← NULL
14:   TCT (J, D, S, R)
15:   RTC (J, D,S, uR)
16:   if Chosen_c ≠ NULL then
17:      if CTimeDDT < CTimeReplication then
18:         Allocation.add(v,Chosen_c)
19:      else
20:         Allocation.add(v,Chosen_c)
21:         replication.add(D, dc(Chosen_dc,Chosen_s))
22:      end if
23:      Updating: Cap(Chosen_c) = Cap(Chosen_c) - Load(v)
24:   end if
25: end for
```

- **(Input):** The NADI scheduler first receives the user's request details (number of VMs, application jobs, related data location and required processing time), then collects the resource information from the DSs such as CN availability (RAM, CPU and bandwidth), SN availability and data location information. This information comprises input parameters for the algorithm so that it matches all VM requests for optimal CNs .

- **(Output):** All VMs in the VM list are assigned to CNs that are listed in the CN list, thus, the CNs' capacities (RAM, CPU and bandwidth) are updated.

- (Line 1): This is a loop that aims to place VMs from the VM request list one by one.

- (Line 14) calls for the first function that calculates the completion time for all application jobs, including data transfer time, among all available replicas. It chooses the best replica for enhancing the application performance and minimising the overall completion time. This is accomplished by retrieving the data remotely or locally depending on the chosen CN and the location of the best replica. Algorithm 5.2 provides full details about this option.

- (Line 14) calls for the first function that calculates the completion time for all application jobs, including data transfer time, among all available replicas. It chooses the best replica for enhancing the application performance and minimising the overall completion time. This is accomplished by retrieving the data remotely or locally depending on the chosen CN and the location of the best replica. Algorithm 5.3 provides full details about this option.

- (Line 15): Similarly, this line calls for the second function, which is responsible for the replication service. This function investigates how costly it is to replicate a new place in the DCs that do not have any replicas for the selected data. It calculates the time for copying the data to the new location plus all job completion times. In this case, local data transfer minimises the overall completion time or perhaps the remote site is better than the local site. Algorithm 5.3 explains this option in detail.

- (Line 16) ensures that the functions above find an optimal CN on which to allocate the VM.

- (Lines 17-24) attempt to choose the optimal CN for minimizing the total completion time between the returning values from the above functions. Once it finds the optimal CN, it places the first VM on it and updates the available CN resources after the placement. At this stage, data can be accessed either remotely or locally. In the case of replication, data, jobs and the VM will be sent to the new location.

- (Line 25): Reaching this line means that all VMs in the VM list have been successfully allocated.

Algorithm 5.2 is responsible for choosing the optimal CN by calculating the job completion times, which include the computing and data transfer times. It will choose the best replica among all those that are distributed across all DCs. Then it will send this information to algorithm 5.1, as mentioned in the description above.

Algorithm 5.2 Calculates Completion Time using data transferring time

Input: S Storage node list and J job List , R data replicas

Output: Calculates Completion time for all jobs using all existing replicas among all CNs in each DCs

1: $MinTCT \leftarrow$ MIN ; $Chosen\_r \leftarrow$ MIN ; $Best\_r \leftarrow$ MIN
2: $Chosen\_dc \leftarrow$ NULL ; $Chosen\_c \leftarrow$ NULL
3: **for** $dc = 1$ to Dc **do**
4: **if** $N \neq 0$ **then**
5: **for** $c = 1$ to C **do**
6: **if** $Load(v) \leq Cap(c)$ **then**
7: **for** $r = 1$ to R **do**
8: **for** $s = 1$ to S **do**
9: **for** j =1 to J **do**
10: **if** s stores r **then**
11: $JT = JT + (r \ / \ BW(s, c))$
12: **end if**
13: **end for**
14: **if** $DTT < JT$ **then**
15: $DTT \leftarrow JT$; $Best\_r \leftarrow r$
16: **end if**
17: **end for**
18: **end for**
19: **for** j =1 to J **do**
20: $PT = PT + (st(j) + \frac{ni(j)}{cap(c) \times core(j)})$
21: **end for**
22: $CT = PT$
23: $TCT = DTT + CT$
24: **if** $TCT < MinTCT$ **then**
25: $MinTCT \leftarrow TCT$; $Chosen\_c = c$; $Chosen\_dc = dc$; $Chosen\_r = r$
26: **end if**
27: **end if**
28: **end for**
29: **end if**
30: **end for**
31: **return** $MinTCT$,$Chosen\_dc$, $Chosen\_c$,$Chosen\_r$

- **(Input):** consists of received application job requirements, all data replicas and the storage lists that are hosting all replicas.

- **(Output):** consists of sending the chosen DC, CN, best replica and the total completion time for all jobs.
- (Lines 1-2) are variables that are decided initially, and the optimal replica is stored in the SN. These lines also choose the optimal CN, Chosen_c, that has the minimal completion time, MinTCT, and excites in a certain DC, Chosen_dc.
- (Lines 3-4) check and calculate the completion time in each DC that has a copy of the data. If there are no replicas on the selected DC, the codes will not be executed.
- (Lines 5-6) provide a condition to ensure that the selected CN has the minimum capacities to place the chosen VM.
- (Lines 7-18): These lines first specify the replica location in the SN, then calculate the job access time for every available replica and choose the best one. In the end, the data access time will be calculated on the selected CN with the optimal replica selection, as shown in equation 3.15.
- (Lines 19-21) only calculate the computing time for all jobs on the chosen CN, as described in equations 3.20 and 3.21.
- (Lines 22-30) calculate the completion time for selected VM jobs, which is the sum of the computing and data transfer times with the optimal replica, as shown in equation 3.23. Then all variables that are mentioned in lines 1-2 are assigned to the best values and sent back to the main algorithm 5.1.

Correspondingly, algorithm 5.3 checks the possibility of replicating data to DCs that do not store any replicas. Then it calculates the replication service cost, which includes storage writing time and job completion times. It subsequently chooses the best CN for minimising the completion time. Finally, it will send the best information about the resources, including the CN, SN, DC and completion time, to algorithm 5.1, which responds by deciding to use replication or remote or local data transfer based on the optimisation decision, as previously discussed.

Algorithm 5.3 Calculates Completion Time using data Replication

Input: S Storage node list and J job List, un-replicas DCs uR, D Required Data, R data replicas

Output: Calculates Completion time for all jobs using by considering replication among all CNs in each DCs

1: $MinComTime \leftarrow$ MIN , $MinST \leftarrow$ MIN,
2: $Chosen\_dc \leftarrow$ NULL $Chosen\_c \leftarrow$ NULL
3: **for** $dc = 1$ to Dc **do**
4: **for** $r = 1$ to R **do**
5: **if** dc doesn't store r **then**
6: **for** $c = 1$ to C **do**
7: **if** $Load(v) \leq Cap(c)$ **then**
8: **for** j =1 to J **do**
9: **for** $s = 1$ to S **do**
10: $RT = d/\ WD(s, dc)$
11: **if** $ST < MinST$ **then**
12: $MinST \leftarrow RT$
13: $Chosen\_s = s$
14: **end if**
15: **end for**
16: **for** j =1 to J **do**
17: $CT = CT + (st(j) + \frac{ni(j)}{cap(c) \times core(j)})$
18: $LDT = d\ /\ BW(Chosen\_s, c)$
19: **end for**
20: **end for**
21: $TMV = RT + CT + LDT$
22: **if** $TMV < MinComTime$ **then**
23: $MinComTime \leftarrow TMV$
24: $Chosen\_c = c$
25: $Chosen\_dc = dc$
26: **end if**
27: **end if**
28: **end for**
29: **end if**
30: **end for**
31: **end for**
32: **return** $MinComTime$,$Chosen\_dc$,$Chosen\_c$,$Chosen\_s$, $MinST$

- **(Input):** consists of received application job requirements, all data replicas and the storage lists that doesn't host any replicas.
- **(Output):** sends the chosen DC, CN and SN, which are considered optimal, and the total completion time for all jobs.
- (Lines 1-2) are some variables that are decided. First, there is the optimal storage

chosen s , which will store the new data replica, and the replica time will be MinST. Second, it chooses the optimal CN chosen c that has the minimal completion time MinComTime and excites in a certain DC chosen dc.

- (Lines 3-15), first, check all DCs (which do not have any copy of data how) for the cost of replication w, including the available bandwidth between the SN and CN and the data writing time in every SN (as mentioned in equations 3.13 and 3.14). Then, it returns the best SN and the optimal time for replication.
- (Lines 16-20) are a set of calculations that compute the queuing, processing and
- (Lines 21-31) are the time required to calculate the completion time for all jobs on all CNs (as discussed in equation 3.24). The completion time consists of the sum of computing, the data transferring time and the replication cost. Then, it will compare these times to select the best DCs' resources, which include CN and SN.
- (Line 32) assigns all the variables that are mentioned in lines 1 - 2 to the best values and sends them back to the algorithm 5.1. These variables are the optimal CN, the minimum time for replication and communication , and the best SN that will store the certain data for replication.

5.3 Simulation results

In this section, the simulation tests are presented aimed at evaluating the proposed adaptive Cloud Placement algorithm using the CloudSim simulator [37], which is used to model the Cloud environment. From the following experimental results, the aim is to enhance application performance and minimise the completion time by considering the network status between DCs' resources. This chapter uses a replication service only if it maximises application efficiency. However, Hussein's approach [96] has another perspective, which tries to generate a replica for all files in DCs in order to improve the overall data reliability in DCs and avoid failure in SNs. This replica could enhance

application performance, but it would be a burden to the network because it would overload the network. It would cause network bottlenecks. In addition, the replication service should be used wisely to avoid wasted overload storage capacities. The previous NADI approach [121] considered the available bandwidth in each link in the DCs' network and chose the optimal one as discussed in the previous chapter, whereas Piao's algorithm [82] considered the best link between the CN and SN, which stores the first file . Then, the system will collect all needed parameters and apply them equally on all the compared algorithms. Next, the simulation set up is described and the results are discussed .

5.3.1 Simulation Set-Up

The simulated model was composed of one Cloud DC that contained 10 CNs. Each CN had two quad-core processors and 16 GB of RAM. The CN hosted user application instances and the mechanism . The DC had one broker that received users' requests and placed the VMs on the appropriate CNs based on the allocation algorithm that it was used. There were 25 VMs that required one core and 2 GB of RAM . The total number of files was 25, which were distributed over the SNs in the DCs; the size for each file was between 0.1 and 3.0 GB (generated randomly). Every VM had a number of jobs (e.g., 50 jobs, gradually increasing to 100 ,..., 250 jobs). Every job needed to access between 1 3 files, which were stored on an SN (chosen randomly). Table 5.1 shows all the parameters that were used. This investigation used the second scenario, which was discussed in section 3.2 of chapter 3.

Parameters	**Values**
Number of CNs	10
Number of VMs	25
Number of jobs	50 -250
Number of files	1-3
File size / job	[100-3000]MB

Table 5.1: Parameter Settings for the First Test in the NADIv2 Scheduler (2nd Scenario)

5.3.2 Simulation Evaluation

Figure 5.3 shows the computing time for all jobs that are discussed in the table above. Both the previous approach and the NADI approach got better results than others because we considered the CPU usages during the scheduling. Upon close examination, the NADI shows the best results due to its possibility of providing more than one replica. Hussein's [96] and Piao's [82] approaches did not consider the CPU load, which is one of the advantages of the NADI approach. Their results are worse whenever the number of jobs and workload are increased (as shown in Figure 5.3). The computing times included the time required to schedule one of the best CNs and execute the job plus the time required to send the data and job back (which was made by a NADI scheduling decision). In this test, NADI was able to select and employ the resources that were least loaded, had smaller queues and could run the jobs more efficiently than other approaches. It may be seen that each algorithm has a lot of variation; however, in all cases, the NADI algorithm shows the best performance. It is scalable for a large number of nodes, and it has the shortest execution times compared to other algorithms (as shown in Figure 5.3).

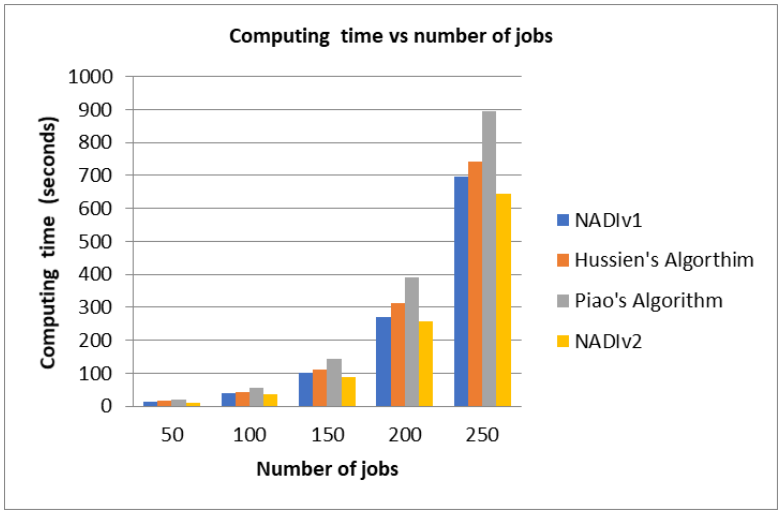


Figure 5.3: The computing time for all jobs

Now the simulation results to test the network efficiency of the NADI algorithm are presented. In a cloud environment, nodes are connected via variable network links; an

algorithm that does not use the network as a resource in the scheduling decisions will be inefficient and will waste resourcesas well as complete tasks in an untimely manner. The network is an especially critical resource for data intensive scheduling. As has been established in earlier chapters and through practical results, ignoring the network can lead to suboptimal scheduling decisions. The response of each algorithm is shown in Figure 5.4 where NADI clearly has a stable and consistent performance, whereas Piao's algorithm [82] only considers a certain file . Hussein's [96] and NADI's approaches are exchanged for the optimal results reciprocally based on the availability of a replica's numbers. This suggests that NADI is suitable for data-intensive jobs; its consistent performance further verifies the suitability of this approach.

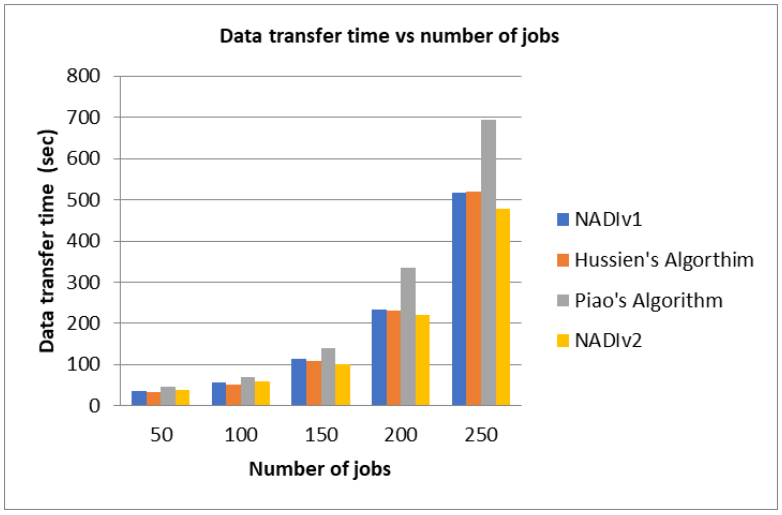


Figure 5.4: The data transfer time for all jobs

The completion time includes the time required for queuing, processing and accessing certain data for a job. The queue time is almost proportional to the execution time because, if the job is running and taking a lot of time, the waiting time of the next job will also increase since it will spend more time in the queue. Although the execution time does not include queue times, a larger number of jobs running in any CN can influence the queue time. Furthermore, increasing the number of jobs in the queue can influence the overall completion times (such as the scheduling time, queuing time and execution time) of the new jobs since they will be competing for the resources to get an execution slot, especially if the jobs are composed of sub-jobs. In addition,

the completion time includes the time required to access the data remotely or locally for every single job. This is primarily due to the fact that NADI preferentially selected those CNs for job execution, which could quickly execute jobs (i.e., those that have short local queues with low latency). Figure 5.5 shows the execution optimization achieved by employing the NADI algorithm. It can be seen that, with an increasing number of jobs, the completion performance increases; this indicates the effect of the NADI scheduling approach for the job scheduling. By comparing the approach with Hussein's [96] and Piao's [82] approaches, the NADI approach here shows the best performance, being a result of considering the data transferring and computing times . In the proposed approach, the network and CPU parameters are considered to select the best CN before making any VMs' scheduling decisions for data-intensive jobs.

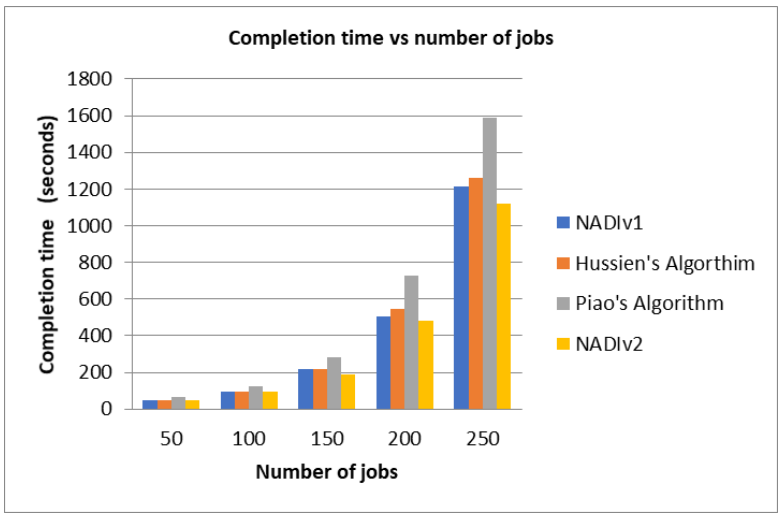


Figure 5.5: The completion time for all jobs

Figures 5.6 illustrates the 10 CNs resources' usage (CPU, bandwidth and RAM) that host the 25 VMs. Figure 5.6.a illustrates the CPU usage, and it may be seen that both NADI approaches still have the advantage over Hussein's and Paio's algorithms. This is because the approaches choose the CNs wisely and based on how busy the CPU is. Figure 5.6.b describes the bandwidth usage, which, in Paio's approach, eventually deteriorates due to the requirement of file transfer times between the application and the SNs. Both NADI algorithms and Hussein's algorithm get better results because they consider the files' total transfer time on each CN. Figure 5.6.c shows the RAM usage

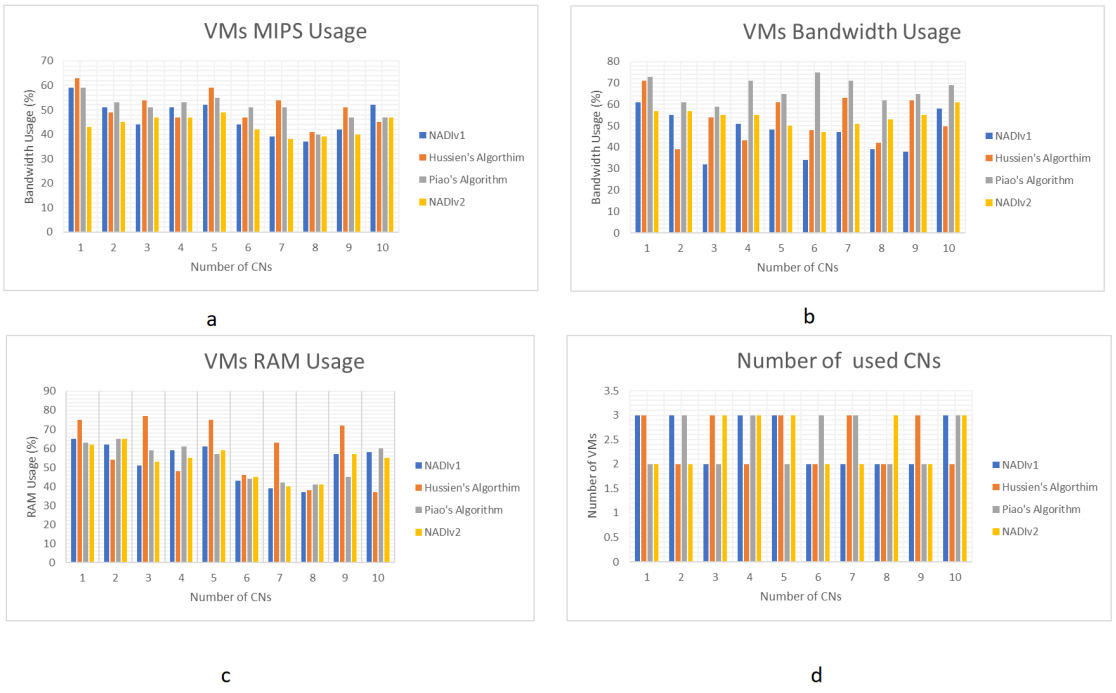


Figure 5.6: Comparison of the CNs overall resource usage

rate among different proposed algorithms that have been mentioned in section 5.4. It was observed that the NADI solution can minimise the RAM usage and will increase the application efficiency. It checks the signification of the CPU load because they have a correlation that is shown by statistics that have been gathered from between RAM and CPU for the whole results. Thus, the RAM usage is minimised. During the run time, if the performance of an application is affected significantly, it is often because of factors such as high CPU usage and high memory usage of the CN where it is hosted. So, CPU usage and RAM usage can often correlate, but this cannot be generalised [140–142]. When the algorithms are compared, it is observed that the NADI algorithms generally have the best values. In contrast, the main disadvantage of Hussein's algorithm is in increasing the bandwidth usage because it replicates the majority of files in order to increase the reliability. This could affect the communication between an application and its files during the replication service. If we looking closely the NADI approaches, is is seen that the previous approach of NADIv1 is slightly better than NADIv2 in the short term, but it will improve whenever the workloads, number of jobs and data

size increase. This will be shown in the next investigation, which will be discussed in the following section. Figure 5.6.d demonstrates the number of CNs utilised by the scheduler to place the VMs. Here, all the proposed algorithms distributed the VMs on CNs evenly because every approach has its own way of optimising these VMs (depending on the optimization objectives). The only objective of Piao's algorithm is to place the VMs on the CN that has the least network bandwidth between SN and CN, whereas Hussein's approach only aims to create many replicas to improve system reliability. The NADI approach is a mixture of this objective and using replication in order to see if the application performance will be enhanced. It is important to decrease the usage of resources and increase the application performance at the same time. As shown in Figures 5.7 and 5.6, the NADI solutions minimise the total usage of resources, and the overall completion time is shown in Figure 5.5, thus improving the performance for the applications.

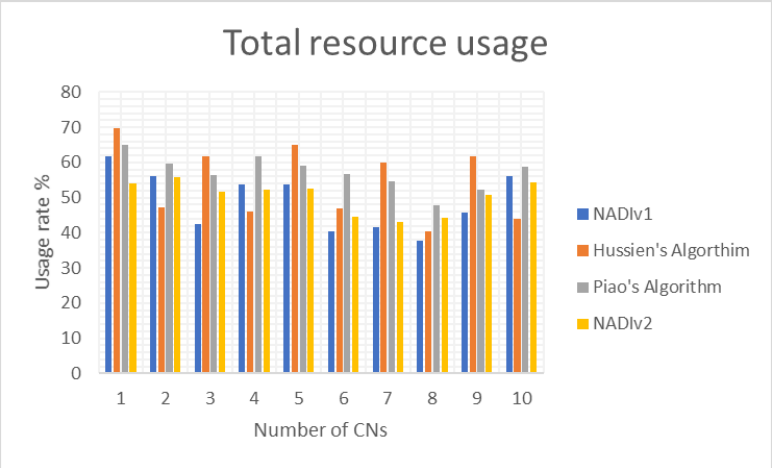


Figure 5.7: The overall resource utilization rate

In the following figures, some of the parameters have been changed (as shown in Table5.2). In this test, a large-scale evaluation was used to make it more complicated by increasing the total number of VMs per each user; every VM has a requested number of files . This test used the second scenario, which was discussed in Section 3.2 of chapter 3. As a result, there were three DCs, and every DC had 350 CNs. The simulation ran for 10 minutes, and each minute there were some requests for VMs that needed to be placed

. These VMs increased gradually every minute (from 150 to 1500 in total). Each VM requested 3 to 5 files that were spread over the distributed SNs. The jobs also increased gradually in each round from 50 to 1500 . In addition, the same comparison as Hussein's approach [96], Piao's algorithm [82] and the previous NADIv1 approach [121] are used.

Parameters	**Values**
Number of DCs	3
Number of Users	50-1500
Number of jobs	50 - 250
Number of files	3- 5
File size / job	[1, 30] GB

Table 5.2: Parameter Settings for the Second Test in the NADIv2 Scheduler (3rd Scenario)

As can be seen from Figure 5.8, the number of VMs and their jobs gradually increased; thus, the workloads in the DCs increased significantly. So with these increments, the application performance could be degraded if the placement of applications is not used wisely. In Figures 5.8.a and 5.8.b, in addition to the overall completion time, we compared the individual components of the schedule (namely the data transfer time and computing time of the scheduling approaches). From Figure 5.8.a, it can be observed that both NADI solutions are the best results among others for data transferring time. Hussein's algorithm is considered to be the worst one in most rounds, although his solution was created to provide high availability and improve performance. One of drawbacks to this approach is replicating the frequently used files more than once in order to improve the overall reliability of the system. Despite the benefits of the replication service, it minimises throughput and maximises the delay in the DCs' network linksif it is used frequently without planning or awareness . So, it could affect the communication between the CNs and SNs if the communication is done remotely; thus, application performance will be degraded. Piao's solution is the best only when the VM requests a single file that is not possible to be achieved; this is because, in the cloud environment, the data is distributed over many SNs. The scheduler should consider all the files and choose the optimal CN that minimises the data transferring time. Also, it should be observed that Hussein's algorithm is better than Piao's algorithm

because there are many replicas over the DCs and because his algorithm determines the best replica location to meet the specified QoS. Figure 5.8.b illustrates the computing time, including the processing and waiting time. Hussein's and Piao's algorithms are not considered to be compute-intensive aware, so their solutions give the worst results. The differences are due to placing the VMs on the unsatisfied CNs. It was observed that Hussein's algorithm is better than Piao's algorithm because, after choosing the best replica, the communication with the data will be carried out locally. This will accelerate the computing of jobs during the stage data in and out and waiting in the processor's queue. Thus, the job will be finished early. The new approach NADIv2 shows better values than the previous algorithm NADIv1 because there is the possibility of replicating files if the replication service enhances the application performance and minimises the total completion time; the previous solution, in contrast, lacked a variety of options.

Since both the data transfer and computing times for Hussein and Piao's algorithms are considered to be the worst results, the total completion time will also be unsatisfactory . In Figure 5.8.c comparison graph, it is clear that the performance (or completion) time of jobs obtained using NADIv2 was better than the other schedulers. The reason for this was that the new approach NADIv2 works on selecting the stable connecting links; this optimises the data-intensive scheduling process and chooses the optimal CN (which has fewer job queues and processing delays) in order to reduce the completion time. These two considerations not only improved the completion and processing times of the jobs but also reduced the queue times of the jobs. Generally, NADI has improved the completion time for all jobs since it selected CNs with the required data for the job execution, had fewer loads and had fewer jobs in the queue; all this contributed to the optimization of execution. CNs with a higher number of jobs already in progress or heavily loaded CNs can lengthen execution times.

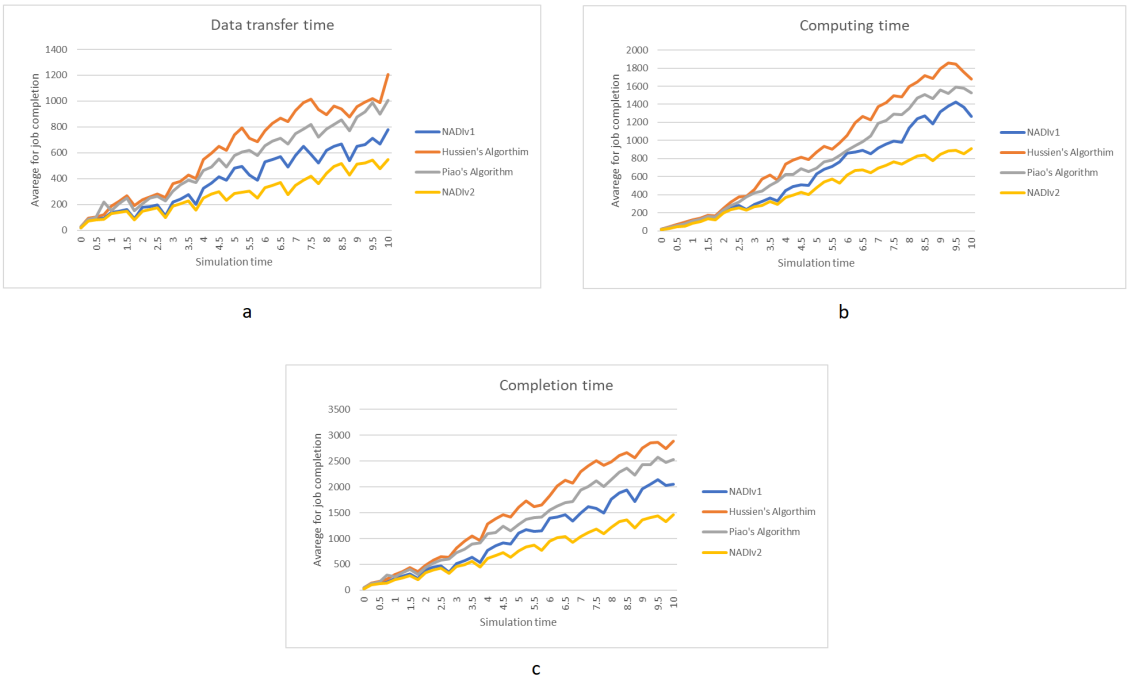


Figure 5.8: The completion time for all Jobs

The usage of bandwidth, CPU, RAM and the total usage of resources (as shown in Figure 5.9) have been evaluated. As can be seen in Figure 5.9.a, the NADI algorithm generally has the best result compared to the other algorithms. The optimal solution is due to the consideration of the CPU load. Hussein's and Piao's algorithms give the worst results compared to these approaches; this is because their solutions did not consider the CPU queues and loads. Figure 5.9.b shows that the bandwidth usage for the NADI algorithm is better than the others because the available bandwidth in each link in the DC's network was considered and the optimal one. Piao's algorithm considers the best link between the CN and SN that stored the first file . Hussein's approach is considered to be the worst solution because it creates a number of replicas for a certain file ; as a result, the available bandwidth is degraded, and the application performance is affected. It can be observed that the percentage of usage is improved continuously whenever the workloads and VMs' requests are increased. The increase in network cost can also affect the overall performance of the distributed systemespecially in terms of transfer and communication time. Therefore, it is an important consideration for all scheduling decisions. A lower bandwidth results in high network costs; the increase in network

costs affects the overall performance of the distributed system. Figure 5.9.c shows that the consideration of availability options (such as bandwidth and CPU) enhances the RAM usage from 10% to 25%, reflecting positively on the application's performance. The new algorithm can decrease the usage of CNs and minimise the total execution time (as shown in Figures 5.8.b and 5.9.c). The overall performance of the application will be sequentially enhanced. By increasing the number of jobs, it is evident from Figure 5.9 that the average time needed to execute a job increases. More jobs competing requires more time for a specific job to be completed. From Figures 5.8.c and 5.9.d, it is clear that the NADI algorithm's scheduling approach has improved the execution times of the jobs. This time is calculated by dividing the available CNs by the number of jobs, and it is indicative of the aggregated execution times.

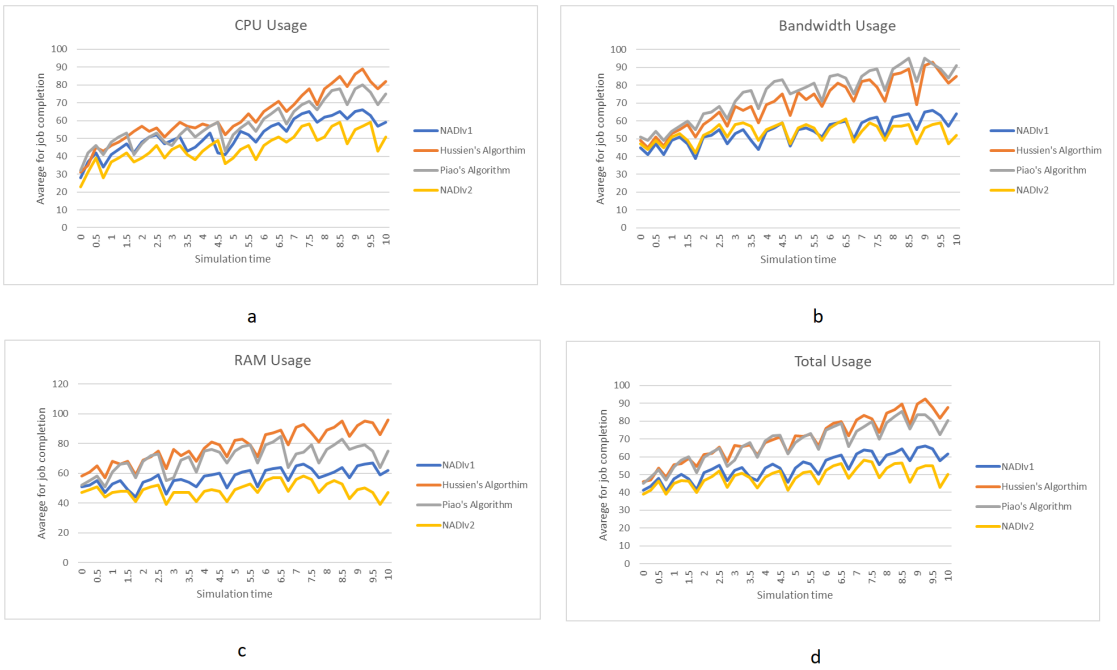


Figure 5.9: Comparison of the CNs overall resource usage

The number of CNs allocated in the NADIv2 algorithm with previous approache NADIv1, which were discussed in the previous section has been compared. Figure5.10.a compares the total execution time for all VMs in each round, whereas Figure 5.10.b compares the number of CNs selected between these algorithms. It can be observed that the NADI algorithms minimise the total execution time while the total number of

used CNs increased. The execution time for these algorithms shows that the execution time for the jobs can be reduced up to 47% if these algorithm's are used. Since these VMs are hosting data-intensive applications, data will be transferred between VMs while the execution is in progress. Data transfer between VMs takes time depending on the distance between these VMs and the amount of data being transferred. The overall performance of the application will be sequentially enhanced. The new algorithm can decrease the usage of CNs and minimise the total execution time (as shown in Figure 5.10). The overall performance of the application will be sequentially enhanced.

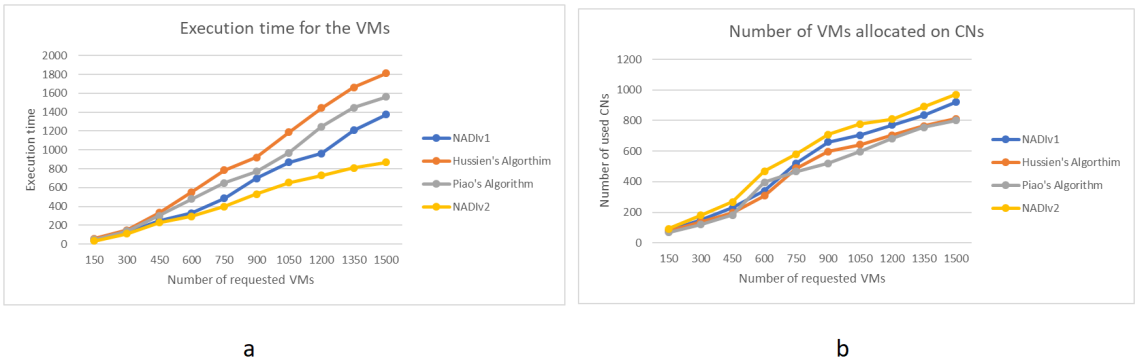


Figure 5.10: Number of VMs Allocated on CNs

Figure 5.11 shows the total number of replicated files that were used in each round. The NADIv2 was compared with Hussein's approach only because these two approaches are used in replication services. Hussein's algorithm is based on replicating all files that are used frequently in order to increase the system's reliability and availability . The number of replicas for every file will increase dramatically in each node if there is a better achievement for the new replication factor. The replication factor is defined as the average rate of the popularity degree and the average availability of replicas on the different SNs of a file. Whereas the popularity degree is defined as the future access frequency for a certain file. In contrast, NADI usually does not use the replication service unless it helps to increase the application efficiency and minimise the completion time. The reason behind this is that the replicated files could increase the rate of network latency, which affects any communication between the application and its related data especially if the communication is done remotely. Thus, the application performance

will be affected. In general, it is not possible to keep the VM access to its related data local; due to the data volume capacity, CNs use constraints that restrict VMs from being placed in the ideal CN [135]. As can be seen in Figure 5.11, the replication services between these approaches are very different due to the research 's main objective. The number of replicated files in the NADI solution is lower in order to keep the application performance high. However, Hussein's algorithm increases the replication to maximise the system's reliability and availability as well as to deal with the data locally. Therefore, it negatively affects the applications' performanceas was discussed in the section on the experiments . The NADIv2 was used for the replication service in order to maintain application performance efficiency and minimise the throughput latency.

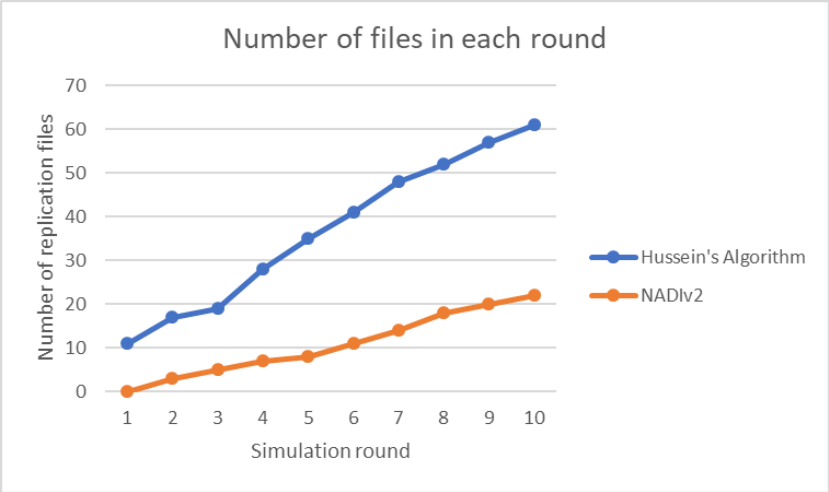


Figure 5.11: Number of Replication Service

5.4 Chapter summary

Sharing a vast quantity of storage resources, knowledge, information and computing resources is greatly benefitted by cloud computing. The applications require data retrieval out of distributed storages; in the meantime , application performance could be influenced by the available bandwidth between SNs and CNs with an unstable network status. A network-aware, VMs-placement algorithm for a data-intensive approach has been described in this chapter while considering computation power and its load, less network traffic, and the locations of replicas. This approach is offered in order to obtain

an optimised completion time for jobs in each VM. The NADI scheduling optimization approach has been developed to generate the time-cost matrix for the overall completion time of jobs and the optimisation of these values. It chooses the optimal site to allocate the VMs, which requires the least amount of time . This approach has a Data Management Service that deals with the data and their replicas' information. The DMS has been introduced; it consists of three major components, which are the Replica Location Service (RLS), Replicator and Replica Catalogue. The RLS was launched to acquire the most effective physical replica of the dataset, and it provides access to this information. In contrast, the Replicator is used to replicate the required replica into the chosen location. Every file has some copies of replicas that are distributed throughout the storages in cloud DCs. So, this file and his replicas only have a Unique Identifier name UID . These names will be stored in a database that maps between a Logical Filename (LFN) and Site Filename (SFN) and is called the Replica Catalogue. However, each replica has a LFN and a SFN. The LFN identified each replica, which can be looked up in a replica catalogue. The SFN is used by the storage management system to allocate the replica's place. The algorithms that are used in this chapter have been explained in depth. The results show a significant improvement on the average completion time by adopting the NADI scheduling approach. The next chapter will discuss using the overlap technique to reduce the computing time. This technique allows the transfer of data about jobs in advance while it is queuing, waiting to be processed and other jobs are being executed.

6

Overlapping Data Transfer With Job Execution

In the previous chapter, the NADIv2 design was introduced and the update in the DMS shown . The DMS consists of three major components, which are the RLS, Replica Catalogue and Replicator; they have been discussed in detail. To summarise the roles of these components, the RLS provides access to the chosen replica, and the Replica Catalogue stores all the information between the replicas' names and physical location. The Replicator replicates the data in a chosen physical locationif it is commanded by NADI. Through simulation results as well as graphical and analytical details , it was demonstrated that this method significantly optimises the completion times of jobs and improves the performance of NADIv2 especially when compared with the previous version. Data-intensive applications require communication between CNs and SN, and the placement of VMs and location of the data affect the overall computation time. For effi-

cient computation, data-intensive applications require a larger volume of data, which is higher than ever . Approaches that are aimed at intensive computation, such as conventional scheduling approaches, do not consider the data requirement of the application, which leads to poor performance. In the cloud environment, the scheduling of data-intensive applications in an efficient manner is very challenging. Additionally, average turnaround time and process utilization should be considered in such environments.

In this chapter, a NADIv3 scheduler approach will be introduced. It considers not only the data's location but also the computational power of CNs and its loads when it makes the VM allocation decision. It also uses the overlap technique that allows a job to pre-fetch the required data in advance while it is waiting in the CPU's queue and other jobs are being executed. This technique reduces the execution turnaround time of jobs. Thus, it minimises the total completion time of jobs and enhances the performance. Then, this chapter shows the new design of the NADI components and how the overlap technique works. The main three components are the Data movement Manager (DMM), Data Scheduler (DSC) and Local Disk Manager (LDM). The DMM is responsible for executable transfer and anticipated data movement. To accomplish this, a data movement specification file was created; then, it was delivered to the DS, which traces all data transfer tracking . The LDS is responsible for allocating and removing the required data on the CNs disk (Execution Site).

The DMM is responsible for executable transfer and anticipated data movement. To accomplish this, a data movement specification file was created; it was then delivered to the DS, which traces all the data transfer tracking . The LDM is responsible for allocating and removing the required data on the CNs disk (Execution Site). Moreover, the results obtained from the simulation indicate considerable improvement of the performance of jobs and shortening of the total completion time.

6.1 Introduction

Cloud computing provides highly scalable, elastic services on demand on a pay-per-use basis. Nowadays, the acceptability of cloud computing is very high, and it is increasing day by day. The impact of cloud computing on our daily lives is significant, varying from social networking to sensor networks. In modern scientific research communities, it is increasingly common for many experiments to produce and process sets of data in terms of terabytes or petabytes. For instance, every year, four high-energy physics experiments conducted at Amazon EC2 generate and process many petabytes. More data is expected to be produced and processed at Amazon EC2 as these experiments are expected to end in 14 to 20 years [143]. Consequently, these applications also require large storage units and resources for computation, which are normally distributed geographically. For the data to be processed, the applications move the data to the computation sites. This data transfer can take several hours or even days [144]. To solve complex problems, cloud computing [145] enables users to tap into the capabilities of the various distributed and heterogeneous resources.

Schedulers [107,146] are used to facilitate cloud computing. Jobs that are submitted to the cloud are dispatched by schedulers to various sites where they are executed and monitored. In conventional scheduling approaches, only the computational requirements are considered while the data requirements are disregarded. As such, the movement of data is coupled together with the computations, which necessitates computational resources to be allocated while the data are being transferred. This approach has some disadvantages. In most cases, there are dedicated nodes on the cloud (that are tasked with the transfer of the data) as well as a shared file system for computational nodes and the data servers. For this reason, no resources on the computational nodes are required for the data transfer. This results in wasted computational resources during the data transfer if the scheduler is unaware of data and when the movement of data is carried out as part of the computation.

Additionally, cloud environments are normally characterised by different conditions

of the network depending on the site. However, the lack of knowledge about the data requirements makes it impossible for the scheduler to optimally select the computational resource for the job, which results in poor performance (such as increased job turnaround time and minimal utilization of the resources). For schedulers such as DAGMan [97] and others that are data aware, data are first transmitted to a computational site by a scheduler before being passed to the local scheduler. Though the use of resources is improved by this approach, there are still limitations. This approach only picks a computation site based on the computational aspects alone and does not consider the data aspects. As such, the approach might pick a site that takes longer to transfer the data compared to others. Also, the queueing of the jobs is only done after the data has been transferred.

In this chapter, the third version of the NADI approach for VMs' scheduling is presented, which considers data and computation as well as the network features during the allocation of VMs and the scheduling of multiple or single jobs. This chapter also aims to improve the data-intensive turnaround time by overlapping the data transfer time with the queuing time of the job and/or the computation time of other jobs.

6.2 NADIv3 design for VMs Placement Scheduling

The proposed NADIv3 takes into account not only computation resource requirements, but also data requirements (storage space, network condition, etc.) when making scheduling decisions. The assumption is that a job is composed of three stages: data transferring-in, execution and data transferring-out. In the first stage, the input data for the job is transferred from the precedent job CNs. In the execution stage, the job runs with the input data and produces output data for the descendent jobs. In the data transferring-out stage, the output data is transferred to the descendent job or is submitted to the last job in the workflow. The execution procedure for all stages is executed in a pipeline. The figure 6.1 illustrates the new scheme. The broker calculates the computing time and the data transmission time for all jobs and chooses the optimal computing node. In this model, the input data is fetched from a remote location in

advance of choosing the computing node, which has the latest data transmission time, queuing time and execution time. Then, every job's data will be staged in advance during the computation time of other jobs (overlapping). This approach attempts to reduce job turnaround times by decoupling the SI and SO from the job execution and utilising the computational resources. Compared to the previous approach, this solution enhances resource utilization. The architecture of NADIv3 is constituted of two components, which are NMS, DS, IS, DLS and WMS. The functionalities, actors and roles of these components were discussed in chapter 4, section 3.3.2. The DLS component has been slightly changed to DMS, which is more comprehensive (as shown in section 5.2.1 of chapter 5). In this chapter, the DMS has been developed to send data in advance (overlap). This component is considered the main contribution , and it will be described in the following sections.

6.2.1 Data Management Service

The main role of the DMS is to manage data requests, allocate data locations, transfer and track data from SNs in advance to the CN-side, and clean up the disk. The DMS contains a single component, which is the DMM.

6.2.1.1 Data Movement Manager

DS has the role of finding a suitable resource for VM's jobs and its related data. JSCS (discussed in detail in 4.2.5.3 of chapter 4) collects information that is useful in his matching process. The moment a resource is selected, extra information is written in the task specification file of the job. This information plays a vital role in guiding the JSCS as the submission process continues as well as in guiding the DMM during the transfer of data in the advanced processes . This means that the job will have no alterations in the specification file. Thus, the DMM moves the expected data and completes the executable transfer of the data. To achieve this, a data movement specification file was created; it is submitted to the DSC and, consequently, the data transfer is tracked. The DSC deals with the storage of data in heterogeneous surroundings and in the process of

transferring data. It performs the following operations: preparation of data, verification of control access and transmission of data. Additionally, it schedules the management of the transfer of data, queues and monitors, and it ensures that they will be finalised. The LDM found in the CN-side has the functionality of placing the data in a prior arranged area inside the CN. The activities done by the LMD are the allocation of free disk space and the creation/removal of the transitory area. If the application requests more data, it usually contacts LDS, which requests the data from the DMM. Then, the DMM offers the requested data by enquiring about its location from the Replica Location Service.

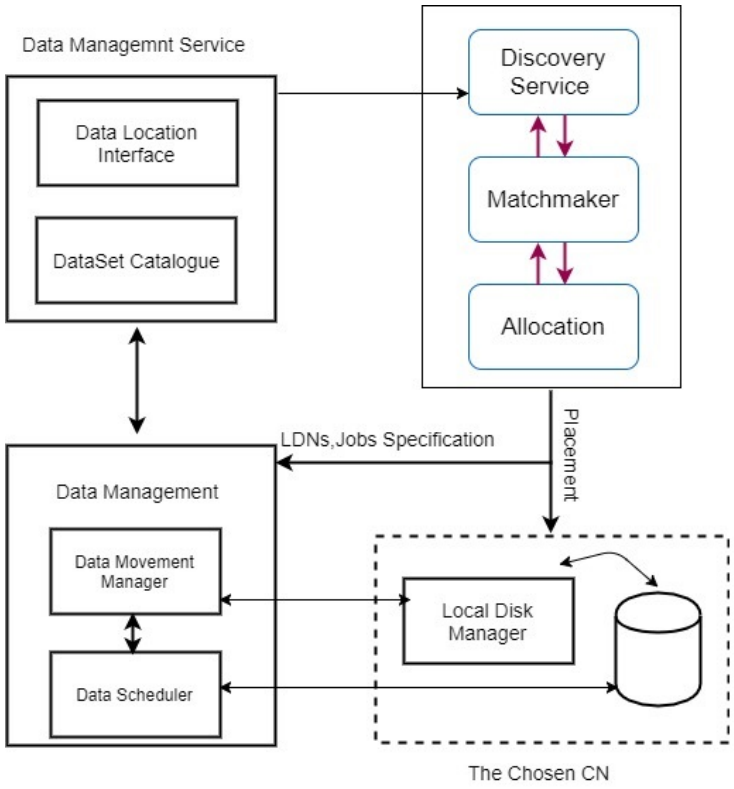


Figure 6.1: NADIv3 Scheduler architecture

6.2.2 Algorithm

The algorithms below perform similarly to the previous algorithms (5.1 - 5.3), which have been discussed in chapter 5 in section 5.2.2. Algorithms 5.1 - 5.3 try to minimise the completion time and enhance the application's efficiency. Thus, they decrease the data

transfer time (whether the communication is local or remote) by creating a new replica if the application performance requires that. They also try to reduce the computing time by considering the queuing, waiting and execution times. In algorithms 6.1 - 6.3, they try to minimise the execution time by staging in the requested data in advance of the next jobs, while the current job is executed normally. This mechanism improves the applications performance and minimises the overall computing time. In addition, the functionalities of algorithms 5.1 - 5.3 are aimed at reducing the data transfer time. In this section, these algorithms' functionality is explained.

First, the broker receives all required resources from the users; the resources include VM numbers, application details and data locations. Then, it communicates with the discovery service to gather all the necessary information, such as the network status, available resources and numbers of replicas. After that, it calculates the completion time cost as was mentioned in chapter 3 (sections 3.5.1 to 3.5.3) and uses the replication service if required or fetches the data remotely . The calculation of completion time is divided into three types: jobs' processing time, the time required for waiting in the CPU queues and data transfer time (as was mentioned in section 3.5). Then, the broker studies the possibility of files' replication if the to DC has no have any replicas of the required file as one of the NADI algorithm option. The replication service is usually used when there are heavy workloads, and these files are used periodically. So, it helps to improve the scheduling and execution optimization by reducing the frequency of remote data access. In principle, to calculate the total time cost on any individual CN, the completion time matrix must be generated by computing the data transfer time and computing time among all available CNs in each of the DCs. This matrix consists of all three options mentioned above (locally, remotely or using replication). If the total cost of time for the remote CN is higher than the local cost (which contains the data or the replica), then the VM is scheduled to the local DC. Otherwise, the VM, jobs and data are moved (replicated) to the third location for efficient execution. At the end , the broker chooses the optimal solution from the various optionswhether the access to data is local, remote or uses the replication service.

In the following algorithms, a new mechanism was added that minimises the computing time by sending the requested data in advance to reduce the stage-in and stage-out. This mechanism works due to the DMS which has two ways of dealing with the required data (or replicas): locally or remotely. In the case of remotely accessed data, the data movement manager will contact the data location service to allocate the replica's location. Then, the DMM will send the required data to the data scheduler, which will send the data to the local disk for the chosen CN. If there are updates to other data, the local disk manager will contact the DMM to provide the requested data. In the case of locally accessed data, the local manager will provide the requested data for the local SN to the local CN disk in advance in order to minimise the computation time. The steps for the proposed VMs' scheduling will be described (as shown in Algorithm 6.1).

Algorithm 6.1 VMs Placement using Overlap

Input: V VM List, D Required Data, C Computing node list S Storage node list and J job List
Output: V List of VMs assigned to C list of computing node

```
1: for v = 1 TO V do
2:   N = 0 ; uN = 0
3:   for dc = 1 to Dc do
4:     for s = 1 to S do
5:       Check Datacentre has a replica R
6:       if R ≠ 0 then
7:         N = N + 1
8:       else
9:         uN = uN + 1
10:      end if
11:    end for
12:  end for
13:  Chosen_c ← NULL
14:  TCT (J, D, S, R)
15:  RTC (J, D,S, uR)
16:  if Chosen_c ≠ NULL then
17:    if CTimeDDT < CTimeReplication then
18:      Allocation.add(v,Chosen_c)
19:    else
20:      Allocation.add(v,Chosen_c)
21:      replication.add(D, dc(Chosen_dc,Chosen_s))
22:    end if
23:    Updating: Cap(Chosen_c) = Cap(Chosen_c) - Load(v)
24:  end if
25: end for
```

- **(Input):** The NADI scheduler first receives the user's request details (number of VMs, application jobs, related data location and required processing time), then collects the resource information from the DSs such as CN availability (RAM, CPU and bandwidth), SN availability and data location information. This information comprises input parameters for the algorithm so that it matches all VM requests for optimal CNs .

- **(Output):** All VMs in the VM list are assigned to CNs that are listed in the CN list, thus, the CNs' capacities (RAM, CPU and bandwidth) are updated.

- (Line 1): This is a loop that aims to place VMs from the VM request list one by one.

- (Line 14) calls for the first function that calculates the completion time for all application jobs, including data transfer time, among all available replicas. It chooses the best replica for enhancing the application performance and minimising the overall completion time. This is accomplished by retrieving the data remotely or locally depending on the chosen CN and the location of the best replica. Algorithm 5 provides full details about this option.

- (Line 14) calls for the first function that calculates the completion time for all application jobs, including data transfer time, among all available replicas. It chooses the best replica for enhancing the application performance and minimising the overall completion time. This is accomplished by retrieving the data remotely or locally depending on the chosen CN and the location of the best replica. Algorithm 5.2 provides full details about this option.

- (Line 15): Similarly, this line calls for the second function, which is responsible for the replication service. This function investigates how costly it is to replicate a new place in the DCs that do not have any replicas for the selected data. It calculates the time for copying the data to the new location plus all job completion times. In this case, local data transfer minimises the overall completion time or

perhaps the remote site is better than the local site. Algorithm 5.3 explains this option in detail.

- (Line 16) ensures that the functions above find an optimal CN on which to allocate the VM.

- (Lines 17-24) attempt to choose the optimal CN for minimizing the total completion time between the returning values from the above functions. Once it finds the optimal CN, it places the first VM on it and updates the available CN resources after the placement. At this stage, data can be accessed either remotely or locally. In the case of replication, data, jobs and the VM will be sent to the new location.

- (Line 25): Reaching this line means that all VMs in the VM list have been successfully allocated.

Algorithm 6.2 is responsible for choosing the optimal CN by calculating the jobs' completion times, which include computing and data transferring times. It also estimates the saving time using the overlap technique. During that , it will choose the best replica among all replicas that are distributed over all DCs. Then, it will send all this information to algorithm 5.1 (as was mentioned in previous chapter).

- **(Input):** will receive the application jobs' requirements, all data replicas and the list of storage locations that are hosting all replicas.

- **(Output):** will send the chosen DC, CN, best replica and the optimal overall completion time for all jobs.

- (Line 1) includes variables that decide, first, the optimal replica, which is stored in the SN. Second, they choose the optimal CN chosen c that has the minimal completion time MinTCT and is executed in the certain datacentre chosen dc.

- (Lines 2-3) check and calculate the completion time in each of the DCs that have a replica of the data. Otherwise, it will not execute the following codes if there are no replicas in the selected DC.

- (Lines 4-5) make sure that the selected CN has at least the minimum capacities needed to place the chosen VM.
- (Lines 6-18), first, specify the replica's location in the SN; then, they calculate the job's access time for every available replica and choose the best one. At the end, the data access time will be calculated on the selected CN with the optimal replica selection (as shown in equation 3.15).
- (Lines 19-20) only calculate the computing time for all jobs on the chosen CN as described in equations 3.20 and 3.21.
- (Lines 21-26) calculate the saving time when using the overlap technique.
- (Lines 27-35) calculate the completion time by using the overlap technique for the selected VMs' jobs, which is the sum of the computing and data transferring time with the optimal replica (as shown in equation 3.23). Then it assigns all variables that are mentioned in lines 1 2 to the best values and sends them back to algorithm 6.1.

Algorithm 6.2 Calculates Completion Time using data transferring time

Input: vm(jobs), vm(Data),StorageList, R
Output: calculates processing time and waiting time CT

```
1: MinTCT ← MIN ; DTT ← MIN ; Best_r ← MIN
2: Chosen_dc ← NULL ; Chosen_c ← NULL
3: for dc = 1 to Dc do
4:   if N ≠ 0 then
5:     for r = 1 to R do
6:       for s = 1 to S do
7:         for c = 1 to C do
8:           if Load(v) ≤ Cap(c) then
9:             for j =1 to J do
10:              JT = r / BW(s, c)
11:              PT = Inst(j)/(cap(c) × core(j))
12:              CT = St(j) + PT
13:              if j > 1 then
14:                DTT = JT - PT(j-1)
15:              else
16:                DTT = JT
17:              end if
18:            end for
19:            if Olap < DTT then
20:              Olap ← DTT ; Best_r ← r
21:            end if
22:            TCT = Olap + CT
23:            if TCT < MinTCT then
24:              MinTCT ← TCT; Chosen_c =c ; Chosen_dc =dc
25:            end if
26:          end if
27:        end for
28:      end for
29:    end for
30:  end if
31: end for
32: return MinTCT,Chosen_dc, Chosen_c , Best_r
```

Correspondingly, algorithm 6.3 checks the possibility of data replicated to DCs that do not store any replicas. Then, it calculates the replication service cost, which includes storage writing time, saving time using the overlap technique and the total completion time of the job. Subsequently, it chooses the best CN, which minimises the completion time. At the end , it will send the best information concerning the resources, including the CN, SN, DC and completion time, to algorithm 6.1, which decides to use replication or data transfer (remotely or locally) based on the optimisation decision (as mentioned

previously).

Algorithm 6.3 Calculates Completion Time using data Replication

Input: vm(jobs), vm(Data),StorageList, uR
Output: calculates processing time and waiting time CT

```
1: MinComTime ← MIN
2: MinST ← MIN
3: ChosenDc ← NULL
4: ChosenCn ← NULL
5: for DC in DcList do
6:   if R ≠ True then
7:     for Cn in CnList do
8:       if Load(vm) ≤ Cap(Cn) then
9:         for job in JobList do
10:          for S in StorageList do
11:            ST = data(job)/ WD(S, DC)
12:            if ST < MinST then
13:              MinST ← ST
14:              ChosenS = S
15:            end if
16:          end for
17:          PCap = Inst(job)/(cap(Cn) × core(job))
18:          ComputingTime = St(job) + PCap
19:          RT = data_(job) / BW(ChosenS, Cn)
20:          if job > 1 then
21:            StartOvelap = starttime(job -1) + RS
22:          end if
23:          JobsComT = StartOvelap + ComputingTime
24:        end for
25:        if JobsComT < MinComTime then
26:          MinComTime ← JobsComT
27:          ChosenCn =Cn
28:          ChosenDc =Dc
29:        end if
30:      end if
31:    end for
32:  end if
33: end for
34: return MinComTime,ChosenDc,ChosenCn,ChosenS
```

- **(Input):** will receive (as in algorithm 6.2) the application jobs' requirements , the storages list and DCs that do not store any replica for the required data and the replica source.

- **(Output):** will send the chosen DC, CN and SN, which are considered to be the

optimal and the total completion times for all jobs.

- (Line 1) has some variables that are decided; the optimal storage chosen s will store the new data replica and the cost of replica time MinTVM . Second, it chooses the optimal CN chosen c that has the minimal completion time MinComTime and is executed in the certain datacentre chosen dc.

- (Lines 2-3) only check each DC that does not have a replica of the data. Otherwise, it will not execute the following codes, if there is at least one replica in the selected DC.

- (Lines 4-5) make sure that the selected CN has at least the minimum capacities to place the chosen VM.

- (Lines 5-14), first, check that each DC that does not have any copy of data how , the costly of replication, including the available bandwidth between the data source and the new SN chosen s and data writing time in every SN as mentioned in equations 3.13 and 3.14. Then, it returns the best SN chosen s and the optimal time for replication MinTVM.

- (Lines 15-17) calculate the intra-data transfers between the new SN that has the new replica and the chosen CN.

- (Lines 18-19) are a set of calculations that compute the queuing, processing and data access times for every job on the chosen CN (as described in equations 3.15 and 3.20).

- (Lines 20-24) are calculating the saving time when using the overlap technique.

- (Lines 25-33): Now, it is time to calculate the completion time for all jobs on each of the CNs as discussed in equation 3.24. The completion time consists of the sum of computing and data transferring time using the prefetching technique and replication cost. Then, it will compare these times to select the best DCs' resources, including CN and SN.

- (Line 34) assigns all variables that are mentioned in lines 1 2 to the best values and sends them back to algorithm 6.1. These variables are the optimal CN, the minimum time for replication and communication, and the best SN that will store the data for replication.

6.3 Simulation results

In this section, the simulation aimed at evaluating the proposed adaptive Cloud Placement algorithm are presented. CloudSim's [37] discrete-event cloud simulation was used to model the cloud environment. Later in the section, both the simulation set up and the workloads used for the evaluation are described.

6.3.1 Simulation Set Up

The simulated model was composed of one cloud DC that contained 10 CNs. Each CN had two quad-core processors and 16 GB of RAM. The CN hosted user application instances and the mechanism . The DC had one broker that received users' requests and placed the VMs on the appropriate CNs based on the allocation algorithm that was used. There were 25 VMs that required one core and 2 GB of RAM. The total number of files was 25, which were distributed over the SNs in the DCs, and the size of each file was between 0.1 MB and 3.0 GB (generated randomly). Every VM had a different number of jobs (e.g., 50 jobs gradually increasing to 100 ,..., 250 jobs). Every job needed to access certain data 1 3 files, which were stored on an SN (chosen randomly). Table 6.1 shows all the parameters that were used. This test used the second scenario, which was discussed in section 3.2 of chapter 3.

Parameters	**Values**
Number of CNs	10
Number of VMs	25
Number of jobs	50 -250
Number of files	1-3
File size / job	[100 - 3000]MB

Table 6.1: Parameter Settings for the First Test in the NADIv3 Scheduler (2nd Scenario)

6.3.2 Simulation Evaluation

As shown in Table 6.1, the second scenario used the same parameters as scenarios described in the previous two chapters. The overlap technique (i.e., prefetching data while jobs queue) is also used in the SPE algorithm, although it neglects CPU load and wait times. Additionally, these factors are not taken into account during VM allocation, which could increase computing time. Figure 6.2 shows the total computing time for the jobs, which increase in number from 50 to 250. SPE [111], NADIv2 were also discussed [122], and NADIv3 was the algorithm used for the comparison [147] because it minimises computing time in the majority of cases by prefetching data during job queuing and considering CPU loading times. Improvement was observed in computing time when comparing NADIv3 to NADIv2.

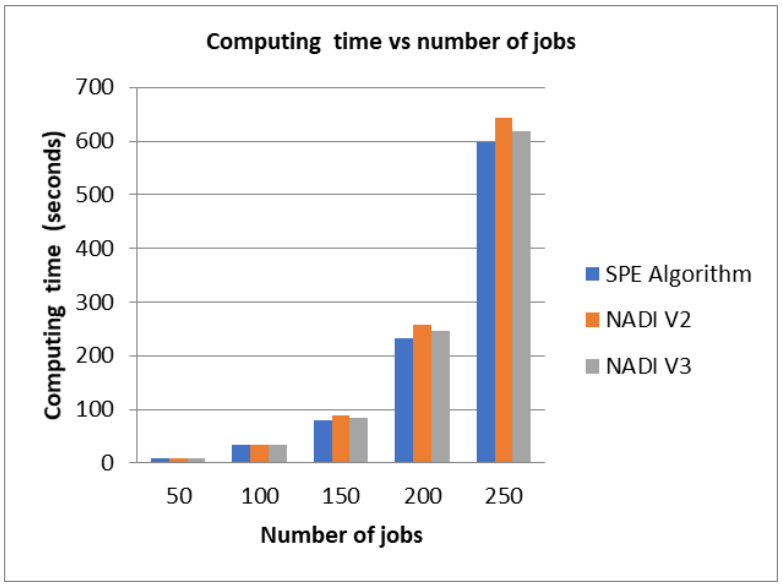


Figure 6.2: The computing time for all jobs

Figure 6.3 illustrates data transfer time for all jobs. NADIv3 does not enhance this time significantly because it is the final version of NADIv2, which considers data transfer time as discussed in Chapter 5. NADIv3 is differentiated from NADIv2 by its use of the overlap technique, among other factors . The SPE approach provides the slowest data transfer time, although it prefetches data to enhance application performance. Poor placement decisions can lead to VMs being allocated in instances of high

network latency or heavy CPU loads, which affects overall application performance. The amount of VMs placed on CNs is a poor allocation policy . Figure 6.4 shows the total completion times for the three compared approaches. The new technique reduced overall completion time and enhanced application performance. The overlap technique alone cannot guarantee the enhancement of the application performance because other factors degrade performance, such as CPU load, available bandwidth, the number of waiting jobs in CPU queues and data location.

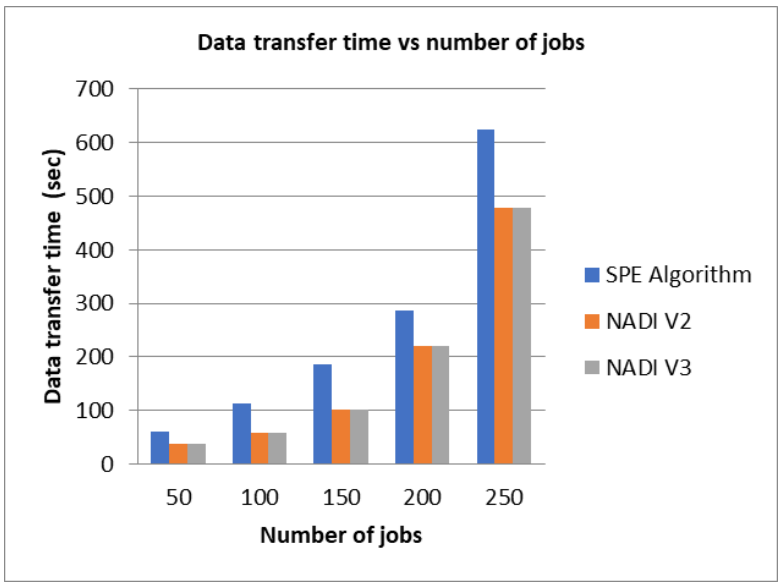


Figure 6.3: The Data transferring time for all Jobs

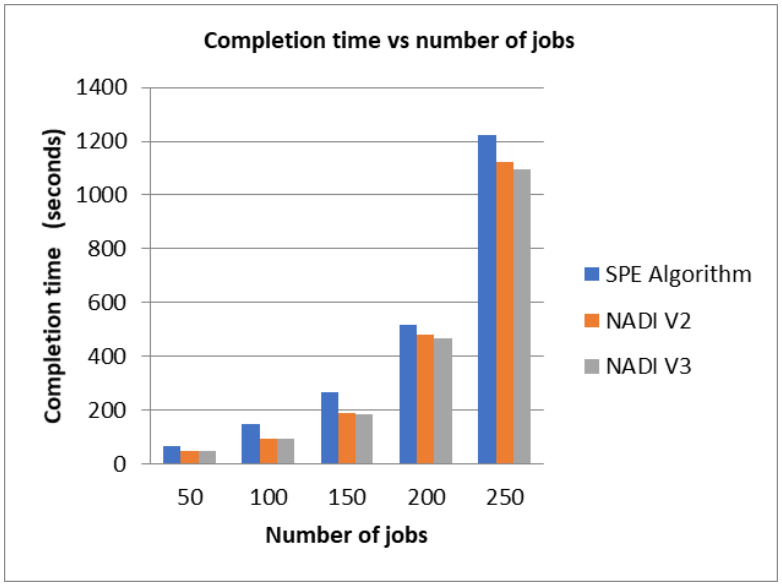


Figure 6.4: The completion time for all Jobs

Figure 6.5 illustrates the resources used during the investigation, including the CPU usage, bandwidth and memory and VM allocation statistics for the CNs. Figure 6.5.a shows that CPU usage for the SPE approach was higher for the first five CNs because the allocation policy placed more than one VM on one CN, which increased CPU usage . Both NADIv2 and NADIv3 allocated VMs on optimal CNs with smaller CPU loads. Figure 6.5.b shows the bandwidth usage for all three approaches; NADIv2 and NADIv3 exhibited the best results, although there was some fluctuation in the rise and decline values due to data fetching for both current and queued jobs. The SPE algorithm had the highest bandwidth usage because available bandwidth was shared among the VMs.

Figure 6.5.c shows memory usage, revealing that the SPE approach used the most memory, which can degrade application performance. In some cases, slight increases and decreases in the memory usage of NADIv3 and NADIv2 resulted from the overlap technique, which also fetched data for current and queued jobs. This increased RAM usage during job execution and decreased RAM usage when job processing was complete. There was a correlation between CPU and bandwidth usage increases and decreases and memory usage. Figure 6.6.d shows the number of used CNs allocated to VMs: the SPE algorithm used five CNs and left the remaining five CNs in an ideal mode. This

technique minimised energy consumption while increasing the rate of VM allocation on one CN. As usage increased, application performance began to degrade. Figure 6.6 shows the total resources used, and Figure 6.4 illustrates total completion time. Thus, NADIv3 minimises overall completion time and resource usage and enhances applications performance.

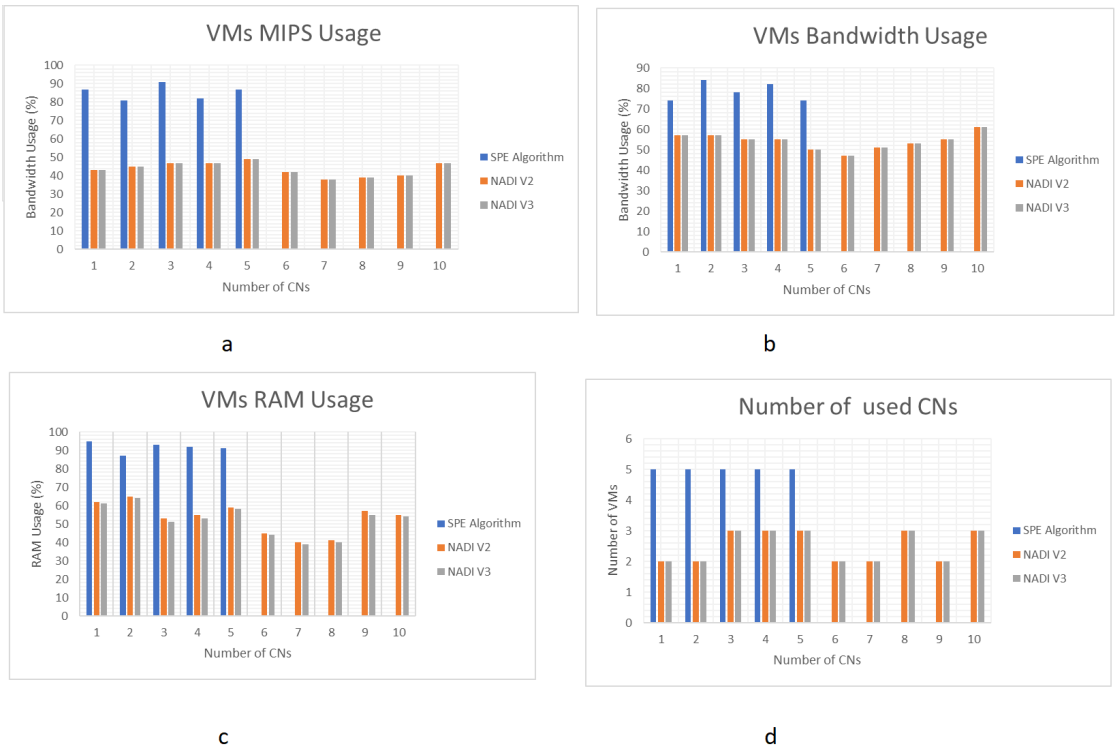


Figure 6.5: Comparison of the CNs overall resource usage

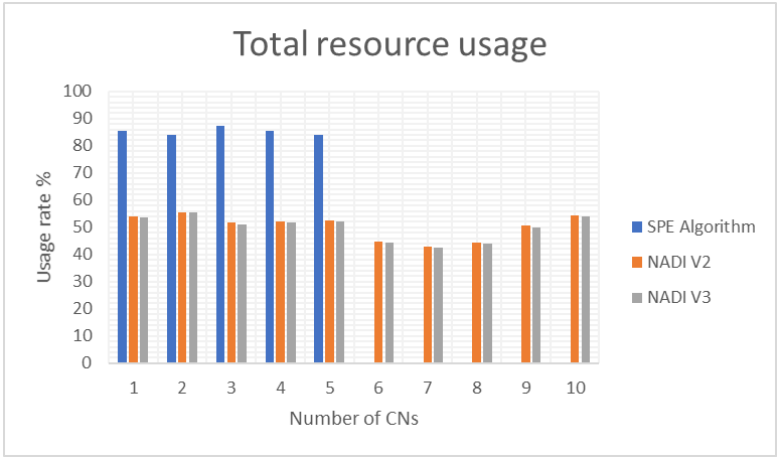


Figure 6.6: The overall resource utilization rate

In the following graphs, the scenario discussed in Chapter 3, Section 3.2 is depicted. This scenario used a large-scale and complex evaluation that increased the total number of VMs per user, with each VM requesting a number of files. There were three DCs, with each DC having 350 CNs. The simulation ran for 10 minutes, during which time requests for VM allocation were continuous, and VMs increased from 150 to 1500. Each VM requested 35 files spread over the distributed SNs. The jobs also increased gradually, from 50 to 1500 in each round, as shown in Table 6.2. Again, the SPE, NADIv2 and NADIv3 algorithms were compared.

Parameters	**Values**
Number of DCs	3
Number of Users	50-1500
Number of Jobs	50 - 250
Number of files	3- 5
File Size / Job	[1, 30] GB

Table 6.2: Parameter settings for the second test in NADIv2 Scheduler (3rd scenario)

Figure 6.7 compares job lifecycle times from the execution stage to completion in scenario two. Figure 6.7.a shows that NADIv2 and NADIv3 have the same results for data transfer time because the algorithm is the same; however, NADIv3 outperforms NADIv2 because it uses the overlap technique to prefetch data and minimise execution and waiting times for jobs. The SPE algorithm increased the burden of the network by increasing the maximum data rate and reducing available bandwidth, leading to a network bottleneck.

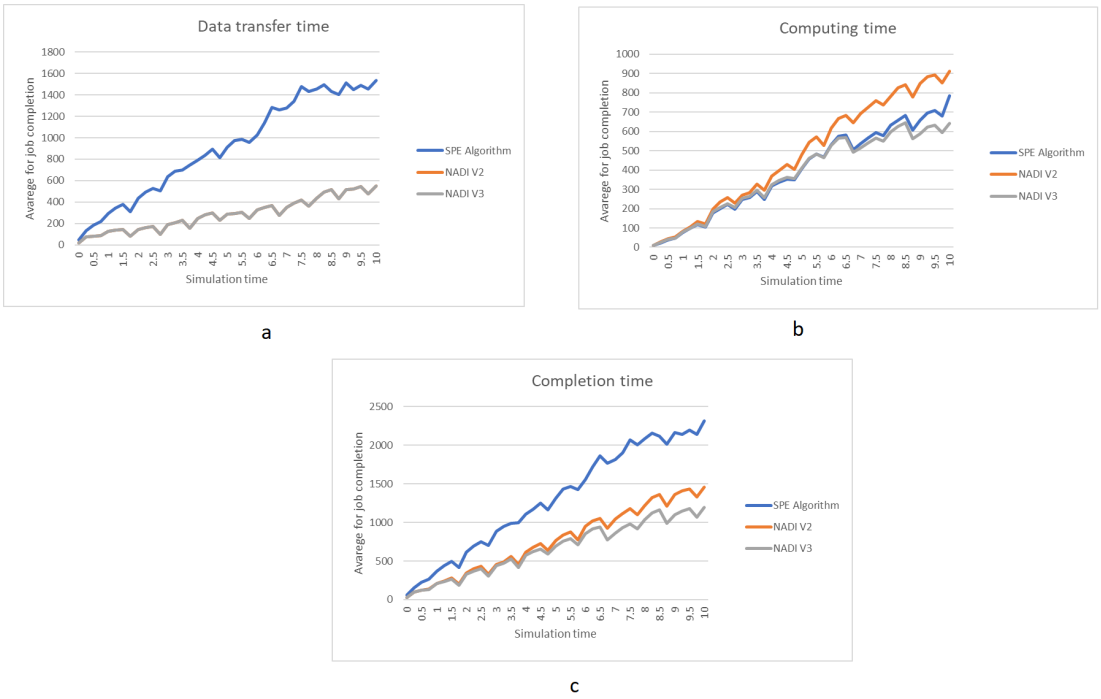


Figure 6.7: The completion time for all jobs

Figure 6.7.b shows that prefetching data enhances computing and job completion times, when using NADIv3. The SPE algorithm also used the overlap technique; however, slight increases in simulation result times revealed that insufficient bandwidth for data transmission of links created heavy data transmission between the application and the SNs. Thus, the required data arrived late, delaying job execution. Minimising both data transfer and computing times reduces job completion time to increase application performance (6.7.c). Whenever jobs increase, so does data size, the number of required files and the workload, and in this case, NADIv3 provides better results and enhances application performance.

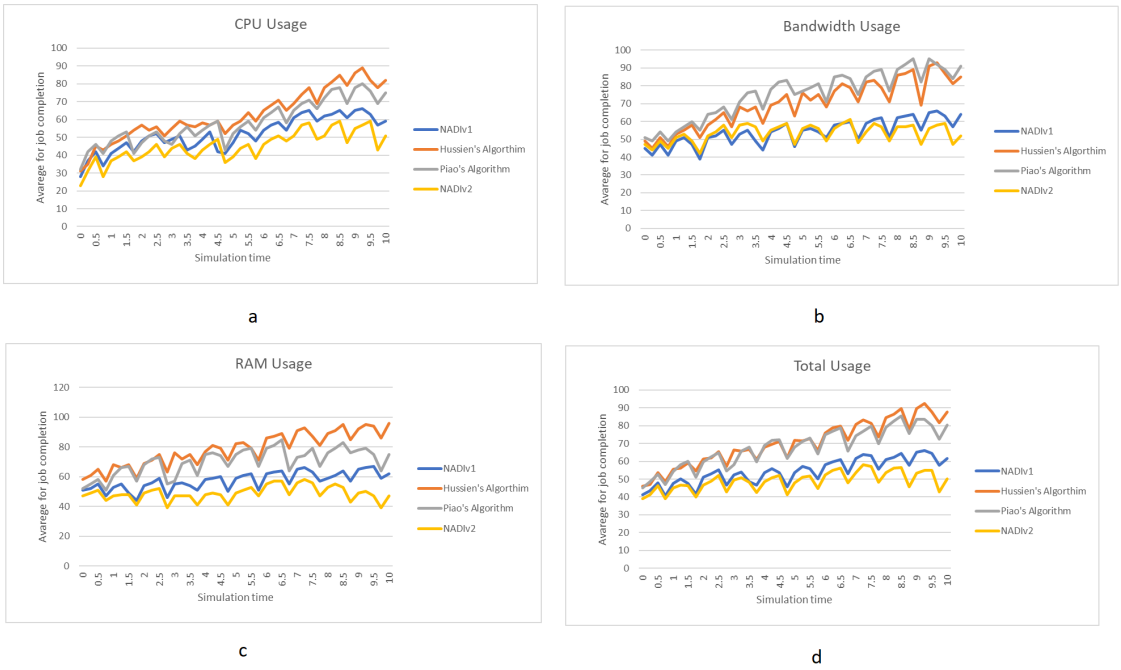


Figure 6.8: Comparison of the CNs overall resource usage

The same test was used to evaluate resource usage (Figure 6.8). Figure 6.8.a shows a slight decrease in CPU usage for NADIv3 compared to NADIv2 because of the overlap method that stages I/O data before job processing, minimising wasted time for queued jobs. A significant negative difference was found between NADIv2 and NADIv3 and SPE, despite the fact that the SPE algorithm uses the same technique. This difference is due to poor bandwidth conditions that cause huge amounts of data to transfer between CNs and SNs, which increases wait times for jobs. The scheduler should take the network state into account because network status is changeable and can affect the job execution cycle dramatically. Figure (6.8.b) shows that ignoring network conditions leads to long data transfer times and creates network bottleneck. NADIv3 created slight fluctuations in values compared to the other algorithms because the overlap technique fetched data for both current and queued jobs.

Figure 6.8.c shows memory usage, and SPE required the most memory, which degraded application performance because of the number of VMs allocated on one CN. These VMs shared the same CN resources (i.e., CPU, RAM, bandwidth and local disks; Figure 6.9), and the VM allocation algorithm did not take into account available band-

width and data location for CPU loads and speed. In contrast, the NADI algorithms minimised memory usage in addition to CPU and bandwidth usage because the VM allocation algorithm considered these factors. Memory usage was reduced by correlating CPU and bandwidth usage to memory usage. Memory usage for NADIv3 was affected instantly due to the overlap technique, which is similar to the results obtained for bandwidth usage (Figure 6.8.b). Such fluctuations show that the NADI algorithms are optimal but depend on the period of time . Thus, resources usage improves sequentially because of enhanced CPU, bandwidth and memory usage (Figure 6.8.d), and NADIv3 enhances application performance and minimise completion time by load balancing DCs and maximising workload throughput (Figures 6.7 - 6.8).

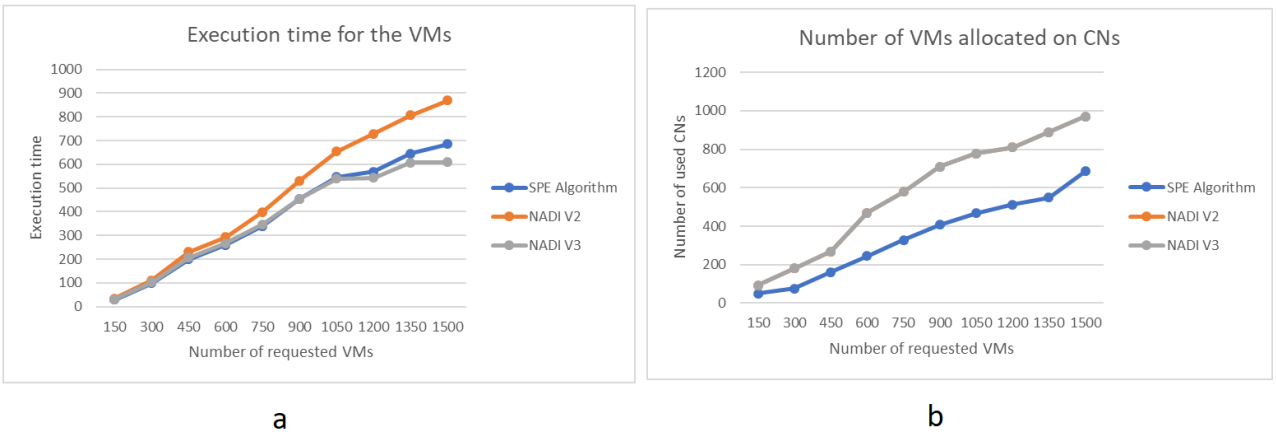


Figure 6.9: Number of VMs Allocated on CNs

The previous test monitored both the number of allocated VMs on the CNs and the total execution time for all VMs in each round (Figure 6.9). Figure 6.9.b shows a comparison of the execution times of these algorithms and shows that NADIv3 minimises total execution time and provides optimal results. Because VMs host data-intensive applications, data transferred between VMs during execution takes time depending on the distance between the VMs and size of the data that is transferred. Figure 6.9.a shows no difference between NADIv3 and NADIv2 VM allocation because the same placement algorithm is used to minimise completion times and enhance application performance. The primary difference is the computing time, which is improved when data are fetched before jobs are processed. Available bandwidth and data location for

the CPU loads are not accounted for when using the SPE approach, which degrades application performance because unequal amounts of VMs are allocated on one CN and share the same resources. NADIv3 improves application performance by reducing execution time by 27%, by decreasing CN usage and by minimising total execution time (Figure 6.9). However, it increases the total number of CNs allocated to VMs, which increases power consumption in DCs.

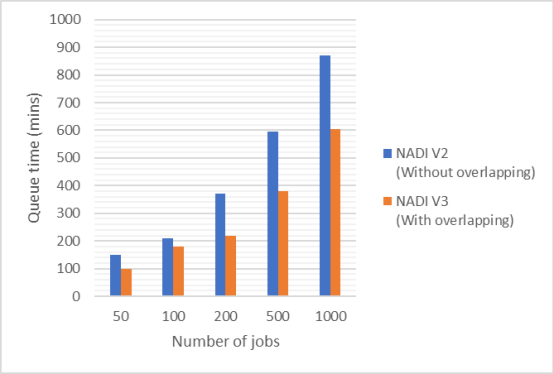


Figure 6.10: Queuing time versus Number of jobs

Execution time is usually longer than processing time because the CPU may be handling other tasks apart from running a process, such as user and operating system tasks or network and disk input or output. Execution time does not include queuing time. Figures 6.10 and 6.11 show queuing and execution times for workload management using NADIv2 and NADIv3, which are compared using an overlapping algorithm scheduler. In this simulation, the same number of jobs (50) was submitted to both algorithms, and the queuing and execution times were calculated; then, job number was increased to 1000 to monitor how queue size increases over time and in what proportion the scheduler submits jobs to determine whether jobs are submitted to a specific site or to a number of CPUs at different locations, depending on the queue size and the computing capability. The queue times were plotted to investigate how this time increased and decreased with the number of jobs. The results of this test are presented in Figures 6.10 and 6.11, which show that both queue and execution times follow similar trends. The overlapping algorithm improves job execution time because it selects only those CNs for job execution that contain the required data and have smaller loads and fewer

queued jobs.

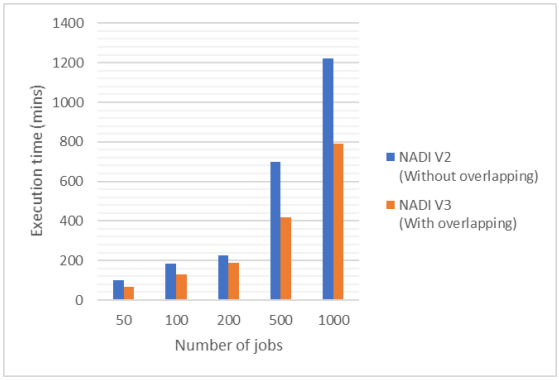


Figure 6.11: Execution time versus Number of jobs

6.4 Chapter Summary

Data-intensive applications are becoming more common but require larger volumes of data to run efficiently. Intensive computation approaches, such as conventional meta-scheduling, do not consider application data requirements, which leads to poor performance. A major challenge in cloud environments is scheduling application data in an efficient manner, and the average turnaround time and processing of jobs should be considered in such environments. In this chapter, a NADI was proposed, with a final version that considers both computational and data requirements for VM allocation decisions. Additionally, NADIv3 overlaps data staging and job execution and transfers data during job queuing as other jobs are being executed. This technique improves the average job turnaround time significantly. This version is final because it considers data location, computation power and load, replication service, available bandwidth and prefetched data. This approach was designed using data management service components that manage, track and transfer data between source and execution nodes. The algorithms were explained in depth, alongside the results from the simulation test, which revealed considerably improved job performance and minimised job completion times. In the next chapter, the contributions of this thesis are summarised, the accomplishments of the thesis are presented, and research issues for future investigations are highlighted.

7

Conclusion

This chapter is divided into two sections. Section 7.1 summarizes the whole thesis and presents a brief review of the issues and approaches taken throughout the text. Section 7.2 discusses the future direction of the work that emerged during this research or that is related to issues that were relevant but outside the scope of this research.

7.1 Summary of the thesis

Cloud computing offers the opportunity to share knowledge, information and DC resources among Cloud users. In cloud environments, the use of data-intensive applications is becoming more common. The applications require data retrieval from distributed storages, and in the meantime, application performance entirely could be influenced by many factors such as the locality of data, CPUs' attributes and loads and the available bandwidth between SNs and CNs with an unstable network status. Data intensive scheduling can be significantly improved by taking into consideration a combination of network, data and compute costs. This thesis takes into account these considerations during VMs allocation to enhance data-intensive applications' and job performance and to minimize total completion time.

Chapter 1 presented the motivation and objectives of the research. It described the organization of the remaining chapters and highlighted the methodology adopted for the research.

Chapter 2 introduced cloud computing and various enabling technologies, including networks and virtualisation. Current trends in the design of cloud DCs architecture were explored in more detail. Additionally, Chapter 2 presented the facilities to develop cloud service by using modelling and simulation. It also identified research efforts in VMs placement and migration in cloud management. The chapter concluded with a summary of research efforts in the area of cloud computing and identified the specific areas that the research focused on.

Chapter 3 outlined the mathematical and theoretical explanation of the NADI scheduling framework and described the various components, actors, and roles in the framework. It has been demonstrated that with aid of mathematical equations, a matrix of various scheduling costs can substantially enhance the process of scheduling when every task is submitted and executed after considering particular related costs. In addition, this chapter offered a general overview of requirements of NADI scheduling and named the salient characteristics of such a system. Of those characteristics, it was noted

that location of the data is the most important consideration to the working of optimum cloud. Therefore, it should to be one consideration affecting scheduling decisions. It was also concluded that the best network path to storage and computing should be acknowledged. Then the scheduler should incorporate and calculate measurements of the network when planning to submit jobs.

Chapter 4 proposed the first version of a network-aware VM placement algorithm for data-intensive applications in a cloud environment (NADI). Then it showed the relationships between NADI design components, including the broker, data location service, network monitoring service, information service and discovery service. The algorithms used in this section were also explored in greater detail, and simulation tests were performed to examine the NADI algorithm. Through these tests, it was demonstrated that data location, network latency and computational resources can significantly affect application performance. Thus, the VM allocation scheduler should be capable of optimising the data-intensive scheduling process. It is necessary to control the location of VMs so the applications hosted by a VM can obtain a shorter data access time. It was also demonstrated that the keys to cloud optimisation include network-managed services and a suitable selection of the network links between DC locations before making scheduling decisions. Also, overall queue and execution times can be significantly reduced if job data requirements and completion times are taken into account. It was also concluded from the simulation results on CloudSim 3.0 that NADI is better in scalability and consistency compared to other contemporary scheduling approaches.

Chapter 5 described the second version of NADI with consideration of computation power and its load, less network traffic and replica locations. This approach obtains an optimized completion time of jobs in each VM. The NADI scheduling optimization approach has been developed to generate the time cost matrix for an overall job completion time and optimized among these values. Then, it chooses the optimal site to allocate the VMs on that costs least in time . If there are any enhancements for job performance by using replication service, NADI makes the decision to replicate the required

replica to the chosen location. Otherwise, it will optimise the best one. Additionally, the changeable component in NADI was in the data management service (DMS) that deals with the data and the replicas information. The main role of DMS is to manage replicas, generate new replicas if needed and give the access and location for these replicas. The algorithms used in this chapter have been explained in depth. The results show a significant improvement on the average completion time by adopting the NADI scheduling approach.

Chapter 6 proposed the final version of NADI scheduling that takes into consideration both the computation requirements and the data requirements to make the VM allocation decisions. Additionally, NADIv3 overlaps between staging the data and execution for the jobs. Consequently, it transfers the data in advance for a job with its own queuing time and at the same time on other jobs are executing. This technique improves the average job turnaround time significantly. This version is considered the final that considers the data location, computation power and load, replication service, available bandwidth and pre-fetching data advanced. This design has been changed slightly in term of the data management service components. These components manage, track and transfer data between the source and execution node. The algorithms used in this chapter have been explained in depth. Results obtained from the simulation experiment indicate considerable improvement of jobs performance. Compared with the previous approach, jobs execution has been enhanced. Sequentially, the overall completion time is improved significantly.

7.2 Future Directions

Cloud computing presents many challenges to system administrators, application developers, service providers and engineers [148–150]. In the following section, the challenges related to scheduling and management of workflows on clouds of data intensive applications are discussed.

7.2.1 Scheduling based on energy efficiency

DCs consume a lot of power, so they are very expensive to operate [151]. For example, the combined energy of worldwide DCs is equal to that of the Czech Republic [152]. The outcome is an increase in carbon footprints in their surroundings. To address this issue, there is a need to come up with an efficient allocation of energy resources and develop an effective algorithm. The challenges faced are :

- ways of balancing the performance and energy consumption of the DCs while making scheduling decisions; and
- ways of selecting DC locality to ensure that energy consumption, operation cost, and data security adhere to the terms in SLA as signed by the users.

DCs performance is based on the usage and provision of the hardware devices provided by the VM management software per the user's requirements. When the number of CPUs increases, it raises the hardware temperature, which calls for DC cooling. Therefore, the DC performance and energy consumption are directly interconnected. When the DC network equipment and commodity hardware prices get cheaper, the important overall operational cost of cloud services is based on the amount of energy that the DC has consumed. Adapting efficient energy scheduling policies can help the DC conserve energy and also save on cooling costs. Application schedulers can use the statistical information acquired by the sensors while submitting data and jobs to be computed across VMs.

As can be seen in Figures 4.10, 5.10 and 6.9, the number of CNs used in the NADI algorithm is the greatest amongst the compared algorithms. The main reason for this is to decrease CN' usage and to minimise the total execution time. Consequently, the jobs performance is enhanced. However, this also increases the total number of CNs hosting VMs, which increases the DC's energy consumption. Therefore, energy consumption will be implemented in the NADI algorithm as a future research direction. The next version of NADI will try to maintain a balance between energy consumption and enhancement

of the jobs performance. As it is known that these two objectives may conflict, only one of the optimisation algorithms will be used.

7.2.2 Management of VM images and data as workflows

Virtualization enhances servers' consolidation to host the services in a multi-tenancy way on autonomous virtual machines. When the number of VMs created is huge, they need to be managed to ensure that users receive high-quality services. To ensure the expected QoS is delivered, VMs are migrated to appropriate servers, and later a dynamic consolidation is done to the physical servers. Migration is concerned with moving huge data across the servers, which can overload the system, especially when migration is forced or occurs regularly in a huge DC. Migration of the jobs can be illustrated as a workflow to maintain the QoS of the user and to prioritize and maintain VMs transition without interfering with the service. These capabilities raise a challenging question.

- How do the service providers handle a huge amount of data and VMs migration?

It is common practice for the manager not to reveal the storage or computing resources contained in the service provider to the clients. A client, therefore, may select a certain provider based on advertisements and reputation. If the service receives a huge number of requests, it is forced to overload the system to meet them. Managing the load in infrastructure and a large number of requests becomes the main challenge [153]. To address this scenario, service providers migrate, scale out and replicate the VMs [154] to underutilized resources. Scheduling and representing this complicated process as a workflow become challenging since instantiation of huge VM numbers in DC introduces hardware and software barriers [155].

Basically, NADI has focussed on VM placement to improve applications and CN resources such as memory, CPU and bandwidth. As network status is changeable, which may significantly influence the jobs' performance, NADI only focusses on the initial placement of VMs, which does not consider how to manage a DC's network state. When the link between an application and its rerated data has been degraded,

NADI should consider migrating the affected VMs to a suitable CN that is guaranteed to enhance the jobs' performance.

7.2.3 Job monitoring and migration

The dynamic environment of the cloud system is prone to predictable changes such as performance degradation, network failure and changes in resource costs. In such a scenario, migrating the task remains the only effective way to ensure that the submitted tasks are accomplished and the constraints of the user are met. The major jobs that are migrated include check-pointing, task monitoring and rescheduling. According to Allen et al. [156] and Huendo et al. [157], systems that deal with task migration solve migration issues from the performance view. The major migration policies considered in these systems are the detection of best, resource, performance slow down, cancellation of jobs and system failures. However, Chauhan and Babar [158] present a system that can manage the migration of jobs under certain economic conditions. In this scenario, there should be consideration of new job migration policies such as discovering cheaper resources and price changes of when the job is being executed. Many different reasons can lead to resources modifying their prices dynamically; for example, a change of prices can occur as a result of demand. Additionally, price fluctuation can occur depending on the day or time. For example, prices can be relatively cheaper during weekends or nights. A task monitoring and submission web service does not offer network- aware scheduling , and it solely depends on other scheduling tools to come up with a scheduling decision. Additionally, it does not provide a policy based scheduling or data intensive scheduling algorithm [159].

The network condition is changeable, which may significantly influence jobs' performance. Thus, NADI should consider the jobs' migration first instead of migrating the VMs due to the small size of a job and the related data location. NADI's objective is to enhance the applications performance and to maximise the links throughput. The VMs' migration may place a larger burden on the network status and degrade the performance of the application.

7.2.4 Distributed parallel processing

The application allows several machines run actions simultaneously to minimize the running time by allocating a bunch of activities to each virtual machine. Cloud computing makes it simple to represent and release an expansive number of virtual machines over time due to the availability of virtual machines and a flexible model. The situation is propelled by the International Telecommunication Union use case in which the traditional client data request and evaluation frameworks require all information to be handled inside a common server. Thus, hardware size becomes the hindrance to efficient productivity, and the existing system takes a lot of time for some applications. In light of the constraints of the system, allowing data inquiry and data extraction tools simultaneously on the distributed processing framework is the best solution and accomplishes huge adaptability by utilizing a well-distributed record framework and fast handling in view concurrent loading, extraction, transformation and computing. The parallel processing system may require association among the different components since they must work concurrently [19].

In this thesis, NADI only used centralised processing for job execution. However, in a cloud computing framework, there are many sorts of applications, including web and distributed applications [10]. The web application is partitioned into web, information and application layers. Distributed applications involving online business or logical calculation are generally separated to different subtasks, and there are calculation practices and data transmission between one particular subtask and another. Correspondence ability and execution time of subtasks or application is influenced by communication rate, which is the main obstacle to multiple task execution in a cloud framework. There is the requirement to claim physical resources, primarily CPU and memory resources, so an application is typically partitioned into several subtasks that are transmitted to the computation hub on a substantial scale. Hence, communication rate among physical devices influences the time an application ends [10]. Therefore, it might be useful if NADI used distributed parallel processing to be compatible with the current cloud framework used in the next version.

References

[1] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *Proceedings of the First first international conference on cloud Computing cloudcom.* Springer Berlin Heidelberg, 1-4 December 2009, Conference Proceedings, pp. 626–631.

[2] B. Furht, *Cloud Computing Fundamentals.* Boston, MA: Springer US, 2010, pp. 3–19. [Online]. Available: https://doi.org/10.1007/978-1-4419-6524-0_1

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010. [Online]. Available: https://doi.org/10.1007/s13174-010-0007-6

[4] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012).* IEEE Computer Society, 2012, Conference Proceedings, pp. 498–506.

[5] M. Rifai, N. Huin, C. Caillouet, F. Giroire, J. Moulierac, D. L. Pacheco, and G. Urvoy-Keller, "Minnie: An sdn world with few compressed forwarding rules," *Computer Networks*, vol. 121, pp. 185 – 207, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617301433

[6] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Proceedings of the 2008 Workshop on Grid Computing Environ-*

ments (GCE'08). IEEE Computer Society, 12-16 November 2008, Conference Proceedings, pp. 1–10.

[7] A. Shawish and M. Salama, "Cloud computing: Paradigms and technologies," in *Inter-cooperative Collective Intelligence: Techniques and Applications*, F. Xhafa and N. Bessis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, ch. 2, pp. 39–67.

[8] L. Lu and E. Smirni, "Effective resource and workload management in data centers," in *Proceedings of the 2014 IEEE/Network Operations and Management Symposium (NOMS)*. IEEE Computer Society, 5-9 May 2014, Conference Proceedings, pp. 1–7.

[9] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, no. Part 3, pp. 849 – 861, 23 September 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17302224

[10] X. Fu, Y. Cang, X. Zhu, and S. Deng, "Scheduling method of data-intensive applications in cloud computing environments," *Mathematical Problems in Engineering*, vol. 2015, pp. 1 – 8, 29 March 2015. [Online]. Available: http://dx.doi.org/10.1155/2015/605439

[11] S. Venugopal and R. Buyya, "A set coverage-based mapping heuristic for scheduling distributed data-intensive applications on global grids," in *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, ser. GRID '06. Washington, DC, USA: IEEE Computer Society, 28 - 29 September 2006, pp. 238–245. [Online]. Available: https://doi.org/10.1109/ICGRID.2006.311021

[12] S. Pandey and R. Buyya, "Scheduling workflow applications based on multi-source parallel data retrieval in distributed computing networks," *The Computer Journal*, vol. 55, no. 11, pp. 1288–1308, 1 November 2012. [Online]. Available: http://dx.doi.org/10.1093/comjnl/bxr128

[13] C. Hanna *et al.*, "Gravitational-wave data analysis overview of compact binary searches using waveforms inspired by numerical relativity," *Classical and Quantum Gravity*, vol. 27, no. 11, p. 114003, 10 May 2010. [Online]. Available: http://stacks.iop.org/0264-9381/27/i=11/a=114003

[14] J. Lakhani and H. Bheda, "Scheduling technique of data intensive application workflows in cloud computing," in *Proceedings of the 2012 Nirma University International Conference on Engineering (NUiCONE)*. IEEE Computer Society, 6-8 December 2012, Conference Proceedings, pp. 1–5.

[15] T. Kosar, *Data Intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management*, ser. Advances in Systems Analysis, Software Engineering, and High Performance Computing:. Hershey, PA, USA: IGI Global, 2012. [Online]. Available: https://books.google.co.uk/books?id=fPsN9dwCI6IC

[16] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 25 December 2013. [Online]. Available: http://dx.doi.org/10.1002/wcm.1203

[17] I. Legrand, R. Voicu, C. Cirstoiu, C. Grigoras, L. Betev, and A. Costan, "Monitoring and control of large systems with monalisa," *Commun. ACM*, vol. 52, no. 9, pp. 49–55, Sep. 2009. [Online]. Available: http://doi.acm.org/10.1145/1562164.1562182

[18] J. Lin, A. Bahety, S. Konda, and S. Mahindrakar, "Low-latency, high-throughput access to static global resources within the hadoop framework," Technical Report HCIL-2009-01, University of Maryland,College Park, USA, Tech. Rep., January 5 2009.

[19] ITU, "Part 1: Introduction to the cloud ecosystem: Definitions taxonomies use cases and high level requirements," ITU-T FG Cloud TR (Focus Group on Cloud Computing), Geneva, Switzerland, Tech. Rep., February 2012.

[20] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "Nist cloud computing reference architecture," in *Proceedings of the 2011 IEEE World Congress on Services (SERVICES)*. IEEE Computer Society, 4-9 July 2011 2011, Conference Proceedings, pp. 594–596.

[21] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet computing*, vol. 13, no. 5, pp. 10 – 13, 09 September 2009.

[22] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing [recommendations of the national institute of standards and technology-special publication 800-145]," *Washington DC: NIST. Recuperado de http://csrc. nist. gov/publications/nistpubs/800-145/SP800-145. pdf*, vol. 53, no. 6, p. 50, September 2011.

[23] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proceedings of the 2008 Grid Computing Environments Workshop, 2008. GCE'08*. IEEE Computer Society, 12-16 November 2008, Conference Proceedings, pp. 1–10.

[24] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer, "Cloud computing–a classification, business models, and research directions," *Business & Information Systems Engineering*, vol. 1, no. 5, pp. 391–399, 24 September 2009.

[25] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 8 April 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128609003387

[26] M. Parashar and S. Hariri, "Autonomic computing: An overview," in *Unconventional Programming Paradigms: International Workshop UPP 2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers*, J.-P. Banâtre, P. Fradet, J.-L. Giavitto, and O. Michel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 257–269. [Online]. Available: https://doi.org/10.1007/11527800_20

[27] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, December 2003. [Online]. Available: http://doi.acm.org/10.1145/1165389.945462

[28] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in *Proceedings of the Linux symposium*, vol. 1. Citeseer, 27th–30th June 2007, Conference Proceedings, pp. 225–230. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.488.2278&rep=rep1&type=pdf

[29] B. Walters, "Vmware virtual platform," *Linux J.*, vol. 1999, no. 63es, July 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=327906.327912

[30] J. Hurwitz, R. Bloor, M. Kaufman, and F. Halper, *Cloud computing for dummies.* 111 River Street Hoboken, NJ 07030-5774: John Wiley & Sons, Inc., 2012, vol. 1.

[31] T. Velte, A. Velte, and R. Elsenpeter, *Cloud computing, a practical approach.* McGraw-Hill, Inc. New York, 2010.

[32] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the 2010 IEEE INFOCOM,*. IEEE Computer Society, 14-19 March 2010, Conference Proceedings, pp. 1–9.

[33] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center

network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, 16 - 21 August 2009. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592576

[34] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 17 - 22 August 2008, Conference Proceedings, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402967

[35] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 75–86, 17 - 22 August 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402968

[36] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, 16 - 21 August 2009. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592577

[37] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *Proceedings of the 2009 International Conference on High Performance Computing & Simulation (HPCS'09)*. IEEE Computer Society, 21-24 June 2009, Conference Proceedings, pp. 1–11.

[38] R. Buyya and M. Murshed, "Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, November 2002. [Online]. Available: http://dx.doi.org/10.1002/cpe.710

[39] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of*

Supercomputing, vol. 62, no. 3, pp. 1263–1283, December 2012. [Online]. Available: https://doi.org/10.1007/s11227-010-0504-1

[40] M. A. F. Gutierrez and N. Ventura, "Mobile cloud computing based on service oriented architecture: Embracing network as a service for 3rd party application service providers," in *Proceedings of the ITU / Kaleidoscope 2011: The Fully Networked Human-Innovations for Future Networks and Services (K-2011)*. IEEE Computer Society, 12-14 December 2011, Conference Proceedings, pp. 1–7.

[41] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, "Modeling virtual machine performance: Challenges and approaches," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 3, pp. 55–60, December 2010. [Online]. Available: http://doi.acm.org/10.1145/1710115.1710126

[42] J. L. L. Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Dynamic placement of virtual machines for cost optimization in multi-cloud environments," in *Proceedings of the 2011 International Conference on High Performance Computing and Simulation (HPCS)*. IEEE Computer Society, 4-8 July 2011, Conference Proceedings, pp. 1–7.

[43] W. Li, J. Tordsson, and E. Elmroth, "Virtual machine placement for predictable and time-constrained peak loads," in *Proceedings of the 2011 8th International Workshop, GECON, Paphos, Cyprus, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 5 December 2011, Conference Proceedings. [Online]. Available: https://doi.org/10.1007/978-3-642-28675-9_9

[44] B. Gerofi, H. Fujita, and Y. Ishikawa, "An efficient process live migration mechanism for load balanced distributed virtual environments," in *Proceedings of the 2010 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE Computer Society, 20-24 September 2010, Conference Proceedings, pp. 197–206.

[45] A. Strunk, "Costs of virtual machine live migration: A survey," in *Proceedings of the 2012 IEEE Eighth World Congress on Services (SERVICES)*. IEEE Computer Society, 24-29 June 2012, Conference Proceedings, pp. 323–329.

[46] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in *Proceedings of the 2009 IEEE International Conference on Cloud Computing. (CLOUD'09)*. IEEE Computer Society, 21-25 September 2009, Conference Proceedings, pp. 17–24.

[47] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective vm sizing in virtualized data centers," in *Proceedings of the 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE Computer Society, 23-27 May 2011, Conference Proceedings, pp. 594–601.

[48] D. Huang, D. Yang, H. Zhang, and L. Wu, "Energy-aware virtual machine placement in data centers," in *Proceedings of the 2012 IEEE / Global Communications Conference (GLOBECOM)*. IEEE Computer Society, 3-7 December 2012, Conference Proceedings, pp. 3243–3249.

[49] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, December 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[50] D. Kliazovich, P. Bouvry, and S. U. Khan, "Dens: data center energy-efficient network-aware scheduling," *Cluster Computing*, vol. 16, no. 1, pp. 65–75, 01 March 2013. [Online]. Available: https://doi.org/10.1007/s10586-011-0177-4

[51] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif *et al.*, "Black-box and gray-box strategies for virtual machine migration." in *Proceedings of the 2007 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 7. IEEE Computer Society, 11–13 April 2007, Conference Proceedings, pp.

229–242. [Online]. Available: https://www.usenix.org/legacy/event/nsdi07/tech/full_papers/wood/wood_html/

[52] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '09. New York, NY, USA: ACM, 04 - 06 November 2009, Conference Proceedings, pp. 202–208. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644918

[53] F. Lopez-Pires and B. Barán, "Virtual machine placement literature review," Polytechnic School National University of Asuncion, Tech. Rep., 4 Jun 2015. [Online]. Available: http://arxiv.org/abs/1506.01509

[54] L. Zhu, R. Tang, Y. Tao, M. Ren, and L. Xue, "Multi-objective ant colony optimization algorithm based on load balance," in *Proceedings of the 2016 International Conference on Cloud Computing and Security(ICCCS 2016)*. Springer International Publishing, 29-31 July 2016, Conference Proceedings, pp. 193–205. [Online]. Available: https://doi.org/10.1007/978-3-319-48671-0_18

[55] S. Georgiou, K. Tsakalozos, and A. Delis, "Exploiting network-topology awareness for vm placement in iaas clouds," in *Proceedings of the 2013 Third International Conference on Cloud and Green Computing (CGC)*. IEEE Computer Society, 30 September - 2 October 2013, Conference Proceedings, pp. 151–158.

[56] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: A scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 16 - 21 August 2009. [Online]. Available: http://doi.acm.org/10.1145/1594977.1592575

[57] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proceedings of the 2008 Third Workshop*

on Workflows in Support of Large-Scale Science, 2008. (WORKS2008). IEEE Computer Society, 17-17 November 2008, Conference Proceedings, pp. 1–10.

[58] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European journal of operational research*, vol. 176, no. 2, pp. 657–690, 16 January 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221705008337

[59] H. Saran and V. V. Vazirani, "Finding k cuts within twice the optimal," *SIAM Journal on Computing*, vol. 24, no. 1, pp. 101–108, 28 May 1995. [Online]. Available: https://doi.org/10.1137/S0097539792251730

[60] S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, and X. Fu, "Reliable virtual machine placement and routing in clouds," *IEEE Transactions on Parallel and Distributed Systems*, pp. 2965 – 2978, 12 April 2017. [Online]. Available: https://doi.org/10.1109/TPDS.2017.2693273

[61] D. Huang, Y. Gao, F. Song, D. Yang, and H. Zhang, "Multi-objective virtual machine migration in virtualized data center environments," in *Proceedings of the 2013 IEEE International Conference on Communications (ICC).* IEEE Computer Society, 9-13 June 2013, Conference Proceedings, pp. 3699–3704.

[62] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proceedings of the 2011 IEEE INFOCOM.* IEEE Computer Society, 10-15 April 2011, Conference Proceedings, pp. 66–70.

[63] B. Zhang, Z. Qian, W. Huang, X. Li, and S. Lu, "Minimizing communication traffic in data centers with power-aware vm placement," in *Proceedings of the 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS).* IEEE Computer Society, 4-6 July 2012, Conference Proceedings, pp. 280–285.

[64] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 09 May 2005. [Online]. Available: https://doi.org/10.1109/TNN.2005.845141

[65] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster computing*, vol. 12, no. 1, pp. 1–15, 01 March 2009. [Online]. Available: https://doi.org/10.1007/s10586-008-0070-y

[66] S. Agrawal, S. K. Bose, and S. Sundarrajan, "Grouping genetic algorithm for solving the serverconsolidation problem with conflicts," in *Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, ser. GEC '09. New York, NY, USA: ACM, 12 - 14 June 2009, Conference Proceedings, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1543834.1543836

[67] R. Gupta, S. K. Bose, S. Sundarrajan, M. Chebiyam, and A. Chakrabarti, "A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints," in *Proceedings of the 2008 IEEE International Conference on Services Computing (SCC'08)*, vol. 2. IEEE Computer Society, 7-11 July 2008, Conference Proceedings, pp. 39–46.

[68] I. Takouna, R. Rojas-Cessa, K. Sachs, and C. Meinel, "Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, ser. UCC '13. Washington, DC, USA: IEEE Computer Society, 09 - 12 December 2013, Conference Proceedings, pp. 251–255. [Online]. Available: http://dx.doi.org/10.1109/UCC.2013.50

[69] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, January 2011. [Online]. Available: http://dx.doi.org/10.1002/spe.995

[70] I. Takouna, W. Dawoud, and C. Meinel, "Analysis and simulation of hpc applications in virtualized data centers," in *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications (GreenCom)*. IEEE Computer Society, 20-23 November 2012, Conference Proceedings, pp. 498–507.

[71] F. Song, D. Huang, H. Zhou, and I. You, "Application-aware virtual machine placement in data centers," in *Proceedings of the 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. IEEE Computer Society, 4-6 July 2012, Conference Proceedings, pp. 191–196.

[72] "Novell platespin recon," 2014, [Accessed 03-07-2015]. [Online]. Available: http://www.novell.com/products/recon/

[73] "Vmware capacity planner," 2014, [Accessed 17-01-2015]. [Online]. Available: http://www.vmware.com/products/capacityplanner

[74] M. Korupolu, A. Singh, and B. Bamba, "Coupled placement in modern data centers," in *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS2009)*. IEEE Computer Society, 23-29 May 2009, Conference Proceedings, pp. 1–12.

[75] D. G. McVitie and L. B. Wilson, "The stable marriage problem," *Commun. ACM*, vol. 14, no. 7, pp. 486–490, July 1971. [Online]. Available: http://doi.acm.org/10.1145/362619.362631

[76] D. Pisinger, "A minimal algorithm for the 0-1 knapsack problem," *Operations Research*, vol. 45, no. 5, pp. 758–767, October 1997. [Online]. Available: https://doi.org/10.1287/opre.45.5.758

[77] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Virtual machine consolidation in cloud data centers using aco metaheuristic," in *Proceedings of the 2014 20th European Conference on Parallel Processing*. Springer International Publishing, 25-29 August 2014, Conference Proceedings, pp. 306–317. [Online]. Available: https://doi.org/10.1007/978-3-319-09873-9_26

[78] T. Wood, G. Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, and M. D. Corner, "Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '09. New York, NY, USA: ACM, 11 - 13 March 2009, Conference Proceedings, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/1508293.1508299

[79] C. Isci, J. E. Hanson, I. Whalley, M. Steinder, and J. O. Kephart, "Runtime demand estimation for effective dynamic resource management," in *Proceedings of the 2010 IEEE Network Operations and Management Symposium (NOMS)*. IEEE Computer Society, 19-23 April 2010, Conference Proceedings, pp. 381–388.

[80] S. Pandey, A. Barker, K. K. Gupta, and R. Buyya, "Minimizing execution costs when using globally distributed cloud services," in *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*. IEEE Computer Society, 20-23 April 2010, Conference Proceedings, pp. 222–229.

[81] J.-L. Kim and R. D. Ellis, Jr., "A framework for integration model of resource-constrained scheduling using genetic algorithms," in *Proceedings of the 37th Conference on Winter Simulation*, ser. WSC '05. Winter Simulation Conference, 04 - 07 December 2005, Conference Proceedings, pp. 2119–2126. [Online]. Available: http://dl.acm.org/citation.cfm?id=1162708.1163078

[82] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Proceedings of the 2010 9th International Conference on Grid and Cooperative Computing (GCC)*. IEEE Computer Society, 1-5 November 2010, Conference Proceedings, pp. 87–92.

[83] S. Hallett, G. Parr, and S. McClean, "Network aware cloud computing for data and virtual machine placement," in *Proceedings of the 2011 12th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking*

and Broadcasting. PGNet, 08 May 2011, Conference Proceedings. [Online]. Available: http://uir.ulster.ac.uk/22003/

[84] Y. M. C. S.-Y. Tsai, "Optimal provisioning of resource in a cloud service," *IJCSI International Journal of Computer Science Issues*, vol. 7, no. 6, pp. 95–99, November 2010. [Online]. Available: https://pdfs.semanticscholar.org/3b5a/e8aa49312ec4c01b15113cac21515141c0d1.pdf

[85] E. Mohammadi, M. Karimi, and S. R. Heikalabad, "A novel virtual machine placement in cloud computing," *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 10, pp. 1549–1555, October 2011. [Online]. Available: http://ajbasweb.com/old/ajbas/2011/October-2011/1549-1555.pdf

[86] D. Chang, G. Xu, L. Hu, and K. Yang, "A network-aware virtual machine placement algorithm in mobile cloud computing environment," in *Proceedings of the 2013 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE Computer Society, 7-10 April 2013, Conference Proceedings, pp. 117–122.

[87] K. Sato, H. Sato, and S. Matsuoka, "A model-based algorithm for optimizing i/o intensive applications in clouds using vm-based migration," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009 (CCGRID'09)*. IEEE Computer Society, 18-21 May 2009, Conference Proceedings, pp. 466–471.

[88] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *Proceedings of the 2012 IEEE INFOCOM*. IEEE Computer Society, 25-30 March 2012, Conference Proceedings, pp. 963–971.

[89] R.-S. Chang and H.-P. Chang, "A dynamic data replication strategy using access-weights in data grids," *The Journal of Supercomputing*, vol. 45, no. 3, pp. 277–295, September 2008. [Online]. Available: https://doi.org/10.1007/s11227-008-0172-6

[90] M. Björkqvist, L. Y. Chen, and W. Binder, "Optimizing service replication in clouds," in *Proceedings of the Winter Simulation Conference*, ser. WSC '11. Winter Simulation Conference, 11-14 December 2011, Conference Proceedings, pp. 3312–3322. [Online]. Available: http://dl.acm.org/citation.cfm?id=2431518.2431911

[91] D.-W. Sun, G.-R. Chang, S. Gao, L.-Z. Jin, and X.-W. Wang, "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 256–272, March 2012. [Online]. Available: https://doi.org/10.1007/s11390-012-1221-4

[92] S.-M. Park, J.-H. Kim, Y.-B. Ko, and W.-S. Yoon, "Dynamic data grid replication strategy based on internet hierarchy," in *Proceedings of the 2003 Second International Conference on Grid and Cooperative Computing (GCC 2003)*. Springer, 7-10 December 2003, Conference Proceedings, pp. 838–846.

[93] K. Sashi and A. S. Thanamani, "Dynamic replication in a data grid using a modified bhr region based algorithm," *Future Generation Computer Systems*, vol. 27, no. 2, pp. 202 – 210, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X10001652

[94] A. Horri, R. Sepahvand, and G. Dastghaibyfard, "A hierarchical scheduling and replication strategy," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 8, pp. 30–35, 30 August 2008.

[95] N. Mansouri and G. H. Dastghaibyfard, "A dynamic replica management strategy in data grid," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1297 – 1303, 2012, intelligent Algorithms for Data-Centric Sensor Networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804512000288

[96] M.-K. Hussein and M.-H. Mousa, "A light-weight data replication for cloud data centers environment," *International Journal of Engineering and Innovative Technology*, vol. 1, no. 6, pp. 169–175, June 2012. [Online]. Available: http://www.ijeit.com/vol%201/Issue%206/IJEIT1412201206_31.pdf

[97] J. Frey, "Condor dagman: Handling inter-job dependencies," 2002, accessed 19-July-2015]. [Online]. Available: http://www.cs.wisc.edu/condor

[98] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005. [Online]. Available: http://dx.doi.org/10.1155/2005/128026

[99] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 22 November 2004. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bth361

[100] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, "Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids," in *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, ser. SWEET '12. New York, NY, USA: ACM, 20 - 20 May 2012, Conference Proceedings, pp. 1:1–1:13. [Online]. Available: http://doi.acm.org/10.1145/2443416.2443417

[101] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, "Task scheduling strategies for workflow-based applications in grids," in *Proceedings of the 2005 IEEE International Symposium on Cluster Computing and the Grid, 2005. (CCGrid2005)*, vol. 2. IEEE Computer Society, 9-12 May 2005, Conference Proceedings, pp. 759–767.

[102] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon: A fast and light-weight task execution framework," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, ser. SC '07. New York, NY, USA: ACM, 10 - 16 November 2007, Conference Proceedings, pp. 43:1–43:12. [Online]. Available: http://doi.acm.org/10.1145/1362622.1362680

[103] W. Johnston, "Esnet: Advanced networking for science," *SciDAC review*, vol. 4, no. 1, p. 48, 2007. [Online]. Available: https://www.es.net/assets/ESnet-News/ESnet-Advanced-Networking-For-Science.pdf

[104] R. Summerhill, "The new internet2 network," in *in 6th GLIF Meeting*, 11 September 2006. [Online]. Available: https://www.glif.is/meetings/2006/plenary/summerhill-internet2.pdf

[105] E.-S. Jung, S. Ranka, and S. Sahni, "Workflow scheduling in e-science networks," in *Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC)*. IEEE Computer Society, 15 August 2011, Conference Proceedings, pp. 432–437.

[106] V. Subramani, R. Kettimuthu, S. Srinivasan, and S. Sadayappan, "Distributed job scheduling on computational grids using multiple simultaneous requests," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, 2002. (HPDC-11)*. IEEE Computer Society, 24-26 July 2002, Conference Proceedings, pp. 359–366.

[107] O. Wäldrich, P. Wieder, and W. Ziegler, "A meta-scheduling service for co-allocating arbitrary types of resources," in *Proceedings of the 6th International Conference on Parallel Processing and Applied Mathematics (PPAM2005)*, vol. 3911. Springer Berlin, Heidelberg, 11-14 September 2005, Conference Proceedings, pp. 782–791.

[108] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," in *Proceedings of the 11th IEEE Interna-*

tional Symposium on High Performance Distributed Computing, 2002. HPDC-11 2002. IEEE Computer Society, 24-26 July 2002, Conference Proceedings, pp. 352–358.

[109] S.-M. Park and J.-H. Kim, "Chameleon: a resource scheduler in a data grid environment," in *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings. (CCGrid 2003).* IEEE Computer Society, 12-15 May 2003, Conference Proceedings, pp. 258–265.

[110] J. Bent, D. Rotem, A. Romosan, and A. Shoshani, "Coordination of data movement with computation scheduling on a cluster," in *Proceedings of the 2005 Challenges of Large Applications in Distributed Environments (CLADE 2005).* IEEE Computer Society, 24-24 July 2005, Conference Proceedings, pp. 25–34.

[111] E.-S. Jung, K. Maheshwari, and R. Kettimuthu, "Pipelining/overlapping data transfer for distributed data-intensive job execution," in *Proceedings of the 2013 42nd International Conference on Parallel Processing (ICPP).* IEEE Computer Society, 1-4 October 2013, Conference Proceedings, pp. 791–797.

[112] j. Ajay and T. Saini, "Scheduling optimization in cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 62–65, April 2013. [Online]. Available: http://ijarcsse.com/Before_August_2017/docs/papers/Volume_3/4_April2013/V3I4-0197.pdf

[113] D. Kim, J. Lee, J. Lee, I. Shin, J. Kim, and S.-E. Yoon, "Scheduling in heterogeneous computing environments for proximity queries," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 9, pp. 1513–1525, 05 April 2013. [Online]. Available: https://doi.org/10.1109/TVCG.2013.71

[114] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 01 -

30 November 2010, Conference Proceedings, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879175

[115] A. Vahdat, H. Liu, X. Zhao, and C. Johnson, "The emerging optical data center," in *Proceedings of the Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2011*. OSA Technical Digest (CD), 6–10 March 2011, Conference Proceedings, p. OTuH2. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2011-OTuH2

[116] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 30 August - 03 September 2010, Conference Proceedings, pp. 339–350. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851223

[117] H. Newman, J. Bunn, I. Legrand, D. Nae, S. Ravot, X. Su, F. van Lingen, and Y. Xia, "The motivation, architecture and demonstration of ultralight network testbed," in *proceedings of 2006 the First CESNET Conference on Advanced Communications and Grids (CESNET)*. Praha, 2006, Conference Proceedings, pp. 11–26. [Online]. Available: http://resolver.caltech.edu/CaltechAUTHORS:20110728-12119631

[118] M. Behrendt, B. Glasner, P. Kopp, R. Dieckmann, G. Breiter, S. Pappe, H. Kreger, and A. Arsanjani, "Introduction and architecture overview ibm cloud computing reference architecture 2.0," Technical Report IBM, Tech. Rep., February 2011. [Online]. Available: https://s3-sa-east-1.amazonaws.com/bucketmanoelveras/manoel_veras_PROFESSIONAL/CLOUDCOMPUTING/Artigos/Artigos_IBM/arq_ibm.pdf

[119] S. Leimeister, M. Böhm, C. Riedl, and H. Krcmar, "The business perspective of cloud computing: Actors, roles and value networks." in *Proceedings of the 18th*

European Conference on Information Systems (ECIS). AIS Electronic Library (AISeL), 6-9 June 2010, Conference Proceedings, p. 56.

[120] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08).* IEEE Computer Society, 25-27 September 2008, Conference Proceedings, pp. 5–13.

[121] Y. Alharbi and K. Yang, "Optimizing jobs' completion time in cloud systems during virtual machine placement," in *Proceedings of the 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC).* IEEE Computer Society, 15-16 March 2016, Conference Proceedings, pp. 1–6.

[122] Y. Alharbi and S. Walker, "Data intensive, computing and network aware (dcn) cloud vms scheduling algorithm," in *Proceedings of the 2016 Future Technologies Conference (FTC).* IEEE Computer Society, 6-7 December 2016, Conference Proceedings, pp. 1257–1264.

[123] R. C. Papademetriou and D. A. Karras, "Towards a thorough evaluation framework of software tools suitable for small and medium size enterprises focusing on modelling and simulating business processes," in *Proceedings of the 2016 International Symposium on Business Modeling and Software Design (BMSD 2016).* Springer, 20-22 June 2016, Conference Proceedings, pp. 161–182.

[124] S. Venugopal and R. Buyya, "An economy-based algorithm for scheduling data-intensive applications on global grids," Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, Tech. Rep. GRIDS-TR-2004-11, Tech. Rep., 2004.

[125] N. Mansouri, "A prediction-based replication algorithm for improving data availability in frid environment," *Journal of Telecommunication, Electronic and*

Computer Engineering (JTEC), vol. 6, no. 1, pp. 35–42, 2014. [Online]. Available: http://journal.utem.edu.my/index.php/jtec/article/view/457/325

[126] P. Nguyen and K. Nahrstedt, "Monad: Self-adaptive micro-service infrastructure for heterogeneous scientific workflows," in *Proceedings of the 2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE Computer Society, 17-21 July 2017, Conference Proceedings, pp. 187–196.

[127] J. Smith and S. K. Shrivastava, "A system for fault-tolerant execution of data and compute intensive programs over a network of workstations," in *Proceedings of the 1996 European Conference on Parallel Processing*. Springer International Publishing, 26-29 August 1996, Conference Proceedings, pp. 487–495.

[128] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[129] K.-i. Kitayama, "A vision toward smart photonic cloud," in *Proceedings of the 2015 International Conference on Photonics in Switching (PS)*. IEEE Computer Society, 22-25 September 2015, Conference Proceedings, pp. 10–12.

[130] P. Jain, D. Rane, and S. Patidar, "A survey and analysis of cloud model-based security for computing secure cloud bursting and aggregation in renal environment," in *Proceedings of the 2011 World Congress on Information and Communication Technologies (WICT)*. IEEE Computer Society, 11-14 December 2011, Conference Proceedings, pp. 456–461.

[131] S. Ghanbari and M. Othman, "A priority based job scheduling algorithm in cloud computing," *Procedia Engineering*, vol. 50, no. Supplement C, pp. 778–785, 01 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705814000022

[132] "Apache software foundation," 2015, [Accessed 25-02-2016]. [Online]. Available: https://hadoop.apache.org

[133] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 1 January 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[134] A. Kathpal, M. Kulkarni, and A. Bakre, "Analyzing compute vs. storage tradeoff for video-aware storage efficiency," in *Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems, HotStorage*. USENIX Association, 12-15 June 2012, Conference Proceedings. [Online]. Available: https://www.usenix.org/conference/hotstorage12

[135] M. Alicherry and T. Lakshman, "Optimizing data access latencies in cloud systems by intelligent virtual machine placement," in *Proceedings of the 2013 IEEE INFOCOM*. IEEE Computer Society, 14-19 April 2013, Conference Proceedings, pp. 647–655.

[136] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 19 June 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613001084

[137] J. D. Davis, T. V. Johnson, and C. C. Ting, "Web-based enterprise management with multiple repository capability," 13 December 2005, uS Patent 6,976,262. [Online]. Available: https://www.google.com/patents/US6976262

[138] T. Kosar and M. Livny, "A framework for reliable and efficient data placement in distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 65, no. 10, pp. 1146–1157, October 2005, design and Performance of Networks for Super-, Cluster-, and Grid-Computing Part I. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731505001012

[139] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and qos in cloud computing environments," in *Proceedings of the 2011 international conference on Parallel processing (ICPP)*. IEEE Computer Society, 13-16 September 2011, Conference Proceedings, pp. 295–304.

[140] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Proceedings of the 2006 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*. IEEE Computer Society, 3-7 April 2006, Conference Proceedings, pp. 373–381.

[141] L. Cherkasova and R. Gardner, "Measuring cpu overhead for i/o processing in the xen virtual machine monitor." in *Proceedings of the USENIX Annual Technical Conference, General Track*, vol. 50. USENIX Association, 10-15 June 2005, Conference Proceedings.

[142] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*, ser. SoCC '12. New York, NY, USA: ACM, 14-17 October 2012, Conference Proceedings, pp. 7:1–7:13. [Online]. Available: http://doi.acm.org/10.1145/2391229.2391236

[143] Y. Pradhananga, S. Karande, and C. Karande, "High performance analytics of bigdata with dynamic and optimized hadoop cluster," in *Proceedings of the 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*. IEEE Computer Society, 25-27 May 2016, Conference Proceedings, pp. 715–720.

[144] T. Kosar, "A new paradigm in data intensive computing: Stork and the data-aware schedulers," in *Proceedings of the 2006 IEEE Challenges of Large Applications in Distributed Environments*. IEEE Computer Society, 19 June 2006, Conference Proceedings, pp. 5–12.

[145] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *The International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, August 2001. [Online]. Available: https://doi.org/10.1177/109434200101500302

[146] GridWayTeam, "Gridway 5.2 documentation: User guide distributed, systems architecture group, technical report," Universidad Complutense de Madrid, Tech. Rep., February 2007.

[147] Y. Alharbi and S. Walker, "Virtual machine placement using pre-fetching data transfer," in *Proceedings of the Computing Conference 2017*. IEEE Computer Society, 18-20 July 2017, Conference Proceedings, pp. 1–6.

[148] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599 – 616, June 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X08001957

[149] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: Outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW '09. New York, NY, USA: ACM, 13 November 2009, pp. 85–90. [Online]. Available: http://doi.acm.org/10.1145/1655008.1655020

[150] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, April 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[151] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "Greencloud: A new architecture for green data center," in *Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*, ser. ICAC-INDST '09. New York, NY, USA: ACM, 15 June 2009, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/1555312.1555319

[152] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," in *Proceedings of the First IEEE International Conference on Cloud Computing (CloudCom 2009)*. Beijing, China: Springer Berlin Heidelberg, 1-4December 2009, Conference Proceedings, pp. 24–44.

[153] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *SIGMOD Rec.*, vol. 34, no. 3, pp. 31–36, September 2005. [Online]. Available: http://doi.acm.org/10.1145/1084805.1084812

[154] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "Snowflock: Rapid virtual machine cloning for cloud computing," in *Proceedings of the 4th ACM European Conference on Computer Systems*, ser. EuroSys '09. New York, NY, USA: ACM, 1-3 April 2009, Conference Proceedings, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1519065.1519067

[155] M. H. Jamal, A. Qadeer, W. Mahmood, A. Waheed, and J. J. Ding, "Virtual machine scalability on multi-core processors based servers for cloud computing workloads," in *Proceedings of the IEEE International Conference on Networking, Architecture, and Storage (NAS 2009)*, IEEE. IEEE Computer Society, 9-11 July 2009, Conference Proceedings, pp. 90–97.

[156] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf, "The cactus worm: Experiments with dynamic resource discovery and allocation in a grid environment," *The International Journal of High Performance Computing Applications*, vol. 15, no. 4, pp. 345–358, November 2001. [Online]. Available: https://doi.org/10.1177/109434200101500402

[157] E. Huedo, R. S. Montero, and I. M. Llorente, "An experimental framework for executing applications in dynamic grid environments," Institute For Computer Applications In Science And Engineering Hampton Va, Tech. Rep., November 2002. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a409442.pdf

[158] M. A. Chauhan and M. A. Babar, "Migrating service-oriented system to cloud computing: An experience report," in *Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD)*. IEEE Computer Society, 4-9 July 2011, Conference Proceedings, pp. 404–411.

[159] N. Saluja, S. Rai, R. D. Nayak, A. P. HOD, and P. Scholar, "Task scheduling using improved task migration consolidation in cloud computing," *International Journal of Engineering Science*, vol. 7, no. 8, August 2017.