Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Enhancing Load Balancing in Cloud Computing through Adaptive Task Prioritization

Hieu Le Ngoc[1] ✉ and Hung Tran Cong[2]
[1]*Ho Chi Minh City Open University, Ho Chi Minh City, Vietnam*
[2]*Posts and Telecommunication Institute of Technology, Ho Chi Minh City, Vietnam*
Corresponding Author: Hieu Le Ngoc, **E-mail**: hieu.ln@ou.edu.vn

| ABSTRACT
Cloud computing has become an increasingly popular platform for modern applications and daily life, and one of its greatest challenges is task scheduling and allocation. Numerous studies have shown that the performance of cloud computing systems relies heavily on arranging tasks in the execution stream on cloud hosts, which is managed by the cloud's load balancer. In this paper, we investigate task priority based on user behavior using request properties and propose an algorithm that utilizes machine learning techniques, namely k-NN and Regression, to classify task-based priorities of requests, facilitate proper allocation, and scheduling of tasks. We aim to enhance load balancing in the cloud by incorporating external factors of the load balancer. The proposed algorithm is experimentally tested on the CloudSim environment, demonstrating improved load balancer performance compared to other popular LB algorithms.

| KEYWORDS
Cloud Computing, Load Balancing, task-based Priority classification, ATPA

| ARTICLE INFORMATION
ACCEPTED: 20 April 2023 **PUBLISHED:** 29 April 2023 **DOI:** 10.32996/jcsts.2023.5.2.1

1. Introduction

Currently, cloud computing is widely used as a platform for various life and scientific applications. Cloud computing enables the sharing of a large number of computing resources such as processors, storage, data, information, and knowledge. However, task scheduling and allocation present a significant challenge in cloud computing. As a result, several studies have been conducted on this topic to enhance load balancing in cloud computing.

According to Iosup et al. (2011), Suakanto (2012), and Kumar et al. (2021), the performance of cloud computing systems depends heavily on task arrangement in the execution flow on hosts to optimize workflow efficiency. As demand for cloud computing continues to rise for business, applications, and personal purposes, the system load and performance are increasingly affected (Kumar et al., 2021). However, job scheduling algorithms can classify tasks to ensure efficient and fast processing, especially with the many challenges faced by cloud computing environments (Agarwal & Jain, 2014). Priority-based scheduling policies can also be used to control the work sequence of a computer system (Shafiq et al., 2022).

There are many load balancing methods and proposals available to improve load balancing in cloud computing (Fang et al., 2010), but they have limitations in some cases of the complex cloud environment. To address this issue, a new approach is proposed that focuses on users' behavior, specifically task priority generated by user requests or behavior, to allocate requests and improve load balancing. The proposed approach involves studying load balancing algorithms and parameter sets related to task priority and algorithms for task priority classification. Machine learning techniques, specifically Linear Regression and k-NN, are used to classify task priority and allocate requests to the corresponding virtual machine. This new proposal is named ATPA, using k-NN classification for Adaptive Task Priority Algorithm. The proposed algorithm is evaluated experimentally on the popular cloud computing simulation environment, CloudSim, and found to perform better than traditional scheduling algorithms.

This article makes several contributions to the study of load balancing in cloud computing. Firstly, it takes an external perspective to examine the factors that influence load balancing, specifically the behavior of cloud users and the priority of tasks. Secondly, the article proposes the use of Linear Regression to predict the resources needed for a given task, and then uses k-NN to classify the task based on the predicted figures. Finally, the article evaluates the proposed algorithm, ATPA, through experimentation and demonstrates that it outperforms existing popular algorithms in the field of load balancing.

To further elaborate on our proposal, we intend to present our paper in six sections. The first section would be an introduction to provide a brief overview of the topic. The following section will discuss the existing works in this field. In the third section, we will discuss the theoretical background of cloud computing and load balancing in cloud computing, including cloud users' behavior and tasks' priorities, as well as k-NN and linear regression of machine learning. The fourth section will present and describe our proposed algorithm, ATPA. In section 5, we will describe the simulation and experiment results, and the evaluations will be discussed. Finally, in section 6, we will give the general conclusions of the paper, including the implications of the research, and suggest future work that can be done in this area.

2. Literature Review

2.1 Cloud Computing and Load Balancing

According to the National Institute of Standards and Technology (NIST) (Mell & Grance, 2011), cloud computing is a service model that allows users to access shared computing resources on demand via a network connection, anytime, anywhere with ease. Daniele et al. (2012) suggest that the on-demand service delivery approach known as cloud computing is implemented using distributed computing and virtualization technology. Gartner.com (n.d.) explains that cloud computing is adaptable and scalable, with the delivery of technological resources as a service over the Internet being highly scalable. Forrester Research (Staten et al., 2009) states that the cloud computing platform offers three deployment methods as infrastructure as a service, each with distinctive qualities and economics that can help achieve deployment objectives for applications and services more effectively.

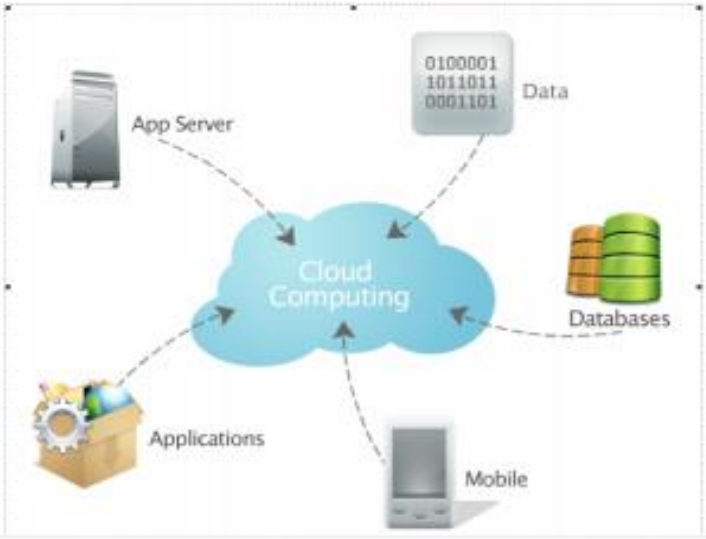

Figure 1. Cloud computing model (Source: 3STechBlog)

In simple terms, cloud computing is a model of providing IT resources over the internet as a service, allowing for easy modifications to suit user needs. The "cloud" refers to the collection of these resources, which are available for use without needing to know the technical details of how they are provided.

The cloud computing architectural model, as described by Mohammad, Qahtan, and Yahya (2016) and Javatpoint (2021), consists of three main components: the frontend, which is the client infrastructure; the internet, which provides the connection; and the backend, which is the grid computing system consisting of data centers that run applications, services, storage, virtualizations, cloud management, and cloud security. *The delivery service model* of cloud computing provides various layers of computing services, such as computing power on the grid, high-performance servers, virtual servers, storage space, operating systems, and development engines, among others. The cloud computing service models are classified into three basic groups, which include application development, application management, and services, as described by Mohammad et al. (2016), Javatpoint (2021), Hung and Phi (2016), and Kumar and Rathore (2018).

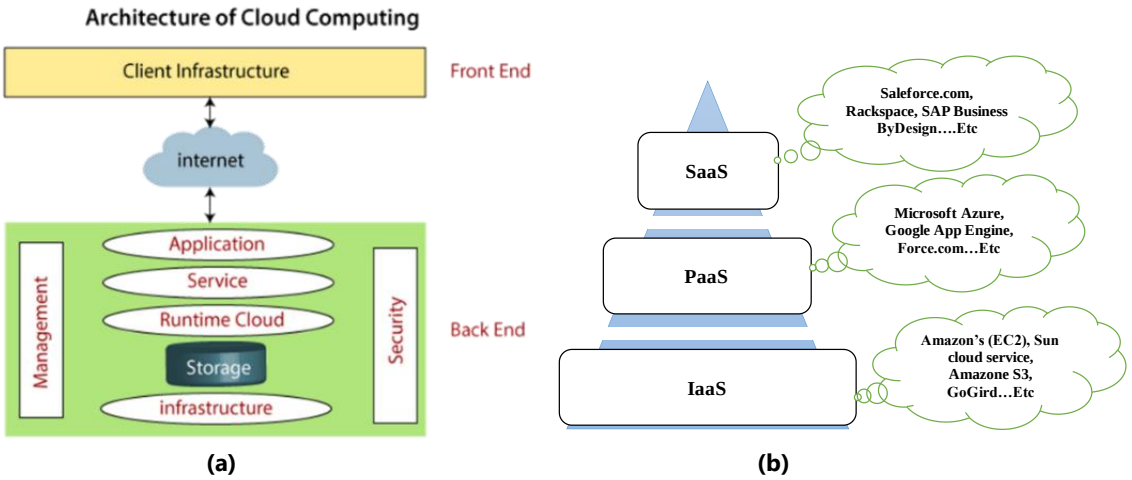

Figure 2. (a) Architectural model of cloud computing (JavaPoint, 2020), (b) Cloud computing service model (Hung T.C., 2016)

Load balancing is a strategy utilized in cloud computing to distribute traffic among multiple servers with the same function in the system, in order to minimize the load on one server while the others are idle. In the event that one server in the system becomes overloaded or malfunctions, the load balancer can allocate its tasks to other servers to maintain system performance and maximize response time. This approach is employed to improve the overall performance of the system. (NGINX, 2019; Asha et al., 2018).

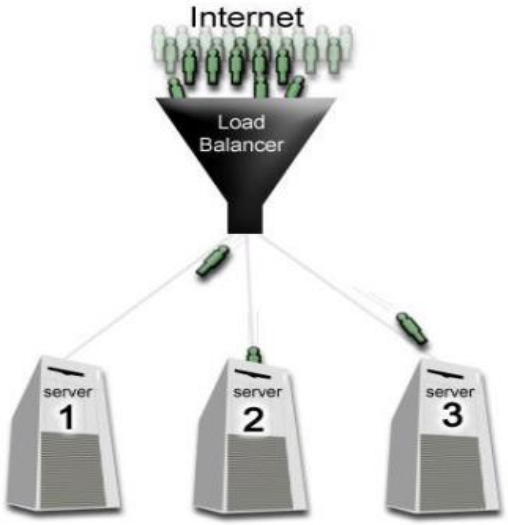

Figure 3. Load Balancing in Cloud Computing (NGINX, 2019)

The load balancer has a crucial role in cloud computing (NGINX, 2019; Asha et al., 2018; Taylor et al., 2019; Ghafir et al., 2021). Firstly, it helps in reducing network traffic to the website (NGINX, 2021). It can also act as a proxy or firewall at the application layer, responsible for load distribution after receiving requests (Asha et al., 2018). Secondly, the load balancer's primary function is to distribute traffic into separate requests and select which servers receive them (Taylor et al., 2019; Ghafir et al., 2021). There are different algorithms for separating traffic for each server. The servers must be kept available to maintain communication with the load balancers, which also helps to check if the server is still operational. Multiple failover scenarios are used to increase redundancy, and load sharing criteria include distributed content recognition through reading URLs, intercepting cookies, and compiling XML.

The benefits of load balancing are significant and can be outlined as follows. Firstly, it offers flexibility (Taylor et al., 2019) to the system by allowing servers to be added, removed, maintained, and repaired with minimal impact on the overall performance. This is achieved by using cookies, parsed URLs, and static/dynamic algorithms to control network traffic and improve load balancing performance. Secondly, load balancing provides high availability (Ghafir et al., 2021) by continuously monitoring server performance and automatically removing unresponsive servers from the system and adding them back when they are functional again. This is an automated process that requires no administrator intervention, ensuring redundancy of the load balancing system when any device fails. Thirdly, scalability is another advantage of load balancing where it distributes the load to multiple servers, reducing costs significantly by investing in many small servers instead of specialized equipment and large server systems.

Additionally, the system can easily change, increase, decrease, or replace servers without affecting system performance, keeping it available at all times.

2.2 Application of Machine Learning in Load Balancing

In the field of computer science, machine learning is a subset of artificial intelligence (AI) that enables computers to learn without explicit programming, as described by IBM Cloud Education (2020). Machine learning algorithms are classified into four groups based on their learning methods: supervised, unsupervised, semi-supervised, and reinforcement learning (IBM Cloud Education, 2020; NVIDIA, 2021). Resource usage prediction in cloud computing is a trending approach that can be achieved using machine learning, and several popular methods are available, including linear regression, regression trees, bagging using regression trees, and artificial neural networks (da Silva et al., 2015; Matsunaga & Fortes, 2010; Monge et al., 2015; Salzberg, 1994; Walczak & Cerpa, 2003). Different algorithms have different prediction accuracy, and no single algorithm can solve all problems, so it is essential to choose the algorithm that best suits the problem to be solved. In this paper, we apply linear regression techniques to historical data from virtual machines (VMs) that have processed previous requests to predict resource usage for the next request (da Silva et al., 2015; Witten et al., 2011).

Linear regression (Witten et al., 2011) is based on the assumption of a linear relationship between the input and output variables, such as resource usage and request parameters input or VM with runtime. The model uses a formula $y = \vec{a}.\vec{x} + b$ to represent the output based on the independent variable $\vec{x}$ and determines the values of $\vec{a}$ and b that minimize the error over a set of observed data. This approach is effective for predicting processing tasks where there is a linear relationship between input and output variables. However, if such linearity does not exist, the accuracy of the method may be unsatisfactory.

There are many metrics available to assess the accuracy of Linear Regression. These include Mean Absolute Error (MAE), Mean Bias Error (MBE), Relative Absolute Error (RAE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Relative Squared Error (RSE), Normalized Root Mean Squared Error (NRMSE), and Relative Root Mean Squared Error (RRMSE). In this study, the RAE metric was employed to evaluate the accuracy of the linear regression model. A smaller RAE value indicates better prediction accuracy. RAE is a metric that can take values between zero and one. A well-performing model would have RAE values closer to zero, and zero would be the optimal value. RAE represents the degree to which the mean residual error is proportional to the mean absolute deviation of the target function from its mean.

The K-nearest neighbor (KNN) algorithm (Ibm.com, 2020) is a non-parametric method that categorizes data points based on their proximity to other available data points. This method works on the assumption that related data points are found near each other. KNN assigns a category based on the most common category or average after determining the distance between data points, usually using Euclidean distance. KNN is popular among data scientists due to its simplicity and fast calculations. However, as the size of the test dataset grows, processing times become longer, making it less suitable for classification tasks. KNN is commonly used in image recognition and recommendation systems. There are several methods in KNN to measure distance, including Euclidean distance, Manhattan distance, Minkowski distance, and Hamming distance. The Euclidean distance (p=2), which is only applicable to real-valued vectors, is the most commonly used method.

The k-NN algorithm has been used in various applications, especially in classification tasks. It can also be used in Request classification to improve the performance of load balancers, not only for categorizing requests but also for meeting the real-time demands of cloud computing.

2.3 Previous studies

The study of clouds and load balancing in cloud computing has been a popular research topic in recent years. While there have been numerous studies and research conducted in this area, only a few approaches have considered the priority of tasks for cloud users and incorporated it as a parameter for load balancing. In this section, we will review the existing literature on this topic and discuss the strengths and weaknesses of each approach as well as the challenges associated with implementing them for optimizing application performance in cloud computing. We will present these works in a chronological sequence.

In 2010, Fang, Wang, and Ge proposed a two-level task scheduling mechanism for load balancing in cloud computing that schedules tasks from users' applications to virtual machines and from virtual machines to host resources to achieve efficient resource utilization. This mechanism has been shown to improve makespan and resource utilization in cloud environments, potentially enhancing the performance of cloud load balancers while maintaining stability (Fang et al., 2010). In 2012, Maheshwari and Bansal proposed a priority scheduler to improve throughput and reduce response time for efficient task scheduling in a grid environment by prioritizing tasks based on their required resources, with lower loading factor tasks assigned higher priority (Maheshwari & Bansal, 2012). In 2013, Rajguru and Apte analyzed priority with popular load balancing algorithms and proposed the multilevel feedback queue scheduling to prioritize load balancing parameters (Rajguru & Apte, 2013).

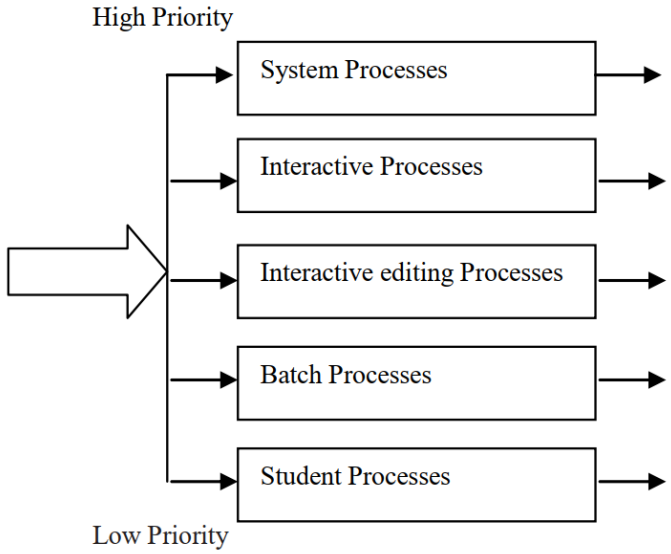

Figure 4. Multilevel queue scheduling with priority level (Rajguru et al., 2013)

Huankai Chen and colleagues (2013) proposed two algorithms that combined Min-Min with user-priority to allocate tasks with different priorities for different users. The Load Balance Improved Min-Min Scheduling Algorithm (LBIMM) and User-Priority Aware Load Balance Improved Min-Min Scheduling Algorithm (PA-LBIMM) were tested in three scenarios: low proportion of VIP tasks, a high proportion of VIP tasks, and different numbers of random tasks. The results showed improvements over the Min-Min algorithm based on Makespan, Average Resource Utilization Ratio (ARUR), and Average VIP Task Completion Time (AVIPCT). Shams Imam and Vivek Sarkar (2014) proposed a work-stealing scheduler approach based on Multi-Level Queue Scheduling (MLQS) for load balancing, utilizing lower priority tasks to reduce execution time. Their algorithm was compared to other schedulers in the JAVA standard library and demonstrated significant improvement. Although their focus was on load balancing, their approach could be applied to cloud environments. Sharma and colleagues (2018) proposed a task scheduler for load balancing in cloud computing based on task length, priority, and deadline. Their credit-based algorithm demonstrated better efficiency, but their method of calculating credit may not accurately reflect the relative importance of tasks, potentially leading to suboptimal scheduling decisions.

Velde and Rama (2019) proposed the User Priority based Scheduling for Load Balancing (UPS-LB) algorithm, which uses user-guided priorities to divide tasks into elastic and inelastic groups for scheduling in order to enhance load balancing. According to empirical experiments, UPS-LB performed comparably better than its predecessors such as Min-Min, LBIMM, and PALBIMM. However, the algorithm's effectiveness in real-world cloud environments may be limited as it only considers user-guided priorities for task scheduling and does not take other factors like resource availability and utilization into account. In the same year, a study (Al-Rahayfeh et al., 2019) proposed an approach that uses dominant sequence clustering (DSC) and mean shift clustering for task scheduling and load balancing. The proposed algorithm clusters users' tasks using DSC, ranks tasks using MHEFT, clusters virtual machines using MSC, and performs load balancing using a WLC algorithm. The proposed algorithm is evaluated using response time, makespan, resource utilization, and service reliability metrics. Although this approach can increase the response time, the clustering process may add overhead and complexity to the scheduling and load balancing process. Additionally, the performance of the algorithm may depend on the choice of clustering algorithms and parameters and may not be suitable for all types of workloads or cloud environments.

Halim and Hajamydeen (2021) proposed a novel approach for task scheduling and load balancing in cloud computing by utilizing task grouping. Their approach aims to optimize work scheduling dynamically using a weighted fair queuing model, which is more efficient than the current approach. The experimental results showed that their proposed algorithm outperformed the current algorithm in terms of various execution parameters such as turnaround time, task size, and average waiting time. Specifically, the round-robin and shortest job first algorithms performed better than the current algorithm.

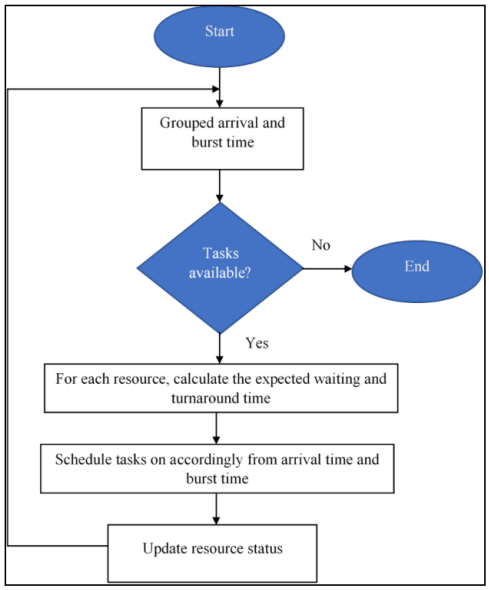

Figure 5. Mutation mechanism for generating new population (Halim and Hajamydeen, 2021)

Kim et al. (2022) proposed privacy-preserving kNN query processing algorithms for encrypted databases in cloud computing using secure two-party computation and garbled circuits [20]. The algorithms not only protect both data and query privacy, but also improve query processing efficiency. The study found that kNN is useful and easy to apply in cloud computing and outperforms existing algorithms in terms of query processing cost. In the same year, Sansanwal and Jain (2022) conducted a survey of load balancing (LB) algorithms on the cloud and found that besides static LB algorithms, there are several new algorithms such as Particle Swarm Optimization Algorithms, Artificial Bee Colony/Honey bee Swarm Algorithms, Grey Wolf Optimization (GWO) Algorithms, Genetic Algorithms (GA) that are developed within the characteristics of previous studies and enhanced them. However, not many of them are combined and integrated with Machine Learning or Datamining techniques. A recent study (Katal et al., 2022) proposed a detailed review of all techniques related to energy efficiency in cloud computing data centers and revealed problem-solving approaches such as load balancing, workload categorization and prediction, etc. The authors found that some load balancers use Machine learning and kNN, while workload categorization and prediction are variant and use many of the ML techniques such as Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), Logistic Regression (LR), Random Forest (RF), Multi-Layer Perceptron (MLP), Backpropagation neural network.

After examining the previous research on task priority and task allocation in cloud computing, we have gained a deeper understanding of these concepts. Identifying a gap in the existing literature, we propose a prediction approach that can anticipate the resources required for upcoming tasks on virtual machines (VMs). We integrate this prediction method into our load balancing algorithm to ensure that requests are allocated to the appropriate VMs, resulting in improved performance. Our approach relies on k-NN and Linear Regression techniques. Moreover, our method is fully automated, eliminating the need for expert analysis.

3. Methodology

3.1Task-based priority

According to A. Iosup et al. (2011), a task refers to a process or multiple processes that are executed on a compute node, which exists within a virtual machine in cloud computing. The priority of tasks is crucial in determining the scheduling of tasks since it greatly impacts the quality of service provided by the service provider (A. Iosup et al., 2011 ; A. Agarwal, and S. Jain, 2014). The task's processing depends on several factors such as CPU usage, RAM, bandwidth, length, and even the task's size and completion time, which are used to determine the priority of tasks. This paper focuses on these factors to calculate the priority of tasks performed by the cloud. We propose a task-based priority approach that considers higher power consumption, greater CPU usage, and more RAM usage or higher cost, to give priority to tasks. The task-based priority is represented as a 3-dimensional vector synthesized from Power consumed, CPU usage, and RAM usage.

$$Priority = \{Po, CPU, RAM\} \quad (1)$$

3.1.1 Classify request with with task-based priority

Based on the historical dataset of processing the previous tasks, we use k-NN to prioritize the coming request or the next tasks. Corresponding to each request (Request) there will be tasks or jobs that the computer needs to perform to serve that request. Therefore, the request sent to the cloud can be classified based on its respective tasks.

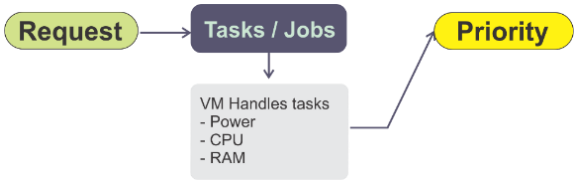

Figure 6. Calculating the priority of Cloud requests

3.2 Research model

In the current research, the kNN (k Nearest Neighborhood) classification algorithm is employed to classify requests based on task-based priority, which is determined by a task's energy consumption (Power consumed), CPU usage (CPU Usages), and RAM usage (RAM Usages) in cloud execution. Once the jobs/tasks are prioritized, the load balancer will assign requests with higher task-based priority to VMs/hosts with better processing capacity, i.e., more available space for high-demand tasks. The aim is to allocate the request requiring the most processing to the virtual machine/host with the lowest activity level (the least busy). This proposed approach aims to improve load balancing processing time in real-time cloud applications. The algorithm is called ATPA (k-NN classification for Adaptive Task-based Priority Algorithm).

The main goal of this research model is to minimize the risks associated with the server system and reduce the lifetime of requests in the cloud. Another objective is to prevent load imbalance between virtual machines by efficiently classifying tasks based on their priority level, and ensuring that resources are being used effectively to provide the best possible user experience. This is achieved by classifying incoming requests based on their priority level and allocating them to virtual machines or hosts that are capable of handling the corresponding load. To allocate tasks in a reasonable way, the system sorts virtual machines, hosts, and resources based on their usage levels, from high to low. By accomplishing these objectives, the system can resolve requests faster and provide better performance for users.

3.3 ATPA workflow

The proposed algorithm is designed to process requests and allocate them to suitable virtual machines to achieve load balancing. To achieve this, the algorithm uses Regression to predict the cloud resource usage required to handle the task associated with the request, based on the properties of the request, such as Power, CPU Usage, and RAM Usage. The Regression technique relies on the dataset containing the cloud resource usage of previous tasks, using the most recent data available. The predicted data is then used in conjunction with the k-NN algorithm, which classifies the task-based priority of the task/job. The k-NN algorithm's dataset is always up to date with the latest resource information. Based on the task-based priority, the algorithm allocates the handling request to the appropriate virtual machines, thereby preventing load imbalance. The algorithm's goal is to simulate the algorithm naturally and plan for the next requests to avoid load imbalance. The proposed algorithm reduces the communication load between virtual machines and existing resources, decreasing unnecessary bandwidth and throughput and increasing service for user requirements.

4. Results and Discussion

This section is a comparative or descriptive analysis of the study based on the study results, previously literature, etc. The results should be offered in a logical sequence, given the most important findings first and addressing the stated objectives. The author should deal only with new or important aspects of the results obtained. The relevance of the findings in the context of existing literature or contemporary practice should be addressed.

The proposed ATPA algorithm, short for Adaptive Task-based Priority Algorithm, considers various factors, including task priority, to classify incoming requests and allocate resources in the most efficient manner. The kNN algorithm is used for this classification, and resources such as virtual machines/hosts are sorted by increasing usage. Additionally, the algorithm is improved by incorporating machine learning techniques to evaluate errors, although this is less likely due to the allowed error. This paper presents the ATPA algorithm, which comprises three main modules.

(1) Module to calculate the request's parameters by the Regression algorithm:

In this module, the Regression algorithm will rely on the properties of the request to calculate the resource usage of the Task/job corresponding to that request. The attributes of the request include: Size, Response Length, Max Length, File Length, etc.

$$Po_{New} = Regression(Request, Power) \quad (2.1)$$

$$CPU_{New} = Regression(Request, CPU) \quad (2.2)$$

$$RAM_{New} = Regression(Request, RAM) \quad (2.3)$$

Where

Request = { X1, X2,..., Xn }, where *Xi* are the properties of Request;

Po_{New} is predicted power consumption.
Power is the dataset of power consumption recorded in the past.
CPU_{New} is predicted CPU usage.
CPU is the dataset of CPU usage recorded in the past
RAM_{New} is predicted RAM usage
RAM is the dataset of RAM usage recorded in the past
Here, a vector of 3 factors {Po, CPU, RAM} can be used to sum up the calculation, or calculate each quantity separately.

(2) Module to classify requests by task-based priority:
In this module, we will use the k-NN classification algorithm (with k= 5) to classify the request in question, based on the nature of the priority of the tasks. This k-NN classifier model uses the requests' data which have been processed in the past, and corresponding labels from 1 to 5 of the priority level. Level 1 is the lowest priority; level 5 is the highest priority. Based on this model, we can classify the Request being processed and determine the corresponding label (from 1 to 5). Then we choose the virtual machine with the corresponding order 1 to 5. This order is sorted based on the idle or low load level of the virtual machine, i.e. level 1 is the most loaded machine, and level 5 is the least loaded machine.

$$VM_{select} = k\text{-}NN(Po, CPU, RAM) \quad (3)$$

Where:
VM_{select} is the selected virtual machine
k-NN is a classification function from the KNN classifier model that has been built based on the past data set of requests
Po is the predicted Power calculated from Module 1
CPU is the predicted CPU usage calculated from Module 1
RAM is the predicted RAM usage calculated from Module 1

(3) Module is for allocating services (select virtual machine)
This module is responsible for allocating requests to virtual machines through the appropriate request label and virtual machine. If a request is sent, the request is classified by module 2, and the VMs under consideration including the unloaded VMs are also clustered according to module 2. Here, virtual machine found from Module 2, will be assign to process the coming request with the right label. After processing that request, all results and resources info of the request will be saved in the dataset for updating the most recent requests processed. This is the dataset for the construction process of KNN classifier model in Module 2.

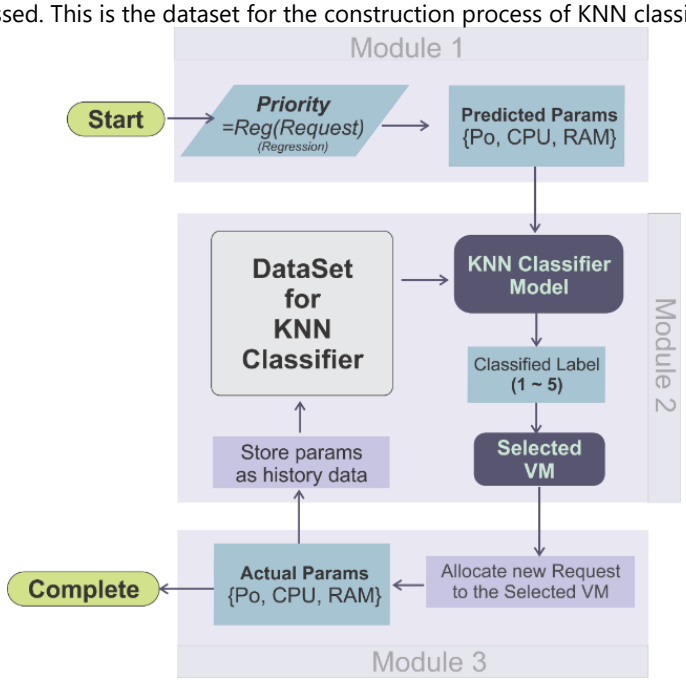

Figure 7. Diagram of Research Model of ATPA

In the above pseudocode of the ATPA, the algorithm will use a loop to listen to all the Requests in the queue of Requests sent to the load balancer (CloudRequests). When this list is empty, there will be no request distributed anymore. In it, the algorithm uses the isLocated variable (logical type) as a flag to mark that the Request whether has been allocated.

Pseudocode of ATPA
Input: the set of coming requests
Output: the allocation of each request

```
1.    For each Request in CloudRequests
2.      isLocated = false;
3.      Priority = {Po, CPU, RAM}new = Regression(T1,T2..... Tn); // Module 1
4.      Request.PriorClass = k-NN(Priority); //k-NN is the task classification model
5.      For each VM in VMList
6.        If isFitSituation(Request.PriorClass, VM)
7.          AllocateRequestToVM(VM, Request); // Module 3
8.          isLocated = true;
9.        End If
10.     End For
11.     If(!isLocated)
12.       VM = VMList.getSelectedVM(); // Module 2
13.       AllocateRequestToVM(VM, Request);
14.     End If
15.   End For
```

At the first loop, the *isLocated* variable is defaulted to false. Then, the algorithm calculates the Priority vector with 3 dimensions: *PowerConsume*, *CPU Usage* and *RAM Usage* (Priority = {Po, CPU, RAM}) needed to perform the Request under consideration. This calculation is based on the historical data of previous requests T_1, T_2,... T_n , where *n* is the number of requests that have been stored in the dataset. T_i is the parameters of the i^{th} Request stored, T_i includes the inputs of *MaxLength, FileSize, OutputSize, etc.*; and the processing resource results performed by Cloud to process the i^{th} Request include *PowerConsume*, *CPU Usage* and *RAM Usage*. This historical *n* Request data will build the Regression function (linear regression) to predict and calculate the Priority vector for the Request. This Priority data is used for k-NN model to classify the Request, and a label is assigned to the *PriorClass* property of the current Request. After receiving the Priority class-label of the Request, the algorithm loops through the virtual machines available in the cloud. Corresponding to each machine, the algorithm considers whether the virtual machine matches the priority of the current Request, through the *isFitSituation(Request.PriorClass, VM)* function. If it is satisfied, it will allocate the request to that virtual machine *AllocateRequestToVM(VM, Request),* and at the same time assign the variable *isLoacated = true*. If no matching virtual machine can be found, the loop ends. At this point, run the loop and the isLocated variable is still false, and now the Request has not been allocated. Therefore, the algorithm allocates this Request to the first VM of the VM list through the *VM = VMList.getSelectedVM()* statement. This allocation ensures that if any request is forecasted not in the data of the algorithm, it is still allocated and processed for the user.

5. Simulation and Experiments

The simulation of the cloud environment in this study is carried out using the CloudSim library (Calheiros, R. N et al., 2011), which is a JAVA programming language-based tool. Additionally, the Weka library (waikato.ac.nz, 2012) is integrated into the simulation environment to use its pre-built LinearRegression function and KNN model. The proposed ATPA algorithm achieves the desired objectives, such as minimizing queued requests and enhancing the processing and response times of the cloud. This indicates that the performance of cloud computing is better with the proposed algorithm when compared to other algorithms such as MaxMin, Round Robin, MinMin, and FCFS.

5.1 Simulation environment

Cloud environment includes 1 Datacenter with 5 hosts running 5 virtual machines, and we create random re-quests with different to experiment the proposal ATPA.

Table 1. Datacenter configuration parameters

Datacenter	***Host in Datacenter***
- Number of hosts (hosts) in datacenter: 5 - Do not use Storage (SAN drives) - Architecture (arch): x86 - Operating system (OS): Linux - Processing (VMM): Xen - Time Zone: +7 GMT - Cost: 3.0 - Cost per Memory: 0.05 - Cost per Storage: 0.1 - Cost per Bandwidth: 0.1	Each host in the Datacenter has the following configuration: - CPU has 4 cores, each core has a processing speed of 1000 (mips)-Ram: 16384 (MB) - Storage: 1000000 - Bandwidth: 10000

Table 2. Virtual Machine Configuration

Size	**Ram**	**Mips**	**Bandwidth**	**PES No.**	**VMM**
10000 MB	512 MB	250	1000	1	Xen

Table 3. Request Variant

Length	**File Size**	**Output Size**	**PEs**
3000 ~ 1700	5000 ~ 45000	450 ~ 750	1

5.1.1 Evaluation criteria:

Experiment on simulating the cloud with the above parameters and run CloudSim's load balancing algorithm available: Round Robin, MaxMin, MinMin and FCFS, install the same input and compare the outputs, especially the parameters of the Execution Time (average, maximum and minimum).

The predicted Execution time of the virtual machines as well as the predictive Execution time of Cloud with less error, the better the effectiveness of the evaluated algorithm, the lower the cost as well, so that is the better technique. We also use the RAE (Relative Absolute Er-ror) to observe and evaluation the accuracy of Linear Regression Model.

5.2 Experimental Results

We conduct the simulation experiments with 04 cases of different inputs, the number of requests is 30, 60, 100 and 1000. The data for generating request we use Sipht which is proposed by https://github.com/WorkflowSim/WorkflowSim-1.0.

In ***case 1***, we consider with 30 requests, we have the execution time as in Table 4.

Table 4. Simulation experimental results of case 1.

Execution Time (ms)	**FCFS**	**ATPA**	**MaxMin**	**MinMin**	**Round Robin**
AVG	306.69	203.39	246.42	810.12	340.91
MAX	5,009.65	2,694.31	2,713.20	13,719.94	6,659.64
MIN	0.24	0.11	0.12	0.15	0.11

In case 1, experimental results with 30 requests, we see that Round-Robin algorithm is dominant and fast processing, MaxMin algorithm is also quite stable. FCFS algorithm is not yet strong. However, the ATPA recommendation algorithm is also quite stable, and proves to be more stable and better when handling more requests.

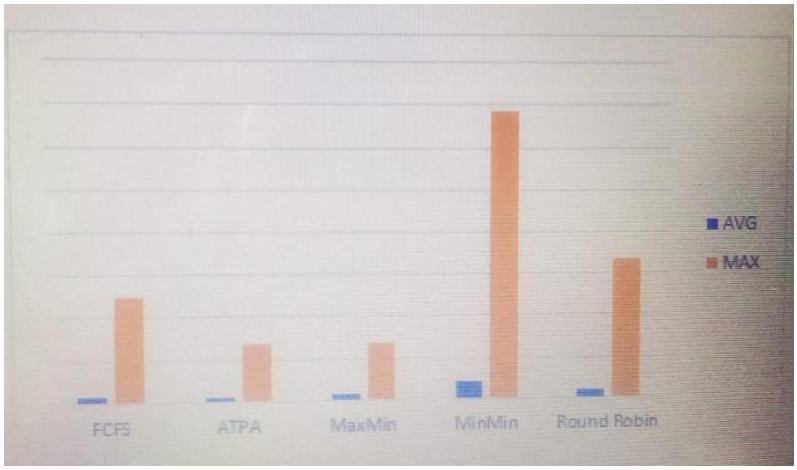

Figure 8. The execution time of 5 algorithms of case 1

In ***case 2***, we process with 60 requests, we have the execution time as in Table 5.

Table 5. Simulation experimental results of case 2.

Execution Time (ms)	FCFS	ATPA	MaxMin	MinMin	Round Robin
AVG	268.58	249.45	364.92	986.94	369.41
MAX	3,599.42	7,311.10	3,491.21	14,318.26	10,084.81
MIN	0.19	0.35	0.13	0.15	0.16

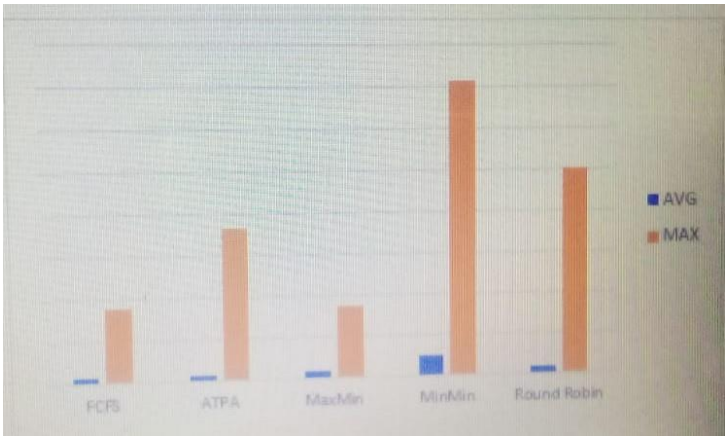

Figure 9. The execution time of 5 algorithms of case 2

In ***case 3,*** we process 100 requests, we have the execution time as in Table 7. From the 100th request onwards, the ATPA algorithm is significantly superior to MaxMin and MinMin. However, there is still no advantage compared to RoundRobin. But with the larger number of requests, ATPA is more advantageous. And gradually prevail over the rest of the algorithms. FCFS clearly shows the lack of intelligence and naturalness of the algorithm.

Table 6. Simulation experimental results of case 3.

Execution Time (ms)	FCFS	ATPA	MaxMin	MinMin	Round Robin
AVG	474.86	346.66	351.66	386.87	424.85
MAX	25,252.80	5,475.33	8,010.48	8,522.92	9,462.06
MIN	0.11	0.20	0.13	0.15	0.17

Testing with 30 to 100 requests, we see that the ATPA algorithm is superior to MaxMin, MinMin. But with the larger the number of requests, ATPA is more advantageous. And gradually prevail over the rest of the algorithms. FCFS clearly shows the lack of intelligence and naturalness of the algorithm. That's why we increase to 1000 requests.

Figure 10. The execution time of 5 algorithms of case 3

In ***case 4,*** we process with 1000 requests, we have the execution time as in Table 7.

Table 7. Simulation experimental results of case 4.

Execution Time (ms)	FCFS	ATPA	MaxMin	MinMin	Round Robin
AVG	360.43	192.82	391.75	230.98	378.06
MAX	9,829.39	3,755.07	17,947.42	9,433.95	20,515.09
MIN	0.12	0.11	0.15	0.10	0.12

In the case of 1000 Request, we see that ATPA is superior to other algorithms, far ahead of other algorithms.

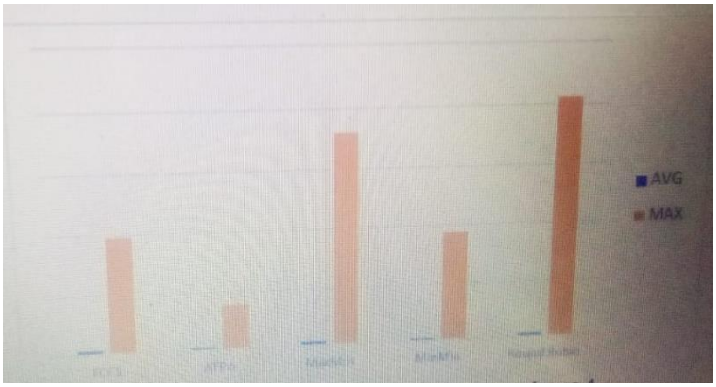

Figure 11. The execution time of 5 algorithms of case 4

Figure 12. Average execution time of 5 algorithms in 4 cases

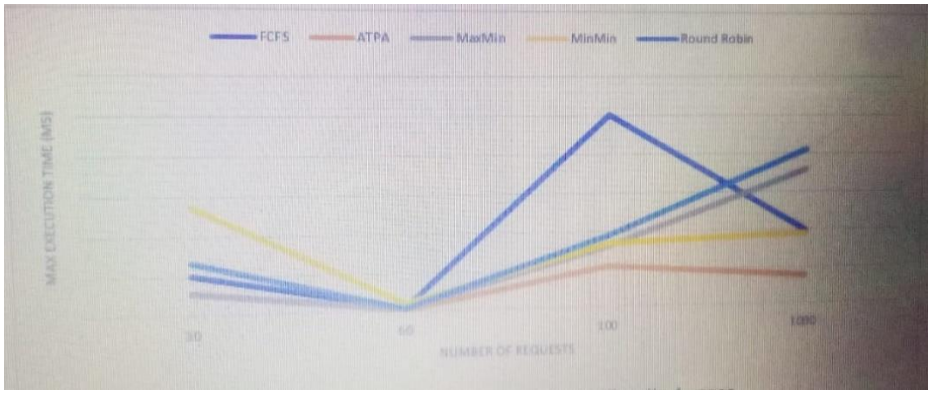

Figure 13. Maximum execution time of 5 algorithms in 4 cases

Through 04 cases of 30, 60, 100 and 1000, comparing the processing time of the algorithms with the same conditions, we can see the stable and reasonable distribution of the proposed ATPA algorithm. The processing time of virtual machines is not too different from the processing time of other algorithms on the cloud (in the case of few and many requests). Figures 17 and figure 18 show that ATPA is always lowest, both mean and max.

5.3 Evaluation Linear Regression Model in ATPA

To evaluate the accuracy of the Linear Regression Model used in ATPA, we use Relative Absolute Error metric (RAE) to see how the model run and give out the exact predicted value for the load balancer. The table 8 shows that the worst RAE happens in RAM usage prediction in case 1, and the best one is in case 3 but it is Power Consumption prediction. We can see that, the RAE are acceptable for this experiment but it is not good at all case due to the variant of the request.

Table 8. Comparing RAE of 4 Cases

	RAE metrics			
	Case 1 (30 requests)	Case 2 (60 requests)	Case 3 (100 requests)	Case 4 (1000 requests)
Power Consume	0.085039	0.095735	0.075476	0.084136
CPU Usage	0.295400	0.284387	0.236654	0.248776
RAM Usage	0.326888	0.292676	0.295666	0.300793

6. Conclusion

This paper focuses on improving load balancing in cloud computing environments through task-based priority and request classification using the kNN algorithm. Through a review of existing algorithms and previous studies, the paper proposes the ATPA algorithm to enhance load balancing on the cloud. The experimental results of the ATPA algorithm show positive results and potential implications for improving the balancing load efficiency, outperforming popular algorithms such as Round Robin, MaxMin, MinMin, and FCFS. However, the study has limitations in terms of not being applied in the physical cloud and the need for further consideration of response time, processing time, and the number and variant of requests. The paper suggests future

works to improve the study by applying more prediction techniques and optimization for the proposed algorithm in practical applications. Overall, this study highlights the potential of using AI, ML, and big data to generate better workload balancing in cloud computing environments.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
ORCID iD: Hieu Le Ngoc - https://orcid.org/0000-0002-1133-1433

References

[1] Agarwal, A., & Jain, S. (2014). Efficient optimal algorithm of task scheduling in cloud computing environment. *International Journal of Computer Trends and Technology (IJCTT)*, 9(7), 344-349. https://doi.org/10.14445/22312803/IJCTT-V9P163
[2] Al-Rahayfeh, A., Atiewi, S., Abuhussein, A., & Almiani, M. (2019). Novel approach to task scheduling and load balancing using the dominant sequence clustering and mean shift clustering algorithms. *Future internet*, 11(5), 109. DOI: 10.3390/fi11050109.
[3] Asha, V., Kumar, B., & Girish, V. (2018). Load Balancing in Cloud Computing. *IJRTER*, 4(3), 118-125. https://doi.org/10.23883/IJRTER.2018.4105.VQPYQ
[4] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software Practice and Experience (SPE),* 41(1), 23-50. doi: 10.1002/spe.995.
[5] Chen, H., Wang, F., Helian, N., & Akanmu, G. (2013). User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. *In 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH)* (pp. 1-8). IEEE. DOI: 10.1109/ParCompTech.2013.6621389
[6] da Silva, R. F., Juve, G., Rynge, M., Deelman, E., & Livny, M. (2015). Online task resource consumption prediction for scientific workflows. *Parallel Processing Letters*, 25(03), 1541003.
[7] Daniele, C., Giles, H., & Thomas, H. L. D. (2012). Cloud computing benefits, risks and recommendations for information security. *The European Network and Information Security Agency* (ENISA).
[8] Fang, Y., Wang, F., & Ge, J. (2010). A task scheduling algorithm based on load balancing in cloud computing. In Web Information Systems and Mining (pp. 271-277). *Springer Berlin Heidelberg*. DOI: 10.1007/'978-3-642-16515-3_34
[9] Gartner. (2020). Definition of Cloud Computing. Retrieved from https://www.gartner.com/en/glossary/. (Accessed on April 4, 2021).
[10] Hung, T. C., & Phi, N. X. (2016). Study the Effect of Parameters to Load Balancing in Cloud Computing. *International Journal of Computer Networks & Communications (IJCNC)*, 8(3), 33–45. https://doi.org/10.5121/ijcnc.2016.8303
[11] IBM Cloud Education. (2020, July 15). What is machine learning? Retrieved May 1, 2022, from https://www.ibm.com/cloud/learn/machine-learning
[12] Imam, S., & Sarkar, V. (2015). Load balancing prioritized tasks via work-stealing. *In Lecture Notes in Computer Science* (pp. 222-234). Springer. DOI: 10.1007/978-3-662-48096-0_18
[13] Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., & Epema, D. H. J. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6), 931-945. https://doi.org/10.1109/TPDS.2011.66
[14] Javatpoint. (2021). Cloud Computing Architecture. https://javatpoint.com/cloud-computing-architecture/
[15] Katal, A., Dahiya, S., & Choudhury, T. (2022). Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing*, 1–31.
[16] Kim, H.-J., Lee, H., Kim, Y.-K., & Chang, J.-W. (2022). Privacy-preserving kNN query processing algorithms via secure two-party computation over encrypted database in cloud computing. *Journal of Supercomputing*, 78(7), 9245–9284.
[17] Kumar, R., Soodan, B. S., Kuaban, G. S., Czekalski, P., & Sharma, S. (2021). Performance analysis of a cloud computing system using queuing model with correlated task reneging. *Journal of Physics: Conference Series*, 2091(1), 012003. https://doi.org/10.1088/1742-6596/2091/1/012003
[18] Kumar, V., & Rathore, R. S. (2018). Security Issues with Virtualization in Cloud Computing. *In 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)* (pp. 487–491). Greater Noida (UP), India. https://doi.org/10.1109/ICACCCN.2018.8748405
[19] Maheshwari, M. K., & Bansal, A. (2012). Process Resource Allocation in Grid Computing using Priority Scheduler. *International Journal of Computer Applications*, 46(11), 20-23.
[20] Matsunaga, A., & Fortes, J. A. B. (2010, May). On the use of machine learning to predict the time and resources consumed by applications. *In 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (pp. 495-504). IEEE.
[21] Mell, P. & Grance, T. (2011). The NIST definition of cloud computing. *NIST Special Publication* 800-145. National Institute of Standards and Technology.
[22] Mohammad, U., Qahtan, M., & Yahya, K. T. (2016). Cloud computing service models: A comparative study. *In Proceedings of the International Conference on Computing for Sustainable Global Development* (INDIACom) (pp. 890-895).
[23] Monge, D. A., Holec, M., Železný, F., & Garino, C. G. (2015). Ensemble learning of runtime prediction models for gene-expression analysis workflows. *Cluster Computing*, 18(4), 1317-1329.
[24] NGINX. (2019). What Is Load Balancing? Retrieved April 4, 2021, from https://www.nginx.com/resources/glossary/load-balancing/

[25] NVIDIA. (2021). What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning? Retrieved from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/
[26] Rajguru, A., & Apte, S. (2013). A performance analysis of task scheduling algorithms using qualitative parameters. *International Journal of Computer Applications*, 74(19), 33-38. https://doi.org/10.5120/13004-0308
[27] Suakanto, S. (2012). Performance measurement of cloud computing services. *International Journal on Cloud Computing Services and Architecture*, 2(2), 9-20. https://doi.org/10.5121/ijccsa.2012.2202
[28] Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2022). Load balancing techniques in cloud computing environment: A review. *Journal of King Saud University - Computer and Information Sciences*, 34(7), 3910-3933. https://doi.org/10.1016/j.jksuci.2021.02.007
[29] Staten, J., Yates, S., Rymer, J. R., & Nelson, L. E. (2009). Which cloud computing platform is right for you? *Forrester Research*, Inc.
[30] Salzberg, S. L. (1994). *C4.5: Programs for machine learning* by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Machine Learning, 16(3), 235-240.
[31] Sharma, A., Gupta, A. K., & Goyal, D. (2018). An optimized task scheduling in cloud computing using priority. *SSRN Electronic Journal.* DOI: 10.2139/ssrn.3166077
[32] Sansanwal, S., & Jain, N. (2022). An improved approach for load balancing among virtual machines in cloud environment. *Procedia Computer Science*, 215, 556–566.
[33] Velde, V., & Rama, B. R. (2019). A framework for user priority guidance based scheduling for load balancing in cloud computing. *International Journal of Simulation Systems Science & Technology*. DOI: 10.5013/ijssst.a.19.06.24.
[34] Walczak, S., & Cerpa, N. (2003). Artificial Neural Networks. *In Encyclopedia of Physical Science and Technology* (pp. 631-645). Elsevier.
[35] Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann Publishers Inc.
[36] Weka Development Team. (2012). Weka 3 - data mining with open source machine learning software in java. Retrieved April 27, 2022, from https://www.cs.waikato.ac.nz/ml/weka/.