Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.034764

ARTICLE

Efficient Cloud Resource Scheduling with an Optimized Throttled Load Balancing Approach

V. Dhilip Kumar[1], J. Praveenchandar[2], Muhammad Arif[3,*], Adrian Brezulianu[4], Oana Geman[5] and Atif Ikram[3,6]

[1]Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, 600062, India

[2]Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, Tamil Nadu, 641114, India

[3]Department of Computer Science, Superior University, Lahore, Pakistan

[4]Faculty of Electronics Telecommunications and Information Technology, Gheorghe Asachi Technical University of Iasi, Iaşi, 700050, Romania

[5]Department of Health and Human Development, Stefan cel Mare University of Suceava, Suceava, 720229, Romania

[6]Faculty of Ocean Engineering Technology and Informatics, University Malaysia Terengganu, Kuala Nerus, Malaysia

*Corresponding Author: Muhammad Arif. Email: arifmuhammad36@hotmail.com

Received: 26 July 2022 Accepted: 14 October 2022 Published: 29 November 2023

ABSTRACT

Cloud Technology is a new platform that offers on-demand computing Peripheral such as storage, processing power, and other computer system resources. It is also referred to as a system that will let the consumers utilize computational resources like databases, servers, storage, and intelligence over the Internet. In a cloud network, load balancing is the process of dividing network traffic among a cluster of available servers to increase efficiency. It is also known as a server pool or server farm. When a single node is overwhelmed, balancing the workload is needed to manage unpredictable workflows. The load balancer sends the load to another free node in this case. We focus on the Balancing of workflows with the proposed approach, and we present a novel method to balance the load that manages the dynamic scheduling process. One of the preexisting load balancing techniques is considered, however it is somewhat modified to fit the scenario at hand. Depending on the experimentation's findings, it is concluded that this suggested approach improves load balancing consistency, response time, and throughput by 6%.

KEYWORDS

Load balancing; throttled algorithm; efficient resource allocation

1 Introduction

Service-oriented architecture (SOA) is a design paradigm that encourages the creation and design of generic applications that could be readily combined and utilized to construct network services. This ensures the growth of optimized and versatile IT infrastructures. The World Wide Web Consortium is being formed with the number of services that can be identified based on their report when published. Modules are offered as stand-alone services that can be accessed consistently. The goal of the SOA is

to improve business functionality modeling and uniformity between participating software solutions. To complete a task, applications call a sequence of separate services. The service provider responds to a customer's query by offering services.

Web services and XML (Extensible Markup Language) are the most widely used version of the Service-Oriented Architecture policy. These are developed to support the web platform with the capability to operate between any modules in which the organization is running.

Web Applications adhere to several World Wide Web protocols, including SOAP (Simple Object Access Protocol) and WSDL (Web Services Definition Language). Such technologies use XML-based connectivity protocols and service specifications to promote compatibility. Web services are vendor, language, and platform-independent because of the usage of XML protocols, making them ideal users for SOA deployments.

2 Load Balancing

Balancing a scientific workflow in a cloud platform is an optimization process to share the running loads or split the loads over the available resources. Cloud load balancing assures the optimized response time with the highest throughput [1]. As said the response time is minimized because the workload is distributed among different computing resources like hard drives, servers, network interfaces, etc. This will help to improve resource utilization and optimizes the response time in the system. In Fig. 1, the load balancing scenario is represented in a diagrammatic form. And the load balancing in cloud applications may also provide better business continuity. The main purposes of workload balancing in a cloud platform are:

- To keep the system stable.
- To preserve the system from crashes.
- To improve the cloud system's performance.

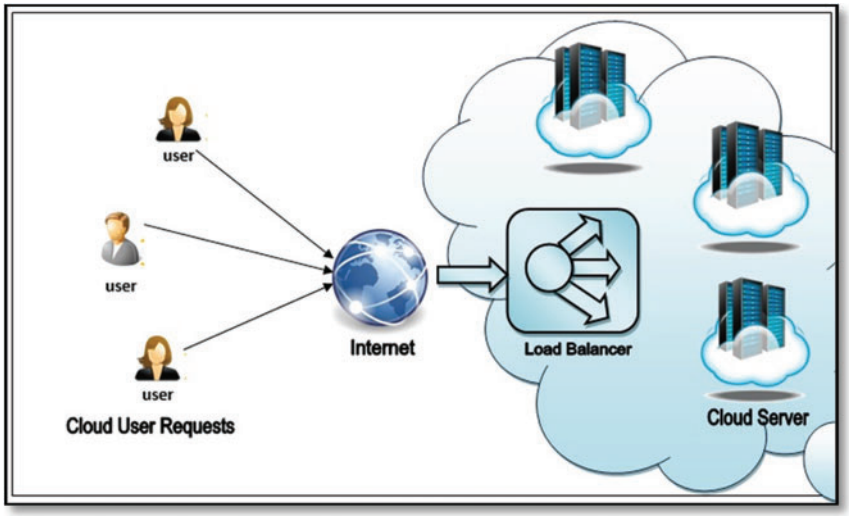

Figure 1: Structure of deep learning

If the load balancer is not there, each user needs to wait until the request gets getting processed. That may take quite a long time and wait for a requested resource that is engaged with some other

task. In this process, different kinds of entities such as CPU's processing rate, jobs waiting in the queue, job arrival time, etc., are being exchanged between each processor [2].

Workload Load Balancing algorithms can be classified into two categories. The first one is static and the next one is dynamic. The load balancing (static) algorithm operates with the previous data of the analytical information and applications of the system and shares the workload equally within the servers [3]. In the case of load balancing (dynamic) algorithms are seeking for the most appropriate server of the system and then the load is assigned to it. In this proposed algorithm, the workload is distributed between the available processors in runtime. During this process, the present workload estimation is an important scenario. Based on that the load balancing system is operated for the environment. Lots of issues may arise in this task since it is a dynamic one and mainly concentrate to achieve optimized response time and maximum throughput.

2.1 Traditional Throttled Algorithm

A throttled load balancer is one of the approaches for balancing workflows in dynamic environments. In the approach, first, the user will request the load balancer to find the most appropriate Virtual Machine to execute the task. There will be many instances of VMs available in cloud environments. All these VMs are arranged based on the kind of user requests they can perform. Then this load balancer checks the groups when it receives any request from any client. After it finds, it allocates the process to the lightly-loaded VM of a specific group. And in the traditional Throttled Load Balancing Algorithm Index Structure is maintained (Table 1) to monitor the VMs and their states to check whether it is busy or available. When a request is raised from the client side to the data center, it searches for the most suitable VM to execute the requested job [4]. To allocate the requested VM for a user request, the data center asks the load balancer. After that, the index Structure of the load balancer is examined from the top till the first idle VM is found. In such a way the Index Structure will be scanned. And in case the searching VM is found, the server contacts the identified VM with the VM identifier. The resource mapping is done. After this, the index structure has been modified as per the allocation done and it initiates the new allocation.

Table 1: Index table

VM	State
VM_1	*Idle*
VM_2	*Busy*
VM_3	*Busy*
:	:
:	:
VM_n	*Idle*

If the most suitable VM is not found during the scanning process of the VM for a client request, −1 will be returned to the data center by the load balancer. Then that request will be queued with the data center. The client query is accepted in the data center once the Virtual Machine executes the allocated task. Then load-balancer advised removing the virtual machine in which the id is given earlier. Overall run time of the process can be measured in 3 phases. During the initial stage, the creation of the VM is done. Then they will be kept in an idle state and kept waiting for state and the scheduler allocates the job in the waiting queue. After the tasks are scheduled with the resource, the corresponding VM is

started executing it. This happened in the second stage. Then during the third stage, after execution, the VM releases the job fetched. Here we can measure the computing throughput using the formula (1.1). Let the total no. of jobs executed (J_{EXT}) and Total no of Jobs arrived (J_{ARV}),

$$Throughput\ (\%) = \frac{(J_{EXT})}{(J_{ARV})} \times 100 \tag{1.1}$$

Here the period of VM creation and VM destruction is negligible.

If an application is transferring the data, how fast it is done [5]? That is called throughput. Bandwidth and the throughput rates are not the same even though both are measured by bits-per Second. But bandwidth can be defined as how fast and how many channels are available between any two nodes of a specific network. And it can be a part of throughput. Both throughput and bandwidth are network metrics.

3 Proposed Approach

Accessing the hash structure to check each resource's availability and access the resources for effective load balancing is required. By optimizing the resource structure, it is possible to minimize the access time of the resources during the dynamic load balancing [6]. In this proposed research work, it is attempted to optimize the hash structure to achieve the scenario. Table 2 is a structure used in the load balancing algorithm to maintain all information regarding all available resources such as VM and PM availability and the current states of both. The data is stored in an associative way.

Table 2: Hash table

PM_1	PM_2	PM_3	....	PM_n
VM_1				
VM_2				
VM_3				
:				
VM_n				

In this hash structure, all data has got an individual index value. The reason behind using this hash structure in cloud resource allocation is faster accessing of data compared to other data structures. Because it is easy to get the data if we have the index of it. And also insertion and deletion of information about the resources are more optimal. And an index value of an available resource is generated using the effective hashing technique. So that quick access is possible and we could get the quick access time of a resource. Following the I-Throttled Load Balancing algorithm and Optimized Load Balancing algorithms represents the modified throttled load balancing approach.

Algorithm 1: I-Throttled Load Balancing

Input: Set of user requests R = {r_1,r_2,r_3 . . . ,r_n}
Output: Updated hash structure with optimized resource allocation
Get the dynamic user request R
Send R {Requests to per unit time} to Optimized Load Balancing module (OLB)
Call **Optimized Load Balancing** ( ) for each R (Per unit time)
End

Algorithm 2: Optimized Load Balancing

Scan for Resources in a hash structure
  Check VM_i has Min load
    If VM_i has Min response time **then**
      Apply Optimize Hash structure Function **(R_n)**
      Send the VM Id to the Scheduler
      Change the status == found
    Accept the Mapping request from the controller
    Map the resource the request
  Return the VM Id to the **hash** structure
Update the Hash structure
End

In this algorithm, Optimize Hash structure Function (R_n) is referred the Robin Hood hashing which is an existing advanced hashing technique [7]. And in this load balancing, the resource structure is a type of data structure to keep track of the key-value pairs of all VM and PM in that way the resources can access instantly using a particular index, which is being generated with the help of a hash function. Here the method used to optimize the hash structure is a fewer collision hash structure. It is one of the effective ways to achieve the task. During the generation of ash functions, key values are kept in the first place. In case, we are doing a string comparison operation, those are encoded as UTF 8. That will not affect the hash functions.

Anyway, every byte of a string is checked by the hash function and it is expected to have a minimum probability of collisions to compute the uniquely crafted values. But the hash function practiced in ASCII strings may not be much faster to compute the values and it may have more collisions too. To solve this issue, a two-level structure is used to generate the optimized hash function [8]. Initially, the hash on the key is calculated after that the hash function is computed exactly for the data which is having more than one entry. If it is a string, the most imminent data might be the hash only the prefix. Typically this does not produce good results in practice. The compiler is a perfect example of this case.

4 Experimentation

GUI-based tool cloud simulation tool Cloud Analyst from CloudSim is used for simulation purposes. It is a toolkit that supports performing simulation, modeling, and other operations in cloud simulation environments. The cloud analyst will allow us to set the location of customers and data centers. The configuration parameters are set based on the taken scenario like the total number of requests that can be made per hour and also per user, the total processors, users, the virtual machines available, network bandwidth, amount of storage, etc., Cloud Analyst computes the results based on the given parameters. Then the results are displayed in graphical form. Fig. 2 represents the execution of the simulation. In that, the following parameters are configured in the cloud Analyst and executed:

- Average Time to process in a data center
- Maximum Time to process in a data center
- Minimum Time to process in a data center
- The Maximum Time interval for response
- The Average Time interval for response
- The Minimum Time interval for response
- VM (Virtual Machine)

- UB (User Base)
- Total Cost
- Data Centers

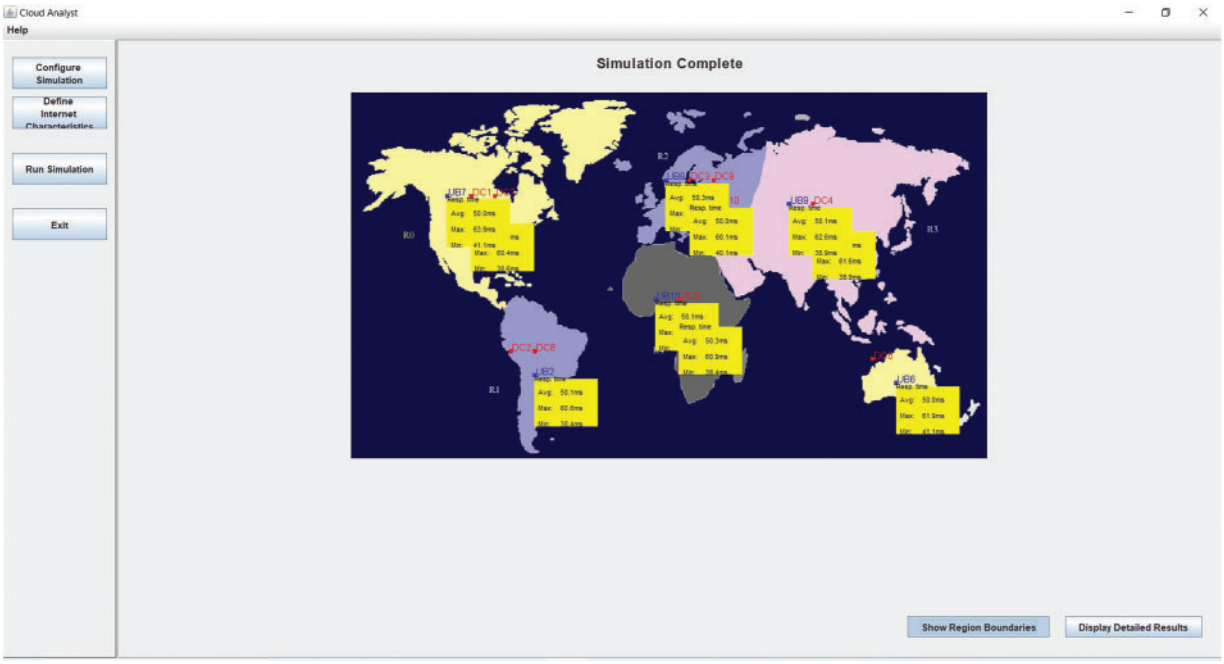

Figure 2: Simulation Environment

5 Results and Discussion

Performance of the cloud operations incorporates various layers, and discovering where bottlenecks are occurring in our applications is necessary. In some instances, the cloud service providers offer the least, a satisfactory level of performance [9]. This may not be sufficient if you need an improved performance for your running applications. So healthy competition between cloud service providers will occur. This pushes service providers to give more attention to the throughput in terms of the performance of running applications.

5.1 Average Time Interval for Response

Based on the simulation outcomes, it is worth noting that the projected load balancing approach produces a better throughput in terms of load balancing. Fig. 3 gives the Average Time Interval analysis for the response time of the proposed approach compared to other existing approaches [10]. In this, it is observed that the Average Time Interval for response and process time in the data center has been optimized and the values are represented in Table 3.

Table 3 clearly showing the Average Time Interval for the response of the round robin algorithm gives better results compared with the Max-Min algorithm excluding minimum response time. The overall response time of the existing Throttled load balancing algorithm gives feasible results compared with both approaches. And the proposed I-Throttled generates better results compared with all these approaches.

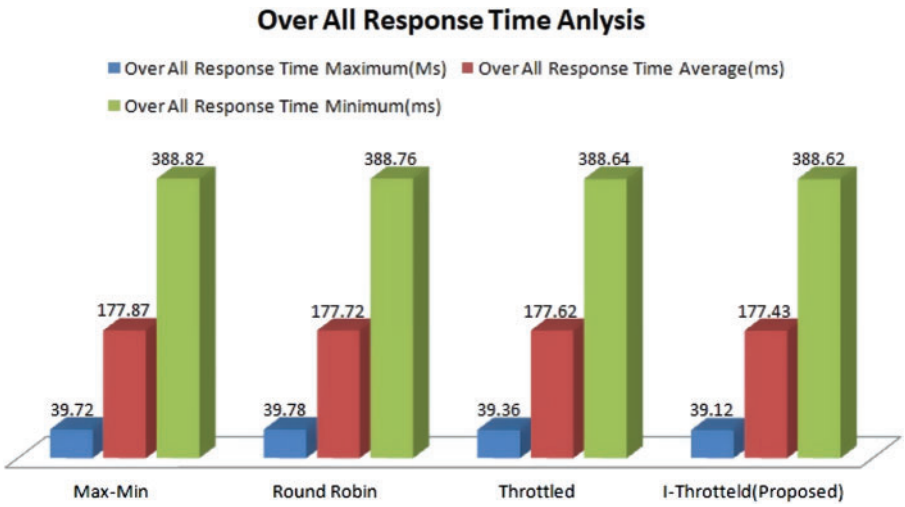

Figure 3: Average time interval for response

Table 3: Response time analysis

S No.	Algorithm/response time	Over all response time minimum (ms)	Over all response time average (ms)	Over all response time maximum (ms)
1	Max-Min	39.72	177.87	388.82
2	Round Robin	39.78	177.72	388.76
3	Throttled	39.36	177.62	388.64
4	**I-Throttled**	**39.12**	**177.43**	**388.62**

5.2 Resource Utilization Analysis

One more factor influencing efficiency in resource utilization. Resource utilization is, how efficiently the available resources are utilized with the existing workload. When the workload increase, resource utilization also gets increased.

But at the same time, that should not affect the quality of service and throughput of the system. To achieve the scenario, we must ensure the proposed system influences the performance without affecting the QoS [11]. Fig. 4 offers a comparison of resource utilization with various load balancing systems that are currently in use. From the results, the proposed approach has got optimized outcomes than existing load balancing approaches.

5.3 Task Waiting Time Analysis

Task waiting time is measured in this proposed approach, we got optimized wait time for the customers, who and all ready to afford the average price of the particular component which will increase user satisfaction [12]. That will improve the business and leads to more profit. Fig. 5 shows the comparative analysis of task waiting for time. It is demonstrated that the overall wait time of tasks is optimized. Overall wait time (TK_w) of the array of tasks could be measured as the time difference

between arrival (TK_{AR}) of the task and resource allocated time (TK_{AL}) of it. It can be represented using the formula (2),

$$TK_W = TK_{AL} - TK_{AR} \tag{2}$$

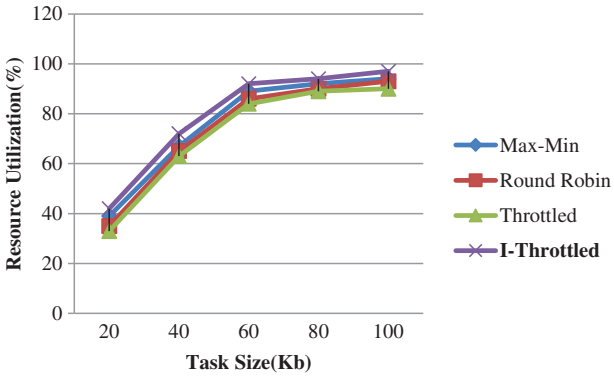

Figure 4: Resource utilization analysis

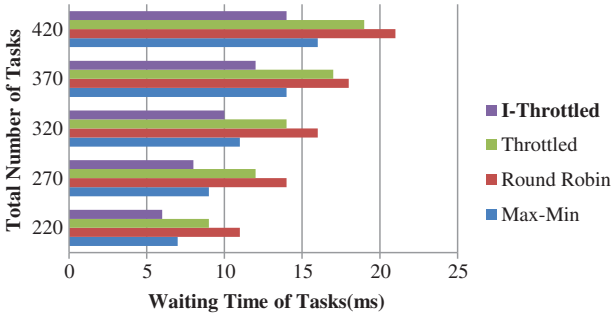

Figure 5: Waiting time analysis

5.4 Throughput Analysis

The number of completed tasks is playing a vital role in the analysis of a cloud system. Because, concerning the utilization of VMs, identified tasks are shifted from one VM to another [13]. During this process, some of the tasks are terminated and some of them are restarted. This affects the running time of the currently executing task. It will degrade the performance of the overall system.

And it is necessary to ensure that all arrived tasks are running until their completion. The task completeness is assured in our proposed algorithm by monitoring them from entry to end during the migration. The number of completed task discussion is given in Fig. 6. The results reveal that the proposed methodology has got the maximum throughput.

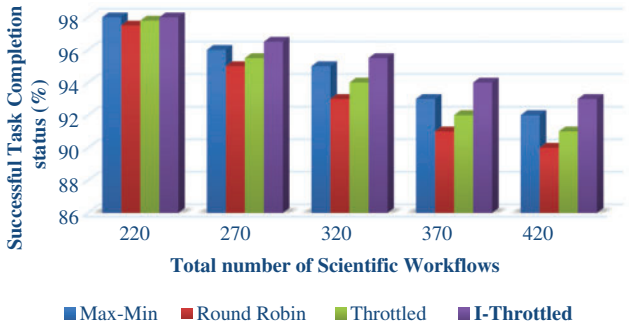

Figure 6: Throughput analysis

6 Conclusion

Even though there are various load balancing approaches available, there is difficulty with the consistency with which the workload is distributed. Because network traffic and cloud users continue to grow at an exponential rate. As a result, server-side fine-tuning in terms of service quality should be done frequently to guarantee that service level agreements are met. As a result, the suggested load balancing approach is derived from the current Throttled load-balancing approach for workload distribution optimization. Based on the simulation findings, it can be observed that the presented I-Throttled load balancing technique improves response time by 6%.

The research has useful directions for academia as well as for practitioners. The cloud resource scheduling concept can also be used along with the concepts and ideas discussed in the studies like [14–16].

Acknowledgement: The authors would like to thank every individual who has been a source of information, support, and encouragement on the successful completion of this manuscript.

Funding Statement: This paper was supported by the project: "Research and Implementation of Innovative Solutions for Monitoring Consumption in Technical Installations Using Artificial Intelligence", beneficiary S.C. REMONI TECHNOLOGIES RO S.R.L in partnership with "Gheorghe Asachi" Technical University of Iasi, Financing Contract No. 400/390076/26.11.2021, SMIS Code 121866, financed by POC/163/1/3.

Author Contributions: Data collection, drafting, and critical revision of the article: V. Dhilip Kumar and J. Praveenchandar, study conception and design: Muhammad Arif and Atif Ikram, analysis and interpretation of results: Adrian Brezulianu and Oana Geman. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: No new data were created or analysed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. L. Li, J. B. Du, D. S. Zhai, X. L. Chu and F. R. Yu, "Task offloading, load balancing, and resource allocation in MEC networks," *IET Communications*, vol. 14, no. 9, pp. 1451–1458, 2020.

[2] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.

[3] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu *et al.,* "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[4] J. Du, L. Zhao, J. Feng and X. Chu, "Computation offloading & resource allocation in mixed fog, cloud computing systems with Min-Max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[5] J. Praveenchandar and A. Tamilarasi, "Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 4, pp. 4147–4159, 2021.

[6] S. Deng, L. Huang, J. Taheri and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3327, 2015.

[7] P. Hofmann and D. Woods, "Cloud computing: The limits of public clouds for business applications," *IEEE Internet Computing*, vol. 14, no. 6, pp. 90–93, 2010.

[8] J. Praveenchandar and A. Tamilarasi, "An optimized resource allocation by improved task scheduling algorithm in cloud environments," *Journal of Computational and Theoretical Nanoscience*, vol. 15, no. 8, pp. 2655–2658, 2018.

[9] S. K. Garg, R. Buyya and H. J. Siegel, "Time and cost trade-off management for scheduling parallel applications on utility grids," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1344–1355, 2010.

[10] M. F. Kacamarga, B. Pardamean and H. Wijaya, "Lightweight virtualization in cloud computing for research," in *Proc. of ICSIIT*, Berlin, Heidelberg, pp. 439–445, 2015.

[11] J. Praveenchandar and A. Tamilarasi, "The feasible job scheduling algorithm for efficient resource allocation process in cloud environment," in *Proc. of ICRTAC*, India, pp. 28–33, 2018.

[12] S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," *Future Generation Computer Systems*, vol. 52, pp. 1–12, 2015.

[13] D. Patel and A. S. Rajawat, "Efficient throttled load balancing algorithm in cloud environment," *International Journal of Modern Trends in Engineering and Research*, vol. 2, no. 3, pp. 463–480, 2015.

[14] A. Ikram, M. A. Jalil, A. B. Ngah and A. S. Khan, "Towards offshore software maintenance outsourcing process model," *International Journal of Computer Science and Network Security*, vol. 20, no. 4, pp. 6–14, 2020.

[15] A. Ikram, H. Riaz and A. S. Khan, "Eliciting theory of software maintenance outsourcing process: A systematic literature review," *International Journal of Computer Science and Network Security*, vol. 18, no. 4, pp. 132–143, 2018.

[16] A. Ikram, M. A. Jalil, A. B. Ngah, A. S. Khan and Y. Mahmood, "An empirical investigation of vendor readiness to assess offshore software maintenance outsourcing project," *International Journal of Computer Science and Network Security*, vol. 22, no. 3, pp. 229–235, 2022.