Multi Optimized Job Scheduling Framework for VM with Enhanced Migration in a Multi Cloud Environment

Md Tauqir Azam Kausar
Research Scholar
Computer Science Engineering
Bir Tikendrajit University, Manipur, India

Sanjay Pachauri, PhD
Dean, Research & Development,
IIMT College of Engineering,
Greater Noida, UP

ABSTRACT
Optimization job scheduling of virtual machines in a cloud computing for tasks is considered as NP-hard problem specifically for large task sizes in the cloud. Hence many techniques for job scheduling have been presented previously but they did not consider the combined task scheduling and resource allocation, which reduces the flexibility, increase traffic, congestion, and reduces computation processing time. Hence a novel technique, namely **Multi Optimized Job scheduling Framework for VM with enhanced migration in a Multi Cloud Environment** has been proposed, in which the load balancers with multi-level optimizations that utilizes the runner root algorithm and Differential evolution algorithm with Levy distribution to schedule the job and determines the VM to be allotted for the job based on international and national level optimization. Moreover, the previous techniques concentrate only on the migration that extends VM lifespan, lacking Quality of Service (QoS) and unsatisfied the end users. Hence a novel technique Active Inactive data migration algorithm is used to prevent fluctuating migration between Virtual Machines and recursive algorithm keeps on iterating the same operation on the server with the lowest virtual load and Optimum Cost Function is to prevent unnecessary migration cost. During VM migration, several applications were affected during a live VM migration that caused a network fault, which is eliminated by a novel **Data replacing approach** which is used to transfer the exact size of data to the active PM. Overall, the proposed method is to perform an efficient job scheduling in multi cloud environment with optimized VM migration.

Keywords
VMs migration, Load balancing**,** Live Migration, Federation, Runner root algorithm, Conjugate function, Steepest Descent Method, Recursive algorithm, Permutated sorting function, Optimum Cost Function, Levy distribution.

General Terms
VM: Virtual Machine

1. INTRODUCTION
Cloud computing is a trending technology that allows users to use computing resources remotely in a pay-per-use model. In this era of rapid growing technology, new opportunities are open for businesses, where recent technologies are replacing old ones. With the advent of cloud, small and big organizations all are progressing without need to concern about the storage and maintenance of their business data [1]. All the responsibility is envisaged upon the cloud service providers (CSPs) and hence cloud computing has become the backbone of modern business world. Organizations contacts various cloud service providers and consumes the services by signing Service-Layer Agreement (SLA) document. A CSP contacts various resource providers at datacenters to satisfy the demands of the customer. Usually, it is said that cloud computing provides infinite resources and elastic services [2]. To raise the flexibility or capacity of cloud service providers and fulfill the ever-growing demand of services, resources from different resource providers need to collaborate, inter-communicate and work in cooperation and coordination. So, collaboration of various cloud service providers gives root to the concept of multi-cloud which simply means that an enterprise can take services from more than one cloud service provider through a common interface or a single API [3].

The principle of multi-cloud paradigm in which each member cloud performs a service level agreement (SLA) with other member clouds that allows them to work together when data becomes too massive for any single cloud to manage [4].In multi cloud computing individual Consumer Service Provider(CSPs) are employed for a particular business or organization's purposes and they all have varying forms of application and SLAs. Moreover, the other benefits of multi cloud computing are that it avoids long-term commitment to a single cloud service provider, addressing concerns like interoperability and vendor lock-in [5]. These platforms develop new means of operability, either via increasing standardization of systems employed by creating new ways for clouds to communicate data with one another on a more global level, because they frequently rely on communication across their diverse cloud components. Furthermore, the users are not required to make any investments in new infrastructure. They can get the services they want from anywhere in the globe for a fee, and they do not have to worry about the intricacy of the IT infrastructure [6].

In multi cloud a Directed Acyclic Graph (DAG) represents an application as a collection of many jobs. Independent tasks in a DAG can be run concurrently by many virtual machines (VMs), however linked tasks must be run in the right sequence as determined by task priority [7]. Scheduling tasks for execution with the shortest makes span (total execution time of all tasks) is an NP-complete issue. Also, the multi cloud business models and technologies create serious problems, such as proprietary APIs and a lack of interoperability [8]. It is crucial that business companies could feed data into bigger, more popular outlets. It is also vital to select an application architecture that matches and fully exploits the peculiarities of the underlying Cloud environments [9]. Also, resource contentions at the infrastructure layer because unexpected performance, requiring more labor for resource management, as well as automated VM and service migration. In recent days, the focus of Multi Cloud Computing has been turned towards answering how to schedule an application's work across numerous clouds which is a difficult problem in a federated heterogeneous multi-cloud system [10]. For diverse computing platforms such as cluster, grid, parallel, and distributed

processing, few noted job scheduling methods have been created. However, they fall short of meeting the cost-effectiveness, dependability, and scalability criteria of multi-cloud computing [11].

Applications come in a variety of sizes and each application is broken down into several tasks and these tasks are assigned to Virtual machines (VMs). Hence task scheduling is extremely important for the overall efficiency of the multi-cloud computing system. It determines the order in which virtual machines execute tasks [12]. As a result, load balancing and scheduling are not two different methodologies but are two different abstraction levels. The concept of resource allocation is more abstract than that of load balancer and scheduler. Resource allocation entails assigning available tasks to VMs in the most efficient way possible, reducing the make span time [13]. Multiple jobs are discovered to be assigned to a single VM, resulting in improved system performance following optimum resource allocation and effective task scheduling. Executing the prioritized job requests/tasks is critical for the system's behavior in many circumstances [14]. One of the most difficult challenges in distributed computing is scheduling the cloud-task pair as the customers' needs are always changing. As the needs of consumers and working environments evolve, many existing algorithms become obsolete [15].

Virtual machine migration between real computers in cloud data centers is an intriguing component of cloud computing that is employed to satisfy the dynamic response to user demands. A server administrator can migrate a running virtual machine or application across physical machines without having to disconnect the client or application [16]. Total migration time and downtime are two significant performance measures that VM service clients frequently consider since they are concerned about service deterioration and the length of time that the service is completely unavailable [17]. When migrating a virtual machine, the transfer must be done in a way that balances the criteria of minimizing both downtime and overall migration time. In multi cloud computing, the strategy of optimum virtual machine placement on real equipment in the cloud data center is critical. When the placement in cloud data centers operates optimally, the quantity of hardware resources used is regulated. As a result, energy usage and resource waste can be decreased [18]. The main contribution of this paper are as follows:

-Distributed multi-cloud scheduling approach addresses scheduling issues in multi-cloud environments to maximize user and provider advantages. Overall time, expense, cloud throughput, energy use, resource use, and load balancing are all factors in the model.

-A new metaheuristic algorithm known as the runner-root algorithm (RRA), which is a task scheduling method based on the general algorithm (GA), is to minimize job completion time and cost while maximizing resource utilization.

-In order to save energy, proposed a method for VM placement in cloud data centers that combines several different techniques, including ensemble prediction algorithm, learning automata theory, and correlation.

Hence the suggested solutions carry out the economical VM migration along with optimal work scheduling. The content of the paper is organized as follows: section 2 describes related works, section 3 provides a novel solution, the implementation results and their comparison are provided in section 4; finally, section 5 concludes the paper.

2. LITERATURE SURVEY

Jena et al [19], this study presents Genetic Algorithm-based Customer-Conscious Resource Allocation and Task Scheduling in multi-cloud computing to bridge the gap between rapidly changing customer requirements and available infrastructure for services. Genetic algorithm-based resource allocation and shortest task first scheduling are the two main phases of the algorithm. The goal is to map jobs to VMs in the multi-cloud federation with the shortest possibly make span time and highest possible customer satisfaction. Extensive simulations were run on synthetic data, and the results were compared to the existing scheduling technique. The simulation results show that the suggested method outperforms the current ones in terms of the metrics that matter. The research parameters are converged towards the make span time schedule of the computing which lowers the efficiency of resource utilization.

Rama Subbareddy et al [20], this study takes job allocation in a multi-cloudlet context to increase user satisfaction. Response time aware task scheduling in the multi-cloudlet environment (RTTSMCE) is presented in this study to address two issues. First, a cloudlet server is chosen based on response time, and then tasks are scheduled across cloudlets using load balancing methods to reduce the cloud server's response time. In comparison to existing load balancing algorithms, the suggested approach performs better in the stimulation. By transferring applications from the mobile device to the remote cloud, mobile cloud computing helps to lower the power consumption. However, because of the large physical distance between a mobile user and the remote cloud, latency concerns arise.

Cai et al [21], this research developed a multi cloud distributed scheduling model for scheduling issues in a multi-cloud environment to optimize the advantages of users and providers. Total time, cost, cloud throughput, energy consumption, resource usage, and load balancing were taken as six goals of the model. The multi-cloud distributed scheduling model was optimized using a many-objective intelligence algorithm based on the sine function (MaOEA-SIN). To increase the algorithm's performance, a sine function penalty selection approach and an angle strategy are used. In conclusion, the MaOEA-SIN algorithm outperforms other algorithms in terms of performance. The user's preference influences the choosing of superior schemes based on steep characteristics leading to higher time consumption.

Chen et al. [22] suggested an Online Workflow Scheduling technique based on Resource Allocation and Consolidation with Adaptive Resource Allocation (OWS-A2C). When executing a SW in OWS-A2C, the deadline reassignment was initially performed for SW tasks depending on the execution performance of instance resources, which improves resource usage from a local perspective. The execution instances then were assigned and aggregated based on the performance needs of numerous SWs, improving resource usage, and lowering the overall costs of running many SWs. Finally, using the earliest-deadline-first (EDF) discipline, the SW tasks were dynamically scheduled to execution instances and finished before their sub-deadlines. Extensive simulation test was conducted to illustrate the efficacy of the proposed OWS-A2C on SW scheduling in MCEs, which outperforms three baseline scheduling approaches in terms of resource usage and execution costs under deadline restrictions, yet the flexibility of the system was constrained.

Farid et al [23], hosted the scientific procedures in multi-cloud systems has led in the development of the multi-objective scheduling (MOS) approach combining fuzzy resource utilization (FR-MOS). The suggested algorithm's major goal is to reduce cost and make span while also taking into consideration reliability restrictions. The scientific workflow schedule considers the following factors: (1) the IaaS cloud platform to be chosen; (2) the kind of VM to be allocated to the tasks; and (3) the sequence in which data should be transmitted. The FR-MOS technique uses particle swarm optimization (PSO) and analyses task ordering and task execution location in its coding approach to overcome these challenges. The coding system considers both the location of task execution and the sequence in which data is sent. But using single optimization to entire process expands the execution time thereby leveraging the exact task allocation.

Thirumalaiselvan et al [24], presented for scheduling virtual jobs in a multi-cloud environment, the rate-based scheduling (RBS), high priority scheduling (HPS), and equal load balancing (ELB). In a multi cloud environment design, multiple scheduling methods are utilized depending on the number of jobs and virtual machines. The ELB scheduling technique is employed when the number of tasks equals the number of virtual machines. The high priority scheduling strategy is employed when the number of tasks exceeds the number of virtual machines. The RBS method is employed if the number of tasks is smaller than the number of virtual machines. The research increased the make span and average efficiency of multi cloud computing by employing the above three alternative scheduling techniques which extended the make span while lowering the delay and energy usage. But the flexible nature of resource handling was constrained to a greater extend.

XAVIER et al [25] handle the issue of job scheduling in numerous heterogeneous virtual machines, a meta-heuristic algorithm called chaotic social spider algorithm. By simulating the social spider's swarm intelligence using chaotic inertia weight based random selection, this work focuses on lowering overall make span with effective load balancing. Here the two phase avoids local convergence and investigates global intelligent searching to identify the most optimized virtual machine for the user job from a set of virtual machines with minimal make span and balanced resource utilization. Later, additional performance metrics like security and dependability could be included, allowing for the identification of trust nodes and security risks. Additionally, we expanded this work to be compatible with independent jobs.

Hamad et al [26] The proposed method aims to reduce task completion times and costs while maximizing resource usage. Using the CloudSim toolbox, the suggested algorithm's performance has been assessed. The key issue is resource management, as cloud computing uses virtualization and the pay-as-you-go model to give IT resources (such as CPU, Memory, Network, and Storage) to users. To solve the job scheduling problem in the context of cloud computing, this research suggests an enhanced genetic algorithm. The suggested method aims to maximize resource use while minimizing completion time and cost. It can be expanded to consider the potential for VMs to have a dynamic quality. Also, the QoS needs of the users would be considered.

Zhang et al [27] The proposed method investigates global intelligent searching to find the best optimized virtual machine for the user task among a set of virtual machines with minimal make span and balanced resource utilization, thereby preventing local convergence. The flexible, and effective in many real-world circumstances through meticulous simulations involving many affecting aspects, algorithm for resource scheduling that reduces system costs. To address the resource needs of users on MCP, the system models of traditional CWAs are utilized. The study concludes that multi-cloud is the most alluring for many CWA implementations and can be used to understand the properties of various resources. Several CSP interconnections and associated load paths data travelling through potential interconnections are introduced. In the future, it will address these issues and take our framework's appropriate computing cost into account.

Tsakalozos et al [28] the suggested GA algorithm is to reduce job completion times and costs while maximizing resource utilization. The developer suggests a scalable, distributed network of brokers that monitors the status of all ongoing migration activities within the context of a provider. Brokers employ an underlying, specialized file system called MigrateFS, which can replicate and maintain synchronization of virtual discs as the hypervisor live-migrates VMs (i.e., RAM and CPU state). Brokers apply policies to reduce SLA breaches while attempting to accomplish all migration operations on time by restricting the resources used during migration.

From the analysis, it is noted that [19] lowers the efficiency of resource utilization, [20] large physical distance, [21] higher time consumption, [22] deadline restrictions yet the flexibility of the system [23] leveraging the exact task allocation [24] extended the make span [25] does not include the performance parameters [26] need to consider the dynamic quality of VM and also the QoS [27] Several CSP interconnections and associated with load paths [28] restricting the resources used during VM migration.

3. MULTI OPTIMIZED JOB SCHEDULING FRAMEWORK FOR VM WITH ENHANCED MIGRATION IN A MULTI CLOUD ENVIRONMENT

The intrinsic benefits associated with cloud computing, both the number of users and their corresponding workloads grow every day, which is essential to improve task scheduling and migration to increase Quality of Service (QoS), end user satisfaction, and with the least amount of energy consumption even under circumstances of high workload. Many earlier studies did not consider the combined job scheduling and migration for optimized work schedule, which decreases the resource management's flexibility and speeds up the execution of computations, traffic, and congestion. Hence, a novel multi-level optimization named, **Multi Optimized Job scheduling Framework for VM with enhanced migration in a Multi Cloud Environment** has been proposed, to consider the combined job scheduling and resource allocation, which utilized the two load balancers for multi-level optimization in multi cloud. When scheduling a task across multiple clouds, one load balancer uses the **Runner Root Algorithm (RRA)** to considering internationally and using the **Steepest Descent Technique**, another load balancer in a chosen cloud locates the VM to be assigned for the job based on a nation. The **Differential Evolution Algorithm with Levy Distribution**, which takes state level optimization into account, is part of the suggested system. Such that the multi-level optimization should consider both the VM's resource allocation and combined task scheduling. Moreover, irregular VM migration in the existing methods increased the duration of the user's VM,

decreased Quality of Service (QoS), and decreased end-user satisfaction. Hence a novel, **Active Inactive data migration algorithm** is used to prevent fluctuating migration between Virtual Machines, in which utilizes the **Recursive Algorithm** to migrate the virtual machine (VM) from the server with the lowest virtual load and repeating the same procedure on the server with the second-lowest virtual load. **Conjugate function** is used to calculate each VM's processing time in relation to the physical machine. An Optimum Cost Function is used to consider the future resource utilization in the each host and avoid unnecessary migration costs using the Automata cellular learning function that calculate the ratio of the cost of running the server in active mode to the cost of running the virtual machines on the replacement host, and if it exceeds the threshold, the VM is moved to the destination, reducing the unnecessary energy consumption while maintaining the quality of service (Qos).

Furthermore, in live VM migrations, the service levels of running applications are severely impacted through a high migration rate causes a network fault and misleading information to be transmitted improperly. Hence a novel, **Data replacing approach** have been proposed, in which Permutated sorting function has been used. If any error occurs while transferring the file, the algorithm copies the data and resends it until the active PM receives the precise amount of data, thereby the unwanted network error is avoided.

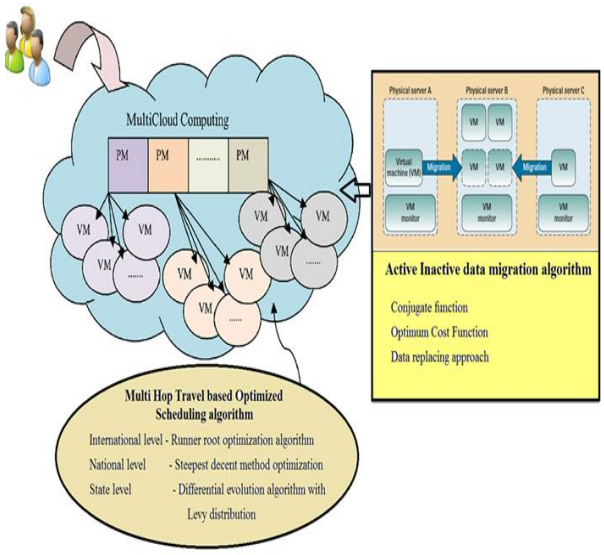

Fig1: Block diagram for Multi Optimize Job Scheduling

Figure 1 shows the proposed system's process flow. In order to allocate resources efficiently and create an optimized work plan, the suggested system would consider combined task scheduling and VMs migration where in two load balancers with multi-level optimization are used to schedule the job and assign VMs for it while taking into account global, national, and state level of optimization. Other unique approach eliminates the limitation during live migration and prevents variable migration across virtual machines, each host avoiding excessive migration costs. Hence, the proposed methods combine efficient VM migration with improved job scheduling.

3.1 Multi Hop Travel based Optimized Scheduling algorithm

In multi-cloud computing, resource allocation is a challenging task because of the numerous restrictions and configurations required by both cloud clients and providers. Because the nature of the traffic is highly arbitrary, the challenge of mapping an incoming task request to available virtual machines (VMs) is not polynomial-complete. The challenge of work scheduling is NP-hard since VMs are diverse and there are several alternative translations. To consider the combined task scheduling and migration for an optimized work schedule, which is crucial for improving the flexibility of resource management and accelerating computation execution, traffic, and congestion. Hence, a novel **Multi Hop Travel based Optimal Scheduling technique** is employed, which divides the entire allocation into two phases and uses two load balancers with multi-level optimization. The purpose of load balancer is to more effectively match the network's available transmission resources to the volume of data that is currently being handled. One balancer in a multi-cloud to schedule the task to the proper cloud computing consideration globally, which is optimizing by Runner root algorithm (RRA). The job scheduling issue is regarded as an NP-Complete issue. Therefore, it could be resolved using optimization techniques while considering performance parameters like completion time, expense, resource utilization, etc. To create a task allocation and execution algorithm based on Runner root algorithms (RRA) for the cloud computing environment that will improve task completion times, lower execution costs, and optimize resource utilization. More specifically, in RRA, the local search (exploitation process) is only used when the global search does not significantly enhance the value of the cost function. In RRA, the global search for the optimal solution (exploration method) is undertaken at all iterations. The runner root algorithm is provided as a job scheduling optimization strategy, which is starts with an initial random population that is evenly distributed over the issue domain. Task scheduling to meet the objectives of better makespan, load balancing and throughput.

Task allocation details are indicated by a task t_k. K represents the number of tasks in a population and ranges from 1 to z. The components of a task t_k are $\alpha[i]$ and $\beta[i]$, which stand for the details of task processing and virtual machine distribution. A task's length is equal to the total amount of tasks entered. A task schedule is expressed through the following encoding process. Prior to task creation, a collection of inputted jobs is sorted. Cloud users pay for computing services in person, in contrast to other distributed computing platforms. Considering this, it is necessary to assign tasks from cloud users with high costs to virtual machines more quickly than other tasks. Due to the fact that cloud computing services are provided through an SLA between cloud users and providers, task scheduling issues in cloud computing vary from problems with general task scheduling.

Given that there are m tasks, such as t1, t2, t3, etc., and that there are m number of resources, and that task i (ti) has n subtasks, with the jth subtask of task i being designated as ti(j), there are a total of m tasks:

$$num = \sum_{i=1}^{m} \sum_{k=1}^{n} t_i(j) \quad (1)$$

Assuming there are three tasks and three labor resources, the first task is divided into five smaller tasks (t1(1), t1(2), t1(3), t1(4), and t1(5)); the second task is divided into five smaller

tasks (t2(1), t2(2)); and the third task is divided into three smaller tasks (t3(1), t3(2), and t3(5)).(3). There are 10 subtasks in total. The length of the work is 10 subtasks, each with a gene value between 1 and 3. The jobs are generated as follows:

{3,2,1,1,1,2,2,2,3,1}

The job is then decoded to reveal the distribution and order of processing of each resource's subtasks.

W1 : { t1(3) , t1(4) , t1(5), t3(3)}

W2 : { t1(2), t2(1), t2(2), t3(1)}

W3 : { t1(1) , t3(2)}

Through decoding, it can determine the subtasks that each worker must complete, and using the RRA algorithm, it can determine how long it will take each worker to complete the task that has been given to them:

$$Workertime(k) = \sum_{j=1}^{n} time\ (k,j), k \in [1,w] \quad (2)$$

time(k, j) represents a k-th worker on the time required to complete the j-th task.

Time is required by the i-th task completion:

$$tasktime(i) = max_{k\ =\ 1}^{w} \sum_{j=1}^{s} time\ (k,j) \quad (3)$$

s is the location of subtask of task i assign to the worker. One of the main problems with cloud computing is task scheduling. Quality of Service (QoS) factors are important in scheduling and load balancing. which is based on international optimization, Resource Allocation and Task Scheduling in Multi-Cloud Computing to close the gap between the continuously changing requirement and the available infrastructure for the services.

Due to the huge solution space, scheduling in cloud computing falls under the issues known as NP-hard problems, making it difficult to find an ideal solution. It has been demonstrated by these techniques based on metaheuristics can solve these issues with near optimal results in a reasonable amount of time. The two categories for steepest decent method -based resource allocation tasks are (I) Advance Reservation (AR) and (ii) Best Effort (BE). The work is distributed among resources utilizing GA operators based on the multi-cloud environment's available resources and the anticipated makespan time. The historical user feedback database keeps track of the performance of the cloud service providers, physical machines, and virtual machines.

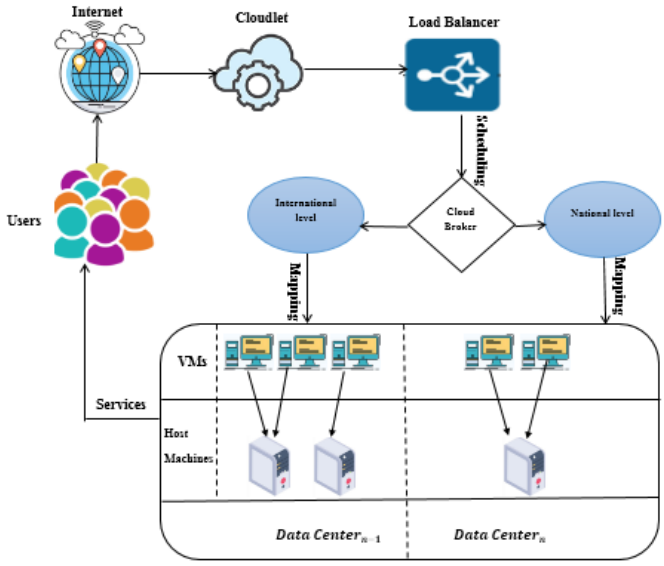

Figure 2: Task scheduling Algorithm

Figure 2 shows a cloud in an abstract form where the scheduler finds a good allocation for incoming tasks. The scheduler establishes a map when tasks are delivered to data centers. A cloud broker receives a mapping scheme, and then assigns jobs to virtual machines. Because the network bandwidth among edge clouds is more limited than the cloud data center networks, VM migration among edge clouds is more difficult than that in cloud computing.

For end users, the virtual machines offer a variety of services, including message transfer, mobile gaming, and video streaming. Any application running on the VM can be referred to as a service, which is an abstract notion. Consider a live VM migration from a cloud-based source computer to a cloud-based destination machine. It presumes that the destination machine will need to receive the state stored in the virtual machine's memory during the migration. The memory of the VM on the source machine would be updated when the state was transmitted to the destination because the application on the VM could still operate throughout the migration. Pre-copy, a live migration method iteratively transmits this memory content from the source computer to the destination machine. Two objective metrics of a live VM migration that we are concerned with are migration duration and Quality of Service (Qos). Imagine a group of C cloud service providers that are linked together to create a multicloud computing, where C = {C1, C2, Ci}. Q is a collection of cloud apps exist, where {P = P1, P2,Pj}. A cloud user may submit an unlimited number of job requests. Each job application is divided into a number of independent tasks, with Pi j = {P11,P12,..., Pq1,Pq2,..., Pqi} and Ci j = {C11,C12,..., Cp1,Cp2,..., Cpi} being the set of tasks and VMs, respectively.

Mapping function M describes: Pi j → Ci j

The service charge for AR work is typically higher than the service charge for BE tasks. Below is a matrix that was created to display the anticipated execution time in equation (4),

$$ETC = \begin{matrix} \\ T_1 \\ T_2 \\ \vdots \\ T_n \end{matrix} \begin{matrix} C_1 & C_2 & \dots & C_n \\ ETC_{11} & ETC_{12} & \dots & ETC_{1m} \\ ETC_{21} & ETC_{22} & \cdots & ETC_{2m} \\ \vdots & \vdots & \dots & \vdots \\ ETC_{n1} & ETC_{n2} & \cdots & ETC_{nm} \end{matrix} \quad (4)$$

ETC_{ij}indicates the anticipated time required to complete the ith task in the jth cloud. Any cloud that has a working job request id can run any task, and any cloud can do several tasks simultaneously according to priority. Chronological order is used by several cloud providers.

$$F(x) = \min(MS) + \left(\frac{1}{max}, CSR\right) \quad (5)$$

$$MS = f\left(MIPS_{temptask}, EST\right) \quad (6)$$

$$MS = \omega_1 * \left(\frac{NIC}{MIPS}\right) + \omega_2 * EST \quad (7)$$

$$CSR = f\left(EST_{temptask}, ETC_{temptask}\right) \quad (8)$$

Whereas Eq.(8) shows the relationship between user satisfaction levels, resource waiting times, and anticipated completion times, make span time (MS), a computability indicator, reveals the rate of utilization of resources expressed in Eq (6)&(7). Where MS is the task's makespan time, CSR is the customer satisfaction rate, NIC is the number of million instructions in the work, MIPS is the number of million instructions the machine can execute, and $\omega_1 \, and \, \omega_2$ are specified weights. Choosing the weights' value might be difficult because it differs from organisation to organisation. The following algorithm shows the runner root-based task scheduling.

Algorithm1: for runner root algorithm-based task scheduling in cloud computing

Input

Step 1: set of customer job requests following Poisson's distribution.

Step 2: set of independent tasks. (each job request is sub divided into single independent task)

Step 3: setoff cloud providers involved in the federation.

Step 4: set of virtual machines. (Multiple cloud providers are further divided into numerous VMs).

Output

(1) Makespan time
(2) Customer Satisfaction rate

Step 1: While $Q_r \neq NULL$

Step 2: Set makespan = 0

Step 3: Breakup job application into multiple tasks.

Step 4: Call GA_MAPPING (ETC, EST, p, q)

Step 5: Call Task Scheduling (ETC, EST, p, q, MS)

Step 6: end while

Temporary queues QT are initialized as part of algorithm 1. The Poisson distribution is used to generate a variety of applications with varying capacities (measured in MIPS, or million instructions per second). The programs divided into numerous separate tasks. In step 3, the relevant physical machines divided into several VMs. Step 4 involve calling the GA-based resource allocation function. Scheduling the numerous tasks assigned to a single VM is step 5 in the process. The algorithm produces the optimal task-VM pair with the shortest makespan time and the highest level of user satisfaction. The following algorithm shows the resource allocation for the scheduled task.

Algorithm 2: For resource allocation

START

1: While $Q_r \neq null$ do

2: If $Q_{AR} \neq null$ (if task ready available is advance reservation then)

3: If$Q_{BE} \neq 0$(if task ready available is Best Effort task then)`

4: For tempcloud = {1,2,3, …..., q)

5: For temptask = {1,2,3, …..., p)

6: temptask← Task (Q_{AR})

7: Find EST (temptask, tempcloud)

8: MS (temptask, tempcloud) =ETC (temptask, tempcloud) +EST (temptask, tempcloud)

9: Call RRA_task_cloud_pair(p_i, q_i) that gives min (MS (temptask, tempcloud))

10: Call BE_PREMPT_TASK (EST (temptask), MS (temptask, tempcloud))

11: endfor

12: endfor

13: else

14: temptask← Task (Q_{BE})

15: CALL UPDATE Q_T

16: CALL SCHEDULE_AR_TASKS_MMS (ETC_AR, temptask)

17: CALL SCHEDULE_BE_TASKS_MMS (ETC_BE, temptask)

18: MS (temptask, k) = ETC (temptask, k) + EST (temptask, k)

19: endif

20: endif

21: endwhile

The step-by-step explanation of our suggested algorithm 2, steepest decent method-based resource allocation is contained in Algorithm 2. The programs tasks are kept in QT. Tasks are stored in QAR or QBE depending on the type of application. All the tasks that need to be completed are saved in the set temptask, and the relevant VMs are kept in the set tempcloud. Step 7 determines the estimated execution time. The makespan time is the total of the predicted completion time and the waiting time, as shown in step 8. The steepest decent method - based resource allocation process is called in step 9. In the initialization stage of steepest decent method, the number of jobs that must be completed in a batch is equal to the size of the cloud. In the first generation, tasks are given at random to VMs that can complete them. Maximizing customer satisfaction rates while minimizing makespan time is the fitness function. Procedure 1 specifies the steps for steepest decent method - based resource allocation as follows. Step 17 indicates that the convergence requirements are satisfied, and the best-fit chromosome is acquired. When numerous jobs are assigned to a single VM, shortest job first scheduling is employed to handle the situation.

As increased user tasks are allocated within the schedule, the VMs risk being quickly overcrowded. In order to make better load balancing decisions to determine the load factor (LF)σ, which is the average load's standard deviation.

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(ET_i - ET)^2} \quad (9)$$

where ET_i , is the execution time of ith VM.

A steepest descent algorithm would be one that applies the update rule, with each iteration taking the steepest possible course in the direction x(k). Which two significant computational benefits are how simple it is to implement an algorithm on a computer and how little storage is required. The line search necessary to calculate the step length α_k, and gradient constitutes the bulk of the task. In other words, given a specific point x, the algorithm's goal is to determine the direction in which f (x + d) is minimized.determining the steepest angle. One can estimate the function by a first-order Taylor expansion and identify the steepest direction in the following equation (7),

$$f(x+d) \approx f(x) + \nabla f(x) T_d \quad (10)$$

The function's minimum direction d suggests the following optimization issue.

$$\min_{d:||\vartheta||} \nabla f(x)\, T_d \quad (11)$$

Algorithm 3: Steepest Descent Method

Given an initial $x_0, d_0 = -g_0$ and a convergence tolerance tol

for k = 0 to maxiter**do**

Set $\alpha_k = argmin\ \varphi_\alpha = f(x_k) - \alpha g_k$

$x_{k+1} = x_k - \alpha_k g_k$

Compute $g_{k+1} = \nabla f(x_{k+1})$

If$||g_{K+1}||_2 \leq tol$ ***then***

Converged

End if

End for

Thus, the other load balancer in that particular cloud chooses the VM to be assigned for the task based on a certain nation, then uses national level optimization with the steepest descent algorithm. Due to its effectiveness in handling a wide range of issues, such as portfolio optimization, picture pixel clustering, data clustering, and multi-level thresholding in image segmentation, Differential Evolution (DE) algorithms, a subset of evolutionary algorithms, are of particular interest. These mutational tactics are used in many evolutionary algorithms, such as DE algorithms, to address a variety of issues, including multi level objective optimization. Thus, the VM are viewed as different states, and the work that must be done is viewed as districts.

DE is an iterative population-based method for locating the state-level optimal. The investigation and application of the algorithm are represented, respectively, by the DE algorithm with levy flight. The levy flights first create a population of answers at random before assessing each one's quality using the fitness function. Using Levy flights, the jobs that are closest to the best one will fly around it as shown in the following equation,

$$x^{t-1} = x_i^t + \frac{S_{max}}{t^2} L(S) \quad (12)$$

where x_i^t represents the position of the i-th task at iteration t. while S_{max} represents the maximum walk step and L(s) represents the step drawn from Levy flights, using parameter s. Hence the Runner Root Algorithm (RRA) and the Steepest Descent Algorithm are combined in the DE algorithm, which is used as a global and local search technique to enhance job scheduling for resource exploitation. By minimizing the makespan, the DE algorithm, which was modelled in the cloudsim environment, aims to increase the output of the cloud system.

3.2 Active Inactive data migration algorithm

The VM migration that comes next, which does not consider prior task knowledge, extends the entire time the user is using a virtual machine (VM), possibly infringing on the deadline requirement with subpar Quality of Service (QoS) standards and unsatisfied end users. To avoid fluctuating migration between Virtual Machines, the suggested algorithm is used. To increase performance and reliability, one mitigating method is VM migration, in which virtual machines are transferred from one physical host to another. There are various methods for migrating VMs, including cold migration, hot migration, and live migration. When migrating a virtual machine to a specific host, cold migration requires shutting down the guest OS first and then restarting the system. Hot migration does not terminate the operating guest OS before it is sent to the designated target host and resumed there it just suspends it.

Although permitting a VM and its operating OS to be relocated from one physical host to another, live migration ensures that the hosted apps will continue to function. A virtual machine (VM) is effortlessly transported between two physical hosts while still running, together with its environment, which includes its OS, memory, vCPU, and occasionally its disc. Improved load balancing, transparent mobility, proactive fault tolerance, and green computing are all advantages of VM migration.

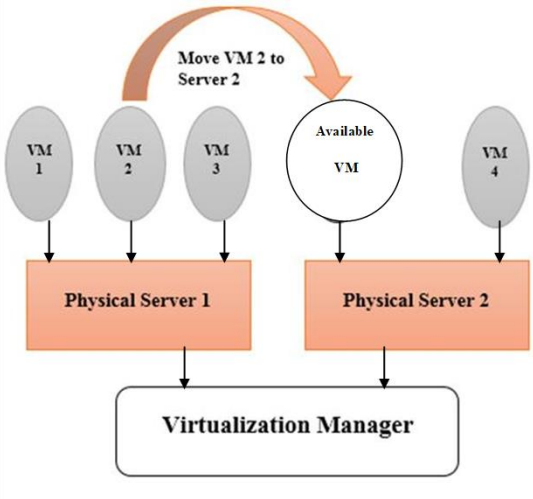

Figure 3: VM live migration between two physical VM

The figure3 demonstrates how live migration lets you relocate an active virtual machine from one physical server to another without interrupting operations. A seamless migration process is ensured since the virtual machine keeps its network identity and connections. High-speed networking is used to transfer the virtual machine's precise execution state and active memory, enabling it to move from executing on the source host to the destination host. Recursive algorithm with the intention of minimizing power interruption for the active machines, the algorithm used to move idle and actively functioning virtual machines from one overloaded or under loaded server to another non-overloaded server to reduce server load and offers more substantial energy and resource savings for data centres. To guarantee the greatest number of active virtual machines on a single server the majority of the time, our approach is to swap out all idle virtual machines from one server with the actively working, fully loaded ones of a no overloaded server. Since idle VMs typically use 50% to 70% of the host server's total power, this means that the power consumption of the actively operating VMs will not be affected. The following situations can coexist in a cloud environment, according to the CPU and RAM usage of a VM, for an instantaneous time t.

$$U_{mvj} > U_{pj}[RAM > CPU] \quad (13)$$

$$U_{mvj} > U_{pj}[RAM < CPU] \quad (14)$$

$$U_{mvj} > U_{pj}[RAM \approx CPU] \quad (15)$$

Here, U_{mvj} represents the memory utilization of a VM, U_{pj} represents the processor or CPU utilization of a VM and 0 ≤ U_{mvj}≤ 1,0 ≤ Ucj U_{pj}≤ 1 i.e U_{mvj} and U_{pj} represents the percentage of RAM & CPU utilization.

The resource utilization percentage of each virtual machine will be used to determine the overall number of active and idle VMs on a single server at a given instant (t). It will be simpler to choose between migrating idle or active VMs as a result. The following formulae can be used to get the total number of idle virtual machines in a server for an instant t:

$$V_o = U_{mvj} - U_{pj} \quad (16)$$

$$V_o = \begin{cases} 0, & 0 \le V_o \le 0.3 \\ 1, & 0.4 \le V_o \le (0.9 \approx 1) \end{cases} \quad (17)$$

$$V_o = \begin{cases} \sum_{o=1}^{N_1} V_o, [V_o \in N_1] \\ 0 \quad ,otherwise \end{cases} \quad (18)$$

Here, N1 is a set of virtual machines. Once more, the following formulae can be used to determine the total number of active virtual machines in a server for a given time.

$$V_a = U_{pj} - U_{mvj} \quad (19)$$

$$V_a = 1; \ if \ 0 \le V_a \le (0.9 \approx 1) \quad (20)$$

$$V_{active} = \begin{cases} \sum_{a=1}^{N_2+N_3} V_a + c, V_a \in [N_2 + N_3] \\ c \quad , \quad otherwise \end{cases} \quad (21)$$

$$c = \sum V_a \quad (22)$$

The following equations can be used to determine the overall CPU and memory usage of the VMs when the quantity of running virtual machines and idle virtual machines at any given moment t, in any server I equals the other.

$$UC_{av} = \sum_{j=1}^{V_{active}} U_{mvj} * C_j \quad (23)$$

$$UM_{av} = \sum_{j=1}^{V_{active}} U_{mvj} * M_j \quad (24)$$

Here UC_{av}UCva and UM_{av} stand for the CPU and Memory usage of virtual machines that are currently in use, respectively.

Algorithm 4: Recursive algorithm (Virtual Machine Migration)

Input

Step 1: Initialization. Compute the number of active and idle VMs in a single host server. Take the number of active VMs as Vactive and the number of idle VMs as Vidle. Compare Vactiveand Vidle.

(1) Vidle>Vactive
(2) Vidle<Vactive
(3) Vidle = Vactive

Step 2: The parameter for ACS is set to τ0. The feasible globally best solution is set as Sgb for placing N VMs on N servers. Thus, the number of minimum servers is set to Mmin = N. Set iteration t=1 and maximum iteration as T.

Step 3: Set Mt = Mmin − 1. In each iteration, m ants construct m solutions and perform local pheromone on each solution.

Step 4: The fitness function f(S) is applied in order to evaluate the fitness value of the constructed solution.

Step 5: The best solution Sb of the current iteration is set after evaluating the fitness value of the constructed solution. If Sb is feasible, Sgb is updated as Sb and Mmin = f1(Sb) is set. Otherwise, OEM local search is performed on Sb. Sgb and Mmin are updated respectively if local search succeeds.

Step 6: After Sgb and Mmin are locally updated, global pheromone update is eventually done on Sb and Sgb.

Step 7: Check if t is less than or equal to T. If not equal, then set t = t + 1 and go to Step 3. Otherwise terminate the algorithm.

Step 8: After t terminates, calculate Vactive and Vidle of the host server. If Vactive = Vactive and V`idle = Vidle, then move forward to step 9. Otherwise update V`active and V`idle and then move to step 9.

Step 9 (a) If Vidle>Vactive, then migrate all the actively working VMs from host server to a nearby (ACS) nonoverloaded (OEM) server with the opposite scenario i.e. the server in which Vidle<Vactive. The idle VMs of the destination server will be exchanged with the actively working ones from the host.

(b) If Vidle<Vactive, then migrate all the idle VMs from host server to a nearby (ACS) non-overloaded (OEM) server with the opposite scenario i.e. the server in which Vidle>Vactive. The actively working VMs of the destination server will be exchanged with the idle ones from the host. (c) If Vidle = Vactive, calculate the CPU utilization (UCva&UMva) and memory utilization (UCvi&UMvi) of the VMs (both actively working and idle) and compare the total utilization of both type VMs.

It migrates the virtual machine from the server by identifying the virtual machine with the lowest virtual load and then repeats the process on the server with the second-lowest virtual load, and so on, using a recursive algorithm. A recursive algorithm is one that calls a copy of itself, or an instance of itself, more precisely. When a set or function is defined recursively, the computation of its members or values follows the definition in a recursive manner. The initial steps of the recursive algorithm identify the basis items and correspond to the basis clause of the recursive definition. The inductive clause's stages are then followed, which reduce the computation for an element of one generation to that of elements of the generation just before it. The algorithm uses a conjugate function to determine the execution time of each VM in relation to the Physical Machine while also taking into account the timeout parameter of each server from the history of data centres.

The majority of earlier studies calculate the start time of the current task as the most recent end time of the previous task. As a result, when it is their time to receive tasks, some virtual machines must wait. The execution time of each task ti depends on the output data size of every task. The execution time of different tasks on different VM(m, k) can be calculated by the following equation

$$T_{exe}(t_i VM(m,k)) = \frac{W(t_i)}{P(m,k)} \quad (25)$$

The execution time of each task can be calculated using the processing capacity of VM(m,k). This is so that other VMs can receive numerous copies of the output that VMs produce. The sequence of the tasks determines how the recipient output is laid out. The efficiency of the process for scheduling applications must be maximized. Reducing execution time and total execution cost are necessary steps to take to meet users' QoS requirements. The time between the start time and end time of the task execution is used by the existing workflow algorithms to compute the VM rent time.

When a task is finished, the virtual machine closes down and the results are passed on to the tasks that come after it. Data transfer priority is influenced by the order of the activities. Using the automata cellular learning function, an Optimum Cost Function considers the future resource utilization in each host to reduce the cost of unnecessary migration. The suggested optimization model restricts that the VMs whose remaining runtimes are smaller than a time slot will not be migrated to prevent pointless VM migrations. VM rent cost of task ti for each considered IaaS platform is calculated below.

For Amazon EC2 that charges per hour, the execution cost of task ti on VM(1, k) is expressed in Eq.

$$cost(t_i, VM(1,k) = \left[\frac{T_{rent}(t_i, VM(1,k))}{T_{minute}}\right] . C(1,k) \quad (26)$$

where T_{minute} = 60.

Microsoft Azure charges per minute, the execution cost of task ti on VM(2, k) is expressed in Eq

$$cost(t_i, VM(2,k) = T_{rent}(t_i, VM(2,k) . C92,k))/T_{minute} \quad (27)$$

A distributed computational mode called cellular learning automata (CLA) model combines the learning capabilities of learning automata with the computational capability of cellular automata. A cellular learning automaton is made up of a lattice of cells that cooperate to complete a computing job, and each cell contains a few learning automata. The CLA is used by each host are together with cellular networks, wireless networks, and evolutionary computation. Cellular learning automata that consider the migration and base future decisions on the experiences of the past. It is determining the ratio of the cost of running the server in active mode to the cost of running the server for the virtual machines on the replacement host, and if it exceeds the threshold, moving the VM to the destination reduces the unnecessary energy usage while maintaining service quality. This capability enhances the flexibility and computing power of automatic learning through associative CLA. Initial state of the cost is set based on the action probability vector of the LA in running server. The LA resident in each VMs then decides on an action in accordance with its decision function after receiving an input vector from the cloud.

Algorithm 5: Operation of Cellular Learning Automata

Step 1: Initialize state of each cell in the CLA.

Step 2: for each cell i in the CLA do

Step 3: Give the input vector from the environment to cell i.

Step 4: Cell i selects an action according to its decision function.

Step 5: Apply the selected action to the environment.

Step 6: Apply the local rule and give the reinforcement signal to the cell i.

Step 7: Update the state of the cell i according to the reinforcement signal.

Step 8: end for

The VMs is a migrated to the cloud that gives feedback in conjunction with actual migration, and Learning Automata is taken as the cost of running server in the VM is migrated to the destination thereby lowering the unwanted energy consumption same while maintaining the quality of service. However, faulty data transfer during live migration caused a network issue.

3.3 Data replacing approach.

In data replacing approach, VMs must be moved to another host with enough resources once a host enters an over or underutilized state. The following circumstances lead to unnecessary requests: One of the request's fields was not accurately recorded (data contains NULL). The user is identified as a spammer, scanning robot, or intrusive user until the data is sent to the active PM, the transaction PM is switched to Mid active mode. Whenever a transfer error occurs, the method duplicates the data and resends it until the active PM receives the exact amount of data using a permutated sorting function, preventing the unintended network fault. Permutated Sorting Function continuously produces input permutations until it discovers one that is sorted. So, it should count the amount of original data and moves in order to examine a sorting algorithm. So, it should count the amount of original data and moves in order to examine a sorting algorithm. We can ignore other procedures and yet get the same result. We can ignore other procedures and yet get the same result. A flexible scheduling method that uses VM migration to effectively service physical servers of different functionality while workflows are being executed

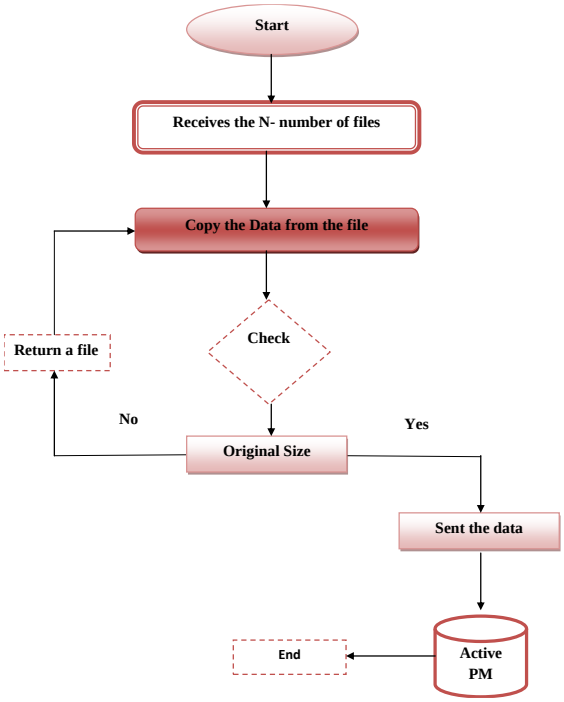

Figure 4: Flowchart of the Data replacing approach

Figure 4 shows the Data replacing method that chooses the best PM from the list of techniques while switching the PM for the data transaction to Mid-active mode until the data is transferred to the Active PM. If a transfer error occurs, the algorithm copies the data and resends it until the precise amount of data is sent to the active PM. Unwanted network errors must be eliminated. Overall, the proposed Multi Hop Travel based optimization algorithm is conduct the economical VM migration while optimizing job scheduling where two load balancers that are optimized at multiple levels, including the international, national, and state levels, are taken into consideration. Active Inactive data migration algorithm for active-inactive data transfer removes virtual machines from servers by selecting the ones with the lowest virtual loads using a recursive algorithm.

4. RESULT AND DISCUSSION

This section provides a comparison section to ensure the suggested system is appropriate, performance data for the suggested system, and adaptive scheduling approaches of the implementation of the VM migration. Using an optimization method, the suggested VM migration strategy was put into practice in MATLAB, and the experimental outcomes were analyzed. The performance of the proposed model has been assessed by calculating the increased flexibility, lowest economic cost, and low computational time.

4.1 Experimental Setup

This work has been implemented in the working platform of Python, Matlab with the following system specification and the simulation results are discussed below.

OS: Windows 10

Software: VMware, Python, Matlab

RAM: 8 GB RAM

Processor: Intel i3

4.2 Performance metrics of the proposed system

The Performance metrics of the proposed Multi Optimized Job scheduling Framework for VM with enhanced migration and to develop an adaptive scheduling approach and the achieved outcome were explained in detail in this section.

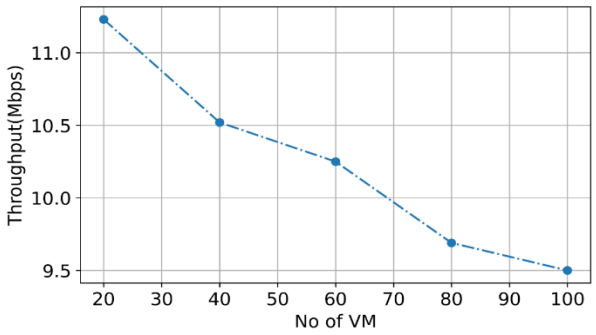

Figure 5: Throughput of the proposed system

Figure 5 depicts the throughput of the suggested system when the number of VMs is changed. As the number of VMs is increased, the proposed system's throughput reaches a minimum of 9.5 Mbps and a maximum of 11.4 Mbps when the number of VMs is decreased. The planned system's throughput has decreased by using the runner root algorithm with consider internationally.

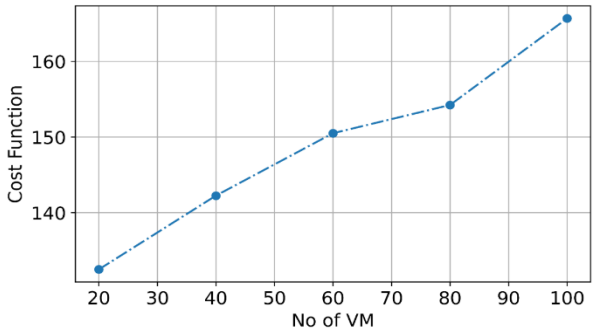

Figure 6: Cost function of the proposed system

Figure 6 depicts the Cost function of the suggested approach for adjusting the number of VM. When the number of VMs is increased to 100, the Cost function of the proposed system reaches a maximum value of 168 and the lowest value of 132 when the number of VMs is decreased to 20. Using the optimal cost function has increased the cost function of the suggested system.

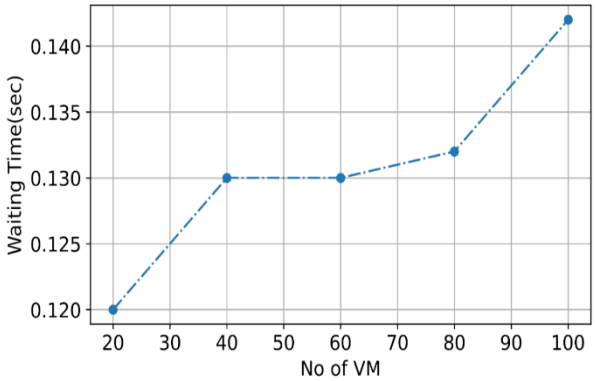

Figure 7: Waiting time of the proposed system

Figure 7 depicts the suggested system's waiting time for varying the number of virtual machines. The Waiting time of the suggested system reaches a maximum value of 0.144 sec. when the number of VMs is increased to 100 and the lowest value of 0.120 when the number of VMs is decreased to 20. The suggested system's waiting period has grown longer by using the steepest descent algorithm with national level.

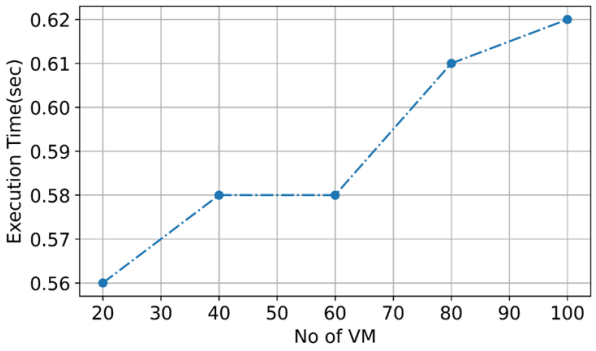

Figure 8: Execution time of the proposed system

The execution time of the proposed system for varying the number of VM has been shown in figure 8. The execution time of the proposed system achieves a maximum value of 0.62, when the number of VMs is increased to 100 and attains a minimum value of 0.56, when the number of VMs is reduced to 20. The execution of the proposed system has increased by differential evaluation algorithm.

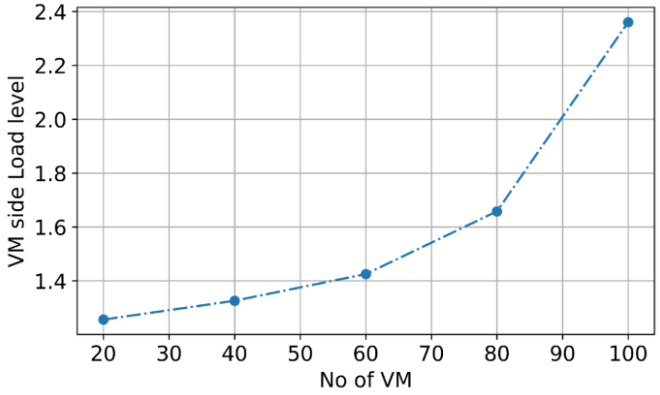

Figure 9: VM side load level of the proposed system

The VM side Load Level of the proposed system for varying the number of VM has been shown in figure 9. The VM side Load Level of the proposed system achieves a maximum value of 2.4, when the number of VMs is increased to 100 and attains a minimum value of 1.3 when the number of VMs is reduced to 20. The VM side Load Level of the proposed system has increased using recursive algorithm.

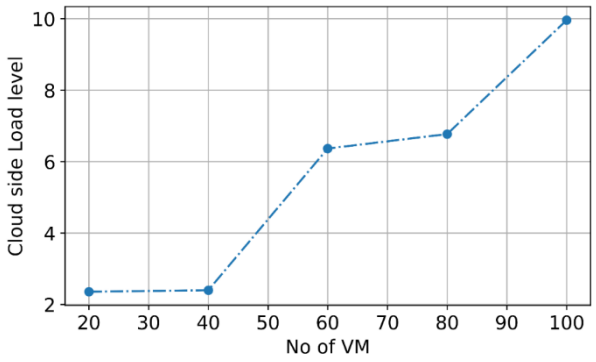

Figure 10: Cloud side load level of the proposed system

The cloud side level of the proposed system for varying the number of VM has been shown in figure 10. The cloud side load level of the proposed system achieves a maximum value of 10, when the number of VMs is increased to 100 and attains a minimum value of 2.4, when the number of VMs is reduced to 20. The Cloud side load level of the proposed system has increased by using active inactive data migration algorithm.

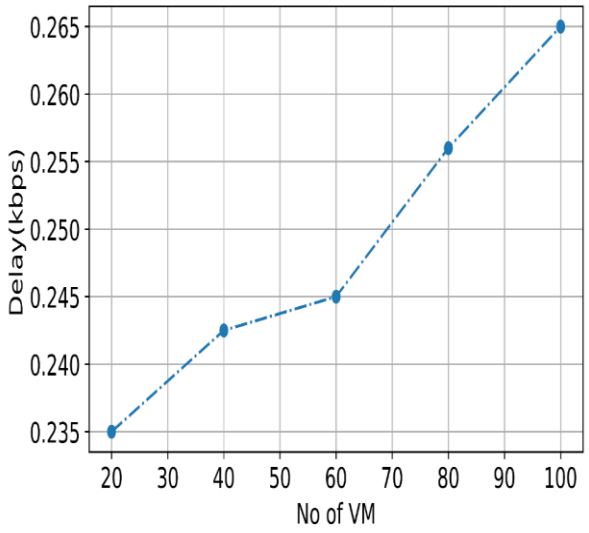

Figure 11: Delay of the proposed system

The delay of the proposed system for varying the number of VM has been shown in figure 10. The delay of the proposed system achieves a maximum value of 0.265 Kbps, when the number of VMs is increased to 100 and attains a minimum value of 0.235, when the number of VMs is reduced to 20. The delay of the proposed system has increased by using automata cellular learning function.

4.3 Comparison of Proposed model with Previous Models

This section highlights the proposed adaptive scheduling approach for effective VMs migration and to provide efficient service of physical servers with varying functionality during workflow execution by comparing it to the outcomes of existing approaches such as FFD [29], VMR [7], CLA-EC [15] and showing their results based on various comparisons is given below.

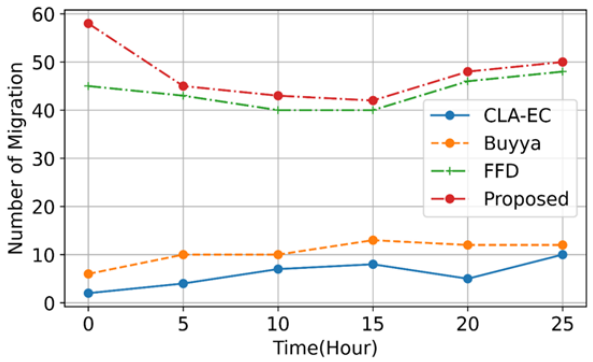

Figure 12: Comparison of migrations

Figure 12 shows a comparison of the number of migrations of the proposed model with existing techniques such as CLA-EC, Buyya, FFD. Whereas the comparison of number of migrations attains a maximum time. The number of migrations of the proposed system achieves a minimum value of 50, when the time s increased to 25 hrs and attains a maximum value of 60, when the time is reduced. Hence the proposed system achieves a smaller number of migration than the existing technique CLA-EC.

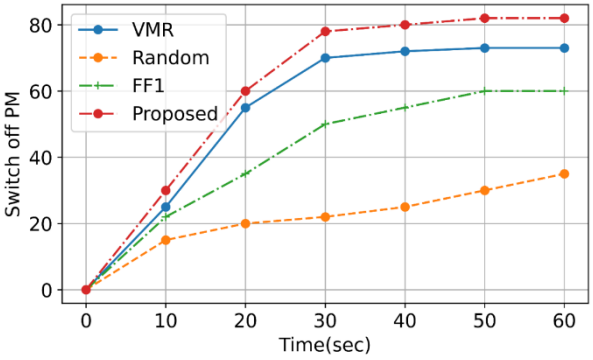

Figure 13: Comparison of Switch off PM.

The comparison of the switch off PM of various models is shown in figure 13. The proposed model has a switch off PM of 80% compared to existing models. The graph also indicates the switch off PM of an increase in the time. Hence the proposed model has achieved high switch off PM, which is compared with the existing techniques VMR.

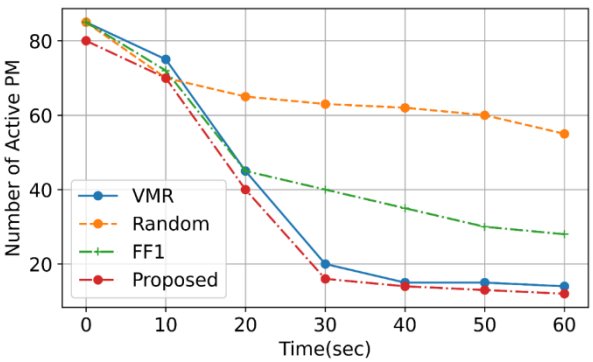

Figure 14: Comparison of Active PMs

The comparison of the number of active PM of various models is shown in figure 14. The proposed model has several active PM of 25% less than existing models. The graph also indicates the number of active PM of adecrease in the time. Hence the proposed model has a smaller number of active PM than the existing technique VMR.

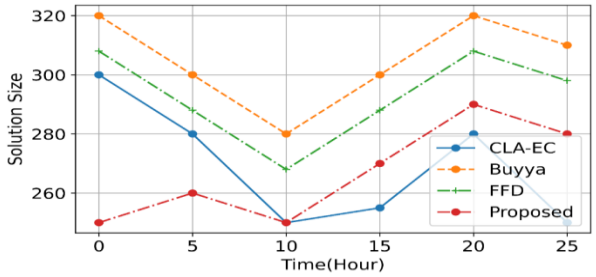

Figure 15: Comparison of Solution size

The comparison of the solution size of various models is shown in figure 15. The proposed model has a solution size of 280 less than existing models. The graph also indicates the solution size PM of a decrease in the time. Hence the proposed model has less solution size, which is compared with the existing technique CLA-Ec. Overall, the proposed model shows that it is more efficient and more accurate when compared to previous models such as CLA-EC, Buyya, FFD, VMR, and Random, which involves VM migration for efficient service of physical servers with varying functionality during workflow execution. The proposed system achieves less through put 9.5 Mbps in a 100 number of VM, when compared to other existing techniques, its Cost function value is 168 which is higher than the existing techniques, and its waiting time is 0.144 sec higher than the existing techniques. This proves that the proposed system performed well when compared to other existing techniques like CLA-EC, Buyya, FFD, VMR, and Random.

5. Conclusion

The Runner root algorithm is deployed by the suggested multi-level-optimized scheduling algorithm with VM migration for scheduling workflows tasks in a multi-cloud to minimize traffic and congestion with efficient work schedule and resource allocation algorithm based on steepest descent method, which is address the issues of flexibility of the resource management and reduces the computation processing time. Where the throughput is reduced 9.5 Mbps, when the migration is increased by using the proposed algorithm and utilizing the steepest descent algorithm at the national level, the waiting time of the suggested system has increased by 0.140 sec. The existing systems such as CLA-EC, FFD & VMR have the number of migrations as 10, 58, and 12. The proposed system achieves 50 no. of migrations. Moreover, the proposed system achieves 25% less than the no. of active Pm, which is compared with existing techniques by using the Data replacing approach. Hence, resource allocation for combined task scheduling & migration is considered when using multi-level optimization. Recursive algorithm to determine the execution duration to prevent erratic migration across virtual machines. Hence the proposed model performs well. Thus, the proposed system has been used to perform a better task scheduling in a complex multi cloud environment with higher flexibility, lowest economic cost, and low computational time according to the results.

6. ACKNOWLEDGEMENT

Our thanks to the experts who have contributed towards development of this paper.

7. REFERENCES

[1] Shukri, S. E., Al-Sayyed, R., Hudaib, A., &Mirjalili, S. (2021). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Systems with Applications*, *168*, 114230.

[2] Alouffi, B., Hasnain, M., Alharbi, A., Alosaimi, W., Alyami, H., &Ayaz, M. (2021). A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies. *IEEE Access*, *9*, 57792-57807.

[3] Orazio, T., Domenico, C., &Pietro, M. (2021). TORCH: a TOSCA-Based Orchestrator of Multi-Cloud Containerised Applications. *Journal of Grid Computing*, *19*(1).

[4] Pinto, A. R. N. (2021). Multi-Site and Multi-Cloud Deployment of Complex Information Systems.

[5] Bello, S. A., Oyedele, L. O., Akinade, O. O., Bilal, M., Delgado, J. M. D., Akanbi, L. A., ... &Owolabi, H. A. (2021). Cloud computing in construction industry: Use cases, benefits and challenges. *Automation in Construction*, *122*, 103441.

[6] Pandey, Ashish, Prasad Calyam, Zhen Lyu, and Trupti Joshi. "Fuzzy-Engineered Multi-Cloud Resource Brokering for Data-intensive Applications." In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 257-266. IEEE, 2021.

[7] Ali, R., Shen, Y., Huang, X., Zhang, J. and Ali, A., 2017, July. VMR: virtual machine replacement algorithm for QoS and energy-awareness in cloud data centers. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)* (Vol. 2, pp. 230-233). IEEE.

[8] Sadeeq, M. M., Abdulkareem, N. M., Zeebaree, S. R., Ahmed, D. M., Sami, A. S., &Zebari, R. R. (2021). IoT and Cloud computing issues, challenges and opportunities: A review. *Qubahan Academic Journal*, *1*(2), 1-7.

[9] Tang, X. (2021). Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems. *IEEE Transactions on Cloud Computing*.

[10] Shahidinejad, A., Ghobaei-Arani, M., &Masdari, M. (2021). Resource provisioning using workload clustering in cloud computing environment: a hybrid approach. *Cluster Computing*, *24*(1), 319-342.

[11] Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., &Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. *IEEE Access*, *9*, 41731-41744.

[12] Cai, X., Geng, S., Wu, D., Cai, J., & Chen, J. (2020). A Multicloud-Model-Based Many-Objective Intelligent Algorithm for Efficient Task Scheduling in Internet of Things. *IEEE Internet of Things Journal*, *8*(12), 9645-9653.

[13] Zhang, B., Zeng, Z., Shi, X., Yang, J., Veeravalli, B., & Li, K. (2021). A novel cooperative resource provisioning

strategy for Multi-Cloud load balancing. *Journal of Parallel and Distributed Computing*, *152*, 98-107.

[14] Masdari, M., &Zangakani, M. (2020). Efficient task and workflow scheduling in inter-cloud environments: challenges and opportunities. *The Journal of Supercomputing*, *76*(1), 499-535.

[15] Gupta, A. and Namasudra, S., 2022. A novel technique for accelerating live migration in cloud computing. *Automated Software Engineering*, *29*(1), p.34.

[16] Khurana, S., & Singh, R. (2020). Workflow scheduling and reliability improvement by hybrid intelligence optimization approach with task ranking. *EAI Endorsed Transactions on Scalable Information Systems*, *7*(24).

[17] Nabi, S., Ibrahim, M., & Jimenez, J. M. (2021). DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing. *IEEE Access*, *9*, 61283-61297.

[18] Sujana, J., Raj, R. V., &Revathi, T. (2022). Fuzzy-Based Workflow Scheduling in Multi-Cloud Environment. In *Operationalizing Multi-Cloud Environments* (pp. 201-215). Springer, Cham.

[19] Xie, F., Yan, J., &Shen, J. (2020, February). A Bandwidth and Latency Based Replica Selection Mechanism for Data-Intensive Workflow Applications in the Multi-Cloud Environment. In *Proceedings of the Australasian Computer Science Week Multiconference* (pp. 1-8).

[20] Ulabedin, Z., &Nazir, B. (2021). Replication and data management-based workflow scheduling algorithm for multi-cloud data centre platform. *The Journal of Supercomputing*, 1-30.

[21] Jena, Tamanna, and J. R. Mohanty. "GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing." *Arabian Journal for Science and Engineering* 43, no. 8 (2018): 4115-4130.

[22] Ramasubbareddy, Somula, and R. Sasikala. "RTTSMCE: a response time aware task scheduling in multi-cloudlet environment." *International Journal of Computers and Applications* 43, no. 7 (2021): 691-696.

[23] Cai, X., Geng, S., Wu, D., Cai, J., & Chen, J. (2020). A Multicloud-Model-Based Many-Objective Intelligent Algorithm for Efficient Task Scheduling in Internet of Things. IEEE Internet of Things Journal, 8(12), 9645-9653.

[24] Chen, Z., Lin, K., Lin, B., Chen, X., Zheng, X., &Rong, C. (2020). Adaptive Resource Allocation and Consolidation for Scientific Workflow Scheduling in Multi-Cloud Environments. IEEE Access, 8, 190173-190183.

[25] Farid, M., Latip, R., Hussin, M., & Hamid, N. A. W. A. (2020). Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. IEEE Access, 8, 24309-24322

[26] Thirumalaiselvan, C., and V. Venkatachalam. "A strategic performance of virtual task scheduling in multi cloud environment." *Cluster Computing* 22, no. 4 (2019): 9589-9597.

[27] XAVIER, VM ARUL, AND ANNADURAI, S. (2018) Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. Cluster Computing, pp. 1- 11

[28] Hamad, S.A., Omara, F.A.: Genetic-based task scheduling algorithm in cloud computing environment. Int. J. Adv. Comput. Sci. Appl. 7(4), 550–556 (2016)

[29] Zhang, B., Zeng, Z., Shi, X., Yang, J., Veeravalli, B. and Li, K., 2021. A novel cooperative resource provisioning strategy for Multi-Cloud load balancing. *Journal of Parallel and Distributed Computing*, *152*, pp.98-107.

[30] Tsakalozos, K., Verroios, V., Roussopoulos, M. and Delis, A., 2017. Live VM migration under time-constraints in share-nothing IaaS-clouds. *IEEE Transactions on Parallel and Distributed Systems*, *28*(8), pp.2285-2298.

[31] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, ``An algorithmfor network and data-aware placement of multi-tier applications in cloud data centers," *J. Netw. Comput. Appl.*, vol. 98, pp. 65_83, Nov. 2017.