**A LOAD BALANCING ALGORITHM FOR THE DATA CENTRES TO OPTIMIZE CLOUD COMPUTING APPLICATIONS**

**ABSTRACT**

Despite the many past research conducted in the Cloud Computing field, some challenges still exist related to workload balancing in cloud-based applications and specifically in the infrastructure as service (IaaS) cloud model. Efficient allocation of tasks is a crucial process in cloud computing due to the restricted number of resources/virtual machines. IaaS is one of the models of this technology that handles the backend where servers, data centres, and virtual machines are managed. Cloud Service Providers should ensure high service delivery performance in such models, avoiding situations such as hosts being overloaded or underloaded as this will result in higher execution time or machine failure, etc. Task Scheduling highly contributes to load balancing, and scheduling tasks much adheres to the requirements of the Service Level Agreement (SLA), a document offered by cloud developers to users. Important SLA parameters such as Deadline are addressed in the LB algorithm. The proposed algorithm is aimed to optimize resources and improve Load Balancing in view of the Quality of Service (QoS) task parameters, the priority of VMs, and resource allocation. The proposed LB algorithm addresses the stated issues and the current research gap based on the literature’s findings. Results showed that the proposed LB algorithm results in an average of 78% resource utilization compared to the existing Dynamic LBA algorithm. It also achieves good performance in terms of less Execution time and Make span.

**EXISTING SYSTEM**

The authors have limited the scope in this research to emphasize enhancing the cloud's performance in terms of Task Scheduling and Load Balancing. Based on the literature discussed in section III, the authors concluded the following points to address the research issues that have been resolved in the proposed work:

* Most researchers do not consider the priority, which is a critical factor in Task Scheduling. This will lead to issues such as an increase in Makespan time, which is the time taken to schedule a task/request, or an increase in the number of task rejections and latency.
* Although Task Scheduling is one of the main goals of providing an efficient Load Balancing and improving performance, most researchers focus on one or two aspects. For example, to enhance Load Balancing and considers few Task Scheduling parameters. Thus, only a few metrics are taken into consideration to improve the overall performance. This is an issue as improper Task Scheduling leads to an imbalanced load in the hosts.
* For example, if tasks arrive simultaneously following the FCFS algorithm's procedure, this could highly increase Makespan as the task will wait longer to finish executing. Each client may also send a different request; this should be indicated by providing random values for Task Length to make up a dynamic workload.
* Several new approaches have been made to improve Load Balancing; however, the workload migration challenge is still not fully addressed. Tasks are still allocated to VM regardless of its SLA violation state, which indicates it doesn't follow the specified Deadline and requirements stated in the agreement document. Each client receives a different SLA contract based on their needs from CSPs; hence, assigning random values for the Deadline parameter is crucial in scheduling since it can illustrate the algorithm's violation problem.

**PROPOSED SYSTEM**

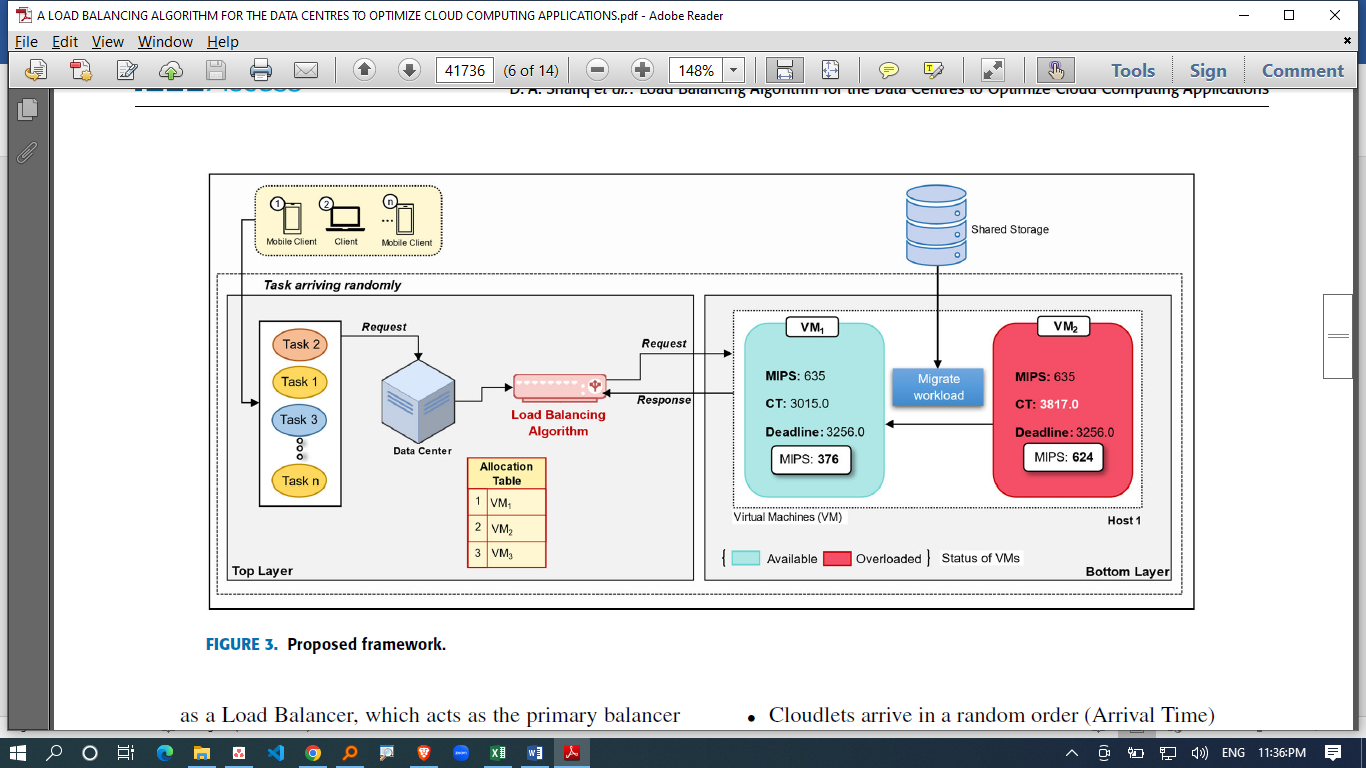
This section explains the proposed and improvised Load Balancing in Cloud Computing Environment. This algorithm's primary goal is to provide services of high quality to clients in Cloud Computing applications. The method consists of

both processes: Task Scheduling process to assign deadline and completion time to cloudlets (tasks) and secondly, Load Balancing process to perform migration of workload in case of VM violation to maintain a balanced load in the cloud environment.

This proposed model's main goal is to provide efficient resource allocation in a cloud environment whereby it avoids unbalanced workload in Cloud Computing applications. This model resolves issues related to workload migration and task rejection in the cloud. The proposed framework consists of two layers:

* Top Layer: deals with requests from multiple different clients (application's users) of both mobile and desktop. Clients can access the Internet using different devices to send requests to the cloud. In this layer, the model uses the Cloudlet Scheduler Time Shared algorithm to submit tasks in a random order (Arrival Time) and schedule them to Virtual Machines by considering two main parameters: Deadline and Completion Time. In Cloud Computing, Data Center (DC) can be described as big storage for cloud servers and data. DC receives requests and sends them to the active load balancer. In this layer of the model, the proposed algorithm is implemented as a Load Balancer, which acts as the primary balancer in the cloud environment to perform migration in the case of violation, which has not been addressed in the previous literature up to the author's knowledge.
* Bottom Layer: deals with allocation of user requests to Virtual Machines (VMs). As the figure illustrates, we have a primary batch of VMs; VM2's status is set to high priority since it violates the SLA requirement, which means its Completion Time is higher than the Deadline. Thus, the proposed LBA should apply a migration technique to transfer the workload to another available Virtual Machine by reconfiguring the MIPS of both VMs before and after allocating the resources to them. The allocation table is then updated whenever a Virtual Machine becomes violated or not, along with the number of requests it's been allocated. There is a case where there is no SLA violation. Suppose the Time to Complete (TTC) is less than SLA (Deadline) given for tasks to run on VMs. Then, no SLA violation occurs.

**SYSTEM ARCHITECTURE**



**HARDWARE REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| MINIMUM (Required for Execution) | | MY SYSTEM (Development) |
| System | Pentium IV 2.2 GHz | i3 Processor 5th Gen |
| Hard Disk | 20 Gb | 500 Gb |
| Ram | 1 Gb | 4 Gb |

**SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Operating System | Windows 10/11 |
| Development Software | Python 3.10 |
| Programming Language | Python |
| Domain | Cloud Computing |
| Integrated Development Environment (IDE) | Visual Studio Code |
| Front End Technologies | HTML5, CSS3, Java Script |
| Back End Technologies or Framework | Django |
| Database Language | SQL |
| Database (RDBMS) | MySQL |
| Database Software | WAMP or XAMPP Server |
| Web Server or Deployment Server | Django Application Development Server |
| Design/Modelling | Rational Rose |