International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307

OPEN ACCESS

Available Online at :www.ijsrcseit.com

doi : https://doi.org/10.32628/IJSRCSEIT

Optimizing Efficient and Accurate Real-Time Process Scheduling in Cloud Computing Systems

Avinash Kumar[1], Neelesh Ray[2]

[1]M Tech Scholar, [2]Associate Professor

Department of Computer Science & Engineering, Millennium Institute of Technology and Science, Bhopal, Madhya Pradesh, India

ARTICLEINFO

Article History:

Accepted: 10 Oct 2023

Published: 22 Oct 2023

Publication Issue

Volume 9, Issue 5

September-October-2023

Page Number

343-355

ABSTRACT

One of the research problems that has to be solved is the scheduling of real-time jobs on the cloud. This problem requires taking into account the appropriate machines as well as the amount of time needed to complete the activities. The challenge with matching Realtime tasks to machines is that, assuming there are a certain number of active hosts, there are also a certain number of virtual machines (VMs) in each host. The greatest conceivable number of virtual machines that may be scheduled for a single job is (pq). In the event that we need to scheduler tasks, there are a total of (p+q)r possible combinations. Therefore, the challenge of scheduling jobs is an NP-hard problem. Real-time tasks have strict completion time constraints, and the only way for them to be helpful is if they are finished before the deadline; otherwise, they are useless. If it does not serve a helpful purpose, then it will not be accepted. The Earliest Deadline First (EDF) method is a well-known technique for scheduling activities that take place in real time. The Event Driven Framework (EDF) is a scheduling technique that assigns priorities in a dynamic manner with regard to the due dates of the tasks. There are three different types of real-time tasks: periodic, sporadic, and aperiodic. In order to evaluate the effectiveness of various scheduling methods, we have employed both aperiodic and periodic models. In most cases, the EDF Scheduler will organise the tasks in such a way that they are assigned to the free machine that is currently available. This is done without taking into account whether or not the tasks on that machine will reach the deadline. Before allocating a job to a machine, this work determined the amount of time required to do a task on all of the free machines that were available. I have utilised three distinct approaches in order to delegate the responsibility. First Fit, Best Fit, Worst Fit. Here When a task is considered fit for task, it

Copyright © 2024 The Author(s): This is an open-access article distributed under the terms of the Creative Commons Attribution **4.0 International License (CC BY-NC 4.0)**

indicates that it will successfully finish its execution on the specified machine before the deadline. The scheduling of aperiodic activities as well as periodic tasks can make use of these three approaches, in addition to the Basic EDF. The performance of the First Fit EDF (FFE), Best Fit EDF (BFE), and Worst Fit EDF (WFE) approaches has been examined by our team. The simulation was carried out in-house using the MATLAB simulator, and the performance metrics that were taken into account were the guarantee ratio (GT), virtual machine utilisation (VU), and throughput (TP). It has been demonstrated through the results of simulations that the FFE, BFE, and WFE algorithms work more effectively than the Basic EDF method.

Keywords: Cloud Computing, Real-Time Scheduling, virtual machines, aperiodic scheduling.

I. INTRODUCTION

Provisional Numbering System in most cases, a real time system is a controlling system, and it is frequently embedded inside other pieces of machinery in such a way that it is not immediately clear that it is even a computer. It takes in information from its surroundings, analyses that knowledge, and then produces a reaction to what it has learned. Real-time systems are capable of reacting, responding, and adapting their behaviour in response to changes in the environment in which they are positioned. A system that operates in real time gives the impression that the amount of time it takes for it to respond is substantial and crucial. One of the most widespread misconceptions about real-time systems is the idea that they should always have a lightning-quick reaction time to events. However, this is not always the case. A response from a real-time system need only be timely, however the precise meaning of "timely" will change significantly depending on the application being used. It can be seconds or even minutes. A real-time system has a guaranteed, calculated (we use the phrase deterministic), worst-case reaction time to an event that is under its control. This response time is known as the real-time response time. Real-time systems are those systems in which the validity of the results generated by the system is dependent on both the logical conclusion of the computation and the moment at which the results are produced. Real-time systems are also known as continuous-time systems. In a system that operates in real time, the accuracy of the results depends not only on the chronological aspects of those findings, but also on the functional aspects of such results. Timeliness on timing constraints, often known as deadline constraints, is the primary metric used to evaluate the performance of real-time systems. Speed and average case performance are less important. By utilising scheduling strategies, schedulers are able to provide the greatest possible run time control for algorithms. Scheduling strategies are in direct competition with the completion of real-time activities and occupy the available computing resources. We are able to test the services' promise of being able to fulfil the deadline by making use of the methods for stimulability testing. Tests of stimulability are timed events.

complex and time-consuming, but ultimately beneficial to the resources since it increases their quality. In the real-time systems, the services with a significant amount of behavioural variation in their run times and the tests should be able to ensure that the services will not miss their deadlines in all of the potential runs.The overarching scheduling issue involves arranging jobs in such a way that they may be completed while meeting a number of different requirements. The readiness time, execution time, deadline, and resource requirements of a job are the elements that define its characteristics. During the process of carrying out the duties, the process could or might not be interrupted. If the work is interrupted, then the scheduling is said to be preventative. The limits that the tasks have to adhere to are determined by the precedence relationship. The beginning of the execution of a job will only occur once its predecessor tasks have been finished being carried out. The quantity of resources that are available serve as a defining factor for the tasks that are carried out. Real-time scheduling has several objectives, the most important of which are to achieve the deadline, enhance utilisation, cut down on context switches caused by pre-emption, and cut down on communication costs. The term "cloud computing" refers to a type of dynamic service provider that makes use of very vast virtualized and scalable resources over the internet. The term "cloud computing" refers to the collection of computer and communication resources across several data centres that are then made available to a wide variety of users simultaneously. Computing in the cloud is now the field of Information Technology (IT) with the most rapidly developing paradigm. Time is the primary criterion that is used to evaluate the quality of any real-time apps or services. The real time services provided through the internet will ensure that all of the jobs fulfil their respective deadlines, much like other real time systems

II. RELATED WORK

In the discussion that follows the Literature Review, the scheduling methods that are employed in those papers are discussed. Xiaomin Zhu et al.[11] The challenge of energy-aware scheduling for separate, aperiodic real-time workloads in virtualized clouds was studied in this study. The stimulability of real-time jobs in the system is one of the goals of the scheduling process, along with the conservation of energy. This paper presents a novel energy aware scheduling algorithm for real time tasks called EARH (Energy Aware Rolling Horizon). In this algorithm, a rolling horizon policy was used to enhance the stimulability of the system. To accomplish the goals, the virtualization technique and a rolling horizon optimisation scheme were used. If this modification is applied to any of the existing real-time scheduling algorithms, such as EDF (Earliest Deadline First), RM (Rate monotonic), and LLF (Least Laxity First), then the behaviour of these algorithms becomes much improved. It is applied for EDF here in this section of the article. Cecilia Edelin et al. [1], In this article, a non-pre-emptive scheduling technique known as Clairvoyant EDF (CEDF) was developed.

The CEDF employs a technique called "look ahead" to detect when a work needs to be postponed to a later time and when leisure intervals need to be inserted. CEDF and EDF both run with the same level of temporal complexity, however CEDF has the ability to schedule up to one hundred percent more task sets in most situations. It is a given that all of the job sets that EDF schedules will be scheduled by it. Jayant R. Haritsa et al. [2], This study makes use of EDFas to provide a novel priority assignment method that has been given the name Adaptive Earliest Dead-line (AED). This algorithm functions as a feedback control mechanism that recognises overload circumstances and adjusts transaction priority assignments appropriately. Shuo Liu et al.[4],

In this article, we describe a utility accrued strategy that takes into consideration both the gain that can be achieved by finishing a real-time work on time and the cost that may be incurred by abandoning or giving up on the assignment. This scheduling algorithm makes execution choices based on high profitability, and it also eliminates jobs that might otherwise result in a significant penalty. These approaches can be used in conjunction with any real-time task scheduler. It is applied for EDF here in this work that we are discussing. R. Santhosh and others, et al. [7], This study offers an online, pre-emptive scheduling with task migration method for cloud computing environment is proposed in or-5. [8], This paper presents an online, pre-emptive scheduling with When it comes to cloud computing, there have been a lot of different scheduling algorithms developed for energy efficiency. The problem of selecting physical machines was addressed by Leverich et al. [1], who presented their method.

for the purpose of carrying out the activities while simultaneously fulfilling the power consumption criteria in a cluster. The solution that is being offered is an approach to the scheduling model that will choose a physical machine from each cluster. Following the completion of the data collection process for each cluster, the suggested model is now working on a scheduling technique to schedule the physical machine to attend each cluster.

In cloud computing, virtualization is one of the most important techniques that can be used to make maximum use of the available resources. The dynamic allocation of resources to cloud users has been accomplished through the development of a number of different virtual machine scheduling algorithms [2]. These allocation methods may be broken down into two distinct categories: the first phase involves the allocation of virtual machines onto actual machines, and the second step involves the assignment of physical machines to virtual machines. A technique was presented by Lang et al. [3] to execute all workloads concurrently in all physical machines in order to conserve more energy depending on the incoming workloads.

This was accomplished with the assistance of a scheduling strategy through the use of scheduling. In order to lessen the amount of power that is consumed as a result of the workload of virtual machines, the suggested technique is working to solve the mapping and allocation problem. DVFS, which stands for dynamic voltage frequency scaling, is the primary method utilized in cloud computing with the purpose of lowering the amount of power that is consumed. With the primary purpose of optimizing power consumption, the DVFS approach is mostly utilized in datacenters for the purpose of making trade-offs between performance and processor power. Wang et al. [4] addressed the problem of how to lower the frequency of the CPU in order to achieve energy efficiency in scheduling policies. They did this with the use of the DVFS approach. In order to do this, the suggested scheduling strategy would reduce the amount of carbon emissions while simultaneously improving the income of cloud providers. RIRA, which stands for Relaxation Iterative Rounding Algorithm, was initially presented by Wu and Li [5] with the use of the DVFS optimization approach. Through this, the primary focus was on reducing the amount of energy that was used.

An approach for scheduling was presented by Watanabe et al. [6] in order to cut down on the amount of energy that is consumed by clusters. The algorithm that was developed offered a scheduling strategy for the implementation of resource allocation. This survey is presented by K.D. Kumar and colleagues [7] with the purpose of examining the roles that machine learning algorithms play in cloud computing systems. An explanation was provided by the authors on machine learning algorithms and the roles that they play in cloud computing systems. Each algorithm that is utilized for the purpose of estimating the resources that will be utilized by cloud apps.

As a cloud provider, one of the most essential tasks is to make predictions about the resources that will be used by cloud apps. During the same time, the challenge of accurately projecting results is likewise a tough one. The efficiency of cloud environments, on the other hand, is improved by the use of precise algorithms. The cloud provider is able to accurately arrange the resources to be allocated to the cloud application in advance if they have accurate forecast results. The strategy that was proposed by S. Mohamed and colleagues [8] was designed to optimize the power consumption that is occurring in cloud datacenter maintenance settings. The authors focused on ways to reduce the amount of electricity that was consumed. In most cases, data centers generate a significant amount of carbon dioxide, which might result in the contamination of the surrounding environment. In order to lessen the amount of electricity that is consumed, the authors suggested a "green cloud" strategy. Through the utilization of prediction algorithms, B. Dinh et al. [9] discussed the answer to the problem of lowering the amount of power that is consumed.

III.PROPOSED WORK

The scheduling of real-time jobs with the goal of delivering certain performance criteria, such as Guarantee Ratio, Utilisation of VMs, and Through Put, is the primary focus of this thesis. We made a proposal for the model of the system that would be used for real-time job scheduling, which will be used throughout the thesis. The fundamental objective of the scheduling of tasks in real-time systems is typically to ensure that the system will carry out the work in such a way that a greater number of tasks should achieve their due dates. Real time systems are designed to physically affect change within a certain window of time defined by the user. A real-time system that is comprised of two different types of systems. The first one is a computer that is termed a controlling system, and another one is called a control system for the environment.

The controlling system communicates with the surrounding environment in a manner that is contingent on the information that is readily available. A set of features that are shared by the real time system and the non-real time systems serve to differentiate the two types of systems. The limitations imposed by time The timing restrictions have an effect on the timing behaviour of the tasks, which may be described in terms of their release time and the absolute time or relative actual deadlines of the tasks.

Models, Performance Metrics the most important thing to think about when it comes to real-time systems is how to get the most out of the resources available in cloud computing. Therefore, the tasks are mapped to the virtual machines in such a way that it enhances the system's overall performance while simultaneously increasing its utilisation. As a result of the jobs in real-time services being finished and their execution deadlines being met, the system will see improvements in both its throughput and its efficiency. For the purpose of providing a description of the behaviour of real time distributed systems within the domain, a system model of the RTS, such as the one illustrated in Figure No.3.1, has been developed.

In most cases, a real time system is a controlling system, and it is frequently embedded inside other pieces of machinery in such a way that it is not immediately clear that it is even a computer. It takes in information from its surroundings, processes that knowledge, and then responds to what it has learned.

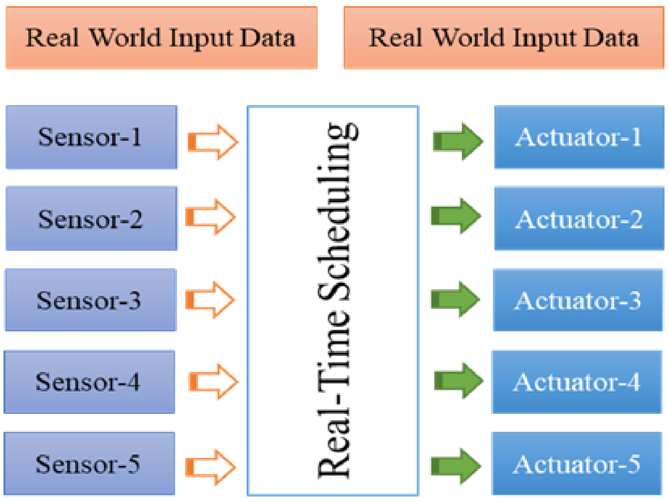

Figure 3.1 A Real-Time System Architecture

computer in the Cloud: Computing in the cloud allocates computer resources to consumers and makes them accessible over the internet. The data that we need was not stored on the hard discs of the computers.

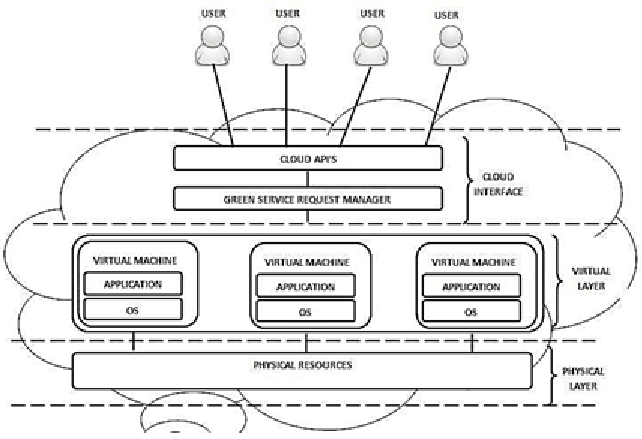

Figure 3.2 Cloud Computing framework

but rather in the virtualized resources. These virtualized resources served as the storage medium for the data that was stored on the hard discs. Internet connectivity is required in order for us to access the data contained inside the virtual resources. Users can utilise the software that is provided by cloud services even if the programme has not been installed on their local devices because cloud services deliver the software that is provided by third party service providers. Online file storage, web mail, social networking sites, and business applications are some examples of the services provided by the cloud. The data stored in the cloud may be accessed by cloud computing models from any location in the world. Figure 3.2 illustrates the architecture of the cloud computing system.

The following is the definition of cloud computing that was created by the National Institute of Standards and Technology (NIST) in the United States of America: Cloud computing provides users with on-demand network access to shared computing resources such as storage, servers, networks, and applications. Cloud computing is also known as utility computing. The computing resources are controlled with a very little amount of work and with a great deal of convenience. The models of cloud computing may be broken down into three different service models, four different deployment models, and five significant qualities. Cloud computing is distinguished by a number of key features, the most important of which are resource pooling, broad network access, quick flexibility, measurable service, and on-demand self-service. In comparison to the private networks, the broad band networks will supply the service. Customers can self-manage their own computer resources using the on-demand version of the self-service model. Customers have access to their own individual resources inside the shared pool of resources. The measured Models, Performance Metric service provides the ability for the user to pay the bill in accordance with their own individual consumption. There are three different service models available with cloud computing. The software as a service paradigm, often known as SaaS, explains how end users may access and make use of software without having to first install it on their local computers. Infrastructure as a service, often known as IaaS, is a model that makes available both hardware and computational resources. The IaaS resources will be accessible to the developers, but they will have limited power to do so. The client is responsible for the creation of or installation of the customer's operating system and application software. PaaS stands for "platforms as a service," and it refers to the service of providing a platform that may be used for developing, testing, and generating applications. The terms "private cloud," "community cloud," "public cloud," and "hybrid cloud" are all terms that refer to different types of clouds. A cloud service provider owns and operates a public cloud, which makes the cloud's services accessible over the internet.

The cloud service provider also manages the cloud. Public cloud services include things like online picture storage, e-mail, and social networking sites. The public cloud also offers some of the corporate application services that are already available. The services offered by a private cloudier, in which the underlying cloud infrastructure is managed or controlled exclusively for a single company, are delivered to that company only. The organisation itself or a third-party service provider is responsible for managing the services. The community cloud is a type of cloud computing in which the resources and services it provides are pooled and shared across a number of different organisations, with only those organisations being allowed to make use of the community cloud.

The infrastructure can either be owned and operated by the organisations themselves or by third party service providers. The Tharybid clouds are a mix of any number of various ways of pooling resources (for instance, it may combine public and community clouds).Host in the cloud In cloud computing technology, a huge number of separate computer systems are combined into a single entity that functions as a single machine. The hosting solutions are only dependent on a single computer, however the security services are provided by a large number of servers. One of the benefits of using cloud technology is that it will combine resources like RAM and storage space, which will result in an improvement in the quality of websites. A virtual machine, often known as a VM, is typically a computer programme or an operating system that does not actually exist on its own but is instead produced in another environment. The host and the guest are the two components that make up a virtual computer. The host is the server that manages the memory, disc I/O, and network I/O for the virtual machine. It also processes the processing power that is given by the computing resources. The visitor is responsible for keeping their own instance of an operating system and application software distinct from each other. The virtual workloads that are housed in the host virtual machine are referred to as guests.

Structure of model

The computers in a real time system are connected to one another over a diverse set of networks in order to tackle a single issue. Therefore, coordinating the operations of the computers is a difficult undertaking, which is made much more difficult by the imposition of deadlines. Real Time System is defined as the first item. A control-ling system, a controlled system, and the environment are the three components that make up a real-time system. The controlling system gathers information about the surrounding environment from the various input devices, then applies a predetermined computation to the collected data before taking control of the various output devices. This model is developed from the conventional real-time periodic job model that was presented in Liu (2000). This model properly characterises many typical hard real-time applications, such as digital control, real-time monitoring, and constant bit-rate voice/video transmission, amongst others.tn, where each tki is released periodically with a period pk, has a deadline dk, and arrival time has an ak. The model is made up of a collection of n periodic tasks, denoted by the notation T = t1, t2, and t3...... tn, where each tki is a periodic job. The period pk of tk represents the shortest possible duration of any and all-time intervals that exist between the release timings of successive tasks of in tk. The amount of work involved in each assignment remains the same throughout. The tasks are carried out on a collection of different types of computer systems, which are referred to collectively as Hosts H = h1, h2, h3, etc., hm. Each host is using the virtual machines with the settings Vi=vi1, vi2, vi3,..., vi| Vi|. Each virtual machine, vij, has the same amount of processing power as C(vij). Since execution times are portrayed as constants in the conventional paradigm, this model does not reflect the dynamic and unpredictable environment. It is insufficient for some real-time systems, which may have actuation tasks, action planning, and one or more of the tasks may have algorithms and execution durations that are impacted by unpredictability in the surrounding environment, such as system load or resource failure, for example. An accurate modelling of the system in terms of its performance requirements is primarily predicated on the robust scheduling of jobs in order to be resilient against a variety of unforeseen circumstances. Taking into consideration each task tkT, it is represented as a tuple with the elements ak, pk, lk, and dk, where ti is issued regularly with a period of pk and has a deadline dandy has size island has arrival time as ak. The model for the perturbation environmental parameters that have an effect on a system looks like this: pp = [pp1, pp2,..., ppn]. For every task Ti, there is an actual execution time ei that is a function of the environmental variable pp, or more simply, ei = f(pp). Utilisation is, in most cases, the foundational parameter that is utilised for performance metrics, and its formula is

$$P(pp) = \sum_{r=1}^{n} \frac{e_i}{P_i}$$

The utilisation of the system, denoted by U, is calculated as a function of the environmental factors. The value of U(0) reflects the fraction of the system's utilisation that is unaffected by the variables in the environment. If we take the same T set of tasks to be aperiodic, then each job tkT may be represented as a tuple with the elements ak, lk, and dk, where tiis having deadline dandy has size as liand has arrival time as ak. Only these three criteria are being considered here.

Workload Model

Applications that run in real time are often made up of a number of separate activities, each of which has a varying amount of importance. A real time application is often made up of a collection of interdependent activities that are triggered at predetermined intervals or on the occurrence of certain events. Typically, a task will detect the current state of the system, carry out a set of computations, and, if required, transmit a command to modify the current state of the system. The definition of the real-time task is of the utmost importance. The precise meaning might vary substantially depending on the area of computer science being discussed. In certain contexts, terms like application, task, subtask, task force, and agent are used to refer to the same thing, yet in other contexts, these terms have entirely distinct meanings. In order to maintain coherence throughout our investigation, we have decided to focus on the 14th Chapter. Models, performance metrics, and other aspects of real-time systems and cloud computing The following is a simple description of a real-time task, as well as a timing diagram for the task, which can be seen in Figure No.

Description of Real Time Task (RTT)

A unit of computation that must adhere to certain timing limitations when used in a distributed real-time system. Each job has a set of timing parameters that are associated with it, such as the arrival time, absolute deadline, relative deadline, duration over which the task is triggered again, approximate execution time, and priority, which indicates how important the task is.

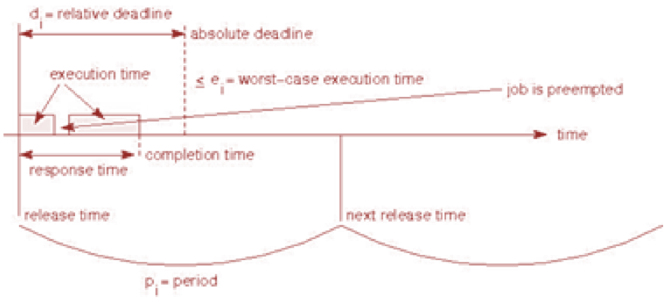

Figure 4.3 Schematic Representation of a Periodic Activity

Models of the real-time activities can be constructed using the parameters presented in Table 3.1. Real-time activities may be broken down into three primary types, depending on how frequently they occur: periodic tasks, sporadic tasks, and aperiodic tasks. Aperiodic tasks do not occur on a regular basis at all.

- A periodic task is a task that is activated (released) periodically at a set rate (period pi). A cyclic task is a task that repeats itself over and over again. In a typical setting, periodic tasks are required to adhere to certain limitations, one of which is that each instance of the task can only run once per period p. The representation of a periodic job is a tuple with the elements i, pi, ci, and di, where i represents the occurrence of the instance of Ti, pi represents the period of the task, ci represents the maximum amount of time that may be spent executing Ti, and di represents the relative deadline for Ti. Additional settings are available for periodic activities, as seen in Table 4.2.

- A sporadic task is a real-time job that is engaged sporadically with some known limited rate.
- Sporadic tasks can be performed in parallel. The bounded rate is distinguished by a minim minter-arrival duration, which refers to the shortest possible gap in time that exists between any two consecutive activations. A tuple, consisting of ei, gi, and di, is used to represent the sporadic job. Models, Performance Metrics where ei is the worst-case execution time, g is the minimal interval between two consecutive instances of Ti, and di is the relative deadline. ei is the worst-case execution time.

Aperiodic Activity: A real time task that can appear at random instants, similar to how sporadic tasks can. In this case, the minimal distance that must exist between any two consecutive occurrences is zero.

The application programme can be thought of as a collection of tasks, denoted by the notation Ai = (TAi,1TAi,2..........TAi,k).Each job is broken down into a series of smaller tasks, denoted by the notation TAi,1=st1i,1, st2i,1, etc. The right states of a particular job are thought to be the sub-tasks that make up that task. The distributed application is organised in such a way that each task is assigned to a certain node. In light of this, the resource requirements of a job are as follows:

$$T_i = P_{i}, \ldots\ldots P_{i+k} \; k \in (0,1,2,\ldots.n)$$

$$T_i = ch_{i}, \ldots\ldots ch_{i+k} \; k \in (0,1,2,\ldots.m)$$

RTS Task Scheduling:

Parameters and Notations

Task Scheduling in RTST he real-time system's primary objective is to ensure that the services are carried out before the specified cut off time. It is possible to split the process of scheduling services in real-time distributed systems into two distinct sections. The first question that must be answered, if a service intends to run on the system, is "how to run." After the job has been sent to the machine and once tasks have been allotted, the problem becomes one in which each node is responsible for creating a workable local schedule. The local scheduling approach and the scheduling method used in systems with a single processor are the same thing. The sequence in which services are going to be executed by operating systems is referred to as the "Real time task scheduling," which is an abbreviation. the key issue is that the services should complete within their deadlines. Real-time applications, which include things like aeroplanes and airports, military systems, factories, and other forms of industrialised technology infrastructure, have requirements that are quite stringent in terms of their performances. The degree of stimulability that real time systems possess is rather high. Therefore, the challenge of scheduling real-time services is an NP-Complete one. In order to obtain the highest possible throughput and resource utilisation of the system depending on how we planned the services on the cloud system, we need to accomplish this goal. The real-time services require their own time for calculation and transmission, in addition to their data resources.

Table 3.1 Parameters of RTWM

Parameter	**Notatio n**	**Description**
Arrival time	ak	Time at which task arrives
Absolute deadline	Dk	rk + dk
Relative deadline	dk	The maximum allowable job response time
Execution time	Ck	Time taken to complete the task
Maximum execution time	Ck+	Maximum time taken to complete the task
Minimum execution time	Ck-	Minimum time taken to complete the task
Completion time	CT_k	The instant at which

		a job completes execution
Laxity	L_i	Li = Dk - ak - Ck
Slack	SL_k	Lk = SLk = ∑ni=1Ck
Response time	RT_k	The length of time from the release time of the job to the time instant when it completes
Task arrival rate	λ	The rate at which tasks arrive
Total number of nodes	M	-
Release time of the task	rk	Time at which the task is ready for processing
Worst case execution time	ckworst	Worst-case execution time of the task at each release
Size of task	l_k	Size of the Task in number of instructions

Table 3.2 Parameters of RTWM

Parameter	**Notation**	**Description**
Period of task	pk	At all times the period and execution time of periodic tasks is known.
Phase of the task	Φk	Φk = rk,1(Jk,1), the release time of the first Jk in Tk
Utilization	uk	uk = ek / pk
Total utilization	U	U = ∑ni=1uk
Hyper period	H	The least common multiple of pk for k = 1, 2, ..., n

If the job is not allocated to any virtual machine (VM), then the scheduler checks once more to see if there is any VM that is free following the finish of its execution. If there is, then the task is assigned to that VM. This procedure will be repeated till the specified completion date of the task.

```
Algorithm 1 Basic EDF
Input: T, H, $V_i$:Set of VMs on Host $H_i$
Output: ETS:Successfully Executed Task Matrix
 1: MAX=max{$d_k, \forall t_k \epsilon T$}
 2: for $i \leftarrow 1$ to MAX do
 3:    RQ is empty set //Ready Queue
 4:    for j←1 to No.of Tasks do
 5:       if ($a_j \leq i$ AND $t_j$ not assign) then
 6:          $RQ \leftarrow RQ \cup t_j$
 7:       end if
 8:    end for
 9:    Sort RQ as Ascending order of Dead line of Tasks
10:    for $m \leftarrow 1$ to length(RQ) do
11:       for $k \leftarrow 1$ to No.of Hosts do
12:          for $l \leftarrow 1$ to No.of VMs in Host(k) do
13:             if $VM_{k,l}$ is free then
14:                Assign Task $RQ_m$ to $VM_{k,l}$
15:             end if
16:          end for
17:       end for
18:    end for
19:    for $k \leftarrow 1$ to No.of Hosts do
20:       for $l \leftarrow 1$ to No.of VMs in Host(k) do
21:          if Task on $VM_{k,l}$ complete execution then
22:             $ETS_{k,l} \leftarrow$ (Executed Task on $VM_{k,l}$)
23:             Free $VM_{k,l}$
24:          end if
25:          if Task on $VM_{k,l}$ reach dead line then
26:             Free $VM_{k,l}$
27:          end if
28:       end for
29:    end for
30: end for
```

IV.RESULT ANALYSIS

Simulation Result discussion with periodic based

In Simulation Results Graph is drawn between sets {GT, UV, TP},{Number of Tasks, Number of VMs}, so totally six graphs are shown. After that explanation of that graphs is done.

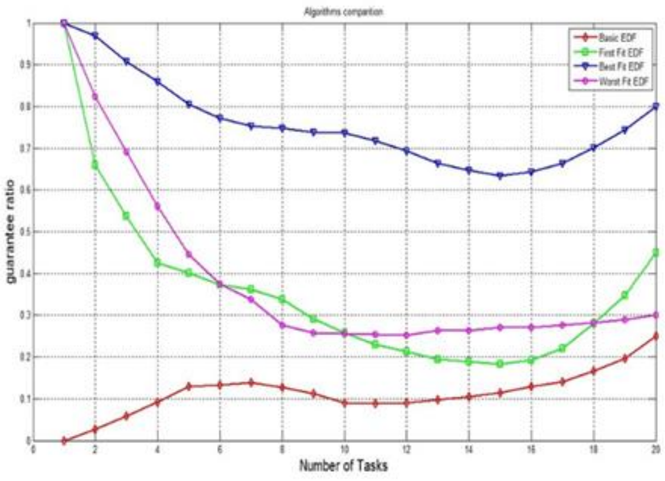

Figure 4.1: Guarantee Ratio Vs Number of Tasks

The behaviour of the four algorithms BEDF, FFE, BFE, and WFE is shown by draw graph. The fig 4.1 is plotted between Guarantee Ration and Number of Tasks, the fig 4.2is plotted between Utilisation of VMs and Number of Tasks, and the fig4.3is plotted between Through Put and Number of Tasks. The number of tasks can be changed while the number of hosts and virtual machines remain the same. It has been discovered that the three newly suggested algorithms perform significantly better than the Basic EDF. When the number of jobs that need to be completed grows, all three of the suggested algorithms see their performance improve.

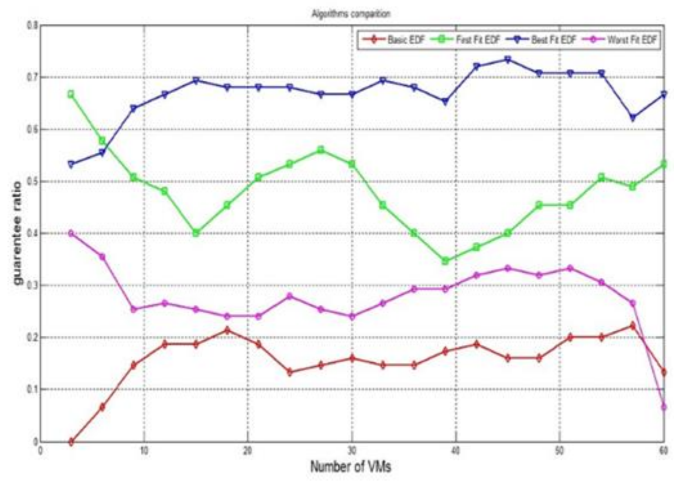

Figure 4.2: Guarantee Ratio Vs Number of VMs

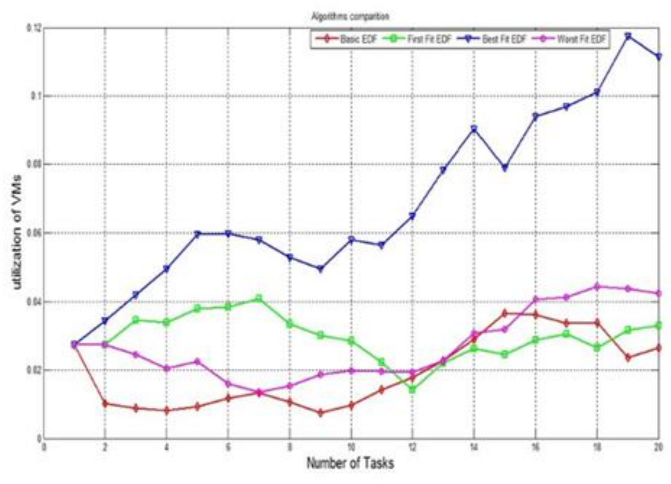

Figure 4.3: Utilization of VMs Vs Number of Tasks

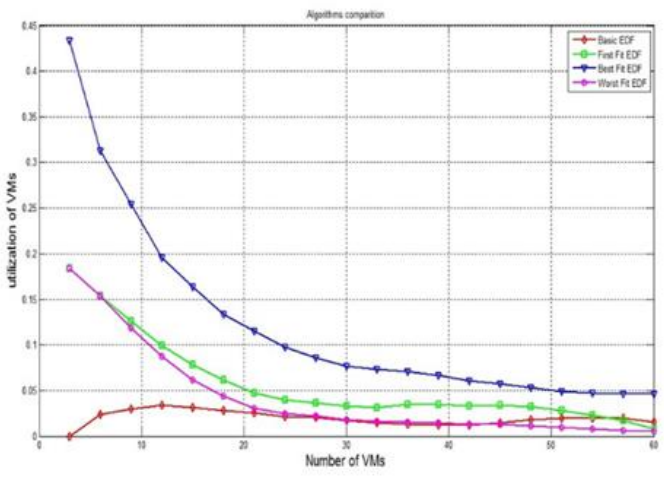

Figure 4.4: Utilization of VMs Vs Number of VMs

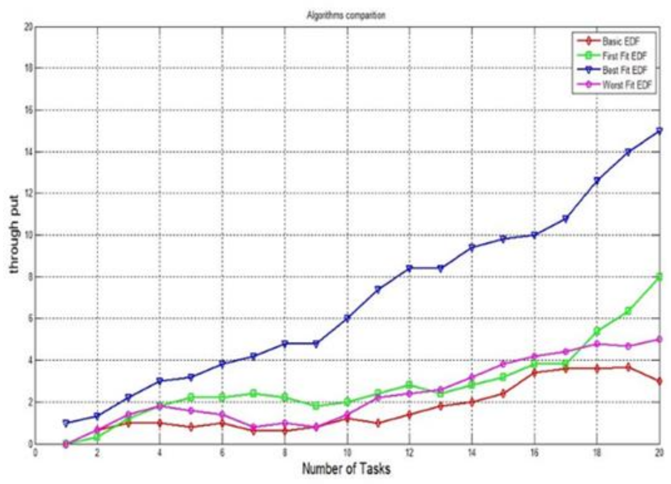

Figure 4.5: Through Put Vs Number of Tasks

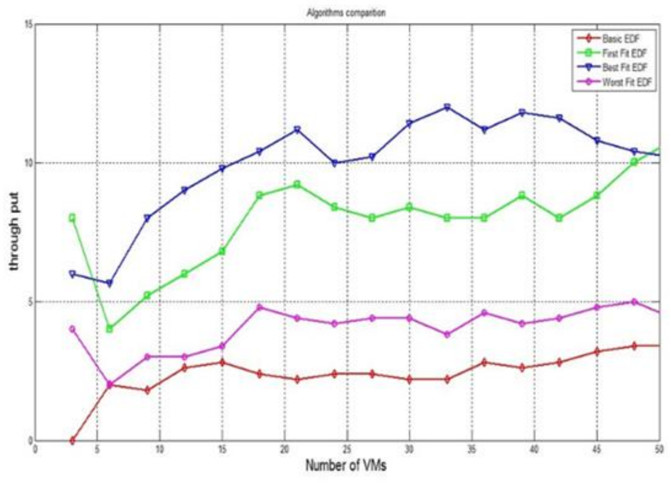

Figure 4.6: Through Put Vs Number VMs

Basic EDF.

Figure 4.4 depicts the behaviour of the BEDF, FFE, BFE, and WFE algorithms by drawing a graph between the Guarantee Ratio and the Number of VMs. Figure 4.5 depicts a graph between the Utilisation of

VMs and the Number of VMs, and Figure 4.6 depicts a graph between the Through Put and the Number of VMs. In this configuration, the number of Hosts and Tasks remains constant while the number of VMs increases dynamically. It has been noted that each of the three suggested algorithms performs much better than the Basic EDF. The performance of all three suggested algorithms improves with increasing numbers of virtual machines, starting with Basic EDF

V. CONCLUSION

the discussed contents include a brief introduction to Real-Time Systems, real-time task scheduling, and cloud computing. A case study, related work, motivation, and problem statement were presented. The thesis examined the real-time system model, workload model, scheduling model, and scheduling steps, along with other issues related to real-time task scheduling. Additionally, the performance parameters in scheduling real-time tasks were discussed. Regarding the scheduling of aperiodic tasks, four scheduling algorithms, namely BEDF, FFE, BFE, and WFE, were analyzed. Through simulation results, it was demonstrated that the three proposed algorithms consistently outperformed the Basic EDF algorithm when considering various performance parameters. Similarly, for the scheduling of periodic tasks, the same four scheduling algorithms were evaluated. The simulation results revealed that the three proposed algorithms consistently exhibited superior performance compared to the Basic EDF algorithm across different performance considerations

The scheduling of aperiodic tasks and periodic tasks are both carried out in the thesis. Work scheduling for sporadic tasks is going to be done at some point in the future. A Real-time job is said to be sporadic when it is triggered in an unpredictable manner at a certain limit. The bounded rate is distinguished by a minimal inter arrival period, also known as a minimum time gap between two consecutive activations. This is the defining characteristic of the bounded rate. The sporadic task is represented by a tuple with the values ei, qi, and di, where ei is the worst case execution time, qi is the minimum spacing between two consecutive instances of ti, and di is the relative deadline. In chapter 5, a concise explanation of real-time task scheduling and MapReduce is provided for the reader. More conversation in the future regarding the MapReduce system, and an attempt to implement a new strategy on the reduction step, with the goal of making the new strategy more effective.

VI. REFERENCES

[1]. Abdelmoneem, R.M., Benslimane, A., Shaaban, E.: Mobility-aware task scheduling in cloud-fog IoT-based healthcare architectures. Comput. Netw. 179(9), 107–348 (2020). https://doi.org/10. 1016/j.comnet.2020.107348

[2]. Abdullahi, M., Ngadi, M.A.: Symbiotic Organism Search optimization based task scheduling in cloud computing environment. Fut. Gener. Comput. Syst. 56, 640–650 (2016). https://doi.org/10. 1016/j.future.2015.08.006

[3]. Abohamama, A.S., Alrahmawy, M.F., Elsoud, M.A.: Improving the dependability of cloud environment for hosting real time applications. Ain Shams Eng. J. 9(4), 3335–3346 (2018). https://doi.org/ 10.1016/j.asej.2017.11.006

[4]. Abohamama, A.S., Hamouda, E.: A hybrid energy: aware virtual machine placement algorithm for cloud environments. Expert Syst. Appl. (2020). https://doi.org/10.1016/j.eswa.2020.113306

[5]. Baniata, H., Anaqreh, A., Kertesz, A.: PF-BTS: a privacy-aware fog-enhanced blockchain-assisted task scheduling. Inf. Process. Manage. (2021). https://doi.org/10.1016/j.ipm.2020.102393

[6]. Binh, H.T.T., Anh, T.T., Son, D.B., Duc, P.A., Nguyen, B.M.: An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment. In: Proc of the Ninth

Int Symp on Inf and CommunTechnol (pp. 397–404), Danang City (2018). https://doi.org/10.1145/3287921

[7]. Bitam, S., Zeadally, S., Mellouk, A.: Fog computing job scheduling optimization based on bees swarm. Enterp. Inf. Syst. 12(4), 373–397 (2018). https://doi.org/10.1080/17517575.2017.1304579

[8]. Boveiri, H.R., Khayami, R., Elhoseny, M., Gunasekaran, M.: An efcient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications. J. Ambient Intell. Humaniz. Comput. 10(9), 3469–3479 (2019). https://doi.org/10.1007/s12652-018-1071-1

[9]. Chen, X., Xu, H., Huang, L.: Online scheduling strategy to minimize penalty of tardiness for realtime tasks in mobile edge computing systems. In: Int Conf. on Big Data and Comput (pp. 107–114), Guangzhou (2019). https://doi.org/10.1145/3335484.3335537

[10]. Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H.: Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet of Things J. 3(6), 1171–1181 (2016). https://doi.org/10.1109/JIOT.2016.2565516

[11]. Fellir, F., El Attar, A., Nafl, K., Chung, L.: A multi-Agent based model for task scheduling in cloud-fog computing platform. In: 2020 IEEE Int Conf. on Informatics, IoT, and Enabling Technol (ICIoT) (pp. 377–382), Doha, Qatar (2020). https://doi.org/10.1109/ICIoT48696.2020.9089625

[12]. Ghobaei-Arani, M., Souri, A., Safara, F., Norouzi, M.: An efcient task scheduling approach using moth-fame optimization algorithm for cyber-physical system applications in fog computing. Trans. Emerg. Telecommun. Technol. (2020). https://doi.org/10.1002/ett.3770

[13]. Gill, S.S., Buyya, R., Chana, I., Singh, M., Abraham, A.: BULLET: particle swarm optimization based scheduling technique for provisioned cloud resources. J. Netw. Syst. Manag. 26(2), 361–400 (2018). https://doi.org/10.1007/s10922-017-9419-y

[14]. Hoseiny, F., Azizi, S., Shojafar, M., Tafazolli, R.: Joint QoS-aware and cost-efcient task scheduling for fog-cloud resources in a volunteer computing system. ACM Trans. Internet Technol. 21(4), 1–24 (2021). https://doi.org/10.1145/3418501

[15]. Kamal, M.B., Javaid, N., Naqvi, S.A.A., Butt, H., Saif, T., Kamal, M.D.: Heuristic min-conficts optimizing technique for load balancing on fog computing. In: Int Conf. on IntellNetw and Collab Syst (pp. 207–219), Bratislava, Slovakia (2018). https://doi.org/10.1007/978-3-319-98557-2_19