IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 February 2022, accepted 17 March 2022, date of publication 18 April 2022, date of current version 12 May 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3168288

Adaptive Scheduling Algorithm Based Task Loading in Cloud Data Centers

DIBYENDU MUKHERJEE[1], SHIVNATH GHOSH[1], SOUVIK PAL[2], (Member, IEEE), AYMAN A. ALY[3], AND DAC-NHUONG LE[4,5]

[1]Department of Computer Science and Engineering, Brainware University, Kolkata, West Bengal 741102, India
[2]Department of Computer Science and Engineering, Sister Nivedita University, Kolkata, West Bengal 741102, India
[3]Department of Mechanical Engineering, College of Engineering, Taif University, Taif 21944, Saudi Arabia
[4]Institute of Research and Development, Duy Tan University, Danang 550000, Vietnam
[5]School of Computer Science, Duy Tan University, Danang 550000, Vietnam

Corresponding author: Dac-Nhuong Le (ledacnhuong@duytan.edu.vn)

This work was supported by Taif University Researchers, Taif University, Taif, Saudi Arabia, under Project TURSP-2020/77.

ABSTRACT Cloud computing is a global storage framework whereby users use the tools, including the computation, storage, network, etc., that are provided automatically. The computational has led the cloud data centers to be stored in digital services that many consumers distribute. The biggest problem with cloud data centers is handling the millions of users' continuous proposals. Therefore, in this paper, Adaptive Scheduling Algorithm Based Task Loading (ASA-TL) has been proposed to manage cloud data centers' task to be stored in digital devices. ASA is implemented in which the task is shared between all the current virtual servers,and the cloud data are protected from overloading. The data assignment is made by taking account of the importance and status of the digital device that helps to assign task fairly and use them efficiently. TL manages these requests efficiently, and the task input must be equally and reliably allocated between various processors. The experimental result suggests that ASA-TL achieves the highest performance assessment measurements, including response time, processing time in the data center and overall expense.

INDEX TERMS Data center, cloud computing, response time, task management.

I. INTRODUCTION

Recently, cloud computing techniques has gain popularity due to the rapid development of Internet of Things (IoT). The data generated using IoT devices are huge. This data is transferred, stored and processed in the cloud [1]. Since the number of devices accessing the cloud data centre is rapidly increasing, proper allocation of cloud resources to these devices is a challenging task [2]. To ensure reliable quality of service, all the IoT devices must be given priority based on the arrival time, type of data, quality of data, bandwidth requirement and virtual server requirement. The overall allocation must be designed such that the energy requirement is low and quality is high [3].

The resources inside the cloud are termed as virtual resources. This resource is essential for effective cloud computing. The computations performed inside the cloud are highly confidential and require less time since the IoT applications are mainly real-time based [4]. Millions of users send access request to the cloud at the same time. To handle this issue, load balancing is performed. Load balancing algorithms are used for selecting suitable request that are of high priority and allocate them to virtual machines based on their availability [5].

The allocated resources are used by the remote users to run their applications in real-time. This allocation is done based on scheduling algorithms. The cloud data centers provide reliable and high quality data compared to the private servers [6]. Load balancing is usually done based on the measurement of constraints [7]. The load request on each virtual machine is computed and evaluated. Based on the load request and the available constraints, decisions are taken to select the best request for resource allocation [8].

Every task is given a certain level of priority. The task with maximum priority is chosen as the balanced load [9]. The main objective is load balancing is to maximize the efficiency of scheduling and minimize the response time. These two objectives aids in improving the overall efficiency of the cloud [10].Various optimizations techniques have been

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

proposed in the literature for the computation of effective scheduling mechanisms. These optimization techniques are based on the evaluation of processing time, overall expense, data transfer rate and the bandwidth rate [11].

Genetic algorithms are popularly used in task allocation as they have high reliability. These algorithms are based on the greedy resource allocation techniques that have high speed. Further, the data assignment in these systems are done based on the available virtual server resource [12]. To reduce the risk of data leakage these schemes usually involve encryption algorithms. The encryption is done so that the intruders do not understand the information being sent through the cloud [13].

Other important criteria involved in scheduling include processor speed, task completion time, load balancing factor, stability factor and the CPU efficiency.The ratio of successful execution determines the rate at which proper execution is performed [14]. This ratio is based on the probabilistic approach. The networks are servers are evaluated to form the balancing task using probabilistic theory. Thus, load balancing helps to balance the performance of the cloud using IoT devices and cloud resources [15].

In the subject of cloud computing, the job scheduling problem is a major research topic. It is concerned with reducing the amount of time spent waiting after applying scheduling algorithms. Job scheduling improves cloud performance so that you may make the most money. The goal of employing different scheduling algorithms is to discover a good set of jobs to execute and lower the overall job execution time. Because the cloud is a distributed environment with heterogeneous systems, there are a variety of scheduling methods available in the cloud environment that differ from traditional scheduling algorithms. Traditional scheduling strategies may not apply to cloud systems. The use of an effective and adaptive scheduling algorithm improves resource efficiency while also lowering customer wait times and job offloading overhead.

Effective scheduling algorithms are essential to produce high quality load balancing to meet the increasing demands of the internet users. The main issue faced here is the decrease in the network speed. To overcome this issue, a novel adaptive scheduling algorithm is proposed in this research. Section 2 involves the literature survey. Section 3 lists the main contributions of this paper. Section 4 explains the proposed methodology. Section 5 explains the results and discussion Section 6 includes the conclusion along with future work.

II. RELATED WORK

Mishra *et al.* [16] proposed a scheme for the load balancing based on optimization techniques. This scheme was designed to overcome both the underload and the overload issues involved in task loading. The main advantage of this technique was that it was capable of handling both dependant and independent virtual machine. However, this scheme was not capable of reducing the power consumption issue. Abualigah *et al.* [17] proposed a framework based on hybrid antlion optimization for task scheduling. This scheme was designed to overcome the multi-objective task loading problem. It was capable of attaining maximum resource utilization. However, the convergence speed of this technique was too low.

Yuan *et al.* [18] presented a system for task scheduling using bio objective. This scheme was designed to support the green data centres. A new algorithm called bio-objective differential evolution technique was adopted. This scheme attained high quality of service with high speed. However, it was unable to support the required stability factor. Naik *et al.* [19] presented a scheme for the adaptive selection of virtual machine. This selection was done using optimized scheduling algorithm. The main objective was to reduce the resource wastage and to enhance the load volume. The overall processor speed was found to be low in this framework.

Yang *et al.* [20] adopted task scheduling using game theory. This theory was used for the management of energy in the cloud environment. The main advantage here was the simplified model involved for the task assignment. Though the model used was simple, this scheme could not achieve rapid allocation. Arani *et al.* [21] presented a methodology based on fog computing for effective task loading. This scheme utilized moth-flame algorithm for optimization of the cloud resources. The proposed algorithm achieved highest cyber security. The main drawback was the high computation requirement since it involved NP hard problem.

Gupta *et al.* [22] presented a scheme for load balancing based on risk management. This scheme utilized the advanced scheduling algorithm for resource allocation. The proposed framework solved the service level agreement issue. However, this scheme could not achieve CPU efficiency. Roy *et al.* [23] proposed a system for load balancing based on cooling techniques. The main advantage of this scheme was the efficient data center energy allocation using green cloud methodology. Though this system achieved increased energy efficiency it was not capable of attaining improved bandwidth rate [24].

Mishra *et al.* [25] presented a scheme for task allocation based on frequency scaling. In this scheme, the available frequency was scaled and evaluated for effective bandwidth allocation. Thus, this scheme attained improved bandwidth rate. However, since the environment involved was of heterogeneous type, it could not attain high data rate, since the optimization involved solving of NP hard problem.

S. Pal *et al.* [26] have discussed resource migration algorithm cloud migration environment. S. Jeyalaksshmi *et al.* [27] have discussed that Reliability Profile algorithms are used in Virtual Machine mapping and allotment in Volunteer Cloud systems [28]–[34]. S. Pal *et al.* [35] have shown acclimatization of Johnson Sequencing in a Cloud Computing Environment for Job Scheduling to Reduce Average Waiting Time.

Khazaei *et al.* [36] offer a fresh estimated methodical approach that deals with server performance evaluation. Their research aims to obtain an approximation of the entire

probability distribution of request response time. The cloud providers can use their model to determine the relationship between input buffer size and the number of Service Centers (SCs). They looked at the likelihood of immediate service, the likelihood of blocking, and the average number of tasks in the system.

In heterogeneous systems, Spicuglia *et al.* [37] demonstrated the techniques for gathering data from various data centers. They talked about how to join the best queue and how to use a plug-and-play workload controller to reduce volatility and the upper percentile of response times [38]–[40].

From the detailed survey, it is evident that the main challenges involved in the task loading of cloud servers include increase in bandwidth rate, decrease in task completion time, improvement in load balancing factor, decrease in computation time, decrease in utilization of virtual server source and improvement in data assignment efficiency. These challenges are overcome using the proposed Adaptive Scheduling Algorithm Based Task Loading (ASA-TL) algorithm since it involved adaptive scheduling based on the evaluation of available resources in the cloud data center.

III. CONTRIBUTION OF THE PAPER

The task is spread across all present virtual servers, and the cloud data is secured from overloading, thanks to ASA. The data assignment is established by considering the value and state of the digital device, which aids in the fair assignment of tasks and efficient use of them. TL efficiently manages these requests, and the task input must be distributed evenly and consistently among the numerous processors. According to the results of the trial, ASA-TL has the best performance in terms of response time, processing time in the data centre, and overall cost.

The contributions of this paper are as follows

- A new algorithm called Adaptive Scheduling Algorithm Based Task Loading (ASA-TL) is proposed.
- A novel technique will be useful for task offloading to make it better resource utilization in cloud environment.
- Overload protection using cloud data center is presented through this algorithm.
- We have tried to focus to minimize the request response time, data processing time, data transfer rate, overall expense rate, and bandwidth efficiency.
- The proposed ASA-TL archives mini- mum task completion time of 16.76ms, high load balancing rate of 94.02%, minimum response time of 7.5ms, minimum processing time of 5.25ms, minimum overall expense rate of 5.36% and archives maximum data transfer rate of 89.81%.

IV. PROPOSED METHODOLOGY

A. PROPOSED ADAPTIVE SCHEDULING ALGORITHM BASED TASK LOADING (ASA-TL) MODEL

The proposed ASA-TL model involves components like users that send access requests, cloud service provider and the cloud servers.

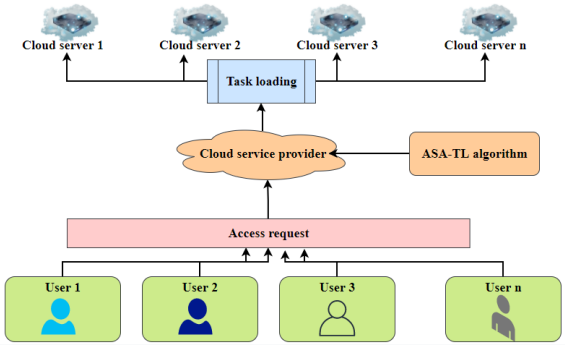

FIGURE 1. Adaptive scheduling algorithm based task loading (ASA-TL) model.

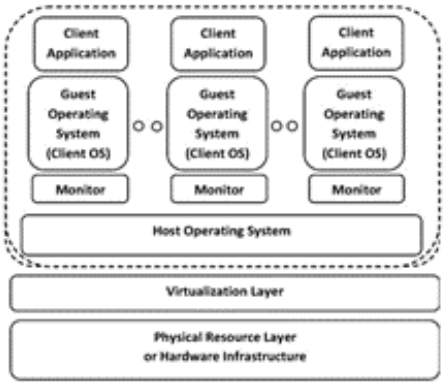

FIGURE 2. Virtual machine for task loading.

Figure 1 shows the proposed Adaptive Scheduling Algorithm Based Task Loading (ASA-TL) model. In this model, the system is designed such that the proposed ASA-TL algorithm is fed to the cloud service provider. The access requests are sent to the cloud from the users. Various users send the requests simultaneously. These requests are evaluated by the cloud service provider and allocation is done based on the proposed ASA-TL algorithm. The task loading is done based on load balancing and the selected requests are assigned to the corresponding cloud servers.

B. VIRTUAL MACHINE FOR TASK LOADING

There are three main layers involved in the proposed methodology. These include application layer, virtual machine and host.

Figure 2 shows the block diagram of the task loading operation. The Bottom layer contains physical Resources i.e. Servers, networks, storage, and data center space, which provides all the resources to the VMs. The next layer is the virtualization layer which is referred as management layer which provides overall management, such as decision making regarding where to deploy the virtual machines, admission control, resource control, usage of accounting, etc. The implementation or application layer provides the hosting environment for virtual machines for the end users.

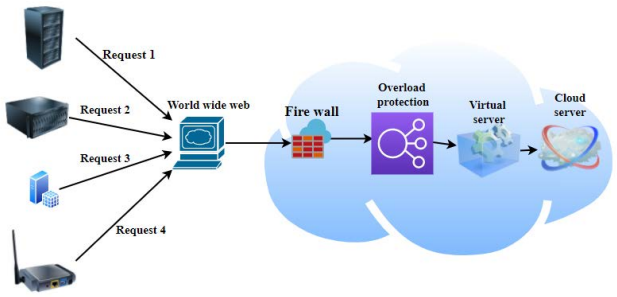

FIGURE 3. Overload protection in cloud data centre.

In this layer, host operating system is running on the hardware infrastructure. It supports multiple guest virtualized operating system on the same physical server and it is capable of maintaining isolation of different guest operating system. In modular approach of cloud computing architecture, there are different software-based components or modules which are assigned to different tasks. So there is need of workflow coordination and proper interaction between different modules. The workflow management system is required for co-ordination and interaction between different modules.

C. OVERLOAD PROTECTION IN CLOUD DATA CENTER

The overload protection is an important component of task balancing. This helps to increase the data rate of the users.

Figure 3 presents the overload protection in the cloud data center. The requests sent by users are given to the cloud using the world wide web server. This server sends the requests to the cloud using the firewall protection. The firewall protection is used for protecting the cloud from the unwanted third party intruders and malware programs. The data filtered by the firewall enters the overload protection center. The overload protection center protects the cloud server from the overloading issue. If the number of requests is more than the level that can be handled by the cloud, few requests are rejected based on the priority rule. Finally, the selected data is transferred to the cloud server through the virtual server.

D. ADAPTIVE TASK SCHEDULING ALGORITHM

The proposed adaptive task scheduling algorithm works based on the evaluation of important parameters like bandwidth requirement, task completion time, task completion rate, processor speed, processor availability, data transfer rate and overall expense.

Figure 4 shows the task scheduling flowchart. From the figure, we see that there are totally 9 tasks involved. These tasks are connected based on directional connections. Each task is dependent on the other. At the first level, the task 1 is performed. At the next level, task 2, task 3 and the task 4 are performed. At the third level, the next four tasks are performed. Finally, the task 9 is performed at the last level.

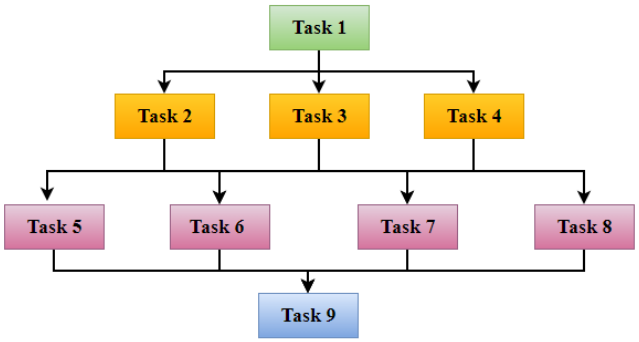

FIGURE 4. Task scheduling flowchart.

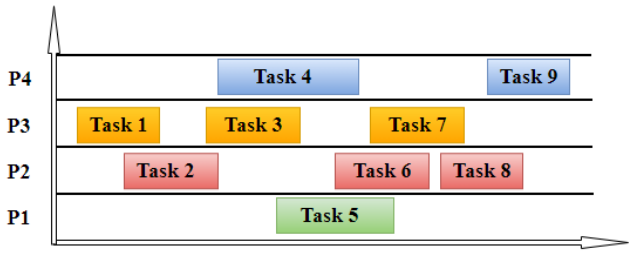

FIGURE 5. Proposed adaptive task scheduling.

Figure 5 shows the proposed adaptive task scheduling. It includes 4 processors. The first task is performed by the third processor P_3. The second task is done by the second processor P_2 followed by the third processor P_3, fourth processor P_4, first processor P_1, second processor P_2, third processor P_3, first processor P_1 and the fourth processor P_4. The Terminologies have been used as follows:

- t_i is the task value, which refers to the number of tasks under processing.
- l_i is the loading value, which refers to the time taken by the server to provide response to the access request.
- d_i is the total data assignment value, which refers to the overall time taken for processing each request.
- c_i is the cloud resource value, which refers to the time of cloud resource utilization during overall processing each request.
- n_i is the network resource value, which refers to the time of network resource allocation for processing.
- s_i is the network speed value, which refers to the network speed at which data is being sent.
- v_i is the virtual server resource value, which refers to total time taken for the completion of the task loading during virtualization process.
- r_i is the date rate value, which refers to the rate at which the task loading is done effectively. The tasks are being loaded to be assigned to its designated processor. (As per allocation policy of Simulator, i.e., VMAllocationPolicy).

Algorithm 1 Adaptive Scheduling Algorithm Based Task Loading (ASA-TL)

Input:
- Task set $T = [t_1, t_2, \ldots, t_m]$;
- Loading set $L = [l_1, l_2, \ldots, l_m]$;
- Data assignment set $D = [d_1, d_2, \ldots, d_m]$;
- Cloud resource $C = [c_1, c_2, \ldots, c_m]$;
- Network resource $N = [n_1, n_2, \ldots, n_m]$;
- Network speed $S = [s_1, s_2, \ldots, s_m]$;
- Virtual server resource $V = [v_1, v_2, \ldots, v_m]$;
- Data rate $R = [r_1, r_2, \ldots, r_m]$;

Output: Task scheduling optimization function *TSO*;

BEGIN

$Iter = 1$;
$Num = 0$;
$Max\_iter = 100$;
While $(iter+ < Max\_iter)$ **do**
Initialize $TS\_array[Num] = 0$;
Compute $RS(t)$;
Compute $PT(t)$;
For $t = 1$ **to** Max_iter **do**
$Pro = 0$;
Compute $OE(t)$;
Compute $DTR(t)$;
Compute $BR(t)$;
Compute $PS(t)$;
endfor
$Pro++$;
Compute $TCT(t)$;
Compute $LBF(t)$;
Compute $SF(t)$;
$TS\_array[Num] = SF(t)$;
endwhile
Calculate Task scheduling optimization function *TSO*;

END

Response time is computed as

$$RT(t) = \sum_{i=1}^{m} [t_i \times \frac{\int d_i}{i} - \arg\max_i l_i] \tag{1}$$

where $RT(t)$ refers to the response time, t_i is the task value, l_i is the loading value, d_i is the data assignment value, $\frac{\int d_i}{i}$ is the integration of data assignment value, $\arg\max_i l_i$ is the argument that gives maximum loading value. This value indicates the time taken by the server to provide response to the access request.

Processing time is computed as

$$PT(t) = RT(t) \times \prod_{i=1}^{m} l_i \times \sum_{i=1}^{m} d_i \times \|RT(t) - c_i\|_2^2 \tag{2}$$

where $PT(t)$ is the processing time, $RT(t)$ refers to the response time, l_i is the loading value, d_i is the data assignment value, c_i is the cloud resource value, $\|RT(t) - c_i\|_2^2$ gives the difference between response time and the cloud resource and $\sum_{i=1}^{m} d_i$ is the total data assignment value. This value indicates the overall time taken for processing each request.

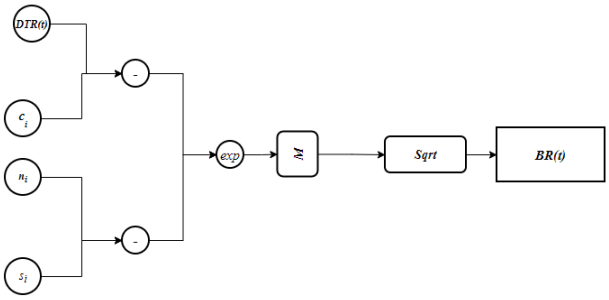

FIGURE 6. Path diagram for the computation of bandwidth rate.

Overall expense is computed as

$$OE(t) = \oint_{i \geq 0} \max\{d_i, 0\} - \|PT(t) - c_i\|_2^2 + n_i \tag{3}$$

where $OE(t)$ is the overall expense, $PT(t)$ is the processing time, d_i is the data assignment value, c_i is the cloud resource value, n_i is the network resource value and $\|PT(t) - c_i\|_2^2$ gives the difference between the processing time and the cloud resource value. The overall expense is the total cost involved in the processing on one request.

Data transfer rate is computed as

$$DTR(t) = \frac{\{OE(t+1) + RT(t)\} \cup \{PT(t) + OE(t+1)\}}{t} \tag{4}$$

where $DTR(t)$ is the data transfer rate, $OE(t)$ is the overall expense, $PT(t)$ is the processing time, $RT(t)$ refers to the response time, t refers to the time instant and $\cup$ is the union operator. This value is the rate at which the data is transferred from the server to the client.

Figure 6 shows the path diagram for the computation of bandwidth rate. Bandwidth rate is given by

$$BR(t) = \sqrt{\sum_{i=1}^{m} \exp(DTR(t) - c_i) + \exp(n_i - s_i)} \tag{5}$$

where $BR(t)$ is the bandwidth rate, $DTR(t)$ is the data transfer rate, c_i is the cloud resource value, n_i is the network resource value, s_i is the network speed value, $\exp(n_i - s_i)$ gives the exponential difference between the network resource value and the network speed value. This value indicates the network speed at which data is sent.

Processor speed is calculated using

$$PS(t) = \frac{1}{m-1} \sum\nolimits_{i=1}^{m} \partial \frac{[BR(t) \times \bmod (BR(t))]}{s_i} - [n_i * v_i] \tag{6}$$

where $PS(t)$ is the processor speed, $BR(t)$ is the bandwidth rate, n_i is the network resource value, s_i is the network speed

value, v_i is the virtual server resource value and the term $n_i \times v_i$ gives the product of the network resource and the virtual server resource value. The processor speed is the speed with which the overall processing is performed.

Task completion time is computed using

$$TCT(t) = \pm \sum_{i=1}^{m} PS(t) \times \frac{s_i * \exp(v_i)}{s_i \times \exp(r_i)} \pm PS(t) \times \sum_{i=1}^{m} \frac{v_i \times \exp(s_i)}{v_i \times \exp(r_i)} \quad (7)$$

where $TCT(t)$ is the task completion time, $PS(t)$ is the processor speed, s_i is the network speed value, v_i is the virtual server resource value, r_i is the data rate value and the term $\exp(r_i)$ is the exponential data rate. This value gives the total time taken for the completion of the task loading.

Load balancing factor is given by

$$LBF(t) = \arg\min_{i} \frac{TCT(t) + \exp(s_i - v_i)}{r_i} \quad (8)$$

where $LBF(t)$ is the load balancing factor, $TCT(t)$ is the task completion time, s_i is the network speed value, v_i is the virtual server resource value, r_i is the data rate value and the term $\exp(s_i - v_i)$ gives the exponential difference between the network speed value and the virtual server resource value. This value gives the rate at which the task loading is done effectively.

Stability factor is calculated as

$$SF(t) = \sqrt{\{PS(t-1) - TCT(t+1)\} \cap \{LBF(t-1) - BR(t+1)\}} \quad (9)$$

where $SF(t)$ is the stability factor, $TCT(t)$ is the task completion time, $PS(t)$ is the processor speed, $LBF(t)$ is the load balancing factor and $BR(t)$ is the bandwidth rate. Stability factor computes the stability of the overall virtual machine.

Task scheduling optimization function TSO is given by

$$TSO \triangleq \arg\max_{t} \left\| \frac{(RS(t) + PT(t) + OE(t))}{(DTR(t) + BR(t) + PS(t))} \right\|^2 \quad subject\ to\ t \geq 0 \quad (10)$$

where TSO is the task scheduling optimization function, $PS(t)$ is the processor speed, $OE(t)$ is the overall expense, $PT(t)$ is the processing time, $BR(t)$ is the bandwidth rate, $DTR(t)$ is the data transfer rate and $PS(t)$ is the processor speed.

Figure 7 shows the task scheduling optimization function generation. In this figure, TSO is the task scheduling optimization function, $PS(t)$ is the processor speed, $OE(t)$ is the overall expense, $PT(t)$ is the processing time, $BR(t)$ is the bandwidth rate, $DTR(t)$ is the data transfer rate and $PS(t)$ is the processor speed.

V. RESULTS AND DISCUSSION

In this section, we have performed the performance analysis of the proposed scheme. The proposed ASA-TL scheme was compared using various algorithms like Imperialist competitive algorithm (ICA), Genetic algorithm (GA), Firefly algorithm (FA), Tabu search (TS), Ant colony optimization (ACO) and Constrained optimization (CO). All the simulations were performed at the frequency of 50Hz using CloudSim Toolkit in a system with 4GB RAM.

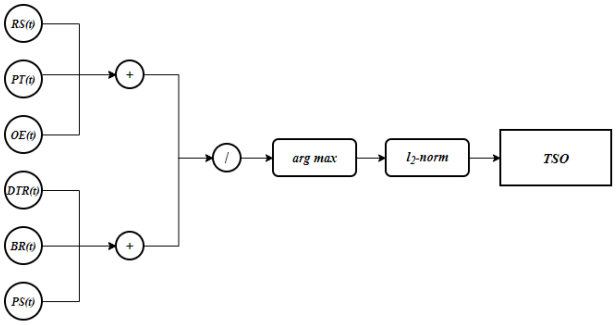

FIGURE 7. Path diagram of task scheduling optimization function computation.

TABLE 1. Task completion time (ms).

Task completion time (ms)							
Tasks	ICA	GA	FA	TS	ACO	CO	ASA-TL
1	59.07	40.47	39.62	56.93	73.70	34.10	17.29
2	76.42	57.62	36.15	64.76	43.51	35.28	17.38
3	59.01	61.50	40.27	54.96	40.42	37.10	10.63
4	30.84	31.59	37.32	56.79	58.25	38.32	18.61
5	36.04	60.74	39.45	52.26	62.02	61.05	19.35
6	73.14	48.12	32.13	36.19	50.85	58.69	19.85
7	54.21	32.47	61.76	54.52	40.30	32.60	18.59
8	72.25	54.48	44.09	72.65	77.40	76.56	17.86
9	57.07	43.57	32.52	51.53	74.76	31.60	11.29

A. PERFORMANCE ANALYSIS

For quantitative evaluation we have employed metrics like task completion time, load balancing rate, response time, processing time, overall expense, data transfer rate, bandwidth rate, processor speed and stability factor.

Table 1 shows the comparison of task completion time. It is clear that the average task completion time for the Imperialist competitive algorithm is 57.56ms. The task completion time for Genetic algorithm is 47.84 ms and for Firefly algorithm is 40.36 ms. Tabu search, Ant colony optimization and Constrained optimization utilize 55.62 ms, 57.91 ms and 45.03 ms respectively. However, the proposed ASA-TL archives minimum time of 16.76 ms. This clearly shows the high speed of the proposed algorithm. This is due to the calculation of response time of each user in the proposed system.

Table 2 shows the comparison of load balancing rate. ICA has average load balancing rate of 43.89%. GA has a rate of 41.51% and the FA has a load balancing rate of 49.57%. The TS algorithm has a rate of 36.81% and the ACO has a rate of 47.82%. CO has load balancing rate of 41.29%. However, the

TABLE 2. Load balancing rate (%).

Load balancing rate (%)							
Tasks	ICA	GA	FA	TS	ACO	CO	ASA-TL
1	39.99	43.38	46.2	33	57.16	35.52	95.13
2	44.01	31.62	51.21	35.34	50.26	47.92	91.77
3	49.45	35.31	59.99	40.79	44.05	39	93.98
4	30.75	49.89	38.63	31.7	57.37	34.02	91.34
5	55.27	39.92	42.43	45.66	33.12	36.38	90.3
6	46.77	56.96	43.94	40.07	52.37	56.85	99.4
7	55.63	33.54	52.92	35.27	52.09	32.14	93.01
8	40.43	59.66	54.55	36.27	46.86	37.27	92.95
9	32.79	23.38	56.32	33.23	37.16	52.52	98.33

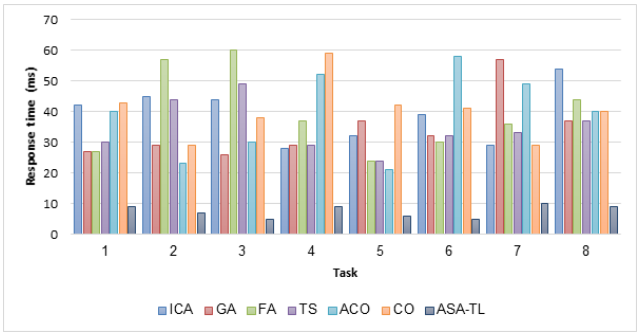

FIGURE 8. Variation of response time.

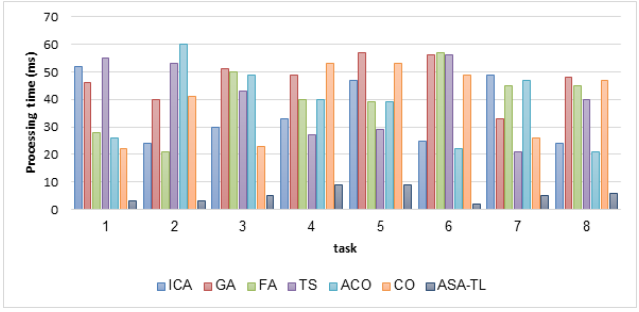

FIGURE 9. Variation of processing time.

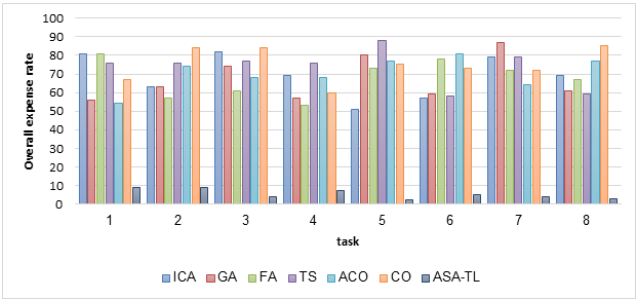

FIGURE 10. Variation of overall expense rate.

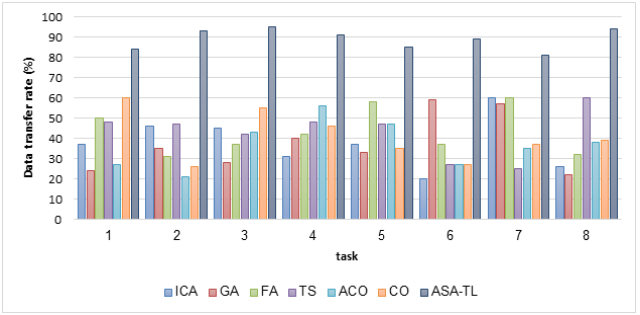

FIGURE 11. Variation of data transfer rate.

proposed scheme has a high rate of 94.02%. This is due to the usage of effective network resource allocation.

Figure 8 shows the variation of response time. It is clear that the average response time for the Imperialist competitive algorithm is 39.12 ms. The response time for Genetic algorithm is 47.84 ms and for Firefly algorithm is 34.25 ms. Tabu search, Ant colony optimization and Constrained optimization utilize 39.37 ms, 34.74 ms and 39.12 ms respectively. However, the proposed ASA-TL archives minimum time of 7.5 ms. This clearly shows the high speed of the proposed algorithm. This is due to the calculation of resource allocation time of each server in the proposed system.

Figure 9 shows the variation of processing time. It is clear that the average processing time for the Imperialist competitive algorithm is 35.2ms. The processing time for Genetic algorithm is 47.52ms and for Firefly algorithm is 40.15ms. Tabu search, Ant colony optimization and Constrained optimization utilize 32.37ms, 34.4ms and 34.12ms respectively. However, the proposed ASA-TL archives minimum time of 5.25ms. This clearly shows the high processing speed of the proposed algorithm. This is due to the incorporation of time efficiency factor in the proposed ASA-TL algorithm.

Figure 10 shows the variation of overall expense rate. It is clear that the overall expense rate for the Imperialist competitive algorithm is 42.52%. The overall expense rate for Genetic algorithm is 41.75% and for Firefly algorithm is 48.43%. Tabu search, Ant colony optimization and Constrained optimization have expense rate of 48.42%, 46.29% and 43.85% respectively. However, the proposed ASA-TL archives minimum-overall expense rate of 5.36%. This clearly shows the high expense capacity of the proposed algorithm.

Figure 11 shows the variation ofdata transfer rate. It is clear that the average data transfer ratefor the Imperialist competitive algorithm is 51.23%. The data transfer ratefor Genetic algorithm is 51.57% and for Firefly algorithm is 50.31%. Tabu search, Ant colony optimization and Constrained optimization have data transfer rate of39.53%, 41.32% and 42.51% respectively. However, the proposed ASA-TL archives maximum data transfer rateof 89.81%. This clearly shows the high data transfer rate of the proposed algorithm. This is due to the implementation of overall expense in terms of bandwidth rate.

Table 3 shows the comparison of stability factor. It is clear that the average stability factor for the Imperialist competitive algorithm is 47.77%. The stability factor for

TABLE 3. Comparison of stability factor (%).

Stability factor (%)							
Tasks	ICA	GA	FA	TS	ACO	CO	ASA-TL
1	52.48	41.07	41.65	37.12	58.94	37.98	89.43
2	47.5	50.55	46.55	45.93	42.97	53.94	86.84
3	52.2	47.94	36.87	32.74	50.84	44.63	81.32
4	37.04	53.68	49.26	42.16	52.75	53.07	87.23
5	52.05	41.03	44.53	33.14	42.98	41.88	81.1
6	59.12	36.18	34.55	33.36	49.67	38.19	81.17
7	56.01	32.6	53.46	53.54	33.29	31.11	86.41
8	32.58	53.16	33.01	38.75	58.02	50.2	83.29
9	40.99	36.17	38.82	48.11	35.62	42.89	86.54

TABLE 4. Comparison of CPU efficiency (%).

CPU efficiency (%)							
Tasks	ICA	GA	FA	TS	ACO	CO	ASA-TL
1	43.55	30.23	32.95	51.1	34.98	59.49	86.86
2	48.3	42.69	34.26	46.67	48.68	42.06	82.94
3	31.78	49.67	35.04	35.53	59.64	48.62	85.31
4	39.47	51.69	35.88	36.36	35.11	34.63	88.33
5	53.18	45.94	39.52	32.32	37.73	41.44	85.98
6	50.89	33.26	39.49	57.42	41.9	34.83	83.35
7	33.76	48.95	36.52	51.2	32.22	52.75	82.99
8	33.9	33.79	37.53	46.73	50.52	56.14	84.53
9	32.77	34.03	56.79	39.4	42.07	40.52	84.23

Genetic algorithm is 43.57% and for Firefly algorithm is 42.07%. Tabu search, Ant colony optimization and Constrained optimization have stability factor of 40.53%, 47.23% and 43.76% respectively. However, the proposed ASA-TL archives maximum stability factor of 84.81%. This clearly shows the high stability of the proposed algorithm. This is due to the involvement of virtual servers for the adaptive scheduling.

Table 4 shows the comparison of CPU efficiency. It is clear that the average CPU efficiency for the Imperialist competitive algorithm is 40.84%. The CPU efficiency for Genetic algorithm is 41.13% and for Firefly algorithm is 38.66%. Tabu search, Ant colony optimization and Constrained optimization have stability factor of 44.08%, 42.53% and 45.60% respectively. However, the proposed ASA-TL archives maximum CPU efficiency of 84.94%. This clearly shows the high efficiency of the proposed algorithm. This is due to the involvement of multiple cloud servers for task loading in this model.

Figure 12 shows the comparison of bandwidth efficiency. It is clear that the average bandwidth efficiency for the Imperialist competitive algorithm is 45.23%. The bandwidth efficiency for Genetic algorithm is 43.57% and for Firefly algorithm is 41.57%. Tabu search, Ant colony optimization and Constrained optimization have stability factor of 42.42%, 46.23% and 41.75% respectively. However, the proposed ASA-TL archives maximum bandwidth efficiency of 85.75%. This clearly shows the high bandwidth utilization capacity of the proposed algorithm. This is due to the involvement of effective scheduling technique based on the network resource availability.

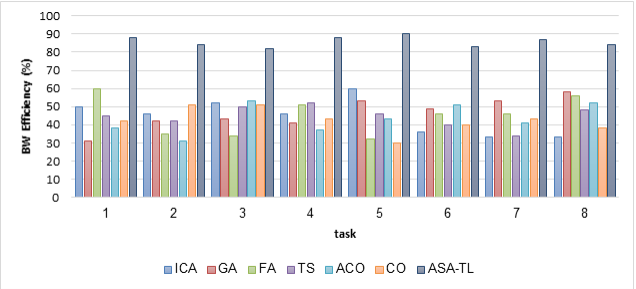

FIGURE 12. Variation of BW efficiency.

In the section, the suggested ASA-TL scheme was compared with Imperialist competitive algorithm (ICA), Genetic algorithm (GA), Firefly algorithm (FA), Tabu search (TS), Ant colony optimization (ACO), and Constrained optimization (CO). The request response time, data processing time, data transfer rate, overall expense rate, and bandwidth efficiency have all been clearly elaborated in tabular format shown in table [1-4]. We have worked to reduce request response time, data processing time, data transfer rate, total expense rate, and bandwidth efficiency to make better performance. The proposed ASA-TL algorithm produces a minimum task completion time of 16.76 milliseconds, a high load balancing rate of 94.02 percent, a minimum response time of 7.5 milliseconds, a minimum processing time of 5.25 milliseconds, a minimum overall expense rate of 5.36 percent, and a maximum data transfer rate of 89.81 percent.

The task is distributed among all available virtual servers, and due to ASA algorithm, the cloud data is protected from overloading. The data assignment is determined on the basis of the worth and condition of the digital equipment, which aids in the equitable assignment of tasks. The task input must be distributed fairly and consistently among the multiple processors, and TL efficiently manages these requests. According to the trial's findings, ASA-TL performs better in terms of response time, data center processing time, and overall cost.

VI. CONCLUSION AND FUTURE WORK

A new model for task loading in cloud data centre based on adaptive scheduling algorithm has been presented in this paper. Here, a new algorithm called Adaptive Scheduling Algorithm Based Task Loading (ASA-TL) has been proposed. In this model, the system is designed such that the proposed ASA-TL algorithm is fed to the cloud service provider. The access requests that are sent to the cloud are then adaptively scheduled using the proposed algorithm to prevent the over- loading issue. This scheme achieved effective load balancing using virtual servers. The proposed ASA-TL archives minimum task completion time of 16.76ms, high load balancing rate of 94.02%, minimum response time

of 7.5ms, minimum processing time of 5.25ms, minimum overall expense rate of 5.36% and archives maximum data transfer rate of 89.81%. In future, we plan to implement the proposed scheme in the cloud server using virtual software resources.

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Scientific Research at Taif University for the grant received for this research.

CONFLICT OF INTERESTS

The authors declare no conflict of interest

REFERENCES

[1] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G. B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *J. Supercomput.*, vol. 76, no. 8, pp. 6302–6329, 2020, doi: 10.1007/s11227-019-02816-7.

[2] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," *J. Netw. Comput. Appl.*, vol. 128, pp. 64–77, Feb. 2019, doi: 10.1016/j.jnca.2018.12.010.

[3] H. M. Nguyen, G. Kalra, and D. Kim, "Host load prediction in cloud computing using long short-term memory encoder–decoder," *J. Supercomput.*, vol. 75, pp. 7592–7605, Aug. 2019, doi: 10.1007/s11227-019-02967-7.

[4] S. M. Shetty and S. Shetty, "Analysis of Load Balancing in Cloud Data Centers," *J. Ambient Intell. Humanized Comput.* Jan. 2019, doi: 10.1007/s12652-018-1106-7.

[5] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019, doi: 10.1016/j.future.2018.09.014.

[6] B. Muthulakshmi and K. Somasundaram, "A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment," *Cluster Comput.*, vol. 22, no. S5, pp. 10769–10777, Sep. 2019, doi: 10.1007/s10586-017-1174-z.

[7] C. Zhang, Z. Qi, J. Yao, M. Yu, and H. Guan, "vGASA: Adaptive scheduling algorithm of virtualized GPU resource in cloud gaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3036–3045, Nov. 2014, doi: 10.1109/TPDS.2013.288.

[8] V. Polepally and K. S. Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing," *Cluster Comput.*, vol. 22, no. 1, pp. 1099–1111, 2019, doi: 10.1007/s10586-017-1056-4.

[9] S. K. Mishra, B. Sahoo, and P. S. Manikyam, "Adaptive scheduling of cloud tasks using ant colony optimization," in *Proc. 3rd Int. Conf. Commun. Inf. Process.*, Nov. 2017, pp. 202–208, doi: 10.1145/3162957.3163032.

[10] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Appl. Soft Comput.*, vol. 76, pp. 416–424, Mar. 2019, doi: 10.1016/j.asoc.2018.12.021.

[11] B. Jana, M. Chakraborty, and T. Mandal, "A task scheduling technique based on particle swarm optimization algorithm in cloud environment," in *Soft Computing: Theories and Applications*. Singapore: Springer, 2019, pp. 525–536.

[12] P. M. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment," *Cluster Comput.*, vol. 22, no. 4, pp. 1241–1251, Dec. 2019, doi: 10.1007/s10586-019-02909-1.

[13] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5792–5805, Jul. 2020, doi: 10.1109/JIOT.2019.2952767.

[14] S. K. Panda and P. K. Jana, "Load balanced task scheduling for cloud computing: A probabilistic approach," *Knowl. Inf. Syst.*, vol. 61, no. 3, pp. 1607–1631, Dec. 2019, doi: 10.1007/s10115-019-01327-4.

[15] W. Lin, G. Wu, X. Wang, and K. Li, "An artificial neural network approach to power consumption model construction for servers in cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 3, pp. 329–340, Jul. 2020, doi: 10.1109/tsusc.2019.2910129.

[16] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 2, pp. 149–158, 2020, doi: 10.1016/j.jksuci.2018.01.003.

[17] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Comput.*, vol. 24, no. 1, pp. 205–223, Mar. 2021, doi: 10.1007/s10586-020-03075-5.

[18] H. Yuan, J. Bi, M. Zhou, Q. Liu, and A. C. Ammari, "Biobjective task scheduling for distributed green data centers," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 731–742, Apr. 2021, doi: 10.1109/tase.2019.2958979.

[19] B. B. Naik, D. Singh, and A. B. Samaddar, "Multi-objective virtual machine selection in cloud data centers using optimized scheduling," *Wireless Pers. Commun.*, vol. 116, no. 3, pp. 2501–2524, Feb. 2021, doi: 10.1007/s11277-020-07807-z.

[20] J. Yang, B. Jiang, Z. Lv, and K.-K.-R. Choo, "A task scheduling algorithm considering game theory designed for energy management in cloud computing," *Future Gener. Comput. Syst.*, vol. 105, pp. 985–992, Apr. 2020, doi: 10.1016/j.future.2017.03.024.

[21] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, "An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, p. e3770, Feb. 2020, doi: 10.1002/ett.3770.

[22] A. Gupta, H. S. Bhadauria, and A. Singh, "SLA-aware load balancing using risk management framework in cloud," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 7, pp. 7559–7568, Jul. 2021, doi: 10.1007/s12652-020-02458-1.

[23] D. Mukherjee, S. Chakraborty, I. G. A. Sarkar, and S. Roy, "A detailed study on data centre energy efficiency and efficient cooling techniques," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 5, pp. 1–22, 2020, doi: 10.30534/ijatcse/2020/332952020.

[24] A. K. Singh and J. Kumar, "Secure and energy aware load balancing framework for cloud data centre networks," *Electron. Lett.*, vol. 55, no. 9, pp. 540–541, 2019.

[25] S. K. Mishra, M. A. Khan, S. Sahoo, and B. Sahoo, "Allocation of energy-efficient task in cloud using DVFS," *Int. J. Comput. Sci. Eng.*, vol. 18, no. 2, pp. 154–163, 2019.

[26] S. Pal, R. Kumar, L. H. Son, K. Saravanan, M. Abdel-Basset, G. Manogaran, and P. H. Thong, "Novel probabilistic resource migration algorithm for cross-cloud live migration of virtual machines in public cloud," *J. Supercomput.*, vol. 75, no. 9, pp. 5848–5865, Sep. 2019.

[27] S. Jeyalaksshmi, M. S. Nidhya, G. Suseendran, S. Pal, and D. Akila, "Developing mapping and allotment in volunteer cloud systems using reliability profile algorithms in a virtual machine," in *Proc. 2nd Int. Conf. Comput., Autom. Knowl. Manage. (ICCAKM)*, Jan. 2021, pp. 97–101.

[28] D. N. Le, R. Kumar, G. N. Nguyen, and J. M. Chatterjee, *Cloud Computing and Virtualization*. Hoboken, NJ, USA: Wiley, 2018.

[29] D.-N. Le, B. Seth, and S. Dalal, "A hybrid approach of secret sharing with fragmentation and encryption in cloud environment for securing outsourced medical database: A revolutionary approach," *J. Cyber Secur. Mobility*, pp. 379–408, Mar. 2018.

[30] B. Seth, S. Dalal, V. Jaglan, D. Le, S. Mohan, and G. Srivastava, "Integrating encryption techniques for secure data storage in the cloud," *Trans. Emerg. Telecommun. Technol.*, vol. 43, Art. no. e4108, no. 4, Sep. 2020, doi: 10.1002/ett.4108.

[31] V. N. Van, N. Q. Long, G. N. Nguyen, and D. N. Le, "A performance analysis of openstack open-source solution for IaaS cloud computing," in *Proc. 2nd Int. Conf. Comput. Commun. Technol.* New Delhi, India: Springer, 2016, pp. 141–150.

[32] V. N. Van, N. Q. Long, and D. N. Le, "Performance analysis of network virtualization in cloud computing infrastructures on openstack," in *Innovations in Computer Science and Engineering*. Singapore: Springer, 2016, pp. 95–103.

[33] D. N. Le, C. Bhatt, and M. Madhukar, Eds., *Security Designs for the Cloud, IoT, and Social Networking*. Hoboken, NJ, USA: Wiley, 2019.

[34] B. Seth, S. Dalal, D.-N. Le, V. Jaglan, N. Dahiya, A. Agrawal, M. Mohan Sharma, D. Prakash, and K. D. Verma, "Secure cloud data storage system using hybrid paillier-blowfish algorithm," *Comput., Mater. Continua*, vol. 67, no. 1, pp. 779–798, 2021.

[35] S. Pal and P. K. Pattnaik, "Adaptation of Johnson sequencing algorithm for job scheduling to minimise the average waiting time in cloud computing environment," *J. Eng. Sci. Technol.*, vol. 11, no. 9, pp. 1282–1295, 2016.

[36] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using M/G/m/m+r queuing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 936–943, May 2012.
[37] S. Spicuglia, L. Y. Chen, and W. Binder, "Join the best queue: Reducing performance variability in heterogeneous systems," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun. 2013, pp. 139–146.
[38] O. I. Khalaf, F. Ajesh, A. A. Hamad, G. N. Nguyen, and D.-N. Le, "Efficient dual-cooperative bait detection scheme for collaborative attackers on mobile ad-hoc networks," *IEEE Access*, vol. 8, pp. 227962–227969, 2020.
[39] Z. Sabir, K. Nisar, M. A. Z. Raja, M. R. Haque, M. Umar, A. A. A. Ibrahim, and D.-N. Le, "IoT technology enabled heuristic model with Morlet wavelet neural network for numerical treatment of heterogeneous mosquito release ecosystem," *IEEE Access*, vol. 9, pp. 132897–132913, 2021.
[40] P. Singh, R. S. Raw, S. A. Khan, M. A. Mohammed, A. A. Aly, and D.-N. Le, "W-GeoR: Weighted geographical routing for VANET's health monitoring applications in urban traffic networks," *IEEE Access*, early access, Jun. 25, 2021, doi: 10.1109/ACCESS.2021.3092426.

DIBYENDU MUKHERJEE received the M.C.A. degree and the M.Tech. degree in the field of computer science and engineering. He is currently a Ph.D. Research Scholar with the Department of Computer Science and Engineering, Brainware University, Kolkata, and also working as an Assistant Professor and the Head of the Computer Application Department, DSMS College of Tourism and Management, Durgapur, West Bengal, India. Prior to that, he was associated with the RICIS College, Raniganj, as a Teacher-in-charge of the IIAM College Durgapur, Softnet Technology, Kolkata, as a SAP Professional. He has overall 12 years of academic and four years of industry experience. He has published a number of research papers in Scopus indexed journals and conferences. His research interests include cloud computing, big data, the IoT, and data analysis. He was a Committee Member of International Business Research Conference 2018 and International Business Research Conference 2019 organized by the DSMS College.

SHIVNATH GHOSH received the master's degree from the National Institute of Technology Raipur and the Ph.D. degree from State Government University Kanpur (CSJMU). He is currently an Associate Professor in computer science and engineering (CSE) with Brainware University, Kolkata, West Bengal, India. He is particularly interested in artificial intelligence, soft computing, machine learning, data science, and quantum computing. He has published more than 30 research papers. He has 16 years of professional experience in the field of educational and IT industry.

SOUVIK PAL (Member, IEEE) received the M.Tech. and Ph.D. degrees in the field of computer science and engineering from KIIT University, Bhubaneswar, India. He is currently an Associate Professor with the Department of Computer Science and Engineering, Sister Nivedita University (Techno India Group), Kolkata, India. Prior to that, he was associated with Global Institute of Management and Technology, Brainware University, Kolkata, JIS College of Engineering, Nadia, Elitte College of Engineering, Kolkata, and Nalanda Institute of Technology, Bhubaneswar, India. He has more than a decade of academic experience. He is the author or coeditor of more than 15 books from reputed publishers, including Elsevier, Springer, CRC Press, and Wiley, and he holds three patents. He is serving as a Series Editor for *Advances in Learning Analytics for Intelligent Cloud-IoT Systems* (Scrivener-Wiley) (Scopus-Indexed), *Internet of Things: Data-Centric Intelligent Computing, Informatics, and Communication* (CRC Press and Taylor & Francis Group, USA), *Conference Proceedings Series on Intelligent Systems, Data Engineering, and Optimization* (CRC Press and Taylor & Francis Group, USA). He has published a number of research papers in Scopus/SCI/SCIE journals and conferences. He is the Organizing Chair of RICE 2019, Vietnam, RICE 2020 Vietnam, and ICICIT 2019, Tunisia. He has been invited as a Keynote Speaker at ICICCT 2019, Turkey, and ICTIDS 2019, 2021 Malaysia. He has also served as a Proceedings Editor for ICICCT, in 2019 and 2020, ICMMCS, in 2020 and 2021, and ICWSNUCA 2021, India. His professional activities include roles as an associate editor, the guest editor, and an editorial board member for more than 100 international journals and conferences of high repute and impact. His research interests include cloud computing, big data, the Internet of Things, wireless sensor networks, and data analytics. He is a member of many professional organizations, including MCSI; MCSTA/ACM, USA; MIAENG, Hong Kong; MIRED, USA; MACEEE, New Delhi; MIACSIT, Singapore; and MAASCIT, USA.

AYMAN A. ALY has been a Professor in mechatronics engineering and the Director of the E-Learning and Distance Education Unit, College of Engineering, Taif University, Saudi Arabia, since 2008. Prior to joining Taif University, he is one of the team, who established the Mechatronics and Robotics Engineering Educational Program at Assiut University. in 2006. He was nominated and selected for inclusion in Marquis Who's Who in the World, 30th Pearl Anniversary Edition, in 2013. He is the author of more than 160 scientific papers and seven textbooks in refereed journals and international conferences. He has supervised and examined some of a M.Sc. and Ph.D. degree students. His research interests include intelligent control of mechatronics systems, robotics design, renewable energy systems, and modelling and simulation.

DAC-NHUONG LE received the M.Sc. and Ph.D. degrees in computer science from Vietnam National University, Vietnam, in 2009 and 2015, respectively. He is currently an Associate Professor of computer science and the Dean of the Faculty of Information Technology, Haiphong University, Vietnam. He also is a Researcher with the Institute of Research and Development, Duy Tan University. He has a total academic teaching experience of more than 15 years with many publications in reputed international conferences, journals, and online book chapters. He has more than 80 publications in the reputed international conferences, journals, and book chapter contributions (indexed by SCIE, SSCI, ESCI, and Scopus). His areas of research in the field of evolutionary multiobjective optimization, network communication and security, cloud computing, and virtual reality/argument reality. Recently, he has been on the technique program committee, the technique reviews, the track chair for international conferences under Springer-ASIC/LNAI/CISC Series. He is currently serving on the editorial board of international journals and edited/authored more than 20 computer science books published by Springer, Wiley, and CRC Press.