J. Parallel Distrib. Comput. 102 (2017) 103–114

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Energy efficiency for cloud computing system based on predictive optimization

Dinh-Mao Bui [a], YongIk Yoon [b], Eui-Nam Huh [a], SungIk Jun [c], Sungyoung Lee [a,*]

[a] Computer Engineering Department, Kyung Hee University, Suwon, Republic of Korea
[b] Department of Multimedia Science SookMyung Women's University, Seoul, Republic of Korea
[c] HPC system research section/Cloud computing department/ETRI, Daejeon, Republic of Korea

HIGHLIGHTS

- Balance between energy efficiency and quality of service in the cloud computing.
- Apply prediction technique to enhance the usefulness of monitoring statistics.
- Design optimal energy efficiency architecture to orchestrate the cloud system.
- After VM consolidation, adaptively turn-off idle physical machines to save energy.

ARTICLE INFO

Article history:
Received 25 January 2016
Received in revised form
26 October 2016
Accepted 6 November 2016
Available online 19 December 2016

Keywords:
Energy efficiency
IaaS cloud computing
Predictive analysis
Convex optimization
Gaussian process

ABSTRACT

In recent years, power consumption has become one of the hottest research trends in computer science and industry. Most of the reasons are related to the operational budget and the environmental issues. In this paper, we would like to propose an energy-efficient solution for orchestrating the resource in cloud computing. In nature, the proposed approach firstly predicts the resource utilization of the upcoming period based on the Gaussian process regression method. Subsequently, the convex optimization technique is engaged to compute an appropriate quantity of physical servers for each monitoring window. This quantity of interest is calculated to ensure that a minimum number of servers can still provide an acceptable quality of service. Finally, a corresponding migrating instruction is issued to stack the virtual machines and turn off the idle physical servers to achieve the objective of energy savings. In order to evaluate the proposed method, we conduct the experiments using synthetic data from 29-day period of Google traces and real workload from the *Montage* open-source toolkit. Through the evaluation, we show that the proposed approach can achieve a significant result in reducing the energy consumption as well as maintaining the system performance.

1. Introduction

Widely accepted as a system model for providing service, cloud computing has gradually obtained the popularity among the platforms that manage the operation of data center. In nature, cloud computing indeed changes the routine of using and scaling the physical infrastructure. Instead of separately utilizing different combinations of servers, storage disks and network facilities for each client, it would be more convenient and robust to provide a transparent access to the computing resources *via* the Internet connection. This can be seen as an effort to unify the capacity of multiple computing nodes to achieve higher level of service composition. In the perspectives of cloud service provider, cloud computing makes it possible to reduce the management cost and the energy consumption which consequently save the budget.

Theoretically, cloud computing mostly relies on the virtualization technology. First, the users interact with the system by choosing the package of service based on their requirement and budget. Depending on the submission of users, the cloud orchestrator allocates the requested resources as virtual machines (VMs). These VMs are usually hosted on the same physical server or cluster according to the objectives of service provider. The procedure of translating from the resource demand to the VM configuration is conducted transparently without the awareness of the user. In fact,

* Corresponding author.
E-mail addresses: mao.bui@khu.ac.kr (D.-M. Bui), yiyoon@sm.ac.kr (Y. Yoon), johnhuh@khu.ac.kr (E.-N. Huh), sijun@etri.re.kr (S. Jun), sylee@oslab.khu.ac.kr (S. Lee).

http://dx.doi.org/10.1016/j.jpdc.2016.11.011
0743-7315/

the resource provisioning is done elastically, which is the main reason that makes the cloud computing flexible enough to engage in the large scale system. However, in order to do the tasks efficiently, the challenge of balancing between the system performance and the power consumption has been emerged as a crucial issue.

In order to reduce the power consumption, it is worth to list out the source of energy burning as well as the methodology to obtain the energy savings. Clearly, most of the computing facilities use the energy to maintain their working status. Therefore, the waste of power can be originated from the inefficient utilization of provisioned facilities. The common approach to achieve the energy reduction is to dynamically scale down the size of running clusters. With the help of virtualization, the target can be done by applying the VM migration. It means that the VMs can be stacked into a minimum number of physical machines (PMs). After that, the decision of turning on/off the running PMs can help to satisfy the requirement of maintaining the performance as well as reducing the used power.

Although the idea to compact the size of cloud clusters is quite interesting, the problem of latency and inaccuracy in measuring and monitoring the resource utilization might affect the precision of making the optimization. This problem might lead to the inappropriate decisions in migrating and stacking the VMs. Consequently, the effectiveness of the energy savings method might be degraded drastically. To overcome this issue, we would like to propose an approach based on the predictive optimization technique. In our approach, the Gaussian process regression (GPR) is applied to provide the predictive monitoring information instead of the aforementioned obsolete data. Theoretically, GPR can be considered as one of the most effective regression techniques in terms of accuracy, flexibility and robustness. By using GPR in processing the monitoring data, the awareness of futuristic trends on the system utilization can be obtained. Subsequently, this benefit helps the convex optimization to produce more adaptive and precise energy savings scheme for the cloud orchestrator. To summarize, in this research, we couple the prediction technique and the convex optimization to solve two questions. The first one is how to effectively reduce the power consumption in cloud computing based on early predictive system statistics. Subsequently, the remaining question to answer is whether it is possible to find the boundary solution that balances the energy savings and the quality of services.

In particular, our contributions focus on two main points as follows:

- We apply our prediction technique [7] to provide the predictive statistics that enhance the usefulness of monitoring data. Note that the useful monitoring data is defined to support the system making early appropriate reactions just in time by providing the futuristic analysis. In our approach, this special kind of data is effectively used to consolidate the virtual machines through the optimization procedure.
- We design an energy efficiency model using convex optimization to optimally create the migrating instructions for the orchestrator. By consolidating the VMs *via* the migration mechanism, the orchestrator can reduce the power consumption as well as improve the cluster utilization by turning-off the idle physical machines, but still keep an acceptable performance for the currently running tasks.

It is worth noting that the meaning of acceptable performance mainly involves two conditions. The first condition is to maintain the quality of services which is specified in the service level agreement (SLA) document. In case the first condition is violated, the second condition is to minimize the catastrophic effects on the system performance as well as balance the penalty cost with regard to the power savings in the optimization module. When these conditions are still firmly hold, the system performance are kept in an acceptable level. The mechanism of how to preserve these conditions can be found in the following sections.

The remainder of the paper is organized as follows. In Section 2, we introduce some related works in the area of energy efficiency. Section 3 presents an overview of our proposed architecture. Next, Section 4 illustrates our analysis and proposal to enhance the nature of the monitoring data by using GPR technique. Section 5 includes the optimization steps to minimize the size of running facilities with regard to the predictive analysis provided by GPR. Section 6 evaluates the effectiveness of the proposed approach. Finally, Section 7 gives the conclusion of paper and outlines our future work.

2. Related works

Energy efficiency in cloud computing can be achieved *via* two main approaches: by optimizing the operation of physical machines (PMs), or by consolidating the virtual machines (VMs) in the clusters. While the former approach targets on reducing the energy consumption of individual server, the latter one tries to schedule and migrate the virtual machines to serve the user's requirement on a minimum number of PMs. Both of these approaches have their own pros and cons as shown below.

For the PMs optimization approaches, the dynamic voltage and frequency scaling (DVFS) [8,24] is a technique to automatically change the frequency and the voltage 'on the fly'. The purpose of this technique is to save the power and mitigate the heat from processors. Less heat generated allows the cooling system to be under-performed or even turned off. This fact also leads to the noise reduction and better energy savings. The main drawback of this technique is to reduce the performance proportionally to the saved energy. Because of this reason, DVFS is usually used in laptops and hand-held devices which prefer the long-lasting battery rather than the power of computation [22]. Besides, this technique is also used in the system which prefers low heat emission. For high performance computing or computational intensive system, DVFS might not be a suitable solution. The DVFS technique can be implemented either on hardware and software levels. For example, Intel and AMD applied DVFS in their hardware throttling technologies, namely SpeedStep and Cool'n'Quite, respectively. On software level, the Linux OS governors can control the frequency of processors *via* the ACPI interfaces.

In the other hand, the power capping [20] or power budgeting is usually used in the power-limited data centers. This solution enables the capability of controlling the power budget on system-level or rack-level. In fact, power capping is considered to be a key factor for engaging the power shifting, which is the technique to individually allocate the power to each server in the cluster. According to the original design of this idea, the high priority servers might receive more power than the lower ones. Because power capping can be done out-of-band, the failure of operating system does not affect the functionality of the power management. The implementation of power capping can be found in HP Intelligent Power Discovery and IBM Systems Director.

Nowadays, most of the recent processors are equipped with the power saving modes, namely C-states [26]. This configuration enables the capability for the idle processors to turn off the unused components in order to save the power. Although C-states existed in laptops for lengthening the battery life for a while, they are rarely used in the server because of the disadvantage of deep sleep. This issue happens when the processor sticks in some energy saving states and yet requires for a long period to wake up. Clearly, the problem of deep sleep state makes the server unable to fulfill the requirement of computational intensive tasks. Another issue of C-states is that this mechanism depends on the functionality of

the system kernel which sometimes fails to provide the suitable instructions.

For the VM consolidation approaches, most of the solutions focus on solving the problem of VMs placement with regard to the performance of PMs [4]. Since the VMs and the PMs can be partially considered as the objects and the bins, respectively, the aforementioned problem is narrowed down to the bin packing issue which is NP-hard [2]. As a consequence, the heuristics approaches such as best fit decreasing [2] and first fit decreasing [10] are known to be the standard solutions. By applying these approaches, the VMs can be distributed to the smaller size of PMs pool. Because of this reason, the family of bin packing algorithms seems to be really attractive to provide a closely optimal solution over the cloud system. Unfortunately, the requirement of heuristics approaches is that the number of VMs must be fixed and known in advance at the starting time. Obviously, this requirement directly violates the elasticity and the multi-tenancy features of cloud computing. Moreover, the fluctuation in the workload of computing nodes definitely makes the bin packing approaches defective and subsequently violates the quality of service.

To overcome the above issue, the prediction techniques can be the potential solutions. By equipping the system with the capability of anticipating the utilization of computing nodes in near future, the orchestrator can make better decision and mitigate the negative effect of workload changes. Basically, there have been a number of approaches to predict the workload such as Hidden Markov model [13] and polynomial fitting [37] in grid systems. However, these solutions are not appropriate in dealing with the cloud workloads which follow the philosophy of on-demanded resource provisioning. Lastly, the approach in [12] uses the Wiener filter as a predictor to estimate the cluster workload. Unfortunately, the Wiener filter only works best with the stationary signal and noise spectrum. This reason makes the solution potentially inapplicable for predicting the utilization of the cloud cluster.

On the other hand, the research in [11,30,3] propose many specific schedulers to cope with other system aspects such as network traffic, communication rates, system powers and resource reconfiguration in the fog computing and the networked data centers. These approaches might help to optimize the network throughput, the wireless transmission rate as well as the resource balancing, particularly for TCP/IP vehicular connections and networking in data centers; finally, also obtaining the energy efficiency.

By examining the related works, we have come to a conclusion that although the research on energy efficiency exists, not many researchers have thoroughly attempted to balance the energy savings and the performance of cloud computing within a reasonable price. The reason is that none of the existing works has considered the nature of the cloud system as a truly probabilistic queuing system and proposed the corresponding suitable solution. Based on this motivation, we would like to propose our approach to practically and thoroughly solve the energy issue of cloud computing by using the knowledge of queuing theory and convex optimization.

3. Proposed architecture

3.1. System description

In the perspectives of physical system, cloud computing can be considered as the clusters as seen in Fig. 1. These clusters consist of a number of physical servers which host many virtual machines inside. Also in this figure, each virtual machine possesses various utilization and capacity of computing resources. It is worth noting that the mentioned infrastructure is homogeneous system for the reason of computational convenience. It means that every

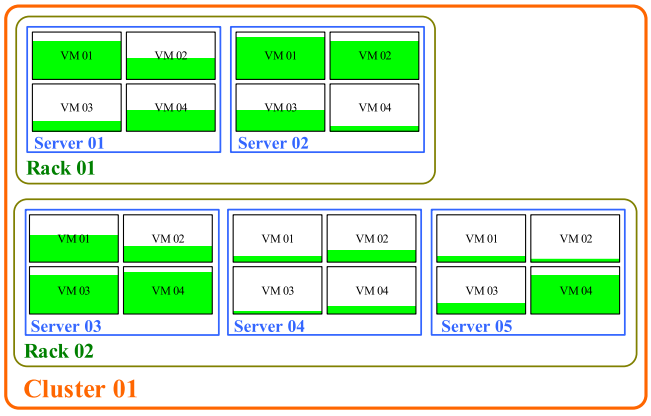


Fig. 1. Representation of resource utilization of virtual machines over a physical cluster.

physical machines are equipped with the same type and the same capacity of facilities. Generally, the heterogeneous system might be treated with a similar approach by adding some extra weighted parameters to reflect the differences in configurations. However, this issue is not in the scope of the paper and would not be considered at this moment.

For setting the final goal of research, the proposed architecture is targeted to reduce the power consumption of the cloud. To achieve this goal, the appropriate decision of turning off the idle physical servers is chosen. Theoretically, a machine in the idle state burns the electricity up to 60% [14,23,18] of the peak energy (which is used to run the machine in peak performance). Contrary to this point, turning on a machine just costs 23.9% [29] of the peak power. Moreover, idle-energy waste is also increased by the losses in delivering the power as well as maintaining the cooling system, which would be an upsurge in power consumption requirements. Therefore, shutting the idle machines down can save more power than keeping them in the idle mode, even we have to pay the extra costs to turn on the machines subsequently. Based on this philosophy, we would like to introduce the architecture of energy efficiency management (E2M) system depicted in Fig. 2. The objective of this architecture is to optimally schedule and re-locate the virtual machines over the physical servers of the cloud clusters. Finally, the idle physical nodes are temporarily terminated to save the power. The functionality of each component in E2M system is as follows:

- The Ganglia [32] monitoring component takes responsibility of collecting the information of resources utilization. Ganglia is considered to be simple but robust and effective to monitor most of the required metrics. The objects of this component are the PMs as well as the VMs denoted as the monitored nodes. Periodically, Ganglia reliably provides the input statistics to the predictor component.
- The predictor, which is built on the Gaussian process regression, is responsible to produce the predictive statistics of the next monitoring epoch based on the data coming from Ganglia. This futuristic utilization information of the virtual machines and the physical servers is subsequently used in the energy optimizer.
- The energy optimizer, as its name, does the optimization to balance between saving the energy and maintaining the system performance. Basically, this component makes the decision of how many physical servers are appropriate to cover the VMs requests. Besides, the energy optimizer also evaluates the potential nodes to issue the migration. After that, the migrating instruction would be sent to the cloud orchestrator for further execution.

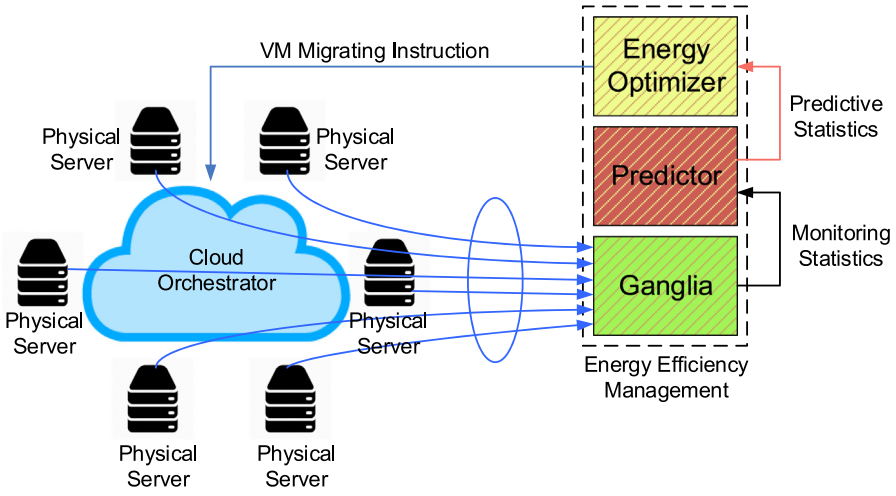


Fig. 2. Architecture of energy efficiency management (E2M) system.

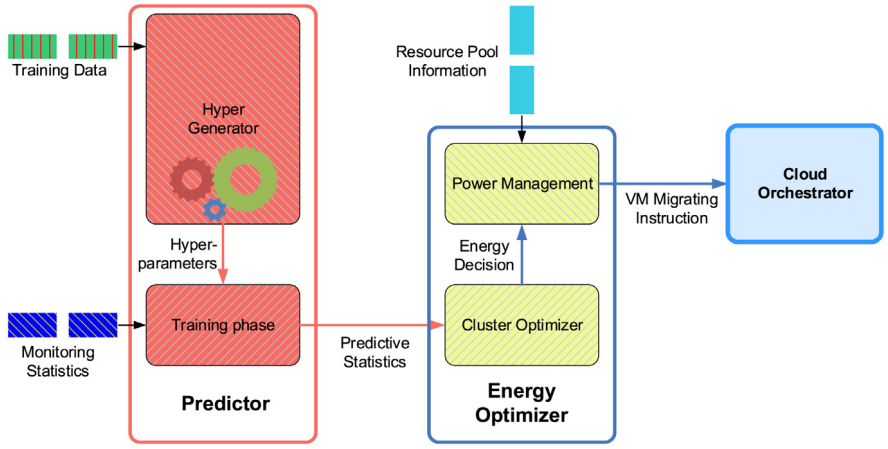


Fig. 3. Flowchart of energy efficiency management (E2M) system.

For the convenience of presentation, the migrating instruction created in the energy optimizer would be introduced in the next section. This instruction comprises the nature of migration as well as the selection of the source and destination PMs to extract and send the VMs, respectively.

3.2. Migrating instruction

In order to clarify the nature of VM migration, we would like to mention that Xen's live migration is engaged in the orchestrator. By definition, live migration is used to ensure the operation and mitigate the down time of the VMs during the transfer period. Due to this reason, the availability and the seamlessness of the service can be preserved. In fact, when this migration mechanism is activated, only the memory of VMs is sent *via* the network. Therefore, the VM is kept running and suffered only for a very short period of postponed state. Indeed, there is an evidence [31] showing that live migration might cost less than 100 ms for the amount of downtime, which is acceptable in the service providing. Besides, this technique helps to get rid of the network contention and the processing overhead. Eventually, in case there is any serious degradation of system performance, the suspended PMs would be started by using Wake on LAN protocol. In our design, the migrating instruction is generated in the power management component of the energy optimizer as shown in Fig. 3.

In order to issue the VM migration, we assume that the energy optimizer has collected enough predictive statistics of monitored nodes and calculated the optimal number of PMs. From this assumption, the problem now is how to decrease/increase the current number of PMs to the aforementioned optimal one. Obviously, choosing the physical source node to extract the target VMs is not an obstacle. In fact, the lowest utilization node in terms of CPU and memory is chosen to be the source node. Then, the only remaining issue is to choose the placement for the extracted VMs. Depending on the knowledge of operating system [35,15], as well as the queuing theory, various critical conditions must be considered in order to judge the destination. First, the destination node should not be the same as the source node. Moreover, the destination node should not be blocked (the node should not be too busy, as it would be failed to accept the target VM). Finally, when these conditions are matched, the node with the highest estimation of utilization is selected as the next destination node. In the following part, we study how to estimate the blocking rate of a physical node. This study relies on the queuing theory of task processing.

Assuming that an arriving task comes to a specified node, the task might 'see' an average number of currently running tasks which are being held in progress. Because the arrival counting follows the Poisson process, the Poisson arrival see time averages (PASTA) theory [16] is applicable when estimating the desired average number of running tasks. Following the PASTA theory, the average arrival time is considered to be equal to the time average or the expectation of waiting time in the considered node. By applying the Pollaczek–Khintchine formula [16], the expectation waiting

time $\mathbf{E}_j$ of node j can be calculated as follows:

$$\mathbf{E}_j(W) = \frac{\lambda_\tau {}^1\!/\!_{\mu^2}}{2(1-\lambda_\tau {}^1\!/\!_{\mu})}, \tag{1}$$

in which, λ_τ is the arrival rate of the tasks at time τ, μ is the service rate of the considered physical server j and W is the expected scheduling latency in this server [16]. After estimating the arrival average, the blocking rate of the server j can be obtained via dividing this value by the service rate as follows:

$$\mathbf{BR}_j = \frac{\mathbf{E}_j(W)}{\mu}. \tag{2}$$

By evaluating the blocking rate $\mathbf{BR}_j$ of each physical server j, it is easy for the power management component to determine the destination to migrate the target VM. If there is no available node for destination promotion, the migration procedure is postponed until there is at least one unblocked node. Finally, in the worst case scenario when all of the running nodes are blocked, a turning-on procedure is issued *via* Wake on LAN protocol in order to start the inactive servers one by one. Practically, this strategy helps the cloud system to maintain an acceptable performance.

4. Prediction model

Before establishing the above migrating instruction, the optimization procedure should be issued in advance. In order to do that, this section illustrates the study of how we build the prediction model. In our architecture, the objective of prediction is to anticipate the utilization of the resources. The anticipation process is mostly executed in the predictor component as seen in Fig. 3. Inside this component, the Bayesian learning and Gaussian process regression (GPR) are employed as the inference technique and probability framework, respectively. Because the input data for this model is the time-series utilization, the curve-fitting is preferred over the function mapping for mapping approach.

The working of prediction depends on the monitoring statistics which reflect the historical utilization of the cluster facilities. In order to conveniently organize and execute these data, we define a special terminology, namely the monitoring window, which describes a fixed period of workload collection. Assuming that the input data is a limited collection of time location $x = [x_1, x_2, x_3, \ldots, x_n]$ (the gap between two consecutive elements of this set x forms up a monitoring window denoted by m), a finite set of random variable $y = [y_1, y_2, y_3, \ldots, y_n]$ represents the corresponding joint Gaussian distribution of historical monitoring statistics of the resource with regard to the time order. This set over the time actually forms up the Gaussian process:

$$f(y|x) \sim \mathcal{GP}\big(m(x), k(x,x')\big), \tag{3}$$

with

$$m(x) = \mathbb{E}\big(f(x)\big), \tag{4}$$

$$k(x,x') = \mathbb{E}\Big(\big(f(x)-m(x)\big)\big(f(x')-m(x')\big)\Big), \tag{5}$$

in which, $m(x)$ is the mean function, evaluated at the time location variable x, and $k(x,x')$ is the covariance function, also known as the kernel function [25]. By definition, the kernel function is a positive-definite function which is used to define the prior knowledge of the underlying relationship. In addition, the kernel function comprises some special parameters that specify its own shape. These parameters are referred to as the hyper-parameters. Because the input data comes to the predictor as a set of n time locations, the kernel should be engaged in the matrix form as follows:

$$\mathbf{K} = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{pmatrix}. \tag{6}$$

Generally, the square-exponential (SE) kernel, also known as the radial basis function (RBF) kernel, is chosen as the basic kernel function. In reality, SE kernel is favored in most of the Gaussian process applications, because this kernel requires only a few parameters to calculate. The formula for SE kernel is described as follows:

$$k_{SE}(x,x') = \sigma_f^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right), \tag{7}$$

in which, σ_f is the output-scale amplitude and l is the time-scale of the variable x from one moment to the next. l also stands for the bandwidth of the kernel and the smoothness of the function. In the next step, we evaluate the posterior distribution of the Gaussian process. Assuming that the incoming value of the input data is (x_*, y_*), the joint distribution of training output y, and test output y_* is as follows:

$$p\left(\begin{bmatrix} y \\ y_* \end{bmatrix}\right) = \mathcal{GP}\left(\begin{bmatrix} m(x) \\ m(x_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K}(x,x') & \mathbf{K}(x,x_*) \\ \mathbf{K}(x_*,x) & \mathbf{K}(x_*,x_*) \end{bmatrix}\right), \tag{8}$$

here, $\mathbf{K}(x_*,x_*) = k(x_*,x_*)$, $\mathbf{K}(x,x_*)$ is the column vector made from $k(x_1,x_*)$, $k(x_2,x_*)\ldots$, $k(x_n,x_*)$. In addition, $\mathbf{K}(x_*,x) = \mathbf{K}(x,x_*)^\top$ is the transposition of $\mathbf{K}(x,x_*)$. Subsequently, the posterior distribution over y_* can be evaluated with the below mean m_* and covariance C_*:

$$m_* = m(x_*) + \mathbf{K}(x_*,x)\mathbf{K}(x,x')^{-1}(y-m(x)), \tag{9}$$

$$C_* = \mathbf{K}(x_*,x_*) - \mathbf{K}(x_*,x)\mathbf{K}(x,x')^{-1}\mathbf{K}(x,x_*), \tag{10}$$

then

$$p(y_*) \sim \mathcal{GP}(m_*, C_*). \tag{11}$$

The best estimation of y_* is the mean of the above distribution:

$$\bar{y}_* = \mathbf{K}(x_*,x)\mathbf{K}(x,x')^{-1}y. \tag{12}$$

In addition, the uncertainty of the estimation is captured in the variance of the distribution as follows:

$$var(y_*) = \mathbf{K}(x_*,x_*) - \mathbf{K}(x_*,x)\mathbf{K}(x,x')^{-1}\mathbf{K}(x,x_*). \tag{13}$$

Theoretically, the predictive system statistics retrieved from GPR is highly accurate compared to other regression methods [28]. However, the standard implementation of GPR costs $O(n^3)$ for computational complexity and $O(n^2)$ for storage complexity when calculating n training points of the dataset [9]. Most of the complexity comes from calculating the matrix inverse and log determinant in both hyper-parameter learning and training phases, which are two main phases of GPR. This can be seen as the drawback of GPR, which prevents the prediction from being quickly calculated on a large dataset. To solve this problem, we use the complexity reduction technique applying to each phase of the prediction process. For the hyper parameters learning phase, the combination of the law of log determinant, the fast Fourier transform and the stochastic gradient descent are proposed to reduce the complexity to $O(n \log n)$ with n stands for the number of training points. For the training phase, we have proposed the parallel improved fast Gauss transform (pIFGT) to achieve $O(n)$ for the complexity, which is really fast to do the prediction. These techniques have been already proposed in our previous

research [7]. Further discussion can be found in detail in the original paper.

5. Energy optimization

After having the predictive statistics, the final step is to calculate the suitable amount of needed physical machines (PMs). It is worth mentioning that the amount is referred to as the energy decision and used to produce the VM migrating instruction. The calculation process is known as the optimization, which is mainly held in the energy optimizer as seen in Fig. 3. Analytically, the energy optimizer consists of two components: the cluster optimizer and the power management that produce the energy decision as well as the VM migrating instruction, respectively. Note that the power management keeps track of the cluster resource pool to estimate the blocking rate, which is presented in the previous migrating instruction section. In this section, the working of the cluster optimizer would be included in detail.

5.1. System description

As mentioned in the system description section, the target system is known in advance to be a homogeneous system. At the monitoring window m, the system comprises $P_m \in \mathbb{N}^+$ physical machines. The number of P_m might be changed as time progresses due to the scaling factor of fault tolerant issue. Each physical machine possesses a set of facilities denoted by $f \in \mathbb{N}^+$. Usually, we focus on two types of facilities which are CPU and memory. These facilities actually provide the major information to operate the optimization. To measure this information in cluster level, the global utilization $U_m^{f_i} \in \mathbb{R}^+$ as well as the individual utilization $I_m^{f_i} \in \mathbb{R}^+$ of each resource are needed to collect continuously at every monitoring window m. Note that f_i stands for a general element i in the set f. For example, f_c might stands for CPU, particularly. Among the set of P_m physical machines, only the running machines, denoted by a_m at the monitoring m, are worth considering.

At this stage, the target is to reduce a_m but still maintains the system performance. If this requirement can be obtained, the cloud system might significantly save the power, condense the utilization and ensure the quality of service. In the next sections, the modeling of these imperative objectives would be described one by one in detail.

5.2. Performance modeling

In this section, we model the performance issue into the cost. It is obvious that consolidating the VMs to a reduced pool of the PMs leads to the high utilization in each node. However, if the utilization increases without the control, the performance is critically affected. This issue might result in system overhead and high latency in task scheduling [38]. Due to this reason, we would like to take into account the performance issue as a penalty cost to ensure the quality of service.

There are evidences [29,6] showing that the utilization of a machine and the average latency in task scheduling (hereinafter: average latency) are in a linear relationship. Therefore, it is reasonable to measure the performance of a cluster *via* the average latency denoted by l_m. Certainly, the variable l_m should be limited by the threshold l, which is described in the service level agreement (SLA) document. Usually, the violation of SLA occurs when the whole system comes to the state of peak performance. We denote the CPU resource of the exhausted nodes by variable z. Due to the goal of getting high utilization for every nodes, the utilization of z is given by:

$$I_m^z = \max_{f_c}\{I_m^{f_c}\} = \max_{f_c}\left\{\frac{U_m^{f_c}}{a_m C^{f_c}}\right\}. \tag{14}$$

We would like to simplify this equation by setting $\delta_m = \max\{U_m^{f_c}/C^{f_c}\}$, then (14) turns into:

$$I_m^z = \max_{f_c}\{I_m^{f_c}\} = \frac{\delta_m}{a_m}. \tag{15}$$

From (15), it is obvious that I_m^z is a decreasing function of a_m. It means that the average latency would increase along with the decline in the number of running machines. Subsequently, the average latency l_m can be calculated by using the expectation waiting time $\mathbf{E}_z$ of the exhausted CPU as described in the migrating instruction section:

$$l_m(I_m^z) = \mathbf{E}(z) = \frac{\lambda_m {}^1/_{\mu^2}}{2(1-\lambda_m {}^1/_{\mu})}, \tag{16}$$

where λ_m is the arrival rate of the tasks at monitoring window m, μ is the service rate of CPU z. After having the result of l_m, we can determine whether the quality of service is still satisfied or not by comparing with the threshold l. If there is any violation, the penalty cost denoted by C_m^p should be applied. This cost is exponentially increased with regard to the number of violations and given by:

$$C_m^p = w_m s_p (l_m(I_m^z) - l)^+, \tag{17}$$

where w_m and s_p stand for the weight factor showing the criticalness of the violation at monitoring window m and the basic price of the penalty, respectively.

Since the penalty cost depends on the measurement of exhausted nodes and the weight factor w_m, the performance degradation is safely limited. Furthermore, the weight factor w_m is calculated based on the fault tolerance β, which measures the difference between the most recent average latency l_{m-1} and the current one l_m. The value of β is normalized into the range of $[-0.1, 0.1]$ by using the Irritates algorithm. Initially, w_0 is set to 1; *i.e.*, $w_0 = 1$, and for the monitoring window m, w_m is updated as

$$w_m = (1+\beta) w_{m-1}^+. \tag{18}$$

The weight factor w_m is designed to make positive impact on the system if the average latency increases. It means there would be a limited number of running nodes that are slightly kept under-performance to deal with the fluctuation in VM requests. Practically, the redundancy is necessary to reduce the criticalness of SLA violation as well as improve the stability of the physical infrastructure.

5.3. Energy modeling

Prior to this section, we know that the power consuming overtime is the result of maintaining the operation of physical machines [1,17]. Regarding to each physical node, the relationship between the cost of energy consumption and the running devices at monitoring window m can be mapped into a linear increasing function as follows:

$$e_m = P_{idle} + P_{running}. \tag{19}$$

Denote s_m as the basic price of electricity at monitoring window m, with a_m running machines of the cluster, the energy cost denoted by C_m^e is given by:

$$C_m^e(a_m) = s_m a_m e_m = s_m a_m (P_{idle} + P_{running}). \tag{20}$$

In (20), the energy of running machines cannot be cut down because of the performance, so this part should be dropped from the energy modeling as follows:

$$C_m^e(a_m) = s_m a_m P_{idle}. \tag{21}$$

5.4. Cluster optimizer

In this section, we include the description of the energy optimizer in detail. After having the modeling of performance and energy, the optimization can be conducted with regard to the goal of saving the energy and maintaining the performance. As discussed in the introduction section, the quantity of running machines needs to be minimized to reduce the energy consumption. This requirement is modeled by using the convex optimization technique as follows:

$$\min_{0\le a_m\le P_m} (w_m s_p (l_m(I_m^z) - l)^+ + s_m a_m P_{idle}). \tag{22}$$

Denote the optimal number of running machines by a_m^*, as shown in the performance modeling section, the function $l_m(I_m^z)$ is a decreasing function of a_m. Thus, we have the condition on a_m^* as below:

$$a_m^* \le \frac{\delta_m}{l_m^{-1}(l)}. \tag{23}$$

The reason for this condition is that if $a_m^* \ge {}^{\delta_m}/_{l_m^{-1}(l)}$ and $(l_m(I_m^z) - l) = (l_m({}^{\delta_m}/_{a_m}) - l)^+ = 0$, then the decline of a_m^* to ${}^{\delta_m}/_{l_m^{-1}(l)}$ can mitigate the energy consumption when still preserving the performance. Because of that, (22) can be clarified as shown below:

$$\min_{0\le a_m\le \frac{\delta_m}{l_m^{-1}(l)}} (w_m s_p (l_m(I_m^z) - l)^+ + s_m a_m P_{idle}). \tag{24}$$

The Lagrangian function of the problem shown by (24) is given by:

$$\mathcal{L}(a_m, \gamma) = w_m s_p \left(l_m\left(\frac{\delta_m}{a_m}\right) - l\right)^+ + s_m a_m P_{idle} + \gamma\left(a_m - \frac{\delta_m}{l_m^{-1}(l)}\right) + \alpha(0 - a_m). \tag{25}$$

Using Karush–Kuhn–Tucker (KKT) conditions on the Lagrangian function shown by (25), we would have the partial derivatives as follows:

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial a} = s_m P_{idle} - w_m s_p \delta_m \frac{\partial l_m\left(\frac{\delta_m}{a_m}\right)}{\partial a}\left(\frac{1}{a_m^2}\right) + \gamma - \alpha, \\ &\alpha a_m = 0, \\ &\gamma\left(\frac{\delta_m}{l_m^{-1}(l)} - a_m\right) = 0, \\ &0 \le a_m \le \frac{\delta_m}{l_m^{-1}(l)}, \\ &\alpha, \gamma \ge 0. \end{aligned} \tag{26}$$

There are two cases of solving this optimization problem, which are:

- $\gamma > 0$ or $\alpha > 0$. In this case, the conditions become boundary conditions, then the feasible solution can be $a_m^* = {}^{\delta_m}/_{l_m^{-1}(l)}$ or $a_m^* = 0$.
- $\gamma = 0$ and $\alpha = 0$. Because the function $l_m(.)$ following Pollaczek Khinchin formula which is convex [21], the solution for the above optimization problem can be achieved by solving the first condition of (26) as shown below:

$$a_m^* = \delta_m + \sqrt{\frac{w_m s_p \delta_m}{2\mu s_m P_{idle}}}, \tag{27}$$

where μ is the service rate of the CPU z as shown in (16). Obviously, the solution shows the fact that the optimization is an effort to keep balance between reducing the energy consumption and maintaining the quality of service as discussed before.

Table 1
Google cluster's organization and configuration.

Quantity of nodes	Category	CPU	RAM
1	1	0.50	0.06
3	3	1.00	0.50
5	1	0.50	0.97
5	1	0.50	0.03
52	1	0.50	0.12
126	2	0.25	0.25
795	3	1.00	1.00
1001	1	0.50	0.75
3863	1	0.50	0.25
6732	1	0.50	0.50

Table 2
Summary of Google traces' characteristics.

Time span	# of PMs	#VM requests	Trace size	# of users
29 days	12 583	>25 M	>39 GB	925

6. Performance evaluation

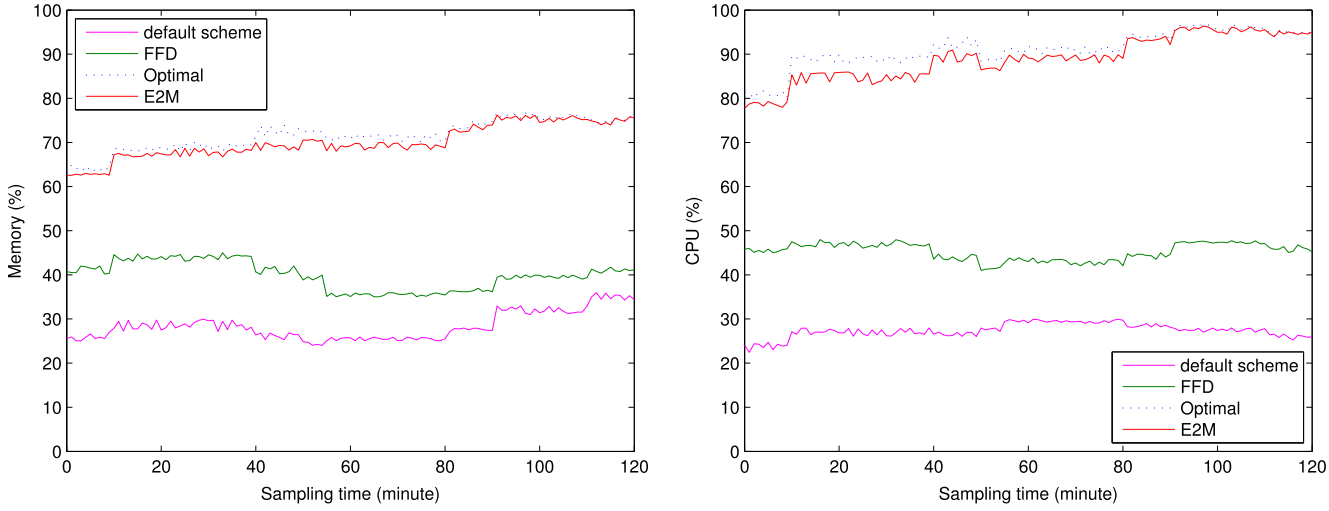
6.1. Experiment design

In this section, we conduct the experiments to evaluate the performance of the proposed approach (E2M). All the experiments are benchmarked on a system of 16 homogeneous servers. Each of them is equipped with the quad-core Intel Xeon E7-2870 2.4 Ghz, 12 GB of RAM and hosts up to 8 VMs. For the connection, the Gigabit Ethernet Controller is used to communicate between these servers. In summary, the physical infrastructure is able to host up to 128 connected VMs to serve the experiments.

There are two kinds of experiments in the evaluation. In the first experiment, Google traces [33] are used to simulate the workload for training the energy efficiency management system. Announced by Google, these traces provide the monitoring data on cluster level from more than 12,500 machines during the period of 29 days. The organization and configuration of these machines can be found in Table 1. Because the whole cluster is heterogeneous in terms of system type and capacity, we decide to choose the largest homogeneous set which consists of 6732 machines to conduct the experiment. It is worth noting that the capacities of traced machines are normalized by Google. Due to the super large size of Google traces (over 39 GB of compressed data), we choose some random parts of the aforementioned homogeneous subset to do the experiments. Each part consists of 24-hour period of traces. The total size of this portion is over 2.26 GB. For the ease of presentation, we scale the maximum length of measurement to 60 s. We also use this duration as the length of our monitoring window.

For more information, Google traces consist of greater than 25 million tasks. Each task can be referred to as a VM request and be scheduled on a physical machine (PM). Since there are a number of VM requests sending to the PM, these requests should follow a queuing model. Obviously, the queuing delay is unavoidable and can be referred to as the aforementioned average latency in task scheduling. The computing resources, which are used to initialize the VMs, are up to the user's demand. It is worth noting that even the user does not utilize all the requested resources intensively, these resources are maintained until the corresponding requests are fully released. Only when all the VMs in a specific PM are released, the PM is allowed to shut down. The summary of Google traces' characteristics can be found in Table 2.

In the second experiment, we use the *Montage* workflow [19,5] to test the energy efficiency of the proposed approach in real-world scenario. It is worth noting that *Montage* is an effort of NASA to assemble the flexible image transport system (FITS) into



(a) Average memory utilization (higher is better).

(b) Average CPU utilization (higher is better).

Fig. 4. Utilization evaluation of the proposed method in Google traces experiment.

custom mosaics [34]. There are four steps to produce an image mosaic in *Montage*. First, the geometry of the input images is discovered from the input FITS keywords and used to calculate the geometry of the output mosaic. After that, the re-projection of the input images to the same spatial scale and rotation is performed. Subsequently, the background radiation in the input images is modeled to achieve common flux scales and background levels. Finally, the re-projected, background-corrected images are co-added to create the output mosaic. The *Montage* execution is measured to consist of 10 429 tasks. The detail information of *Montage* can be found in the original references and website.

6.2. Implementation

For the information of scheduling algorithm, we engage the default first fit algorithm which is popularly used in many well-known orchestrators such as Nimbus, OpenNebula and OpenStack. We conduct the experiments under four schemes for comparison purpose. These schemes are as follows:

- The default schemes: which is the scheme that all the machines of the cluster are turned on from the beginning to the end of the benchmark. No power saving technique is applied in this scheme.
- The greedy first fit decreasing (FFD) approach: represents the modified VM placement algorithm which is originated from an interesting research of VM consolidation [27] to improve the energy efficiency in cloud computing environment. This approach actually sorts the VMs by the decreasing order of CPU requests and places the queue of VMs into the first host fitting the resource requirement. The VM placement is conducted according to the bin packing methodology.
- The proposed approach (E2M): is implemented and evaluated with all the prediction and optimization enabled.
- The optimal energy-aware approach: represents the scheme in which the optimizer cares only for the energy consumption. Because of that, the power management component in this case always produce the optimal solutions to greatly save the energy regardless of the performance.

6.3. Utilization analysis

All four schemes start running with the same configuration of the cluster. For the evaluation purpose, the memory and the CPU utilizations of these schemes are collected continuously and shown in Fig. 4(a) and (b), respectively, during the benchmarking time. By analyzing the monitoring data, it is clear to comment that the default scheme with no power saving solution possesses very low resource utilization. The reason for this issue is that many PMs are left idle. Meanwhile, the FFD approach achieves a better utilization in comparison to the default scheme. Unfortunately, this approach is unable to cover the fluctuation in the cluster workload and leave many nodes in under-utilized circumstances after placing the VMs. Obviously, due to this reason, FFD falls behind in the race with the optimal solution. Contrary to the FFD algorithm, the proposed approach predicts exactly the required number of running machines and scales the cluster based on this number. Obviously, the utilization of the system is enhanced significantly and sometimes can closely catch the level of optimal solution. It is worth noting that the optimization of E2M is limited according to the performance. Because of that, there would be always a distance compared with the optimal solution. Also, the context switching might hurt the final result. In fact, this issue happens to every solutions in any realistic system and can be considered as an unavoidable problem. Finally, the summary of overall cluster utilization can be found in Table 3.

Table 3
Average utilization of cluster facilities.

	Default	FFD	E2M	Optimal
CPU	27.40%	45.26%	89.02%	91.13%
Memory	28.35%	39.87%	70.12%	71.49%

6.4. Energy vs. performance

Due to the fact that our first experiment based on Google traces which only consist of the cluster workload, we have to use an external study of energy measurement, which is defined in the CloudESE research [29], to estimate the energy consumption. The detail of measuring parameters can be found in the original paper and summarized in Table 4. As shown in Figs. 5 and 6, the default scheme burns a huge amount of electricity by keeping all the physical machines running even without any workload. The FFD approach, in the other hand, burns less energy than the default scheme but still wastes the power in many under-utilized nodes. Without the migration and workload anticipation, FFD cannot preemptively compact the size of PMs pool. This drawback is the

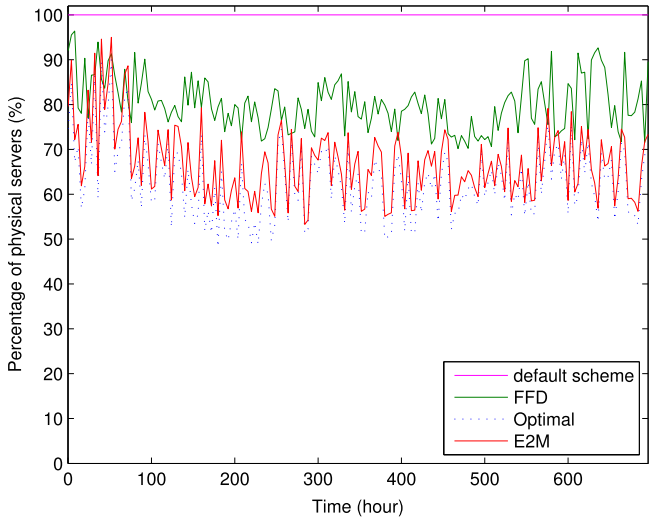


Fig. 5. Percentage of active physical servers in Google traces experiment.

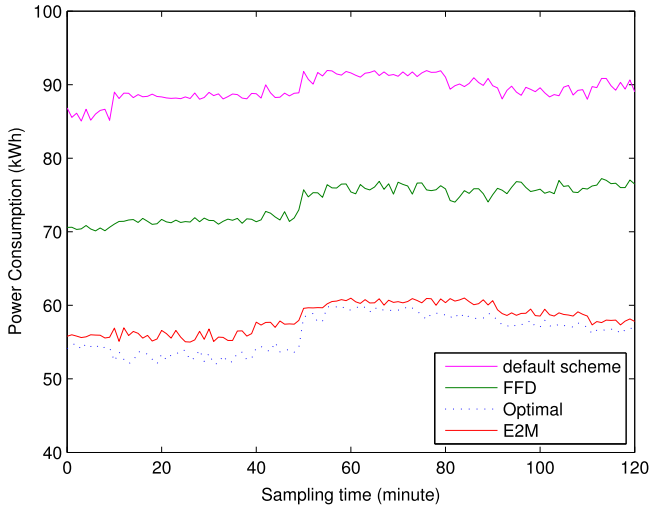


Fig. 6. Power consumption evaluation of the proposed method in Google traces experiment (lower is better).

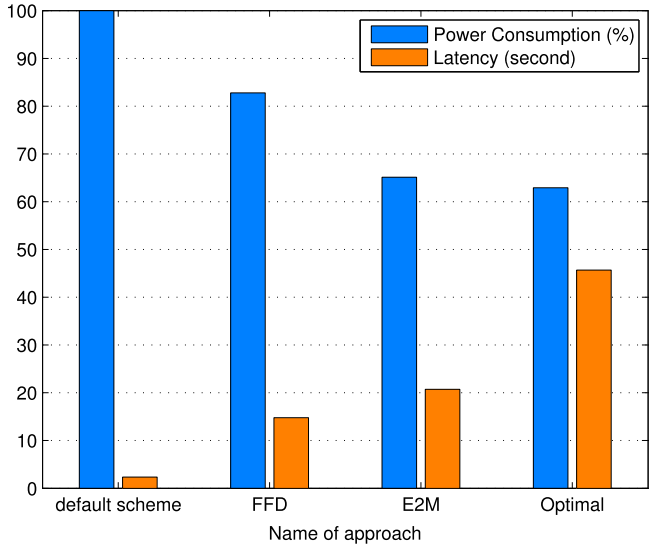


Fig. 7. Power consumption vs. average latency in Google traces experiment.

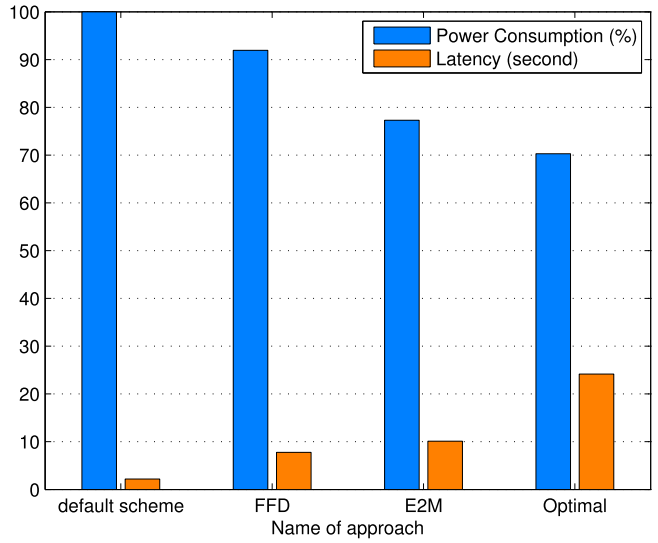


Fig. 8. Power consumption vs. average latency in *Montage* experiment.

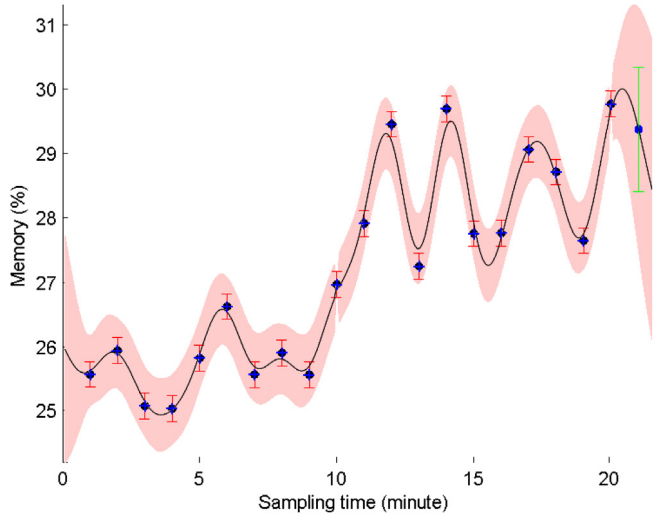
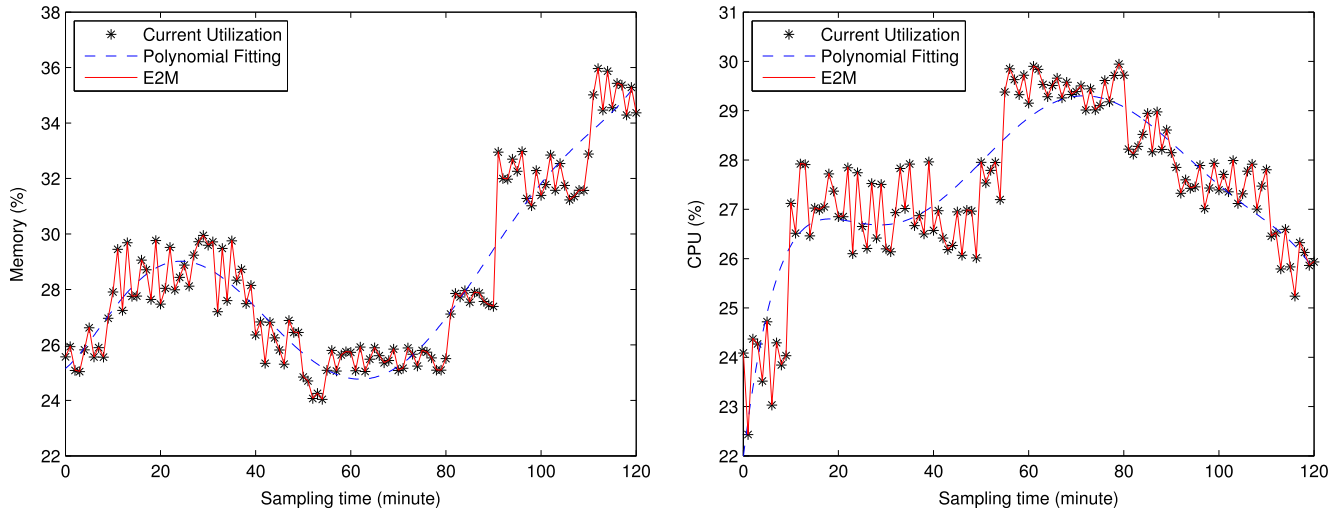


Fig. 9. Mean prediction and error bar of the proposed method given 20 training points in the memory utilization benchmark.

nature of bin packing algorithm family as discussed before. In contrast to the first two schemes, the energy consumption of the proposed approach is dropped dramatically due to the prediction of the futuristic workload and the optimization of PMs pool with regard to the performance. Note that most of the idle machines are completely turned-off rather than keeping in the idle state. Obviously, there is a short distance between E2M and optimal solution as discussed in the utilization analysis section.

For the energy saving measurement, as seen in Fig. 7, the proposed method can reduce up to 34.89% of energy consumption in comparison to the default scheme. It is worth noting that the optimal solution only reduces 37.08% of energy consumption. It means that our approach closely achieves the final goal of energy efficiency. Even in the *Montage* experiment, as seen in Fig. 8, the proposed method can cut down 22.71% of the energy consumption compared with the default scheme. Clearly, this amount of energy savings is a significant result in real-world scenario. Besides, it is worth mentioning that the energy loss in the empirical experiment is always higher than in the experiment based on synthetic data due to the environmental conditions and devices. Last but not least, also in Figs. 7 and 8, the average latencies in task scheduling of the proposed method are around 20.74 and 10.12 s which are 54.72%



(a) Memory. (b) CPU.

Fig. 10. Prediction evaluation of the proposed method in Google traces experiment.

and 58.14% less than the optimal solution, when conducting on Google traces and *Montage* toolkit, respectively.

6.5. Prediction evaluation

The last part is the evaluation of the prediction in terms of accuracy. It is worth noting that we also use Google traces as our training and validation datasets. The training set is engaged in the hyper-generator of the predictor to find the hyper-parameters. Fig. 9 shows that for an accuracy benchmark of 20 consecutive testing points in the memory utilization, the mean prediction adapts quite well to the testing data with 95% of confidence maintained by the variance. Subsequently, the validation set plays the role of investigator to evaluate the correctness of the prediction. In order to assess this metric, we would like to include the error measurement mechanism. The error measurement is also referred to as the error function which is based on the root mean square error (RMSE) method. Note that the RMSE method is stricter than the popularly-used mean square error (MSE) method. Let n stands for the size of validation dataset, by using the RMSE error function, the gap between the current value and the predicted one can be evaluated at every steps of the anticipation. Following is the equation of RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \mathcal{R}_i^{(m)} - \mathcal{P}_i^{(m)}}{n}}, \tag{28}$$

where $\mathcal{R}_i^{(k)}$ and $\mathcal{P}_i^{(k)}$ stand for the current and predicted values of the resource utilization at monitoring window m, respectively. The variable i stands for the order of the values in validation array. Since our prediction technique based on the improved fast Gauss transform (IFGT) [36], which strictly controls the RMSE limited to 10^{-11}, the result of the anticipation is very close to the real resource utilization. In fact, as shown in Fig. 10(a) and (b), the proposed method outperforms the polynomial fitting regression for both memory and CPU utilizations, respectively.

7. Conclusion

In this research, we propose a comprehensive energy-efficient resource orchestrating solution for cloud computing system. Based on the workload prediction and the convex optimization, the proposed approach can provide the appropriate migrating instructions for the cloud to stack the VMs and cut down the number of running PMs. Consequently, the power consumption is significantly reduced while still maintaining the quality of service. Besides, by evaluating on both Google traces and *Montage* toolkit, we show that our approach is not only effective on the synthetic data, but also save the energy on the realistic system. Furthermore, due to the fact that most of the computation focuses on creating the predictive workload information, the complexity of the corresponding component, which is $O(n \log n)$, should be taken into account as the complexity of the proposed approach. In reality, this level of low complexity is quite feasible to be integrated as an energy savings module into any popular cloud orchestrators (such as OpenStack, OpenNebula, Nimbus, etc.). In future, we plan to extend our idea to the heterogeneous system to improve the adaptation of the proposed approach for real-life scenario.

Table 4
Energy estimation parameters.

Parameter	Value	Unit
E_{sleep}	107	Watt
E_{idle}	300.81	Watt
E_{peak}	600	Watt
$E_{\text{active}\rightarrow\text{sleep}}$	1.530556	Watt-hour
$E_{\text{sleep}\rightarrow\text{active}}$	1.183333	Watt-hour
$E_{\text{active}\rightarrow\text{off}}$	1.544444	Watt-hour
$E_{\text{off}\rightarrow\text{active}}$	11.95	Watt-hour

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1015) supervised by the IITP (Institute for Information & communications Technology Promotion).

This work was also supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. R0101-15-237, Development of General-Purpose OS and Virtualization Technology to Reduce 30% of Energy for High-density Servers based on Low-power Processors).

This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) NRF-2014R1A2A2A01003914.

This work was also supported by the Industrial Core Technology Development Program (10049079, Develop of mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea).

References

[1] Z. Abbasi, G. Varsamopoulos, S.K. Gupta, Thermal aware server provisioning and workload distribution for Internet data centers, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, 2010, pp. 130–141.

[2] Y. Ajiro, A. Tanaka, Improving packing algorithms for server consolidation, in: Int. CMG Conference, 2007, pp. 399–406.

[3] E. Baccarelli, N. Cordeschi, A. Mei, M. Panella, M. Shojafar, J. Stefa, Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study, IEEE Netw. 30 (2) (2016) 54–61.

[4] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.

[5] G.B. Berriman, E. Deelman, J.C. Good, J.C. Jacob, D.S. Katz, C. Kesselman, A.C. Laity, T.A. Prince, G. Singh, M.-H. Su, Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand, in: SPIE Astronomical Telescopes+ Instrumentation, International Society for Optics and Photonics, 2004, pp. 221–232.

[6] M.A. Blackburn, G. Grid, Five Ways to Reduce Data Center Server Power Consumption, Green Grid, 2008.

[7] D.-M. Bui, H.-Q. Nguyen, Y. Yoon, S. Jun, M.B. Amin, S. Lee, Gaussian process for predicting cpu utilization and its application to energy efficiency, Appl. Intell. 43 (4) (2015) 874–891.

[8] T.D. Burd, T. Pering, A.J. Stratakos, R.W. Brodersen, et al., A dynamic voltage scaled microprocessor system, IEEE J. Solid-State Circuits 35 (11) (2000) 1571–1580.

[9] K. Chalupka, C.K.I. Williams, I. Murray, A framework for evaluating approximation methods for gaussian process regression., CoRR abs/1205.6326.

[10] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: a survey, in: Approximation Algorithms for NP-Hard Problems, PWS Publishing Co, 1996, pp. 46–93.

[11] N. Cordeschi, M. Shojafar, E. Baccarelli, Energy-saving self-configuring networked data centers, Comput. Netw. 57 (17) (2013) 3479–3491.

[12] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, Energy-efficient resource allocation and provisioning framework for cloud data centers, IEEE Trans. Netw. Serv. Manag. 12 (3) (2015) 377–391.

[13] C. Dabrowski, F. Hunt, Using markov chain analysis to study dynamic behaviour in large-scale grid systems, in: Proceedings of the Seventh Australasian Symposium on Grid Computing and e-Research-Volume 99, Australian Computer Society, Inc., 2009, pp. 29–40.

[14] X. Fan, W.-D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: ACM SIGARCH Computer Architecture News, Vol. 35, ACM, 2007, pp. 13–23.

[15] C. Fiandrino, D. Kliazovich, P. Bouvry, A. Zomaya, Performance and energy efficiency metrics for communication systems of cloud computing data centers, IEEE Trans. Cloud Comput. PP (99) (2015) http://dx.doi.org/10.1109/TCC.2015.2424892, 1–1.

[16] R. Gallager, Stochastic Processes: Theory for Applications, Cambridge University Press, 2013, URL http://books.google.co.kr/books?id=CGFbAgAAQBAJ.

[17] B. Guenter, N. Jain, C. Williams, Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning, in: INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 1332–1340.

[18] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, X. Zhu, Vmware distributed resource management: Design, implementation, and lessons learned, VMware Tech. J. 1 (1) (2012) 45–64.

[19] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Gener. Comput. Syst. 29 (3) (2013) 682–692.

[20] C. Lefurgy, X. Wang, M. Ware, Power capping: a prelude to power shifting, Cluster Comput. 11 (2) (2008) 183–195.

[21] C. Loch, Time competition is capability competition, INSEAD, 1994.

[22] J.R. Lorch, A.J. Smith, Improving dynamic voltage scaling algorithms with pace, in: ACM SIGMETRICS Performance Evaluation Review, Vol. 29, ACM, 2001, pp. 50–61.

[23] D. Meisner, B.T. Gold, T.F. Wenisch, Powernap: eliminating server idle power, in: ACM Sigplan Notices, Vol. 44, ACM, 2009, pp. 205–216.

[24] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, R. Rajkumar, Critical power slope: understanding the runtime effects of frequency scaling, in: Proceedings of the 16th International Conference on Supercomputing, ACM, 2002, pp. 35–44.

[25] K. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Trans. Neural Netw. 12 (2) (2001) 181–201. http://dx.doi.org/10.1109/72.914517.

[26] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall, Reducing network energy consumption via sleeping and rate-adaptation., in: NSDI, Vol. 8, 2008, pp. 323–336.

[27] T.K. Okada, A.D.L.F. Vigliotti, D.M. Batista, A.G. vel Lejbman, Consolidation of vms to improve energy efficiency in cloud computing environments, 2015, pp. 150–158.

[28] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, in: Adaptive Computation and Machine Learning, MIT Press, 2005, URL http://www.gaussianprocess.org/gpml/chapters/.

[29] I. Sarji, C. Ghali, A. Chehab, A. Kayssi, Cloudese: Energy efficiency model for cloud computing environments, in: 2011 International Conference on Energy Aware Computing, (ICEAC), IEEE, 2011, pp. 1–6.

[30] M. Shojafar, N. Cordeschi, E. Baccarelli, Energy-efficient adaptive resource management for real-time vehicular cloud services, IEEE Trans. Cloud Comput. PP (99) (2016) http://dx.doi.org/10.1109/TCC.2016.2551747, 1–1.

[31] A. Verma, P. Ahuja, A. Neogi, pmapper: power and migration cost aware application placement in virtualized systems, in: Middleware 2008, Springer, 2008, pp. 243–264.

[32] What is ganglia? http://ganglia.sourceforge.net/, (accessed: 13.08.15).

[33] What is google traces? https://github.com/google/cluster-data/, (accessed: 21.01.16).

[34] What is montage? http://montage.ipac.caltech.edu/, (accessed: 02.08.16).

[35] X. Wu, Performance Evaluation, Prediction and Visualization of Parallel Systems, in: The International Series on Asian Studies in Computer and Information Science, Springer, US, 1999, URL http://books.google.co.kr/books?id=IJZt5H6R8OIC.

[36] C. Yang, R. Duraiswami, N. Gumerov, L. Davis, Improved fast gauss transform and efficient kernel density estimation., in: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, 2003, pp. 664–671 vol.1. http://dx.doi.org/10.1109/ICCV.2003.1238383.

[37] Y. Zhang, W. Sun, Y. Inoguchi, Cpu load predictions on the computational grid*, in: Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006. CCGRID 06. Vol. 1, IEEE, 2006, pp. 321–326.

[38] Q. Zhang, M.F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, J.L. Hellerstein, Dynamic energy-aware capacity provisioning for cloud computing environments, in: Proceedings of the 9th International Conference on Autonomic Computing, ACM, 2012, pp. 145–154.

Dinh-Mao Bui received the B.S. degree in Computer Science from the Computer Engineering Department at Ton Duc Thang University, Vietnam, in 2009 and the M.S. degree in Data Communication and Networking from the Posts and Telecommunications Institute of Technology, Vietnam, in 2012. He is now working toward the Ph.D. degree in the Department of Computer Engineering at Kyung Hee University, Korea. His research interests include Convex Optimization, Stochastic Process and Big Data.

Yongik Yoon is a Professor for Dept. of Multimedia Science in SookMyung Women's University, South Korea. He received M.S and Ph.D. degree from Computer Science of KAIST, in 1985 and 1994. He had researched for 15 years (1983–1997) like a member of senior Researcher of ETRI, in Korea. Also he had worked a visiting professor in University of Colorado at Denver, in USA, for three years (2004–2007). His Research Interests are smart services for future life, middleware for smart/future life, collaboration service model in mobile cloud environment, and intelligent mobile customer application platform. He is now a Member of ACM, IEEE, KIISE (Korean Institute of Information Scientists and Engineers), KIPS (Korean Information Processing Society), and OSIA (Open Standard and Internet Association).

Eui-Nam Huh is a chair professor in Dept. of Computer Engineering. His interesting research areas are: Cloud Computing, Big Data Computing, IoT, Distributed Real Time System, Network Security. He earned Master's degree in Computer Science from University of Texas, USA in 1995 and Ph.D. degree from the Ohio University, USA in 2002. He has also served for the WPDRTS/IPDPS, SOICT, APIST, ICUIMC and ICCSA community as various chair positions since 2003.

SungIk Jun is a principal member of engineering staff for High-Performance System Research team in ETRI (Electronics Telecommunication Research Institute), South Korea. He received M.S degree from Computer Science of Chung-Ang University, in 1987. He had researched for Real-time OS team during 15 years (1987–2001) like a member of senior Researcher of ETRI, in Korea. Also he had worked a team leader Wireless Security Application Research team for eight years (2001.3–2009.4). His research interests are Operating System, Wireless Security, and M2M for future life.

Sungyoung Lee received his Ph.D. degree in Computer Science from Illinois Institute of Technology (IIT), Chicago, Illinois, USA in 1991. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Korea since 1993. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, Illinois, USA from 1992 to 1993. His current research focuses on Ubiquitous Computing, Cloud Computing, Intelligent Computing and eHealth.