Proceedings Paper:
Aldossary, M and Djemame, K (2018) Performance and Energy-Based Cost Prediction of Virtual Machines Auto-Scaling in Clouds. In: Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018). 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018), 29-31 Aug 2018, Prague, Czech Republic. IEEE, pp. 502-509. ISBN: 978-1-5386-7383-6.

https://doi.org/10.1109/SEAA.2018.00086

Performance and Energy-based Cost Prediction of Virtual Machines Auto-Scaling in Clouds

Moahammad Aldossary[1, 2], and Karim Djemame[2]
[1]Prince Sattam Bin Abdulaziz University, KSA
MM.Aldossary@psau.edu.sa
[2]School of Computing, University of Leeds, Leeds, U.K.
{scmmald, k.djemame}@leeds.ac.uk

***Abstract*— Virtual Machines (VMs) auto-scaling is an important technique to provision additional resource capacity in a Cloud environment. It allows the VMs to dynamically increase or decrease the amount of resources as needed in order to meet Quality of Service (QoS) requirements. However, the auto-scaling mechanism can be time-consuming to initiate (e.g. in the order of a minute), which is unacceptable for VMs that need to scale up/out during the computation, besides additional costs due to the increase of the energy overhead. This paper introduces a Performance and Energy-based Cost Prediction Framework to estimate the total cost of VMs auto-scaling by considering the resource usage and power consumption, while maintaining the expected level of performance. A series of experiments conducted on a Cloud testbed show that this framework is capable of predicting the auto-scaling workload, power consumption and total cost for heterogeneous VMs, with a cost-saving of up to 25% for the predicted total cost of VM self-configuration as compared to the current approaches in literature.**

Keywords*— *Cloud Computing, Cost Prediction, Workload Prediction, Auto-Scaling, Power Consumption, Energy Efficiency.

I. INTRODUCTION

Cost mechanisms that are employed by different cloud service providers significantly influence the role of cloud computing within the IT industry. With the increasing cost of electricity, cloud providers consider energy consumption as one of the biggest operational cost factors to be managed within their infrastructures.

Most of the existing studies have focused on minimising the energy consumption and maximising the total resource usage, instead of improving the performance of applications. Further, cloud providers such as Amazon EC2 [1], have established their Service Level Agreements (SLAs) based on service availability without such an assurance of the performance. For instance, during service operation, consider the situation where a number of VMs are running on the same Physical Machine (PM), and each VM is allocated its fair share of resources. If the VM's workload increases and no resources are available to handle that increment (e.g. the workload exceeds the acceptable level of CPU such as 95% threshold), resource competition may occur leading to VMs' performance degradation which may affect the fulfilment of the SLAs and hence the cloud infrastructure provider's revenue. Hence, to prevent such performance loss effects, it is necessary to have preventive actions, e.g. VMs auto-scaling or VMs re-allocation through live migration.

VMs auto-scaling is an important technique to provision additional capacity to the VMs on the fly. There are two types of VMs auto-scaling: 1) vertical scaling (scale-up): request for more resources (e.g. virtual CPUs and memory) inside the VMs, and 2) horizontal scaling (scale-out): request for creating additional VMs. However, VMs auto-scaling may take a few minutes to initiate, which is unacceptable for VMs that need to quickly scale up/out during the computation. Besides, there are additional costs in terms of scaling time (booting/rebooting) and energy overhead that need further consideration. Hence, understanding the impact of VMs auto-scaling is essential for the design of an effective resource provision technique.

To enable VMs auto-scaling on the fly without any performance loss or latency, some form of prediction mechanism is needed to prepare the VMs in advance. Thus, a *proactive* framework has the advantage of taking preventive actions (e.g. VMs auto-scaling) at earlier stages to avoid service performance degradation. The effectiveness of such framework will depend on potential actuators/decisions to implement at service operation. Accordingly, predicting the future cost of cloud services can help the service providers offer suitable services that meet their customers' requirements.

The first step towards this is a *Performance and Energy-based Cost Prediction Framework* that supports the potential actuators (e.g. VMs auto-scaling) to handle the performance variation. Therefore, this framework is proposed to predict PMs, and VMs workload using an Autoregressive Integrated Moving Average (ARIMA) model. The relationship between the predicted VMs and PMs workload (CPU utilisation) is investigated using regression models in order to estimate the VMs power consumption, as well as predict the total cost of the VMs incurred by auto-scaling decision. This paper's main contributions are summarised as follows:

- A Performance and Energy-based Cost Prediction Framework that predicts the auto-scaling cost for heterogeneous VMs/PMs by considering their performance, resource usage and power consumption.
- An evaluation of the proposed framework in an existing Cloud testbed in order to demonstrate its usability with clear cost savings.

The remainder of this paper is organised as follows: a discussion of the related work is summarised in Section II.

Section III presents the performance and energy-based cost prediction framework. Section IV presents the experimental setup followed by results and discussion in Section V. Finally, Section VI concludes this paper and discusses the future work.

II. RELATED WORK

Previous work has addressed specific issues relating to the cost of VMs auto-scaling in a Cloud environment. For example, a lightweight scaling approach to enable cost-effective elasticity for cloud applications is proposed in [2]. The approach uses fine-grained scaling mechanism at resource-level scaling and VM-level scaling in order to improve resource utilisation while reducing cloud providers' operating costs. However, the work only mentions the impact of vertical scaling on the cloud providers' cost without taking horizontal scaling into account. Likewise, an automatic scaling framework called (SmartScale) is presented in [3]. The framework uses a combination of horizontal and vertical scaling in order to minimise the operating costs. However, the authors claim that horizontal scaling allows the application to achieve higher performance while the cost incurred due to this scaling is higher than vertical scaling. Therefore, this leads us to investigate further the vertical scaling technique and the impact on performance and cost. Another approach for auto-scaling is proposed in [4]. It used a second order Auto-Regressive Moving Average (ARMA) model for workload prediction based on historical workload. The model aims to minimise the resource usage and satisfy QoS requirements while keeping operational costs low. Further, a reactive auto-scaling framework that allows a broker to obtain resources from a public cloud to handle customers' requests is proposed in [5]. This framework can effectively lead to a profit for the broker and a cost reduction for the customers'. However, all of the studies presented above do not consider the energy overhead caused by auto-scaling.

Other work in the literature has proposed an auto-scaling approach to improve the performance in Clouds. For instance, a new performance metric called the Auto-scaling Demand Index (ADI) is introduced in [6]. The approach evaluates several auto-scaling strategies, including (reactive, conservative and predictive) using log traces from Google datacentres and used the utilisation level as a performance indicator. However, this approach is dealing with VM utilisation only, without reference to the auto-scaling costs, including (e.g. energy cost) or the SLA violation. Besides, no details are provided on where/how the experiments were conducted. An efficient auto-scaling approach to dynamically scale cloud instances based on task's deadline constraints and cost is presented in [7]. This approach is implemented on Microsoft Azure platform using both simulated and real applications. However, the energy efficiency of the candidate PM that will host the scaled instance is not considered when designing such mechanism.

Several prediction techniques have been proposed to predict the resource provisioning for Cloud applications. For example, a Cloud Resource Prediction and Provisioning scheme (RPPS) based on the ARIMA model is presented in [8]. The scheme automatically predicts future demand (CPU usage of VMs) and perform proactive resource provisioning for cloud applications. The results show that the prediction error on average is less than 10% in most time. However, the energy consumption and the characterisation of the workload before making the prediction decision are not considered. Moreover, a predictive elastic resource scaling scheme (PRESS) for Cloud systems is presented in [9]. The approach uses a short-term pattern matching and state driven approach (Markov chain) to predict the workloads. This approach is implemented on top of Xen using RUBiS[1] and an application load traces from Google. Nonetheless, in this work only the workload as standalone application is predicted. In contrast, our approach predicts the workload that will be added to the existing VMs that are already overloaded. Likewise, an automatic elastic resource scaling system for multi-tenant cloud computing infrastructures called (CloudScale) is presented in [10]. The framework automatically scales VMs according to predicted workloads while considering energy consumption and SLA. CloudScale can increase or decrease the CPU frequency/voltage to achieving energy savings without impact the SLA. However, this approach does not consider the costs caused by scaling.

Compared with the work presented in this paper, we propose a proactive auto-scaling technique that considers the heterogeneity of PMs/VMs with respect to predicting the performance variation, resource usage, power consumption and the total cost. Our approach dynamically determines the most cost-effective scaling decision, including (scale-out/scale-up) that will result in the agreed performance for any given workload.

III. PERFORMANCE AND ENERGY-BASED COST PREDICTION FRAMEWORK

In this paper, we extend our work [11] by taking the performance variation into account and introduce a new **Performance and Energy-based Cost Prediction Framework**. This framework supports decision-making regarding auto-scaling cost while at the same time being aware of the impact on other quality characteristics such as energy consumption and performance of the application [12]. The auto-scaling resource provisioning technique is usually driven by the MAPE control loop (Monitor, Analyse, Plan and Execute) to provision resources when needed, as depicted in Figure 1.

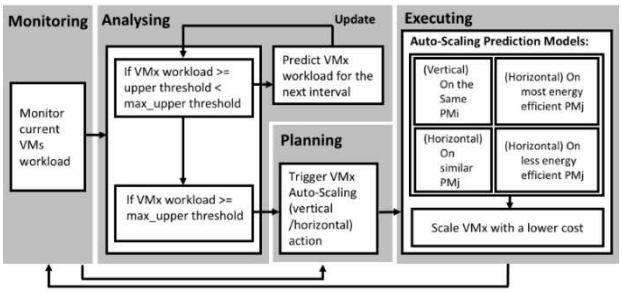

Fig. 1. Performance and Energy-based Cost Prediction Framework.

[1] http://rubis.ow2.org/

The proposed framework is aimed towards predicting workload and power consumption as well as the total cost of the VMs incurred by the auto-scaling decision. To achieve this aim, several steps are required in order to first predict the PMs/VMs workload and power consumption, then estimate the total cost of auto-scaled VMs as explained below.

Step 1: The CPU utilisation and RAM usage upper and max_upper thresholds (e.g. 85% and 95%) are set and the VM*x* workload is monitored. If the VM*x* workload is in the range of [upper and max_upper threshold], then predict the VM*x* workload for the next time interval (e.g. every 5 minutes) using the ARIMA model based on historical workload patterns (see Step 3). This prediction helps detect the workload and avoid unnecessary scaling caused by the small peaks in the workload (false alarm). If the predicted workload for the next interval exceeds the max_upper threshold, VM auto-scaling decision is performed as described in Algorithm 1. The list of parameters and their notations is shown in Table I.

Step 2: if the VM*x* workload equals or exceeds the max_upper threshold (e.g. 95%), VM auto-scaling decision is performed as shown in Algorithm 1. Algorithm 1 is used to identify the overloaded VM*x* to be scaled and potentially the most energy efficient candidate PM*j* to host it, if there is no capacity to perform a vertical scaling in the first place. The VMs are ranked in decreasing order of their workload whereas the PMs are ranked in decreasing order according to their energy efficiency. The energy efficiency of the hosts (source PM*i* and candidate PM*j*) can be computed as: PM power = $\frac{\text{PM}i\text{ (idle power)}}{\text{PM}j\text{ (idle power)}}$. It is also checked that the candidate PMs would have sufficient resources to handle the scaled VM*x* workload in order to prevent service performance degradation (e.g. when VM resource utilisation increases beyond the predefined threshold). Furthermore, this Algorithm demonstrates the comparison between vertical scaling (scale-up) and horizontal scaling (scale-out) in order to obtain the most cost-effective scaling decision. The task is to scale the overloaded VM*x* and select the candidate PM to host it. To do so, the following conditions are tested in this order and the subsequent action performed: 1) vertical scaling on the same PM*i* (vertical scaling is limited to the capacity of PM*i*); 2) horizontal scaling on the PM*j* with the most energy efficient; 3) horizontal scaling on similar source configuration PM*j* (e.g. on any homogeneous PM*j* with same source PM*i* configuration such as the CPU type and the ratio of idle power), or 4) horizontal scaling takes place on a less energy efficient PM*j*, as illustrated in Figure 2.

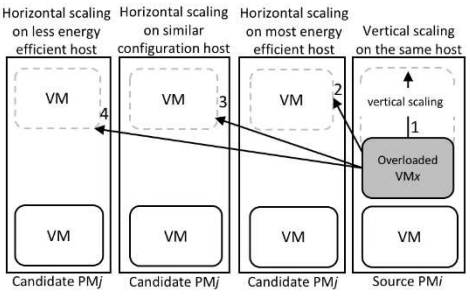

Fig. 2**.** The process of VM auto-scaling (vertical scaling vs. horizontal scaling).

TABLE I. List of parameters and their notations.

PMi	the source physical machine
PMj	the candidate physical machine
VM*x*	the overloaded VM to scale
C_CPU_PM	total CPU capacity of the PM
C_RAM_PM	total memory capacity of the PM
U_CPU_PM	used CPU capacity of the PM ($\sum_{y=1}^{VMCount}$(vCPU))
U_RAM_PM	used memory capacity of the PM ($\sum_{y=1}^{VMCount}$(RAM))
C_CPU_VM	total CPU capacity of the VM
C_RAM_VM	total memory capacity of the VM
U_CPU_VM	used CPU capacity of the VM
U_RAM_VM	used memory capacity of the VM
I_CPU_VM	increment CPU capacity of the VM
I_RAM_VM	increment memory capacity of the VM

Algorithm 1: VMs Workload Prediction and Auto-Scaling Decision

Initialise: VM workload = $\left(\frac{\text{U\_CPU\_VM}}{\text{C\_CPU\_VM}}, \frac{\text{U\_RAM\_VM}}{\text{C\_RAM\_VM}}\right)$;
VM upper threshold = 0.85 × (C_CPU_VM, C_RAM_VM);
VM max_upper threshold = 0.95 × (C_CPU_VM, C_RAM_VM);
PM workload = $\left(\frac{\text{U\_CPU\_PM}}{\text{C\_CPU\_PM}}, \frac{\text{U\_RAM\_PM}}{\text{C\_RAM\_PM}}\right)$;
PM upper threshold = 0.85 × (C_CPU_PM, C_RAM_PM);
PM power = $\frac{\text{PM}i\text{ (idle power of the source)}}{\text{PM}j\text{ (idle power of the candidate)}}$; // to check the energy efficiency
Predicted VM workload = null;
Resource Increments = (I_CPU_VM, I_RAM_VM) = (null, null);
Scaling Decision = null.
Input: VMs list, PMs list. // Assuming all the PMs in running/active status
Output: Scaling Decision.

```
1: Sort the PMs list in decreasing order of the PM power;
2: Sort the VMs list on PMi in a decreasing order of the workload;
3: for each (VMx in VMs list) do
4:    if (VMx workload ≥ VMx upper threshold) &&
        (VMx workload < VMx max_upper threshold) then
5:      Predicted VMx workload ⟵ predict the (VMx workload) for the next
                                 interval using the ARIMA model.
6:      if (Predicted VMx workload > VMx workload) then
7:        Resource Increments = Predicted VMx workload − VMx workload
8:      else
9:          break.
10:     end if
11:   end if
12: if (Predicted VMx workload ≥ VMx max_upper threshold) then
13:   if (PMi workload + Resource Increments) < PMi upper threshold) then
        // The resource availability on the same host is met (Resize VMx)
14:     {Scaling Decision ⟵ perform VMx vertical scaling based on
                            (Resource Increments);
15:         break.
16:     else // Lack of resources on the same host (PMi)
17: for each (PMj in PMs list) do
18:   if ((PMj workload + Resource Increments) < PMj upper threshold) then
19:     {Scaling Decision ⟵ perform VMx horizontal scaling based on
                            (Resource Increments);
        // Create a New VM on: a) the most energy efficient host, if possible;
                               or b) a similar host configuration to source,
                               or c) the less energy efficient host
20:         break.
21:     end if
22:   end for
23: end if
24: end if
25: return Scaling Decision.
26: end for
```

Step 3: Algorithm 2 is used to select the right size of the VMs to be scaled in an economic way based on the closest predefined instance sizes set by Cloud providers (e.g. small, medium and large). However, this mechanism sometimes leads to resource over-provisioning (e.g. if the requested resources for the auto-scaling are less than the predefined instance sizes set by Cloud

providers). This may result in resource wasted (needless capacity is created) and the customers might pay more without any benefit, which is not the aim of VMs auto-scaling. Therefore, a self-configuration approach to resize/create the VMs based on the right size of the requested resources is proposed. Thus, this mechanism will help Cloud providers to maximise their resources usage and the customers will pay for what they actually use, as described in Algorithm 2.

Algorithm 2: Self-configuration – Resizing/Creating VMs

Initialise: Scaling VM = null.
Input: Scaling Decision; // From Algorithm 1 (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)
Resource Increments = (I_CPU_VM, I_RAM_VM) // From Algorithm 1
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 Scaling Decision
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 Scaling Decision
8:          break;
9:        end if
10:   end if
11: end for
12: return Scaling VM.
```

After identifying the right size of the VM*x* to be scaled and the candidate PM*j* to host it, an ARIMA model is used to predict the scaled VM*x* workload (including CPU, memory, disk and network) utilisation and identify the best fit model. The ARIMA model is a time series prediction model that has been used widely in different domains, including finance, owing to its sophistication and accuracy. Unlike other prediction methods, like sample average, ARIMA takes multiple inputs as historical observations and outputs multiple future observations depicting the seasonal trend; further details about the ARIMA model can be found in [13]. Once the scaled VM*x* workload is predicted using the ARIMA model based on historical data, the next step is to predict the PMs (source and candidate) workload and PMs/VM*x* power consumption using regression models. Before predicting the power consumption for PMs/VM*x*, understanding how the resource usage affects the power consumption is required. Therefore, an experimental study is setup to investigate the effects of the resource usage on the power consumption. An experiment was carried out on a local Cloud Testbed (see Section IV), and the findings show that the CPU utilisation correlates well with the power consumption, as supported, for example, by [14].

Step 4: to predict the PMs workload represented as (PMs CPU utilisation), would require measuring the relationship between the number of virtual CPUs (vCPUs) and the PM CPU utilisation for the PMs, as shown in Figures 3, 4 and 5.

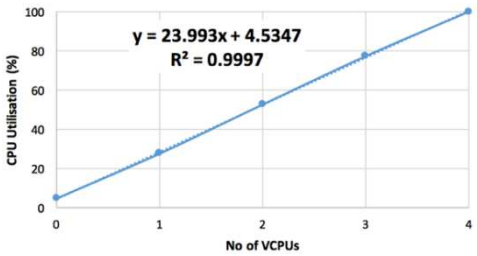

Fig. 3. Number of vCPUs (VM*x*) vs PM CPU Utilisation (Source PM*i*).

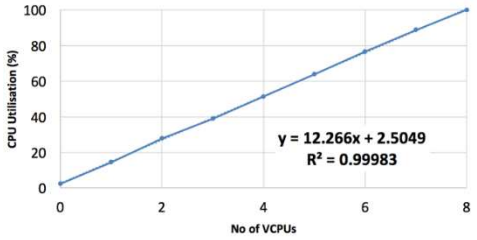

Fig. 4. Number of vCPUs (VM*x*) vs PM CPU Utilisation (candidate PM*j* - most energy efficient).

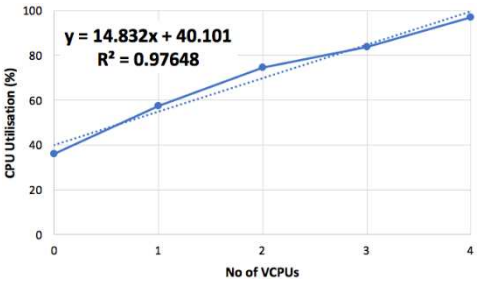

Fig. 5. Number of vCPUs (VM*x*) vs PM CPU Utilisation (candidate PM*j* - less energy efficient).

A linear regression model has been applied to predict the PMs CPU utilisation based on the used ratio of the requested number of vCPU for the VM*x* with consideration of its current workload as the PMs may be running other VMs already [15]. The following equation is used (1):

$$\mathrm{PM}i_{Pred\_U} = \left(\alpha \times \left(\sum_{y=1}^{VM\_Count}(\mathrm{VM}x_{ReqvCPUs} \times \frac{VMx_{Pred\_U}}{100})\right) + \beta\right) + \left(PMi_{Curr\_U} - PMi_{Idle\_U}\right) \quad (1)$$

$\mathrm{PM}i_{Pred\_U}$ is the predicted PM*i* CPU utilisation; α is the slope and β is the intercept of the CPU utilisation. The $\mathrm{VM}x_{ReqvCPUs}$ is the number of requested vCPU for each VM and VMx_{Pred_U} is the predicted utilisation for each VM. The PMi_{Curr_U} is the current PM*i* utilisation and PMi_{Idle_U} is the idle PM*i* utilisation. Consequently, the workload for the candidate PM*j* will be predicted using Equation 1, but substituting PM*i* with PM*j*.

Step 5: the PM*i* power consumption is predicted based on the relationship between the predicted PM*i* workload (PM CPU utilisation) with PM*i* power consumption on the PM*i*. Using a

regression analysis, the relation is best described as linear regression for this particular PM*i*, as shown in Figure 6.

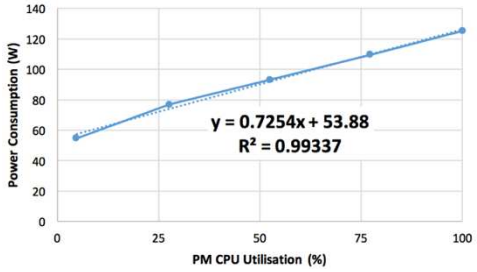

Fig. 6. The PM CPU Utilisation vs Power Consumption (Source PM*i*).

Thus, the predicted PM*i* power consumption $\mathrm{PM}i_{Pre\_P}$ measured by Watt, can be identified using the following formula (2).

$$\mathrm{PM}i_{Pred\_P} = \left(\alpha \times \mathrm{PM}i_{Pred\_U} + \beta\right) \quad (2)$$

Where α is the slope, β is the intercept and $\mathrm{PM}i_{Pred\_U}$ is predicted PM*i* CPU utilisation.

In the candidate PM*j* other regression models such as polynomial can be used to characterise the relation between the power consumption and CPU utilisation for these particular hosts, as shown in Figures 7 and 8.

The predicted PM*j* power consumption $\mathrm{PM}j_{Pred\_P}$ measured by Watt, can be identified using the following formula (3).

$$\mathrm{PM}j_{Pred\_P} = \left(\alpha(\mathrm{PM}j_{Pred\_U})^3 + \gamma(\mathrm{PM}j_{Pred\_U})^2 + \delta(\mathrm{PM}j_{Pred\_U}) + \beta\right) \quad (3)$$

Where α, γ and δ are all slopes, β is the intercept and $\mathrm{PM}j_{Pred\_U}$ is predicted PM*j* CPU utilisation.

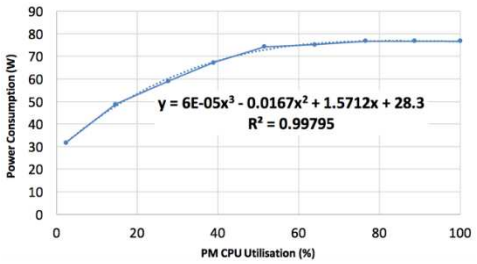

Fig. 7. The PM CPU Utilisation vs Power Consumption (candidate PM*j* - most energy efficient).

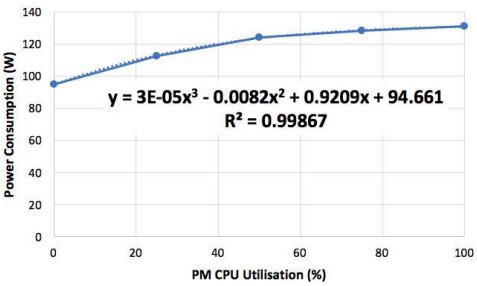

Fig. 8. The PM CPU Utilisation vs Power Consumption (candidate PM*j* - less energy efficient).

Step 6: based on the requested number of vCPU and the predicted vCPU utilisation, the VM*x* power consumption is predicted on PM*i* using the proposed formula, as shown in equation (4).

$$\mathrm{VM}x_{Pred\_P\_PMi} = PMi_{Idle\_P} \times \left(\frac{\mathrm{VM}x_{ReqvCPUs}}{\sum_{y=1}^{VM\_count}\mathrm{VM}x_{ReqvCPUs}}\right) + \left(PMi_{Pred\_P} - PMi_{Idle\_P}\right) \times \left(\frac{\mathrm{VM}x_{(Pred\_U \times ReqvCPUs)}}{\sum_{y=1}^{VM\_count}\mathrm{VM}x_{(Pred\_U \times ReqvCPUs)}}\right) \quad (4)$$

Where $\mathrm{VM}x_{Pred\_P\_PMi}$ is the predicted power consumption for VM*x* running on the PM*i* measured by Watt. $\mathrm{VM}x_{ReqvCPUs}$ is the requested number of vCPU and $\mathrm{VM}x_{pred\_U}$ is the predicted VM CPU utilisation. $\sum_{y=1}^{VM\_count}\mathrm{VM}x_{ReqvCPUs}$ is the total requested number of vCPU for all VMs on the PM*i*. PMi_{Idle_P} is the idle power consumption and PMi_{Pred_P} is the predicted power consumption for PM*i*. Hence, the VM*x* power consumption on the candidate PM*j* will be predicted using Equation 4, but substituting PM*i* with PM*j*.

The energy providers usually charge by the Kilowatt per hour (kWh). Therefore, the conversion of the power to energy $\mathrm{VM}x_{Pred\_E\_PMi}$ is required using the following equation (5):

$$\mathrm{VM}x_{Pred\_E\_PMi} = \frac{\mathrm{VM}x_{Pred\_P\_PMi}}{1000} \quad (5)$$

Substituting PM*i* with PM*j* to get the energy consumption for the VM*x* on the candidate PM*j*.

Step 7: this step predicts the total cost for the scaled VM*x* based on the predicted VM*x* resource usage in *step 3* and the predicted VM*x* energy consumption in *step 6*.

The total time required for auto-scaling VM*x* can be given by:

$$T_{Scaling\_VMx} = (T_{End\_Scaling} - T_{Start\_Scaling}) \quad (6)$$

$$T_{Existing\_VMx} = \left(T_{End\_Run} - T_{Start\_Run}\right) - (T_{Scaling\_VMx}) \quad (7)$$

where $T_{Scaling\_VMx}$ is the time required for scaling VM*x* measured by seconds. $T_{Start\_Scaling}$ is the time when the scaling is started and $T_{End\_Scaling}$ is the time when the scaling is ended. $T_{Existing\_VMx}$ is the running time of the existing VM*x* before scaling. T_{Start_Run} is the start time of the running task and T_{End_Run} is the end time of the running task.

To predict the cost of VM*x* before scaling, equation (8) is proposed:

$$\begin{aligned}\mathrm{VM}x_{Pred\_Cost\_PMi} = &\left(\left(\mathrm{VM}x_{ReqvCPUs\_PMi} \times \frac{VMx_{Pred\_U\_PMi}}{100}\right) \times \left(Cost\_vCPU \times T_{Existing\_VMx}\right)\right) \\ &+ \left(\mathrm{VM}x_{Pred\_R\_U\_PMi} \times \left(Cost\_GB \times T_{Existing\_VMx}\right)\right) \\ &+ \left(\mathrm{VM}x_{Pred\_D\_U\_PMi} \times \left(Cost\_GB \times T_{Existing\_VMx}\right)\right) \\ &+ \left(\mathrm{VM}x_{Pred\_N\_U\_PMi} \times \left(Cost\_GB \times T_{Existing\_VMx}\right)\right) \\ &+ \left(\mathrm{VM}x_{Pred\_E\_PMi} \times \left(Cost\_kWh \times T_{Existing\_VMx}\right)\right)\end{aligned} \quad (8)$$

where $\mathrm{VM}x_{Pred\_Cost\_PMi}$ is the predicted total cost of the VM*x* before scaling on the source PM*i*. $\mathrm{VM}x_{Pred\_R\_U\_PMi}$ is the predicted resource usage of RAM times the cost for that resource for a period of time before scaling $T_{Existing\_VMx}$. We

consider the similar notation for the CPU, disk and network resources on PM*i*. $VMx_{Pred\_E\_PMi}$ is the predicted energy consumption of the VM*x* times the electricity cost as announced by the energy providers. Thus, the cost of the scaled VM*x* after scaling decision on the destination PM*j* will be predicted using Equation 8, but substituting PM*i* with PM*j*, $T_{Existing\_VMx}$ with $T_{Scaling\_VMx}$ and so on for each resource such as CPU, RAM, disk, network and energy. Besides, additional license fees α for the new VM when (horizontal scaling) is performed which considered as constant (£0.1/hr).

To get the predicted total cost for VM*x* before and after scaling can be given by:

$$VMx_{Total\_Pred\_Cost} = VMx_{Pred\_Cost\_PMi} + VMx_{Pred\_Cost\_PMj} \quad (9)$$

IV. EXPERIMENTAL SETUP

This section describes the environment and the details of the experiments conducted in order to evaluate the proposed Performance and Energy-based Cost Prediction Framework. The prediction process starts by firstly predicting the PMs/VMs workload using the (**`auto.arima`**) function in R package[2] and then completing the cycle of the framework and considering the correlation between the physical and virtual resources to predict power consumption of the VMs on multiple PMs. After that, the total cost is predicted for the scaled VMs' based on their predicted workload and power consumption.

A number of experiments have been designed and implemented on a local Cloud Testbed with the support of the Virtual Infrastructure Manager (VIM), OpenNebula[3] version 4.10, and KVM hypervisor for the Virtual Machine Manager (VMM). This Cloud Testbed includes a cluster of 8 commodity Dell servers. Four of these servers with four core X3430 and eight core E31230 V2 Intel Xeon CPU were used. The servers include 16GB RAM and 500GB hard drives. Also, each server has a Watt meter[4] attached to directly measure the power consumption. Heterogeneous VMs are created and their monitoring is performed through Zabbix[5], which is also used for resources usage monitoring. Rackspace[6] is used as a reference for the VMs configurations. Three types of VMs, small, medium and large are provided with different capacities. The VMs are allocated with 1, 2 and 4 vCPUs, 1, 2 and 4 GB RAM, 10 GB disk and 1 GB network, respectively. The cost of the virtual resources are set according to ElasticHosts[7] and VMware[8]; and the cost of energy according to CompareMySolar[9].

In terms of the workload patterns, Cloud applications can experience different workload patterns based on the customers' usage behaviours, and these workload patterns consume power differently based on the resources they utilise. Several cloud workload patterns are identified in [16]. The periodic workload pattern is considered as it fits nicely with the performance variation modelling. Thus, a number of direct experiments have been conducted to synthetically generate periodic workload by using *Stress-ng*[10] in order to stress all resources on different types of VMs. The generated workload of each VM type has four-time intervals of 30 minutes each. The first three intervals will be used as the historical data set for prediction, and the last interval will be used as the testing data set to evaluate the predicted results.

V. RESULTS AND DISCUSSION

This section presents the quantitative evaluation of the Performance and Energy-based Cost Prediction Framework. The figures below show the predicted results for three types of VMs, small, medium and large, running on multiple PMs based on historical periodic workload pattern. Because of space limitation, only medium VM results are shown.

In Algorithm 1, when VM*x* is overloaded and exceeds the predefined (upper threshold), instead of immediately auto-scaling VMs, the prediction model is used to minimise the number of VMs scaling and avoid unnecessary scales caused by the small peaks in the workload. However, when VM*x* is overloaded and exceeds the predefined (max_upper threshold), the overloaded VM*x* will be scaled in order to prevent service performance degradation and allocated to an appropriate PM*j* which has sufficient resources and is potentially most energy efficient. In order to achieve the auto-scaling without degrading the performance of VM*x*, the candidate PM*j* (CPU and RAM) resources need to be carefully managed. Since the PM*i* upper threshold (85%) is predefined and PM*j* has available resources to accept the allocated VM*x*, the performance of the auto-scaled VM*x* is not affected. It is also checked that the candidate PM*j* utilisation will not exceed the upper threshold for allocating of the incoming VM*x*. Figure 9 (a, b, c and d) depict the results of the scaled VM*x* predicted versus the actual workload, including CPU, RAM, disk, and network usage for the VM*x*. Despite the periodic utilisation peaks, the predicted VM*x* CPU, RAM and network workload results closely match the actual results, which reflects the capability of the ARIMA model to capture the historical seasonal trend and give a very accurate prediction accordingly. The predicted VM*x* disk workload also matches the actual workload, but with less accuracy as compared to the CPU, RAM and network prediction results. This can be justified because of the high variations in the generated historical periodic workload pattern of the disk not closely matching in each interval. Beside the predicted mean values, the figures also show the high and low 95% and 80% confidence intervals.

In terms of prediction accuracy, a number of metrics have been used to evaluate the results, such as *Mean Error (ME), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), and Mean Absolute Percent Error (MAPE)*; further details about these accuracy metrics can be found in [17].

---

[2] http://www.r-project.org/
[3] https://opennebula.org/
[4] https://www.powermeterstore.com
[5] https://www.zabbix.com/
[6] https://www.rackspace.com/cloud/servers/pricing
[7] https://www.elastichosts.co.uk/pricing/
[8] https://www.vmware.com/cloud-services/pricing-guide
[9] http://blog.comparemysolar.co.uk/electricity-price-per-kwh-comparison-of-big-six-energy-companies/
[10] http://kernel.ubuntu.com/~cking/stress-ng/

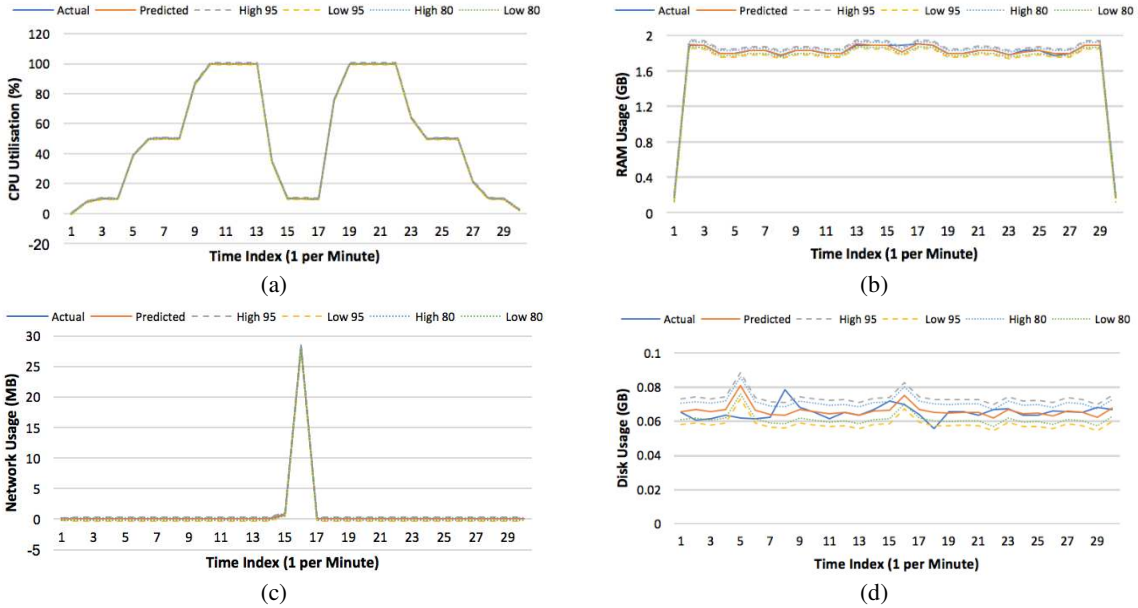

Fig. 9. Prediction Results for a Medium VM.

The accuracy of the predicted VM*x* workload (CPU, RAM, disk, network) based on periodic workload is evaluated using these accuracy metrics, as summarised in Table II.

TABLE II. Prediction Accuracy for a Medium VM.

Parameters	ME	RMSE	MAE	MPE	MAPE
CPU	0.019355	0.2451	0.12275	-3.1443	3.576033
RAM	0.001976	0.0189	0.00588	0.11509	0.333648
Disk	-0.00005	0.0030	0.00181	-0.2380	2.716369
Network	0.000197	0.0940	0.01848	-181.96	190.5482

The proposed framework can predict the power consumption for a number of VMs when running on the source PM*i* and the candidate PM*j* (based on Step 6, Equation 4 in Section III). Figures 10 and 11 show the results of the predicted power consumption for the VM*x* running on a number of PMs using different scaling strategies based on the predefined instance size and the self-configuration instance size. By observing the figures, the self-configuration auto-scaling outperforms the predefined one, since the predicted power consumption is lower. It should be mentioned that the predicted power consumption attribution for each VM is affected by the variation in the predicted PM CPU utilisation of all the VMs.

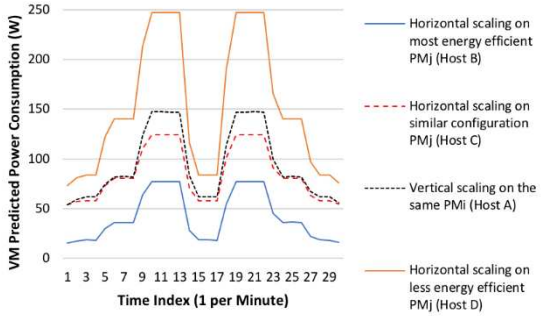

Fig. 10. Predicted VM*x* Power Consumption using a Predefined VM Size - Scaling on a number of candidate PMs.

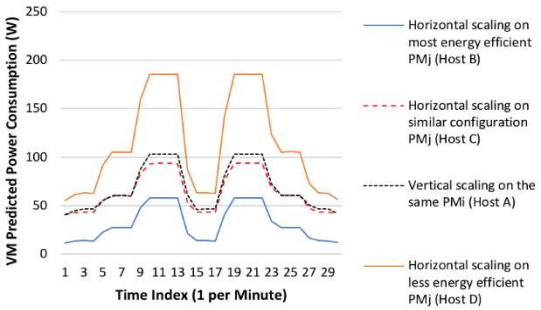

Fig. 11. Predicted VM*x* Power Consumption using Self-Configuration VM Size - Scaling on a number of candidate PMs.

This framework is also capable of predicting the auto-scaling total cost for VM*x* running on a number of PMs using different scaling strategies as shown in Figure 12, along with self-configuration cost (based on Step 7, Equation 9 in Section III). This helps select the most suitable cost-efficient scaling strategy. As shown in Figure 12, the choice of scaling can have a significant impact on the cost of the scaled VM*x* (e.g. horizontal scaling using most energy efficient PM can be more cost-effective than horizontal scaling when using less energy efficient PM).

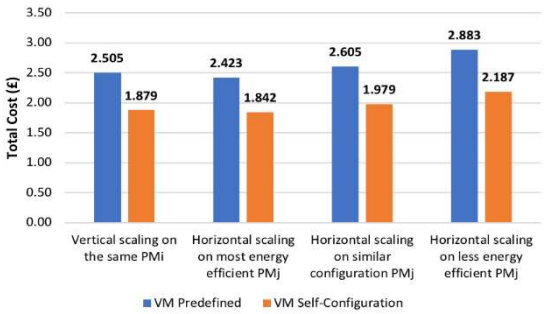

Fig. 12. Predicted Auto-Scaling Total Cost (Predefined VM*x* Size Scaling vs Self-Configuration VM*x* Size Scaling).

In addition, Figure 13 shows the results of the predicted self-configuration cost that can incur less VM*x* scaling cost compared to predefined instance size choices. The cost comparison shows that choosing self-configuration VM*x* size can achieve 25% cost-saving compared to the predefined VM*x* size on the same PM*i* when vertical scaling is performed. In case of horizontal scaling, about 24% cost-saving can be gained on a most energy efficient host, on a similar host configuration as well as on a less energy efficient host PM*j*. Furthermore, a similar cost-saving can be gained when performing the self-configuration mechanism for small and large VMs, as shown in Figure 14.

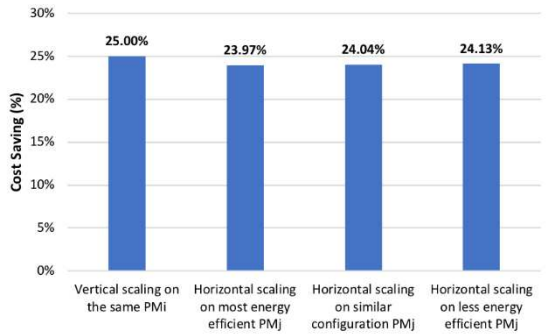

Fig. 13. Cost Saving by Self-Configuration VM*x* Size Scaling.

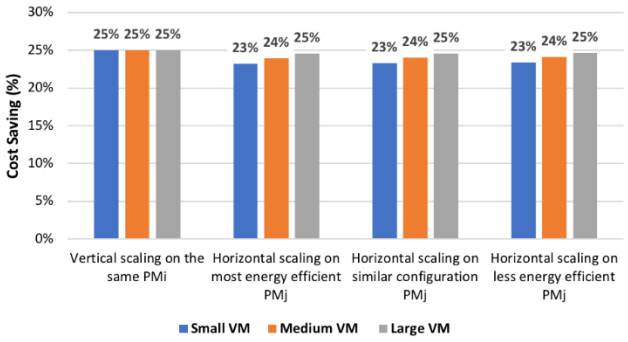

Fig. 14. Cost Saving by Self-Configuration for all VMs Size Scaling.

Despite the high variation of the workload utilisation in the periodic pattern, the accuracy metrics indicate that the predicted VMs workload and power consumption achieve good prediction accuracy along with the predicted auto-scaling total cost.

VI. CONCLUSION AND FUTURE WORK

This paper has presented and evaluated a new Performance and Energy-based Cost Prediction Framework that dynamically supports VMs auto-scaling decision, and demonstrates the trade-off between cost, power consumption, and performance. This framework predicts the auto-scaling total cost by considering the resource usage, power consumption and performance variation of heterogeneous VMs based on their usage and size, which reflect the physical resource usage and power consumption by each VM. The results show that the proposed framework can predict the resource usage, power consumption, total cost for the auto-scaled VMs with a good prediction accuracy based on periodic workload patterns.

As a part of future work, we intend to extend our approach by considering the live migration aspects (re-allocation) to further understand the capability of the proposed work.

REFERENCES

[1] Amazon_EC2, "Amazon EC2 Service Level Agreement," 2013. [Online]. Available: https://aws.amazon.com/ec2/sla/. [Accessed: 01-Oct-2017].

[2] R. Han, L. Guo, M. M. Ghanem, and Y. Guo, "Lightweight resource scaling for cloud applications," in *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 2012, pp. 644–651.

[3] S. Dutta, S. Gera, A. Verma, and B. Viswanathan, "SmartScale: Automatic application scaling in enterprise clouds," in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 221–228.

[4] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 500–507.

[5] A. Biswas, S. Majumdar, B. Nandy, and A. El-Haraki, "An auto-scaling framework for controlling enterprise resources on clouds," in *Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015*, 2015, pp. 971–980.

[6] M. A. S. Netto, C. Cardonha, R. L. F. Cunha, and M. D. Assuncao, "Evaluating auto-scaling strategies for cloud computing environments," in *Proceedings - IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*, 2015, pp. 187–196.

[7] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *Proceedings - 11th ACM/IEEE International Conference on Grid Computing*, 2010, no. Grid, pp. 41–48.

[8] W. Fang, Z. H. Lu, J. Wu, and Z. Y. Cao, "RPPS: A novel resource prediction and provisioning scheme in cloud data center," in *Proceedings - 2012 IEEE 9th International Conference on Services Computing, SCC 2012*, 2012, pp. 609–616.

[9] Z. Gong, X. Gu, and J. Wilkes, "PRESS: PRedictive Elastic reSource Scaling for cloud systems," in *Proceedings of the 2010 International Conference on Network and Service Management, CNSM 2010*, 2010, vol. 2010, no. Cnsm, pp. 9–16.

[10] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd Symposium on Cloud Computing*, 2011, p. 5:1--5:14.

[11] M. Aldossary, I. Alzamil, and K. Djemame, "Towards Virtual Machine Energy-Aware Cost Prediction in Clouds," in *14th International Conference on Economics of Grids, Clouds, Systems, and Services*, 2017, pp. 119–131.

[12] K. Djemame *et al.*, "PaaS-IaaS Inter-Layer Adaptation in an Energy-Aware Cloud Environment," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 127–139, 2017.

[13] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[14] W. Dargie, "A stochastic model for estimating the power consumption of a processor," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1311–1322, 2015.

[15] I. Alzamil and K. Djemame, "Energy Prediction for Cloud Workload Patterns," in *13th International Conference on Economics of Grids, Clouds, Systems, and Services*, 2016, pp. 160–174.

[16] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns*. 2014.

[17] R. J. Hyndman and G. Athanasopoulos, "Measuring forecast accuracy," *OTexts*, 2013. [Online]. Available: at www.otexts.org/fpp/2/5. [Accessed: 01-Oct-2017].