AN ENHANCED ALGORITHM FOR SCHEDULING DEPENDENT TASKS IN CLOUD COMPUTING ENVIRONMENT

BY

HAMZA, Hafsat

P15SCMT8003

**A DISSERTATIONSUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA
IN PARTIAL FULFILLMENT FOR THE AWARD OF MASTER OF SCIENCE (M.Sc.)
DEGREE IN COMPUTER SCIENCE**

**DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF PHYSICAL SCIENCE,
AHMADU BELLO UNIVERSITY,
ZARIA, NIGERIA**

DECEMBER, 2019

DECLARATION

I declare that the work in this dissertation titled "**AN ENHANCED ALGORITHM FOR SCHEDULING DEPENDENT TASKS IN CLOUD COMPUTING ENVIRONMENT**" has been carried out by me in the Department of Computer Science under the supervision of Dr. A. F. Donfack Kana and Dr. S. Aliyu.The information derived from literature has been duly acknowledged in the text and a list of reference provided. No part of this thesis was previously presented for another degree or diploma at any University or Institution.

_____

HAMZA, Hafsat Date

Name of Student

CERTIFICATION

This dissertationtitled,**"AN ENHANCED ALGORITHM FOR SCHEDULING DEPENDENT TASKS IN CLOUD COMPUTING ENVIRONMENT"**by**HAFSAT HAMZA (P15SCMT8003)**meets the regulations governing the award of Master of Science ofAhmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Dr A. F. Donfack Kana Chairman, Supervisory Committee	_____ Signature	_____ Date
Dr. S. Aliyu Member, Supervisory Committee	_____ Signature	_____ Date
Prof. S. B. Junaidu Head of Department	_____ Signature	_____ Date
_____ External Examiner	_____ Signature	_____ Date
Prof. A.A. Sani Dean, School of postgraduate studies	_____ Signature	_____ Date

DEDICATION

This dissertation is dedicated to my beloved Parents Alh. HamzaMadawaki and HajiyaMaryam NaladoMadawaki, whose love and passion for education have won me this priceless gift.

ACKNOWLEDGEMENT

My profound gratitude goes to Almighty ALLAH the Omnipotent on whom my life solely depends, who by His Grace this dream becomes a reality. My sincere appreciation goes to my supervisors Dr. A. F. Kana and Dr. S. Aliyu for their efforts to make this work a success. Their suggestions and strictness has really brought out the best in me. My special thanks go to my family and friends for their prayers, words of motivation and words of comfort especially to Shehu Ibrahim Dabai. My appreciation to Mal. M. Y. Tanko for his continued support throughout this research work and the entire 2015/2016 students of Computer Science Department, I want to say thank you all.

ABSTRACT

Cloud computing is a model which aims to deliver a reliable, customizable and scalable computing environment for end-users.Cloud computing is one of the most widely used technologies embraced by sectors and academia, offering a versatile and effective way to store and retrieve documents. The performance of cloud computing services always depends upon the performance of the execution of user tasks submitted to the cloud system. Scheduling of the users' tasks plays significant role in improving performance of the cloud services..This research provided an enhanced dependent tasks scheduling technique that extends a recent research and aims at improving the overall performance of the system. The Dependent tasks were represented as a Directed Acyclic Graph(DAG) and the number of dependent tasks and their total runningtime were used as heuristicfordetermining which path should be explored first and using the defined heuristic,a Best First Search (BFS) approach was used to traverse the graph in order to determine which task should be scheduled next. The results of the simulation using WorkflowSim toolkit showed that keeping the number of cloudlet constant and varying the number of resources, an average improvement of 18% and 19% on waiting time and turnaround time were achievedrespectively.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 Background of Study

Cloud computing refers to a model used to manage and deliver services over the web and it isdefined as a model for enabling ubiquitous, 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(Xioa, et al., 2013). Cloud computing is one of the most widely used technologies embraced by sectors and academia, offering a versatile and effective way to store and retrieve documents (Anthony, et al., 2010).A cloud, through capabilities called services, can engage in a multitude of ways with a client. Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) are the three maincategories of cloud computing services(Bhaskar et al., 2009). Software as a Servicerefers toContent on the web that clients use aninternet browser to access. SaaS is mainly accessed through a web portal and service oriented architectures based on web service technologies. The advantages of SaaS solutions are simplicity of integration, cost and scalability. The disadvantages of SaaS solutions is the perception of security issues (e.g. Facebook, YouTube etc). Furthermore, Platform as a Service (PaaS) includesthe development and delivery environment for cloud applications. The main users of this layer are developers wanting to develop and run a cloud application for a particular platform. The hardware and software within a PaaS solution is managed by the platform provider. Well known PaaS solutions providers include Windows Azure and Google app engine. Moreover, Infrastructure as a Service (IaaS)offers a cloud-based virtual data center. IaaS primarily offers conceptual Internet infrastructure (e.g. compute cycles

or storage)IaaS offers the opportunity to expand its IT infrastructure on request by organizations and developers.

Models of cloud computing services are either deployed as a public cloud, private cloud, community cloud, or hybrid cloud.

Public cloud is preserved for public at large. It can be controlled or retained by, or some combination of, academic, company or statecraft organizations. Public cloud is managed by cloud provider andusually the cheapest and less secure (Ertual, et al., 2010). Private cloud isgrown for an individual organisation usage that consists of various customers, for example, company units. It can be governed and managed by the single organization for its own usage. The primary benefit of these schemes is that the company maintains complete authority over system efficiency, corporate data and safety rules (Ertual, et al., 2010). Community cloud is retained to define consumer groups from organizations that have to experience the same information within each other.These organizations have comparable cloud needs and their ultimate objective is to work together to attain their company goals. It is structured by one or more community-based organizations (Dillan, et al., 2010). Hybrid cloud is a mixture of public, privateor community clouds which are solitary entities but coupled with technology that allows information and application mobility such as cloud bursting for cloud load balancing (Liu, et al, 2011).

In the increasing era of cloud computing, requirements in nearly all sectors make the price of cloud infrastructure very high. Therefore, the notion of virtualization in cloud computing systems must be understood and implemented. The technology that permits various virtual machines, known also as guest machines, to work on a single physical machine and to share the resources of the physical machine is called virtualization(Ryan & Jiangchuan, 2012).

This allows a single physical server to host many virtual machines (VMs), operating systems and apps without the extra expenses and complexity of these various physical computers operating individually (Tupakula & Varadharajan, 2011).Cloud computing services ' performance and effectiveness always relies on the performance of user tasks presented to the cloud system. Scheduling of tasks submitted by the user plays an important role in enhancing cloud services efficiency. One of the primary kinds of scheduling carried out in cloud computing is a task scheduling, the unplanned incoming of jobs with irregularprocessing time criteria can strongly overload a particular resource while other resources are doing nothing or less loaded (Livny & Melman, 2010).

The demand for cloud-based services raised some cloud-based issues(Alharba & Yang, 2016). The open and dynamic characteristics of a Cloud Computing environment are easy to cause some issues like availability and allocation of physical resources, low utilization of resources, user priority management, excessive energy consumption(Ben Alla et al., 2019). The restricted amount of resources and the infinite amount of demands generate some cloud-based overloading scenarios.Different job and task planning algorithms are used to balance overloaded scenarios. The scheduling algorithms for tasks and jobs in cloud computing are basically based on the CPU scheduling algorithm idea. However, there is a restricted amount of requests for CPU scheduling algorithms, but here the request scenario is unlimited (Nahir, et al., 2016; Ananth & Chandrasekan, 2015).

Load balancing is among the biggest problems in cloud computing(Rima, et al., 2009).It is a technique that uniformly distributes the workload throughout the cloud in order to prevent a scenario in which some resourcesare heavily loaded whereas other resources are less loaded or idle. Load balancing aides in obtaining a high level of user fulfilment and resource usage efficiency, thus enhancing the cloud computing system's performance and resource usage.

It also guarantees that each resource is effectively and evenly distributed (Wang, et al., 2010). In addition, the overhead effect on the identification of resource use and migration of task must be weighed in the algorithms for load balancing (Balagoni & Roa, 2014).

There exist variousscheduling algorithmsin cloud computing like, weighted round robin, round robin, Equally Spread Current Execution (ESCE) Algorithm, Ant Colony algorithm, First Come First Serve and Throttled algorithm. The most commonly used scheduling algorithms for a nonpreemptive tasks are first in first out (FIFO) and weighted round robin(WRR)(Babu & Krishna, 2013).

The goal of this research work is to improve the weighted round robin algorithm for scheduling non preemptive dependent tasks in cloud computing environment proposed by(Devi & Uthariaraj, 2016) by enhancing the efficiency of the task dependent scheduler in order to improve the overall performance of the system.

1.2 Motivation

The notion of cloud computing has transformed the field of parallel and distributed computing systems today.Cloud computing allows access to scalable, distributed,virtualized hardware and/or software infrastructure on the internet for a broad spectrum of users. Scheduling plays a crucial role by disseminating loads on virtual machines to maximize the use of resources while decreasing the total time for task execution in the cloud.Some existing scheduling algorithms can deliver better strategies through efficient resource allocation and job scheduling method.

1.3 Statement of Problem

In cloud computing environment, scheduling tasks on virtual machines (VMs) is a significant feature.Some researchers have proposed many dependent task algorithms for heterogeneous systems in recent years.However some of these proposed algorithms often create some overloading scenarios, poor resource utilization and the techniques in determining which task should schedule next are inefficient (Xie et al., 2016).

Devi &Uthariaraj,(2016) proposed an algorithmfor schedulingnonpreemptive dependent tasks in cloud computing environment. However, their technique for determining which task to be scheduled next is based on random selection. This research work enhances the dependent task scheduling algorithm used in (Devi &Uthariaraj, 2016) by using a heuristic approach for determining which task to be scheduled next in order to improve the overall performance of the system.

1.4 Aim and Objectives

The aim of this research is to enhance theefficiency of the dependent tasks scheduling algorithm for scheduling dependent task in cloud computing environmentproposed in(Devi & Uthariaraj, 2016)next in order to improve the overall performance of the system.

The objectives are to:

(i) Use an appropriate data structure to represent the dependencies between the tasks.

(ii) Define a heuristic to prioritize the tasks to be scheduled.

(iii)Implement and simulate the enhanced algorithm.

(iv)Evaluate performance of the enhanced algorithm.

1.5 Research Methodology

The following procedures are to be adopted for this research work:

(i) The dependency between the tasks is represented using a Directed Acyclic Graph (DAG).

(ii) The number of descendentand run time of each node areused as heuristic to determine the order of scheduling

(iii) The best first searchapproachis used for scheduling based on the heuristic defined in (ii).

(iv)The proposed algorithm was simulated using workflowsim 1.0.

(v) The proposed algorithm was evaluated based on waiting time and turnaround time of tasks side by side with that of Devi&Uthariaraj (2016).

1.6 Contribution to Knowledge

The major contributions made are:

(i) The dissertation provided an enhanced algorithm for scheduling dependent tasks in cloud computing environment

(ii) The priority assigned on each task is based on their properties.

(iii)The performance in terms of waiting time and turnaround time were improved.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Cloud computing combines a technology with a platform that offers web hosting and storage services (Harjit & Gurdav, 2011).In such an environment, users do not need to own the infrastructure for assorted computing services. In fact, they are often accessed from any computer in almost any part of the world. Cloud computing offers dynamic allocation of resources on demand, the feature which makes it to stand apart providing great performance, scalability, cost efficient and less maintenance, thus making it an apt choice(Krishnadoss & Jacob, 2018).Compared to current computing techniques, it also provides enhanced flexibility. It can deploy, allocate or dynamically re-allocate resources through the ability to monitor their performance continually (Peter & Timothy, 2011). In addition, cloud computing minimizes spending on the administrative center. Cloud computing approach is independent of user location. Cloud computing's ultimate goal is to render scalable and low-cost on-request computing infrastructures with outstanding quality and data levels for services.

Cloud computing is typically a generic word for any cost involving the provision of hosting facilities over the Internet. Unlike the static system architecture, it promotes the likelihood of improving dynamically and scaling down rapidly. This provides greater reliability and fastest response times for cloud users, as well as flexibility in handling large traffic fluctuations and demand (Wang, et al., 2008).Furthermore, cloud computing promote multi-tenancy, offering configured systems to the level that many organisations or individuals can share them together. Virtualization technology in cloud computing enables cloud service providers to convert a single server into various VMs(Foster, et al., 2008). This feature eliminates the need for a single-purpose computing system for a client-server. Cloud computing virtualization maximizes the capability of hardware and also enables users to manage economies of scale (Wang, et al., 2008).

As the amount of cloud users increases over time, it has become very important to provide a quality service.

2.2 Virtualization

Virtualization refers to a technology that allows multiple Virtual Machines (also called guest machines) to run on a single physical machine (also called host machine) and share the resources of the physical machine (Ryan & Jiangchuan, 2012). Cloud computing virtualized a single system into numerous virtual machines using virtualization method(Angeles, 2014). Virtualization allows users to use the cloud's various apps or services, making this the primary part of the cloud environment(kaur & luthra, 2014). It enables various operating systems to be run on a single physical system and shares the fundamental hardware resources(Sotomayor, et al., 2009). Virtualization can be implemented to a wide spectrum such as operating system, hardware level, and virtualization of servers.Virtualization technology is cost-saving technology and energy-saving technology that transforms the vital computing method quickly.Virtual machine (VM) is an execution unit that serves as a cloud computing technology bedrock.The VM generally utilizes two distinct task execution processes; space shared and time shared (Devi & Uthariaraj, 2016).

The tasks are executed one by one in space shared system. It means that in its central processing unit only a single task per central processing unit / processing element is executed.While in a time shared technique, tasks are simultaneously executed in a time shared manner which resembles the execution of tasks in parallel mode (Devi & Uthariaraj, 2016).

The computer software, which generates and operates virtual machines is called a virtual machine monitor (VMM) or hypervisor.

On a machine named a host machine, one or more virtual machines are run by a virtual machine monitor. This machine can be a server and a computer (Gouda, et al., 2014).Hardware

virtualization, desktop virtualization and storage virtualization are the three classifications of virtualization.

2.2.1 Hardware Virtualization

Hardware or platforms virtualization relates to creating a virtual machine that functions as a true working computer with an OS. On these virtual computers, software is segregated from the underlying hardware resources. The physical machine is the host machinewhere the virtualization is being carried out in hardware virtualization and the multiple virtual machines is the guest machine(Sotomayor, et al., 2009).

Hardware virtualization can be divided into the following categories(Rajwinder & Pawan , 2014):

i. Full virtualization: This virtualization refers to installing a full machine on a different machine. And the virtual machine offers the whole of the initial machine's functionality. It enables various clients to share a computer system. It offers hardware emulation on another machine.

ii. Partial virtualization: Partial virtualization implies simulating a part but not the whole target environment. Therefore, some guest programs may require changes to operate in this virtual environment.

iii. Para virtualization: This virtualization means that the hardware enables numerousOS to run on a specific machine. Para virtualizationenables the effective utilization of system resources like memory and processor.

2.2.2 Desktop Virtualization

Desktop Virtualization virtualizes a conventional desktop or workstation into virtual machine that consist of all its apps, customizations and preferences made by the user. Using any workstation, users can access this virtual machine from anywhere in the organisation,thus

reducing the cost of licensing software installation on individual computers. This decreases the need for hardware duplication and other economic aspects(Gouda, et al., 2014).

2.2.3 Storage Virtualization

This virtualization implies that the accessible storage is virtualized for big access to virtual storage and is also used to allocate memory to cloud client (Malhotra, et al., 2014). This becomes highly helpful for Business Continuity, Planning and Disaster Recovery, as you can readily replicate your storage across to another place or even a separate geographic region with a virtualized infrastructure.

2.3 Task Scheduling

A task refers to a tiny piece of job to be carried out within a specified timeframe. Task scheduling dispatches the tasks provided by the cloud users to the cloud provider on available resources(Arya & Verma, 2014). Scheduling is carried out based on various criteria (task length, priority, VM capacity, etc.) to increase the general cloud efficiency. The task can involve data entry, processing, software access, or storage functions.The data centrecategorise duties by service-level agreement and services demanded.Then each task is allocated to one of the servers available,the servers then execute the required task, and client receives a reply or outcome.Scheduling tasks in cloud computing setting offers "multi-programming capabilities."It may be in preemptive or non-preemptive mode.

In the preemptive mode, interrupting the present task execution and migrating the task to another resource is possible. Preemption may be helpfulif task priorities are to be regarded as one of the constraints while a task should be completed in the resource in the non-preemptive mode, i.e. the resource cannot be removed from the task (George & muthulakshmi, 2011).

Task Scheduling plays a major part in enhancingreliability and flexibility of cloud systems (Singh, et al., 2014; Shimpy & Sidhu, 2014). Due to its parallel and distributed architecture, task scheduling is a challenging matter in cloud computing. The time to finish the execution of task

is difficult to define in the cloud since the tasks can be spread amongmultiple virtual machines(Kaur & Dhindsa, 2017).

Users send their jobs to the cloud scheduler in the task scheduling process.The Cloud Information Service obtain the state of the resources available and their properties on the request of the cloud scheduler and assign different tasks to different resources as required by the task. Multiple tasks submitted by the user are allocated to various virtual machines by the cloud scheduler. The main process of task scheduling is as shown in Figure 2.1.

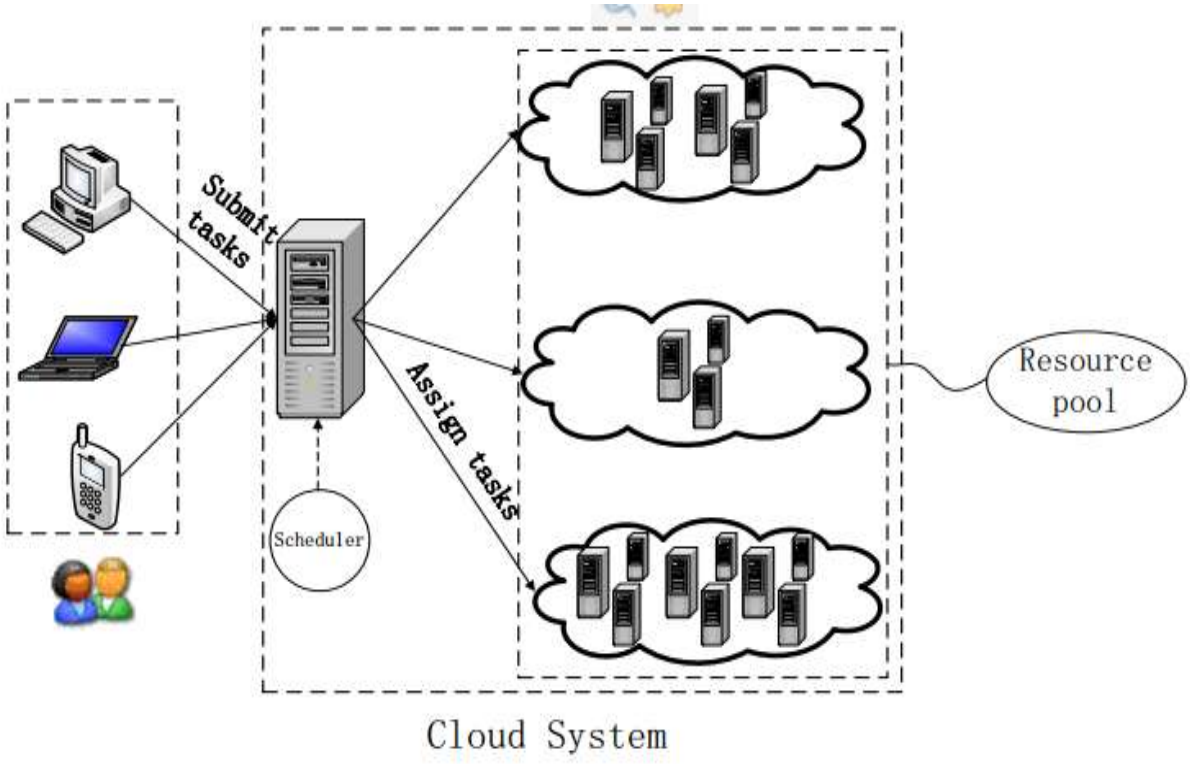

Figure 2.1: Task Scheduling in Cloud (Kaur&Dhindsa, 2017)

An efficient task scheduling technique involves not only meeting the requirements of the user, but also enhancing the system's efficiency by scheduling tasks on the appropriate virtual machine.

2.3.1 Classification of task scheduling

Methods of task scheduling can be centralized or distributed. It can be carried out on dependent or independent tasks and in a homogeneous or heterogeneous environment (Vivek, et al., 2017):

i. Centralized scheduling: In centralized scheduling a specific scheduler has the responsibility to make the global scheduling decisions. The advantage of centralized scheduling is ease on implementation, but the drawbacks are lack of fault tolerance, scalability and performance (Casavant & Kuhl, 1988).

ii. Distributed scheduling: In distributed scheduling, scheduling is divided between distinct schedulers (Arora, et al., 2002). Distributed scheduling is highly complex in implementation. There are two kinds of scheduling methods in the distributed environment; heuristic and hybrid techniques (Mathew , et al., 2014). Heuristic technique iscategorised into static anddynamic scheduling. Every task is known before scheduling in static scheduling and are statically allocated to virtual resources while ina dynamic scheduling, all the tasks are scheduled immediately, as they reach the system, it can be executed in batch mode or online mode (Tracy, et al., 2001).

The dynamic algorithm works better than static algorithms due to all the tasks are scheduled immediatelyas they reach the system. However, a dynamic scheduling algorithm has higher overhead since we want to schedule and as well update the information on the system instantly. Static scheduling is simple to implement from the programmer's point of view, while dynamic scheduling is more appropriate for real-world situations(Mathew , et al., 2014).

There are threekinds of hybrid algorithms:minimization-maximization approach, multi-objective and energy aware methods.

2.4 Scheduling parameters

The following qualitative parameters are used to analyse or compare the performance of various task scheduling algorithms.

i. Central Processing Unit (CPU) utilization: CPU is one of the various resources that a userused in a cloud environment. CPU utilization shows the value between 0 and 100, which explains the percentage of CPU usage. Value 100 means that CPU is not available for further processes. In this scenario if an incoming process is placed in the queue, it will create a huge mess up for the CPU. Therefore, it is necessary to have a deep look on the utilization of resources so that CPU is always ready for new processes. In general, CPU utilization means how processors are used by different processes(Waqas & Awais, 2018).

ii. Throughput: Throughput is a measure of how many information units the system can process in a given timeframe. (Rouse, 2015). By maximizing the amount of jobs processed per unit time, the throuput is maximized.

iii. Waiting Time: Is the time processes spent in the ready queue to get on the CPU. By minimizing the process waiting time, an efficient scheduling can be accomplished(Waqas & Awais, 2018).

iv. Execution Time: Execution time is the exact time taken to execute a given task. The ultimate goal of any scheduling algorithm is minimizing the execution time.

v. Turnaround Time: The amount of time to execute a specifictask is called the turnaround time i.e. from the time of submission until completion time(Abilijit & Apte, 2013)

vi. Completion Time: The time it takes to finish the whole task execution is called the completion time. It also involves the delay caused by the cloud system. A scheduling algorithm is optimized when completion time is minimized (Nidhi , 2017).

vii. Makespan: Makespan is defined as the completion time of the algorithm. It is calculated by measuring the finishing time of the exit task by the algorithm(Nidhi , 2017).

2.5 Related Works

Several works have been carried out in the recent time to enhance scheduling in cloud computing in order to reducethe overall completion, waiting time, balance load across the VMs and effective utilization of the VMs.

Chun et al., (2006)Proposed a Critical-Path-on-a-Processor (CPOP) algorithm for distributed heterogeneous computing systems.The CPOP schedules all tasks by their rank.The tasks in a DAG's critical path will obtain higher rank and these tasks are referred to as critical tasks. Scheduling tasks in CPOP consist of two stages; assignment of priority stage and selection of processor stage. In assignment of priority stage, the upward and downward rank values are computed and higher-ranking tasks will be given higher priority. The processor selection stagerefers to selection of a processor, that is, a Critical Path Processor (CPP) is selected for assigning critical tasks.In processor selection stage, the calculation of the execution time of all critical tasks is done on each processor and the processor that has the least execution time is chosen as the CPP. A critical task will be assigned to the CPP processor and non critical tasks are assigned to a processor with greedy algorithm.In most instances, however, there may be very little distinction in duration between critical and non-critical tasks, and there can exist dependency between the critical and non critical tasks, as a consequence, critical path tasks must wait for other non-critical path tasks to be completed.

Liang, et al. (2009) proposed a Non-critical Path Earliest-Finish (NEPF) Algorithm for dependent Task in Hetrogenous Computing Settings. The proposed method adopt DAG to represent the task to be scheduled, the execution order of task is based on the task’s priority and during scheduling it does not consider critical path tasks. Scheduling tasks in NPEF consist of two stages: the task priority assigning stage to calculate the prioriy of all tasks and the processor selection stage to choose the tasks according to the order of the priority and assigned them to the processor based on Earliest Complete Time (ECT). By repeatedlycomputing the value of ECT,

the task with the least ECT will be chosen to be allocated to the processor. However the overall completion time of tasks can be reduced when critical path tasks are considered during task scheduling and NPEF is not suitable in a dynamic hetrogenous computing platform because in the dynamic hetrogenous computing environment the computing unit's capability and network transfer rate will be altered during runtime.

Pasha, et al. (2014) Proposed and implemented a round robin approach for VM load balancing algorithm in Cloud computing environment. The main focus of the Round Robin algorithm is to equally spread load to all nodes.The scheduler assigned one VM to a node cyclically using this algorithm. The round robin scheduling in the cloud and the round robin scheduling used in the process scheduling are very similar. The scheduler begins with a node and moves to the next node after that node has been allocated a VM. This is repeated until all the nodes have been assignedto at least one VM and then the scheduler comes back to the first node again.The scheduler does not pause in this case for the exhaustion of a node's resources before it moves on to the next. Although the distribution of workloads between processors is the same, the processing time for different processes is not the same.

Basker, et al. (2014)proposed an enhanced scheduling in weighted round robin (WRR) for the cloud infrastructure services. This approach takes account of theVMs ' resource capacities and Attributes more tasks to greater capacity VMs depending on the weight of each VM. But to choose the rightful VM, the length of the tasks was not taken into account.

Devi & Uthariaraj (2016) Proposed load balancing in cloud computing environment using an Improved Weighted Round Robin (IWRR) algorithm for nonpreemptive dependent tasks.In the design of IWRR scheduling and load balancing; job request is provided by the user through an interface and it is transferred for dependency and independent task analysis to the task manager.This segment gets the job and checks if the job is a fully independent task or hasvarious task. If it includes various tasks, the interdependency between them is verified. Dependent task

queue and independent task queue are discovered. The dependent tasks are notified to the scheduler in order to schedule parent tasks after child tasks are executed. The scheduler chooses the suitable VM on the basis of information provided by the VM, such as, VMs current load, the length of the arrived tasks, the tasks priority and VMs processing capacity.

The dependent task scheduling algorithm and dynamic scheduling algorithms are shown in Algorithm 2.1 and 2.2.

```
Algorithm: Priority (A[0…n-1])
Input: A list of Dependent Task to Priotize (A[0…n-1])
Output: A list of Task Sorted (B[0….n-1])

while A[ ] is not empty {
        task = A.dequeue ( )
          PriorityHelper(B, task);
        }
// End while
Return b

Algorithm: PriorityHelper (B[ ], Task task)
Input: A Queue and a Task to be schedule
Output: A list of Task Sorted (B[0….n-1])

list[] childtasks = task.children();
        if(empty(childtasks)){
          B.add(task);   //Schedule task
          Update parents priorities
          return B;
        } else {
          while (!empty(childtasks)){
                    Task t = childtasks.dequeue();
                    PriorityHelper(B, t)
```

Algorithm2.1: Dependent tasks algorithm (Devi & Uthariaraj, 2016)

1) Identify the Pending Execution Time in each of the VMs by collecting the Pending Execution

length from executing, waiting &paused list.

(a) Set pendingJobsTotLength = JobsRemainingLengthInExecList + JobsRemainingLengthInWaitList + JobsRemainingLengthInPauseList

(b)$\mathcal{C}$V$m$is the processing capacity of the VM.

(c) Set pendingETime = pendingJobsTotLength/$\mathcal{C}$Vm

(2) Arrange the VMs based on the least pending execution time to the highest pending execution time and group it, in case twoVMs fall in the same pending length. This Map should contain pending execution time as key and it's associated VMs as a value.

(a) Sort the VMMap by the Pending Execution Time of each VM

(3) Re-arrange the incoming Jobs based on the length & priority of the Jobs.

(a)SorttheJobSubmittedListbasedonlength&priority.

(4) Initiate the vmIndex, jobIndex variable &totalJobs

(a) Set vmIndex = 0

(b) Set totalJobs = length of JobSubmittedList

(c) Set totalVMsCount = size of VMMap

(d)SetjobIndex=0

(e) Set jobToVMratio = totalJobs/totalVMsCount

(5) Assign the incoming jobs to the VMs based on the least Pending Execution Time in the VMs & its processing capacity.

(a) While (true)

Set job = JobSubmittedList.get(jobIndex)

Set jobLength = lengthOf(job)

Set newCompletiontimeMap = EmptyMap

For startNumber from 0 by 1 to totalVMsCount do{

Set vm = VMMap.getValue(startNumber)

Set probableNewCompTime = jobLength/$\mathcal{C}$Vm+VMMap.getKey(startNumber)

newCompletiontimeMap.add(probableNewCompTime, vm

SortByCompletionTime(newCompletiontimeMap)

Set selectedVM = newCompletiontimeMap.getValue(0)

selectedVM.assign(job)

For startNumber from 0 by 1 to totalVMsCount do{

Set vm = VMMap.getValue(startNumber)

If (vm equals selectedVM)

Set currentLength = VMMap.getKey(startNumber)

Set newCurrentLength = currentLength + newCompletiontimeMap.getKey(0)

VMMap.removeItem(startNumber)

VMMap.add(newCurrentLength, vm)

EndIf

}

sortByCompletionTime(VMMap)

Increase the jobIndex by 1

If (jobIndex equals totalJobs)

Break

(b) End While

(6) Remove all the assigned Jobs from the JobSubmit

Algorithm2.2: IWRR Dynamic Scheduling algorithm (Devi & Uthariaraj, 2016)

However the scheduling algorithm for scheduling dependent tasks is inefficient due to random selection in determining which task should to be scheduled next and the current status of job is lost during job migration.

Xie et al., (2016)proposed a scheduling algorithm for cloud computing based on the driver of dynamic essential path (DDEP). This algorithm uses a priority approach for the predecessor-task layer to fix the issue of constraint relationships between task nodes. The approach gives distinct priority values to each task nodeaccording to the scheduling order of task node.

The dynamic essential long path strategy was proposed to tackle the issue of scheduling order in which task nodes have the same priority value. This approach calculates the dynamic essential path of the pre-scheduling task nodes based on the actual computation cost and communication cost of task node in the scheduling process. The task node that has the longest dynamic essential path is scheduled first.

Mishra & Patidar (2017) Proposed an Improved Weighted Round Robin (IWRR) method based on efficient load balancing in cloud computing, the authors proposed an improved weighted round robin method which will use multiple processing elements in the participating heterogeneous VMs along with the heterogeneous multiple processing elements capable jobs with distributed computing capabilities and load balancing also applies at the time of transferring the state of jobs between the VMs at the job migrations. However they adapted the work of (Devi & Uthariaraj, 2016)and only one processing element was usedin configuring setup parameters.

2.6 Gap in the literature

In recent years, researchers have proposed many dependent task scheduling algorithms for heterogeneous systems. These algorithms consider both the task node itself and thecommunication cost between task nodes, such as CPOP (Chun et al., 2006),NEFP(Liang, et al. 2009).

In cloud computing, scheduling is one of the most significant functions. Efficient task scheduling mechanism can satisfy the needs of users, and enhance the use of resources, thusimproving the cloud computing setting general performance. However, most of the scheduling algorithm in cloud computing give more emphasis on how to effectively and efficiently optimize the load balancing algorithm neglecting the complexity of the scheduling algorithm (Xie et al., 2016).

This research work willenhance the efficiency of the task dependent scheduling algorithmin cloud computing environment proposed by (Devi & Uthariaraj, 2016) .

CHAPTER THREE

METHODOLOGY

3.1 Proposed System Architecture

The system architecture described how the components of the system are integrated. There may be many nodes in a cloud and these nodes can be situated in various places. But for the purpose of this research, the experiment setup considers a single datacenter broker. The datacenter broker functions as the system access point and all client requests enter the cloud environment via the broker. In this experiment setup the datacenter broker has several nodes (datacenters) and multiple hosts on each datacentre and also multiple VMs on each host (Devi & Uthariaraj, 2016). The system implementation consists of three major modules

i. Scheduler
ii. Dependent Task scheduler
iii. Resource monitor

3.1.1 Scheduler

The role of the scheduler is to find the most appropriate VM in which the task should be assigned based on the scheduling algorithm that is applied to it. The tasks are allocated to VMs according to the loadon each VM and the length of the task. This is the same with the static/dynamic scheduler defined in (Devi & Uthariaraj, 2016). However in this work a single centralized scheduler is used.

3.1.2 Dependent Task Scheduler

A user gives job request through an interface which consists of multiple dependent tasks. In workflow simulation software, graph is constructed for these dependant tasks, where nodes represents the tasks to be scheduled and edges their dependencies. The task dependent scheduler define the execution order (child tasks must execute before parent tasks) based on the algorithm applied to it. `

3.1.3 Resource monitor

The Resource monitor interact with all resource probers of VMs and gathers present load on each VM, VM capacities, and the tasksin the execution / waiting queues in each VM to decide the suitable VMs to the tasks, which is adapted from the work in Devi & Uthariaraj, (2016)

The system architecture is shown in the figure 3.1.

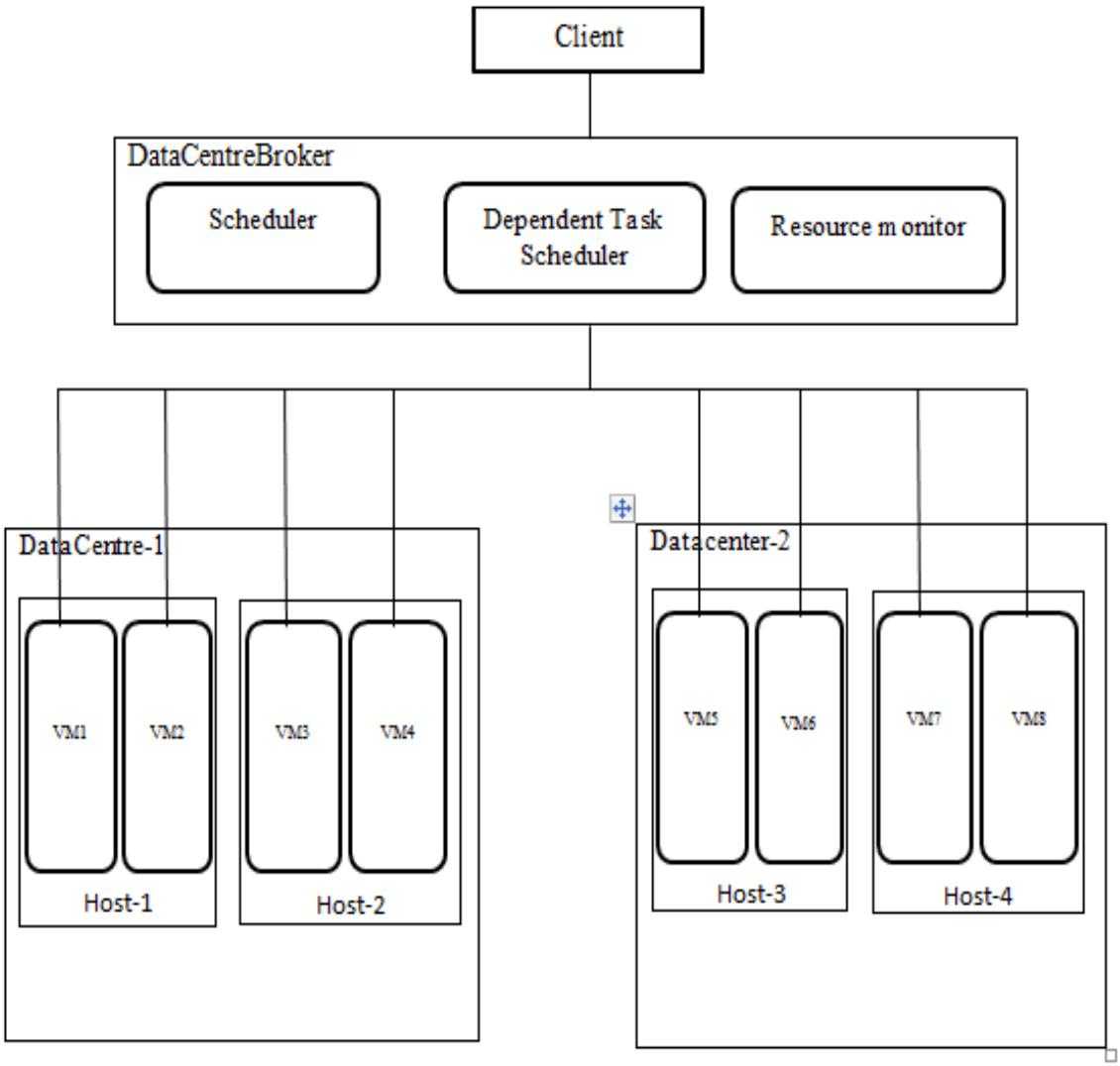

Figure 3.1: System Architecture

3.2 System Model

The proposed method adopts the graph to represent the dependent tasks in the scheduling process. Let G = {V,E} be a DAG, where V = $\{V_1, V_2, V_3, \ldots V_n\}$ represent a set of task node on the graph and *n* the number of task nodes. A directed edge E = $e_{vi,vj}$ where $vi, vj \in V$represent the execution order, meaning task V_j is the descendent of task V_i and under the regulation order, task V_j must finish execution before commencing the execution of task V_i. In other to determine the execution order of the tasks, a priority P is assigned on each node of the DAG based on the total number of its descendent and Run Time (RT). In this model, the path with the maximum total number of descendent run time are first explored and the task node with a no descendent is first removed to be scheduled.

For example, assumetable 3.1is a tabular representation of ten dependent tasks to be scheduled.

Table 3.1: Dependent Tasks

Task	T_1	T_2	T_3	T_4	T_5	T_6	$T_{7`}$	T_8	T_9	T_{10}
Run Time(ms)	2	3	1	2	4	3	5	2	1	4
Precedence Task		T_1	T_1	T_1	T_2	T_2, T_3, T_4	T_3	T_4	T_3, T_4	T_5, T_6, T_7 T_8, T_9

In our model, each task node maintain three attributes(A,B,C), whereA defines the run time of the task node, B is the number of tasks that depend on that task node and C is the total runtime of the dependent tasks. The attribute are computed in the table 3.2.

Table 3.2: Dependent Tasks with their attributes

Task	T_1	T_2	T_3	T_4	T_5	T_6	$T_{7`}$	T_8	T_9	T_{10}
Run Time(ms)	1	2	3	1	2	1	1	2	3	2
Precedence Tasks		T_1	T_1	T_1	T_1	T_1	T_2, T_3	T_3, T_4, T_6	T_5, T_6	T_7, T_8, T_9
No. of Descendent tasks	9	3	4	4	1	1	1	1	1	0
Total Run time	25	11	13	10	4	4	4	4	4	0

From the Table3.2 the DAG representation is shown in the Figure3.2.

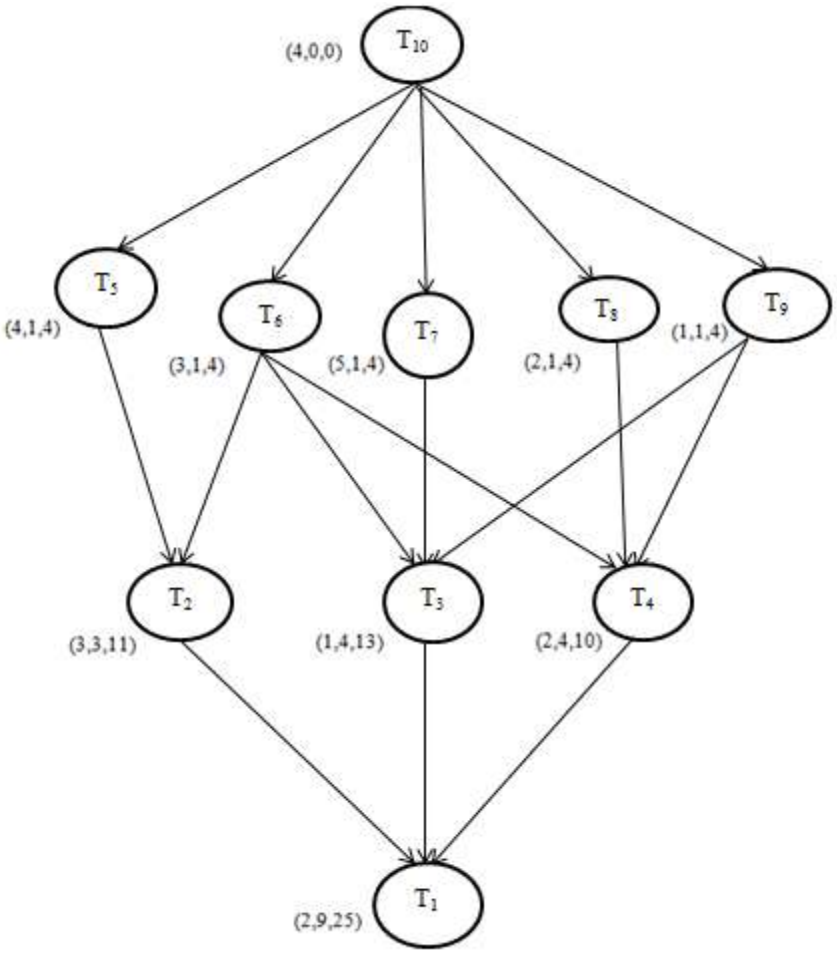

Figure 3.2: An Example of a DAG of tasks with priority

The task dependent scheduler then removes tasks based on the best first search approach and priority then input it to the scheduler.

From the example in the Figure 3.1 the best first search approach will remove the tasks in the following order: $T_{10,}$ $T_{7,}$ $T_{5,}$ $T_{6,}$$T_{8,}$$T_{9,}$ $T_{2,}$ $T_{4,}$ $T_{3,}$ $T_{1.}$

The scheduler gathers the resource information from the resource manager, compute the processing capacity of each VM and then the scheduling algorithm is applied to find the appropriate VM for the job.

For the scheduling problems, let VM = (VM_1 ,VM_2,VM_3……VM_m) be the set of$m$virtual machines, which is supposed to process n tasks. It is assumed that each VM operate on its own resources, all the virtual machines are unrelated and are running in parallel, that is, the VMs do not share their own resources. Nonpreemptive dependent tasks are scheduled to those VMs in which 'n' tasks are allocated to 'm' VMs which is represented as Linear Programing model from (1) to (2) as defined in (Devi & Uthariaraj, 2016).

The processing time of assigning task i to virtual machine j let sayPT_{ij} is defined as:

$$x_{ij} = \begin{cases} 1 \ \ if \ \ task \ \ i \ \ is \ \ assigned \\ \\ 0 \ \ otherwise \end{cases} \qquad (1)$$

Then the linear programming model is given as:

$$\text{Minimize} \quad Z = \sum_{i=1}^{n} \sum_{j=1}^{m} PT_{ij} x_{ij} \qquad (2)$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_{i,j} = 1, \qquad j = 1,2,3.....,m$$

$$x_{ij} = 0 \ or \ 1$$

Resource utilization is considered as the optimization criterion for the scheduling.Maximizing the utilization of cloud resources is another important objective. The average resource utilization is defined in (3)(Xhafa & Abraham, 2008):

$$\text{Average utilization} = \frac{\sum_{j \in VMs} CT_{j.}}{Makespan * Number \ \ of \ \ VMs}. \qquad (3)$$

Where Makespan can be expressed in (4)

$$\text{Makespan} = \max \left\{ \frac{CT_j}{j \in VMs} \right\} \qquad (4)$$

Where CT = Completion Time

3.3 Proposed Algorithm

3.3.1 Dependent Tasks Scheduling Algorithm

The dependent task scheduling algorithm is divided into two phases:

a. Priority assignment algorithm: Given an unlabelled graph, a priority is assigned based on the total number of descendent of each node and their run time. Each node in the graph maintained two attributes namely; number of its descendent and the run time of these descendent. The algorithm for assigning the priority is shown in Algorithm 3.1.

```
Algorithm: Labeling (G:graph)
Input: unlabeled graph where nodes represent the task to be scheduled and edges their dependency
Output: A labeled graph
Foreach node N in G
     X[ ]  ← get descendent (N)
    Count = X.length( )
RunTime = 0
    For i = 0 to X.length do
          RunTime = RunTime + X[i].RunTime
N.descendantcount=count
N.desendantruntime= RunTime
Return G
```

Algorithm 3.1: Labelling Algorithm

b. Graph traversal algorithm: Given a labelled graph of tasks a best first search approach is used to remove a task node for scheduling based on the total runtime of the descendant and the priority until all the task nodes are scheduled. A greedy best first search is a search with heuristic that intend to predict how close the end of a path is to a solution, so that the path which are evaluated to be closer to a solution are first extended. The graph traversal algorithm is shown in Algorithm 3.2.

```
Algorithm: PriorityScheduling(G:graph)
Input: A prioritized directed acyclic graph (G:graph)
Output: A queue of tasks to be scheduled ([ ])

PriorityQueue [ ]
CurrentNode ← G:root
While G is not empty do {
    If  count(CurrentNode.Neighbor ) = 0
        P = path(CurrentNode)
        PriorityQueue [ ] ← CurrentNode
        G.Remove(CurrentNode)
            For each X in P do {
                X.Priority ← X.Priority-1
                X.TotalRunTime = X.TotalRunTime – CurrentNode.RunTime
            }End For
            If G is not empty do {
                CurrentNode = G.root
            }End If
    Else
        C [ ] ← CurrentNode.Neighbors
        Max ← 0
        For i = 1 to C [ ].length-1{
            If (C[i].TotalRunTime> C[Max].TotalRunTime) ∨
            ((C[i].TotalRunTime=C[max].TotalRunTime) ∧ (C[i].Priority>C[Max].Priority))
                Max ← i
        }End For
    }End If
    CurrentNode ← C[Max]
}End While
}End
```

Algorithm 3.2: Graph Traversal Algorithm

The flow chart of the dependent task scheduling algorithm is shown in Figure 3.3.

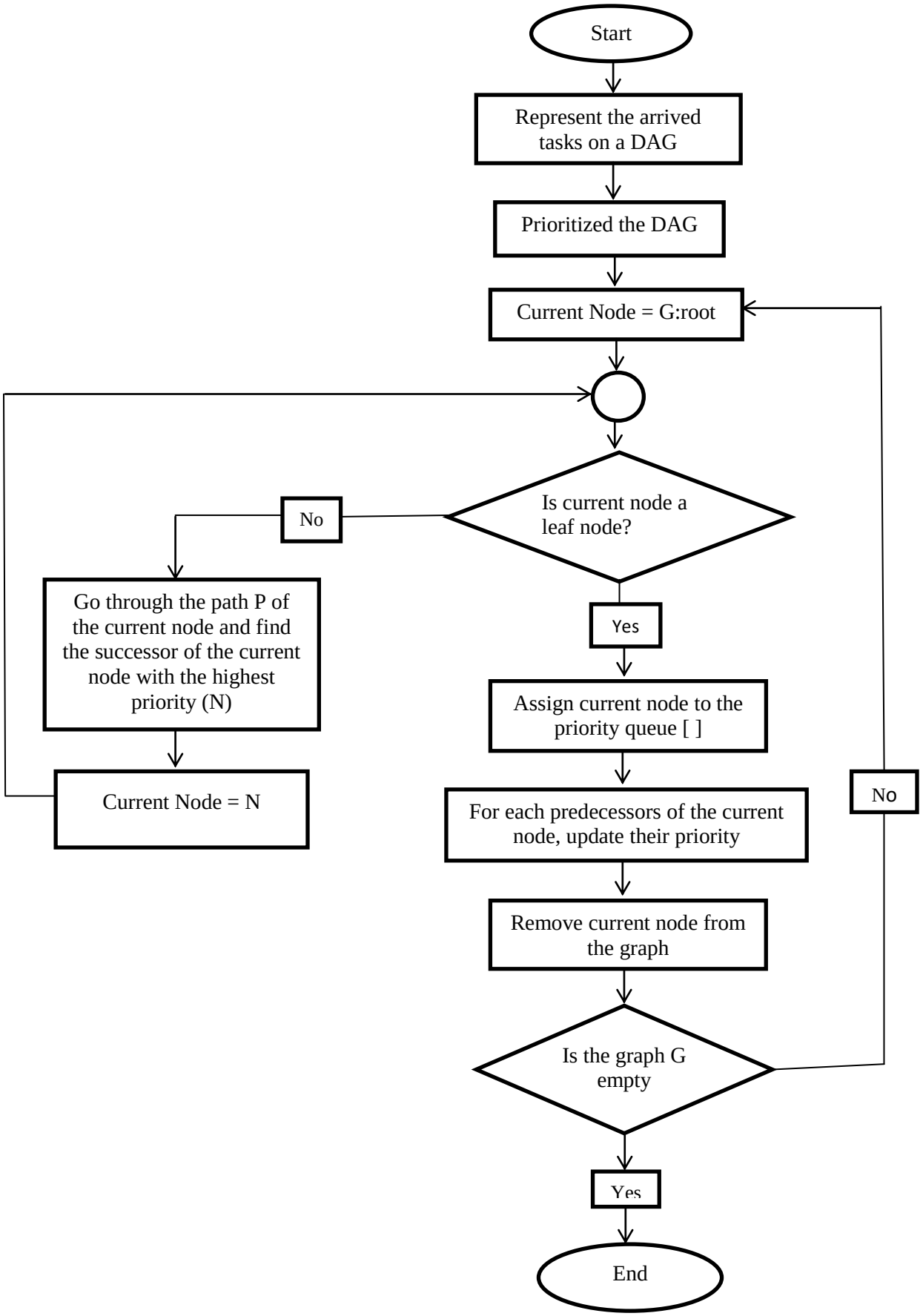

Figure 3.3: Flow chart of dependent task scheduling algorithm

3.4 Setup Configuration

This work adopted the same setup configurations used in Devi & Uthariaraj, (2016). The cloud setup configuration details are shown in the table 3.3.

Table 3.3: Cloud Setup Configuration details

S/N	Item	Values	Functions
1	Data center	1	Data center having the physical hosts in the test environment.
2	Number of host in the data center	500 hosts (200 Nos-4-CoreHyperThreadedOpteron270, 200 Nos-4-Core HyperThreadedOpteron2218 and 100 Nos 8-Core HyperThreaded XeonE5430)	Number of physical hosts used in the experiment
3	Number of process elements	8/8/16	Number of executing elements in each of the hosts. The host has 8 or 16 processing elements
4	Processing elements processing capacity	174/247/355 MIPS	Each host has any one of the processing capacities
5	Capacity of the Host RAM	16/32 GB	Each host has any of these RAM memories
6	Number of Virtual machines	10 to 100 with an increment of 10	Number of virtual machines used in the experiment
7	Number of processing element to virtual machine	1	Processing element allocated in each VM
8	VM's PE processing capacity	150/300/90/120/93/112/105/225	Virtual machine's processing capacity
9	Virtual machine RAM capacity	1920 MB	The RAM's memory capacity of the VM
			The operating system that manage the VMs.

10	Virtual machine manager	Xen	
11	Number of processing element in tasks	1	The job/task's maximum accesible processing element
12	Number of tasks	Montage_1000	The tasks used in the experiment
13	Task length/instructions	225 to 1393	Variation of the task length

Montage is an astronomy application that is used to construct large image mosaicsof the sky. Input images are reprojected onto a sphere and overlap is calculated for each input image. The application reprojects input images to the correct orientation while keeping background emission level constant in all images. The images are added by rectifying them to a common flux scale and background level. Finally the reprojected images are co-added into afinal mosaic. The resulting mosaic image can provide a much deeper and detailed understanding of the portion of the sky in question. The size of the workflow depends on the number of images used in constructing the desired mosaic of the sky(Berriman, et al., 2004).

3.5 Performance Metrics Analysis

The criteria for evaluating the performance of the proposed algorithm are theturnaround time and waiting time of the scheduled tasks.

$$Waiting\ \ time\ (WT) = ST(i) - AT(i) \qquad (7)$$

where ST is the start time and AT is the arrival time of tasks.

$$Turnaround\ \ Time\ \ (TAT) = EX(i) + WT(i) \qquad (8)$$

where EX is the execution time and WT is the waiting time of tasks.

Graphs were plotted to represent the results of the simulation side by side with that of (Devi & Uthariaraj, 2016).

CHAPTER FOUR

RESULT ANALYSIS AND DISCUSSION

4.1 Introduction

The comparison of the proposed scheduling algorithm and the scheduling algorithm implemented inDevi & Uthariaraj, (2016) were performed and the simulation was done using WorkflowSim toolkit in order to evaluate the improvement of the proposed approach. Results of the experiments are analyzed based on waiting time and turnaround time of the tasks.

4.2 WorkflowSim

Workflowsim is atoolkit for simulating scientific workflows in distributed environments. WorkflowSim simulator extends the existingCloudSim simulator by providing a higher layer of workflow management. The WorkflowSim has three components namely: the Workflow planner, Workflow scheduler and Workflow engine. The System was simulated using WorkflowSim version 1.0 and eclipse as the integrated development environment.

4.3 Results and Discussion

The performance of the enhanced algorithm has been analyzed based on the results of simulation done in the WorkflowSim. The classes of the WorkflowSim simulation toolkit have been extended (overridden) to utilize the enhanced algorithm and the result of the waiting time and turnaround time are as follows.

4.3.1 Waiting and Turnaround Time

The scheduling experiments were carried out by maintaining a constant number of cloudlet and setting various values for the number of resources; the number of resources was varied from 10 to 100 with an increment of 10, as similarly implemented in Devi & Uthariaraj, (2016)for comparison.

The result of the waiting time is shown in Table 4.1 and Figure 4.1.

Table 4.1: Combined Waiting Time

S/N	Number of VMs	Existing Waiting Time(ms)	Enhanced Waiting Time(ms)
1	10	5692285	4227786
2	20	2498160	2216715
3	30	1548007	1245643
4	40	1348347	963936
5	50	904562	883608
6	60	787817	637144
7	70	654676	593731
8	80	545615	477089
9	90	512134	475021
10	100	474398	418910

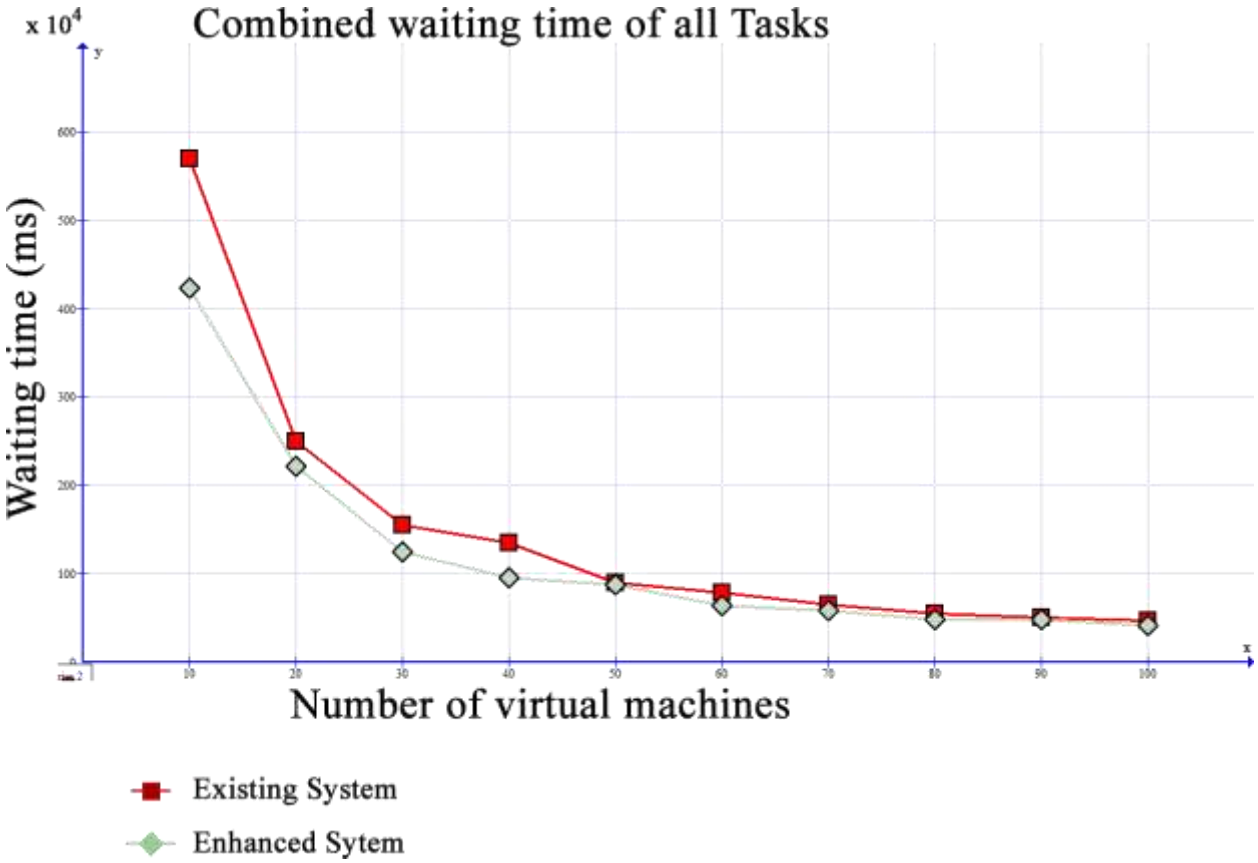

Figure 4.1: Combined Waiting time

Waiting time measures the delay time of task before they starts execution, less waiting time was achieved in the proposed dependent task scheduling algorithm compared with the dependent task algorithm implemented in Devi & Uthariaraj, (2016)by an average of about 18%.

The result of the turnaround time is shown in Table4.2 and Figure 4.2.

Table 4.2: Combined Turnaround Time

S/N	Number of VMs	Existing Turnaround Time(ms)	Enhanced Turnaround Time(ms)
1	10	5790222	4308304
2	20	2584201	2292136
3	30	1630417	1315092
4	40	1427035	1032997
5	50	980401	952558
6	60	862631	703725
7	70	727903	662115
8	80	618772	542126
9	90	586401	544139
10	100	547105	486417

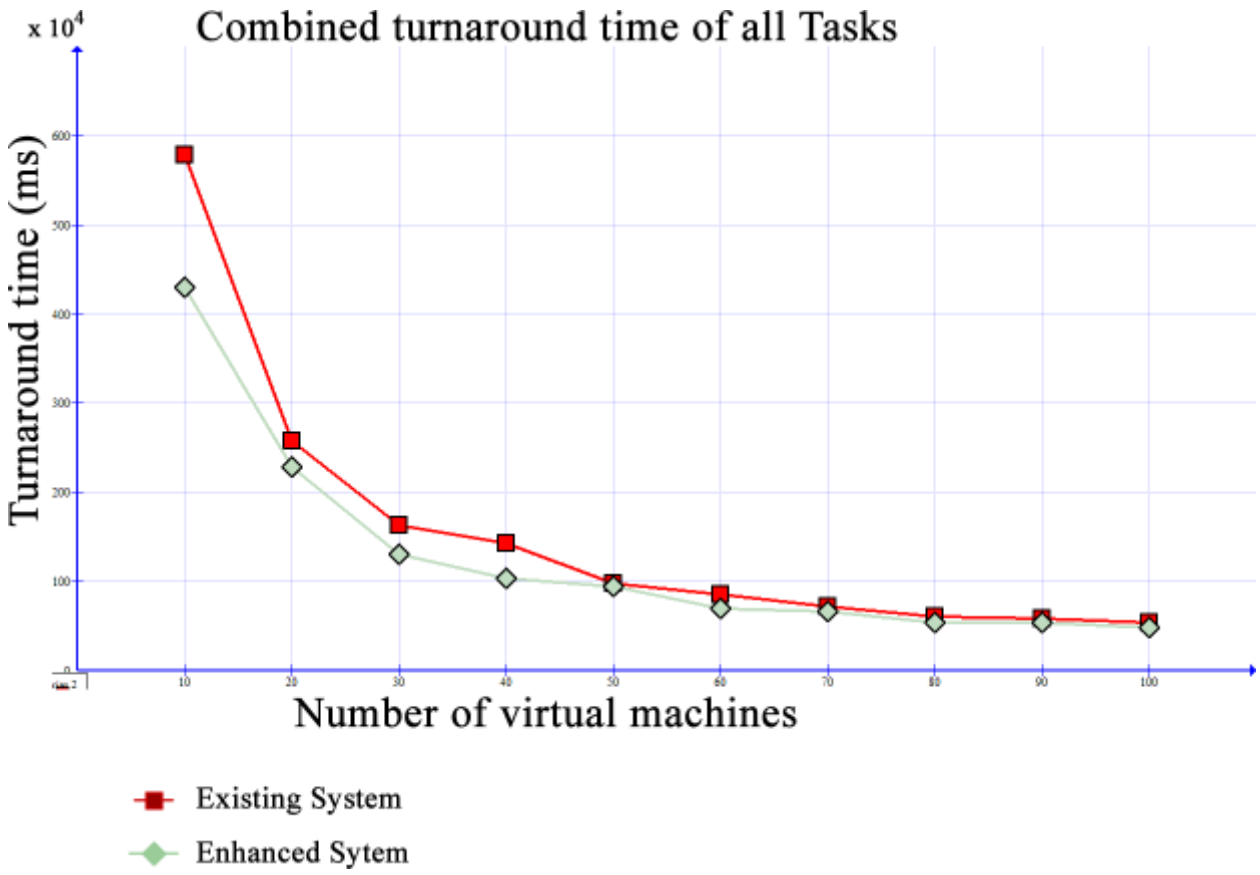

Figure 4.2: Combined Turnaround time

Turnaround time measures the total time a task spends on the system, from submission until completion, less turnaround time was also achieved in the proposed dependent task scheduling algorithm compared with the dependent task algorithm implemented in Devi & Uthariaraj, (2016)by an average of about 19%.

Based on the results obtained, our system out performs that of Devi & Uthariaraj, (2016) when little numberand this is because our technique and algorithms used in determining the order of scheduling the dependent tasks better, with an average improvements of 18% and 19% on waiting time and turnaround time respectively, by using the formula in (1):

$$\frac{Y - X}{Y} * 100 \qquad (1)$$

Where Y is the existing waiting /turnaround time and X is the enhanced waiting/turnaround time.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

This dissertation proposed an enhanced dependent task scheduling algorithm in cloud computing environment in order to improvethe overall performance of the entire system. In this research work, a Directed Acyclic Graph (DAG)was used to represent the arrived task where nodes represent the task to be scheduled and the edges their dependencies, the number of descendant and total runtime were used as heuristic in determining the path to be explored first and best first search approach was adopted to determine the order in which the task nodes are removed from the DAG for scheduling. For the experiment,Montage_1000 were used on 10 to 100 virtual machine with an increment of 10 and the results obtained on waiting time and turnaround time where recorded and analysed.

5.2 Conclusion

The simulation results summarized in Table 4.1 and Figure 4.1showed that keeping the number of cloudlet constant and varying the number of resources, an average improvements of 18% on waiting time was achieved and also in Table 4.2 and Figure 4.2showed that keeping the number of cloudlet constant and varying the number of resources, an average improvements of19%on turnaround time was achieved. These results indicate the potential usefulness of our research contribution to practical cloud computing environments.

5.3 Recommendation

The recommendations for future work are as follows

a. The proposed system does not consider migrating tasks from heavy loaded virtual machine to less loaded virtual machine (load balancing)
b. Multiple processing elements can be considered in the virtual machines
c. A study can be made to enhance the scheduling algorithm

References

Abilijit, A. R., & Apte, S. S. (2013, July). A Performance Analysis of Task Scheduling Algorithms Using Qualitative Parameters. *International Journals of Computer Application, 74*(19): 33-38.

Alharba, Y., & Yang, k. (2016). Optimizing Jobs' completion time in Cloud System during Virtual Machine Placement. *International Conference on BigData and Smart City, 1*: 1-6. Oman.

Ananth, A., & Chandrasekan, K. (2015). Cooperative Game Theoretic Approach for Job Scheduling in Cloud Computing. *Computing and Network Communications, 2*(1): 147-156.

Angeles, S. (2014, January 20). *Virtualization vs Cloud Computing.* Retrieved June 1, 2019, from Business news daily: http://www.businessnewsdaily.com/579-virtualization-vs-cloud-computing.html

Anthony, T. V., Toby, J. V., & Elsenpeter, R. (2010). *Cloud Computing: Apractical Approach.* HILL: TATA McGRAW.

Arora, M., Das, S. K., & Biswas, R. (2002). A Decentralized Scheduling and Load Balancing Algorithm for heterogenous Grid Environment. *Proc. of International Conference on Parallel Processing Workshop*, 400-505. Vancouver British Canada.

Arya, L. K., & Verma, A. (2014). A Survey- Workflow Scheduling Algorithm in Cloud Environment. *Recent Advances in Engineering and Computational Science, 3*(1): 1-4.

Babu, L. D., & Krishna, P. V. (2013). Honey bee behaviour inspired load balancing of tasks in cloud computing environment. *Applied Soft Computing Journal, 13*(5): 2292-2303.

Balagoni, Y., & Roa, R. R. (2014). Importance of Load Balancing in Cloud Computing Environment: A Review. *International Journal of Advanced Trends in Computer Science and Engineering, 3*(5): 77-82.

Basker, R., Uthariaraj, V. R., & Devi, D. C. (2014). An enhanced scheduling in weighted round robin for the cloud infastructure services. *Internatonal Journal of Recent Advanced in Engineering and Technology, 2*(3): 81-86.

Ben Alla, S., Ben Alla, H., Touhafi, A., & Ezzati, A. (2019). An Efficient Energy-Aware Tasks Scheduling with Deadline-Constrained in Cloud Computing. *Internatioanl Conference on Cloud Computing Technologies and Applications, 8*: 1-7. Brussels, Belgium.

Berriman, G., Deelman, E., Good, J., Jacob, J., Katz, D., Kesselman, C., et al. (2004). Montage: A Grid-enabled Engine for Delivering custom Science-grade Mosaics on demand. *In SPIE Conference on Astronomical Telescopes and Instrumentation*, 1-10. Callifornia.

Bhaskar, P. R., Eunmi, C., & Lan, L. (2009). A Taxonomy and Survey of Cloud Computing System. *Fifth International Joint Conference on INC, IMS, IDC*, 44-51. Seoul, South Korea.

Casavant, D., & Kuhl, J. (1988, February). A taxonomy of scheduling in General Purpose Distributed Computing Systems. *IEEE Trans. on Software Engineering, 14*(3): 141-154.

Chun, H. L., Chai, F. L., Kuan , C. L., & Chao, C. W. (2006). A Dynamic Critical Path Duplication Task Scheduling Algorithm for Distributed Hetrogenous Computing Systems. *proceedings of the 12th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 1*: 1-8. Minneapolis, MN USA.

Devi, D. C., & Uthariaraj, V. R. (2016). Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Task. *Hindawi Publishing Corporation The Scientific World Journal, 2016*(1): 1-14.

Dillan, T., Wu, C., & Chang, E. (2010). Cloud Computing: Issues and Challenges in. *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 27-33. Perth.

Ertual, L., Singhal, S., & Saldamli, G. (2010). Security Challenges in Cloud Computing in Security and Management. *Proceedings of 2010 International Conference on Security and Management*, 36-42. Las Vegas, Nevada USA.

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree compared. *Grid Computing Environment workshop, 1*(1): 1-10.

George, A., & muthulakshmi, P. (2011, April). A review of scheduling policies and algorithms in Grid Computing. *International Journal of Research and Reviews in Computer Science (IJRRCS), 2*(2): 280-294.

Gouda, K. C., Patro, A., Dwivedi, D., & Bhatt, N. (2014, June). Virtualization Approaches in Cloud Computing. *International Journal of Computer Trends and Technology (IJCTI), 12*(4): 161-166.

Harjit, S. L., & Gurdav, S. (2011). Cloud Computing future framework foe e-management of NGO's. *IJoAT, 2*(1).

Kaur, A. K., & Dhindsa, K. S. (2017). Analysis of Task Scheduling Algorithms using Cloud Computing. *Grenze International Journal of Engineering and Technology (GIJET), 5*(4): 1-6.

kaur, R., & luthra, P. (2014). Load balancing in cloud computing. *proc. of ACEEE Int. Conf on Recent Trends in Information, Telecommunication and Computing, ITC, 4*: 1-7. India.

Krishnadoss, P., & Jacob, P. (2018). OCSA: Task Scheduling Algorithm in Cloud Computing Environment. *International Journal of Intelligent Engineering and Systems, 11*(3): 271-279.

Liang, T. L., Ching, W. C., Hung, Y. C., Chih, C. T., & Kun, C. P. (2009). A Non-critical path Earliest-Finish Algorothm For Interdependent Task in Hetrogenous Computing Environment. *In Preceedings of the 11th IEEE International Conference on High Performance Computing and Communication*, 603-608. Seoul, Republic of Korea.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., et al. (2011, September). *Cloud Computing Refrence Architecture.* Gaithersburg: National Institute of Standard Technology (NIST).

Livny, M., & Melman, M. (2010). Load Balancing n Homogenous Broadcast Distributed systems. *Proceedings of ACM Computer Network: performace symposium*, *2*(1): 47-55. Maryland, USA.

Malhotra, L., Agrawal, D., & Jaiswal, A. (2014). Virtualization in Cloud Computing. *Information Technology and Sofware Engineering, vol. 4* (1): 1-3.

Mathew , T. K., Chandra, S., & John, J. (2014). Study and analysis of various task scheduling algorithms in the cloud computing environment. *IEEE International Conference on Advances in Computing, Communication and Infor-matics (ICACCI), 2*(1): 24-27. Delhi, India.

Mishra, S., & Patidar, K. (2017). An IWRR method based on efficient load balancing in cloud computing. *International Journal of recent Trends in Engineering and Research, 3*(1): 46-54.

Nahir, A., Orda, A., & Raz, D. (2016). Replication-Based Load Balancing. *IEEE Transition on Parallel Distrubuted System, vol. 27*(2): 494-507.

Nidhi , A. (2017). Review on Task Scheduling Algorirhms in Cloud Computing Environment. *International Conference on Emerging Trends on Engineering and Technology*, *8*(1): 401-403. Punjab, India.

Pasha, N., Argarwal, A., & Rastogi, R. (2014, May). Round Robin Approach for Vm Load Balancing Algorithm in Cloud Computing Environment. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(5): 34-39.

Peter, M., & Timothy, G. (2011). The NIST Definition of Cloud Computing (Draft). *NIST special publication 800-145*, 1-6.

Rajwinder, K., & Pawan , L. (2014). Load Balancing in Cloud Computing. *Proceedings of International Conference on Recent Trends in Information Telecommunication and Computing (ITC)*, 1-7. Delhi, India.

Rouse, M. (2015, May). *Search Networking*. Retrieved March 25, 2019, from Tech Target: https://searchnetworking.techtarget.com/definition/throughput

Ryan, S., & Jiangchuan, L. (2012). Understanding the Impact of Denial of Service attacks on Virtual Machines. *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*, 1-8. Coimbra, Portugal.

Shimpy, E., & Sidhu, J. (2014, September). Different Scheduling algorithm in different cloud environment. *International Journal of Advanced Research in computer and Communication Engineering, vol. 3*(9): 1-7.

Singh, R. M., Paul, S., & Kumar, A. (2014). Task Scheduling in Cloud Computing: Review. *International Journal of Computer Science and Information Technologies (IJCSIT), 5(6)*: 7940-7944.

Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I. (2009). Virtual Infrastructure Management in private and hybrid clouds. *IEEE Internet Computing, 13* (5): 14-22.

Tracy, D. B., Howard, J. S., & Nuah, B. (2001). A comparison of Eleven Static Heuristic or Mapping of Independent Task onto Hetrogenous Distributed Computing System. *Journal of Parallel and Distributed Computing, 61*(1): 810-837.

Tupakula, U., & Varadharajan, V. (2011). TVDSEC: Trusted Virtual Domain Security. *Institute of Electrical and Electronic Engineers (IEEE), 2*(3): 57-63.

Vivek, M., Abhilasha, J., & Vivek, P. (2017, April). Task Scheduling in Cloud Computing. *International Journal of Advance Research in Computer Science, 8*(3): 821-825.

Wang, L., Gregir, L., Marcel, K., & Jie, T. (2008). Cloud Computing: A Perspective Study. *Andvances of Distributed Information Processing, 28*(2): 137-146.

Wang, S. C., Yan, K. Q., Lio, W. P., & Wang, S. S. (2010). Toward a Load Balancing in a three level Cloud Computing Network. *In proc, 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, 108-113. Chengdu, China.

Waqas, A., & Awais, S. Q. (2018). Analysis of Interactive Utilization of CPU between Host and Guests in Cloud Setup. *Computer Science and Engineering, 8*(1): 7-15.

Xhafa, F., & Abraham, A. (2008). Metaheustics for Grid Scheduling Problems. *Springer*, 1-37.

Xie, Z., Xia, S., & Xin, Y. (2016). A Scheduling Algorithm for Cloud Computing System Based on the Driver of Dynamic Essential path. *PLOS one, 11*(8): 1-19.

Xioa, Z., Song, W., & Chen, Q. (2013). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems, 24* (6): 1107-1117.