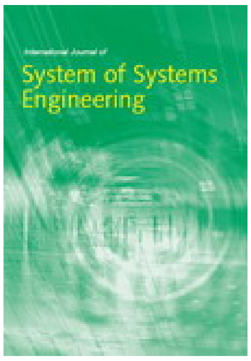


International Journal of System of Systems Engineering

ISSN online: 1748-068X - ISSN print: 1748-0671
https://www.inderscience.com/ijsse

Improved round-robin rule learning for optimal load balancing in distributed cloud systems

S. Sree Priya, T. Rajendran

DOI: 10.1504/IJSSE.2023.10053120

Article History:

Received:	07 June 2022
Accepted:	03 August 2022
Published online:	16 February 2023

Improved round-robin rule learning for optimal load balancing in distributed cloud systems

S. Sree Priya* and T. Rajendran

PG and Research Department of Computer Science,
Government Arts & Science College (Affiliated to
Bharathiar University, Coimbatore),
Kangeyam-638108, Tamil Nadu, India
Email: sspriya.sivakumar@gmail.com
Email: rajendran_tm@yahoo.com
*Corresponding author

Abstract: A newly emerging technology that distributes virtualised computer resources across the internet is referred to as 'cloud computing' and is gaining popularity. These clouds' ability to evenly distribute the load is vital. Load balancing in cloud computing distributes dynamic workloads so no database server is overcrowded or underloaded (LB). As a result, a dynamic load balancing technique in the cloud may contribute to improved service stability and resource utilisation. In this study, we model a load balancing system that balances the cloud's available resources using a rule-based round-robin method. Cloud parameters are used to calculate and allocate resources using a rule-based round-robin technique. The study analyses cloud metrics to determine cloud loads and resources. The cloud sim tool is used to allocate resources, and its effectiveness is tested against a variety of performance measures such as overhead, migration time, and throughput rate. Simulation results show that rule-based round-robin improves cloud performance. When compared to existing algorithms, the proposed methodology improves task performance by about 4%.

Keywords: optimisation; load balancing; cloud computing; rule-based round robin; overcrowded or underloaded (LB); cloud parameters; CloudSim-3.0.3; weighted round-robin technique; virtual machines.

Reference to this paper should be made as follows: Priya, S.S. and Rajendran, T. (2023) 'Improved round-robin rule learning for optimal load balancing in distributed cloud systems', *Int. J. System of Systems Engineering*, Vol. 13, No. 1, pp.83–99.

Biographical notes: S. Sree Priya is an Assistant Professor at Guru Nanak College (Autonomous), Chennai, and a PhD candidate at Government Arts and Science College, Kangeyam (Affiliated to Bharathiar University, Coimbatore). She has taught information security, big data, cloud security, cloud computing and management, Python & R Programming, and IT for over 12 years, gathering requirements, conducting reviews, designing products, documenting, and maintaining. Her 2021 Java book was for beginners. *UGC-CARE Wesleyan Journal of Research*. She's interested in data mining, science, and the cloud.

T. Rajendran has 22 years of teaching experience in various Colleges. He has completed his PhD degree in 2012 at Anna University, Chennai in the Department of Information and Communication Engineering. Now, he is working as an Assistant Professor at Government Arts and Science College, Kangeyam, Tamilnadu, India. His research interests include distributed systems, web services, network security and web technology. He is a life member of ISTE and he is also a member in various professional societies like IAENG and IACSIT. He has published more than 75 papers in international/national conferences/journals. He has organised 3 AICTE sponsored FDP, 3 DRDO sponsored seminar/conference, 3 CSIR sponsored seminar/symposium/conference, 9 self supporting workshops and 3 national conferences for the benefit of faculty members and students. He is a reviewer of *Journal of Engineering and Technology Research*, and 4 international conferences. He was honoured with Best Professor Award 2012 by ASDF Global Awards 2012, Pondicherry.

1 Introduction

The cloud is a shared computing environment in which computing resources are made available to devices on demand. Since the cloud, it is now possible to charge a fee for the usage of a certain resource for a set period of time. Virtual machines (VMs) are the primary execution units in cloud computing, which is why it is called virtualisation. It is referred to as ‘virtualisation’ and refers to the process of setting up, running, and maintaining a shared computer environment in which a wide range of software and resources can be utilised. Load balancing is essential for both maintaining system stability and preventing poor performance from occurring simultaneously. As a result, developing an algorithm that can more effectively divide the workload among virtual machines is essential (Cao et al., 2014).

Gharooni-Fard et al. (2010) and Ghanbari and Othman (2012) described the load balancing strategies available, including round-robin (RR), weighted RR, first-come, first-served (FCFS), dynamic load balancing (ESCE), and throttled algorithm. An environment known as CloudSim-3.0.3 is used to carry out the cloud computing research and analysis in this case. Cloud computing system components such as VMs and resource provisioning strategies that may be described are supported by both system and behaviour modelling, which makes it a powerful tool for architects. It may be used to model and simulate cloud computing scenarios that involve both single-cloud and networked clouds, among other things. In internetworked cloud computing settings, it enables customised APIs for embedding load balancing techniques in VMs, as well as strategies for distributing VMs to customer requirements. It has the ability to use virtualised services on-the-fly in order to adapt to changing needs. Fard and Deldari (2008) demonstrated static and dynamic scheduling, as well as personalised load balancing using improved weighted round robin (IWRR) for achieving better VM utilisation under different load patterns. This was accomplished for the application jobs in a considerably shorter amount of time than the rest of the team, which was a big advantage.

The cloud dynamic pool of VMs must be used to assign computational jobs based on the requirements of each task and the pressure placed on the VM processors. Client inquiries can be routed to any of the cloud data centres that are available. As a result, according to cloud management standards and taking into consideration the load on each individual VM, the data centre sends similar requests to their most appropriate virtual machines. When applying the round-robin policy, factors such as resource capabilities, work priority, and task length are not taken into consideration. As a result, the jobs that are more important and time-consuming receive the quickest responses. On the other hand, the proposed and implemented technique takes into account the length and priority of tasks in addition to the task length while selecting the ideal VM to perform tasks with the quickest reaction times.

One of the most important tasks in cloud computing is load balancing, which involves balancing network loads based on factors including execution time, reaction time, quality of services (QoSs), fault tolerance, load balance, scalability, and resource utilisation. In this paper, we use an Enriched Resourceful Weighted Round Robin technique to create a load-balancing model that attempts to balance the cloud based on a number of input factors. Finding the intensity of loads in a cloud environment is aided by the factors of response time, processing, and resource utilisation. The round-robin approach predicts the load and assigns the resources based on this, and these inputs define the dynamic character of load balancing in clouds. In order to maximise the performance of virtual machines, static and dynamic load balancing proposed by Rahman et al. (2013) will be used to detect the interdependency of different workloads, identify underutilised virtual machines, and avoid overloading any of the VMs. Thanks to this new choice, the job length consideration can assist in planning jobs at any time of day or night and providing responses in a very short period of time. The efficient scheduling provided by this technique will also lower the workload on a virtual machine, resulting in a reduction in the number of task migrations.

This study develops IWRR, where a job normally comprises a number of tasks that are interconnected with one another. It is feasible for a job to fulfil its entire processing instructions by utilising a large number of VMs, each of which performs a specific task. Depending on the task design and availability, it may also make use of several processing units contained within a single virtual machine.

2 Related works

Task scheduling in the cloud is primarily reliant on virtual machine load balancing. When some VMs are overcrowded, the burden must be shared with other VMs that are not overloaded in order to maximise resource utilisation and reduce task completion times for everyone. Furthermore, the load balancing algorithms must include the overhead impact on determining resource use and task relocation when evaluating resource use and task relocation. When it comes to job completion, the VM generally employs two unique approaches: space sharing and time sharing. The tasks will be completed in a sequential manner through the use of the space-sharing system. suggesting that only one job is being executed per CPU/core at a given time. All of the tasks that have been assigned to this VM should be queued up and completed as soon as possible. When the overloaded VM has a waiting list of tasks, the underloaded VM can be assigned one of these tasks to make the task migration procedure on this space sharing approach a little bit easier on everyone involved.

A time-shared system, on the other hand, has some advantages over a traditional system in that jobs are completed simultaneously and in a time-sliced manner (Lazar et al., 2021). Because all jobs are done in time slices, task migration in load balancing is made more complex in this scenario as a result. As a result of the time-slicing strategy, a specific percentage of jobs will be completed nearly 90% of the time. Due to the previously completed component of the job on high-traffic computers and the earlier execution influence on other job execution timings on high-traffic machines, task migration from high-traffic machines to low-traffic machines is time-consuming and expensive (Laxmi Lydia et al., 2020). This method/process contributes to the achievement of the quickest possible execution time on the cloud. Job allocation algorithms should take into account the interdependencies and unpredictable nature of jobs involving a large number of tasks when allocating work to virtual machines. Algorithms should be able to function in both homogeneous and heterogeneous environments, as well as for a variety of job durations. As a result of our research and the development of our algorithm technique, we were able to achieve our target (El-Gamal et al., 2020).

Based on the Honey Bee algorithm (Krishna, 2013), it was hypothesised that the method would produce a well-balanced load across VMs in order to maximise throughput while also maintaining a balance between the priority of jobs on the VMs. It is as a result of this that the amount of time that tasks in the queue must wait is minimal. With the use of this strategy, it was possible to lower the average execution time while simultaneously improving queue waiting time. This method is appropriate for non-preemptive independent jobs in heterogeneous systems that do not require preemption.

Gharooni-Fard et al. (2010) explained how to use cloud computing and a queuing strategy to manage a cluster of heterogeneous multicore servers of varied sizes and speeds. The optimisation of performance and the decrease in power consumption were major factors. In particular, the study is essentially an experiment to determine whether or not it is possible to model power.

Ijaz et al. (2013) presented a weighted round-robin scheduling strategy for cloud infrastructure services that takes into account the length of jobs and the capability of the resources being used to schedule them. VMs are utilised to their maximum potential through the use of a combination of scheduling, which determines the workload length and their resource requirements, and then anticipates which VMs will be underutilised and which VMs will be overloaded ahead of time. We take into consideration the interconnectedness of the responsibilities at different levels. Despite the fact that task migrations were performed under heavy loads, load balancing was not taken into consideration (Aoudni et al., 2022).

Munir et al. (2013), Hu et al. (2020) and Kadri et al. (2012) proposed how to use a grid-based scheduling strategy to schedule dependent jobs. Kiruthiga et al. (2020), Lin et al. (2009) presented an efficient mapping of the DAG-based application to the underlying data structure. This algorithm uses the list scheduling approach as its scheduling strategy. Basker et al. (2014) described a noncritical scheduling strategy that was based on the earliest-finish principle. When this strategy is applied to a heterogeneous computing environment, the performance of the system is improved. A comparable issue to this one has been identified by Lee et al. (2009) in terms of load distribution in cloud computing. A comparison was made between the soft computing-based techniques (Subramani and Vijayalakhsmi, 2016).

An efficient workflow scheduling setting reduces the time it takes for workflows to complete and keeps scheduling overhead to a bare minimum (Rahman et al., 2013). Robine and Michel (2004) proposed a genetic algorithm called the Chaos Genetic Algorithm that was created with the goal of handling the scheduling problem while taking into consideration the deadline and cost specified by the user, among other factors. Make use of this algorithm if you want better results in a shorter amount of time. The methods (Shenai, 2012; Thakur and Goraya, 2022; VinayaKulkarni et al., 2020; Xu et al., 2013) deal with concerns that are comparable to those addressed. When developing (Xu et al., 2011) a cloud task scheduling system, the priority of jobs was employed as a fundamental QoS criterion. The literature has been reviewed in light of the goal, and an algorithm has been designed to achieve it.

According to Hu et al. (2020), both the network and the host can be used to measure the risk associated with network security. Additionally, it suggests a brand-new framework for a multidimensional assessment method that entails two steps—risk calculation and risk identification—to identify network security risks. The research suggests a mechanism for a multidimensional hierarchical index that evaluates the risk related to cyber security at the risk identification stage (Nandakumar et al., 2021). The three different elements of the security system's status – vulnerabilities, threats, and fundamental operation – are what direct the data collection process.

Based on the services of security (Xu et al., 2009), advised network security. The structure and use of network security approaches are described in detail in the study. The amount of data resources that today's society uses enhances the need for security. Some of these apps' security is achieved via encryption techniques like Data Encryption Standard (DES). However, there are several flaws in the current algorithms that must be fixed. In order to secure the confidentiality and privacy of data, new algorithm mechanisms are now the focus of study in the domains of networks and data security. The current scheduling algorithms used for tasks are as follows:

2.1 First-come, first-served (FCFS) scheduling

In this algorithm, the main concept of First Come, First Serve has been carried out. Here, the first task that has been scheduled will be carried out first, followed by the next task scheduled.

2.2 Shortest-job-next (SJN) scheduling

In this algorithm, the main concept of 'Shortest Job Next' has been carried out. Here the task has been scheduled according to the total number of tasks scheduled. Here, the task which can be finished soon is given priority, followed by consecutive short jobs.

2.3 Priority scheduling

In this algorithm, the main concept of priority scheduling has been carried out. Here the task has been scheduled based on its priority. Each and every task has a different priority of execution. According to the needs of the task, the task's priority is given and executed.

2.4 Shortest remaining time scheduling

In this algorithm, the main concept of shortest remaining time scheduling has been carried out. Here the task has been scheduled based on priority of time. Each and every task has a different time of execution. According to the needs of the task, the task is given and executed based on time.

2.5 Multiple-level queues scheduling

In this algorithm, the main concept of Multiple-Level Queues Scheduling has been carried out. Here the task has been scheduled based on the priority of multiple-level queues. Each and every task has a different time of execution and queues. According to the needs of the queues, tasks are being executed.

2.6 Multilevel feedback queues scheduling

In this algorithm, the main concept of Multiple-Feedback Queues Scheduling has been carried out. Here the task is being scheduled based on the priority of Multiple-Feedback Queues. Each and every task has a different time of execution and feedback queues. The Feedback Queues task has been executed according to plan.

2.7 Highest response ratio next scheduling

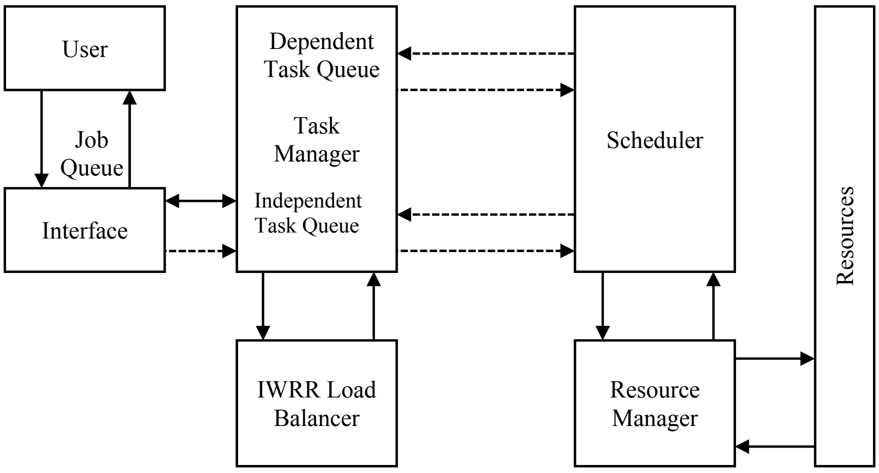
In this algorithm the main concept of Highest Response Ratio Next Scheduling is been carried out. Here the task is being scheduled based Highest Response Ratio. Each and every task has different time of execution and Highest Response Ratio. According to the need of the Highest Response Ratio task is been executed.

3 Proposed method

As shown in Figure 1, the scheduler determines which VMs are best suited for each task and then distributes those tasks to those VMs using the proposed technique. Jobs that arrive at runtime are assigned to VMs that are least likely to be overcrowded at that precise point in time, allowing them to be finished as fast and efficiently as possible. The below architecture diagram consists of blocks such as User, Interface, Task Manager, Scheduler, Resource Manager, IWRR Load Balancer, and finally resources. Here, the Task Manager plays a vital role in scheduling the task which has been received from the user. The user makes a clear debut in giving the total task to the Task Manager. The block interface acts as the interface link between the user and the task manager. The Task Manager gives the instruction to the Scheduler, and the Scheduler is connected to the Resource Manager, which has been filled with enough resources. Both Task Manager and Scheduler Tasks will be balanced by the IWRR Load Balancer, which acts as our proposed system that helps in balancing data during task scheduling.

Based on the current condition of the resources, the load balancer determines whether a job should be transferred from a heavily loaded to a lightly loaded or idle loaded VM during runtime. In addition, it communicates with each individual VM resource prober in order to acquire information such as their current load and the number of jobs running or waiting to be processed on each VM through the Resource Monitor. The user provides the scheduler with the task requirements, which includes the time of the tasks that need to be done. The scheduler then takes operational decisions based on the information provided by the user.

Figure 1 Load balancing design



3.1 *Load balancing design*

A request for a job is sent over the interface, and the task manager examines it to see whether it has any dependencies or independent tasks. This module accepts a job and determines if it consists of a single task or if it contains a number of tasks that must all be done. When there are numerous jobs to do, the interdependence of the various tasks is confirmed. Both the dependent and independent task queues have been identified and located. Due to the fact that the scheduler will be notified of the dependent tasks, parent tasks will be able to be scheduled after the completion of child tasks. It is planned to include a dependency task queue for tasks that are dependent on the execution of other tasks on the VMs.

In contrast, once the tasks in the queue are completed, and the VMs are allocated to their parent jobs, the independent task queue contains jobs that are fully independent of one another. The scheduler receives information about the job queues that are independent and dependent on one another. The resource management system supplies this scheduler with information about the resources that are now accessible. An algorithm is used to identify which virtual machine is most appropriate for a certain task based on the processing capabilities of each virtual machine.

In addition, each virtual machine maintains its own distinct ExecutionList, PauseList, and WaitingList lists of tasks. The jobs currently running on the VM are listed in the ExecutionList, while the jobs currently paused on the VM are listed in the PausedList. During each request, each of the virtual machines transmits a list of its execution, pause, and waiting lists to the WaitingList Queue, which uses this information to calculate which VM is being utilised the least for each request that comes in. After that, the scheduler will be informed of the VMs that are used the least frequently.

In order to gather the capabilities of all of the VMs, the resource management team communicates with each one individually to determine the number of processors and processing capacity available for each of them. VMs are weighted according to their processing capacity, which is determined by the virtual machine processing capacity allotted to them. Here you can also see how much memory has been allocated to each of the virtual machines and how much of it has been configured for usage. The load balancer is in charge of determining the ratio of active workloads to virtual machines on the network. There are times when a VM needs to be identified for a job, and the scheduler communicates this information to them by referring to the job execution list of each VM in the system pool.

After the job has been identified, it will be assigned to the appropriate VM. The computing resources available in the configured data centres are made up of hosts, virtual machines, and processing elements. It is necessary to monitor for inactivity and severe strain on resources in order to correctly assign task requests to the most useful resource.

3.2 *Load imbalancing factor (LIF)*

The total VM loads is what this term refers to.

$$L = \sum_{i=1}^{k} l_i \tag{1}$$

where i – total VMs.

$$LPC = \frac{L}{\sum_{i=1}^{m} c_i} - \tag{2}$$

$$T_i = LPC * c_i \tag{3}$$

As an example of load per unit capacity, consider the following:

where c_i – node capacity.

$$\text{if } VM \begin{cases} < |T_i - L| & \text{Underloaded} \\ > |T_i - L| & \text{Overloaded} \\ = |T_i - L| & \text{Balanced} \end{cases} \tag{4}$$

The LIF for a VM is calculated in the following way:

It is possible to allow task migration from the overloaded to underloaded VM when the cloud load goes below the threshold and hence the difference between the two is equal to μ_i.

If the overall load of all the virtual machines running on a virtual machine is less than the threshold value established for that particular VM, the virtual machine is called underloaded. In order to prevent the underloaded VM from becoming overheated, the overloaded VM transfers its load to it until it exceeds the threshold value of λ_j, as illustrated in the diagram above.

The overloaded VM is moved until the load falls below the threshold value. The result is that the virtual machine cannot become overloaded if it is operating at a low enough load level. The amount of load that can be transferred from a virtual machine that is underloaded should fall between μ_i and λ_j in the range.

3.3 *Improved round robin*

As part of the proposed IWRR technique, jobs are distributed to the most appropriate virtual machines based on information about the virtual machines such as their processing capability, load on the VMs, and duration of tasks that have been received in order of priority. The processing capability of the virtual machine, the number of incoming jobs, and the length of each task are all taken into consideration in the static scheduling of this algorithm.

The dynamic scheduling algorithm is used in conjunction with the VM load and the previously specified information to determine where a job should be assigned. In some cases, run-time data makes the job execution more difficult, resulting in a longer task execution time than anticipated.

As a result, the load balancer intervenes and transfers pending work from the severely overloaded VMs to the other under-utilised or unutilised VMs, saving the scheduling controller from failure. A resource probe is used by the load balancer to determine whether virtual machines are under-utilised or not used at all. If there are no unutilised virtual machines, load balancing will not perform any job migration.

When an under-utilised or unused VM is discovered, the workload will be transferred there from the overburdened one. Performing the resource load analysis is only performed once any of the VM responsibilities have been finished. As a result, when it comes to evaluating the resource load, there is no overhead on the virtual machines. Task migrations across VMs and resource probe executions within VMs will both be decreased as a result of this change.

3.4 Load balancing algorithm

The load balancer collects the execution time of pending tasks from all of the freshly formed virtual machines and sorts them ascendingly to discover how many tasks are present in each one. An unfinished paper with a long-term pending time is identified by estimating the job execution time in VMs and then assigning it to the VM with the best likelihood of completing it. The final stage in load balancing is to reorganise the order of the tasks in accordance with the altered execution times of each virtual machine (refer to Algorithm 1).

The trained model size is around 1 GB. First, the task schedule work has been stimulated. Then we must identify executing tasks in VM and arrange them on a queue in ascending order. Now estimate total number of tasks in queue using waiting task in VMs. Give condition and If the first queue tasks ≥ 1 now terminate the load balancing logic or Else. If the last queue tasks $\leq$ now terminate the load balancing logic. Now find pending execution time and arrange VMs from low to high pending task and eliminate higher pending task and stop the process.

Algorithm 1 Load Balancing

```
Step 1:  Start
Step 2:  Identify executing tasks in VM and
Step 3:  Arrange the task in ascending order on a Queue.
         a.  Estimate number of tasks in queue using waiting task in VMs
Step 4:  If the first queue tasks $\geq 1$
         a.  Terminate the load balancing logic
```

```
Step 5:  Else
           a. Go to step 6
Step 6:  End
Step 7:  If the last queue tasks ≤ 1
           a. Terminate the load balancing logic
Step 8:  Else
           a. Go to step 3
Step 9:  End
Step 10: Find pending execution time
Step 11: Arrange VMs from low to high pending task
Step 12: Eliminate higher pending task
Step 13: Execute from step 1
Step 14: End
```

4 Results and discussions

The performance of the IWRR algorithm was evaluated in light of the results of the CloudSim simulation. CloudSim simulator classes have been enhanced in order to accommodate the new technique. When heterogeneous and homogeneous job lengths are mixed with heterogeneous resources, the RR, WRR, and IWRR algorithms are used to assess reaction time, the number of work migrations, the total idle task time, and the delayed tasks, among other things. In our research, we have used parameters such as time complexity (TC), average time complexity (TC), space complexity (SC), average space complexity (ASC), and retrieval time (RT).

4.1 Dataset information

This dataset was downloaded from the Kaggle Repository, which is a repository for online databases in general. This collection includes the Aadhar information for individuals who reside in about 8 states across our nation. It has 12 columns and 44,800 rows. The dataset uses a total of 12 attributes in total.

4.2 Time complexity

Time complexity is generally defined as the time taken for the entire process of encryption to decryption. That is the time calculated from the first process of loading the dataset followed by encryption and decryption stages.

$$P_1(n)T_1(n)+P_2(n)T_2(n)+P_3(n)T_3(n) \tag{5}$$

where $T_1(n), T_2(n), T_3(n)\ldots$ represents the execution time.

$P_1(n), P_2(n), P_3(n)\ldots$ indicates the input probability values.

4.3 Average time complexity

$$T(n) = \frac{I}{D} + BiometricProcessTime \tag{6}$$

where I denotes the total input file or text file

D indicates distributed size of each block.

4.4 Average space complexity

Average time complexity is generally defined as the overall time taken by each algorithm in the entire iteration process. The lower the time complexity, the higher the performance.

$$AverageSpaceComplexity = Temp.space + Input_{Space} \tag{7}$$

where Temp. Space denotes Temporary Space.

Input_Space denotes Space occupied by Input File.

4.5 Retrieval time

Average space complexity is defined as the space occupied by the dataset that has been divided into blocks and temporarily stored at some particular place during the entire encryption and decryption process. The lower the space complexity, the lower the retrieval time.

$$RTime = TotalFile \frac{Size}{Average} TimeComplexity \tag{8}$$

where R denotes Retrieval Time.

Total File Size denotes the actual size of the Input File.

Average Time Complexity denotes the overall time consumed by each algorithm.

4.6 Overall execution time on heterogeneous resources with homogeneous task

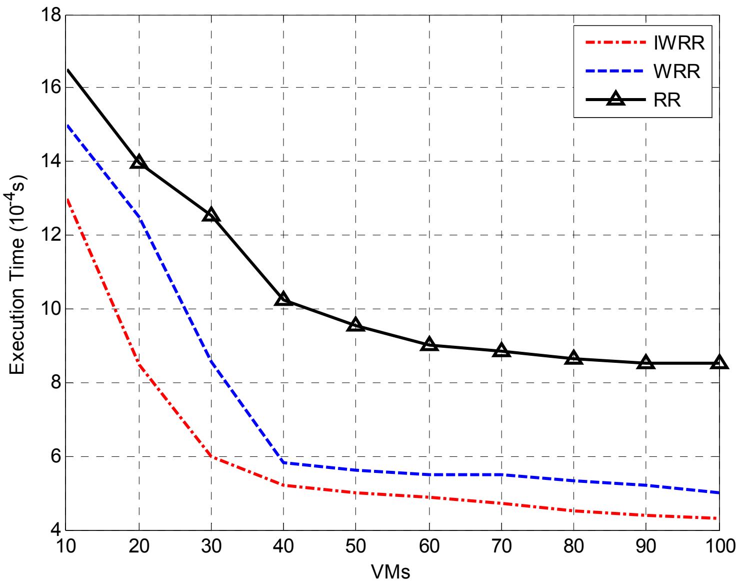
Following the results of Figure 2, IWRR by job length outperforms RR and WRR in the case of the heterogeneous resources and homogeneous workloads domains, respectively. The IWRR static scheduler method considers the length of the job as well as the processing capabilities of the heterogeneous virtual machines when allocating a job to them. Table 1 and Figure 2 represent the Execution Time taken during each set of tasks. From the results we can prove that the proposed IWRR Execution Time ranges from 13 ms to 2 ms whereas existing WRR Execution Time ranges from 15 ms to 5.3 ms and RR Execution Time ranges from 16 ms to 8.1 ms. So it is proved that the proposed IWRR execution time is much less than other existing techniques.

The outcome is that in heterogeneous systems with homogeneous workloads, virtual machines with higher capacity are assigned to a greater number of jobs than virtual machines with lower capacity. This assists in completing the task in a shorter amount of time than previously anticipated. When assessing the current load of the dynamic scheduler, it takes into account all of the virtual machines that have been configured and estimates when the current load will be completed. The scheduler then estimates the completion time of a job for each VM that is created, and this time is added to the current load on each VM by the scheduler. In light of the aforementioned computations, the VM that would complete this particular paper the quickest is identified, and finally the job is allocated to a VM. As a result, this algorithm performs optimally in data centres that have a diverse collection of resources. The following Table 2 and Figure 3 represent the Task Migration during each set of tasks. From the results we can prove that the proposed IWRR Task Migration ranges is 0.1 ms whereas existing WRR Task Migration ranges from 0.5 ms to 0.1 ms and RR Task Migration ranges from 5.5 ms to 0.1 ms. So its proved that proposed IWRR Task Migration is very less than other existing techniques.

Table 1 Execution time

No. of tasks	*IWRR* (ms)	*WRR* (ms)	*RR* (ms)
10	13	15	16
20	9	12	14
30	6	8	12
40	5	6	10
50	4	6	9
60	3	5.8	8.5
70	3	5.8	8.5
80	2	5.5	8.3
90	2	5.5	8.2
100	2	5.3	8.1

Figure 2 Execution time (see online version for colours)

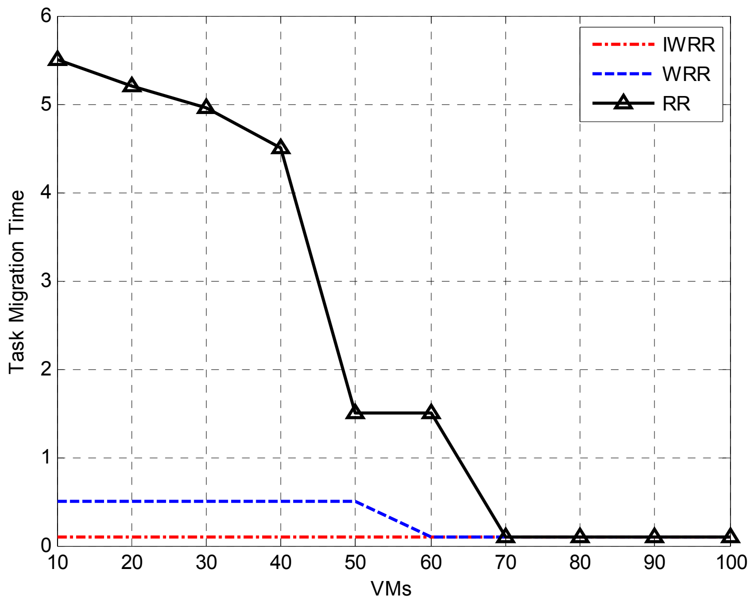


As soon as each task is completed, the IWRR job-length load balancer kicks in and begins balancing the workload. The load balancer identifies the heavily loaded VM and estimates the calculated job completion time for those that are currently running in the most heavily loaded VM and those that are currently running in the least heavily loaded/idle VM. It is planned to transfer one of those jobs from the highly strained virtual machines to the least stressed virtual machine as soon as one of those jobs is completed on the highly stressed VMs (Figure 3).

Table 2 Task migration

No. of tasks	*IWRR* (ms)	*WRR* (ms)	*RR* (ms)
10	0.1	0.5	5.5
20	0.1	0.5	5.2
30	0.1	0.5	5
40	0.1	0.5	4.5
50	0.1	0.1	1.5
60	0.1	0.1	1.5
70	0.1	0.1	0.1
80	0.1	0.1	0.1
90	0.1	0.1	0.1
100	0.1	0.1	0.1

Figure 3 Task migration (see online version for colours)



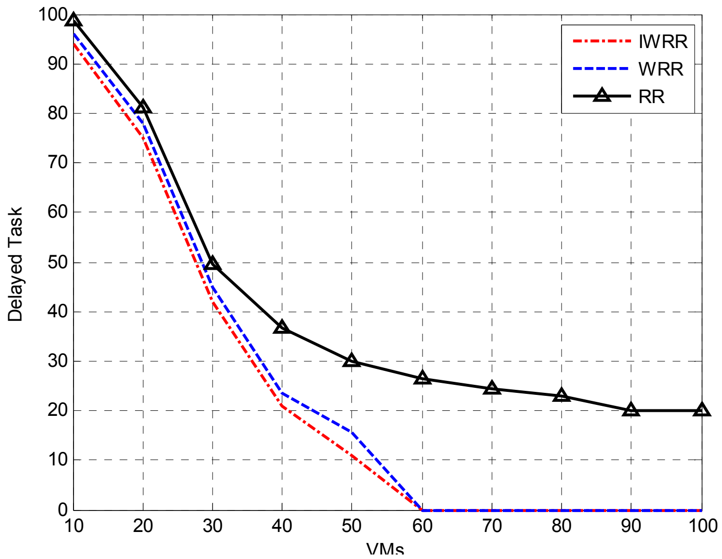
While calculating the WRR, it considers the relationship between VM capacity and overall VM capacity and then distributes the proportionate quantity of work that has arrived to each VM. As a result, it is able to achieve higher levels of performance.

Because of the above computation, jobs that are longer in duration will take longer to finish if they are allocated to virtual machines with low capacity (Figure 4). The following Table 3 and Figure 4 represent the Delayed Task during each set of tasks. From the results we can prove that the proposed IWRR Delayed Task ranges is 92 ms to 0 ms whereas existing WRR Delayed Task ranges from 95 ms to 0 ms and RR Delayed Task ranges from 100 ms to 20 ms. So its proved that proposed IWRR Delayed Task is very less than other existing techniques.

Table 3 Delayed task

No. of tasks	*IWRR* (ms)	*WRR* (ms)	*RR* (ms)
10	92	95	100
20	72	80	80
30	42	50	50
40	20	23	38
50	10	15	30
60	0	0	28
70	0	0	23
80	0	0	22
90	0	0	20
100	0	0	20

Figure 4 Delayed task (see online version for colours)

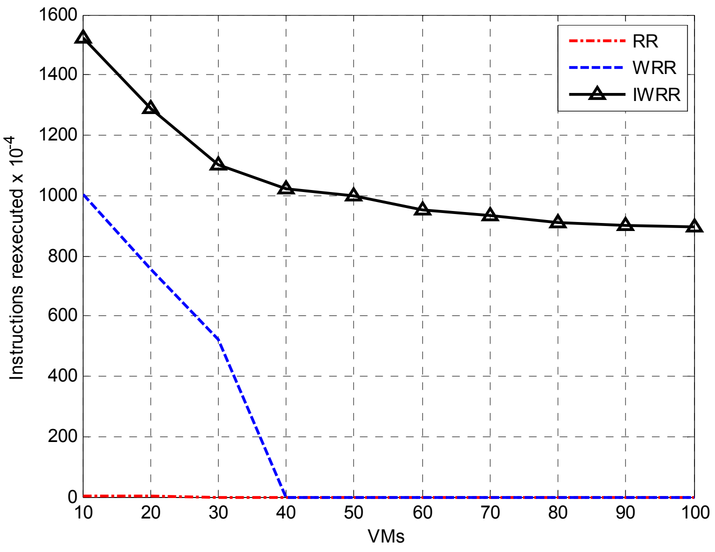


The environment, VM capabilities, and job durations are not taken into consideration by the simple RR (Figure 5). An orderly distribution of VM lists is ensured by assigning them one after the other. As a result, it takes significantly longer time to complete work than the other two algorithms combined. The following Table 4 and Figure 5 represent the Million Instructions Re-executed during each set of tasks. From the results we can prove that the proposed IWRR Million Instructions Re-executed ranges from 1550 to 900 whereas existing Million Instructions Re-executed range from 1000 to 0 and RR Million Instructions Re-executed ranges from 100 ms to 20 ms. So it is proved that the proposed IWRR Million Instructions Re-executed is better than other existing techniques.

Table 4 Million instructions re-executed

No. of tasks	*IWRR*	*WRR*	*RR*
10	1550	1000	1500
20	1300	800	1290
30	1100	500	1100
40	1010	0	1000
50	1000	0	1000
60	980	0	980
70	970	0	970
80	960	0	960
90	950	0	950
100	900	0	950

Figure 5 Million instructions re-executed (see online version for colours)



From the above results obtained, the parameters such as execution time, task migration, delayed task, and million instructions re-executed obtained by the proposed IWRR perform better and in a unique way in all aspects related to task scheduling. It is clearly proved that the proposed IWRR task scheduling is unique as it gives importance to all tasks in the same amount of time and does not give individual priority, which results in better performance. Our proposed IWRR has overcome the major drawbacks, such as Execution Time, Task Migration, Delayed Task, and Millions of instructions re-executed.

5 Conclusion

It is proposed in this research to employ a round-robin technique to construct a cloud-based load balancing system that makes use of the available cloud resources in a fair and equitable manner. Resources are allocated according to a rule-based round-robin technique, with cloud parameters serving as input tools for the algorithm. The IWRR load balancer is operated once each task has been completed. When a job is completed, this ensures that there is no idle time for participating VMs and that the workload is evenly distributed across all of those participating in the activity. Based on the findings of the performance analysis and the experiments, it was determined that the modified weighted round robin technique was the most acceptable solution for the heterogeneous and homogeneous tasks performed with virtual machines. When using this strategy, response time is the most important QoS indicator. According to the findings of the study, several cloud factors were investigated in order to evaluate the loads and available resources in a distributed cloud environment. Our resource allocation is done using cloudsim, and our outcomes are then evaluated in relation to several performance metrics like throughput and migration times, among other things.

References

Aoudni, Y., Donald, C., Farouk, A., Sahay, K.B., Babu, D.V., Tripathi, V. and Dhabliya, D. (2022) 'Cloud security based attack detection using transductive learning integrated with hidden markov model', *Pattern Recognition Letters*, Vol. 157, pp.16–26.

Basker, R., Uthariaraj, V.R. and Devi, D.C. (2014) 'An enhanced scheduling in weighted round robin for the cloud infrastructure services', *International Journal of Recent Advance in Engineering and Technology*, Vol. 2, No. 3, pp.81–86.

Cao, J., Li, K. and Stojmenovic, I. (2014) 'Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers', *IEEE Transactions on Computers. Institute of Electrical and Electronics Engineers*, Vol. 63, No. 1, pp.45–58, doi: 10.1109/tc.2013.122.

El-Gamal, A.H., Mostafa, R.R. and Hikal, N.A. (2020) 'Load balancing enhanced technique for static task scheduling in cloud computing environments', *Internet of Things–Applications and Future*, Springer, Singapore, pp.411–430.

Fard, H.M. and Deldari, H. (2008) 'An economic approach for scheduling dependent tasks in grid computing', *2008 11th IEEE International Conference on Computational Science and Engineering-Workshops*, July, IEEE, San Paulo, pp.71–76.

Ghanbari, S. and Othman, M. (2012) 'A priority based job scheduling algorithm in cloud computing', *Procedia Engineering*, Vol. 50, No. 1, pp.778–785.

Gharooni-fard, G., Moein-darbari, F., Deldari, H. and Morvaridi, A. (2010) 'Scheduling of scientific workflows using a chaos-genetic algorithm', *Procedia Computer Science*, Vol. 1, No. 1, pp.1445–1454.

Ijaz, S., Munir, E.U., Anwar, W. and Nasir, W. (2013) 'Efficient scheduling strategy for task graphs in heterogeneous computing environment', *Int. Arab J, Inf. Technol.*, Vol. 10, No. 5, pp.486–492.

Hu, J., Guo, S., Kuang, X., Meng, F., Hu, D. and Shi, Z., 2020) 'I-HMM-based multidimensional network security risk assessment', *IEEE Access*, Vol. 8, pp.1431–1442.

Kadri, W., Yagoubi, B. and Meddeber, M. (2012) 'Efficient dependent tasks assignment algorithm for grid computing environment', *Proceedings of the 2nd International Symposium on Modelling and Implementation of Complex Systems (MISC'12)*, May, Constantine, Algeria.

Kiruthiga, G. and Mary Vennila, S. (2020) 'Intelligent resource scheduling with neutrosophic knowledge and optimized cache management using cuckoo search method in cloud computing', *International Journal of Intelligent Engineering and Systems*, Vol. 13, No. 3, pp.327–338, doi: 10.22266/ijies2020.0630.30.

Krishna, P.V. (2013) 'Honey bee behavior inspired load balancing of tasks in cloud computing environments', *Applied Soft Computing*, Vol. 13, No. 5, pp.2292–2303.

Laxmi Lydia, E., Kannan, S., Rajest, S.S. and Satyanarayana, S. (2020) 'Correlative study and analysis for hidden patterns in text analytics unstructured data using supervised and unsupervised learning techniques', *Int. J. Cloud Computing*, Vol. 9, Nos. 2–3, pp.150–162.

Lazar, A.J.P., Sengan, S., Cavaliere, L.P.L., Nadesan, T., Sharma, D., Gupta, M.K., Palaniswamy, T., Vellingiri, M., Sharma, D.K. and Subramani, T. (2021) 'Analysing the user actions and location for identifying online scam in internet banking on cloud', *Wireless Personal Communications*, pp.1–24.

Lee, L.T., Chen, C.W., Chang, H.Y., Tang, C.C. and Pan, K.C. (2009) 'A non-critical path earliest-finish algorithm for inter-dependent tasks in heterogeneous computing environments', *2009 11th IEEE International Conference on High Performance Computing and Communications,* June, IEEE, Korea, pp.603–608.

Lin, C., Lu, S., Fei, X., Chebotko, A., Pai, D., Lai, Z. and Hua, J. (2009) 'A reference architecture for scientific workflow management systems and the VIEW SOA solution', *IEEE Transactions on Services Computing*, Vol. 2, No. 1, pp.79–92.

Nandakumar, K., Vinod, V., Akbar Batcha, S.M., Sharma, D.K., Elangovan, M., Poonia, A., Mudlappa Basavaraju, S., Dogiwal, S.R., Dadheech, P. and Sengan, S. (2021) 'Securing data in transit using data-in-transit defender architecture for cloud communication', *Soft Computing*, Vol. 25, No. 18, pp.12343–12356.

Rahman, M., Hassan, R., Ranjan, R. and Buyya, R. (2013) 'Adaptive workflow scheduling for dynamic grid and cloud computing environment', *Concurrency and Computation: Practice and Experience*, Vol. 25, No. 13, pp.1816–1842.

Robine, J.M. and Michel, J.P. (2004) 'Looking forward to a general theory on population aging', *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, Vol. 59, No. 6, pp.M590–M597.

Shenai, S. (2012) 'Survey on scheduling issues in cloud computing', *Procedia Engineering*, Vol. 38, pp.2881–2888.

Subramani, R. and Vijayalakhsmi, C. (2016) 'Design of Lagrangian decomposition model for energy management using SCADA system', *Proc. of the 3rd International Symposium on Big Data and Cloud Computing Challenges. Smart Innovation, Systems and Technologies*, Vol. 49, pp.353–361.

Thakur, A. and Goraya, M.S. (2022) 'RAFL: a hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment', *Simulation Modelling Practice and Theory*, 102485.

VinayaKulkarni, S., Patil, S. and Patil, C.H. (2020) 'Study on network security algorithm', *International Journal of Engineering Research and Technology Conference Proceedings,* Bangalore, Vol. 8, No. 05, pp.1–3.

Xu, B., Zhao, C., Hu, E. and Hu, B. (2011) 'Job scheduling algorithm based on Berger model in cloud environment', *Advances in Engineering Software*, Vol. 42, No. 7, pp.419–425.

Xu, M., Cui, L., Wang, H. and Bi, Y. (2009) 'A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing', *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, August, IEEE, China, pp.629–634.

Xu, Y., Li, K., He, L. and Truong, T.K. (2013) 'A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization', *Journal of Parallel and Distributed Computing*, Vol. 73, No. 9, pp.1306–1322.