Exploring Support Vector Machine Learning for Cloud Computing Workload Prediction

OMAR ALOUFI[1†]

Information Systems department, College of Computer Science and Engineering. Taibah University, Saudi Arabia

Summary
Cloud computing has been one of the most critical technology in the last few decades. It has been invented for several purposes as an example meeting the user requirements and is to satisfy the needs of the user in simple ways. Since cloud computing has been invented, it had followed the traditional approaches in elasticity, which is the key characteristic of cloud computing. Elasticity is that feature in cloud computing which is seeking to meet the needs of the user's with no interruption at run time. There are traditional approaches to do elasticity which have been conducted for several years and have been done with different modelling of mathematical. Even though mathematical modellings have done a forward step in meeting the user's needs, there is still a lack in the optimisation of elasticity. To optimise the elasticity in the cloud, it could be better to benefit of Machine Learning algorithms to predict upcoming workloads and assign them to the scheduling algorithm which would achieve an excellent provision of the cloud services and would improve the Quality of Service (QoS) and save power consumption. Therefore, this paper aims to investigate the use of machine learning techniques in order to predict the workload of Physical Hosts (PH) on the cloud and their energy consumption. The environment of the cloud will be the school of computing cloud testbed (SoC) which will host the experiments. The experiments will take on real applications with different behaviours, by changing workloads over time. The results of the experiments demonstrate that our machine learning techniques used in scheduling algorithm is able to predict the workload of physical hosts (CPU utilisation) and that would contribute to reducing power consumption by scheduling the upcoming virtual machines to the lowest CPU utilisation in the environment of physical hosts. Additionally, there are a number of tools, which are used and explored in this paper, such as the WEKA tool to train the real data to explore Machine learning algorithms and the Zabbix tool to monitor the power consumption before and after scheduling the virtual machines to physical hosts. Moreover, the methodology of the paper is the agile approach that helps us in achieving our solution and managing our paper effectively.

Keywords:
Cloud Computing; Optimising Quality of Service (QoS); Resource management, Machine Learning.

1. Introduction

Cloud computing (CC) is a relatively recent advance in technology which has changed the concept of computing considerably. CC is defined by the U.S. National Institute of Standards and Technology (NIST), as the pooling of resources to be shared appropriately to meet the requirements of users with respect to accessing resources, storing data and processing data [1]. The NIST further describes cloud computing as a frame or a model that enables the universal pooling of computer resources, such as storage, applications, services and shared data, All these resources are made available and released with less effort or interactions [2]. Accordingly, it is possible that many groups, such as organisations, academia or industries, can take advantage of the concept of cloud computing to facilitate their work and the services they provide to the end-user. In addition, QoS has become one of the key pursuits of cloud computing. CC can meet users' requirements and satisfy their needs while minimising the provider's costs for power consumption by applying such approaches to resource management.

Resource management is considered one of the major challenges in cloud computing because of the complexity and heterogeneity of systems. According to Kumar and Manoj [3], resource management is the method of distributing demands, such as storage resources, for cloud providers and cloud consumers. Cloud resource management is also affected by massive interactions, some of which are unpredictable, such as failure of the system. Another challenge for cloud providers involves elasticity, particularly that from a fluctuating large load. The decisions on resource utilisation must be made using accurate measurements of the physical and virtual resources needed to distribute applications [3]. A cloud service provider tries to fulfil the requirements of customers, but this requires complex policies and decisions; resource management requires optimisation of multiple objectives, such as load balancing, energy usage, costs, utilisation of processors and availability of machines.

Scheduling is about deciding how to allocate system resources, such as Central Processing Unit (CPU), memory, disk, storage, and network bandwidth, etc. It could be said that scheduling has become one of the most significant issues for cloud computing. The researchers in [4, p.16] stated that 'scheduling algorithms should order the jobs in a

Manuscript received October 5, 2022
Manuscript revised October 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.10.50

way where balance between improving the performance and quality of service and at the same time maintaining the efficiency and fairness among the jobs'. Hence, resource scheduling is about efficiently assigning jobs to be run on machines. Cloud computing can embrace a vast amount of data because of the existence of power computing, therefore, the computation of algorithms will be performed faster with this power computing because of MLs. Thus, ML assists experts and developers in their jobs in the cloud instead of their local machines. In this sense, ML is particularly useful to improve the mechanism of resource allocation, optimise the usage of resources and minimise the use of energy. Therefore, the benefits of ML have attracted researchers and participants to apply it to the cloud, such as in [5], where it was concluded that the optimisation of resources could be increased by implementing ML techniques. Clearly, applying ML to resource allocation and task scheduling contributes positively to meeting Service Level Agreements (SLAs), including the Quality of Service (QoS), and reducing power consumption.

Efficiently exploiting the elasticity of a cloud is critical for the instantaneous provision and de-provision of resources in cloud. it is essential to achieve high performance in the cloud. However, workloads must be predicted to avoid scaling delay which impact negatively on QoS [6]. Previous studies have used two methods for auto-scaling resources: proactive and reactive. The proactive method applies ML techniques to predict upcoming workloads to efficiently provision resources. Hence, to improve the elasticity mechanisms in the cloud, ML algorithms must be applied to predict workloads more accurately and, subsequently, to assign them with the scheduling algorithm. This allows for excellent and efficient provisioning or de-provisioning of cloud resources, improving QoS and minimising power consumption. CC, by its nature, changes state frequently; therefore, exploring ML algorithms to exploit elasticity supports the drive to satisfy customers and minimise the total cost of power consumption for cloud providers.

Additionally, in spite of the cloud computing is considered as one of the solutions that assist users to meet their requirements, CC has faced with some challenges, which can be as obstacles in this technology. These challenges are such as the waste of resources and energy consumption which make the cloud providers pursue to find the optimal solutions of resource management to meet the requirements of users, efficiently. According to Kumar and Singh in [7], the data centre needs to be scaled efficiently as well as dynamic resource scaling and allocation policy. Therefore, optimisation the scheduling of Virtual Machines (VM's) based on the prediction of the workload and energy consumption is really required.

Predicting workloads is fundamental to provisioning resources smoothly. Provisioning resources in the cloud to accomplish different objectives, such as improving QoS and minimising power consumption, has been broadly studied. Researchers have tried to predict workloads using different approaches; for example, the authors in [6] applied mathematical models to predict demands to solve delays in scaling regarding resource management.

Several studies is reviewed in this paper, to begin with, Islam et al. [8], developed a prediction-based model for resource management and strategy provision using neural networks and linear regression. The aim of their model was to solve the issue of delays in allocation resources by anticipating clients' demands. They have approached ML techniques with respect to time, applying two algorithms: error correction neural network (ECNN) and linear regression. They have justified using these algorithms because they are effective for forecasting [8]. Their model has showed promising results which revealed greater success from using the neural network model with a sliding window for estimating resource usage in the cloud.

In the same context of the author Islam's objectives [8], Wang et al. [6] proposed a new trigger strategy for provisioning resources to improve QoS and meet the user's needs as they appeared in an SLA that involved an automatic-scaling mechanism. Wang et al. [6] has predicted the workload by monitoring the data of CPU utilisation using Aliyun VMs as tools. They have used the mean absolute percentage error (MAPE) metric to evaluate the results, which showed improved prediction accuracy and a reduction of delays in the automatic scaling. However, even though the mathematical modelling in that study was an advanced step in provisioning resources, there was still a need to implement ML techniques to predict the workload and achieve QoS, and there was a need to improve the accuracy of the prediction.

Furthermore, Bin et al. [9] tried to predict the precise level of demand for cloud resources. They have considered the planning of cloud capacity as a classification problem. They also have proposed an integrated framework that forecasts changing demands to minimise the cost of providing cloud resources. They have evaluated their results by applying PLR to two other traditional time series segmentations—sliding window and bottom-up—and by comparing weighted SVM with several classifiers, such as the k-neighbours classifier. They showed that their model could customise the degree of changing demands and the importance of different types of changing trends to reduce the overall cost of provisioning. The segmentation strategy of a time series still has some limitations, such as threshold selection and the unknown relationship between the threshold and the degree of changing cloud demands. It may also be possible to integrate regression and classification approaches to increase the accuracy of the predictions.

Moreover, Kumar et al. [7] developed a model based on ML techniques and neural networks and the time interval used in their study was one minute. They have combined a neural network with a self-adaptive differential evaluation to predict demand, to improve QoS and to avoid any violations of the SLA. Their model was also considered to be an expert in its domain [7]. The researchers have determined that their model reduced both the number of violations of the SLA and operational costs. Their outcomes indicated that the proposed approach in this research should further explore ML techniques such as SVM while their results showed higher accuracy in predictions.

Additionally, Montero et al. [13] observed the problems of achieving QoS in the cloud domain, such as long response time, particularly during high traffic loads and fluctuating demands. They found that the SVM model they used to predict the upcoming demand provided an optimal and unique solution. This is because SVM is a global solution, whereas artificial neural networks (ANNs) might suffer from local minima as they mentioned in their study. The researchers proposed a novel mechanism to guarantee QoS that provided an optimal number of resources during peak demands and decreased resource over-provisioning to save power and decrease the total cost of the infrastructure. Their study has used a time series approach and forecasted using ML techniques (SVM). The proposed mechanism in [13] was based on a proactive (predictive) time series mechanism. Montero et al. [13] forecasted workload based on historical observations of a web server. Their proposed method estimated the optimal resources needed to ensure QoS and reduce over-provisioning. They have implemented an SVM technique using different functions of kernel, such as a normalised polynomial kernel and a polynomial kernel, and they also have applied different configuration constraints to obtain optimal results. However, the study showed a strong need to apply ML (SVM) techniques to predict the workloads of big data clusters such as Hadoop or Spark and implementing these techniques in the private cloud.

The objectives of this paper are similar of Montero et al. [13] which are to improve QoS and reduce the over-provisioning of resources causing power consumption. This paper also developed methods based on SVM techniques to predict workloads of PHs to allocate the upcoming demands efficiently to the suitable PHs in order to maintain QoS and save power consumption.

2. Materials and Methods

The method has adopted several steps scientifically to achieve the primary goals of this paper. Hence, this paper investigates the ML techniques to schedule the upcoming VMs and evaluate the performance and power consumption of the proposed method. Therefore the paper used an agile approach to complete the milestones and tasks of experiments of this paper. The tasks involve direct experiments, simulation, mathematical modelling and monitoring technique. Hence, the agile methodology of this paper will base on direct experiments, which need to have an access to SoC to implement the experiments.

To begin with, direct experiments are used in paper to implement several experiments on the real cloud, which were SoC, to schedule the VMs based on the proposed solution and other algorithms, which are benchmarks. In this method, we can set up experiments to provide the results of the implementations. Besides, in our paper, we can use mathematical modelling to explain the behaviour of the novel scheduling algorithm. This means that we could use models to provide mathematical explanations of the proposed scheduling algorithm. To conclude, a monitoring technique is used to keep track of the results of the experiments. In this paper, we monitored the experiments by collecting the results to be discussed critically at the end of each experiment. The monitoring technique involved observing the change of power consumption with each experiment. Consequently, agile methodology has followed due to its benefits like its flexibility to manage the tasks and provide support for delivering the subtasks of the work in this paper regularly [10]. The agile methodology has been defined as ‘a way of thinking about software development which enables developers to accommodate changes while developing a project’ [11, p.436]. As the initial steps, the requirements of the paper were identified. After that, the following steps were taken.

Firstly, data collection is one of the most important tasks in this paper to make the correct prediction of the workload of PHs. In this stage, the proposed method collected data from physical hosts (PHs) of the SoC to be analysed in the next phase. However, there was a need for reliable and credible data to be used in the experiments of this paper. Consequently, instead of collecting the data from PHs on the SoC testbed, the public data from the Google cloud datasets are downloaded to be analysed and implemented in the paper's experiments which seek the appropriate ML algorithms. Furthermore, Google datasets are publicly available and widely used in research. Therefore, Google’s Cloud 29-day usage datasets were downloaded by following the steps in [12]. The second step is data analysis, for which the paper used Microsoft Excel and the WEKA tool. These were used to prepare the data for the next phase: choosing the appropriate ML algorithms. In addition, Tableau software is used in this paper to visualise the data before and after implementing the proposed method. Third, after collecting and analysing the data, the relevant ML algorithms were identified, consistent with the data format. Applying ML algorithms is an essential step because of the high degree of accuracy required since the results from later steps depend on choosing the right ML algorithms. For this paper, SVM techniques were chosen. The benefit of SVM is that the solution is always unique and distinctive with reasonable training times [13]. The fourth step was to design the proposed solution based on the previous phases. Therefore, our paper needed to access the SoC testbed to design the proposed solution and set up many VMs to be used in the experiments. Moreover, several PHs were identified based on the proposed solution. We designed the novel algorithm to be coded in java programming so it would be ready for the next phase. Additionally, metrics were identified to evaluate the proposed method. In this step, the PHs and VMs were prepared for the next phase which is the implementations of the experiments. Fifth, several experiments were conducted. Consequently, implementing these experiments was an essential step for determining the effectiveness of the proposed solution of the paper. Besides, several implementing experiments provided opportunities to discuss the final results critically. The last step of the paper was collecting the results of the experiments by monitoring technique to be discussed. The monitoring technique allowed us to evaluate the proposed method against the benchmarks.

2.1 Datasets

2.1.1 Google datasets for PHs

Google datasets are publicly available and have been used widely in the research domain of Google traces. Hence, Google traces, in our paper, is used to predict the workload of PHs correctly. Therefore, Google's Cloud 29-day usage datasets were downloaded by following the steps in [12]. The traces were generated on at 2014-11-18 09:58Z.

2.1.1 Dataset characteristics for PHs

This part analyses the datasets of PHs. Among the Google datasets, the authors found that the task usage dataset was closest to our interests in this paper because it has CPU usage. A task usage dataset has 20 parameters; nevertheless, in this paper, the authors consider only two parameters: the start time and CPU utilisation. The following tables illustrate the data characteristics of each PH and its analysis in the WEKA tool.

Table 2.1: Dataset characteristics for PH1

Parameter name	***Description***	***Data Type***	***Range***	Interval
Start time	Record of the event	Integer	From 600000000 to 900000000 (microseconds)	5 minutes
CPU sampled Usage	The Usage of CPU	Float	From zero to 0.481	

Table 2.2: Analysis of the parameter (data) in the WEKA tool

Parameter	***Type***	***Missing Value***	***Unique***
Start time	Numeric	No	100%
CPU sampled Usage	Numeric	NO	55%

Table 2.3: Dataset characteristics for PH2

Parameter name	***Description***	***Data Type***	***Range***	Interval
Start time	Record of the event	Integer	From 150100000 to 180000000 (microseconds)	5 minutes
CPU sampled Usage	The Usage of CPU	Float	From zero to 0.209	

Table 2.4: Analysis of the parameter (data) in the WEKA tool

Parameter	***Type***	***Missing Value***	Unique
Start time	Numeric	No	100%
CPU sampled Usage	Numeric	NO	50%

Table 2.5: Dataset characteristics for PH3

Parameter name	***Description***	***Data Type***	***Range***	Interval
Start time	Record of the event	Integer	From 180100000 to 210000000 (microseconds)	5 minutes
CPU sampled Usage	The Usage of CPU	Float	From zero to 0.436	

Table 2.6: Analysis of the parameter (data) in the WEKA tool

Parameter	***Type***	***Missing Value***	Unique
Start time	Numeric	No	100%
CPU sampled Usage	Numeric	NO	63%

Table 2.7: Dataset characteristics for PH4

Parameter name	***Description***	***Data Type***	***Range***	Interval
Start time	Record of the event	Integer	From 210100000 to 240000000 (microseconds)	5 minutes
CPU sampled Usage	The Usage of CPU	Float	From zero to 0.315	

Table 2.8: Analysis of the parameter (data) in the WEKA tool

Parameter	***Type***	***Missing Value***	Unique
Start time	Numeric	No	100%
CPU sampled Usage	Numeric	NO	59%

2.2 The proposed method

Cloud computing architecture comprises three layers. These layers are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The proposed method is focusing on the IaaS layer, as shown in Figure 3.2, which illustrates the cloud architectures. However, the components of the proposed design method are within the virtualisation layer, which is above the IaaS layer, as shown in Figure 3.3, which illustrates the proposed method. The components are as follows: a scheduler, which is comprised of two components which are scheduling and the ML-based prediction components, monitoring the infrastructure, and VM manager. This paper focuses on ML-based prediction because it is a new component in our design solution. To begin with, the Monitoring Infrastructure component is responsible for monitoring the infrastructure resources to obtain the infrastructure data, such as CPU utilisation. After that, the monitoring Infrastructure component stores such data into the database. Moreover, a scheduler component within the virtualisation layer is responsible for where the VM is allocated. Then, the scheduler component calls the VM manager component to allocate VM according to the prediction of ML-based prediction components. To end, the VM manager component is responsible for interacting with the scheduler component to decide whether the scheduler component occurs or not.

2.2.1 ML-based prediction component

This component is used to generate a valuable model to serve the objective of this paper. The valuable model is derived from the volume of data or previous experience [14]. ML is the field that leverages historical volume data within the clouds. By leveraging ML, the proposed solution would use the historical data for mPH to apply SVM algorithms to predict the CPU utilisation of PHs. The main focus of our proposed solution is ML-prediction component. The ML-prediction component is based on the SVM algorithm to predict CPU utilisation. The data, which is used to train the model of the SVM algorithm, is obtained from the database. After that, the scheduler will schedule the nVM to the appropriate mPH. This would contribute to the paper's objectives, saving power consumption and achieving the QoS.

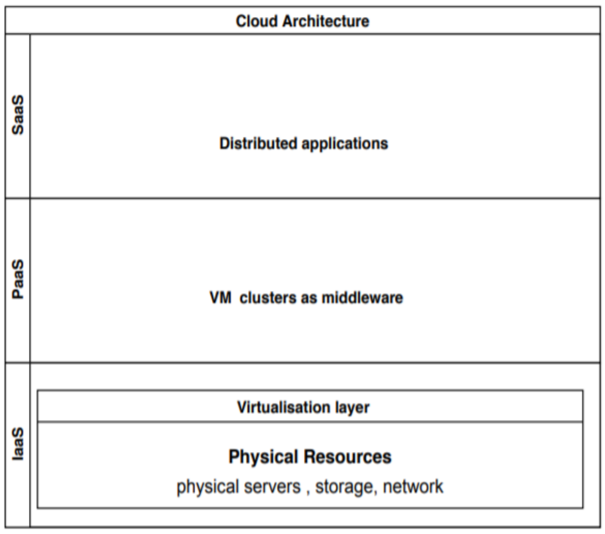


Fig. 2.1 The cloud architecture.

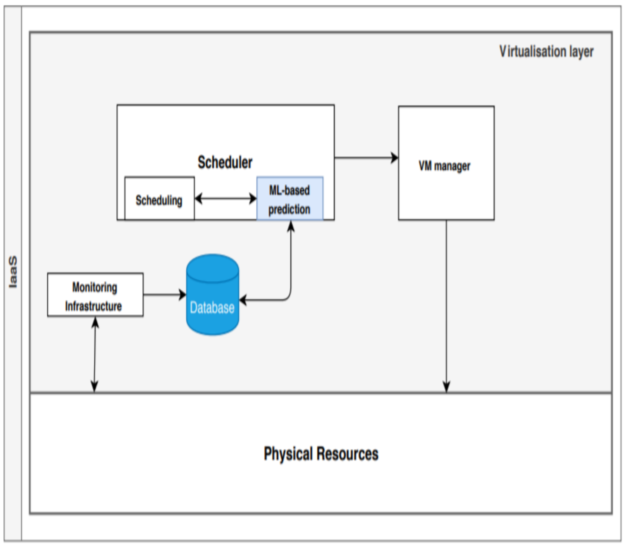


Fig. 2.2 The Proposed method.

2.2.2 Support Vector Machine (SVM)

This algorithm is used widely to accomplish predictions, especially in time series forecasting. Vapnik mentioned that, as cited by Flake et al. [15, p.271] 'A support vector machine (SVM) is a type of model that is optimized so that prediction error and model complexity is simultaneously minimized'. Additionally, SVM regression showed promising findings to provision resources in the clouds with actual data as well as presented high accuracy in Montero et al. [13] works. Moreover, the data in this paper is in time series; therefore, SVM regression is suitable for this case. That is why the authors chose SVM regression for our proposed solution. In this paper, SVM is trained with Google dataset for t intervals as well as SVM algorithm splits the dataset into two separate sets, a training set and a test set. The authors applied the SVM algorithm by using the Weka tool. The authors use the WEKA tool because it has plenty set of ML algorithms which can be used to extract the patterns of data [16]. Furthermore, the WEKA tool has features that ease dealing with datasets, such as pre-processing and visualizing data. Therefore, the authors used this tool to deal with data effortlessly according to its features and capabilities.

2.2.2.1 Model Metrics

The model of predictions (SVM) has several evaluation metrics to measure the accuracy of predictions, such as mean absolute error (MAE), mean squared error (MSE), mean absolute percentage error (MAPE), and direction accuracy (DA).

1- MAE is the evaluation metric used for the regression model. This metric is the average error in this model for all the instances in the test set [17]. In this paper, the instances in the test set are the CPU utilisation.

2- MSE is the metric of the average squared error in this model for all the instances in the test set [17]. The instances are the CPU utilisation values.

3- DA is a measure of the accuracy of prediction CPU utilisation values. DA is the comparison of the prediction direction to the actual direction [18].

2.2.3 The Heart of the Novel Algorithm

The VMs were allocated to the PH because the novel algorithm knew in advance that the PH would accommodate the upcoming VMs based on the CPU utilisation of PHs. Therefore, the allocation is based on the minCPU among the mPHs. The novel algorithm is designed as follows:

- Step 1: The scheduling algorithm is fed with predictions of the CPU utilisations for t seconds for the mPH.

- Step 2: According to the ML-based prediction component, the minCPU utilisation is checked for the mPHs at t1, t2, t3, and t4. In. Then, the VM is scheduled based on the predictions of CPU utilisation.

- Step 3: After knowing the minCPU of mPHs, the VM requested is scheduled for the minCPU utilisation of the mPH.

2.2.3.1 The Novel Algorithm (Version 1)

The algorithm checked the CPU utilisation at t1, t2, t3, and t4 seconds for mPH that we had on the SoC private cloud testbed. Afterwards, we found the minCPU utilisation among mPH at a specific time (t1, t2, t3, and t4 seconds). For example, the most economical, which is the minCPU utilisation, at t3 seconds was for mPH. In this case, after allocating nVM to mPH, we considered that the workload does not change. We repeated the same exercise at t4 seconds. This resulted in the discovery of the limitations of the algorithm. As shown in the Table 3.9, nVMs were allocated to the same mPHs. This means that the algorithm does not consider the change of the workload when nVM was allocated to mPH at t3 seconds. This encouraged us to improve and further update this algorithm. The illustration of the algorithm (version 1) is shown in Table 2.9.

Table 2.9: The illustration of the novel algorithm (version 1)

Host / VM	VM1	VM2	VM3	VM4	Minimum CPU utilisation
PH1			√	√	At 30 seconds
PH2					At 40 seconds
PH3	√				At 10 seconds
PH4		√			At 20 seconds
	10 seconds	20 seconds	30 seconds	40 seconds	

2.2.3.2 The Novel Algorithm (Version 2)

This algorithm found the minCPU utilisation at t1, t2, t3, and t4 seconds for mPH that we had on the SoC private cloud testbed. For example, the minCPU utilisation at t3 seconds was for mPH, as it happened for algorithm version 1. However, in this case, after allocating nVM to mPH, we do not consider mPH to check again for the minCPU utilisation at t4 seconds. This was because the workloads are changing after allocating nVM to mPH. Again, we repeated the same exercise at t4 seconds, but the first mPHs were not counted in the check for the minCPU utilisation. This resulted in solving the limitation of the algorithm in version 1. As shown in Table 2.10, nVMs were allocated to the mPHs at t1, t2, t3, and t4, respectively. This means that the algorithm considers the change of workloads when nVM is allocated to mPH at t seconds. The illustration of the algorithm (version 2) is shown in Table 2.10.

Table 2.10: The illustration of the novel algorithm (version 2)

Host / VM	VM1	VM2	VM3	VM4	Minimum CPU utilisation
Ph1			√	X	At 30 seconds
Ph2				√	At 40 seconds
Ph3	√	X	X	X	At 10 seconds
Ph4		√	X	X	At 20 seconds
	10 seconds	20 seconds	30 seconds	40 seconds	

Novel Algorithm Pseudocode

```
Input: mPH, nVM
Output: Allocate nVM to mPH, which has minCPU Utilisation among
mPH at times t1, t2, t3, and t4.
Input: List of mPH , List of nVM
For every VM allocation request
        Let utList be an empty list
        minIdx←0
        minUt ← max number
For each PH in listOfmPH
                ut ← get the minCPU utilisation at time t for PH
                if ut < minUt then
                        minUt←ut
                        minIdx ← index of the PH
                end if
        end for
        selectedPH ← get the PH at index minIdx from listOfmPH
        VM ← pop the first virtual machine from listOfnVMs
        assign VM to selectedPH
        remove selectedPH from listOfmPH
        if listOfmPH is empty or listOfnVMs is empty then
                exit for loop
        end if
end for
```

2.2.4 Metrics for the proposed solution

There are some metrics to evaluate the proposed solution. They are the novel algorithm's efficiency, the scheduling algorithm's complexity, and CPU utilisation.

2.2.4.1 Efficiency of the novel algorithm (version 2)

The assessment among benchmark algorithms is to show the efficiency of the novel algorithm (version 2). The benchmarks of the novel algorithm are matchmaking scheduler of OpenNebula and the random scheduling algorithm.

2.2.4.2 The complexity of the scheduling algorithm

The novel algorithm is assessed to see the complexity of the scheduling algorithm, which is how long it takes to run the novel algorithm (version 2).

2.2.4.3 CPU utilization

CPU Utilisation is the primary metric in our paper to execute the experiments. The CPU utilisation for all PHs in our environment is considered to make the prediction by the SVM technique. Accordingly, the VM will be allocated to the minimum CPU utilisation of PHs.

3. Experimental Design

3.1 High-Level System Architecture

Cloud computing systems have three models. In our paper, the focus is on the infrastructure level. This level includes several components, as shown in Figure 3.1, which is taken from [19].

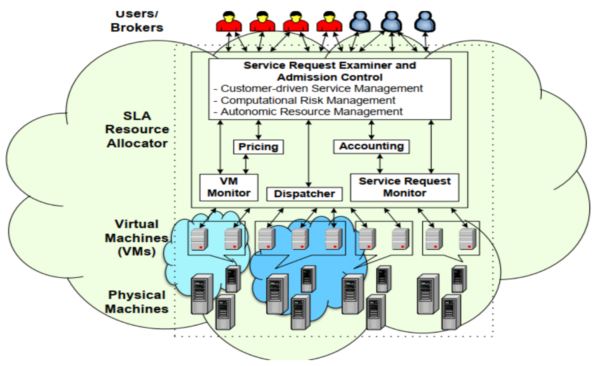

Users/ Brokers
Service Request Examiner and Admission Control
- Customer-driven Service Management
- Computational Risk Management
- Autonomic Resource Management
SLA Resource Allocator
Pricing
Accounting
VM Monitor
Dispatcher
Service Request Monitor
Virtual Machines (VMs)
Physical Machines

Fig. 3.1 High-Level System Architecture.

3.1.1 Users/Brokers

The users or brokers received requests to create VMs with their specifications, such as CPU utilisations. Then, the resource manager deploys these VMs on the SoC private cloud testbed.

3.1.2 Resource Manager

The resource manager is responsible for managing client requests and the cloud infrastructure. Therefore, the resource manager plays an essential role in managing the demands and the availability of the resources on the cloud. In addition, the resource manager acts as a coordinator among various components, such as the VM scheduler and SLA manager. Our paper focuses on two components: 1) the VM scheduler allocates the requested VM to the PH on the SoC private cloud, and 2) the SLA manager makes sure the agreement between clients and cloud service providers is met by avoiding any violations in the SLA. Overall, the resource manager ensures that the system stays up and running by improving its performance.

3.2 Virtual Machines (VMs)

A VM is a virtual representation of a physical machine using software that has the capabilities to accommodate and run an operating system [20]. Therefore, it can act like a separate physical machine. VMs have become one of the technologies that have been exploited in the context of virtualisation. Therefore, in our paper, we dealt with VMs to assign them to the lowest CPU utilisations of PHs. Allocating the VMs to the PHs based on the proposed solution would positively contribute to the objectives of our paper. The proposed solution focused on providing high QoS and saving power by forecasting workloads.

3.3 Physical Hosts (PHs)

In this paper, the SoC private cloud testbed consists of several PHs. PHs are the underlying hardware resources of the infrastructure of this private cloud. PHs, which accommodate VMs, use hypervisor software to abstract the infrastructure and create VMs. The creation of VMs was completed using OpenNebula, which is a Virtual Infrastructure Manager (VIM).

3.4 Introduction of the Proposed Solution

The proposed solution in our paper depends on predicting the workload using the SVM technique. The workload is the CPU utilisation of PHs. The proposed solution, which has an ML-based prediction component, would be implemented to predict the CPUs' utilisation of PHs. Therefore, the upcoming VMs will be allocated to an appropriate PH, which meets the conditions of the proposed solution, on the SoC private cloud testbed (cluster). At this point, the proposed solution would consider power consumption. This means identifying the server that would accommodate the VM efficiently while saving power. Therefore, the main component of the proposed design solution is the resource manager. The resource manager will manage the requests of users/brokers and manage computing resources of PHs. In our paper, we assumed several PHs out of the total PHs on the SoC private cloud testbed because we have a limited number of hosts. The resource manager collected the requests from the users/brokers and created a new VM. Then, the VM scheduler deployed the VM based on the proposed solution. This means that the schedule of the VMs depends on the predictions of CPU utilisations of PHs. The allocation algorithm chose the lowest CPU utilisation among the available PHs to save power. Therefore, the role of our proposed solution is to allocate the VM to the lowest CPU utilisation of PHs to save power consumption while providing QoS. Therefore, our solution aimed to provide QoS and minimise power consumption to meet user requirements and avoid any violations that could arise in the SLA.

To evaluate the proposed solution, it would be better to compare it with other algorithms that schedule VMs. First, the basic algorithm used by OpenNebula was the matchmaking scheduler. OpenNebula checked the specifications of PHs, such as the CPUs, and then decided which physical host could house the VM. Matchmaking is a basic algorithm that tries to match the VM requirements to what is available for them in a PH. The matchmaking method is used in this paper as a benchmark for our proposed solution. Second, a random algorithm allocates VMs randomly to the PHs. The random method is implemented to use the results as a benchmark for our proposed solution. Finally, by assuming that the VM is deployed, it is randomly allocated among the available PHs on the SoC private cloud testbed.

3.4.1 Introduction of a Novel Algorithm

Before introducing the novel algorithm, it is fundamental to mention that the window of prediction to allocate the VM to the appropriate PH is in seconds. The reason is that the records of Google dataset were a vast number in a millisecond, and we also studied more experiments to find optimal results in seconds. Furthermore, in the proposed solution, we assume that we have:

1- nVM = number of VMs (VM1, VM2, VM3, and VM4).

2- mPH = number of physical hosts with dataset predictions (PH1, PH2, PH3, and PH4).

3- minCPU = the minimum CPU utilisation for a PH among mPH.

4- t = which is a specific time to be checked for CPU utilisation (t1, t2, t3, and t4).

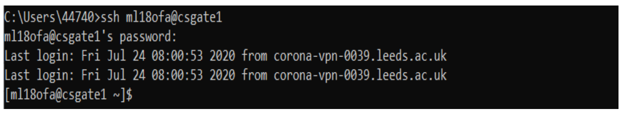
In our environment, several VMs are allocated to the PHs on the SoC private cloud testbed based on different scheduling algorithms. The CPU utilisation is considered to be one of the most significant parameters that impact the power consumption of servers [21]. Since our proposed solution is limited to the number of PHs, we only consider CPU utilisation in applying the novel algorithm. By implementing the proposed solution, the VM is allocated to the minCPU utilisation among the mPH at the specific time (t1, t2, t3, and t4). The novel algorithm knew in advance which server is with the minCPU utilisation for the next t seconds. All of that occurred because we used the mPH datasets. Google cloud datasets are used as data for mPH in our design solution. The Google cloud datasets were analysed, and applying SVM on these datasets. The paper considers several datasets from Google traces for the mPH where each dataset is in t interval.

4. Experimental Results

4.1 Implementation

4.1.1 SoC Testbed

The school of computing provides an SoC testbed as a gateway to the University of Leeds. This testbed is used in this paper to demonstrate the proposed solution and runs the different algorithms of allocating VMs and use these results as benchmarks to assess the novel algorithm. It can be accessed by using the credentials of the University of Leeds via the SSH protocol as "ssh usernam@csgate1.ac.uk" or "ssh username@csgate1" as shown in Figure 4.1. According to Ylonen et al. in [22, p.1] SSH is defined as 'The Secure Shell (SSH) Protocol is a protocol for secure remote login and other secure network services over an insecure network'. Therefore, SSH provides a secure bridge to interact remotely with resources of the SoC private cloud. Regarding the SoC testbed, users can run different actions on cloud resources, such as benchmark and novel algorithms, after signing in successfully into the SoC testbed via SSH. The benchmark and the novel algorithm results will be assessed on the SoC testbed.



```
C:\Users\44740>ssh ml18ofa@csgate1
ml18ofa@csgate1's password:
Last login: Fri Jul 24 08:00:53 2020 from corona-vpn-0039.leeds.ac.uk
Last login: Fri Jul 24 08:00:53 2020 from corona-vpn-0039.leeds.ac.uk
[ml18ofa@csgate1 ~]$
```

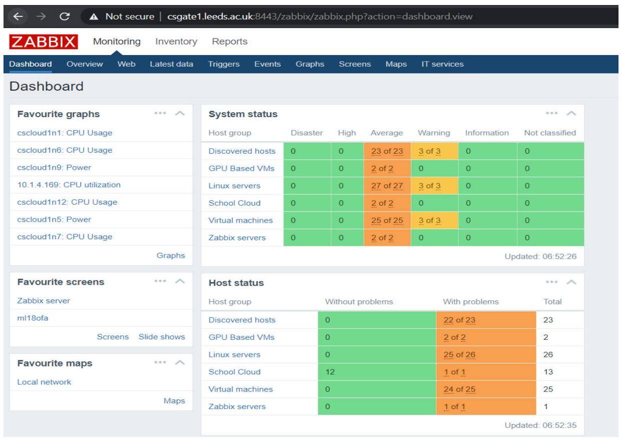
Fig. 4.1 SoC Testbed.

4.1.2 Monitoring Infrastructure

There are some tools which are used to monitor the infrastructure of clouds, such as Zabbix. Zabbix is the possible tool that would be used in this paper to collect the data of PHs on the SoC private cloud testbed. This tool is available on the School of Computing Cloud platform. Zabbix is an open-source tool that can be used to monitor networks, server monitoring, application monitoring, and service monitoring [23]. Zabbix aimed to analyse and monitor the status PHs. This tool is used to monitor the CPU utilisation of PHs in order to be stored in the database. The PHs on the SoC private cloud testbed are 14 PHs. The details of these PHs can be seen in Appendix A. In this paper, we assumed only 4 PHs out of 14 PHs on the SoC private cloud testbed to be monitored for CPU utilisations. Nevertheless, in our paper, instead of collecting the data of the SoC private cloud testbed for 4 PHs, we would use the data of Google datasets for these 4 PHs to implement the proposed solution. Zabbix tool can be accessed from the following address:

https://csgate1.leeds.ac.uk:8443/zabbix/index.php.

We used the "sign in as a guest" option to have access to the Zabbix monitoring system. The main dashboard of the Zabbix monitoring system, as shown in Figure 4.2, is under the option Monitoring. Zabbix monitoring system is used to monitor the changes in workloads (CPU utilisation) in PHs after allocating the VM. It can also be used to monitor the power consumption for 4 PHs.



Dashboard

Favourite graphs

- cscloud1n1: CPU Usage
- cscloud1n6: CPU Usage
- cscloud1n9: Power
- 10.1.4.169: CPU utilization
- cscloud1n12: CPU Usage
- cscloud1n5: Power
- cscloud1n7: CPU Usage

Graphs

System status

Host group	Disaster	High	Average	Warning	Information	Not classified
Discovered hosts	0	0	23 of 23	3 of 3	0	0
GPU Based VMs	0	0	2 of 2	0	0	0
Linux servers	0	0	27 of 27	3 of 3	0	0
School Cloud	0	0	2 of 2	0	0	0
Virtual machines	0	0	25 of 25	3 of 3	0	0
Zabbix servers	0	0	2 of 2	0	0	0

Updated: 06:52:26

Favourite screens

- Zabbix server
- ml18ofa

Screens Slide shows

Favourite maps

- Local network

Maps

Host status

Host group	Without problems	With problems	Total
Discovered hosts	0	22 of 23	23
GPU Based VMs	0	2 of 2	2
Linux servers	0	25 of 26	26
School Cloud	12	1 of 1	13
Virtual machines	0	24 of 25	25
Zabbix servers	0	1 of 1	1

Updated: 06:52:35

Fig. 4.2 Zabbix Monitoring System.

4.1.3 Virtual Machine (VM)

The VM is one of the most necessary parts of our paper to perform the proposed solution. The proposed solution schedule the VMs requested by users/brokers to PHs. These VMs can be different templates which contain details of the VM. The SoC private cloud testbed provides several templates of the VMs, which could be used in our paper, to be scheduled based on the proposed solution. In this paper, the template of the VM is Debian Stretch (Hadoop) x86_64 Base, with CPU="0.1" and MEMORY="1024" Mb. The list of other templates in OpenNebula is in Appendix B. In this paper, the VM will be scheduled by the proposed system regardless of the type of VM.

4.2.1 Implementation of the ML-based prediction component

After collecting the CPU utilisation with the Zabbix tool and storing it into a database, the SVM technique will use such data to make the prediction of the upcoming workloads of 4 PHs for the next 60 seconds. It is important to mention that the window of predictions to allocate the VMs to the appropriate PHs (among the 4 PHs) is 60 seconds. This dataset considered only two parameters for scheduling the upcoming 4 VMs. Therefore, the paper considers four datasets from Google traces for the 4 PHs, where each dataset is in 5-minute intervals. In this paper, SVM is trained with the Google dataset for a 5-minute interval. The SVM algorithm splits the dataset into two separate datasets as a training set and a test set. The training sets in our paper are 80% of the datasets, and the remaining is for the test set. The segmentation of data sets into 80% and 20% shows better results for SVM models than other segmentations, such as 70% for training and 30% for the test set.

To implement the prediction of the CPU utilisation of 4 PHs, we implement that by using the WEKA tool. The datasets in this tool must be prepared and in appropriate formats to make the prediction correctly. Some steps must be followed to prepare the datasets to make the datasets ready for the ML-based prediction component.

Data preparation: the data must be cleaned, modified, and adjusted to be well-suited for the WEKA tool. The datasets were in microseconds which made the datasets in a huge number of records. Therefore, we made the datasets for each PH in seconds by assuming that the maximum value for each of the microseconds records represents the value of the seconds' records. By implementing this step, we reduced the significant number of records to 300, representing a 5-minutes interval.

Data preparation comprised two stages: cleaning data and transforming the datasets to the format acceptable to the WEKA tool. In the cleaning step, the CPU utilisation parameter was cleaned from unneeded values or missing values that might cause model metrics. In addition, duplicate records were removed. To explain that, if the start time and sampled CPU usage are the same on more than one record, the duplicate record was removed. The transformation step took the cleaned parameters (start time and the CPU utilisation) and exported them into a format that WEKA could read, such as .arff or .cvs files.

4.2.2 Implementation of the SVM

This section implements the SVM algorithm to predict the next 60 seconds of each PHs while the training set is in 240 seconds. Therefore, we use Tableau software to visualise the actual CPU utilisation and prediction one. We also present the different segmentations of training and test sets, and we learned that better results occurred when 70% was used for training and 30% for testing. Therefore, we outline two scenarios for implementing SVM out of several scenarios. Different scenarios have been done, and from these scenarios, we find the better results as follows:

1- The dataset is applied in the WEKA tool as a training set that includes 80% of the dataset and a test set with 20%. The results of the model with metrics are shown in Table 4.1 and illustrated in Figure 4.3.

Table 4.1: The results of the experiments for 80% and 20%

Filename	PH2
Instances	300
Execution	yes
Number of training records	240
Number of predict records	60
MAE	0.0337
DA	45.7627
MAPE	451.2559

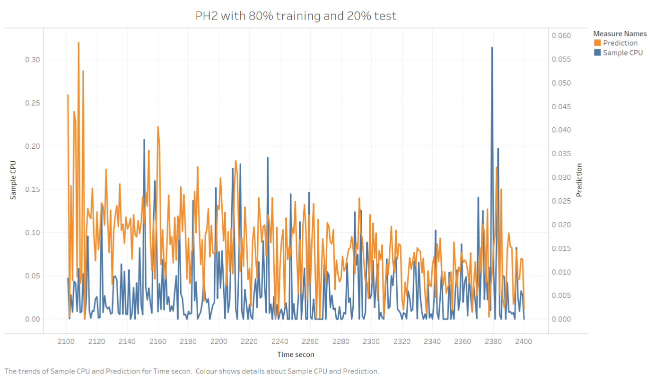


Figure 4.3: The 80% training and 20% test set.

2- The WEKA tool is fed with the datasets into two sets: a training set that includes 70% of the dataset and a test set with 30%. The outcomes of the model SVM with metrics are shown in Table 4.2 and illustrated in Figure 4.4.

Table 4.2: The results of the experiments for 70% and 30%

Filename	PH2
Instances	300
Execution	yes
Number of training records	240
Number of predict records	60
MAE	0.0283
DA	55.0562
MAPE	490.7062

4.2.2 Implementation of the Novel Algorithm (version 2)

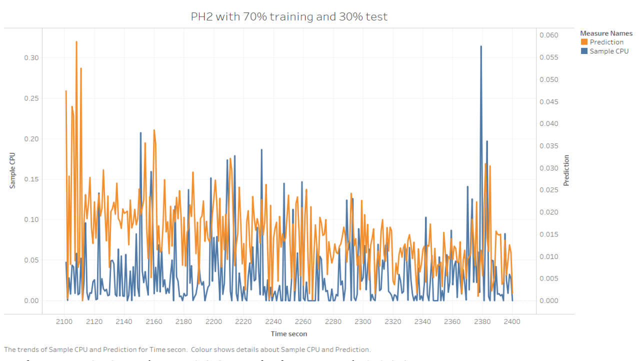


Figure 4.4: The 70% training and 30% test set

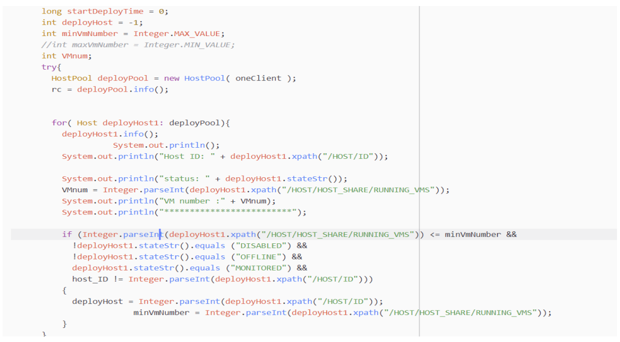
The scheduling component relies on the results of prediction workloads. This component plays an important role in scheduling the VM to PH. This section presents the details of the steps and implementations of the novel algorithm. The steps were:

1- The scheduling algorithm checks the minCPU utilisation at 10, 20, 30, and 40 seconds for mPHs, based on the prediction of the ML-based prediction component.

2- The scheduling algorithm decides which PH will host the VM at 10 seconds. The same will happen at 20, 30, and 40 seconds.

3- The VM manager is called to allocate VM based on the results of Step 1 (the proposed system).

This means that the algorithm checks the minCPU utilisation at 10, 20, 30, and 40 seconds for 4 PHs in our environment, which is the SoC private cloud. For instance, the minCPU utilisation at 30 seconds was for PH3. In that case, the scheduling algorithm does not consider PH3 to check again at 40 seconds since the workload is changing after assigning VM3 to PH3. To explain that the algorithm does not check the minCPU utilisation after allocating VM. To implement the algorithm in SoC cloud testbed, we assumed that the snip of the code in Figure 4.4 was to implement the proposed solution.



```java
long startDeployTime = 0;
int deployHost = -1;
int minVmNumber = Integer.MAX_VALUE;
//int maxVmNumber = Integer.MIN_VALUE;
int VMnum;
try{
  HostPool deployPool = new HostPool( oneClient );
  rc = deployPool.info();


  for( Host deployHost1: deployPool){
    deployHost1.info();
              System.out.println();
    System.out.println("Host ID: " + deployHost1.xpath("/HOST/ID"));

    System.out.println("status: " + deployHost1.stateStr());
    VMnum = Integer.parseInt(deployHost1.xpath("/HOST/HOST_SHARE/RUNNING_VMS"));
    System.out.println("VM number :" + VMnum);
    System.out.println("************************");

    if (Integer.parseInt(deployHost1.xpath("/HOST/HOST_SHARE/RUNNING_VMS")) <= minVmNumber &&
      !deployHost1.stateStr().equals ("DISABLED") &&
      !deployHost1.stateStr().equals ("OFFLINE") &&
      deployHost1.stateStr().equals ("MONITORED") &&
      host_ID != Integer.parseInt(deployHost1.xpath("/HOST/ID")))
    {
      deployHost = Integer.parseInt(deployHost1.xpath("/HOST/ID"));
                  minVmNumber = Integer.parseInt(deployHost1.xpath("/HOST/HOST_SHARE/RUNNING_VMS"));
    }
}
```

Fig. 4.5: Allocating VM to the lowest CPU utilisation

4.2.3 Technical Evaluation

In this section, we will present the technical evaluation of our proposed solution and benchmark results. This section will discuss the results of the proposed solution, which takes place on the SoC private cloud testbed. Additionally, the benchmark results will be discussed in comparison with the novel algorithm to show the differences between them and their effectiveness.

4.2.4 Scheduling VMs to PHs

For all experiments, we recorded the power consumption for the 4 PHs before and after hosting 4 VMs. Moreover, all experiments were subjected to the time that it took to schedule the 4 VMs to 4 PHs. In this paper, we performed several experiments to schedule 4 VMs to the 4 PHs, based on the benchmarks and novel algorithm.

4.2.5 Benchmarks Evaluation

The first experiment was a benchmark for the novel algorithm. It scheduled VMs using a matchmaking scheduler, which was provided by OpenNebula. For this experiment, the time it took to allocate the 4 VMs to the 4 PHs was considered as the standard for evaluating the complexity of this algorithm. We also considered power consumption. This algorithm took different times to schedule the VMs requested by users/brokers. Furthermore, the algorithm did not count the minimum CPU utilisation of the PHs; therefore, the 4 VMs were allocated to the same PH 4, which was host ID 21 on the SoC private cloud testbed. This meant that the power consumption for PH 4 on this experiment was the highest among the PHs. The results of the first experiment are shown in Table 4.3. Figure 4.6 presents the power consumption during this experiment. Appendix E shows the 4 VMs allocated to the same PH, which was PH 4.

The second experiment was to schedule the 4 VMs to 4 PHs randomly. In this experiment, we also evaluated the complexity of the random algorithm to schedule 4 VMs to the 4 PHs. We also studied the power consumption for the 4 PHs. This algorithm presented changed times to schedule the 4 VMs. Moreover, this algorithm did not check the minimum CPU utilisation of the 4 PHs. Therefore the 4 VMs were allocated to the 4 PHs randomly on the SoC private cloud testbed. This resulted in more power consumption, and it might allocate the VM to the highest CPU utilisation of PHs. This contradicts our novel algorithm, which seeks to find the lowest CPU utilisation to allocate the VM. The power consumption in this experiment for the PHs was diverse and it could not be predicted. The results of the second experiment are shown in Table 4.3.

Table 4.3: The results of the experiments of the random algorithm

Algorithm: Random Algorithm						
Host/ VM	VM 1	VM2	VM 3	VM4	Minimum CPU utilisation	Power consumption
	Time to allocate VM in millisecond (ms)					
Ph1 = host ID 9	√	√	√		Not Considered	118 Watts
	3000 ms	3089 ms	2841 ms			
Ph2 Host ID 1					Not Considered	110 Watts
Ph3 Host ID 8					Not Considered	110 Watts
Ph4 = Host ID 21				√	Not Considered	112 Watts
				3083 ms		
Average	3003.25 ms					Total Power consumption
Standard Deviation	115.540					
Time	10 seconds	20 seconds	30 seconds	40 seconds		450 Watts

4.2.6 Evaluation of the novel algorithm

The novel algorithm was used in the last experiment to prove its efficiency over the previous ones. The heart of the novel algorithm was to schedule the requested VM to the minimum CPU utilisation among the 4 PHs. That is, if VM 1 was requested, the algorithm finds the minCPU utilisation at 10 seconds for 4 PHs to schedule the VM. The same exercise used on the rest of the VMs. For example, VM 2 was requested by users/brokers, and the novel algorithm allocated the VM 2 to the minimum CPU utilisation at 20 seconds of the rest of PHs. We hypothesised the complexity of novel algorithm to schedule 4 VMs to the 4 PHs was the most efficient compared to the random and matching algorithms due to the promising results of the novel algorithm. Hypothetically, the results showed the quickest scheduling VMs to PHs compared to the benchmarks. This algorithm allocated the upcoming VM to the known PH, and this would contribute to saving power consumption by avoiding allocating VMs to the same PHs. This means that the novel algorithm considered the power consumption while seeking QoS. Therefore, we looked at the power consumption for the 4 PHs in this experiment, and hypothetically, the power consumption was the most efficient one compared to the results of the benchmarks. The results of the novel experiment are presented in Table 4.4.

Table 4.4: The results of the experiments of the novel algorithm

Algorithm: The novel algorithm						
Host/ VM	VM1	VM2	VM3	VM4	Minimum CPU utilisation	Power consumption
	Time to allocate VM in millisecond (ms)					
Ph1 = host ID 9			√ 1258 ms	🟥	Considered	112 Watts
Ph2 = host ID 1				√ 1261 ms	Considered	112 Watts
Ph3 = host ID 8	√ 1267 ms	🟥	🟥	🟥	Considered	112 Watts
Ph4 = host ID 21		√ 1252 ms	🟥	🟥	Considered	112 Watts
Average	1259.5 ms				Total Power consumption	
Standard Deviation	6.245					
Time	10 seconds	20 seconds	30 seconds	40 seconds	448 Watts	

4.2.7 VM Scheduling Times and Power consumption

Scheduling 4 VMs to 4 PHs went through three experiments to provide the results and prove the efficiency of the novel algorithm. As we can see in Table 4.6, the scheduling time for every algorithm was different, and the power consumption also was different for each algorithm. First, the Matchmaking scheduler By OpenNebula took just under 3 seconds, on average, to schedule 4 VMs to the 4 PHs, and its power consumption was the highest, at 456 Watts, compared to the other algorithms. This means saving power consumption, in this case, will not be considered. Second, the random algorithm took an average of just over 3 seconds to schedule the upcoming 4 VMs to the 4 PHs, and power consumption was different each time. Even though the random algorithm used less power than the matchmaking algorithm, it could be the worst option by showing more power consumption. In this experiment, the power consumption was 450 Watts which was less than in the first experiments. Concurrently, power consumption was more than the novel algorithm. Third, the novel algorithm showed promising results compared with the benchmarks. This algorithm was the fastest one to schedule the upcoming VMs to PHs. The reason is that it had a component to predict the workload, which is CPU utilisation of PHs for the next 60 seconds. In addition, power consumption was the lowest among the scheduling algorithms, at 448 Watts. Our novel algorithm showed the best performance compared to the other experiments and reached the minimum power consumption among all the experiments. Finally, since the novel algorithm showed better results compared with the benchmarks, we would recommend using ML techniques for provisioning cloud computing instead of traditional methods. From the literature review and our experiments, the promising results of the ML techniques would enhance managing data centres by ensuring that the QoS and saving power consumption are achieved optimally. To conclude the experiments, Table 4.6 compares the results of the experiments.

This section described the environment of the proposed solution and it outlined the preparation of data to be implemented on the WEKA tool. The benchmarks were presented to assess the efficiency of the novel algorithm. The novel algorithm showed optimal results in terms of ensuring QoS and saving power. Tables and figures were presented to demonstrate the results for each algorithm. Table 4.5 shows that the novel algorithm provided optimal results in scheduling the VMs to PHs by using an ML algorithm.

Table 4.5: The results of all the experiments

Scheduling Algorithm	Average time	Standard deviation	Power consumption for scheduling 4 VMs
Matchmaking scheduler By OpenNebula	2790.75 ms	24.76	456 Watts
Random	3003.25 ms	115.540	450 Watts
Novel algorithm	1259.5 ms	6.245	448 Watts

5. Comparison with the previous methods

In this section, the results of the paper are compared with related work, which was in the literature review. This comparison of results aims to evaluate the achievements of this paper and how these achievements contribute to QoS and saving power. Many researchers have studied ML techniques to predict workload. However, there was a need for further investigation to explore ML algorithms and implement those algorithms in a real cloud data centre. In the literature review, we found that the results of previous efforts were promising for finding ML techniques and implementing them in real clouds. For example, Islam et al. [8] had promising results when they used ML algorithms to predict the workload. Their results showed how many resources would be used in clouds. These results were an important key for further studies. They noted that it could be possible to accommodate other ML methods to predict the workload for CPU utilisation by SVM. Therefore, our paper with novel algorithms used Google datasets for CPU utilisation to conduct the experiments with real data – meaning operating in a real cloud – to see how our algorithms with SVM worked. The results of our paper were promising in terms of performance and saving power for PHs.

Bin et al. [9] and Montero et al. [13] have made predictions about using the ML algorithm, which was using SVM with different platforms and data. Bin et al. [9] and Montero et al. [13] contributed in achieving the QoS, and Montero et al. [13] also reduced over-provisioning, which was necessary with respect to power consumption. Evaluating our paper against the results of Bin et al. in [9] and Montero et al. [13], we found that our work presented a direct contribution to estimating resource usage for the next 60 seconds. From our results, our work sought to achieve QoS and not neglect power consumption.

Finally, the points listed below show the contributions of our work and compare our work with the research discussed in the literature review. In addition, Table 5.1 provides the results for related works.

- The proposed method has used real data, which were Google dataset traces. In comparison, previous works used different data.
- The proposed method has investigated ML algorithms to find one appropriate for implementation in a real cloud with a small time unit. We found in some previous works intended to use SVM with big data and different time units.
- The proposed method has monitored the power consumption for experiments by using the Zabbix tool.
- The metrics of The proposed method have accurately shown the results of ML techniques.
- The results of the proposed method proved its effectiveness in scheduling the VMs to the PHs, which had the lowest CPU utilisation among PHs.

Table 5.1: Results compared with related research

# of criteria	Criteria / Paper	Islam et al. [18]	Wang et al. [1]	Bin et al. [19]	Kumar et al. [2]	Montero et al. [6]	Our proposed method
1	Predict workload which was the CPU utilisation	√	√	X	X	X	√
2	ML techniques	√	X	√	√	√	√
3	Promising results related to QoS	X	√	√	√	√	√
4	Contributions in Power consumption	X	X	X	√	√	√
5	Monitor power consumption before and after scheduling VMs	X	X	X	X	√	√
6	Identify the trade-off between performance and energy	X	X	X	X	X	√

6. Conclusion and Future works

This paper highlights cloud computing topics and trends. The paper also demonstrates the importance of using ML techniques for scheduling VMs, while using less power than occurs with traditional scheduling. Therefore, scheduling the upcoming VMs based on the proposed solution has been shown to achieve the primary goals of this paper. The proposed solution used real data from Google to implement ML techniques. The proposed solution is based on scheduling the VMs to the lowest CPU utilisation on a cloud testbed.

Moreover, this paper analysed several papers in this area to identify appropriate ML techniques and analyses the methods used in those papers. Based on that research, the paper designed the proposed solution to implement several experiments and compare the findings to benchmarks. These experiments were implemented to evaluate the effectiveness of the proposed solution on the SoC testbed. The benchmarks in this paper were the matchmaking scheduler by OpenNebula and a random algorithm. These benchmarks were used to evaluate the novel algorithm proposed in this paper.

Therefore, this paper has studied and investigated ML techniques in the SoC testbed to evaluate their performance and energy efficiency. Moreover, the paper tested the performance of the proposed solution by using real data from the Google trace dataset, and it analysed the data to be implemented with the ML techniques of the paper using SVM regression.

The results are promising and vital to cloud providers to support them in achieving the QoS for their customers and reducing power consumption. By reducing the amount of power wasted, the total cost of power consumption in data centres will be reduced. Accordingly, this paper recommends further work in this area to investigate additional machine learning techniques to be used in real clouds or in edge computing environments. The goal is to better management of cloud data centres and edge computing environments.

Work that builds on this paper can be in different areas and sections. They could be further explored using our experiments and parameters. They are:

- Different ML algorithms: in this paper, we used the ML algorithm SVM. Therefore, we could investigate other ML algorithms such as linear regression and Naïve Bayes (NB). In addition, we could further investigate SVM regression by changing the parameters of the algorithm to find the appropriate factors for the equation of SVM regression.

- Different datasets: the experiments in this paper were done with the Google trace dataset. Therefore, we could implement our work with different datasets such as Alibaba open dataset or dataset of Amazon cloud. Moreover, such work could be done on the private clouds of organisations that have their own datasets.

- Scalability of the algorithm: the evaluation of our paper was limited to scheduling 4 VMs to 4 PHs. The scalability of our paper was one of the critical points, as it raises the question of how our algorithm could be scaled to schedule, for example, 1000 VMs to the cloud.

- Memory usage: as is mentioned in the limitation section (section 5.4), this work considered only CPU utilisation as the workload. Therefore, we could investigate further SVM regression for CPU and memory usage as the upcoming workloads.

- Edge environments: our paper was conducted for use in the real cloud. We could investigate furthermore how our solution functions in an edge computing environment.

References

[1] Badger, M.L., Grance, T., Patt-Corner, R. and Voas, J.M. Cloud Computing Synopsis and Recommendations. *National Institute of Standards & Technology*, 2012, pp. 1-81.

[2] Mell, P. and Grance, T. The NIST definition of cloud computing. Special Publication 800-145. Gaithersburg: National Institute of Standards and Technology. 2011, pp.1-7.

[3] Kumar, E.M. Cloud Computing in Resource Management. *International Journal of Engineering and Management Research (IJEMR).* 2018, **8**(6), pp.93-98.

[4] Kaur, D. and Sharma, T. Scheduling Algorithms in Cloud Computing. *International Journal of Computer Applications.* 2019, **975**, pp.16-21.

[5] Buyya, R., Srirama, S.N. and Bahsoon, R. A Manifesto for Future Generation Cloud Computing. 2018, pp.1-51.

[6] Hu, Y., Deng, B., Peng, F. and Wang, D. Workload Prediction for Cloud Computing Elasticity Eechanism. In: *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*: IEEE, 2016, pp.244-249.

[7] Kumar, J. and Singh, A.K. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems.* 2018, **81**, pp.41-52.

[8] Islam, S., Keung, J., Lee, K. and Liu, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems.* 2012, **28**(1), pp.155-162.

[9] Xia, B., Li, T., Zhou, Q.-F., Li, Q. and Zhang, H. An Effective Classification-based Framework for Predicting Cloud Capacity Demand in Cloud Services. *IEEE Transactions on Services Computing.* 2018, pp.1-13.

[10] Flora, H.K. and Chande, S.V. A Systematic Study on Agile Software Development Methodologies and Practices. *International Journal of Computer Science and Information Technologies.* 2014, **5**(3), pp.3626-3637.

[11] Uikey, N. and Suman, U. Tailoring for agile methodologies: A framework for sustaining quality and productivity. *International Journal of Business Information Systems.* 2016, **23**(4), pp.432-455.

[12] Reiss, C., Wilkes, J. and Hellerstein, J.L. *Google cluster-usage traces: format+ schema,* 2011.

[13] Moreno-Vozmediano, R., Montero, R.S., Huedo, E. and Llorente, I.M. Efficient resource provisioning for elastic Cloud services based on machine learning techniques. *Journal of Cloud Computing.* 2019, **8**(1), p.5.

[14] Alpaydin, E. *Introduction to machine learning.* Cambridge: MIT press, 2020.

[15] Flake, G.W. and Lawrence, S. Efficient SVM Regression Training with SMO. *Machine Learning.* 2002, **46**(1), pp.271-290.

[16] Markov, Z. and Russell, I. An introduction to the WEKA data mining system. *ACMSIGCSE Bulletin.* 2006, **38**(3), pp.367-368.

[17] Sammut, C. and Webb, G.I. eds. *Encyclopedia of Machine Learning*. Mean absolute error. Boston, MA: Springer US, 2010.

[18] Usha, T. and Balamurugan, S.A.A. Seasonal Based Electricity Demand Forecasting Using Time Series Analysis. *Circuits and Systems.* 2016, **7**(10), pp.3320-3328.

[19] Djemame, K. "Cloud Resource Management and Scheduling". COMP580 Cloud Computing. University of Leeds, 2020.

[20] Djemame, K. "Introduction to Cloud Computing: Enabling Technologies and Distributed System Models (2)". COMP580 Cloud Computing. University of Leeds, 2020.

[21] Sun, X., Ansari, N. and Wang, R. Optimizing Resource Utilization of a Data Center. *IEEE Communications Surveys & Tutorials.* 2016, **18**(4), pp.2822-2846.

[22] Ylonen, T. and Lonvick, C. The secure shell (SSH) protocol architecture. *RFC 4251.* 2006, pp.1-29.

[23] Anon 2020. *Server monitoring. Zabbix.com.* [Online]. [Accessed 24 July 2020]. Available from: https://www.zabbix.com/server_monitoring.