Profit-Aware DVFS Enabled Resource Management of IaaS Cloud

Shahzad Ali[1], Si-Yuan Jing[2] and She Kun[1]

[1] School of Computer Science & Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China

[2] College of computer Science, Leshan Normal University Leshan, 614004, China

Abstract

Power consumption of IaaS Cloud is growing at a rapid pace during the last decade. This leads to high operational and maintenance cost, and also reducing the profit margin of IaaS Cloud providers. Apparently profit margin depends on operational cost and revenue earned to comply with the Service Level Agreement (SLA) on the basis of achieved performance level. In this paper, we are building the relationship between energy consumption of servers and fulfillment of SLA (QoS) in IaaS Cloud. So we propose a framework to maximize revenues from SLA achieved requests on the energy efficient resource management in IaaS Cloud, in which the servers are equipped with DVFS module. Servers are not allowed to be turned on or turned off. Moreover, we consider a common cost of virtual machine (VM) migration or start/stop. This framework explains four sub-problems, i.e. service placement which encapsulating in VM, resource capacity allocation, load balancing and dynamic voltage/frequency scaling. A hybrid optimization solution is proposed to solve these problems and its effectiveness, compared to existing approach, is evaluated using synthetic workloads. Results show that our solution as compared to others can yield significant profit gain for the cloud provider.

Keywords: *IaaS Cloud Data center, Energy Efficiency Resource Management, Service Level Agreement (SLA), Virtualization, Dynamic Voltage and Frequency Scaling (DVFS).*

1. Introduction

Cloud computing is defined as, "A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet."

Although cloud computing has been widely adopted by the industry, the research on cloud computing is still at developing stage. Some existing problems have not been completely addressed, while new challenges keep emerging from industry applications. Energy management is one of them is challenging research issues [1] [2].

Only recently has the hardware and software been available to support the concept of utility computing on a large scale. The concepts inspired by the notion of utility computing have recently combined with the requirements and standards of Web 2.0 [3] to create Cloud computing [1]. It is becoming increasingly difficult to ignore the importance of power saving issues in cloud computing environment because last few years have witnessed a rapid development of cloud data centers. In the same way that performance has been central to systems evaluation (e.g., measuring completed tasks per unit of time), energy-efficiency (e.g., completed tasks per unit of energy) is quickly growing in importance for minimizing IT costs. Reducing processor energy consumption has received considerable attention in recent years.

Power consumption of Internet and Web servers accounts for 8% of the US electricity consumption. In [4] it is shown that service centers consume 1.5% of the power produced in the USA, and will reach 4.5% within 5 years. By reducing the demand for energy, the amount of CO_2 produced each year by electricity generators can also be mitigated. For example, generating electricity for the next generation large-scale data centers would release about 25M tons of additional CO_2 each year [5].

A sustainable and power aware computing system needs to provide services with a tradeoff between performance and energy consumption: Resource management should not be solely focused on performance but also equally take energy efficiency into account. Resource allocation in cloud is already a challenging problem due to the need to address response time, job completion time constraints and system heterogeneity. The problem becomes more challenging when power management is an additional design objective because power consumption of the system must be carefully balanced against other performance measures [6].

Many commercial realizations of computing clouds are already available today (e.g., Amazon, Google, IBM, Yahoo, etc.) [9].Many efforts have been made to improve the energy efficiency in IaaS Cloud. Generally, there are

two primary ways of reducing the processor power consumption are ***shutdown*** *(On/Off servers)* and ***slowdown***. We only discuss the slowdown option through DVFS and is known to reduce the dynamic power consumption at the cost of increased execution time for any computation task. Energy management can be achieved by regulating the instantaneous power consumption by controlling Dynamic Voltage and Frequency Scaling (DVFS). DVFS exploits the convex relation between the CPU supply voltage and power consumption. This is a practical approach because the performance under DVFS can be computed as the execution time at a nominal clock frequency multiplied by a scaling factor.

The Quality of Service (QoS) plays a crucial role in attracting and retaining customers, and has a direct impact on IaaS providers' revenues. QoS requirements are usually ruled by Service Level Agreements (SLAs), which are contracts negotiated between service providers and their customers to determine costs and penalties based on the achieved performance level. However, it is difficult to optimally allocate resources to different services due to high variability of Internet workloads and multi-service contention for limited resources. In many cases, traffic load received by the data center is not only irregular but also very sensitive to external environmental conditions. Because of those traffic characteristics and the imperious need of meeting the SLA, most data centers over-provision computing capacity to ensure providing an acceptable service even to the detriment of the facility's energy efficiency. This practice has a negative impact on energy usage, profitability, and green environment.

In this paper we model energy efficient resource management with DVFS mechanism and improving the services placement, resource allocation, and load balancing, as a commercial-based optimization problem whose aim is to maximize the total profit for IaaS cloud provider. They host applications in virtualized Physical resources (e.g., CPU, disks, communication network) and are partitioned into multiple isolated VMs each running at a fraction of the physical system capacity. In this paper we describe unified techniques, implemented by architecture controllers which can establish the set of services executed by each server (i.e., service placement problem), the request volumes at various servers (i.e., load balancing problem), and the resource capacity allocated for the execution of each service at different servers (i.e., dynamic resource allocation problem). Architecture controllers can also decide to scale the voltage or frequency of each server depending on its load (i.e., frequency scaling problem).

These four problems have clear mutual interrelations. Anyone of them changes will lead to different policies of other three problems. Despite these interrelations, previous literature has considered each decision problem in isolation.

The main contribution of this paper is to integrate all the previous problems in a unifying framework and provide efficient and robust solutions. The optimization objective is to maximize the total profit which is comprised of three parts, i.e. the revenue from providing services, the cost from energy consumption and the revenue loss caused by VM migration or start/stop (i.e. service placement change). Moreover, traditional resource management approaches focused on the optimization of performance measured by average response time and resource utilization. In contrast, this paper's optimization model considers the tail distribution of request response times, thus providing guarantee on the response time observed by each request. The effectiveness of the solution, compared with existing techniques, is evaluated through simulation and results show that our solution can yield significant profit gain for the IaaS cloud provider when compared to alternative technique.

The rest of this paper is organized as follows. Section 2 reviews the related existing works. The IaaS resource management approach and the system model are discussed in Section 3. The problem formulation is presented in Section 4 while Section 5 highlights the optimization problem. Section 6 presents the simulation results and Section 7 concludes the paper work.

2. Related Work

Here we review prior research problems, relevant to our work. There has been significant attention by the research community to the topic of effective QoS management as well as reduction of energy usage in IaaS Cloud environment. A lot of work has been done to achieve power reduction at the device level, e.g., in mobile systems to extend battery life. Due to availability of advanced hardware, nowadays low power techniques and energy savings mechanisms are progressively introduced in server environments.

From the literature, five main problems have been considered to manage system management policies: (1) admission control, (2) resource allocation, (3) load balance, (4) service placement and (5) energy consumption optimization. Moreover, to manage these problems, existing solutions and applying techniques fall into three main categories: (1) the control-theory based feedback loop, (2) machine learning techniques and (3) utility-based optimization techniques. We summarize the main results in each category and emphasize existing utility-based optimization solutions, which are closely

related to our approach. In [7] the descriptions of all these approaches, including [8] [10] 11] [12] [13] [14] [15] [16] [17] have discussed in detail.

Other study is focused on service differentiation in server farms by physically partitioning the farm into separate clusters, each serving one of the request classes, e.g., [18]. New technology trends advocate the sharing of physical resources among multiple request classes [19] supported also by virtualization and server consolidation technologies. In [20] a virtualization framework able to support heterogeneous workloads (batch and web interactive system) is presented, but energy saving policy is not considered. Authors in [21] have presented a multi-layer and multi-time scale solution for the management of virtualized systems, but they do not consider the trade-off between performance and system costs. The work in [7] have suffered the problem of admission control and capacity allocation in multi-tier virtualized environment, however the solutions are not integrated and only homogeneous systems with a single physical host per tier are considered. Finally, in [22], a framework able to coordinate independent resource managers running on distributed physical nodes of a virtualized data center has been presented. Note that, autonomic management today is implemented also in virtualization products, e.g. VMware DRS or Virtuoso VSched [20], with the aim at equalizing the use of resources instead of determining performance and energy trade-off. Considering the energy consumption optimization, authors in [23] have presented the result of a server consolidation project on blade servers based on a power budget mechanism. Processors frequencies are throttled in order to achieved CPU utilization and energy savings goals, but the approach cannot provide any SLA guarantee a priori. Similarly, the work in [4] have provided power budget policies for virtualized environments and also proposed an accurate model to predict server system average power consumption. The work in [24] integrates utility-based and control oriented techniques for the management of energy in hosting centers. Nevertheless, only homogeneous servers, working at the same operating frequency, dedicated to a single application are considered.

3. IaaS Cloud Resource Management Model

3.1 Virtualized Targeted Environment

The concept of virtualization was introduced a few decades ago, but it was not widely used because it required high performance hardware. However, as hardware performance has improved, virtualization has emerged as a most promising solution for eliminating the computing wastage through physical resource sharing in enterprise computing server systems in general and data centers in particular. Moreover, virtualization makes system maintenance much easier, improves system availability, and reduces the cost of a large computer system. The actual performance is affected by the resource capacity available to the service, and the time varying demands of the workload due to changes of workload intensity. Our goal is to meet the application-level performance targets for the multiple services hosted in the virtualized environment through dynamic resource allocation, while avoiding the over-provisioning the total amount of resources to the applications. When a request arrives at system model, it goes to a free server if there is any available; otherwise the incoming request is rejected by admission control module.

The target environment of this work is shown in figure 1. Requests arrive first at the Broker, where they are classified according to their URI pattern as belonging to one of several configured flows. A dispatched request passes through the Admission Control and Load Balancer which selects the instance where the request should be sent. Local controller is responsible for monitoring and executing the commands which are directed by Global controller. The Global Controller receives as input (i) the utility functions from every local controller and (ii) system-level performance metrics (e.g. CPU load) from virtual and physical servers (iii) Incoming workload and workload history. The output of the Global Controller consists of management actions directed to the server hypervisor and notifications sent to Local Controller. The latter notifies the Local Controller that (i) a new VM with specific resource capacity has been allocated to the service, (ii) an existing VM has been upgraded or downgraded, i.e.

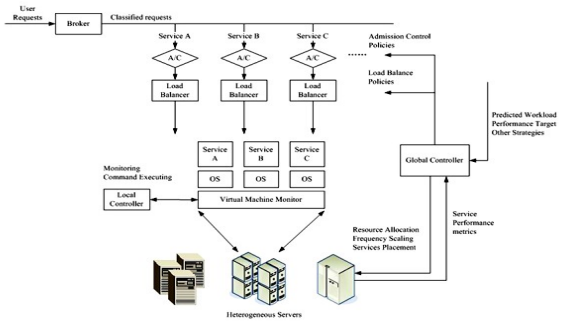

Fig. 1 IaaS Cloud Resource Management Model

its resource capacity has been changed and (iii) a VM belonging to the service is being preempted and that the

application should relinquish it promptly. Management actions include the life cycle management of VM (starting/stopping) and the trigger of a live migration of a running VM, the latter operation being transparent as far as the hosted services are concerned. At the same time, the policies for load balancing and admission controlling are also sent to the specific modules, i.e. A/C and Load Balancer in figure.

This paper considers the design of four interrelated problems of IaaS resource management including resource allocation, load balance, service placement and energy consumption (frequency scaling in this work). We assume that the incoming workload, admission control and service demands are obtained by the monitoring infrastructure with prediction ability. Our Global controller combines a performance model and an optimization model, and has the goal of maximizing revenues from SLAs, while minimizing the total cost associated with the usage of resources and power etc. In order to achieve these goals, Global controller can choose the dynamically adjust the fraction of capacity assigned to each VM and/or to limit the incoming workload by serving only the subset of requests that maximizes profits, i.e., revenues minus costs. Resource allocation is performed periodically, every 5 to 10 min, depending on workload profiles. At the end of each period, on the basis of a prediction of future workloads, the performance model generates an estimate of the future performance of each VM, i.e., determines future SLA violations for each service. The optimization model uses these estimates to determine the fraction of capacity to be assigned to each VM (i.e., resource allocation) as well as the number of requests effectively served for each service from service set (i.e., admission control) that maximizes profits.

3.2 System Model

The under discussion IaaS cloud model consisting of variant computing resources and decision making functions, which are explain in the following section.

Symbol	Description
System Parameters	
N	The set of heterogeneous servers
S	The set of services
$C_j^{\max}$	Maximal capacity of server j when it is working at the maximal frequency Remark: CPU cycles per second, or MIPS
$P_j^{\max}$	Maximal dynamic power consumption of server j when it is working at the maximal frequency
F_j	Set of discrete feasible frequency of server j.
β_i	Maximal resource utilization of service i, $0 < \beta_i < 1$
D_i	Average resource demand of service i Remark: desired number of CPU cycles or MIPS
T	Control time horizon
I	The placement matrix. $I_{i,j} = 1$ If service i is running on server j; $I_{i,j} = 0$ otherwise.
SLA Parameters	
c	Unit charge for energy consumption
v	Average cost associated with starting or stopping VMs on a server
w_i	Revenue of each request of service i if the response time is lower than R_i^{SLA}
R_i^{SLA}	Response time threshold guaranteed for requests of service i
$R_{i,j}$	Average response time of executing the request of service i on server j
Input from workload predictor, monitor and admission control	
$\lambda = \{\lambda_i\}$	λ_i is the request arrival rate of service i
Decision Variables	
$L = \{L_{i,j}\}$	$L_{i,j}$ is the fraction of capacity of server j devoted for executing service i
$W = \{\lambda_{i,j}\}$	$\lambda_{i,j}$ is the workload of service i assigned to server j
$\Omega = \{\Omega_j\}$	Ω_j is the Factor of frequency adjusting ratio for server j

Physical servers: The model includes n set of N heterogeneous physical servers, which are denoted as $N = \{N_1, N_2, \cdots, N_n\}$, each of which runs a Virtual Machine Monitor (VMM) such as VMware or Xen [4]. It is supposed that each server has individual maximal dynamic power consumption $P_j^{\max}$ and as well as maximal resource capacity is $C_j^{\max}$. Each server's physical resources (i.e., CPU, disks, communication network) are partitioned among multiple virtual machines, each running a service. Virtualization enables the flexible reduction and expansion of the resource capacity assigned to each VM. As a first approximation, we assume that, once a capacity assignment has been performed then each VM is guaranteed its resources, regardless of the actual load of other VMs. In other words, we assume that virtualization provides performance isolation and facilitates service differentiation, preventing the contention for resources among services. In this work, we only consider one type of resource which is CPU, because it is easy to be extended to other resources. In the system, each server has a single CPU which supports DVFS by varying both its supply voltage and operating frequency from a limited set of values. The adoption of DVFS is very promising since it does not introduce any system overhead, while hibernating and restoring a server require time and energy.

The frequency of server can be adjusted to arbitrary values from the set of values: $F_j = \{f_{j,1}, f_{j,2}, \cdots f_{j,k}\}$ where symbol $f_{j,k}$ is the feasible value of frequency which depends on the DVFS module and it is discrete. Each server has a PS scheduler .We define a variable Ω_j to represent the frequency level which is computed from current frequency divided by maximal frequency. We have $C_j = C_j^{\max} \cdot \Omega_i$ where C_j represent the current resource capacity.

Service: There are m *set* of request services S, which are denoted by $S = \{S_1, S_2, \cdots, S_m\}$. Each service $S_i \in S$ request can be served by a set of server applications according to the client/server paradigm. For simplicity we assume that each service S_i request is associated with a single customer. Each service S_i can be characterized by 1) average resource demand D_i ; 2) maximal resource utilization β_i .We use D_i to indicate the average service demand in a virtualized server when a set of service i receives the whole physical capacity of the server and can be evaluated by continuously monitoring the system. We also assume the service-requests scheduling policy running on each server is modeled as an M/M/1 open queue with FCFS scheduling.

SLA and utility function: Recently, the problem of maximization of SLA's revenues in IaaS Cloud has attracted vast attention by the research community. In our model, the SLA is defined by considering only service response times. For each set of service, a step-wise utility function is defined to specify per request revenue (or penalty) incurred when the corresponding average response times assumes a given value. Sometime customers are willing to pay a higher rate per request with lower response times. The technique proposed in this paper expresses resource requirement in the form of a utility function which encodes application satisfaction from a specific allocation. The utility function is calculated based on observed workload intensity, profile of resource consumption, and performance objective. The optimization objective is to maximize the minimum utility value among all services. Our objective is to maximize the total profit based on optimal strategies of resource allocation, workload assignment, frequency adjustment, services placement.

3.3 The Performance Model

The observation of the application performance as function of utilization levels provides us insights for definition of performance management problems. In this section we present an analytical queuing model that helps to predict the performance metrics of each VM in this system. This model is a base of the optimization solution to solve the dynamic resource management problem.

Our result deals the demand to guarantee the response time observed by each individual request from set of services, thus setting guarantees on the response time tail distribution. In other words, the motto of this performance model is to estimate the probability that a class i request goes through a response time that violates the SLA contract of the corresponding service $P\left(R_{i,j} \geq R_i^{SLA}\right)$.

We suppose in a model, each VM coupling with a service as a single queue. This assumption provides a proof of concept, allowing us to evaluate the overall applicability and effectiveness of our framework. Moreover, given the assumptions of Poisson request arrival rates and exponentially distributed service time, furthermore in order to estimate the probability of response time violations for service i requests, we need the probability distribution of user service i response times. The arriving rate of request is $\lambda_{i,j}$ (i.e. workload), then the expected average response time $E\left(R_{i,j}\right)$:

$$E\left(R_{i,j}\right) = \frac{1}{\mu - \lambda} = \frac{1}{\frac{\beta_i C_j^{\max} \Omega_j L_{i,j}}{D_i} - \lambda_{i,j}} = \frac{D_i}{\beta_i C_j^{\max} \Omega_j L_{i,j} - \lambda_{i,j} D_i} \quad (1)$$

According to the Markov inequality, the probability can be approximately estimated as:

$$P\left(R_{i,j} \geq R_i^{SLA}\right) \leq \frac{R_{i,j}}{R_i^{SLA}}$$

$$\Rightarrow P\left(R_{i,j} \geq R_i^{SLA}\right) \approx \min\left(1, \frac{1}{R_i^{SLA}} \cdot \frac{D_i}{\beta_i C_j^{\max} \Omega_j L_{i,j} - \lambda_{i,j} D_i}\right) \quad (2)$$

3.4 The Power Model

Power consumption by computing nodes in data centers is mostly determined by the CPU, memory, disk storage and network interfaces. In comparison to other system resources, the CPU consumes the main part of energy. We are managing CPU power and efficient usage. The main power consumption in CMOS circuits is composed of dynamic and static power. We only consider the dynamic power dissipation because it is more dominating factor in the total power consumption [5]. IaaS Cloud provider can increase their profit by reducing dynamic power consumption. The dynamic energy consumption by an application is proportional to supply voltage and the frequency [6]. Recent studies [21][25][26] have shown that the application of DVFS on the CPU results is almost linear power-to-frequency relationship for a server. The

reason lies in the limited number of states that can be set to the frequency and voltage of the CPU and in fact DVFS is not applicable on other system components. The CPUs can run at a finite number of operating frequencies in an interval [Fmin, Fmax] with associated power consumption [Pmin, Pmax]. So in this paper the dynamic power consumption of processor is given by $P_j = P_j^{\max} \cdot \Omega_j^3$ where P_j represent the server j. This allows a tradeoff between performance and power costs.

4. Problem Formulation

In this paper our objective is to search the optimal strategies of resource allocation, workload assignment, frequency adjustment, services placement to maximize the total profit. Currently, resource allocation in a Cloud data center aims to provide high performance while meeting SLAs, without focusing to minimize energy consumption. For the completion of both performance and energy efficiency we consider three crucial factors that affect the final profit of service providers which are given below:

Firstly, we consider the revenue from providing services. In our model, the SLA is defined by considering the response time of each service invocation. If the response time of a service invocation is above a given threshold R_i^{SLA}, then SLA is violated and there is no revenue can be obtained from users. Vice versa, if the response time is lower than R_i^{SLA}, the user will pay w_i to the service provider.

Secondly, we also consider the cost of energy consumption of servers and the energy related cost is computed using the full system power models adopted in [21],[19], which allow estimating servers energy consumption. For this purpose we denote c as the unit charge of energy consumption of server. This cost is control by applying the DVFS set of values F_j on servers.

Thirdly, we calculate the cost of service assignment on servers via VMs. This calculation includes the power spent by system on starting a new virtual machine or stop existing virtual machine, and, possibly, the revenue lost when the computer is unavailable to perform any useful service for this duration. It considers the performance effect during that time. Finally we limit the number of VMs movements in the system we add this movement cost v which avoids system instability.

$$(\text{P1}) = \max \sum_{i=1}^{m}\sum_{j=1}^{n}\left(1-\min\left(1,\frac{1}{R_i^{SLA}}\cdot\frac{D_i}{\beta_i C_j^{\max}\Omega_j L_{i,j}-\lambda_{i,j}D_i}\right)\right)\lambda_{i,j}w_i T$$
$$-cT\sum_{j=1}^{n}P_j^{\max}\Omega_j^3 - v\sum_{i=1}^{m}\sum_{j=1}^{n}\left|I_{i,j}-I_{i,j}^*\right|$$

Subject to:

$$I_{i,j}=1 \text{ or } I_{i,j}=0\ ,\forall i,\forall j \quad (3)$$

$$1\geq L_{i,j}\geq 0\ ,\forall i,\forall j \quad (4)$$

$$L_{i,j}=0 \Leftrightarrow I_{i,j}=0, L_{i,j}\neq 0 \Leftrightarrow I_{i,j}=1\ ,\forall i,\forall j \quad (5)$$

$$\sum_{i=1}^{m} L_{i,j}\leq 1\ ,\forall j \quad (6)$$

$$\sum_{j=1}^{n}\lambda_{i,j}=\lambda_i \quad (7)$$

$$0\leq \lambda_{i,j}\leq \lambda_i I_{i,j}\ \ \forall i,\forall j \quad (8)$$

$$I_{i,j}=0 \Rightarrow \lambda_{i,j}=0 \qquad \forall i,\forall j \quad (9)$$

$$\Omega_j \in F_j\ ,\forall j \quad (10)$$

$$\frac{1}{R_m^{SLA}}\cdot\frac{D_m}{\beta_m C_n^{\max}\Omega_n L_{m,n}-\lambda_{m,n}D_m}\leq 1 \quad (11)$$

$$i\in[1,m], j\in[1,n] \quad (12)$$

Constraints family (3) entails express that the set of services executed by each server, introducing the binary variable $I_{i,j}$ equal to 1 if service i is assigned to server j, 0 otherwise. Constraints family (4) express that the capacity usage (or ratio) of server j allocated to service i Constraints family (5) means that if the $I_{i,j}$=0, then there is no any instance of service i on server j, so the capacity is 0 and other condition is vice versa. Constraints family (6) means that the overall allocating resource capacity cannot exceed the total resource capacity, in other words guarantee that resources are not saturated. Constraints family (7) entails that the sum of workload of service i on each server is equal to the total workload of service i. Constraints family (8) means that the workload of service i on a dedicated server j is between 0 and the total workload. Constraints family (9) express that if there is no service assigned to server j, we cannot allocate workload to that target server. Constraints Family (10) entails that the frequency is selected in the pre-defined frequency set F_j. Finally, Constraints (11) express that the probability must be lower than 1 and we have indicated that if not, then there is no optimal solution.

The constraint (11) is reasonable because in any optimal solution, the value of probability must be in the interval [0, 1]. Therefore the objective function can be restated as:

$$(\text{P1}) = \max \sum_{i=1}^{m}\sum_{j=1}^{n}\left(1-\frac{1}{R_i^{SLA}}\cdot\frac{D_i}{\beta_i C_j^{\max}\Omega_j L_{i,j}-\lambda_{i,j}D_i}\right)\lambda_{i,j}w_i T$$
$$-cT\sum_{j=1}^{n}P_j^{\max}\Omega_j^3 - v\sum_{i=1}^{m}\sum_{j=1}^{n}\left|I_{i,j}-I_{i,j}^*\right|$$

5. Optimization Solutions

5.1 Overall Solution

Here we discuss the optimization and the corresponding solution procedure for the problems mentioned above. The objective is to maximize the difference between revenues from SLA contracts and the costs associated with server's energy consumption and assigning VMs to servers for the next control time horizon T. The problem (P1) is difficult to solve because if we fix the problems such as service placement and the frequency adjustment of each server in the system, the sub-problem of load balancing and resource allocation is still difficult. This is due to the objective function is neither concave nor convex. Due to this reason the procedure of computing is complex, so we cancel the detailed proof. Commercial nonlinear optimization tools can solve such type of problems only for small size instances. For realistic problems of a reasonable size, we have considered a solution combining a decomposition approach with an optimization loop.

SLA-EERM Algorithm
Input: a) a set of heterogeneous servers N and their current frequencies Ω^*; b) a set of services S and their current capacity L^*; c) request arrival rate of each service λ.
Output: a) A new solution includes resource allocation strategy L, load balance strategy W, frequency adjustment strategy Ω; b) final profit value f

0. Build an initial solution $\Theta' = (L, W, \Omega)$ by algorithm *Initialize* (L^*, Ω^*, λ) and compute the profit f'.
1. While true
2. $\Theta = \Theta'$, $f = f'$.
3. Apply algorithm *Frequency_and_placement* (Θ, I^*, f) to get a new solution $\Theta' = (L', W', \Omega')$ and Compute the new profit f'.
4. If $f' == f$
5. Apply algorithm *Resource_and_workload* (Θ) to get optimal resource allocation and load Balancing strategies L'', W'' and update Θ', let $L' \leftarrow L'', W' \leftarrow W''$ and compute the new profit f'.
6. If $f' == f$, break;

As we discussed before, the model (P1) consists of three parts: the first part represents the overall services revenue depending on the strategies of resource allocation and load balance; the second part represents the cost of energy consumed by servers depending on the strategy of frequency adjustment; the third part is the cost from VMs migration depending on the services placement change. Our solutions comprise in two groups and these groups are further divided in four sub-problems. The first one includes frequency adjustment and service placement change, and the other includes resource allocation and load balance.

The overall solution of problems is mentioned in SLA-EERM algorithm and comprises in two main phases: (1) an initial feasible solution is obtained from current placement of services using a greedy heuristic, (2) and obtained solution is repeatedly and incrementally optimized by a loop exploring two neighborhoods mentioned above. If the optimization procedure can find a new solution which achieves the most utility, we will replace the old solution with new one and continue; otherwise, the procedure is over and a final optimum solution is obtained.

5.2 Initial Solution

We consider an intuitive way to generate an initial solution for our model with basic rules which are given below:

1. Assign incoming workload λ_i to all existing instances of each service according to previous allocating resource capacity $\overline{L_{i,j}}$ and frequency $\overline{\Omega_j}$, the basic idea is the workload of each instance $\lambda_{i,j}$ must lower than a threshold of resource utilization $\alpha\beta_i C_j^{\max} \overline{\Omega_j} \overline{L_{i,j}}$ (This method is adopted in many research work). Loop this operation until there is no capacity for workload dispatching or all the workloads have been dispatched over.
2. Collect all the remaining workload and compute the remaining capacity. Sort the services according to non-decreasing value of remaining workload and sort the servers according to non-deceasing value of frequency firstly and non-increasing value of remaining capacity secondly.
3. Dispatch the remaining workload as follow. Select one service at the beginning of the queue, and select the first server, compute that if the energy and migration cost of starting new instance on this server for dispatch the workload is higher than the energy cost of dispatching the workload to one of the existing instance which lead to minimal energy cost, then we increase the existing instance's capacity by frequency adjusting and dispatch the workload, otherwise, start a new instance on the selected server. Refresh both of the queues. Repeat this operation until all the workloads have been dispatched over.

$$\Delta(N_j,\lambda_i)=\begin{cases}E(\Omega_j')-E(\Omega_j) & \text{If there exists instance of service i on server j}\\ E(\Omega_j')-E(\Omega_j)+v & \text{Otherwise}\end{cases} \quad (13)$$

Here, $E(\Omega_j)=cTP_j^{\max}\Omega_j^3$

Moreover, we can easily find that in first case $\Delta(N_j,\lambda_i)>0$ and it is easy to a proof. In second case, the cost $\Delta(N_j,\lambda_i)>v$ and the increasing cost of energy maybe zero.

5.3 The Frequency Adjustment and Service Placement Problems

In this section, we introduce the solution of frequency adjustment and service placement problem which is based on the current solution. We keep the total capacity of each service invariable and the rule of workload assignment is that mentioned in 5.2. Then, the first part in objective function is fixed. Therefore the optimization function can be changed to

(P2) $\min\ cT\sum_{j=1}^{n}P_j^{\max}\Omega_j^3+v\sum_{i=1}^{m}\sum_{j=1}^{n}\left|I_{i,j}-I_{i,j}^*\right|$

Subject to (3), (5), (10), (12)

Such optimization problem is similar to the problem solved by authors in [19]. In this part, we follow this algorithm. The assigning the workload is proportional to the capacity. The conditions of workload shifting and service re-allocation also follow equation (13). The detailed algorithm can be found in the specific material.

5.4 The Resource Allocation and Load Balancing Problems

If we fix the value of Ω_j and the service placement $I_{i,j}$, the problem becomes resource allocation and load balance;

(P3)= $\max\ \sum_{i=1}^{m}\sum_{j=1}^{n}\left(1-\frac{1}{R_i^{SLA}}\cdot\frac{D_i}{\beta_i C_j^{\max}\Omega_j L_{i,j}-\lambda_{i,j}D_i}\right)\lambda_{i,j}w_iT$

Subject to (4), (5), (6), (7), (8), (9), (11), (12)

We apply fixed point iteration (FPI) techniques to solve this problem. The procedure is given in figure 4. In the first step, the value of resource allocation variables $L_{i,j}$ is fixed and the optimal workload dispatching variables $\lambda_{i,j}$ are identified. In the next step, the values of $\lambda_{i,j}$ are held fixed and the optimal resource allocation variables $L_{i,j}$ are determined. The algorithm stops when the difference between subsequent values of objective function is lower than a given threshold ζ. We apply Karush–Kuhn–Tucker (KKT) conditions to optimally solve the sub-problems separately.

Initialize Algorithm
Input: a) a set of heterogeneous servers N and their current frequencies Ω^*; b) a set of services S and their current capacity L^*; c) request arrival rate of each service λ.
Output: a) an initial solution includes resource allocation strategy L, load balance strategy W, frequency adjustment strategy Ω.

1. Initialize $L\leftarrow L^*, W\leftarrow 0, \Omega\leftarrow\Omega^*$
2. For $i=1$ to n do
3. For $j=1$ to m do
4. Compute $\lambda_{i,j}=\beta_i C_j^{\max}\overline{\Omega_j L_{i,j}}$; if $0<\lambda_{i,j}<\lambda_i$, assign Workload $\lambda_{i,j}$ of service m to Server n, $\lambda_i=\lambda_i-\lambda_{i,j}$; else if $\lambda_{i,j}\geq\lambda_i$ assign workload λ_i, $\lambda_i=0$, break.
5. If all of λ_i is equal to zero, return.
6. Sort the servers according to non-deceasing value of their frequencies firstly and non-increasing value of remaining capacity secondly, denoted by N'.
7. Sort the services in non-decreasing value of their λ_i, except whose $\lambda_i=0$, denoted by S'.
8. While $S'\neq null$
9. Select the first service S_1 and the first server N_1'.
10. Select the server N_{S_1} which hosts instance of service S_1 meanwhile has minimal index in N'.
11. If $N_{S_1}==N_1'$, assign the remaining workload of S_1 and update Ω, W and L;
12. Else if $\Delta(N_1',\lambda_1)<\Delta(N_{S_1},\lambda_1)$, increase the frequency of N_1' and start a new instance of S_1 on N_1' and Assign remaining workload to it, update Ω, W and L;
13. Else assign the remaining workload λ_i to N_{S_1}, update Ω, W and L.
14. Remove S_1 from S' and re-sort the sequence of servers follow step 6).

5.4.1 Load balance sub-problems

If we fix the allocated fraction of resource capacity $\overline{L_{i,j}}$, then the problem become how to assign the workload for achieving the maximal revenue. It suffices to solve n load balance sub-problems independently. Each of the sub-problems can be given as follow:

$$(\text{P4}) = \max f\left(\lambda_{i,j}\right) = \sum_{j=1}^{n}\left(1 - \frac{1}{R_i^{SLA}} \cdot \frac{D_i}{K_{i,j} - \lambda_{i,j} D_i}\right)\lambda_{i,j} w_i T$$

Subject to (7), (8), (9), (12)

Additionally, according to constraint (11), we have constraint (14)

$$\lambda_{i,j} < \frac{K_{i,j}}{D_i} - \frac{1}{R_i^{SLA}} \tag{14}$$

Where $K_{i,j} = \beta_i C_j^{\max} \overline{\Omega_j}\,\overline{L_{i,j}}$ represents the total allocated resource of service i at server j (Cycles). Because it is obvious that $\lambda_{i,j}$ is independent, therefore we can easily calculate the Hessian $Diag\left(\frac{\partial^2 f}{\partial^2 \lambda_{i,j}}\right)$,

$$\frac{\partial^2 f}{\partial^2 \lambda_{i,j}} = -\frac{w_i T D_i^2 K_{i,j}}{R_i^{SLA}\left(K_{i,j} - \lambda_{i,j} D_i\right)^3} < 0$$

So, *it is concave.*

Resource_and_workload Algorithm
Input: a) solution $S = (L, W, \Omega)$ from *Frequency_and_placement* algorithm; c) request arrival rate of each service λ.
Output: a) optimal solutions of resource allocation L and load balance W; b) final profit value f

1. Initialize $\Delta = 0$;
2. While $\Delta > \zeta$ do
3. Fix $\overline{L_{i,j}}$ and search the optimal $\lambda_{i,j}$, update W, compute the profit f_1.
4. Fix $\overline{\lambda_{i,j}}$ and search the optimal $L_{i,j}$, update L, compute the profit f_2.
5. $\Delta = |f_1 - f_2|$

5.4.2 Resource allocation sub-problem

If we fix the workload $\overline{\lambda_{i,j}}$, then the problem become how to allocate the resource for achieving the maximal revenue, which is as follow:

$$(\text{P5}) = \max f(\lambda) = \sum_{i=1}^{m}\left(1 - \frac{1}{R_i^{SLA}} \cdot \frac{D_i}{\Psi_{i,j} L_{i,j} - \overline{\lambda_{i,j}} D_i}\right)\overline{\lambda_{i,j}} w_i T$$

Subject to (4), (5), (6), (12)

Furthermore, according to constraint (11), we have constraint (15)

$$L_{i,j} > \frac{\overline{\lambda_{i,j}} D_i}{\Psi_{i,j}} + \frac{D_i}{\Psi_{i,j} R_i^{SLA}} \tag{15}$$

Where $\Psi_{i,j} = \beta_i C_j^{\max} \overline{\Omega_j}$

Because it is obvious that $L_{i,j}$ is independent, therefore we can easily calculate the Hessian $Diag\left(\frac{\partial^2 f}{\partial^2 L_{i,j}}\right)$

$$\frac{\partial^2 f}{\partial^2 L_{i,j}} = -\frac{\overline{\lambda_{i,j}} w_i D_i T \Psi_{i,j}^2}{R_i^{SLA}\left(\Psi_{i,j} L_{i,j} - \overline{\lambda_{i,j}} D_i\right)^3} < 0$$

So, *it is concave.*

6. Simulation Work

To evaluate the approach, we build an experimental environment that simulates a data center with three groups of services (S1,S2,and S3) and 20 servers. Each Service runs multiple transactions/jobs. The services were simulated using CSIM and each simulated server has one CPU and one disk. While the data center and all its nodes were simulated in one machine, a different machine executed the controller code, which runs a search algorithm and uses queuing network analytic models to decide the best allocation of servers to services. The controller communicates with the machine that simulates the data center through Windows named pipes. A local controller in each service collects its basic measurements parameters (number of allocated servers, arrival rates, frequency adjustment) and sends them to the global controller. Transactions for each group of the services are generated by separate workload generators with their own arrival rates. In this experiment the SLA's for group S1, S2 and S3 is 60 ms , 75 ms and 90 ms respectively. These three groups of services have a different SLA and service demand due to variability of workload. In this set of experiments we considered that the switching cost is zero, i.e., servers are moved from one group of services to another instantaneously. This assumption is based on the fact that all applications are installed in all servers and that the switching cost amounts to launching an application, which is in the order of a few seconds.

During the experiments, the transaction average arrival rates at the three services S1, S2, S3 was varied as shown in Fig.2. The x axis is the Time Horizon, set at hours. For S1, S2, and S3 the arrival rates have three peaks and valleys, at different time interval.

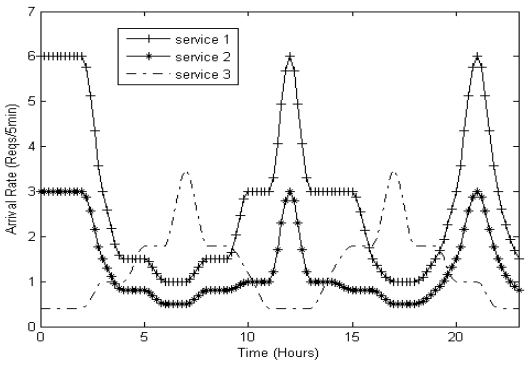

Fig. 2 Variation of Average Arrival Rate of three services (S1,S2,and S3)

It is interesting to analyze how the response times vary as the experiment progresses and as the number of servers allocated to the services. Figure 3 shows the variation of the average response time for transactions of services S1, S2, and S3 in the two solutions. The frequency level of the running servers is thus modulated in order to have a high utilization of the physical servers while reducing the energy consumption. The DVFS leads to an increase in revenues.

Our solution uses the resources more efficiently throughout of the day; it provides better response times while adopting a lower number of servers. Overall, during the 20 hours, our solution gives net revenues higher than the alternative one which is never profitable with respect to objective functions and the improvement which can be obtained by our solution is significant with respect to the alternative objective function: During the 20 hour our overall profit is $4.8 while the alternative solution incurs in penalty (see Fig 4).

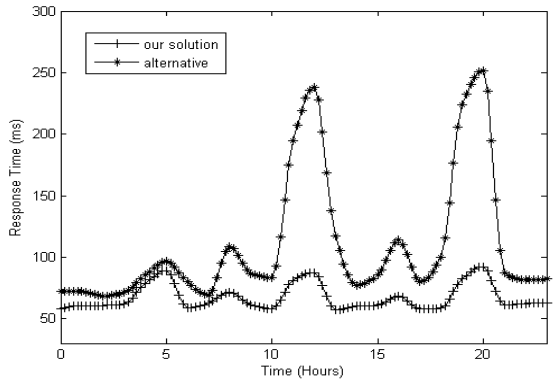

Fig. 3 Comparison of Variation of response time

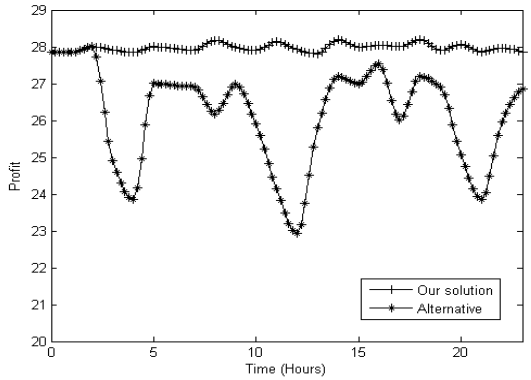

Fig. 4 Comparison of variation of Profit ($)

7. Conclusions

As research indicates that energy efficiency and QoS performance are opposite ends in the server related energy reduction. We have proposed energy efficient resource allocation model for IaaS cloud data center in which the servers are equipped with DVFS module, maximizing the profits coupled with SLAs. The cost model consists of utility functions which include revenues and penalties incurred depending on the achieved level of performance and the energy costs associated with the use of DVFS. The overall optimization problem involves the allocation of VMs to servers, servers operating frequencies, the load balancing and capacity allocation of VMs. The corresponding optimization problem is NP-hard and a heuristic procedure has been proposed. A hybrid optimization solution is proposed to solve the problem and its effectiveness, compared to existing techniques, is evaluated using synthetic workloads. Results show that our solution can yield significant profit gain for the cloud provider when compared to alternative technique.

References

[1] R. Buyya, C. S. Yeo, S. Venugopa, “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities”, In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, September 2008, pp. 5-13.

[2] Q. Zhang, L. Cheng, R. Boutaba, “Cloud computing: state-of-the-art and research challenges”, Journal of Internet Service and Applications, vol. 1, no. 1, pp. 7-18, 2010.

[3] B. Alexander, "Web 2.0: A new wave of Innovation for Teaching and Learning?" Learning, vol. 41, no. 2, pp. 32-44, 2006.

[4] W. Feng, X. Feng, R. Ge, "Green Supercomputing Comes of Age", IT Professional, vol. 10, no. 1, 2008.
[5] B. Khargharia, S. Hariri, F. Szidarovszky, et al., "Autonomic Power & Performance Management for Large-Scale Data Centers", NSF Next Generation Software Program Workshop, in 21st IEEE (IPDPS), 2007.
[6] Y. Yu, V. K. Prasanna, "Power-Aware Resource Allocation for Independent Tasks in Heterogeneous Real-Time Systems", in 9th IEEE International Conference on Parallel and Distributed Systems, 2002.
[7] J. Almeida, V. Almeida, D. Ardagna, et al., "Joint admission control and resource allocation in virtualized servers", Journal of Parallel and Distributed Computing, vol. 70, no. 4, pp. 344-362, 2010.
[8] P. Padala, K. G. Shin, X. Zhu, et al., "Adaptive control of virtualized resources in utility computing environments", in Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, 2007.
[9] Boss, G., Malladi, P., Quan, D., Legregni, L., and Hall, H., "Cloud Computing", (HiPODS), (October, 2007).
[10] P. Padala, K. Y. Hou, K. G. Shin, et al., "Automated control of multiple virtualized resources", in Proceedings of the 4th ACM European conference on Computer systems, 2009.
[11] G. Tesauro, N. K. Jong, R. Das, "On the use of hybrid reinforcement learning for autonomic resource allocation", Cluster Computing, vol. 10, no. 3, pp. 287-299, 2007.
[12] X. Liu, J. Heo, L. Sha, et al., "Adaptive control of multi-tiered web applications using queuing predictor", in IEEE Network Operations and Management Symposium, pp. 106-114, 2006.
[13] R. Das, J. O. Kephart, C. Lefurgy, et al., "Autonomic multi-agent management of power and performance in data centers", in Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pp. 107-114, 2008.
[14] M. N. Bennani, D. A. Menasc´e, "Resource allocation for autonomic data centers using analytic performance models", in Proceedings of the 2nd international conference on Autonomic computing, 2005.
[15] V. Gupta, M. H. Balter, "Self-adaptive admission control policies for resource-sharing systems", in Proceedings of the 11th international joint conference on Measurement and modeling of computer systems, 2009.
[16] L. Zhang, D. Ardagna , "SLA based profit optimization in autonomic computing systems", in Proceedings of the 2nd international conference on Service oriented computing, pp. 173-182, 2004.
[17] J. Kephart, H. Chan, R. Das, et al., "Coordinating multiple autonomic managers to achieve specified power performance tradeoffs", in Proceedings of the 4th international conference on autonomic computing, 2007.
[18] B. Urgaonkar, G. Pacifici, P. J. Shenoy, et al., "Analytic modeling of multitier Internet applications", ACM Trans. on Web, vol. 1, no.1, 2007.
[19] C. Tang, M. Steinder, M. Spreitzer, et al., "A scalable application placement controller for enterprise data centers", in Proceedings of the 16th international conference on World Wide Web, 2007.
[20] M. Steinder, I. Whalley, D. Chess, et al., "Server virtualization in autonomic management of heterogeneous workloads", 10th IFIP/IEEE International Symposium on Integrated Network Management, pp. 139-148, 2007.
[21] X. Zhu, D. Young, B. J. Watson, et al., "1000 islands-an integrated approach to resource management for virtualized data centers", Cluster Computing, vol. 12, no.1, pp. 45-57, 2009.
[22] S. Kumar, V. Talwar, V. Kumar, et al., "vManage: loosely coupled platform and virtualization management in data centers", In Proceedings of the 6th international conference on Autonomic computing, 2009.
[23] P. Ranganathan, P. Leech, D. Irwin, et al., "Ensemble-level power management for dense blade servers", in Proceedings of the 33rd annual international symposium on Computer Architecture, 34(2), 2006
[24] Y. Chen, A. Das, W. Qin, et al., "Managing server energy and operational costs in hosting centers", in Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2005.
[25] S. U. Khan, I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids", IEEE Trans. on Parallel and Distributed Systems, vol. 20, no. 3, pp. 346-360, 2009.
[26] G. Wei, A. V. Vasilakos, Y. Zheng, et al., "A game-theoretic method of fair resource allocation for cloud computing services", The Journal of Supercomputing, vol. 54, no. 2, pp. 252-269, 2010.