Computers, Materials & Continua
DOI:10.32604/cmc.2021.017477
Article

Tech Science Press

A Hybrid Approach for Performance and Energy-Based Cost Prediction in Clouds

Mohammad Aldossary[*]

Department of Computer Science, College of Arts and Science, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia
[*]Corresponding Author: Mohammad Aldossary. Email: mm.aldossary@psau.edu.sa
Received: 31 January 2021; Accepted: 02 March 2021

Abstract: With the striking rise in penetration of Cloud Computing, energy consumption is considered as one of the key cost factors that need to be managed within cloud providers' infrastructures. Subsequently, recent approaches and strategies based on *reactive* and *proactive* methods have been developed for managing cloud computing resources, where the energy consumption and the operational costs are minimized. However, to make better cost decisions in these strategies, the performance and energy awareness should be supported at both Physical Machine (PM) and Virtual Machine (VM) levels. Therefore, in this paper, a novel hybrid approach is proposed, which jointly considered the prediction of performance variation, energy consumption and cost of heterogeneous VMs. This approach aims to integrate auto-scaling with live migration as well as maintain the expected level of service performance, in which the power consumption and resource usage are utilized for estimating the VMs' total cost. Specifically, the service performance variation is handled by detecting the underloaded and overloaded PMs; thereby, the decision(s) is made in a cost-effective manner. Detailed testbed evaluation demonstrates that the proposed approach not only predicts the VMs workload and consumption of power but also estimates the overall cost of live migration and auto-scaling during service operation, with a high prediction accuracy on the basis of historical workload patterns.

Keywords: Cloud computing; energy efficiency; auto-scaling; live migration; workload prediction; energy prediction; cost estimation

1 Introduction

Cloud Computing is an effective and emerging model that has changed the industry of Information Technology (IT), in which it can deliver a wide variety of services for their customers in the form of applications, platforms and infrastructures. Nevertheless, the wide adoption of Cloud Computing leads to increase the number of customers and thereby increase the cloud providers' operational costs [1–5]. Therefore, minimizing the operational costs for the Cloud services has attracted considerable attention from industry and academic researchers in recent years, where numerous cost approaches and techniques (e.g., *on-demand, subscription* and *auction* cost models) have been developed by providers which can significantly affect the adoption of Cloud Computing industry. However, these approaches are sophisticated and have a set of limitations because the customers are charged on the basis of pre-defined tariffs for the resources, even if they have not used them [6–8]. In addition, the variation of energy cost is not considered in these approaches [9,10]. With the increasing cost of electricity, energy consumption is considered as one of the major cost factors that has a great effect on the Cloud infrastructure's operational cost [1–3,11]. Therefore, building a novel cost approach for Cloud services that are adapted to the energy costs is challenging and has attracted the attention of many researchers [1–3].

The consumed energy at the clouds is dependent on two main factors, which are the physical resources' efficiency and the strategies employed to manage these resources [12,13]. Consequently, numerous *reactive and proactive-based* methods are used to manage Cloud resources in an efficient way such as *dynamic consolidation* and *resource provisioning* [14]. For example, when the workload exceeds a specific threshold, (i.e., 95% of CPU utilization), the right decision is made (e.g., VMs migration or resources scaling) to avoid the degradation of service performance. Indeed, *proactive-based* methods can make the corrective decisions (e.g., auto-scaling, live migration and re-allocation) at earlier stages, in which the violation of Service Level Agreement (SLA) is prevented, and the service performance is maintained acceptable. Therefore, studying and understanding the impact of these decisions is important to develop cost-efficient strategies and energy efficient resource allocation methods. Moreover, estimating future Cloud services' cost can help service providers for making effective-cost decisions and offering suitable services that meet the requirements of their customers. Furthermore, recent developments in the management of Cloud paradigm at different levels and the reduction of energy consumption have received attention and thereby reduce the operational expenditure (OPEX) costs for the Cloud providers.

The aim of this paper to overcome the identification's challenge of cost-effective approaches for cloud services by enabling energy consumption's awareness, performance variation and the virtual level cost in the environment of Cloud. In addition, the result of this work can be integrated with *reactive* and *proactive* resource management methods for making effective-cost decisions in which it is reinforced by performance and energy awareness to manage Cloud resources efficiently. Moreover, the consumed energy is reduced and then the total cost of Cloud providers is minimized while the service performance is maintained. The contributions reported in this paper can be summarized as follows:

- A novel hybrid approach is introduced for predicting the VMs total cost, energy consumption and performance variation, where auto-scaling and live migration are integrated in order to provide cost-effective strategies. Additionally, the trade-off among cost, energy and performance in the cloud environment are considered.
- Set of models and algorithms are proposed for enhancing VMs consolidation and resource provisioning techniques, as well as supporting energy consumption's awareness, performance variation and cost in the infrastructure of Cloud.
- The evaluation results based on a real Cloud testbed proved the usability of the proposed approach and verified the capability of the prediction models.

The reminder of this paper is organized as follows: Section 2 introduces the hybrid approach for performance and predication cost of energy that integrates VMs auto-scaling with live migration. The experimental setup and design are presented in Section 3. This is followed by the evaluation and results discussion in Section 4. Finally, Section 5 concludes this paper and discusses future steps.

2 Integration of VMs Auto-Scaling with Live Migration: A Hybrid Approach

Recently, resource provisioning and VMs consolidation are used to address workload fluctuations issues, in the cloud environment. From on hand, the resource provisioning-based solutions (e.g., auto-scaling) can provide VMs with needed additional resource capacity to satisfy the requirements of Quality of Service (QoS). For instance, when one or more VMs are detected as overloaded (e.g., the workload surpasses the predefined percentage of upper threshold), the VMs should be scaled up/out to meet the application demands. Generally, VMs auto-scaling are categorized into two main types, namely, *vertical scaling* (i.e., scale-up/resizing) and *horizontal scaling* (i.e., scale-out) [15–17]. In the case of *vertical scaling*, the resources (e.g., vCPUs and memory) are added into VMs, while in the *horizontal scaling* case, an additional VMs are created. Note that, the additional resources are determined regarding the application requirements. Nevertheless, the second scaling type (i.e., *horizontal scaling*) requires a few minutes to be initiated [15,18,19], which is unacceptable for real-time and delay-sensitive computation [20,21]. Additionally, there are extra costs for vertical and horizontal scaling [17] which are scaling time (booting/rebooting), new VMs' license fees and energy overhead that need further consideration [16]. From the other hand, VM consolidation-based solutions (e.g., live migration) can improve the resource utilization and achieve energy efficiency in Clouds, in which the live migration process allows VMs to be moved from one PM to another without service interruption [22]. This approach plays an important role to balance the load among the PMs and reduces the overall energy consumption. For instance, when a host is detected as underloaded (e.g., the workload less than the predefined percentage of lower threshold), it is a candidate for being switched off or to enter power saving mode. However, the process of live migration is considered as a resource-intensive operation [23], in which the migrating VMs' service performance, and the services running on other VMs, are effected [24–27]. Moreover, it's needs to take attention that there are extra costs for the migration process [17], which includes the migration time and energy overhead [28,29]. Therefore, studying and understanding the influence of VMs auto-scaling and live migration is important to develop cost-effective strategies for Cloud services.

Numerous approaches for resource provisioning and VM consolidation independently emerged in the literature [16,17,21,24,25,30–32] with the goal of balancing the load, increasing the capacity of VMs resources and reducing the energy-related costs. To minimize the operational costs while achieving performance objectives, Cloud providers may automatically perform VMs consolidation and resource provisioning to match workload changes and prevent any performance loss. Indeed, *a proactive-based* framework can make the preventive actions on-the-fly (e.g., VMs auto-scaling, migrating and re-allocating) at earlier stages, in which the degradation of service performance is avoided. In addition, the framework's effectiveness depends on possible actuators/decisions to be implemented at service operation. This solution would enable Cloud providers to better use of their infrastructure in terms of maintaining service performance, reducing power consumption and operating cost. Besides, estimating future Cloud services' cost supports the service providers for offering appropriate services that satisfy the requirements of their customers [33].

Therefore, the proposed framework in [34] has been extended to support a new hybrid approach for predicting the total cost of heterogeneous VMs, considering their energy consumption and performance variation, as depicted in Fig. 1. More specifically, the costs of auto-scaling and live migration processes are integrated to determine the decision, in which the issues related to quality characteristics (e.g., energy consumption and application performance) are considered.

In addition, Autoregressive Integrated Moving Average (ARIMA) model is utilized to predict the workload of PMs/VMs in order to handle the performance variation of the applications and perform the most effective decision(s) (e.g., auto-scaling, live migration or both). Moreover, regression models exploit VMs and PMs workload's correlation to predict the VMs power consumption for an efficient allocation/re-allocation of the VMs. Consequently, on the basis of the predicted workload and energy consumption for each VM, VMs' total cost caused by the most effective decision(s) can be estimated. Finally, to reach this goal, numerous steps are needed to detect PMs' workload (i.e., underloaded and overloaded), predict PMs/VMs workload and consumption of power, and thereby estimating the scaled/migrated VMs' total cost, that are described in detail in the following subsections.

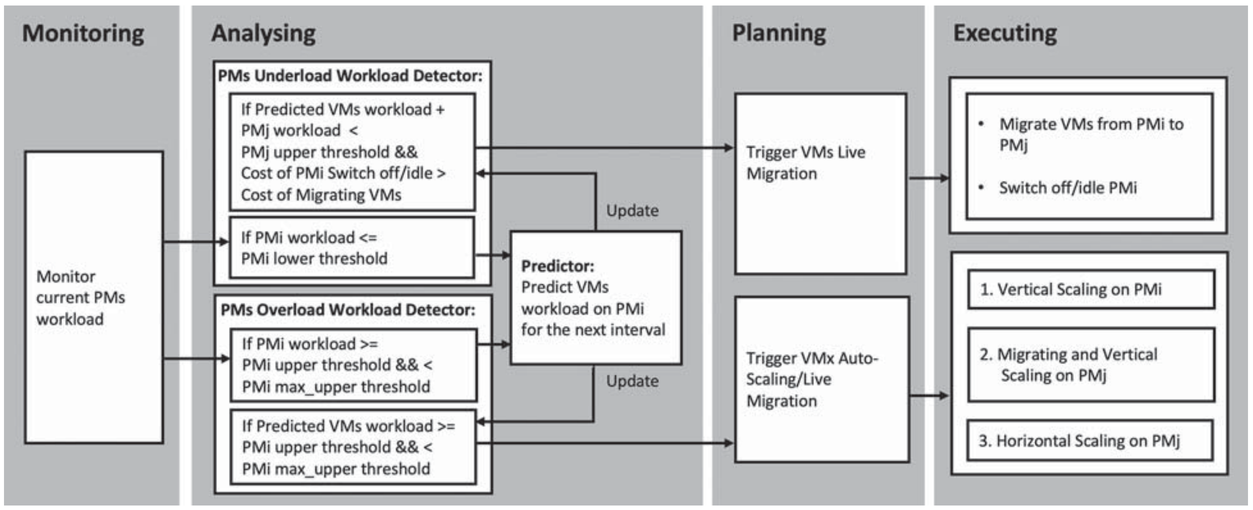
Monitoring
Analysing
Planning
Executing
Monitor current PMs workload
PMs Underload Workload Detector:
If Predicted VMs workload + PMj workload < PMj upper threshold && Cost of PMi Switch off/idle > Cost of Migrating VMs
If PMi workload <= PMi lower threshold
PMs Overload Workload Detector:
If PMi workload >= PMi upper threshold && < PMi max_upper threshold
If Predicted VMs workload >= PMi upper threshold && < PMi max_upper threshold
Predictor: Predict VMs workload on PMi for the next interval
Update
Update
Trigger VMs Live Migration
Trigger VMx Auto-Scaling/Live Migration
Migrate VMs from PMi to PMj
Switch off/idle PMi
1. Vertical Scaling on PMi
2. Migrating and Vertical Scaling on PMj
3. Horizontal Scaling on PMj

Figure 1: A hybrid approach for performance and energy-based cost prediction

2.1 PMs Performance Detection

Step 1: The threshold percentages of PMs' CPU utilization and RAM usage are determined (i.e., lower, upper and max_upper; 25%, 85% and 95%, respectively) and the workload of PMs is monitored periodically. Algorithm 1 shows the detailed process for detecting the underloaded and overloaded PMs. This Algorithm combines two sub-algorithms: 1) live migration with VMs re-allocation in order to switch the underloaded host to power saving mode, hence save energy-related costs. Also, this mechanism aims to minimize the overall cost of migration by re-allocation the VMs to the most energy efficient host (if possible), as presented in Algorithm 2; and 2) an integration of auto-scaling, live migration and re-allocation in order to prevent the host to be overloaded. This mechanism would help to select the most cost-effective action(s) in order to minimize VMs' total cost that cased by scaling and migration decisions, as presented in Algorithm 3. The list of the algorithms notations and their definitions are summarized in Tab. 1.

Step 2: The underloaded PMs is detected through Algorithm 1 and then appropriate actions are made such as live migration and re-allocation to save cost. Therefore, if the PMi workload ($\sum_{i=1}^{n} VMs\ Workload$) is less than or equals to the threshold lower percentage (e.g., 25%), then ARIMA model is utilized for predicting next time interval VMs' workload (e.g., every 5 min) on the basis of historical patterns of workload (see *Step 4*). Indeed, this process can detect the PMi with the underloaded workload in advance, and thereby can migrate the VMs and switch PMi to power saving mode. Afterward, if the next time interval VMs' predicted workload is still less than or equals to the lower threshold, then VMs live migration and re-allocation are performed using Algorithm 2.

Also, this algorithm (i.e., Algorithm 2) is used to select a matching destination PM*j* to host the migrated VMs, and to check whether the cost incurred by VMs live migration is less than the cost of switching the source PM*i* to power saving mode. To do so, PMs are descending sorted regarding their energy efficiency. This is aimed to migrate the VMs to the host with the most appropriated energy efficient. In this regard, the estimation of the energy efficiency for both host (i.e., source PM*i* and destination PM*j*) can be calculated as: PM Power $= \frac{PM\mathrm{i}(idle\ power\ of\ the\ source)}{PM\mathrm{j}(idle\ power\ of\ the\ destination)}$. For example, if the PM power > 1, the destination host is more energy efficient than the source; if the PM power $= 1$, the destination host is similar to the source in terms of the energy efficient and if the PM power < 1, the destination host is less energy efficient than the source.

Algorithm 1: PMs Underload/Overload Workload Detector and Performance Prediction

```
Initialise: PM workload = ($\frac{U\_CPU\_PM}{C\_CPU\_PM}$, $\frac{U\_RAM\_PM}{C\_RAM\_PM}$);
PM lower threshold = 0.25×(C_CPU_PM, C_RAM_PM);
PM upper threshold = 0.85×(C_CPU_PM, C_RAM_PM);
PM max_upper threshold = 0.95 × (C_CPU_PM, C_RAM_PM);
VM workload = ($\frac{U\_CPU\_VM}{C\_CPU\_VM}$, $\frac{U\_RAM\_VM}{C\_RAM\_VM}$);
Predicted VM workload = null;
Predicted $\sum_{i=1}^{n}$VMs workload = null;
Predicted VMs list workload = empty.
Input: PMs list.
for each (PMi in PMs list) do
  if (PMi workload $\leq$ PMi lower threshold) then
    for each (VMx in PMi) do
      Predicted VMx workload $\leftarrow$ predict VMx workload for the next interval using the ARIMA model.
      Predicted $\sum_{i=1}^{n}$VMs workload $\leftarrow$ Predicted VMx workload ++; // The sum of the predicted VMs workload on PMi.
    end for
      if (Predicted $\sum_{i=1}^{n}$VMs workload $\leq$ PMi lower threshold) then  // Underloaded PM
        Perform Algorithm 2.
      end if
  else
    if (PMi workload $\geq$ PMi upper threshold) && (PMi workload $<$ PMi max_upper threshold) then
      for each (VMx in PMi) do
        Predicted VMx workload $\leftarrow$ predict VMx workload for the next interval using the ARIMA model.
        Predicted $\sum_{i=1}^{n}$VMs workload $\leftarrow$ Predicted VMx workload ++; // The sum of the predicted VMs workload on PMi.
        Predicted VMs list workload $\leftarrow$ Predicted VMx workload ++; // The list of the predicted VMs workload on PMi.
      end for
        if (Predicted $\sum_{i=1}^{n}$VMs workload $\geq$ PMi upper threshold) && (Predicted $\sum_{i=1}^{n}$VMs workload $<$ PMi max_upper threshold) then  // Overloaded PM
          Perform Algorithm 3.
        end if
    end if
  end if
end for
```

Starting with the lowest idle power for PM*j* (the most energy efficient host), and then check if PM*j* has enough resources to meet the migration requirements, while simultaneously ensuring that the destination host PM*j* does not exceed the upper percentage threshold after allocating the migrated VMs. This algorithm ensures: 1) the destination PM*j* is not overloaded after VMs migration process, 2) the source PM*i* will be switched to power saving mode once the migration takes place in order to save cost.

```
Algorithm 2: Switching PMs to Power Saving Mode
Initialise: PM workload =($\frac{U\_CPU\_PM}{C\_CPU\_PM}$, $\frac{U\_RAM\_PM}{C\_RAM\_PM}$);
PM upper threshold = 0.85 × (C_CPU_PM, C_RAM_PM);
PM power = $\frac{PMi(idle\ power\ of\ the\ source)}{PMj(idle\ power\ of\ the\ candidate)}$; // To check the energy efficiency
Decision = null.
Input: PMs list; // Assuming all the PMs in running/active state
Predicted $\sum_{i=1}^{n}$ VMs workload; // From Algorithm 1
Cost of Switching PMi to Power Saving Mode;
Cost of Migrating VMs. // To any targeted host PMj
Output: Decision.
1: Sort the PMs list in decreasing order of the PM power;
2:   for each (PMj in PMs list) do
3:     if ((Predicted $\sum_{i=1}^{n}$ VMs workload + PM j workload) < PMj upper threshold) &&
         (Cost of Switching PMi to Power Saving Mode > Cost of Migrating VMs) then
4:       Decision ← perform VMs migration to target host PMj;
5:       Switch PMi to Power Saving Mode.
6:       break;
7:     end if
8:   end for
9: return Decision
```

Step 3: The overloaded PMs is detected through Algorithm 1 and then the candidate VMs that need to be scaled/migrated are identified. Therefore, if the workload value of PMi within the range of [upper and max_upper percentage of threshold], then ARIMA model is utilized for predicting the next time interval of VMs workload in PMi (e.g., every 5 min) on the basis of historical patterns of workload (see *Step 4*). Obviously, this process can detect the PMi with the overloaded workload, and thereby can perform preventive actions such as VMs auto-scaling and live migration. Further, this mechanism would help to control the number of scaling and migrations decisions in order to prevent unnecessary scaling/migration incurred by small workload peaks (false alarm). Afterward, if the workload of predicted VMs is still within the range of [upper and max_upper percentage of threshold] for the next interval, VMs auto-scaling/live migration process is employed using Algorithm 3.

The proposed Algorithm 3 combines the auto-scaling (vertical/horizontal scaling) with live migration for obtaining the appropriated cost-effective decision(s). The overloaded VMs is firstly scaled/migrated (e.g., resize VMs, migrate existing VMs and resize them, or initiate new VMs), then select appropriate destination PMj for hosting it. To achieve this, the following set of conditions are tested in the same order and perform the associated actions: 1) if PMi (the source host) has enough resources to meet the scaling requirements, then the vertical scaling is performed on this PMi (hint: vertical scaling is restricted by PMi capacity [17,35,36]); 2) if PMi does not have enough resources, the PMs are descending sorted regarding their energy efficiency, as described in *Step 2*. After sorting the PMs based on their energy efficiency, the *migration and vertical scaling decision* is performed in order to firstly *migrate* the overloaded VMs to appropriate host PMj and then *vertically scaling* them. It also checks if PMj has enough resources to meet the *migration* and *scaling* requirements, while at the same time ensuring that the destination host PMj does not exceed the upper percentage threshold after allocating the migrated VMs along with their scaling requirements (the additional resources); otherwise, 3) horizontal scaling is made on PMj in a similar manner as the previous action by placing the new VM to an appropriate destination. This algorithm ensures: 1) the destination PMj is not overloaded after VMs scaling and migration processes, 2) workload of the source PMi is significantly decreased once scaling/migration process is performed, and 3) minimize VMs' total cost caused by scaling/migration decisions.

Algorithm 3: Integrate Auto-Scaling Decisions with Dynamic VMs Allocation

```
Initialise: PM workload =($\frac{U\_CPU\_PM}{C\_CPU\_PM}, \frac{U\_RAM\_PM}{C\_RAM\_PM}$);
PM upper threshold = 0.85 $\times$ (C_CPU_PM, C_RAM_PM);
PM max_upper threshold = 0.95 $\times$ (C_CPU_PM, C_RAM_PM);
PM power = $\frac{PMi(idle\ power\ of\ the\ source)}{PMj(idle\ power\ of\ the\ candidate)}$; // To check the energy efficiency
Candidate PM = false;
VM workload = ($\frac{U\_CPU\_VM}{C\_CPU\_VM}, \frac{U\_RAM\_VM}{C\_RAM\_VM}$); // From Algorithm 1
Candidate VM = false;
Overloaded VM = null.
VM Resource Increments = (I_CPU_VM, I_RAM_VM) = (null, null);
Decision = null.
Input: PMs list; // Assuming all the PMs in running/active state
Predicted VMs list workload; // From Algorithm 1
Predicted VM workload; // From Algorithm 1
Output: Decision.
 1: Sort the Predicted VMs list workload on PMi in a decreasing order;
 2:   for each (VMx in Predicted VMs list workload) do
 3:     if (Predicted VMx workload > VMx workload) then
 4:       Overloaded VMx = Predicted VMx workload;
 5:       VMx Resource Increments = Predicted VMx workload − VMx workload;
 6:       Candidate VM = true;
 7:       break;
 8:     end if
 9:   end for
10: if (Candidate VM = false) then
11:     break; // no candidate VM is found
12:   else
13:     if ((PMi workload + VM Resource Increments) < PMi max_upper threshold) then
14:         // The resource availability on the same host is met (Resize VMx)
15:       Decision ← perform VMx vertical scaling based on (VMx Resource Increments);
16:     else // Lack of resources on the same host
17:       Sort the PMs list in decreasing order of the PM power;
18:       for each (PMj in PMs list) do
19:         if ((PMj workload + Overloaded VMx) < PMj upper threshold) then
20:           Decision ← perform VMx migration to target host PMj; and
                        perform VMx vertical scaling based on (VMx Resource Increments); // Migrate existing VM and resize it
21:           Candidate PM = true;
22:           break;
23:         end if
24:       end for
25:     if (Candidate PM = false) then
26:       for each (PMj in PMs list) do.
27:         if ((PMj workload + VM Resource Increments) < PMj upper threshold) then
28:           Decision ← perform VMx horizontal scaling based on (VMx Resource Increments); // Create a New VM
29:           break;
30:         end if
31:       end for
32:     end if
33:   end if
34: end if
35: return Decision.
```

Algorithm 4: Self-configuration - Resizing/Creating VMs

```
Initialise: Scaling VM = null.
Input: Decision; // From Algorithm 3 (Vertical or Horizontal Scaling)
VMs size list; // List of VMs sizes set by Cloud providers
VM size. // Based on the predefined VM-sizes list such as (small, medium and large)
VM Resource Increments = (I_CPU_VM, I_RAM_VM) // From Algorithm 3
Output: Scaling VM.
1: Sort the VMs size list in increasing order of the VM sizes;
2: for each (VM size i in VMs size list) do
3:   if (Resource Increments = VM size i) then // To ensure that the predefined VM capacity is matched with the actual load
4:     Scaling VM = VM size i; // Resize or Create using a predefined VM size based on the Decision from Algorithm 3
5:   else
6:     if (Resource Increments < VM size i) then
7:       Scaling VM = Resource Increments; // Resize or Create using a Self-configuration VM size based on the Decision from Algorithm 3
8:       break;
9:     end if
10:  end if
11: end for
12: return Scaling VM.
```

Table 1: Summary of notations

Notation	Description
PMi	Source physical machine
PMj	Destination physical machine
VMx	Candidate VM for migration/the overloaded VM for scaling
C_CPU_PM	PM's CPU capacity
C_RAM_PM	PM's memory capacity
U_CPU_PM	Used PM's CPU capacity ($\sum_{y=1}^{VM\_Count}$(vCPU))
U_RAM_PM	Used PM's memory capacity ($\sum_{y=1}^{VM\_Count}$(RAM))
C_CPU_VM	VM's CPU capacity
C_RAM_VM	VM's memory capacity
U_CPU_VM	Used VM's CPU capacity
U_RAM_VM	Used VM's memory capacity
I_CPU_VM	Increment VM's CPU capacity
I_RAM_VM	Increment VM's memory capacity

For VMs scaling, Algorithm 4 is utilized to select the appropriate VMs size for cost-effective scaling on the basis of closest cloud provider-defined instance sizes set (e.g., small, medium and large VM). However, over-provisioning may be happened in this mechanism (e.g., in the case of there is a difference between the requested scaling resources and the predefined cloud providers' instance sizes). Subsequently, the extra-resources are wasted, where set of capacity is created and useless, and thereby more money might paid without any benefit [17,37], which is not the auto-scaling goal of VMs. Moreover, wasted resources may lead to an increase in energy costs because of their under-utilization and a decrease in the revenue of Cloud providers, as the number of resource requests acceptable may decrease. Therefore, motivated by these considerations, a new self-configuration algorithm is proposed for resizing/creating VMs on the basis of the right requested resources size, where the self-configuration algorithm aims to allocate adequate resources to VMs and avert resources' over-provisioning. In this way, this algorithm (i.e., Algorithm 4) can maximize cloud providers' resources usage and thereby their profits are maximized, and only the actually used resources will be paid by customers.

2.2 VMs Workload Prediction

Step 4: In this step, VMs workload is predicted for the next time interval using ARIMA model, which is expressed as the number of VMs requested and their capacity (i.e., vCPUs, memory, disk and network) for executing the application, and can be calculated based on historical workload patterns retrieved from a knowledge database. Different patterns of workload can be achieved in Cloud applications based on the behaviors of customers usage, which consume different power depending on the utilized resources. As stated in [38], there are several workload patterns such as *static, periodic, continuously changing, unpredicted, and once-in-a-lifetime*. The static workload pattern can be easily predicted, but there are many challenges that can obstruct the workload prediction when using other patterns. For example, other patterns may reflect temporary fluctuations of the workload such as continuously changing and once-in-a-lifetime or may be difficult to predict in advance such as the unpredicted pattern. These patterns do not necessarily occur in data centers on a daily basis [32]. Therefore, it is essential to have approximated workload patterns that occur in the time series history to achieve a high prediction accuracy [32]. Thus, the periodic workloads can be more appropriate and precise to allow Cloud services to quickly scaling or descaling the capacity to meet demand and dynamically control the cost of the infrastructure. Therefore, the simulated periodic workload pattern is considered for the historical data to be used in this work. Due to the sophistication and ARIMA model accuracy, it is widely used in different fields, such as economics and finance [39], where a further details exist in [39]. After predicting the workload of VMs using a model of ARIMA, which is based on historical data, the workload of PMs as well as the power consumption of PMs/VMs are predicted using regression models.

2.3 PMs Workload Prediction

Step 5: Workload prediction of PMs is represented as (PMs CPU utilization), which can be calculated based on the relationship between vCPUs number and the PM CPU utilization for the heterogeneous PMs, as shown in Figs. 2–4, respectively.

In this step, a model of linear regression is utilized for predicting the PMs CPU utilization on the basis of the ratio of used vCPUs requested number by VMs, considering their current workload as the PM can already running other VMs [40,41]. The following Eq. (1) is used.

$$\mathrm{PM}x_{PredUtil} = \left(\alpha \times \left(\sum_{y=1}^{VMCount} (\mathrm{VM}y_{ReqvCPUs} \times \frac{VMy_{PredUtil}}{100})\right) + \beta\right) + (PMx_{CurrUtil} - PMx_{IdleUtil}) \tag{1}$$

where $PMx_{PredUtil}$ indicates the predicted CPU utilization of PM; α and β denote the slope and the CPU utilization's intercept, $VMy_{ReqvCPUs}$ and $VMy_{PredUtil}$ denote the number of vCPU requested and the predicted utilization for each VM. The $PMx_{CurrUtil}$ and $PMx_{IdleUtil}$ denote the current and idle PM utilization, respectively.

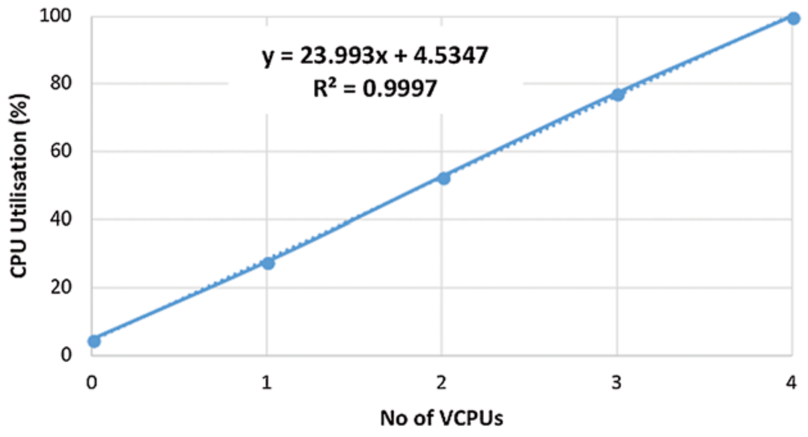

Figure 2: Number of vCPUs (VMx) *vs*. PM CPU utilization (source PMi), Host A

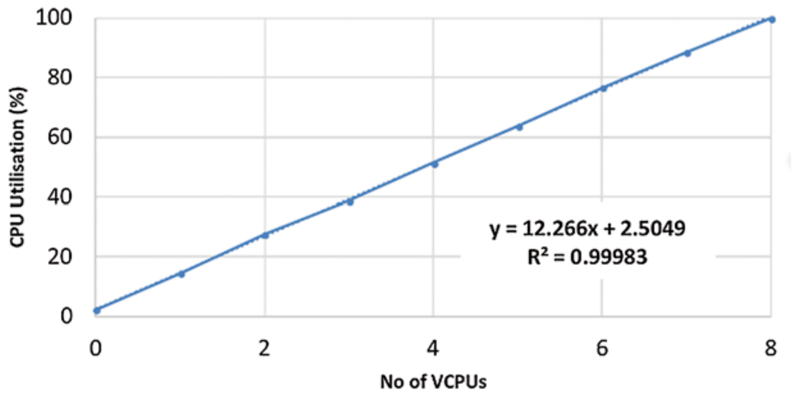

Figure 3: Number of vCPUs (VMx) *vs*. PM CPU utilization (destination PMj-most energy efficient PM), Host B

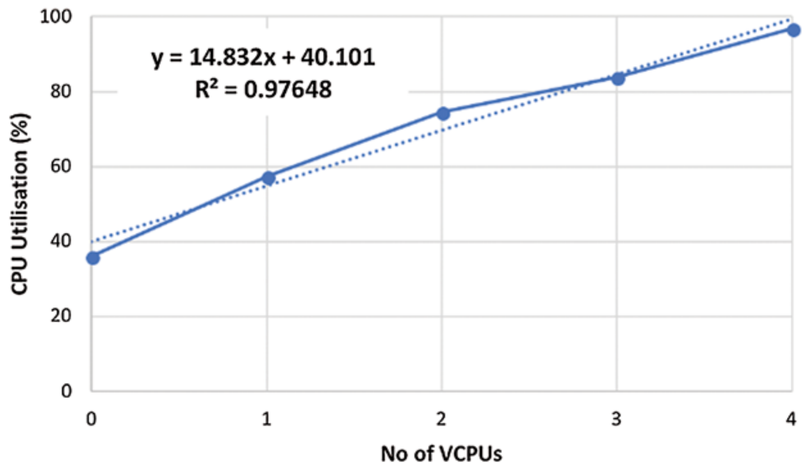

Figure 4: Number of vCPUs (VMx) *vs*. PM CPU utilization (destination PMj-less energy efficient PM), Host D

2.4 PMs Energy Consumption Prediction

Step 6: After the workload of PMs was predicted in the previous step, the power consumption of PMs is predicted in this step, which can be computed on the basis of the relationship between the predicted workload (CPU utilization) and power consumption of PM. Therefore, the considered PM should be characterized using regression models with respect to the correlation of their power consumption and CPU utilization, as shown in Fig. 5.

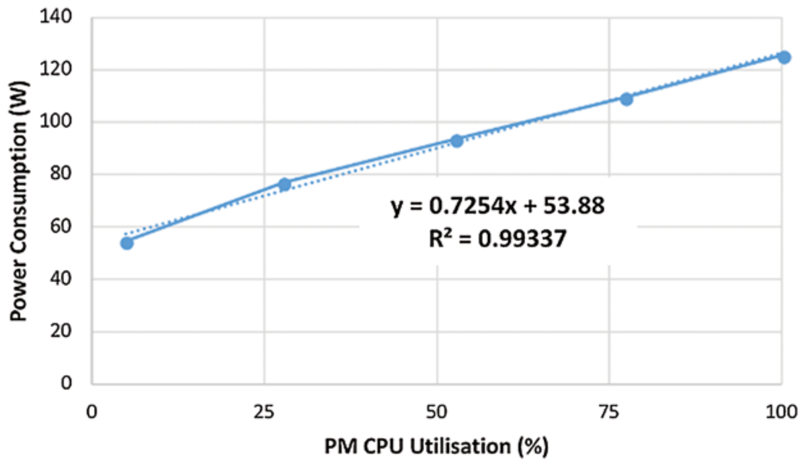

Figure 5: The PM CPU utilization *vs*. power consumption (source PMi), Host A

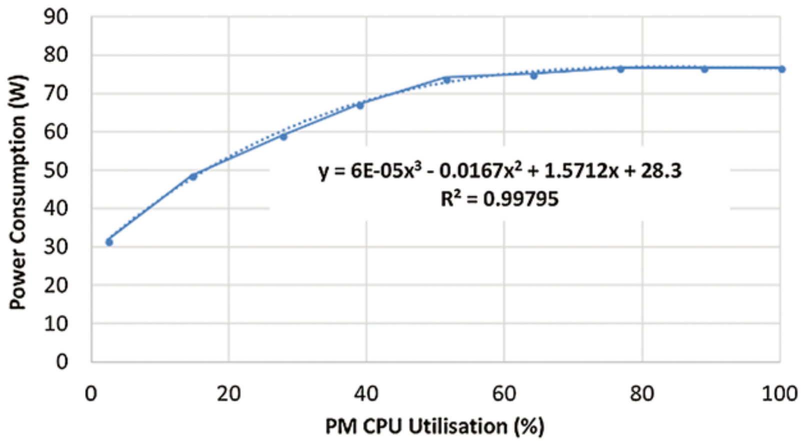

Figure 6: The PM CPU utilization *vs*. power consumption (destination PMj-most energy efficient PM), Host B

Consequently, the power consumption of the predicted PM, $PMx_{PredPwr}$ measured in Watt, can be linearly expressed using the CPU utilization of the predicted PMs, as shown in Fig. 5 and in Eq. (2). Where α and β denote the values of slope and interceptor obtained using regression relation, and $PMx_{PredUtil}$ denotes CPU utilization of the predicted PM.

$$PMx_{PredPwr} = (\alpha \times (PMx_{PredUtil}) + \beta) \tag{2}$$

However, all existing PMs do not necessarily follow this linearity relation due to the natural heterogeneity of PMs' resources, as shown for example in Figs. 6 and 7. Therefore, in this case, the correlation among power consumption and the targeted PM's CPU utilization can be characterized using other regression models, such as polynomial, as presented in Eq. (3).

$$\mathrm{PM}x_{PredPwr} = \left(\alpha(\mathrm{PM}x_{PredUtil})^3 + \gamma(\mathrm{PM}x_{PredUtil})^2 + \delta(\mathrm{PM}x_{PredUtil}) + \beta\right) \tag{3}$$

where α, γ and φ represent slopes, β denotes the intercept and $PMx_{PredUtil}$ denotes the CPU utilization of the predicted PM.

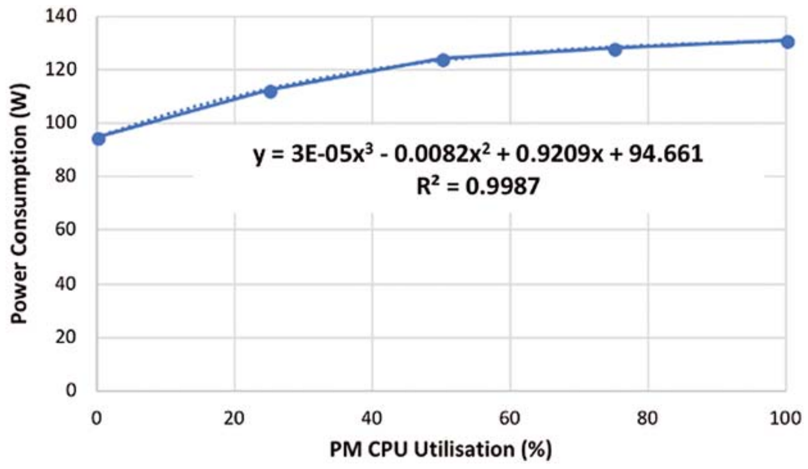

Figure 7: The PM CPU utilization *vs.* power consumption (destination PMj-less energy efficient PM), Host D

2.5 VMs Energy Consumption Prediction

Step 7: The goal of this step is to assign the PM predicted consumption of power to the newly requested VM, $VMx_{PredPwr}$, as well as to the running VMs on the physical host, which can be computed using Eq. (4).

$$\mathrm{VM}x_{Predpwr} = PMx_{IdlePwr} \times \left(\frac{\mathrm{VM}x_{ReqvCPUs}}{\sum_{y=1}^{VMcount} \mathrm{VM}y_{ReqvCPUs}}\right) + (PMx_{PredPwr} - PMx_{IdlePwr}) \times \left(\frac{\mathrm{VM}x_{(PredUtil \times ReqvCPUs)}}{\sum_{y=1}^{VMcount} \mathrm{VM}y_{(PredUtil \times ReqvCPUs)}}\right) \tag{4}$$

where $VMx_{PredPwr}$ denotes one VM's predicted power consumption in Watt. The $VMx_{ReqvCPUs}$ and $VMx_{PredUtil}$ denote the vCPU requested number and the predicted CPU utilization of VM, respectively. $\sum_{y=1}^{VMcount} VMy_{ReqvCPUs}$ denotes VM's total vCPU on the same PM. The $PMx_{IdlePwr}$ and $PMx_{PredPwr}$ denote the idle and predicted consumption of power for a single PM.

Note that, the energy providers are usually charged by Kilowatt per hour (kWh). Therefore, the consumption of power needs to be converted to energy based on the following Eq. (5).

$$VMx_{PredEnergy} = \frac{VMx_{PredPwr}}{1000} \times \frac{Time_s}{3600} \tag{5}$$

2.6 VMs Total Cost Estimation

Step 8: Finally, regarding the obtained values of predicted VMx resource usage and energy consumption in *Steps 4* and *7*, the VMx total cost is estimated in this step. In the case of VM migration, the total time $Time_s$ needed to migrate VMx can be expressed as follows:

$$T_{Mig} = (T_{Mig\_End} - T_{Mig\_Start}) \tag{6}$$

$$T_{Run\_Sou} = (T_{Run\_Sou\_Bef\_Mig} + T_{Mig}) \tag{7}$$

$$T_{Run\_Des} = (T_{Run\_Des\_Aft\_Mig} + T_{Mig}) \tag{8}$$

where T_{Mig} denotes the total migration time of VMx in seconds. T_{Mig_Start} and T_{Mig_End} denote the start and end time of the migration process. T_{Run_Sou} denotes the summation of VMx running time on the PMi before the migration process starts plus the migration time T_{Mig} itself, and $T_{Run\_Sou\_Bef\_Mig}$ denotes the VMx running time before migration. T_{Run_Des} denotes the VMx running time on the PMj during and after the migration process, and $T_{Run\_Des\_Aft\_Mig}$ denotes the VMx running time after migration. In the case of VM auto-scaling, the total time $Time_s$ needed to scale VMx can be expressed as follows:

$$T_{Scaling\_VMx} = (T_{End\_Scaling} - T_{Start\_Scaling}) \tag{9}$$

$$T_{Existing\_VMx} = \left(T_{End\_Run} - T_{Start\_Run}\right) - (T_{Scaling\_VMx}) \tag{10}$$

where $T_{Scaling\_VMx}$ denotes the VMx scaling time in seconds. $T_{Start\_Scaling}$ and $T_{End\_Scaling}$ denote the start and end time of scaling. $T_{Existing\_VMx}$ denotes the VMx running time before the scaling process starts. T_{Start_Run} and T_{End_Run} denote the start and end time for the operating task. For estimating the total cost of VMx based on the suitable action(s), Eq. (11) is proposed.

$$\begin{aligned} \mathrm{VM}x_{Est\_Cost\_PMi} &= \left(\left(\mathrm{VM}x_{ReqvCPUs\_PMi} \times \frac{VMx_{Pred\_U\_PMi}}{100}\right) \times (Cost\_vCPU \times Time_s)\right) \\ &\quad + \left(\mathrm{VM}x_{Pred\_RAM\_U\_PMi} \times (Cost\_GB \times Time_s)\right) + \left(\mathrm{VM}x_{Pred\_Disk\_U\_PMi} \times (Cost\_GB \times Time_s)\right) \\ &\quad + \left(\mathrm{VM}x_{Pred\_Net\_U\_PMi} \times (Cost\_GB \times Time_s)\right) + (VMx_{Pred\_Energy\_PMi} \times Cost\_kWh) \end{aligned} \tag{11}$$

where $VMx_{Est\_Cost\_PMi}$ denotes the VMx total estimated cost before and during the action is made on the source PMi. The $VMx_{ReqvCPUs\_PMi}$ denotes the vCPUs requested number for each VM and $VMx_{Pred\_U\_PMi}$ denotes the predicted CPU utilization for each VM, times vCPUs requested cost for a period of time (the time is based on the performed action). $VMx_{Pred\_RAM\_U\_PMi}$ denotes the predicted usage of memory, times the cost for that resource for a period of time. The notation of disk and network resources are similarly considered. $VMx_{Pred\_Energy\_PMi}$ denotes the VMx predicted energy consumption, times the energy providers-considered energy cost. Therefore, Eq. (11) can be used to estimate the VMx cost during and after the action is made at the destination PMj, while the resources of PMi (e.g., CPU, RAM, disk, network and energy) are substituted with PMj resources. Moreover, an additional license fee α is added for the new VM (i.e., constant £0.1/h.) in the case of making horizontal scaling decision. Thus, Eq. (12) can be used to obtain the VMx estimated cost, $VMx_{Total\_Est\_Cost}$, before and after the action is made.

$$\text{VM}x_{Total\_Est\_Cost} = \text{VM}x_{Est_{Cost_{PMi}}} + \text{VM}x_{Est\_Cost\_PMj} \tag{12}$$

3 Experimental Setup

In this section, the experimental details to critically evaluate the performance of the proposed approach is conducted, in which a real Cloud testbed is used. First, the cloud testbed and monitoring infrastructure are introduced. Afterward, experimental design is presented.

3.1 Cloud Testbed and Monitoring Infrastructure

The experiments are undertaken on a Cloud testbed composed of commodity Dell servers' cluster, in which each server is equipped with 16 GB RAM and 500 GB of SATA HDD storage, running Linux CentOS version 6.6 platform. In addition, four different PMs are considered on the cloud testbed, where three of them (Host A, C and D) equipped with an Intel Xeon CPU with four core X3430 and the last one (Host B) equipped with an Intel Xeon CPU with eight-core E3-1230 V2. Also, the testbed offers a share of Network File System (NFS), which runs on the cluster head node and provides with a 2 TB of VM images' storage. Moreover, this testbed is supported by Virtual Machine Manager (VMM) with OpenNebula [42] version 4.10.2, and used a Kernel-based Virtual Machine (KVM) [43] hypervisor version 4.0.1 built on Linux Kernel with version 2.6.32.24. Further, in this testbed, Host A is considered as the source host (PMi) and Host B, C and D are considered as the destination's hosts (PMj). Where Host B is the most energy efficient host, Host C is the similar host configuration to the source Host A, and Host D is the less energy efficient host.

In addition, Cloud testbed monitors and measures the usage of resources and energy. At the physical level of host, each PM is equipped with a WattsUp meter [44], for measuring the power consumption values per second, which then pushed to Zabbix (infrastructure monitoring tool) [45]. Besides, for each of the running PMs and VMs, the usage of resources (e.g., CPU, memory, network and disk) are monitored by Zabbix. Moreover, the power consumption of PMs and resource usage of VMs are transferred together to the proposed approach, which can measure the energy consumption as well as VMs' total cost.

Finally, with regard to VMs considered in this paper, Rackspace [46] is utilized as VMs configurations reference, to provide an extensive range of VM types, that allows customers plenty of flexibility to satisfy their needs. Additionally, in this paper, three types of VMs (i.e., small, medium and large) with different capacities are utilized. Furthermore, with regard to ElasticHosts [47] and VMware [48], the virtual resources' cost is determined, whereas they describe a service cost breakdown in detail as follows: 1 vCPU, 1 GB Memory, 1 GB Storage and 1 GB Network are respectively assigned £0.008/h, = £0.016/h, = £0.0001/h, = £0.0001/h; and energy cost = £0.14/kWh [49]. Note that customers are charged based on the resources they utilize in British Pound Sterling (GBP/£).

3.2 Design of Experiments

In this subsection, an experimental design is conducted to prove the capability of the proposed approach to detect and predict the underloaded/overloaded PMs in order to handle the service performance variation in a cost-effective manner. This approach also aims for predicting the workload and power consumption and can estimate the total cost associated with migrated and scaled VMs in the case of heterogeneous PMs are used for running. Additionally, the approach focuses on total saving cost which can be achieved if VMs are migrated/scaled to/on different hosts with different energy characteristics. In the experimental design, a real pattern of workload for Cloud applications is represented by generating historical data, in which all the resources (i.e., CPU, memory, network and disk) are stressed to their full utilization on different VMs types using the *Stress-ng* tool [50]. Then, guided by the intuition in [51], the time interval for each VM type's generated workload is divided into four slots (30 min each), in which three of them are used a set of prediction historical data, whereas the last one can be utilized as a set of testing data for evaluating the predicted results. The applying process of the proposed approach is as follows. Firstly, the underloaded and overloaded hosts are detected in order to handle the service performance variation. Then, the ***auto.arima*** R package's function [52] is used to predict the workload of VMs, in which ARIMA best fit model is automatically selected on the basis of Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) value. Afterwards, the process passes through the framework cycle [34] and takes into account the relationship among the physical and virtual resources for predicting VMs' consumption of power, in the case of several PMs are used for running. Finally, the most cost-effective decision(s) is performed, and the total cost is estimated for both migrated and scaled VMs on the basis of their power consumption and workload predicted.

4 Evaluation and Results Discussion

In this section, an extensive and quantitative evaluation of the proposed approach based on the performance and predication cost of energy are presented, in which VMs total cost during service operation is estimated with the consideration of migration and scaling decisions. First, the predicted workload results for VMs with three different types (i.e., small, medium and large) that operated on several PMs on the basis of historical periodic workload pattern, are introduced. Afterward, the results of VMs power consumption prediction with live migration and auto-scaling processes are discussed. Finally, the estimation cost of VMs is presented.

4.1 VMs Workload Prediction

Regarding Algorithm 1, the predictive model is employed to reduce the number of VM migration/scaling decisions in the case of PM*i* is underloaded/overloaded, while the predefined thresholds (i.e., lower, upper and max_upper) are satisfied, and thereby prevent the unnecessary migration and scaling incurred by small workload peaks (false alarm). Generally, with regard to Algorithm 2, VMs are migrated or re-allocated/allocated to the appropriate destination PM*j* (i.e., with sufficient resources and energy-efficient) in the case of PM*i* is underloaded, in which the predicted workload is less than or equals to the predefined lower threshold, and thereby PM*i* is switched to power saving mode and then the cost is saves. Furthermore, according to Algorithm 3, the most cost-effective scaling/migration decision(s) (e.g., resize VMs, migrate existing VMs and resize, or initiate new VMs) is performed and VMs are re-allocated/allocated to the selected destination PM*j* (i.e., with sufficient resources and energy-efficient) in the case of PM*i* is overloaded, where the workload predicted is with the range of [upper and max_upper percentage of threshold]. Also, the VMs predicted results in relation to the actual workload are illustrated in Figs. 8–10, where CPU, RAM, disk, and network usage are included. It is observed from the figures that the predicted and actual workload results of VMs for CPU, RAM and network are approximately matched where the utilization peaks are periodic. This is due to the fact that the ARIMA model is capable of captures the trend of historical seasonal and therefore gives an accurate prediction. Whereas the accuracy for predicted disk workload is less with respect to the results of CPU, network and RAM prediction. This is trace to the high variations of disk workload pattern of the generated historical periodic which is not nearly matched in each interval.

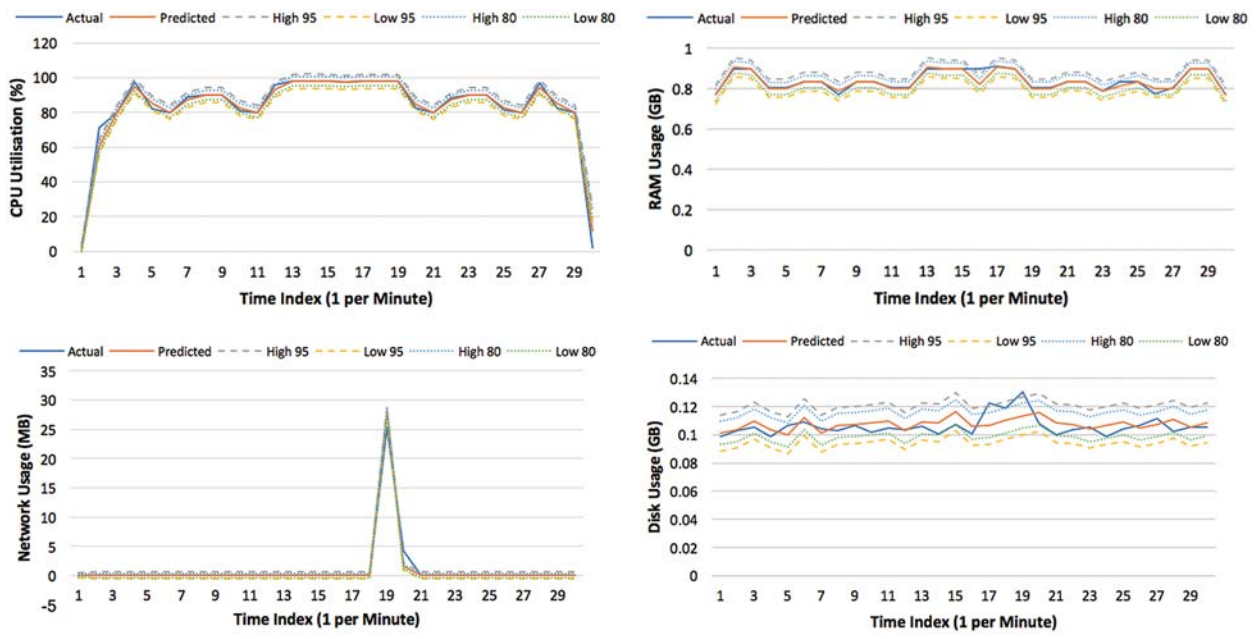

Figure 8: The workload prediction results for small VM

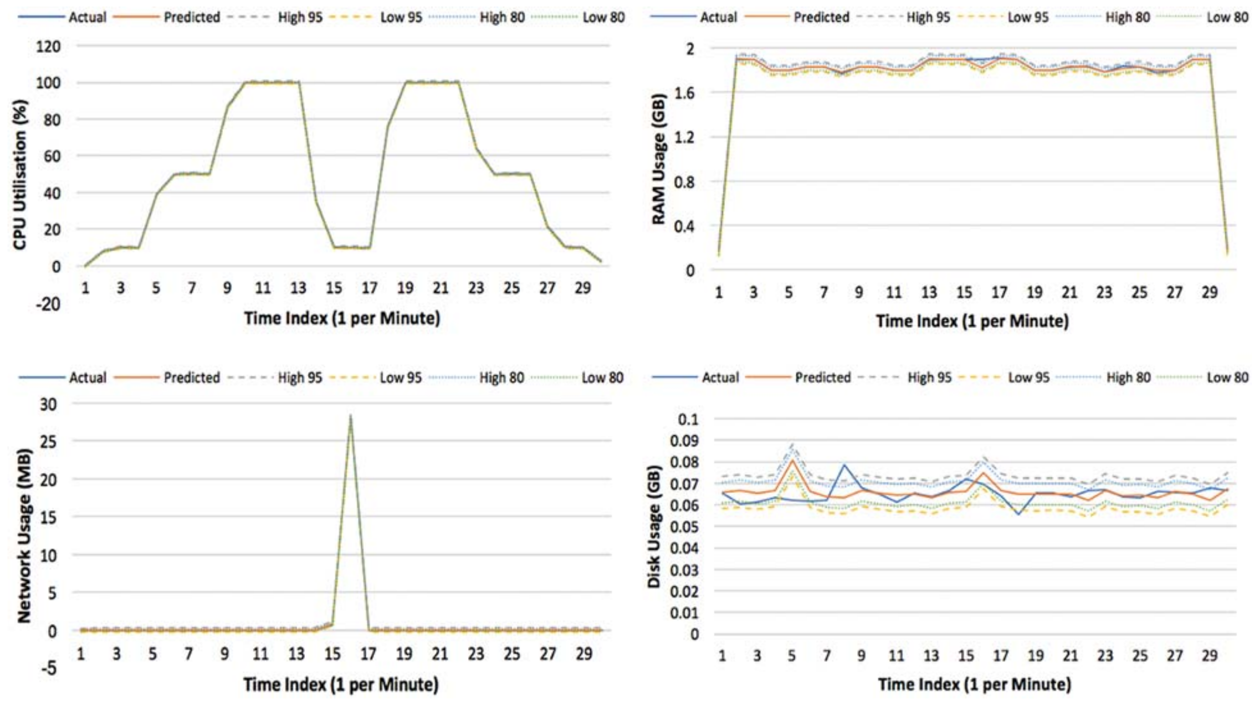

Figure 9: The workload prediction results for medium VM

Besides the mean values of VMs predicted workload, the results also illustrated the confidence intervals of each VM predicted workload with low and high percentage (i.e., 80% and 95%) based on the ARIMA model.

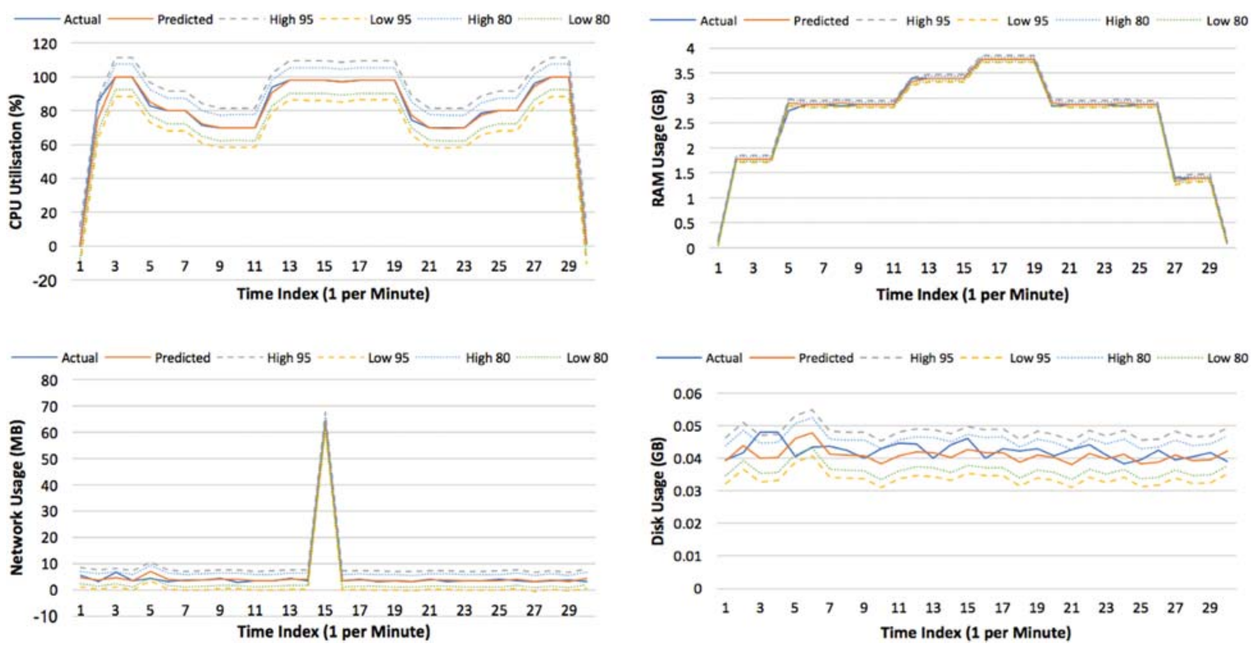

Figure 10: The workload prediction results for large VM

Meanwhile, Tab. 2 presents different number of metrics used to verify the accuracy of the prediction of CPU, RAM, disk and network workload using three VMs types (i.e., small, medium and large) on the basis of periodic workload patterns.

Table 2: Prediction accuracy for three types of VMs

VM Type	Parameters	ME	RMSE	MAE	MPE	MAPE
Small VM	CPU utilization	0.00486	1.7101	0.5652	−3.4611	4.978
	RAM usage	0.00167	0.0189	0.0055	0.1618	0.6585
	Disk usage	−0.0052	0.1869	0.0461	3.459	6.940
	Network usage	0.00072	0.0051	0.0030	0.64200	2.8612
Medium VM	CPU utilization	0.019355	0.2451	0.12275	−3.1443	3.576033
	RAM usage	0.001976	0.0189	0.00588	0.11509	0.333648
	Disk usage	0.000197	0.0940	0.01848	−8.96	9.5482
	Network usage	−0.00005	0.0030	0.00181	−0.2380	2.716369
Large VM	CPU utilization	0.437240	4.8481	1.39113	0.86261	2.095702
	RAM usage	−0.00097	0.0308	0.00791	−0.0621	0.328699
	Disk usage	−0.08418	1.4943	0.47049	−3.3323	11.57954
	Network usage	−0.00001	0.0028	0.00156	−0.3278	3.637562

These metrics include, *Mean Error (ME)*, *Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)*, *Mean Absolute Percent Error (MAPE), and Mean Percentage Error (MPE)*. The first metric (ME) is used to measure the predicted average error, whereas the measured variance square root with the average absolute error is measured by (RMSE). Moreover, (MAE) is utilized to compute the average absolute value for the predicted and actual value's difference. Finally, (MPE) is used to compute the average of percentage variation errors between the predicted and actual values, while (MAPE) is used to compute the average absolute value for the predicted actual value's difference, which is explained as the actual value percentage [53]. When the values of these metrics are too low or close to zero, it indicates that the prediction method has achieved very high prediction accuracy.

4.2 VMs Power Consumption Prediction

Besides the VMs workload prediction, the power consumption is predicted using the proposed approach for different number of VMs, while PM*i* and PM*j*, source and destination, are running for both live migration and auto-scaling, as described next.

4.2.1 VMs Live Migration Power Consumption Prediction

VMs predicted results in relation to actual value of power consumption is shown in Figs. 11–13, in which different VMs are running on source PM*i* (Host A) and destination PM*j*. Note that, the destination PM*j* can be the most energy efficient (Host B), which is similar to the configuration of the source host (Host C) or with less energy efficient (Host D) comparing to source PM*i*, all of which were based on the migration decision.

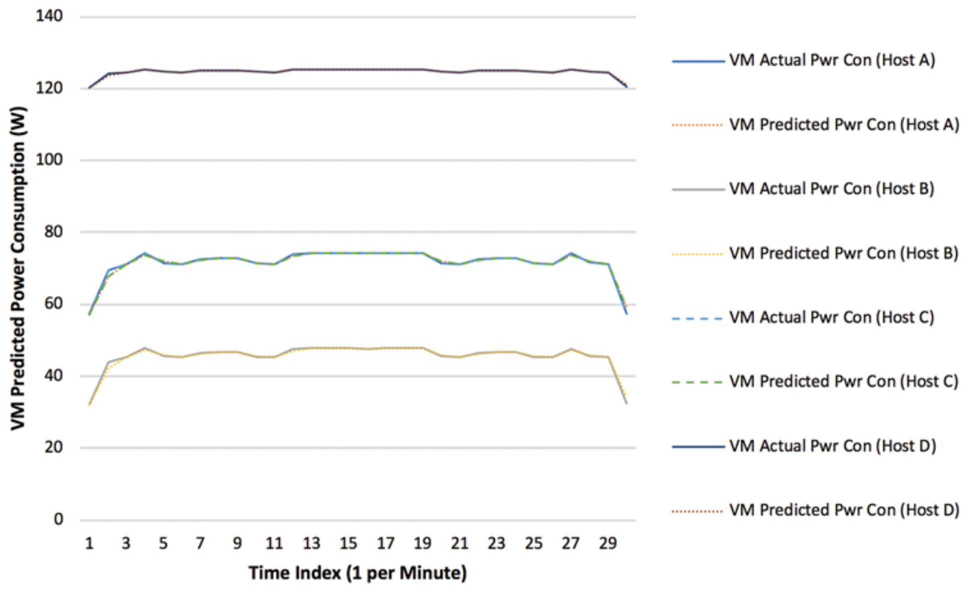

Figure 11: Small VM predicted *vs*. actual power consumption on (source PMi and destination PMj)

According to Algorithm 2, the migration is performed for the underloaded PM*i* if the selected destination PM*j* has enough resources and the upper value of threshold is not exceeded once the VMs migration takes place. Also, it is worth mentioning that a change in the PM predicated value of CPU utilization will influence the value of predicted power consumption of all VMs. Meanwhile, Tab. 3 presents different number of metrics used to verify the accuracy of the prediction of power consumption using three types of VMs (i.e., small, medium and large) based on a periodic workload pattern.

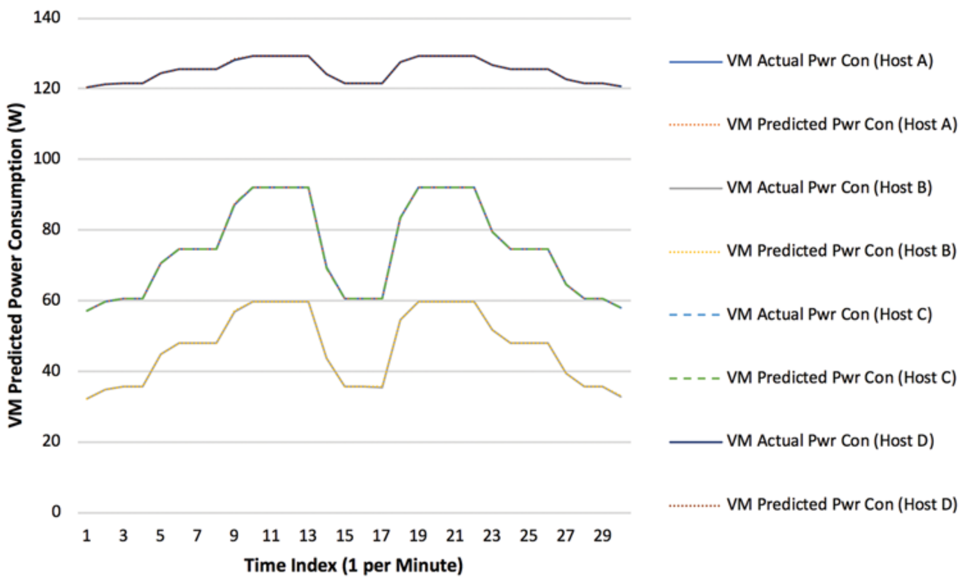

Figure 12: Medium VM predicted *vs.* actual power consumption on (source PMi and destination PMj)

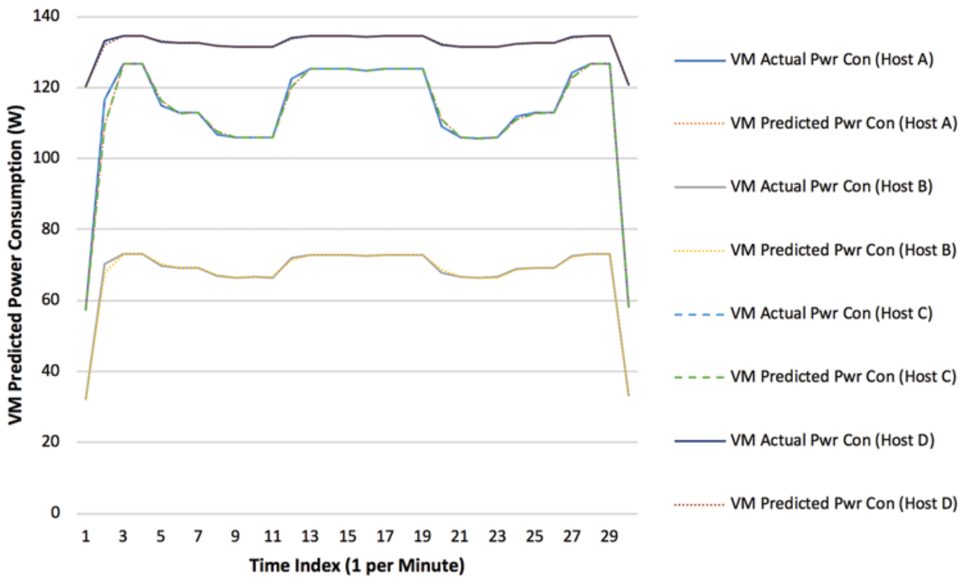

Figure 13: Large VM predicted *vs.* actual power consumption on (source PMi and destination PMj)

Table 3: Prediction accuracy for the predicted power consumption for all VMs on source (Host A) and destination (Host B, Host C and Host D)

Parameter	VMs	Hosts	ME	RMSE	MAE	MPE	MAPE
VMs power consumption	Small VM	Host A	−0.00551665	0.5150904	0.2493285	0.00539324	0.3674461
		Host B	0.005655233	0.4750381	0.2190667	0.04799226	0.5281281
		Host C	−0.00551665	0.5150904	0.2493285	0.00539324	0.3674461
		Host D	0.00246747	0.1537848	0.07028654	0.0023619	0.05689478
	Medium VM	Host A	0.01939327	0.07113483	0.04306951	0.02648363	0.05983904
		Host B	0.01529777	0.05683427	0.03492552	0.03521646	0.08024377
		Host C	0.01939327	0.07113483	0.04306951	0.02648363	0.05983904
		Host D	0.004925887	0.01823638	0.01120869	0.003956332	0.00901164
	Large VM	Host A	−0.2564522	1.533448	0.5685501	−0.2213621	0.5101096
		Host B	−0.07265782	0.5223516	0.193443	−0.0954475	0.2912161
		Host C	−0.2564522	1.533448	0.5685501	−0.2213621	0.5101096
		Host D	0.00000132	0.0000031	0.00000278	0.00000099	0.0000021

4.2.2 VMs Auto-scaling Power Consumption Prediction

Figs. 14–22 show the predicted results in relation to the actual value of VMs' power consumption that are operating on different hosts using different techniques (vertical scaling, *migration and vertically scaling* and horizontal scaling).

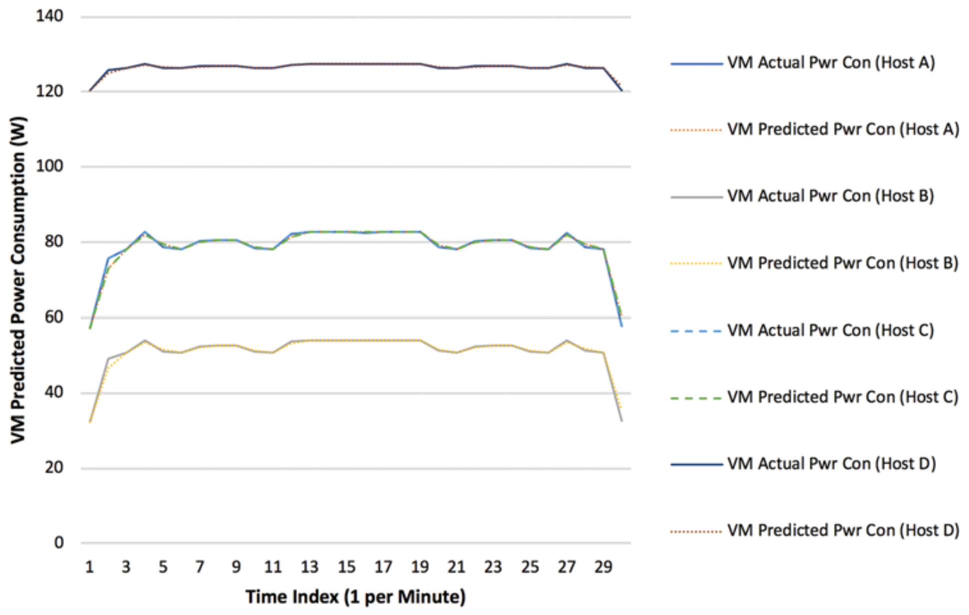

Figure 14: Small VM predicted *vs*. actual power consumption using vertical scaling on a number of PMs

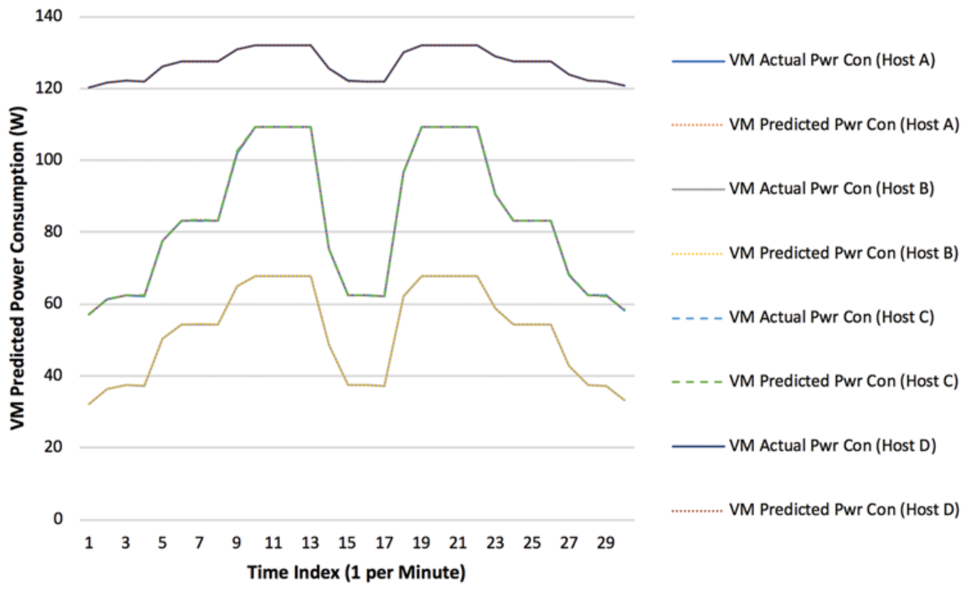

Figure 15: Medium VM predicted *vs.* actual power consumption using vertical scaling on a number of PMs

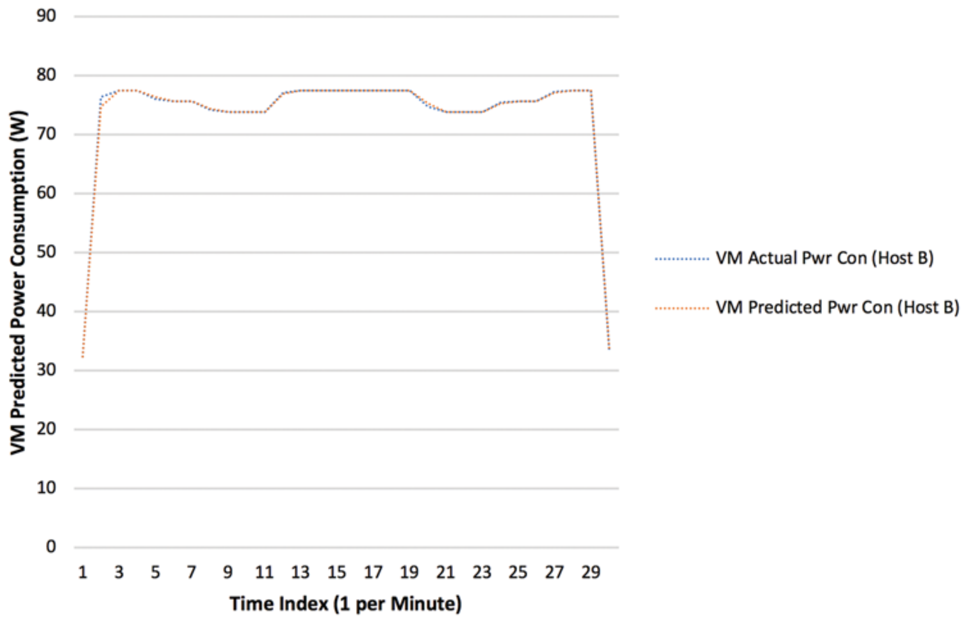

Figure 16: Large VM predicted *vs.* actual power consumption using vertical scaling on a number of PMs

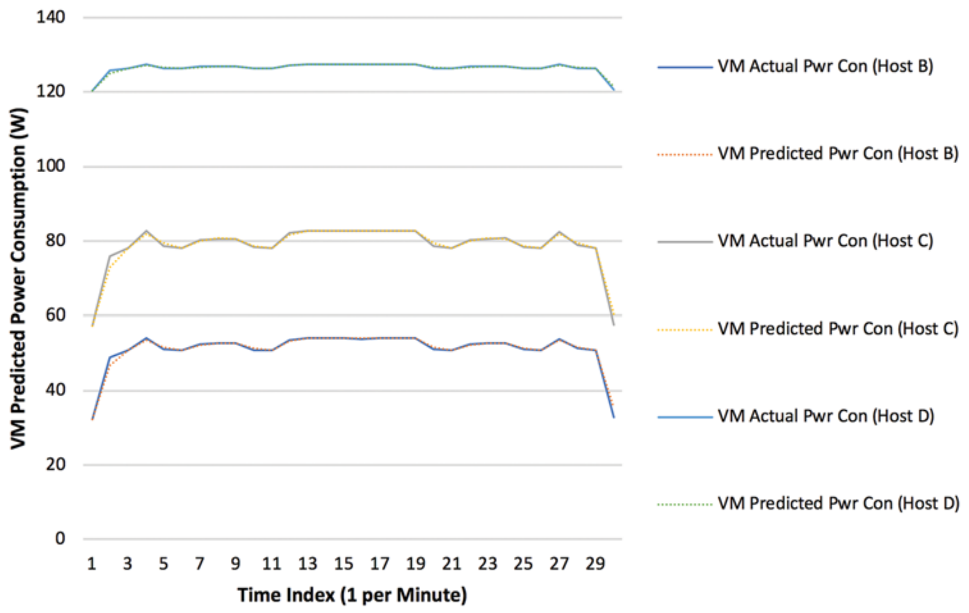

Figure 17: Small VM predicted *vs.* actual power consumption using migration and vertically scaling on a number of PMs

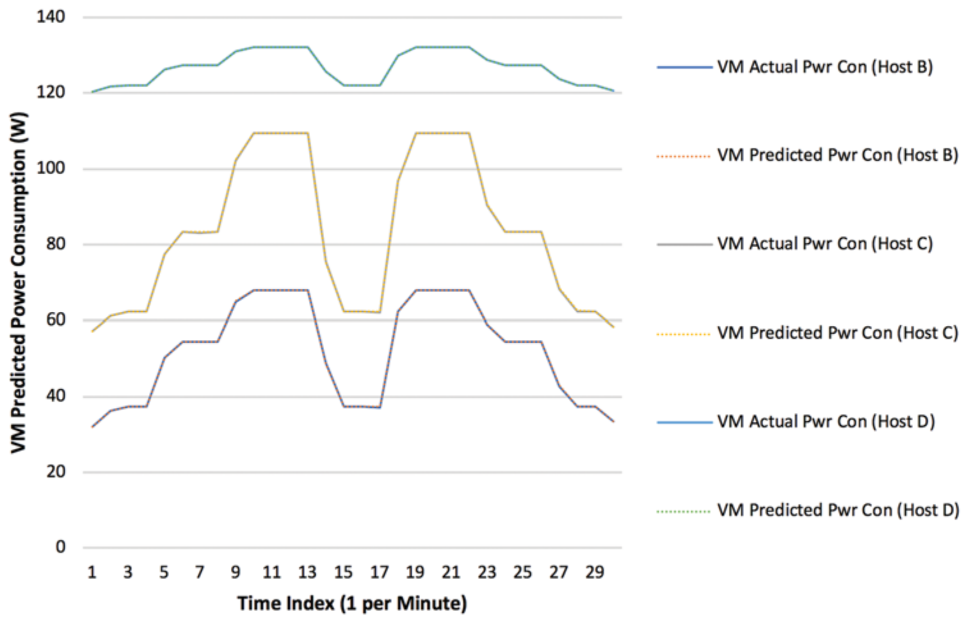

Figure 18: Medium VM predicted *vs.* actual power consumption using migration and vertically scaling on a number of PMs

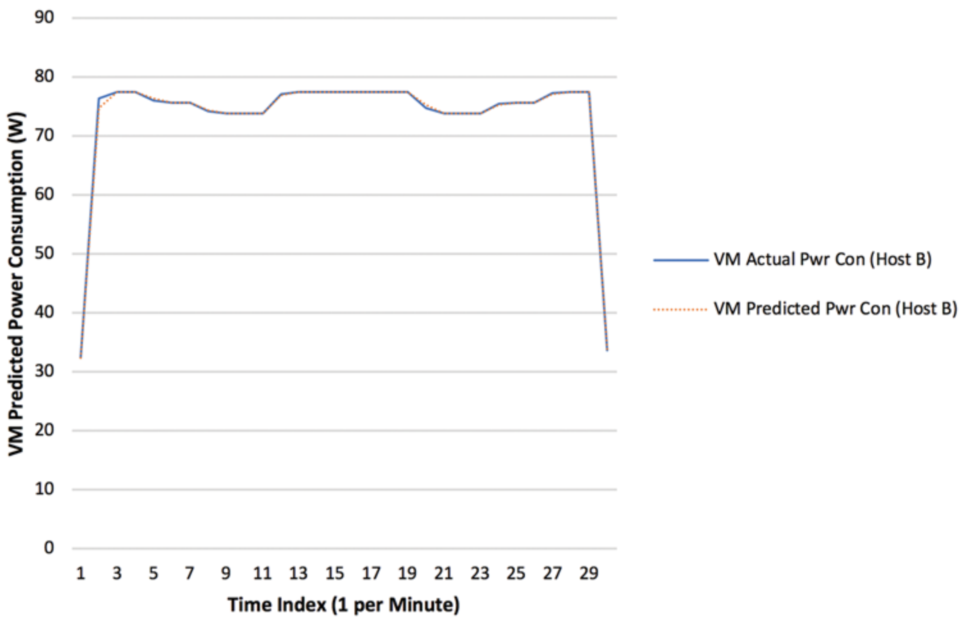

Figure 19: Large VM predicted *vs.* actual power consumption using migration and vertically scaling on a number of PMs

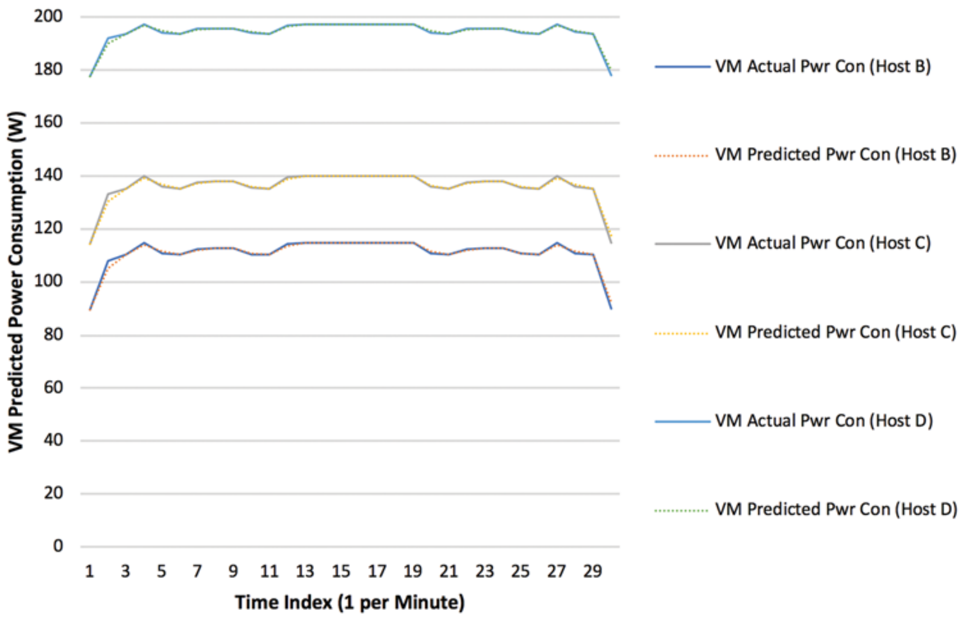

Figure 20: Small VM predicted *vs.* actual power consumption using horizontal scaling on a number of PMs

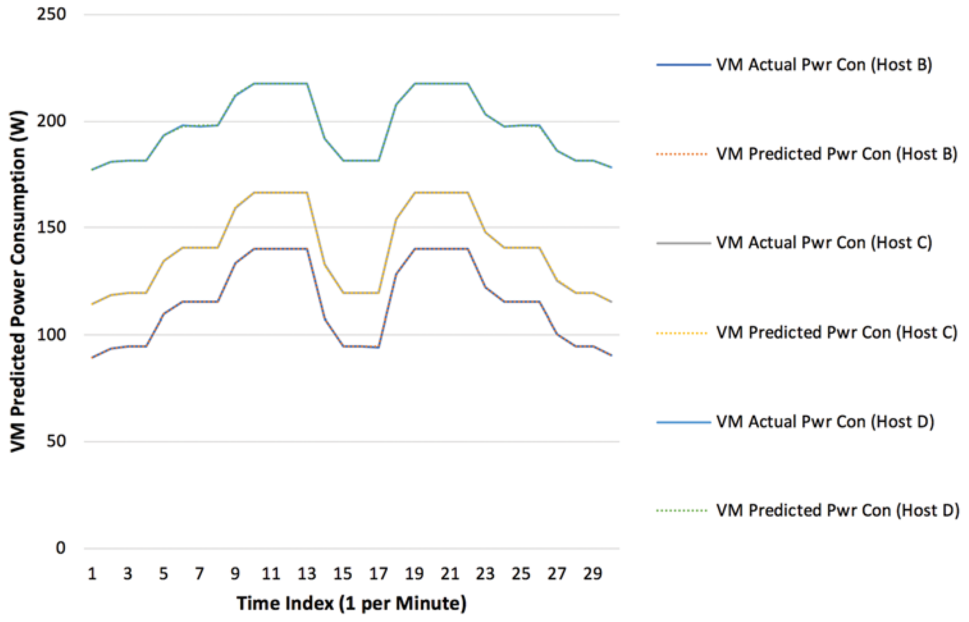

Figure 21: Medium VM predicted *vs.* actual power consumption using horizontal scaling on a number of PMs

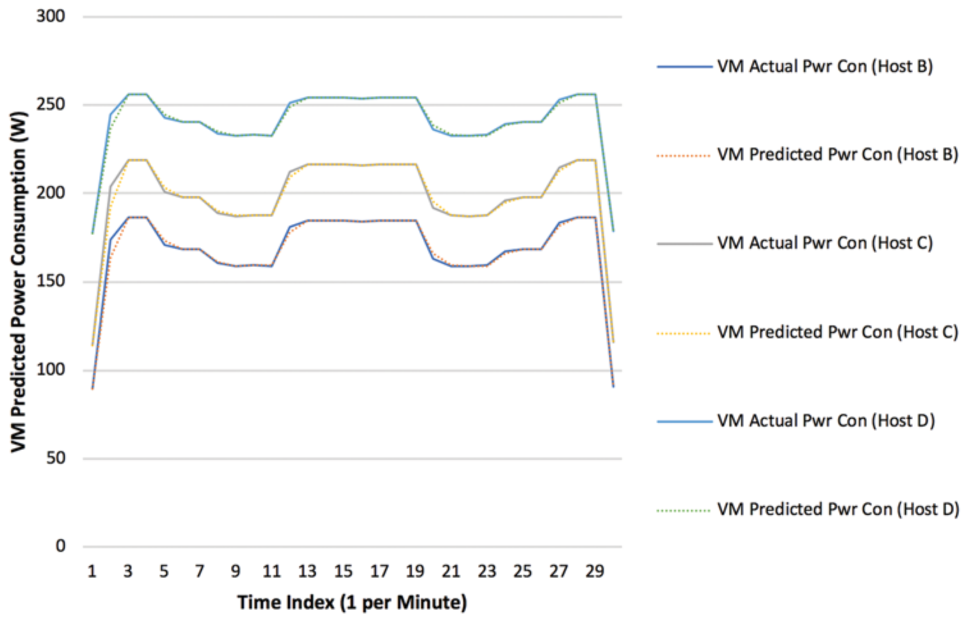

Figure 22: Large VM predicted *vs.* actual power consumption using horizontal scaling on a number of PMs

According to Algorithm 3, the vertical scaling is performed for the overloaded VMs on the same host, if the host has enough resources to meet the scaling requirements (PM*i* capacity is limited by vertical scaling). Otherwise, the VMs *migration and vertically scaling* or the horizontal scaling are performed for the overloaded VMs on a number of hosts, e.g., (Host B) is the most energy efficient, (Host C) has a configuration similarity to the source host (Host A), and (Host D) is the less energy efficient. Note that, the vertical scaling was not performed for all types of VMs (e.g., large VM), since the large VM has four CPU cores and the capacity of the hosted PM (e.g., on Host A, Host C or Host D) has the same number of CPU cores as the VM. Thus, vertical scaling on the same host is not available. Therefore, only horizontal scaling can be performed with this type of VM. On the other hand, the vertical scaling or *migration and vertically scaling* can be performed for the large VM only on (Host B), since it has eight CPU cores.

Meanwhile, Tabs. 4–6 present different number of metrics used to verify the accuracy of the prediction of power consumption using three types of VMs (i.e., small, medium and large).

Table 4: Prediction accuracy for the predicted power consumption for all VMs performs (vertical scaling) on different hosts

Parameter	VMs	Hosts	ME	RMSE	MAE	MPE	MAPE
VMs power consumption	Small VM	Host A	−0.00827498	0.7726357	0.3739927	0.01611016	0.5129771
		Host B	0.01625062	0.6698005	0.2994985	0.09771351	0.6698014
		Host C	−0.00827498	0.7726357	0.3739927	0.01611016	0.5129771
		Host D	0.00627207	0.2159668	0.09559397	0.00575502	0.07665149
	Medium VM	Host A	0.02908991	0.1067022	0.06460426	0.03629048	0.08216754
		Host B	0.01980407	0.07504576	0.04582536	0.04335952	0.09834909
		Host C	0.02908991	0.1067022	0.06460426	0.03629048	0.08216754
		Host D	0.006596707	0.02445368	0.01499183	0.005249759	0.01192901
	Large VM	Host B	−0.03093269	0.3194241	0.1197874	−0.02547422	0.174785

Table 5: Prediction accuracy for the predicted power consumption for all VMs performs (migration and vertically scaling) on different hosts

Parameter	VMs	Hosts	ME	RMSE	MAE	MPE	MAPE
VMs power consumption	Small VM	Host B	0.01625062	0.6698005	0.2994985	0.09771351	0.6698014
		Host C	−0.00827498	0.7726357	0.3739927	0.01611016	0.5129771
		Host D	0.00627207	0.2159668	0.09559397	0.00575502	0.07665149
	Medium VM	Host B	0.01980407	0.07504576	0.04582536	0.04335952	0.09834909
		Host C	0.02908991	0.1067022	0.06460426	0.03629048	0.08216754
		Host D	0.006596707	0.02445368	0.01499183	0.005249759	0.01192901
	Large VM	Host B	−0.03093269	0.3194241	0.1197874	−0.02547422	0.174785

Table 6: Prediction accuracy for the predicted power consumption for all VMs performs (horizontal scaling) on different hosts

Parameter	VMs	Hosts	ME	RMSE	MAE	MPE	MAPE
VMs power consumption	Small VM	Host B	−0.0054593	0.7680777	0.3691814	0.00744378	0.3509133
		Host C	−0.0082749	0.7726357	0.3739927	0.00169475	0.2865351
		Host D	−0.0052990	0.5977032	0.2882975	−0.0005908	0.1515757
	Medium VM	Host B	0.02823278	0.1035619	0.06297163	0.02507514	0.05674791
		Host C	0.02908989	0.1067022	0.06460427	0.02091843	0.04715986
		Host D	0.02224723	0.08158656	0.04950406	0.01128311	0.02533147
	Large VM	Host B	−0.3341869	2.027128	0.7513777	−0.1932349	0.4520876
		Host C	−0.3846782	2.300172	0.8528251	−0.1904874	0.4350779
		Host D	−0.2811938	1.68985	0.6266054	−0.1155186	0.2615967

4.3 *VMs Total Cost Estimation*

The proposed approach is also able to estimate VMs' total cost of live migration and auto-scaling when they are running on different PMs.

4.3.1 VMs Live Migration Cost Estimation

Fig. 23 shows the results of the estimated VMs' total cost before live migration on (Host A) and after live migration takes place on (Host B, Host C and Host D) along with their migration cost. According to Algs. 1 and 2, the migration is performed for the underloaded PMi only if the cost of VMs incurred by live migration to the selected destination PMj is less than the cost of switching the source PMi to power saving mode. In this case, only small VMs can meet these conditions to be migrated to the selected destination PMj, if it is the only VM running on the source PMi.

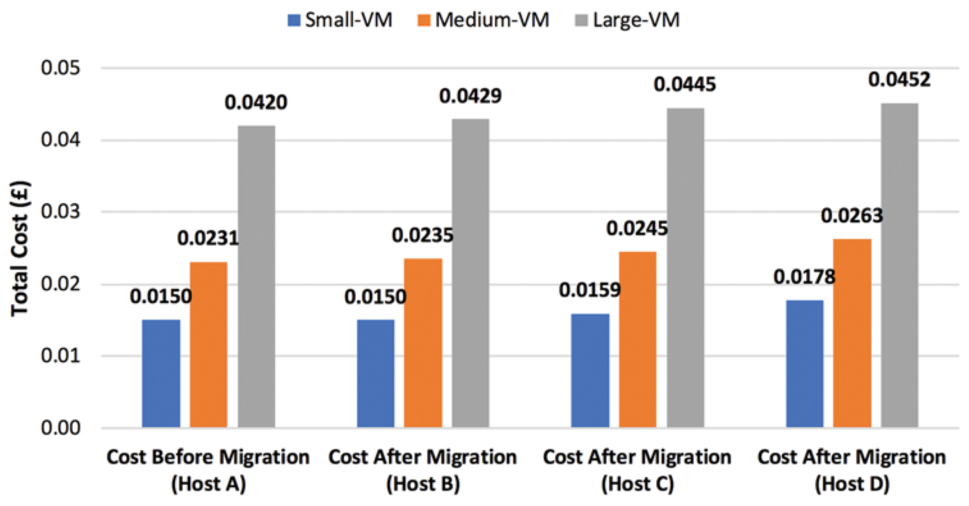

Figure 23: Estimated total cost before *vs.* after migration on different PMs

In addition, Fig. 24 presents the estimated cost saving results, which can be achieved for the small VM when being migrated to different hosts. This power savings cost is gained by switching the source (Host A) to power saving mode. For example, when the VM is migrated to the most energy efficient host (Host B), it can achieve approximately 100% cost saving which means the cost that can be saved by switching PM*i* to power saving mode minus the cost incurred by live migration decision. With a similar host configuration to the source (Host C), it can achieve around 76% and with the less energy efficient host (Host D), it can achieve about 27%. Further, hosts' energy efficiency plays a major role in reducing overall energy consumption.

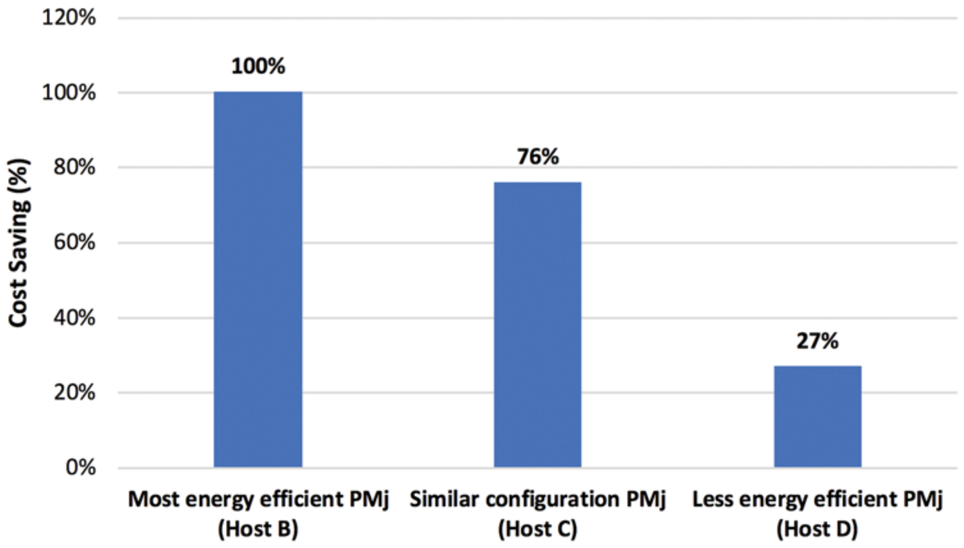

Figure 24: Estimated cost saving for migrating small VM to different PMs

Thus, selecting the appropriate hosts to migrate the VMs have a great effect on the overall cost saving (e.g., migrating the VMs to most energy efficient PM (Host B) is more cost-efficient than migrating the VMs to less energy efficient PM (Host D)).

4.3.2 VMs Auto-Scaling/Migration Cost Estimation

Figs. 25–27 show the results of the estimated total cost for three VMs types operating on multiple PMs using different scaling/migration strategies. According to Algorithms 3 and 4, the scaling of the VMs will be respected to the requested resources' right size using the self-configuration technique. Thus, the vertical scaling is performed on the same host, if the host has enough resources to meet the scaling requirements. Otherwise, the VMs *migration and vertically scaling* or the horizontal scaling are performed on a number of hosts, e.g., (Host B) is the most energy efficient, (Host C) is a host with a similarity configuration to the source host (Host A), and (Host D) is the less energy efficient. As mentioned earlier, the vertical scaling was not performed with the large VM, since it has the same number of CPU cores as the hosted PM (e.g., on Host A, Host C or Host D), which means that the host is fully utilized via the VM. However, the vertical scaling can be performed for the large VM only on (Host B) that has eight cores, as shown in Figs. 25 and 26, respectively.

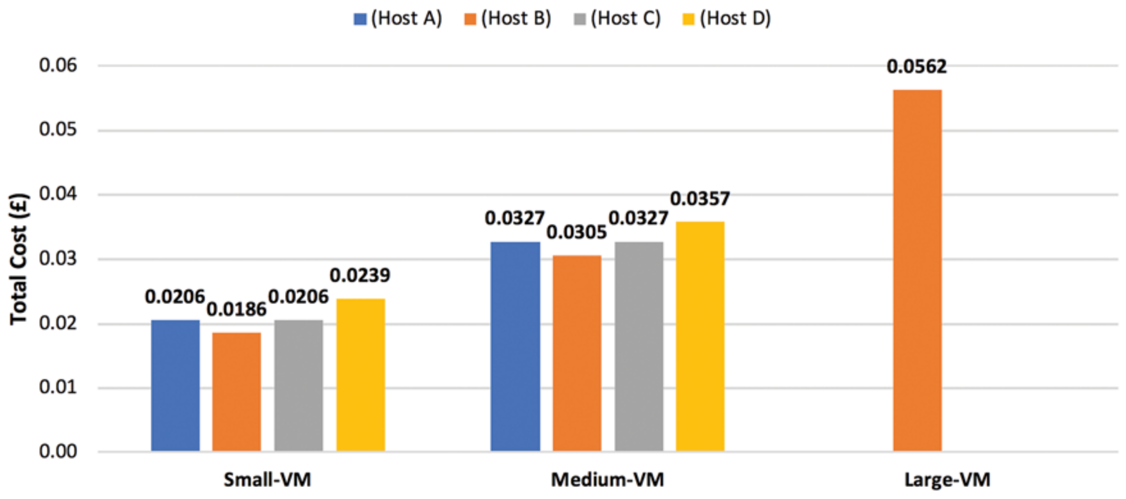

Figure 25: Estimated vertical scaling VMs total cost

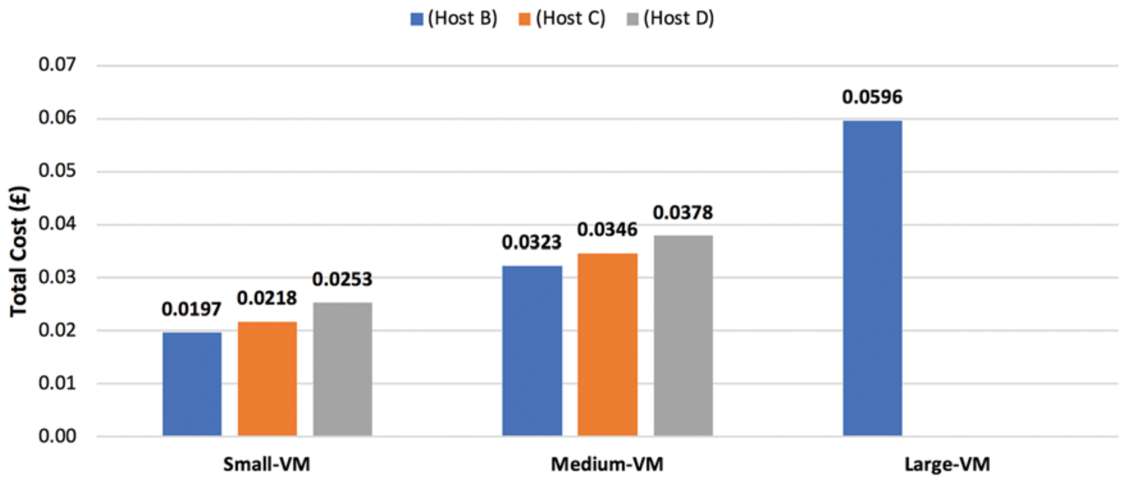

Figure 26: Estimated migration and vertically scaling VMs total cost

Choosing between different scaling strategies have a great effect on the scaled VMs' cost (e.g., vertical scaling can be more cost-efficient than the proposed *migration and vertically scaling* or the horizontal scaling when the VMs are scaled on the host with a similar configuration), as shown in Figs. 25–27, respectively. This can be justified because vertical scaling has no additional costs in terms of migration cost (e.g., in the case of *migration and vertically scaling*) or software license for new VMs [36] (e.g., in the case of horizontal scaling). However, the technique of vertical scaling is limited to the host's capacity [15,36]. Therefore, the proposed *migration and vertically scaling* mechanism helps for selecting the most appropriate cost-effective scaling strategy, rather than just only choosing between scaling up/out. As shown in Figs. 26 and 27, the proposed *migration and vertically scaling* mechanism outperforms the horizontal scaling one. This can be justified because of the additional cost in terms of new VMs' software license in the case of performing horizontal scaling is larger than the cost of live migration for the VMs when *migration and vertically scaling* is performed. Furthermore, selecting the appropriate hosts in terms of their energy efficiency to scale the VMs have a great effect on the scaled VMs' total cost (e.g., it is a cost-efficient decision if a PM with most energy efficient is used for horizontal scaling rather than a PM with less energy efficient). Although the workload utilization with high variation, the metrics of accuracy show that the VMs predicted workload and consumption of power are accurately predicted as well as the live migration and auto-scaling's estimated cost. Thus, the cost decision of cloud providers can be enhanced using the proposed approach in terms of selecting the most suitable cost-efficient migration and scaling techniques.

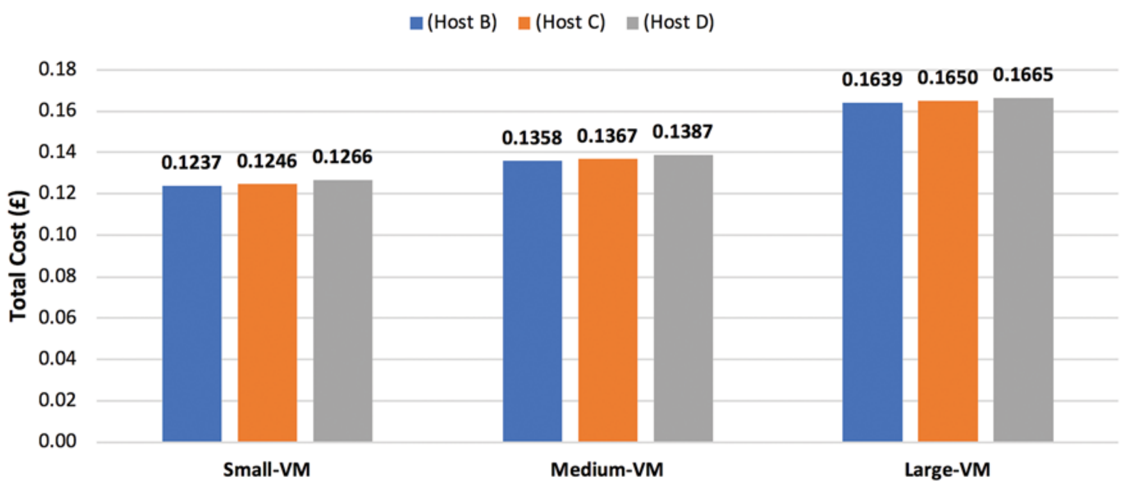

Figure 27: Estimated horizontal scaling VMs total cost

5 Conclusion and Future Work

This paper has proposed and verified a new hybrid approach for performance and predication cost of energy. This approach can support decision-making dynamically, regarding auto-scaling and live migration costs, while simultaneously aware of the energy consumption effect and application performance during service operation. This hybrid approach integrates auto-scaling with live migration in order to estimate the total cost of heterogeneous VMs by considering their resource usage and power consumption, while maintaining the expected level of application performance. The overall results show that the proposed hybrid approach can detect the underloaded and overloaded hosts to perform the most cost-effective decision(s) to handle the service performance variation. It can also accurately predict the workload and power consumption as well as estimate the total cost for both migrated and scaled VMs when operating on different PMs, on the basis of historical workload patterns.

In ongoing and future work, this approach will be extended to consider the impact of hardware accelerators, such as Graphic Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs) on the energy consumption and service performance. This extension would be useful when modelling and identifying the energy consumption and total cost of Cloud services. Furthermore, it is hard and costly to conduct large-scale experiments in a real Cloud environment (e.g., a Cloud testbed), especially with limited resources. Therefore, simulation can be considered to further study the scalability-related issues. Finally, additional Cloud applications workload patterns such as unpredictable and continuously changing as well as different prediction algorithms such as Deep Neural Network (DNN), can be further considered as an expansion of the scope for the workload prediction, power consumption prediction and the VMs' cost estimation on the basis of different types of workload patterns.

Acknowledgement: The author would like to thank the Deanship of Scientific Research, Prince Sattam Bin Abdulaziz University, Saudi Arabia, for supporting this work. Also, the author would like to acknowledge Prof. Karim Djemame for his help and guidance. This work was implemented on an existing Cloud testbed, available at the University of Leeds, UK.

Funding Statement: The author received no specific funding for this study.

Conflicts of Interest: The author declares that he has no conflicts of interest to report regarding the present study.

References

[1] T. Mukherjee, K. Dasgupta, G. Jung and H. Lee, "An economic model for green cloud," in *Proc. of the 10th Int. Workshop on Middleware for Grids, Clouds and e-Science*, Montreal, Canada. pp. 1–6, 2012.

[2] X. Zhang, J. Lu and X. Qin, "BFEPM: Best fit energy prediction modeling based on CPU utilization," in *IEEE Eighth Int. Conf. on Networking, Architecture and Storage*, Xi'an, China. pp. 41–49, 2013.

[3] J. Conejero, O. Rana, P. Burnap, J. Morgan, B. Caminero *et al.,* "Analyzing hadoop power consumption and impact on application QoS," *Future Generation Computer Systems*, vol. 55, no. 1, pp. 213–223, 2016.

[4] M. Bagein, J. Barbosa, V. Blanco, I. Brandic, S. Cremer *et al.,* "Energy efficiency for ultrascale systems: Challenges and trends from nesus project," *Supercomputing Frontiers and Innovations*, vol. 2, no. 2, pp. 105–131, 2015.

[5] A. Beloglazov, Y. C. Lee and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in computers*, vol. 82, pp. 47–111, 2011.

[6] Amazon, "Amazon EC2 pricing," 2021. [Online]. Available: https://aws.amazon.com/ec2/pricing/.

[7] Microsoft, "Microsoft Azure virtual machines pricing," 2021. [Online]. Available: https://azure.microsoft.com/en-gb/pricing/details/virtual-machines/linux/.

[8] Google, "Google cloud pricing," 2021. [Online]. Available: https://cloud.google.com/pricing/.

[9] A. Narayan, S. Member, S. Rao and S. Member, "Power-aware cloud metering," *IEEE Transactions on Services Computing*, vol. 7, no. 3, pp. 440–451, 2014.

[10] M. Hinz, G. P. Koslovski, C. C. Miers, L. L. Pilla and M. A. Pillon, "A cost model for IaaS clouds based on virtual machine energy consumption," *Journal of Grid Computing*, vol. 16, no. 3, pp. 493–512, 2018.

[11] D. Laganà, C. Mastroianni, M. Meo and D. Renga, "Reducing the operational cost of cloud data centers through renewable energy," *Algorithms*, vol. 11, no. 10, pp. 145, 2018.

[12] W. N. S. Wan Nik, B. B. Zhou, Z. Mohamad and M. A. Mohamed, "Cost and performance-based resource selection scheme for asynchronous replicated system in utility-based computing environment," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 7, no. 2, pp. 723–735, 2017.

[13] M. Aldossary and K. Djemame, "Energy-based cost model of virtual machines in a cloud environment," in *Fifth Int. Symp. on Innovation in Information and Communication Technology*, Amman, Jordan. pp. 1–8, 2018.

[14] K. Djemame, R. Bosch, R. Kavanagh, P. Alvarez, J. Ejarque *et al.,* "PaaS-IaaS inter-layer adaptation in an energy-aware cloud environment," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 127–139, 2017.

[15] J. Yang, C. Liu, Y. Shang, Z. Mao and J. Chen, "Workload predicting-based automatic scaling in service clouds," in *IEEE Sixth Int. Conf. on Cloud Computing*, Santa Clara, USA. pp. 810–815, 2013.

[16] A. Gandhi, P. Dube, A. Karve, A. Kochut and L. Zhang, "Modeling the impact of workload on cloud resource scaling," in *IEEE 26th Int. Symp. on Computer Architecture and High Performance Computing*, Jussieu, France. pp. 310–317, 2014.

[17] S. Dutta, S. Gera, A. Verma and B. Viswanathan, "SmartScale: Automatic application scaling in enterprise clouds," in *IEEE Fifth Int. Conf. on Cloud Computing*, Honolulu, USA. pp. 221–228, 2012.

[18] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *SC'11: Proce. of 2011 Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, Seatle, USA. pp. 1–12, 2011.

[19] A. Y. Nikravesh, S. A. Ajila and C. H. Lung, "An autonomic prediction suite for cloud resource provisioning," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 3, 2017.

[20] Y. Jiang, C. Perng, T. Li and R. Chang, "ASAP: A self-adaptive prediction system for instant cloud resource demand provisioning," in *IEEE 11th Int. Conf. on Data Mining*, Vancouver, Canada. pp. 1104–1109, 2011.

[21] Q. Zhang, H. Chen and Z. Yin, "PRMRAP: A proactive virtual resource management framework in cloud," in *IEEE Int. Conf. on Edge Computing*, Honolulu, USA. pp. 120–127, 2017.

[22] M. Aldossary and K. Djemame, "Performance and energy-based cost prediction of virtual machines live migration in clouds," in *Proc. of the 8th Int. Conf. on Cloud Computing and Services Science*, Madeira, Portugal. pp. 384–391, 2018.

[23] Q. Huang, K. Shuang, P. Xu, J. Li, X. Liu *et al.,* "Prediction-based dynamic resource scheduling for virtualized cloud systems," *Journal of Networks*, vol. 9, no. 2, pp. 375–383, 2014.

[24] H. T. Vu and S. Hwang, "A traffic and power-aware algorithm for virtual machine placement in cloud data center," *International Journal of Grid and Distributed Computing*, vol. 7, no. 1, pp. 21–32, 2014.

[25] F. Farahnakian, R. Bahsoon, P. Liljeberg and T. Pahikkala, "Self-adaptive resource management system in IAAS clouds," in *IEEE 9th Int. Conf. on Cloud Computing*, San Francisco, USA. pp. 553–560, 2016.

[26] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila and H. Tenhunen, "Utilization prediction aware VM consolidation approach for green cloud computing," in *IEEE 8th Int. Conf. on Cloud Computing*, New York, USA. pp. 381–388, 2015.

[27] F. Xu, F. Liu, L. Liu, H. Jin, B. Li *et al.,* "iAware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Transactions on Computers*, vol. 63, no. 12, pp. 3012–3025, 2014.

[28] M. Ficco, C. Esposito, F. Palmieri and A. Castiglione, "A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation," *Future Generation Computer Systems*, vol. 78, no. 6, pp. 343–352, 2018.

[29] H. Zhou, Q. Li, K. K. R. Choo and H. Zhu, "DADTA: A novel adaptive strategy for energy and performance efficient virtual machine consolidation," *Journal of Parallel and Distributed Computing*, vol. 121, no. 9, pp. 15–26, 2018.

[30] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *IEEE 30th Int. Conf. on Distributed Computing Systems*, Genova, Italy. pp. 62–73, 2010.

[31] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu *et al.,* "Energy-aware VM consolidation in cloud data centers using utilization prediction model," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 524–536, 2019.

[32] W. Fang, Z. Lu, J. Wu and Z. Cao, "RPPS: A novel resource prediction and provisioning scheme in cloud data center," in *IEEE Ninth Int. Conf. on Services Computing*, pp. 609–616, 2012.

[33] M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis *et al.,* "Energy-aware cost prediction and pricing of virtual machines in cloud computing environments," *Future Generation Computer Systems*, vol. 93, no. 3, pp. 442–459, 2019.

[34] M. Aldossary and K. Djemame, "Performance and energy-based cost prediction of virtual machines auto-scaling in clouds," in *44th Euromicro Conf. on Software Engineering and Advanced Applications*, Prague, Czech Republic. pp. 502–509, 2018.

[35] H. Yang, Q. Zhao, Z. Luan and D. Qian, "iMeter: An integrated VM power model based on performance profiling," *Future Generation Computer Systems*, vol. 36, pp. 267–286, 2014.
[36] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao *et al.,* "A cost-aware auto-scaling approach using the workload prediction in service clouds," *Information Systems Frontiers*, vol. 16, no. 1, pp. 7–18, 2014.
[37] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet *et al.,* "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proc. of the 7th Int. Conf. on Autonomic Computing*, pp. 11–20, 2010.
[38] C. Fehling, F. Leymann, R. Retter, W. Schupeck and P. Arbitter, *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer Vienna: Springer, 2014.
[39] G. E. P. Box, G. M. Jenkins, G. C. Reinsel and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, New Jersey: John Wiley & Sons, 2015.
[40] M. Aldossary, I. Alzamil and K. Djemame, "Towards virtual machine energy-aware cost prediction in clouds," in *Int. Conf. on Economics of Grids, Clouds, Systems, and Services*, Biarritz, France. pp. 119–131, 2017.
[41] G. E. P. Box, G. M. Jenkins and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*, 4th ed. Hoboken, New Jersey: John Wiley & Sons, 2008.
[42] OpenNebula, "The simplest cloud management experience," 2021. [Online]. Available: https://opennebula.org/.
[43] KVM, "Kernel-based virtual machine," 2021. [Online]. Available: https://www.linux-kvm.org/.
[44] WattsUp, "Watts Up pro portable power meter," 2021. [Online]. Available: https://www.powermeterstore.com/p1206/watts_up_pro.php.
[45] Zabbix, "The enterprise-class monitoring solution for everyone," 2021. [Online]. Available: https://www.zabbix.com/.
[46] Rackspace, "Cloud servers pricing and cloud server costs," 2021. [Online]. Available: https://www.rackspace.com/cloud/servers/pricing.
[47] ElasticHosts, "Pricing elastichosts linux," 2019. [Online]. Available: https://www.elastichosts.co.uk/pricing/.
[48] VMware, "VMware cloud," 2019. [Online]. Available: https://www.vmware.com/.
[49] CompareMySolar, "Electricity price electricity price per kWh comparison of big six energy companies," 2021. [Online]. Available: http://blog.comparemysolar.co.uk/electricity-price-per-kwh-comparison-of-big-six-energy-companies/.
[50] Stress-ng, "Stress tests," 2019. [Online]. Available: http://kernel.ubuntu.com/~cking/stress-ng/.
[51] R. N. Calheiros, E. Masoumi, R. Ranjan and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.
[52] R, "The R project for statistical computing," 2021. [Online]. Available: http://www.r-project.org/.
[53] R. J. Hyndman and G. Athanasopoulos, "Measuring forecast accuracy," 2020. [Online]. Available: www.otexts.org/fpp/2/5.