SRRA: A Novel Skewness-Based Algorithm for Cloudlet Scheduling

Sanjaya Kumar Panda* and Shidhanta Sen
Department of Computer Science and Engineering
National Institute of Technology Warangal, Warangal - 506004, Telangana, India
sanjaya@nitw.ac.in, ssmc20118@student.nitw.ac.in
*corresponding author

Abstract—Cloud computing enables developers to deploy and host applications without focusing on installing and maintaining the infrastructure. The developers can utilize the services provided by the cloud service providers (CSPs) to offer scalable solutions to customer applications. As a result, CSPs are deluged with different batches of cloudlets (tasks) from diverse customer applications. Therefore, developing an algorithm that selects and processes applications intelligently to minimize the execution time and maximize the throughput becomes challenging. Many researchers have shown the round-robin (RR) scheduling algorithm variants to tackle this problem. One such variant is the dynamic RR heuristic algorithm (DRRHA) that utilizes the mean of the burst times (BTs) of cloudlets in the ready queue (RQ) to calculate the time quantum (TQ). However, DRRHA has not considered skewness. This paper introduces a novel skewness-based RR algorithm (SRRA) for cloudlet scheduling. The algorithm dynamically determines the TQ for each cloudlet based on the skewness of the BTs of cloudlets in the RQ. The algorithm has two variants: SRRA with minimum TQ (SRRA-Min) and SRRA with median TQ (SRRA-Med). The two variants of the proposed algorithm exhibit improved performance in terms of total execution time (TET) and throughput compared to DRRHA, individually and collectively. These comparisons are conducted using CloudSim Plus under two scenarios: constant skewness with varying cloudlets and constant cloudlets with varying skewness.

Keywords–Cloud Computing; Cloud Service Provider; Cloudlet Scheduling; Round-Robin; Time Quantum; CloudSim Plus; Skewness.

1. Introduction

Cloud computing is renowned for providing a variety of services, including infrastructure, platform, and software, via the Internet [1]. It offers cost-effective, flexible, and on-demand solutions to support businesses of all sizes. Consequently, numerous organizations have sought services from various CSPs, such as Google, Amazon, Microsoft, Oracle, and International Business Machines (IBM), to develop on-demand applications without the burden of managing infrastructure and scalability concerns [2]. Cloud significantly streamlines the software development and deployment processes [3]. Therefore, it plays a pivotal role in the strategic framework of any organization.
Cloud computing leverages virtualization technology to provide services to multiple customers concurrently [4]. This technology efficiently utilizes the resources of physical machines, including storage, computation, and networking, among others [5]. Virtualization technology enhances resource utilization, reduces downtime and enables faster resource allocation, which leads to energy and cost savings. It's worth mentioning that virtualization can be implemented at various levels, such as operating system (OS) virtualization, server virtualization, and hardware virtualization [6]. These services are delivered to customers through the deployment of virtual machines (VMs), which emulate the functionalities of physical machines [7]. Physical machine resources are allocated to VMs according to their specific requirements. Each VM operates its own OS and functions independently.
Customers submit applications, referred to as cloudlets, for execution within VMs [8]. A cloudlet is typically defined by parameters like the number of instructions to be executed, the required processing elements, central processing unit (CPU), random access memory (RAM), and bandwidth utilization models [9]. Cloudlets are mapped to one or more VMs, and an appropriate VM is selected based on factors like processing power, bandwidth, and RAM. The challenging task is the development of an efficient cloudlet scheduling algorithm, which determines the order of a large set of cloudlets for execution in a distributed and scalable environment like cloud to achieve a predefined objective [10], [11].
The cloudlet scheduling algorithm can be broadly divided into two types: non-preemptive scheduling algorithm and preemptive scheduling algorithm [12]. The non-preemptive scheduling algorithm aims to complete the selected cloudlet without any interruption. On the contrary, the preemptive scheduling algorithm can suspend a cloudlet intermediary and start another cloudlet. The objectives of these scheduling algorithms are to minimize the overall execution time, overall cost and energy consumption of all the cloudlets and maximize the resource utilization and the throughput [13]. One of the well-known preemptive scheduling algorithms in CPU scheduling and cloud computing is the RR algorithm [14]–[24]. RR is widely used for its fairness, simplicity and ease of implementation. It assigns a specified execution time called TQ to each cloudlet. TQ allows a larger cloudlet to be completed in two or more rounds with preemption, whereas a smaller cloudlet can be completed in the first round. Nevertheless, selecting the appropriate TQ can be challenging to prevent starvation [25]. On the other hand, the RR algorithm can be implemented with static and dynamic TQ, called static RR and dynamic RR, respectively [24]. In static RR, TQ is fixed for each cloudlet. On the contrary, TQ is determined

individually for each cloudlet in every round of execution. Many researchers have presented different variants of the RR scheduling algorithm [14]–[24]. DRRHA [23], one of the recent algorithms, utilizes the mean BT of a batch of cloudlets in the RQ to determine the TQ. While DRRHA has demonstrated superior performance compared to other RR variants in terms of turnaround time (TAT), average waiting time (AWT) and response time (RT), it does not take into account the skewness of the BTs of cloudlets in the RQ. This phenomenon motivates us to think about skewness and develop a novel skewness-based algorithm for cloudlet scheduling.

In this paper, we address the problem of assigning n cloudlets with their resource requirements to a single VM with resource constraints in an RR fashion. For this, we develop SRRA for cloudlet scheduling and present its variants, SRRA-Min and SRRA-Med. Both variants dynamically determine the TQ for each cloudlet based on the skewness of the BTs of cloudlets in the RQ. We consider all types of skewness, namely zero, positive and negative, to evaluate the proposed variants over the DRRHA algorithm regarding TET and throughput. We compare the proposed variants individually and collectively with DRRHA using CloudSim Plus in two scenarios: one with a constant skewness while varying the number of cloudlets and the other with a constant number of cloudlets while varying the skewness to show their efficacy. The primary research contributions are summarized as follows.

1) We develop a novel algorithm, called SRRA, for dynamically calculating TQ based on different types of skewness for cloudlet scheduling.
2) We present two variants of the proposed algorithm, called SRRA-Min and SRRA-Med, to illustrate the influence of negatively skewed BTs of the cloudlets within the RQ.
3) We compare the proposed algorithm with one of the recent existing algorithms, DRRHA, using two performance metrics and two scenarios.

The remaining part of this paper is structured as follows. Section 2 provides an overview of the existing variants of the RR scheduling algorithm for cloudlet scheduling. Section 3 introduces the CloudSim Plus environment in the model and presents the problem statement. Section 4 introduces the proposed algorithm, SRRA, and its two variants, SRRA-Min and SRRA-Med, for cloudlet scheduling. Section 5 describes the two performance metrics to evaluate the proposed and existing algorithms, simulation environment and simulation results in three cases and two scenarios of each case. We conclude this paper in Section 6 by summarizing our findings and discussing future research directions.

2. Related Work

This section presents the recent studies on the well-known RR scheduling algorithm and its determination of TQ [14]–[24]. Wang et al. [26] have used the Hadoop platform to implement the weighted RR scheduling algorithm. The incoming cloudlets are reached to the master VM controller, which subsequently assigns these cloudlets to the slave VM based on the scheduling algorithm. However, they have not used the CloudSim Plus tool. Khurma et al. [18] have proposed the modified RR (MRR) algorithm. The TQ in MRR is calculated based on the mean of the BTs of all the cloudlets in the RQ. However, the first cloudlet is executed till completion. Their results show that the MRR has improved the AWT. Banerjee et al. [19] have improved the performance of the traditional RR algorithm (RRA) and improved RR cloudlet scheduling algorithm (IRRCSA). They have considered TQ for each VM by looking into the number of cloudlets allocated to that VM. More specifically, TQ is calculated based on the median of the average BT of the cloudlets and the maximum BT in the RQ. Note that each VM has its local RQ. Finally, they have shown the results regarding TAT, AWT, resource utilization and context switches. However, they have not followed any BT distribution of the batch of cloudlets in RQ.

Balharith and Alhaidari [20] have performed a review on the RR algorithms developed in CPU scheduling and cloud computing environments. They have categorized the review into RR based on static TQ and RR based on dynamic TQ. They have further categorized the RR based on dynamic TQ to dynamic TQ for each round and dynamic TQ for each process. Some of the RR variants reported are shortest remaining burst RR (SRBRR), amended dynamic RR (ADRR), time slice priority-based RR (TSPBRR), IRRCSA, MRR, priority RR and modified RR scheme with vigorous TQ. They have also described each RR variant's scheduling parameters, optimization criteria, environment (CPU, cloud or both), and tools. Jbara [21] has proposed a dynamic RR algorithm named the eighty-five percentile RR (EFPRR). Here, cloudlets are sorted in the non-decreasing order of BT. Then, TQ is calculated by multiplying the cloudlets' average BT by 0.85 (i.e., 85%). If BT of the cloudlet is less than the TQ, then the cloudlet is executed until completion. Otherwise, the cloudlet is placed at the end of the RQ. The algorithm shows improved average execution time over other algorithms.

Sanaj and Prathap [22] have presented an improved version of the RR Algorithm (ERR), intending to enhance its performance while preserving the favourable qualities of the traditional RR algorithm intact. The TQ is calculated dynamically for each cloudlet in the RQ. ERR uses the mean of the BTs of all cloudlets in the RQ to calculate TQ. In the subsequent rounds, the cloudlet is executed for the specified TQ and is either shifted to the end of the RQ or completed its execution. However, upon arrival of the first cloudlet, it is assigned the VM and executed till completion. They have implemented and tested their algorithm on the CloudSim toolkit. The results show that the ERR algorithm's AWT is less than the conventional RR algorithm. However, they have not examined the left or right-skewed batch of cloudlets.

Alhaidari and Balharith [23] have proposed a technique called the DRRHA in the cloud environment. DRRHA determines dynamic TQ for the cloudlet based on the current mean of the BTs of cloudlets in the RQ and continues the cloudlet based on the remaining BT. Note that the TQ is determined for each cloudlet dynamically. They have used CloudSim Plus to show the DRRHA's efficacy over other RR variants, such

TABLE I
A SUMMARY OF RELATED WORK

Article	Algorithm	TQ	Parameters	Tool
Khurma et al. (2018) [18]	MRR	Static	AWT	CloudSim
Banerjee et al. (2018) [19]	RRA and IRRCSA	Dynamic	TAT, AWT and Context Switches	CloudSim
Jbara (2019) [21]	EFPRR	Dynamic	Execution Time	CloudSim
Sanaj and Prathap (2020) [22]	ERR	Dynamic	AWT and Execution Time	CloudSim
Alhaidari and Balharith (2021) [23]	DRRHA	Dynamic	TAT, AWT and RT	CloudSim Plus

as dynamic time slice RR, improved RR algorithm, improved RR CPU scheduling algorithm with varying TQ (IRRVQ) and shortest job first and RR with dynamic variable TQ (SRDQ) in terms of the TAT, AWT and RT. However, they have not considered the skewness of the BTs of the batch of cloudlets in the RQ into consideration. A summary of related work is shown in Table I.

This paper presents a novel algorithm, SRRA, which is different from the other algorithms [17]–[24] (especially, DRRHA [23]) as follows.

1) The proposed algorithm considers different skewnesses, namely zero, positive and negative, to calculate the dynamic TQ, which is not used in [17]–[24]. Specifically, DRRHA only uses the mean for calculating the TQ of the cloudlet in the RQ.
2) The proposed algorithm uses different dynamic TQs based on the type of skewness in the RQ. On the contrary, DRRHA uses only one dynamic TQ without considering the type of skewness.
3) The datacenter broker simple class of CloudSim Plus uses the RR policy to execute the cloudlets without considering the cloudlet, user and provider requirements [9]. However, the proposed algorithm incorporates such requirements.

3. Model and Problem Statement

This section introduces the model employed in this paper and outlines the specific problem addressed within this research's scope.

3.1 Model

We consider the CloudSim Plus environment for cloudlet scheduling. This environment enables the user to create on-demand cloudlets and VMs. CloudSim Plus provides the features to postpone the creation of cloudlets, allowing us to create cloudlets in the system at different time instances. The cloudlets are submitted to the datacenter broker, which takes the appropriate actions in response to their requirements. Mostly, the datacenter broker allocates the cloudlets inside the VM. The datacenter broker simple class follows the RR policy to select the VM to execute the cloudlets. However, this class does not consider the requirements of the cloudlet, the user and the provider. Therefore, this paper introduces a novel policy to deal with such requirements.

3.2 Problem Statement

Consider a set of n cloudlets, $C = \{C_1, C_2, \ldots, C_n\}$, that are waiting in the RQ for execution as per their arrival to the system. Each cloudlet C_i, $1 \leq i \leq n$, defines a set of resource requirements, namely RAM, bandwidth, CPU or BT, storage and many more. These resource requirements are generated in a random manner using a pseudo-random number generator. Moreover, we use the Gaussian distribution as a statistical distribution to generate numbers. However, other distributions, such as exponential, Pareto and uniform, can also be used for the same. The problem is to assign the n cloudlets with their resource requirements to a single VM with resource constraints in an RR fashion, such that TET is minimized and throughput is maximized. Many researchers have developed variants of RR by introducing the selection of TQ to minimize the context switches. This paper also attempts to determine an adaptable value for dynamic TQ by monitoring the BTs of the cloudlets within RQ, aiming to achieve the stated objectives.

4. Proposed Algorithm

This section introduces the proposed algorithm, SRRA, and its two variants, SRRA-Min and SRRA-Med, for cloudlet scheduling. The objective of the SRRA is to minimize the TET and maximize the throughput. The pseudocodes of the two variants of SRRA are shown in Algorithm 1 and Algorithm 2, respectively. The primary motivation behind the SRRA-Min and SRRA-Med is based on Pearson's skewness coefficient (p). Mathematically, p of the BTs of the current batch of cloudlets in RQ is calculated as follows (Line 4 of Algorithm 1 and Algorithm 2) [27], [28].

$$p = \frac{3 \times (\mu - Med)}{\sigma} \tag{1}$$

where

$$\mu = \frac{1}{n}\sum_{i=1}^{n} BT_i, |RQ| = n \tag{2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(BT_i - \mu)^2}{n}} \tag{3}$$

where μ is the mean, Med is the median and σ is the standard deviation. Note that we use Gaussian distribution to generate the BTs.

Skewness can be zero, left or right based on the BTs distribution of the current batch of cloudlets in the RQ. It is equal to zero if the Med of BTs distribution is equal to the μ. On the contrary, it is towards the left if the Med of BTs distribution is greater than the μ. Similarly, it is towards the right if the Med of BTs distribution is less than the μ. On the other hand, if p lies between $[-\tau, \tau]$, skewness is set to zero. The skewness is positive if $p > \tau$. Alternatively, the BTs of the batch of cloudlets in the RQ are positively skewed. The skewness is negative if $p < \tau$. Alternatively, the BTs of the batch of cloudlets in the RQ are negatively skewed. Mathematically,

$$\text{Skewness} = \begin{cases} \text{Zero} & \text{if } p \in [-\tau, \tau] \\ \text{Positive} & \text{if } p > \tau \\ \text{Negative} & \text{if } p < -\tau \end{cases} \quad (4)$$

Selecting an appropriate value for τ is challenging and varies concerning different applications. However, it is taken as 0.4 based on [29]. Based on the skewness, we select and assign dynamic TQ for the execution of the cloudlets. The two variants of SRRA are shown in the following subsections.

4.1 SRRA-Min

SRRA-Min removes i^{th} cloudlet from the RQ and determines the dynamic TQ based on the calculated values, namely p, maximum BT (i.e., $BTmax$), minimum BT (i.e., $BTmin$), mean BT (i.e., $BTmean$) and the n^{th} root of the product (i.e., $prod$) of cloudlets in the RQ (Line 6 to Line 11 of Algorithm 1). Note that if only one cloudlet exists in the RQ, execute it until completion. If skewness is zero or positively skewed, the dynamic TQ of i^{th} cloudlet (i.e., TQ_i) is calculated as follows (Line 7 and Line 8).

$$TQ_i = \frac{\mu}{2} + \frac{\mu}{2 \times BT_i} \quad (5)$$

where BT_i is the burst time of i^{th} cloudlet. Note that this calculation is similar to the DRRHA. It is noteworthy to mention that the above TQ is applicable for $p \geq \tau$. If skewness is negatively skewed (Line 9), the TQ_i is determined as the minimum of the TQ calculated using the $difference$ and $nthroot$ procedures (i.e., Procedure 1 and Procedure 2) in line 10. Mathematically,

$$TQ_i = min(difference(), nthroot()) \quad (6)$$

In Procedure 1, the maximum, minimum, and mean BT are calculated (Line 4, Line 5 and Line 8). Then, the TQ is calculated based on the maximum, minimum, and mean BT (Line 9). In Procedure 2, the TQ is calculated based on the n^{th} root of $prod$ (Line 6). Note that $prod$ is determined by multiplying all the BTs (Line 3 to Line 5).

Once the TQ of i^{th} cloudlet is determined, the i^{th} cloudlet starts its execution and continues till the TQ is over (Line 12). Then, the remaining BT is updated (Line 13). If the remaining BT is less than the TQ, then the i^{th} cloudlet continues its execution (Line 14 and Line 15). Once the i^{th} cloudlet completes its execution, it is moved to the finished queue (FQ) (Line 15). Otherwise, the i^{th} cloudlet is moved to the end of RQ (Line 16 to Line 18). On the other hand, if the new cloudlet(s) is/are arrived, then the cloudlet(s) is/are added to the RQ (Line 20 to Line 22). The above process is iterated until the RQ is empty (Line 1 to Line 23).

Algorithm 1 SRRA-Min

```
Input(s): An $RQ$ with $n$ cloudlets and their properties
Output(s): A list of finished cloudlets
1: while $RQ$ is not empty do
2:     Sort the cloudlets in the non-decreasing order of their $BT$
3:     Calculate $\mu$, $\sigma$ and $Med$ of the sorted cloudlets
4:     Set $p = \frac{3\times(\mu - Med)}{\sigma}$
5:     for each cloudlet $C_i$ in the $RQ$ do
6:         Remove cloudlet $C_i$
7:         if $p \geq -\tau$ then
8:             Set $TQ_i = \frac{\mu}{2} + \frac{\mu}{2\times BT_i}$
9:         else
10:            Set $TQ_i = min(difference(RQ, BT), nthroot(RQ, BT))$
11:        end if
12:        Execute the cloudlet $C_i$
13:        Set $BT_i$ -= $TQ_i$
14:        if $BT_i < TQ_i$ then
15:            Continue the execution of the cloudlet $C_i$ and/or move the cloudlet $C_i$ to the $FQ$
16:        else
17:            Move the cloudlet $C_i$ to the end of $RQ$
18:        end if
19:    end for
20:    if new cloudlet(s) is/are arrived then
21:        Add the cloudlet(s) to the end of $RQ$
22:    end if
23: end while
```

Procedure 1 $difference(RQ, BT)$

```
Input(s): $RQ$, $BT$
Output(s): $TQ$
1: Set $BTmax = -\infty$, $BTmin = \infty$ and $BTmean = 0$
2: Set $n$ = $RQ.size()$
3: for each cloudlet $C_i$ in the $RQ$ do
4:     Set $BTmax$ = $max(BTmax, BT_i)$
5:     Set $BTmin$ = $min(BTmin, BT_i)$
6:     Set $BTmean$ += $BT_i$
7: end for
8: Set $BTmean = \frac{BTmean}{n}$
9: Set $TQ = \frac{BTmax - BTmin + 2\times BTmean}{2}$
10: return
```

4.2 SRRA-Med

SRRA-Med removes i^{th} cloudlet from the RQ and determines the dynamic TQ based on the μ and Med values of cloudlets in the RQ (Line 6 to Line 11 of Algorithm 2). Here, τ is varied on the basis of the p value. If skewness is zero or positively skewed, the dynamic TQ of i^{th} cloudlet (i.e., TQ_i) is calculated as per SRRA-Min (Line 7 and Line 8). It is noteworthy to mention that the above TQ is applicable for $p \geq \tau$. If skewness is negatively skewed (Line 9), the TQ_i is determined based on the Med (Line 10). Mathematically,

$$TQ_i = \frac{Med}{2} + \frac{Med}{2 \times BT_i} \quad (7)$$

Once the TQ of i^{th} cloudlet is determined, the i^{th} cloudlet starts its execution and continues till the TQ is over (Line 12). Then, the remaining BT is updated (Line 13). If the remaining BT is less than the TQ, then the i^{th} cloudlet

Procedure 2 $nthroot(RQ, BT)$

```
    Input(s): $RQ$, $BT$
    Output(s): $TQ$
1: Set $prod = 1$
2: Set $n = RQ.size()$
3: for each cloudlet $C_i$ in the $RQ$ do
4:     Set $prod \times= BT_i$
5: end for
6: Set $TQ = \sqrt[n]{prod}$
7: return
```

Algorithm 2 SRRA-Med

```
    Input(s): An $RQ$ with $n$ cloudlets and their properties
    Output(s): A list of finished cloudlets
1: while $RQ$ is not empty do
2:     Sort the cloudlets in the non-decreasing order of their $BT$
3:     Calculate $\mu$, $\sigma$ and $Med$ of the sorted cloudlets
4:     Set $p = \frac{3\times(\mu - Med)}{\sigma}$
5:     for each cloudlet $C_i$ in the $RQ$ do
6:         Remove cloudlet $C_i$
7:         if $p \geq -\tau$ then
8:             Set $TQ_i = \frac{\mu}{2} + \frac{\mu}{2\times BT_i}$
9:         else
10:            Set $TQ_i = \frac{Med}{2} + \frac{Med}{2\times BT_i}$
11:        end if
12:        Execute the cloudlet $C_i$
13:        Set $BT_i$ -= $TQ_i$
14:        if $BT_i < TQ_i$ then
15:            Continue the execution of the cloudlet $C_i$ and/or move the cloudlet $C_i$ to the $FQ$
16:        else
17:            Move the cloudlet $C_i$ to the end of $RQ$
18:        end if
19:    end for
20:    if new cloudlet(s) is/are arrived then
21:        Add the cloudlet(s) to the end of $RQ$
22:    end if
23: end while
```

continues its execution (Line 14 and Line 15). Once the i^{th} cloudlet completes its execution, it is moved to the FQ (Line 15). Otherwise, the i^{th} cloudlet is moved to the end of RQ (Line 16 to Line 18). On the other hand, if the new cloudlet(s) is/are arrived, then the cloudlet(s) is/are added to the RQ (Line 20 to Line 22). The above process is iterated until the RQ is empty (Line 1 to Line 23). Note that SRRA-Med is similar to SRRA-Min except the line 10.

5. Performance Metrics, Simulation Environment and Results

This section describes the performance metrics, simulation environment and results.

5.1 Performance Metrics

We consider two performance metrics, namely TET and throughput, for measuring the performance of the proposed and existing algorithms. TET is the total time required for a batch of cloudlets to run until completion. It also includes the time taken for context switching between two cloudlets. Note that the context switch time (CST) is the time required to move a cloudlet from primary memory to secondary memory and load another cloudlet from secondary memory to primary memory. However, it is calculated as the ratio of the cloudlet size and the corresponding VM's bandwidth in the simulation results. For instance, let us assume that the cloudlet size is 300 bytes (i.e., 2400 bits) and the VM bandwidth is 50000 Mbps. Therefore, the CST is calculated as $\frac{2400}{50000}$ = 0.048 seconds and is fixed for all the cloudlets. Net execution time (NET) is the total time taken for all the cloudlets to finish execution, assuming the overhead of context switching to be negligible (i.e., zero). Therefore, TET is mathematically expressed as follows.

$$TET = NET + 0.048 \times |CS| \tag{8}$$

where $|CS|$ is the total number of context switches carried out during the execution of a batch of cloudlets.

Throughput is the number of cloudlets completed in a fixed time interval (say, δ). It is mathematically expressed as follows.

$$\text{Throughput} = \frac{n}{\delta} \tag{9}$$

where n is the number of cloudlets completed within the time interval of δ.

5.2 Simulation Environment

TABLE II
DATACENTER CHARACTERISTICS

Parameter	Value
OS	Linux
Virtual Machine Monitor (VMM)	Xen
Architecture	x86
Host	1

The two variants of the proposed algorithm, SRRA-Min and SRRA-Med, and the existing algorithm, DRRHA, were tested using CloudSim Plus [9], a Java 17-based cloudlet simulation framework. Note that CloudSim Plus can be described as an extended version of CloudSim 3. The simulation environment was set up in a new program in which the modified CloudSim Plus library was imported as an external Java archive file. The datacenter, host, virtual machines and cloudlet with specific characteristics were configured in the simulation environment, as detailed in Table II to Table V, respectively. These configurations were kept constant for all the algorithms. The proposed and existing algorithms were implemented by modifying and overwriting the CloudletScheduler parent class and creating three new classes.

The BT of cloudlets was determined through random sampling from a normal distribution, with parameters such as μ, σ, and skewness. We varied the number of cloudlets and skewness, one at a time, while keeping all other parameters of the cloudlets constant, such as input size, output size, CPU requirements and others. It ensured that our simulations were independent of these parameters' influence and simplified the calculations of performance metrics. For instance, we

TABLE III
HOST CHARACTERISTICS

Parameter	Value
Size	1000000 MB
Processing Power	100 MIPS
Bandwidth	100000 Mbps
Processing Elements (PEs)	2
RAM	4096 MB

TABLE IV
VM CHARACTERISTICS

Parameter	Value
Image Size	10000 MB
Processing Power	100 MIPS
Bandwidth	50000 Mbps
PEs	2
RAM	1024 MB

increased the batch size of cloudlets from 50 to 180 while maintaining a constant skewness of -15 during the simulation of SRRA-Min. On the other hand, we varied the skewness from -20 to -40 while keeping the batch size of cloudlets constant at 200. In both cases and their variations, we recorded the TET, NET and throughput. On the contrary, we increased the batch size of cloudlets from 50 to 200 while maintaining a constant skewness of -15 during the simulation of SRRA-Med. On the other hand, we varied the skewness from -20 to -40 while keeping the batch size of cloudlets constant at 150. Like SRRA-Min, in both cases and their variations, we recorded the TET, NET and throughput.

5.3 Simulation Results

We considered three cases in the simulation results: (1) SRRA-Min versus DRRHA, (2) SRRA-Med versus DRRHA and (3) SRRA-Min versus SRRA-Med versus DRRHA. In each case, we showed the comparison results in terms of TET and throughput. SRRA-Min and SRRA-Med improved TET while maintaining similar or improved throughput for all the negatively skewed BTs of cloudlets in the RQ. By defining TQ based on skewness, we effectively captured the effectiveness of the DRRHA algorithm that prioritizes the mean, while enhancing TQ for negatively skewed batches by employing a different TQ calculation. This enhancement is primarily evident in the significant reduction in the total number of context switches. Subsequently, it was substantial enough to affect the TET. It's important to highlight that this performance improvement was achieved without any additional loss in throughput.

5.3.1 SRRA-Min versus DRRHA

We considered two scenarios: constant skewness with varying cloudlets and constant cloudlets with varying skewness. In scenario 1, the comparison results of SRRA-Min and DRRHA are shown in terms of TET in Table VI and throughput in

TABLE V
CLOUDLET CHARACTERISTICS

Parameter	Value
File Size	300 bytes
Output Size	300 bytes
Required PEs	1
PE Utilization	100%
Bandwidth Utilization	100%
RAM Utilization	50%

TABLE VI
SRRA-MIN VERSUS DRRHA IN TERMS OF TET WITH VARYING CLOUDLETS AND SKEWNESS = -15

	NET (in Seconds)		**CST (in Seconds)**		**TET (in Seconds)**	
Number of Cloudlets	**SRRA-Min**	**DRRHA**	**SRRA-Min**	**DRRHA**	**SRRA-Min**	**DRRHA**
050	**0478.00**	0528.31	**012.67**	032.88	**0490.67**	0561.18
100	**0947.57**	0952.19	**072.00**	095.04	**1019.57**	1047.23
150	**1361.41**	1374.04	113.90	**108.57**	**1475.31**	1482.62
180	1734.38	**1725.52**	**055.39**	113.28	**1789.77**	1838.80

Table VII. Note that bold indicates the best-performing value in both tables. In Table VI, we increased the cloudlets from 50 to 180 by keeping -15 as the constant skewness. In Table VII, we observed the throughput in 100 seconds by increasing the cloudlets from 50 to 180 and keeping -15 as the constant skewness. As seen from Table VI, SRRA-Min performs better in reducing TET (in seconds) than the DRRHA, irrespective of the number of cloudlets in a single batch. It is also shown in Figure 1 for easy comparison. It can be observed that in most cases, SRRA-Min performs better in reducing NET (in seconds) and CST (in seconds) than the DRRHA. On the other hand, in throughput, SRRA-Min outperforms DRRHA for smaller batches (i.e., 50 and 100 cloudlets) and gives similar performance for larger batches (i.e., 150 and 180 cloudlets).

In scenario 2, the comparison results of SRRA-Min and DRRHA are shown in terms of TET in Table VIII and throughput

TABLE VII
SRRA-MIN VERSUS DRRHA IN TERMS OF THROUGHPUT WITH VARYING CLOUDLETS AND SKEWNESS = -15

Number of Cloudlets	SRRA-Min	DRRHA
050	**0.20**	0.19
100	**0.20**	0.18
150	**0.20**	**0.20**
180	**0.20**	**0.20**

TABLE VIII
SRRA-MIN VERSUS DRRHA IN TERMS OF TET WITH VARYING SKEWNESS AND CLOUDLETS = 200

	NET (in Seconds)		**CST (in Seconds)**		**TET (in Seconds)**	
Skewness	**SRRA-Min**	**DRRHA**	**SRRA-Min**	**DRRHA**	**SRRA-Min**	**DRRHA**
-20	**1836.29**	1883.08	122.49	**115.72**	**1958.79**	1998.81
-25	**1846.19**	1862.94	118.22	**115.92**	**1964.42**	1978.86
-30	1944.98	**1891.14**	**037.87**	116.40	**1982.85**	2007.54
-40	1890.41	**1865.67**	**075.45**	120.09	**1965.86**	1985.76

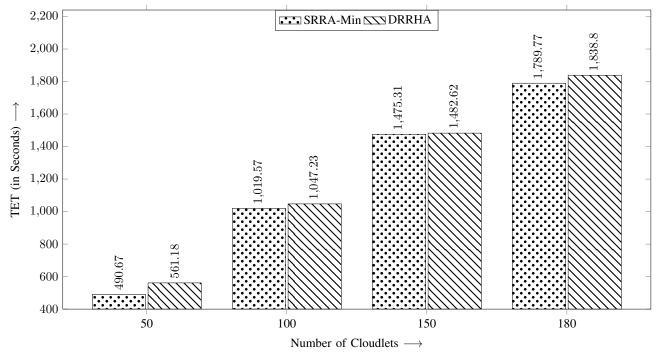

Figure 1. Graphical comparison of SRRA-Min versus DRRHA in terms of TET with varying cloudlets and skewness = -15

TABLE IX
SRRA-MIN VERSUS DRRHA IN TERMS OF THROUGHPUT WITH VARYING SKEWNESS AND CLOUDLETS = 200

Skewness	SRRA-Min	DRRHA
-20	**0.19**	**0.19**
-25	**0.19**	**0.19**
-30	**0.19**	**0.19**
-40	**0.19**	**0.19**

in Table IX. In Table VIII, we increased the skewness from -20 to -40 by keeping 200 as the constant cloudlets. In Table IX, we observed the throughput in 800 seconds by increasing the skewness from -20 to -40 and keeping 200 as the constant skewness. As seen from Table VIII, SRRA-Min performs better in reducing TET than the DRRHA, irrespective of the skewness in a single batch. It is also shown in Figure 2 for easy comparison. It can be observed that in two out of four cases, SRRA-Min performs better in reducing NET and CST than the DRRHA. On the other hand, in throughput, SRRA-Min gives a similar performance to DRRHA.

The rationality behind the better performance of SRRA-Min is that the TQ of SRRA-Min is larger compared to the TQ of DRRHA for negatively skewed BTs of cloudlets in the RQ. Hence, each cloudlet executes for extended periods, leading to improved TET, NET and throughput. Nevertheless, as the number of cloudlets increases, NET performance worsens, but CST decreases, ultimately leading to an improvement in TET.

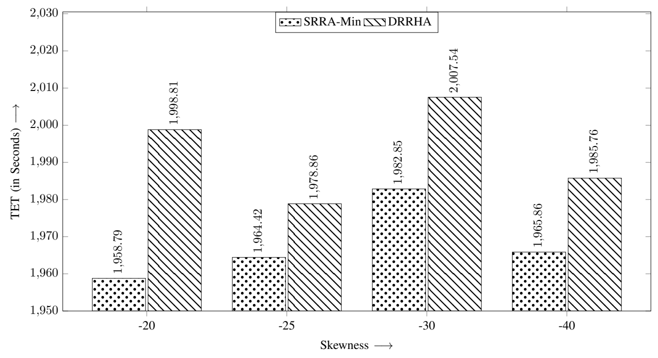

Figure 2. Graphical comparison of SRRA-Min versus DRRHA in terms of TET with varying skewness and cloudlets = 200

TABLE X
SRRA-MED VERSUS DRRHA IN TERMS OF TET WITH VARYING CLOUDLETS AND SKEWNESS = -15

	NET (in Seconds)		CST (in Seconds)		TET (in Seconds)	
Number of Cloudlets	SRRA-Min	DRRHA	SRRA-Min	DRRHA	SRRA-Min	DRRHA
050	0488.97	**0486.42**	**02.30**	34.65	**0491.27**	0521.08
100	**0989.73**	1001.68	**13.96**	95.95	**1003.70**	1097.64
150	**1552.84**	1580.88	**07.58**	91.96	**1560.42**	1672.85
200	2094.94	**2024.59**	**09.74**	93.84	**2104.68**	2118.43

5.3.2 SRRA-Med versus DRRHA

Like SRRA-Min versus DRRHA, we considered two scenarios: constant skewness with varying cloudlets and constant cloudlets with varying skewness. In scenario 1, the comparison results of SRRA-Med and DRRHA are shown in terms of TET in Table X and throughput in Table XI. Note that bold indicates the best-performing value in both tables. In Table X, we increased the cloudlets from 50 to 200 by keeping -15 as the constant skewness. In Table XI, we observed the throughput in 100 seconds by increasing the cloudlets from 50 to 200 and keeping -15 as the constant skewness. As seen from Table X, SRRA-Med performs better in reducing TET (in seconds) than the DRRHA, irrespective of the number of cloudlets in a single batch. It is also shown in Figure 2 for easy comparison. It can be observed that SRRA-Med performs better in reducing NET (in seconds) in two out of four cases and CST (in seconds) in all the cases than the DRRHA. On the other hand, in throughput, SRRA-Med outperforms or performs similarly to DRRHA.

In scenario 2, the comparison results of SRRA-Med and DRRHA are shown in terms of TET in Table XII and throughput in Table XIII. In Table XII, we increased the skewness from -20 to -40 by keeping 150 as the constant cloudlets. In Table XIII, we observed the throughput in 800 seconds by increasing the skewness from -20 to -40 and keeping 150 as the constant skewness. As seen from Table XII, SRRA-Med performs better in reducing TET (in seconds) than the DRRHA, irrespective of the skewness in a single batch. It is also shown in Figure 4 for easy comparison. It can be observed that SRRA-Med performs better in reducing NET (in seconds) in two out of four cases and CST (in seconds) in all the cases than the DRRHA. On the other hand, in throughput, SRRA-Med outperforms DRRHA. Note that greater skewness resulted in an increase in throughput.

A larger disparity between the mean and median contributes to enhanced performance for SRRA-Med. This can be attributed to the higher associated TQ for each cloudlet. As a result, TET, NET and throughput were improved. However, in certain cases, the performance of NET may degrade, but CST decreases, resulting in an overall improvement in TET.

TABLE XI
SRRA-MED VERSUS DRRHA IN TERMS OF THROUGHPUT WITH VARYING CLOUDLETS AND SKEWNESS = -15

Number of Cloudlets	SRRA-Min	DRRHA
050	**0.060**	**0.060**
100	**0.123**	**0.123**
150	**0.186**	0.182
200	**0.192**	**0.192**

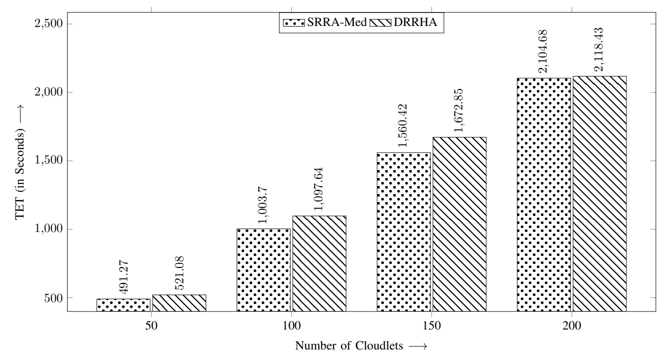

Figure 3. Graphical comparison of SRRA-Med versus DRRHA in terms of TET with varying cloudlets and skewness = -15

5.3.3 SRRA-Min versus SRRA-Med versus DRRHA

Like previous cases, we considered two scenarios: constant skewness with varying cloudlets and constant cloudlets with varying skewness. In scenario 1, the comparison results of SRRA-Min, SRRA-Med and DRRHA are shown in terms of TET in Table XIV and throughput in Table XV. Note that bold indicates the best-performing value in both tables. In Table XIV, we increased the cloudlets from 50 to 200 by keeping -15 as the constant skewness. In Table XV, we observed the throughput in 100 seconds by increasing the cloudlets from 50 to 200 and keeping -15 as the constant skewness. As seen from Table XIV, SRRA-Med performs better in reducing TET in three out of four cases than the SRRA-Min and DRRHA, irrespective of the number of cloudlets in a single batch. It is also shown in Figure 5 for easy comparison. In throughput, SRRA-Min outperforms SRRA-Med and DRRHA in three out of four cases.

In scenario 2, the comparison results of SRRA-Min, SRRA-Med and DRRHA are shown in terms of TET in Table XVI and throughput in Table XVII. In Table XVI, we increased the skewness from -20 to -40 by keeping 150 as the constant cloudlets. In Table XVII, we observed the throughput in 800 seconds by increasing the skewness from -20 to -40 and keeping 150 as the constant skewness. As seen from Table XVII, SRRA-Med performs better in reducing TET in three out of four cases than the SRRA-Min and DRRHA, irrespective of

TABLE XII
SRRA-MED VERSUS DRRHA IN TERMS OF TET WITH VARYING SKEWNESS AND CLOUDLETS = 150

	NET (in Seconds)		CST (in Seconds)		TET (in Seconds)	
Skewness	SRRA-Med	DRRHA	SRRA-Med	DRRHA	SRRA-Med	DRRHA
-20	**1486.82**	1505.55	**012.86**	090.52	**1499.68**	1596.08
-25	1504.72	**1416.63**	**007.20**	114.09	**1511.92**	1530.73
-30	**1482.79**	1499.31	**009.55**	088.94	**1492.34**	1588.26
-40	1518.72	**1443.30**	**007.20**	111.79	**1525.92**	1555.09

TABLE XIII
SRRA-MED VERSUS DRRHA IN TERMS OF THROUGHPUT WITH VARYING SKEWNESS AND CLOUDLETS = 150

Skewness	SRRA-Med	DRRHA
-20	**0.183**	0.180
-25	**0.186**	0.176
-30	**0.185**	0.180
-40	**0.187**	0.180

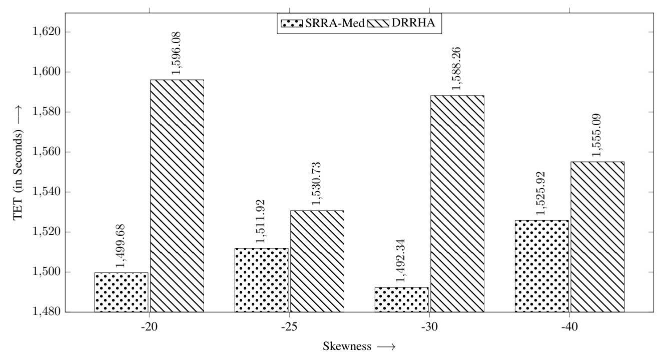

Figure 4. Graphical comparison of SRRA-Med versus DRRHA in terms of TET with varying skewness and cloudlets = 150

TABLE XIV
SRRA-MIN VERSUS SRRA-MED VERSUS DRRHA IN TERMS OF TET WITH VARYING CLOUDLETS AND SKEWNESS = -15

	NET (in Seconds)			CST (in Seconds)			TET (in Seconds)		
Number of Cloudlets	SRRA-Min	SRRA-Med	DRRHA	SRRA-Min	SRRA-Med	DRRHA	SRRA-Min	SRRA-Med	DRRHA
050	0518.49	**0505.97**	0511.68	11.42	**02.30**	035.80	0529.91	**0508.27**	0547.48
100	1034.03	1021.45	**1005.40**	48.48	**10.89**	095.37	1082.51	**1032.35**	1100.78
150	1517.98	1558.87	**1504.25**	77.18	**07.29**	109.20	1595.16	**1566.16**	1613.45
200	**2030.65**	2035.61	2053.17	54.00	**53.52**	107.23	**2084.65**	2089.13	2160.40

TABLE XV
SRRA-MIN VERSUS SRRA-MED VERSUS DRRHA IN TERMS OF THROUGHPUT WITH VARYING CLOUDLETS AND SKEWNESS = -15

Number of Cloudlets	SRRA-Min	SRRA-Med	DRRHA
050	**0.061**	0.060	0.060
100	**0.125**	0.121	0.110
150	0.176	**0.181**	0.171
200	**0.196**	0.192	0.188

TABLE XVI
SRRA-MIN VERSUS SRRA-MED VERSUS DRRHA IN TERMS OF TET WITH VARYING SKEWNESS AND CLOUDLETS = 150

	NET (in Seconds)			CST (in Seconds)			TET (in Seconds)		
Skewness	SRRA-Min	SRRA-Med	DRRHA	SRRA-Min	SRRA-Med	DRRHA	SRRA-Min	SRRA-Med	DRRHA
-20	1521.70	1541.26	**1501.80**	83.71	**17.61**	112.32	1605.41	**1558.87**	1614.12
-25	1520.99	**1503.06**	1506.30	71.76	**07.24**	104.78	1592.75	**1510.31**	1611.08
-30	1515.10	1554.11	**1505.84**	55.15	**17.90**	110.35	**1570.25**	1572.01	1616.19
-40	1536.30	1508.91	**1506.44**	56.16	**07.10**	109.39	1592.46	**1516.02**	1615.83

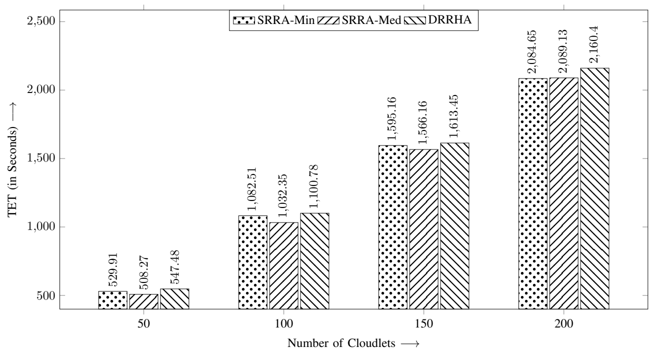

Figure 5. Graphical comparison of SRRA-Min versus SRRA-Med versus DRRHA in terms of TET with varying cloudlets and skewness = -15

TABLE XVII
SRRA-MIN VERSUS SRRA-MED VERSUS DRRHA IN TERMS OF THROUGHPUT WITH VARYING SKEWNESS AND CLOUDLETS = 150

Skewness	SRRA-Min	SRRA-Med	DRRHA
-20	0.180	**0.185**	0.178
-25	0.182	**0.183**	0.175
-30	0.185	**0.186**	0.185
-40	**0.187**	0.185	0.180

the skewness in a single batch. It is also shown in Figure 6 for easy comparison. On the other hand, in throughput, SRRA-Med outperforms SRRA-Min and DRRHA.
The reason behind the better performance of SRRA-Med is stated as follows. The DRRHA algorithm only utilizes the mean in its calculation of TQ, implying that TQ for a left-skewed RQ would be smaller. However, we specify different formulations of TQ for left and right-skewed distributions, which results in more significance for running cloudlets.

6. Conclusion and Future Work

We have presented a novel skewness-based algorithm for cloudlet scheduling. The algorithm is based on the batch characteristics of the cloudlets in the RQ. The algorithm has been shown in two distinct variations: SRRA-Min and SRRA-Med. We have demonstrated the proposed algorithm's efficacy by comparing its performance in terms of TET and throughput with the DRRHA algorithm that uses the mean of the RQ to calculate TQ. Moreover, the proposed and existing algorithms have been compared by taking constant skewness with varying cloudlets and constant cloudlets with varying skewness. We found that both the variants of the proposed algorithm show improved TET and throughput compared to the DRRHA for negatively skewed BTs of cloudlets in the RQ. Moreover, the proposed variants reduce the context switches due to the dynamic TQ, improving the TET. This work can be extended by performing an in-depth analysis of complex cloudlet batches. Further, we plan to shift the sampling of cloudlets from an unimodal distribution to a multi-modal distribution by emphasizing statistical properties like modality. On the other hand, we intend to broaden the scope of TQ by incorporating other statistical measures like kurtosis.

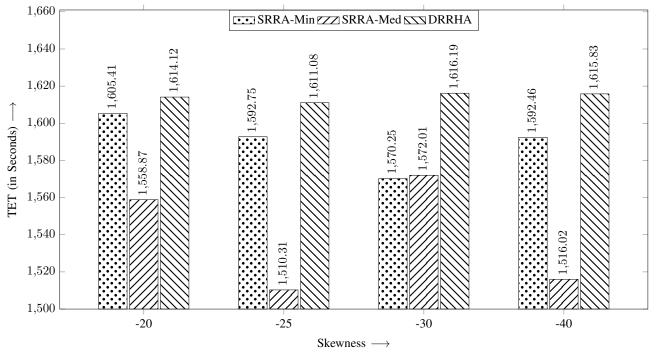

Figure 6. Graphical comparison of SRRA-Min versus SRRA-Med versus DRRHA in terms of TET with varying skewness and cloudlets = 150

References

[1] Caesar Wu, Adel Nadjaran Toosi, Rajkumar Buyya, and Kotagiri Ramamohanarao. Hedonic pricing of cloud computing services. *IEEE Transactions on Cloud Computing*, 9(1):182–196, 2021.

[2] Sanjaya K Panda and Prasanta K Jana. Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment. *Information Systems Frontiers*, 20:373–399, 2018.

[3] S Santhosh and Narayana Swamy Ramaiah. Cloud-based software development lifecycle: A simplified algorithm for cloud service provider evaluation with metric analysis. *Big Data Mining and Analytics*, 6(2):127–138, 2023.

[4] Jaspreet Singh and Navpreet Kaur Walia. A comprehensive review of cloud computing virtual machine consolidation. *IEEE Access*, 2023.

[5] Limali Sahoo, Sanjaya Kumar Panda, and Kunal Kumar Das. A review on integration of vehicular ad-hoc networks and cloud computing. *International Journal of Cloud Applications and Computing (IJCAC)*, 12(1):1–23, 2022.

[6] Vijay Kumar Sharma, Arjun Singh, Krishna R Jaya, Amit Kumar Bairwa, and Devesh Kumar Srivastava. Introduction to virtualization in cloud computing. In *Machine Learning and Optimization Models for Optimization in Cloud*, pages 1–14. Chapman and Hall/CRC, 2022.

[7] Spyridon Chouliaras and Stelios Sotiriadis. An adaptive auto-scaling framework for cloud resource provisioning. *Future Generation Computer Systems*, 2023.

[8] Kenneth K Azumah, Lene T Sørensen, Raffaele Montella, and Sokol Kosta. Process mining-constrained scheduling in the hybrid cloud. *Concurrency and Computation: Practice and Experience*, 33(4):e6025, 2021.

[9] Manoel C Silva Filho, Raysa L Oliveira, Claudio C Monteiro, Pedro RM Inácio, and Mário M Freire. Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)*, pages 400–406. IEEE, 2017.

[10] Jungmin Son, TianZhang He, and Rajkumar Buyya. Cloudsimsdn-nfv: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. *Software: Practice and Experience*, 49(12):1748–1764, 2019.

[11] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa A Cherif, Hossam Afifi, and Mohsen Guizani. Edge and fog computing for iot: A survey on current research activities & future directions. *Computer Communications*, 180:210–231, 2021.

[12] Mohammed Alaa Ala'anzy, Mohamed Othman, Zurina Mohd Hanapi, and Mohamed A Alrshah. Locust inspired algorithm for cloudlet scheduling in cloud computing environments. *Sensors*, 21(21):7308, 2021.

[13] Amine Chraibi, Said Ben Alla, and Abdellah Ezzati. An efficient cloudlet scheduling via bin packing in cloud computing. *International Journal of Electrical and Computer Engineering*, 12(3):3226, 2022.

[14] Sourav Kumar Bhoi, Sanjaya Kumar Panda, and Debashee Tarai. Enhancing cpu performance using subcontrary mean dynamic round robin (smdrr) scheduling algorithm. *Journal of Global Research in Computer Science*, 2(12):17–21, 2011.

[15] Sanjaya Kumar Panda and Sourav Kumar Bhoi. An effective round robin algorithm using min-max dispersion measure. *International Journal on Computer Science and Engineering*, 4(1):45, 2012.

[16] Sanjaya Kumar Panda, Debasis Dash, and Jitendra Kumar Rout. A group based time quantum round robin algorithm using min-max spread measure. *International Journal of Computer Applications*, 975:8887.

[17] Sanjaya K Panda, K Ramesh, Karanam Indraneel, Manam Ramu, and N Navya Damayanthi. Novel service broker and load balancing policies for cloudsim-based visual modeller. In *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 232–237. IEEE, 2022.

[18] Ruba Abu Khurma, HEBA Al Harahsheh, and Ahmad Sharieh. Task scheduling algorithm in cloud computing based on modified round robin algorithm. *Journal of Theoretical & Applied Information Technology*, 96(17), 2018.

[19] Sourav Banerjee, Akash Chowdhury, Swastik Mukherjee, and Utpal Biswas. An approach towards development of a new cloudlet allocation policy with dynamic time quantum. *Automatic Control and Computer Sciences*, 52:208–219, 2018.

[20] Taghreed Balharith and Fahd Alhaidari. Round robin scheduling algorithm in cpu and cloud computing: a review. In *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–7. IEEE, 2019.

[21] Yosef Hasan Jbara. A new improved round robin-based scheduling algorithm-a comparative analysis. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–6. IEEE, 2019.

[22] MS Sanaj and PM Joe Prathap. An enhanced round robin (err) algorithm for effective and efficient task scheduling in cloud environment. In *2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, pages 107–110. IEEE, 2020.

[23] Fahd Alhaidari and Taghreed Zayed Balharith. Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. *Computers*, 10(5):63, 2021.

[24] Abbas Noon, Ali Kalakech, and Seifedine Kadry. A new round robin based scheduling algorithm for operating systems: Dynamic quantum using the mean average. *International Journal of Computer Science Issues*, 2011.

[25] Abraham Silberschatz, James L Peterson, and Peter B Galvin. *Operating system concepts*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[26] Jilan Chen, Dan Wang, and Wenbing Zhao. A task scheduling algorithm for hadoop platform. *Journal of Computers*, 8(4):929–936, 2013.

[27] Israel Cohen, Yiteng Huang, Jingdong Chen, Jacob Benesty, Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*, pages 1–4, 2009.

[28] Sanjaya Kumar Panda, Pratik Agrawal, Pabitra Mohan Khilar, and Durga Prasad Mohapatra. Skewness-based min-min max-min heuristic for grid task scheduling. 2014.

[29] Range of values of skewness and kurtosis for normal distribution. https://stats.stackexchange.com/q/245953. Accessed: 2023-09-30.