SLA-based Optimization of Power and Migration Cost in Cloud Computing

Hadi Goudarzi, Mohammad Ghasemazar, and Massoud Pedram
University of Southern California
Department of Electrical Engineering - Systems
{*hgoudarz, ghasemaz, pedram*}*@usc.edu*

Abstract—**Cloud computing systems (or hosting datacenters) have attracted a lot of attention in recent years. Utility computing, reliable data storage, and infrastructure-independent computing are example applications of such systems. Electrical energy cost of a cloud computing system is a strong function of the consolidation and migration techniques used to assign incoming clients to existing servers. Moreover, each client typically has a service level agreement (SLA), which specifies constraints on performance and/or quality of service that it receives from the system. These constraints result in a basic trade-off between the total energy cost and client satisfaction in the system. In this paper, a resource allocation problem is considered that aims to minimize the total energy cost of cloud computing system while meeting the specified client-level SLAs in a probabilistic sense. The cloud computing system pays penalty for the percentage of a client's requests that do not meet a specified upper bound on their service time. An efficient heuristic algorithm based on convex optimization and dynamic programming is presented to solve the aforesaid resource allocation problem. Simulation results demonstrate the effectiveness of the proposed algorithm compared to previous work.**[1]

I. INTRODUCTION

Demand for computing power has been increasing due to the penetration of information technologies in our daily interactions with the world both at personal and community levels, encompassing business, commerce, education, manufacturing, and communication services. At the personal level, the wide scale presence of online banking, e-commerce, software as a service (SaaS), social networking, etc., produce workloads of great diversity and enormous scale. At the same time computing and information processing requirements of various public organizations and private corporations have been increasing rapidly. Examples include digital services and functions required by the various industrial sectors, ranging from manufacturing to finance and from transportation to housing. To cope with these trends, a scalable, efficient, and dependable information technology (IT) infrastructure comprised of servers, storage, networking gear, physical facilities, IT workforce, and so on are needed.

Virtualization technology makes the independence of applications and servers feasible. Nowadays, computing systems heavily rely on this technology. Virtualization technology provides a new way to improve the power efficiency of the datacenters i.e., (server) consolidation, which enables the assignment of multiple virtual machines (VMs) to a single physical server. Consequently, some of the servers can be turned off or put into sleep state, thereby, lowering power consumption of the cloud computing system. The technique works because modern servers tend to consume 50% or so of their peak power in idle (or very low utilization) state—this effect is known as the non-energy-proportionality of modern servers [1]. Consolidation involves power-performance tradeoffs. More precisely, if workloads are consolidated on servers, performance of the consolidated VMs may decrease because of reduction of available physical resources (CPU, memory, I/O bandwidth) although the overall power efficiency of the system improves because fewer servers are needed to service VMs and the active servers will be highly utilized.

Low utilization of servers in a datacenter is one of the biggest factors in low power efficiency of the datacenter. For example, the average utilization of servers in a Google datacenter was reported to be 30% [2]. This fact has motivated the design of energy-proportional servers [3] to minimize the overall power consumption. State-of-the-art commercial servers are, however, non-energy-proportional. It is thus prudent from an energy efficiency viewpoint to have as few servers as possible turned on with each active server being highly utilized. Hence, there is a strong justification for server consolidation in current datacenters.

Operational cost and admission control policy in the cloud computing system are affected by its power control and VM management policies. Power management techniques [4]-[6] control the average and/or peak power dissipation in datacenters in a distributed or centralized manner. VM management techniques [7]-[11] control the VM placement in physical servers as well as VM migration from a server to another one. In this paper, we focus on the SLA-based VM management to minimize the operational cost in a cloud computing system.

The IT infrastructure provided by the datacenter owners/operators must meet various SLAs established with the clients. The SLAs may be resource related (e.g., amount of computing power, memory/storage space, network bandwidth), performance related (e.g., service time or throughput), or even quality of service related (e.g., 24-7 availability, data security, percentage of dropped requests.)

Infrastructure providers often end up over provisioning their resources in order to meet the clients' SLAs. Such over provisioning may increase the operational cost of the datacenters in terms of their monthly electrical energy bill and carbon emission. Therefore, optimal provisioning of the resources is crucial in order to reduce the cost incurred on the datacenter operators as well as minimize the environmental

[1] This work is supported in part by a grant from National Science Foundation.

impact of datacenters. The problem of optimal resource provisioning is challenging due to the diversity present in the clients (applications) that are hosted as well as in the SLAs. For example: some applications may be compute-intensive while others may be memory intensive, some applications may run well together while others do not, etc. Note that in general, there are two types of applications in the datacenter: (i) service applications and (ii) batch applications [2]. Service applications tend to generate many requests with low processing needs whereas batch applications tend to small number of requests with large processing needs. Unlike the batch applications that are throughput-sensitive, service applications are typically response time-sensitive. In this work we focus on service applications in cloud computing systems.

A datacenter comprises of thousands to tens of thousands of server machines, working in tandem to provide services to the clients, see for example [2]. In such a large computing system, energy efficiency can be maximized through system-wide resource allocation and server consolidation—this is in spite of non-energy-proportional characteristics of current server machines. Clients in a cloud computing system are software applications that require processing, memory and communication resources in "on-demand capacity provisioning" or "lease model of the IT infrastructure" bases (cf. [12] and [13].) Our goal in this paper is to minimize the total cost of the cloud computing system under performance-related SLAs—in particular, upper bounds on the response times (service latencies) for serving clients' requests.

The paper outline is as follows. Related work is discussed in section II. In section III, cloud computing system model is presented. The optimization problem and the proposed algorithm is presented in section IV and V. Simulation results and conclusions are given in the sections VI and VII.

II. Related Work

Distributed resource management is one of the most challenging problems in the resource management field. This problem has attracted a lot of attention from the research community in the last few years. Below we provide a review of most relevant prior work.

Srikantaiah et al. [13] presented an energy-aware consolidation technique to decrease the total energy consumption of a cloud computing system. The authors empirically modeled the energy consumption of servers as a function of CPU and disk utilization rates. Next, they described a simple heuristic to consolidate the processing works in the cloud computing system. The simple heuristic does not capture the effect of SLA on VM resource provisioning and only for very small input size performance of the solution is shown.

A VM placement heuristic to maximize the number of serviced applications, minimize the migration cost, and balance the load in physical machines is presented in [7]. The main focus of this work is on the scalability of the problem but the problem of assigning VMs on physical servers in a case that all VMs can be placed and power and migration cost minimization is the objective is not investigated.

Power and migration cost aware application placement in virtualized systems is proposed in [9]. Authors present a power-aware VM placement controller in a system with heterogeneous server clusters and virtual machines. pMapper architecture and placement algorithms to solve the problem of minimizing power subject to a fixed performance requirement are investigated. The proposed solution is presented based on assumption of predetermined performance level for VMs which is not applicable for all different kind of SLA contracts.

Liu et al. [15] described a SLA-based profit optimization problem in electronic commerce hosting datacenters. A fixed set of servers are assumed to be active and application placement on the servers are done to maximize the total SLA profit. SLA in this work is modeled as a response time constraint and less than a portion (e.g. 2%) of request's response time can violate that constraint. This kind of SLA is used in [16] to model the optimization of workload distribution to manage the brown energy consumption.

In [17], Ardagna et al. proposed a solution for SLA-based VM placement to maximize the profit in the cloud computing system. The presented problem considers only soft SLA contracts in which client pays the cloud provider based on the average response time provided to its requests. This type of SLAs is adopted in various prior works such as [18]-[20], but we believe that it cannot capture the complexity of the existing SLA contracts. In reference [21], we worked on multi-tier resource allocation in datacenters with guaranteed SLAs.

In this paper, we investigate the SLA-based VM placement to minimize the total operational cost in the cloud computing system. Operational cost includes power and migration cost and the expected penalty of serving clients. A lower bound on the total operational cost is presented, and the effectiveness of the proposed algorithm is demonstrated by comparing with previous works' algorithms and lower bound value.

III. System Model

A SLA-aware resource allocation method for a cloud computing system is presented to minimize the total operational cost of the system. The structure of the datacenter, the VMC, as well as performance model and type of SLA used by the clients are explained in this section. To increase paper readability, Table I presents key symbols used throughout this paper along with their definitions.

A. Datacenter Configuration

In the following paragraphs, we describe the type of the datacenter that we have assumed as well as our observations and key assumptions about where the performance bottlenecks are in the system and how we can account for the energy cost associated with a client's VM running in a datacenter.

A datacenter comprises of a large number of potentially heterogeneous servers chosen from a set of known and well-characterized server types. In particular, servers of a given type are modeled by their processing capacity (C_*^p) and main memory size (C_*^m) as well as their operational expense (energy cost), which is proportional to their average power consumption. We assume that local (or networked) secondary storage (disc) is not a system bottleneck. Each server is identified by a unique id, denoted by index j.

The operational cost of the system includes a term related to the total energy cost (in dollars) of serving clients' request.

The energy cost is calculated as server power dissipation multiplied by duration of each decision epoch in seconds (T_e) and cost of energy consumption in US dollars (C_p). The power of a server is modeled as a constant power cost (P_*^0) plus another variable power cost, which is linearly related to the utilization of the server (with a slope of P_*^p). Note that the power cost of communication resources and cooling and air conditioning modules are amortized over the servers and communication/networking gear in datacenter, and are thus assumed to be relatively independent of the clients' workload. More precisely, these costs are not included in the equation for power cost of the datacenter.

Each client is identified by a unique identifier, represented by index *i*. Each client produces one or more VMs, which are executed on some servers in the datacenter. Each client has also established an SLA contract with the datacenter owner (cloud provider.)

TABLE I. NOTATION AND DEFINITIONS

Symbol	***Definition***
λ_i	Predicted average request rate of the i[th] client
R_i^c, f_i^c	Contract target response time and penalty values for each request in the SLA contract
h_i^c	Hard constraint on the possible percentage of violation of the response time constraint in the SLA contract
μ_{ij}	Average service rate of requests of the i[th] client on a unit processing capacity of j[th] server
m_i	Required memory for the i[th] client
C_j^p, C_j^m	Total processing and memory capacities of the j[th] server
$cost_i^m$	Migration cost of the i[th] client
P_j^0, P_j^p	Constant and dynamic (in terms of utilization) power consumption of the j[th] server operation.
T_e	Duration of the decision epoch in seconds
C_p	Cost of energy consumption
y_{ij}^p	Pseudo Boolean parameter to show that if the i[th] client is assigned to the j[th] server in previous epoch (1) or not (0)
x_j	A pseudo-Boolean integer variable to determine if the j[th] server is ON (1) or OF (0)
y_{ij}	Pseudo Boolean parameter to show that if the i[th] client is assigned to the j[th] server (1) or not (0)
α_{ij}	Portion of the i[th] client's requests served by the j[th] server
ϕ_{ij}	Portion of processing resources of the j[th] server that is allocated to the i[th] client

B. VM Management System

Datacenter management is responsible for admitting the VMs into the datacenter, servicing them to satisfy SLAs, and minimizing the operational cost of the datacenter. We consider two main resource managers in the datacenter: VM controller (VMC) and power manager (PM). An exemplary architecture for the datacenter management system with emphasis on the VMC and per server PM is depicted in Figure 1.

Power manager is responsible for minimizing the average power consumption and satisfying the peak power constraints (thermal or because of power distribution unit limitation) subject to providing the required performance to the VMs. Power management system in datacenter includes hierarchical power provisioners and a power manager for each server. Power provisioners distribute the peak power allowance between lower level power consumers and make sure that these power budget constraints are met. Servers are located at the lowest level of this hierarchy. Power manager in each

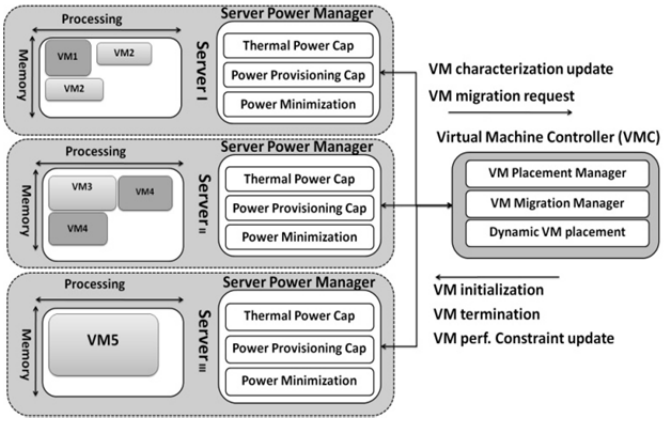

Figure 1. VM management structure in a datacenter

server tries to minimize the average power consumption subject to satisfying the peak power constraint and performance requirements of the assigned VMs. This manager uses different dynamic power management techniques such as dynamic voltage and frequency scaling (DVFS), clock throttling and etc. to minimize the power.

VMC is responsible for determining the performance requirements of the VMs and migrating them if needed. VMC performs these tasks based on two optimization procedures: semi-static and dynamic. The semi-static optimization procedure is performed periodically whereas the dynamic optimization procedure is performed when it is needed.

In the semi-static optimization procedure, VMC considers the whole active set of VMs, the previous assignment solution, feedbacks generated from power, thermal and performance sensors, and workload prediction to generate the best VM placement solution for the next decision epoch. The length of this epoch depends on the type and size of the datacenter and its workload. In dynamic optimization procedure, VMC finds a temporary VM placement solution by migrating, admitting, or removing a number of VMs in order to respond to performance, power budget, or critical temperature violations.

We focus on semi-static optimization procedure for the VMC. Here, clients' SLA, expected power consumption of servers, and migration cost of VMs are considered. Migrating a VM between servers causes a downtime in the client's application. Duration of the downtime is related to the migration technique used in the datacenter. For example, live migration causes a downtime amount of less than 100ms [9]. We assume that there is a defined cost in SLA contracts for these short but infrequent downtimes. In this paper, $cost_i^m$ denotes the migration cost of the i^{th} client's VM in the datacenter. Previous assignment variable y_{ij}^p (=1 if the i^{th} client was assigned to j^{th} server and 0 otherwise) is used to calculate the migration cost in the system.

C. Performance Modeling

Performance of each client in the cloud computing system should be monitored and necessary decisions should be taken to satisfy the SLA requirements. We focus on the online services that are sensitive to latency. A client in this system is application software that can produce a number of requests in each time unit. To model the response time of clients, we assume that the inter-arrival times of the requests for each

client follow an exponential distribution function similar to the inter-arrival times of the requests in the e-commerce applications [15]. The minimum allowed inter-arrival time of the requests is specified in the SLA contract. However, the average inter-arrival time (λ_i) of the requests for each client is predicted for the optimization procedures.

Streams of requests generated by each client (application) may be decomposed into a number of different VMs. In case of more than one VM serving the i^{th} client, requests are assigned probabilistically i.e., α_{ij} portion of the incoming requests are forwarded to the j^{th} server (host of a VM) for execution, independently of the past or future forwarding decisions. Based on this assumption, the request arrival rate in each server follows the Poisson distribution function.

There are different resources in the servers that are used by VMs such as processing units, memory, communication bandwidth, and secondary storage. These resources can be allocated to VMs by a fixed or round-robin scheduling policy. In this work, we consider the processing unit and memory to have fixed allocation policy whereas others are allocated by round-robin scheduling. Our algorithm determines the portion of processing unit and memory allocated to each VM, which is assigned to a physical server. The amount of memory allocated to a VM does not significantly affect performance of the VM under different workloads as long as it is no less than a certain value [7]. Hence, we assign a fixed amount of memory (m_i) to the i^{th} client's VM on any server that the client is assigned to.

Share of a VM from the processing unit determines the performance of that VM in the cloud computing system because the applications we are considering are compute-intensive. The portion of processing unit allocated to different VMs (ϕ_{*j}) on a server is determined by the VMC at the beginning of the decision epoch. However, these values can be changed in each server as a function of workload changes or power/performance optimization at the server. VMC considers the clients' workload to determine the resource allocation parameters to control the wait time of the processing queue for different applications based on the SLA requirements.

A multi-class single server queue exists in servers that have more than one VM (from different clients). We consider *generalized processor sharing* (GPS) model at each queue; GPS model approximates the scheduling policy used by most operating systems, e.g., weighted fair queuing and the CPU time sharing in Linux. Using this scheduling policy, multi-class single server queue can be replaced by multiple single-server queues. Note that the processing capacity of the j^{th} server allocated to the i^{th} client's VM is calculated as $C_j^p \phi_{ij}$.

The exponential distribution function is used to model the service time of the clients in this system. Based on this model, the response time ("sojourn time") distribution of a VM (placed on server *j*) is an exponential distribution with mean:

$$\bar{R}_{ij} = \frac{1}{C_j^p \phi_{ij} \mu_{ij} - \alpha_{ij} \lambda_i} \quad (1)$$

where μ_{ij} denotes the service rate of the i^{th} client on the j^{th} server when the a unit of processing capacity is allocated to the VM of this client.

The queuing model used in this paper is M/M/c, which is simplified to M/M/1 with probabilistic request assignment. In case of service times with general distribution, this model is an approximation. This approximation is not appropriate for M/G/1 queues with a heavy-tail service time distribution. However, since we defined the SLA based on the response time constraint, these kinds of service time distribution functions are not encountered. A more general case than the model in (1) would be the M/G/c queuing model. It is not possible to predict the response time distribution of these queues without a numerical approach unless for specific service time distributions. For this reason we believe that using the M/M/c model for this high-level decision making process is sufficient, and more complex models can be used in problems with smaller input size.

D. SLA model for the clients

We use soft SLA constraints, in which cloud provider guarantees that the response time constraint is satisfied for most of the time (h_*^c for example for 95 percentile point of the requests) and for each violation of constraint, the cloud provider pays back the client a fixed penalty value (f_*^c). Having defined the SLAs for clients enables the cloud provider to vary the VM resource size and improve the power efficiency in the system.

The constraint on the response time of the i^{th} client may be expressed as:

$$Prob\{R_i > R_i^c\} \leq h_i^c \quad (2)$$

where R_i and R_i^c denote the actual and target response times for the i^{th} client's requests, respectively.

Using the model provided in the subsection C, the response time constraint for each VM can be expressed as follows:

$$\begin{aligned} & e^{-\left(C_j^p \phi_{ij} \mu_{ij} - \alpha_{ij} \lambda_i\right) R_i^c} \leq h_i^c \Rightarrow \\ & \phi_{ij} \geq \left(\alpha_{ij} \lambda_i - \ln h_i^c / R_i^c\right) / \mu_{ij} C_j^p \end{aligned} \quad (3)$$

IV. PROBLEM FORMULATION

In this work, we focus on an algorithm for solving semi-static VM placement problem in datacenter. The goal of this optimization problem is to minimize the total operational cost of the system including power and migration costs and penalty of violating response time constraint. VMC uses different methods to achieve this goal, including turning on/off servers, migrating VMs, and changing the VM sizes. The cost minimization problem (P1) is provided next:

$$\begin{aligned} Min\ & C_p \sum_j \left[x_j P_j^0 + P_j^p \sum_i \phi_{ij} \right] T_e + \sum_i \sum_j z_{ij} cost_i^m \\ & + T_e \sum_i f_i^c \lambda_i \sum_j \alpha_{ij} e^{-\left(C_j^p \phi_{ij} \mu_{ij} - \alpha_{ij} \lambda_i\right) R_i^c} \end{aligned} \quad (4)$$

Subject to:

$$x_j \geq \textstyle\sum_i \alpha_{ij}, \qquad \forall j \qquad (5)$$

$$\phi_{ij} \geq y_{ij}\left(\left(\alpha_{ij} \lambda_i - \ln h_i^c / R_i^c\right) / \mu_{ij} C_j^p\right), \qquad \forall i,j \qquad (6)$$

$$\textstyle\sum_i \phi_{ij} \leq 1, \qquad \forall j \qquad (7)$$

$$\textstyle\sum_j y_{ij} m_i \leq C_j^m, \qquad \forall j \qquad (8)$$

$$\textstyle\sum_j \alpha_{ij} = 1, \qquad \forall i \qquad (9)$$

$$y_{ij} \geq \alpha_{ij}, y_{ij} \leq 1 + \alpha_{ij} - \varepsilon, \qquad \forall i,j \qquad (10)$$

$$z_{ij} \geq y_{ij} - y_{ij}^p \qquad \forall i,j \qquad (11)$$

$x_j \in \{0,1\}, y_{ij} \in \{0,1\}, z_{ij} \in \{0,1\},$ $\forall i,j$ (12)

$\phi_{ij} \geq 0, \alpha_{ij} \geq 0,$ $\forall i,j$ (13)

where ε is a very small positive value, and, x_j is a pseudo-Boolean integer variable to determine if the j^{th} server is ON (x_j=1) or OFF (x_j=0). We call α_{ij}'s and ϕ_{ij}'s assignment and allocation parameters, respectively throughout the paper.

The first term in the objective function is the energy cost of the system, which is composed of the idle energy cost if the server is active (x_j=1) plus a cost proportional to the utilization of the server. The second term in the objective function captures the migration costs whereas the third term represents the expected penalty that cloud provider will pay to the clients when SLA violations occur.

Constraint (5) ensures that if a client is assigned to a server, this server will be active. Constraint (6) is the SLA constraint for the clients. Constraints (7) and (8) are the processing and memory capacity constraints in a server. Constraint (9) makes sure that all requests of each client are served in the system. Constraint (10) is used to generate a helping pseudo Boolean parameter (y_{ij}) which determines if the i^{th} client is assigned to the j^{th} server (y_{ij}=1) or not (y_{ij}=0). If the value of α_{ij} is more than 0, the first inequality of (10) sets the value of y_{ij} to one and if the value of α_{ij} is zero, the second inequality of (10) force the value of y_{ij} to be zero. Constraint (11) is used to generate a pseudo Boolean parameter (z_{ij}) which indicates whether the migration cost for the i^{th} client from the j^{th} server should be considered (y_{ij}=1 and y_{ij}^p=0) or not. Finally, constraints (12) and (13) specify domains of the variables.

P1 is a mixed integer non-linear programming problem. Integer part of problem comes from the fact that servers can be active or sleep (x_j) and VMs can be placed on a physical machine or not (y_{ij}).

The problem of minimizing the energy cost plus the expected violation penalty is an NP-Hard problem. It can be shown that the NP-hard bin-packing problem [22] can be reduced to P1. Indeed, even deciding whether a feasible solution exists for this problem, does not have an efficient solution. So, we utilize a simple greedy algorithm (similar to First Fit Decreasing (FFD) heuristic [22]) to find a feasible solution to P1 for the given inputs. Another important observation about this problem is that the numbers of clients and servers are very large; therefore, a critical property of any proposed heuristic should be its scalability.

Different versions of this problem are considered in the literature. The shortcoming of the proposed solutions in the previous work is an assumption about knowing the size of VMs based on SLA requirements. Although this assumption is valid for Platform as a Service (PaaS), it is not completely true in case of SaaS. There are two problems with this assumption incase of SaaS: First, SLA contracts in SaaS do not specify the amount of required resource and cloud provider needs a method to translate the target performance metric to the amount of resource for each client; Second, considering fixed resource requirement eliminates the fact that cloud provider may overbooked the datacenter and needs to sacrifice the performance of some of the clients to be able to provide performance guarantee for others. Based on these reasons, we consider the problem of determining the VM sizing and placement together.

V. Cost Minimization Algorithm

In this section, a heuristic for problem P1 is presented. The output of this heuristic is the VM placement and request forwarding policy and the expected performance level of the VMs in the next epoch.

A two step algorithm is proposed for this problem. In the first step, clients are ordered based on their status in the previous decision epoch and their estimated resource requirements for the next decision epoch. Based on this ordering, VMs are placed on servers one by one using dynamic programming and convex optimization methods. This constructive approach may result in servers with low utilization or uncompetitive resource sharing policy within the server. So, in the second step of the algorithm, two local searches are executed to fix these issues.

Details of the SLA-based Power and Migration Cost Minimization algorithm or SPMCM for short are presented below.

A. Initial Solution

To find an initial solution for P1, a constructive approach is used to assign clients to servers and allocate resources to them based on the assignment solution in the previous epoch. For this purpose, clients are divided into four groups. Clients that were served in the previous epoch are placed in one of the first three groups. The first group includes clients that leave the datacenter in the new epoch. The second group includes clients whose request arrival rates drop in the new epoch and the third group includes clients whose request arrival rates rise in the new epoch. Finally, the fourth group includes clients that were not served in the previous epoch.

Clients within these groups are picked in the order of their average minimum processing requirement for VMs (biggest VM first) but the groups are processed in increasing order of their IDs. For clients in the first group, VMC releases their resources and updates the resource availabilities. Resource availability in each server is defined as the amount of processing and memory allocated to the existing VMs.

From other groups, the picked client is assigned to available servers to minimize the operational cost of the cloud computing system. After finding a solution, resource availabilities are updated and the next client is picked for the next assignment. The formulation below describes the operational cost minimization problem for a picked client (P2) (i^{th} client).

$$Min\ C_p \sum_j \left[\left(P_j^0 + P_j^p\right)\phi_{ij}\right] T_e + \sum_j z_{ij} cost_i^m + T_e f_i^c \lambda_i \sum_j \alpha_{ij} e^{-\left(c_j^p \phi_{ij} \mu_{ij} - \alpha_{ij}\lambda_i\right) R_i^c} \quad (14)$$

subject to:

$\phi_{ij} \geq y_{ij}\left(\left(\alpha_{ij}\lambda_i - \ln h_i^c / R_i^c\right)/\mu_{ij} C_j^p\right),$ $\forall j$ (15)

$\phi_{ij} \leq 1 - \phi_j^p,$ $\forall j$ (16)

$y_{ij} m_i \leq \left(1 - \phi_j^m\right) C_j^m,$ $\forall j$ (17)

with the addition of constraints (9)-(13).

ϕ_j^p and ϕ_j^m denote the previously-committed portion of the processing and memory resources on the j^{th} server, respectively.

To eliminate the effect of integer parameter (x_j) on the complexity of the problem, the constant energy cost is replaced by a cost linearly proportional to the CPU utilization of the server. Even with this relaxation, it can be shown that the Hessian matrix for P2 is not guaranteed to be positive or negative definite (or semi-definite). This means that the convex optimization methods cannot be directly applied to solve this problem. However, fixing the value of α_{ij} (between zero and one) makes the problem P2, a convex optimization problem. More precisely, for a fixed α_{ij}, the allocation parameter ϕ_{ij} can be found using convex optimization methods to minimize the energy cost and SLA violation penalty. The complete solution for P2 called DPRA (dynamic programming resource assignment) can thus be found by applying a dynamic programming (DP) technique to examine different assignment parameters for different servers and find the best solution as explained next.

Optimal solution of P2 for constant α_{ij} values and for each server is calculated using Karush-Kuhn-Tucker (KKT) conditions. Using this method, the partial cost of assigning an α_{ij} portion of the i^{th} client's requests to the j^{th} server is calculated which includes the energy cost, migration cost (if the client was not assigned to the j^{th} server in the previous decision epoch) and expected SLA violation penalty value for this resource allocation. Then assignment on different servers should be combined to construct a complete solution for each client with the least total cost. DP technique is used to find the best VM placement for the client (determining best α_{ij}) such that constraint (9) is satisfied.

Since we are dealing with cost minimization for one client at a time, there is no need to consider the complete set of servers for each DP calculation; Instead we can use a small number of servers from each server type plus servers that had served the client in the previous epoch to find the best VM placement solution. Decreasing the number of servers for each DP calculation decreases the time complexity of the solution.

The set of selected servers for the DP technique is different for clients in different groups. In particular, for clients in the second group, only servers that served the client in the previous epoch are considered. For clients in the third and fourth groups, servers that served the client in the previous epoch, a set of servers from each server type with the most unallocated resources, and possibly some inactive servers (from each server type) are considered. To differentiate between using active and inactive servers for VM placement, different slopes for energy cost for active and inactive servers are considered. More precisely, P_j^p and $P_j^p + P_j^0$ may be used as the slopes of energy cost for active and inactive servers, respectively. Note that, to change the slope of the energy cost for different servers, we need to change the first term in the objective function in (14). Algorithm 1 shows pseudo code of DPRA method.

To improve the initial solution, we have used two local search methods; the first one fixes the resource allocation parameters and the second one tries to make under-utilized servers inactive and service their clients with higher energy efficiency on other active or inactive servers.

Algorithm 1: Dynamic Programming Resource Assignment

Inputs: C_p, T_e, $cost_i^m$, λ_i, f_i^c, C_j^p, R_i^c, μ_{ij}, P_j^0, P_j^p, ϕ_j^p, ϕ_j^m, C_j^m, C_j^p, h_i^c and m_i

Outputs: ϕ_{ij}, α_{ij} (i is constant in this algorithm)

```
1  ga= granularity of alpha;
2  For (j = 1 to number of servers)
3     For (α_ij = 1/ga to 1)
4        φ_ij= optimal resource shares based on KKT conditions
5        C(j, α_ij) = C_p T_e (P_j^0 + P_j^p) φ_ij + z_ij cost_i^m
                      + T_e f_i^c α_ij λ_i exp(−(C_j^p φ_ij μ_ij − α_ij λ_i) R_i^c)
6     End
7  End
8  X = ga, and Y = number of servers
9  For ( j =1 to Y)
10    For  (x = 1 to X)
11       D[x,y]= infinity; //Auxiliary X×Y matrix used for DP
12       For (z = 1 to x)
13             D[x,y]=min(D[x,y],D[x-1,y-z]+ C(j, z/ga))
14       D[x,y]=min(D[x,y], D[x-1,y])
15    End
16 End
17 Back-track to find best α_ij's and φ_ij's to minimize cost
```

B. Resource allocation adjustment

If more than one client is assigned to a server, constructive resource allocation may not generate the global optimum allocation policy. We formulate resource allocation problem in a server with fixed assignment parameters (α_{ij}) to minimize the energy cost and SLA violation penalty as a convex optimization problem (P3):

$$Min\ C_p P_j^p \sum_{i\in I_j} \phi_{ij} + \sum_{i\in I_j} f_i^c \alpha_{ij}\lambda_i e^{-\left(C_j^p \phi_{ij}\mu_{ij} - \alpha_{ij}\lambda_i\right)R_i^c} \quad (18)$$

subject to:

$$\phi_{ij} \geq y_{ij}\left(\left(\alpha_{ij}\lambda_i - \ln h_i^c / R_i^c\right)/\mu_{ij}C_j^p\right), \qquad \forall i \in I_j \quad (19)$$

$$\sum_{i\in I_j} \phi_{ij} \leq 1, \quad (20)$$

where I_j denotes the set of VMs assigned to the j^{th} server.

P3 is a convex optimization problem and the solution can be found using KKT optimality conditions. Note that this part of the VM placement algorithm is parallelizable and can be implemented in power managers of the servers.

C. Turn OFF under-utilized servers

To decrease the total cost in the system, it may be possible to turn off some of the under-utilized servers (after finding the initial solution) to reduce the idle energy cost of the servers at the expense of more migration cost (for clients that were assigned to these under-utilized servers in the previous epoch) or more SLA violation penalty.

An iterative method is proposed to find the minimum cost solution based on the results of the previous steps. In each iteration, a server with utilization less than a threshold (e.g., 20%) is chosen and its VMs are removed. To assign the removed VMs to other servers, DPRA method is used. Considering the high energy cost for inactive servers, the

DPRA method encourages the VMC to choose more SLA violation penalty or pay for the migration cost instead of turning on a server. Note that these iterations do not always decrease the total cost in the system; therefore, the global lowest total cost is compared to the total cost after turning off a server, and the move is rejected if it is not beneficial.

This iterative method is continued until all servers with low utilization have been examined.

VI. Simulation Results

To evaluate the effectiveness of the proposed VM placement algorithm, a simulation framework is implemented. Simulation setups, baseline heuristics and numerical results of this implementation are explained next.

A. Simulation Setup

For simulations, model parameters are chosen based on true-to-life cloud computing systems. The number of server types is set to 10. For each server type, an arbitrary number of servers are placed in datacenter. Processors in server types are selected from a set of Intel processors (e.g. Atom, i5, i7 and Xeon) [23] with different number of cores, cache, power consumptions and working frequencies. Active power consumptions for different server types (excluding processor power consumption) are set to vary uniformly between three to six times the power consumption of their fully-utilized processor. Memory capacities of the servers are selected based on cache size of the processors with a constant scaling factor of 1,500. Energy cost is assumed to be 15 cents per KWhr at all times. Request arrival rates of the client are chosen uniformly between 0.1 and 1 request per second. The memory requirements for clients are also selected uniformly between 256MB and 4GB. These parameters are borrowed from the simulation setup of [17].

In each simulation, five different client classes are considered. Each client is randomly picked from one of the client classes. The amount of penalty for different client classes is selected based on the on-demand rates of Amazon EC2 cloud service [24]. Migration costs are set to be equal to downtime penalty of 65ms for each client. In addition, μ_{ij}'s are set based on the highest clock frequency for the servers,.

Each simulation is repeated at least 1000 times to generate acceptable average results for each case. In each simulation, a number of clients are assigned to the servers for the first decision epoch. At the end of each epoch, an arbitrary number of clients leave the datacenter while an arbitrary number of clients join the datacenter. Less than 10% of current clients join or leave the datacenter at the beginning of each epoch. Moreover, inter-arrival rate of the remaining clients in the system are chosen uniformly between 0.1 and 1 request per second for the next epoch. To account for the physical infrastructure overhead, energy cost of the servers in the datacenter is multiplied by a factor of 1.3 as a typical power usage effectiveness of current datacenters [2].

B. Heuristics for Comparison

We implemented a slightly modified version of the FFD [22] for VM placement, called FFDP, and the PMaP heuristic [9] as baseline. These approaches consider VMs that have fixed processing size. We choose $1/2\, h_*^c$ as the expected violation rate of the SLA response time constraints. From (3) the amount of processing units required for different VMs on different physical servers were calculated.

The FFDP method picks clients based on the size of their VM (highest to lowest) and assigns them to the first server with available resources from the server type that has the lowest execution time for the client's requests. The PMaP method is a VM placement heuristic that tries to minimize the power and migration cost. PMaP computes the amount of resources that VMs need, determines the active servers and place the VMs on the servers. After these steps, a power and migration-aware local search is done to find the final solution. Details of PMaP may be found in [9].

C. Numerical Results

Table II shows the performance of the SPMCM method for different number of clients with respect to the lower bound on the total cost. This lower bound is the summation of the lowest cost VM placement solution for each client. This table shows that SPMCM generates a near optimal solution for VM placement that minimizes the power and migration cost. Note that increasing the number of clients decreases the distance between the total cost of SPMCM and the lower bound because of higher consolidation possibility with higher number of clients.

TABLE II. PERFORMANCE OF SPMCM W.R.T. LOWER BOUND COST

# of clients	Average performance	Worst-case performance
250	1.15	1.36
500	1.14	1.23
1000	1.12	1.20
1500	1.09	1.21
2000	1.10	1.24
3000	1.09	1.19
4000	1.10	1.18

Figure 2 demonstrates the normalized total cost of the datacenter using SPMCM, FFDP and PMaP heuristics. It is seen that the SPMCM algorithm generates solutions with total cost, which is on average 35% less than FFDP solutions and 18% less than PMaP solutions. Figure 3 shows the average run-time of SPMCM, FFDP and PMaP methods for different number of clients. Average number of servers in each configuration is equal to the 1.5 times the number of clients. It is clear that SPMCM is more complex than the baseline and PMaP algorithms and hence the run-time of SPMCM is greater than two other algorithms. SPMCM solution is found in less than 25 seconds when the number of clients is equal to 1,000 and the number of servers in the datacenter is 1,500. The VM placement algorithm is called only a few times in each charge cycle (one hour in Amazon EC2 service [24]), e.g., 2-3 times per hour.

Figure 4 shows the average ratio of the expected violation rate of the response time constraint to the maximum allowed violation rate under different penalty values after VM placement using SPMCM. As expected, this ratio decreases by increasing the penalty so as to avoid paying a large penalty cost. In other words, increasing the penalty value forces the cloud provider to provision more resources for the client so that the violation rate (and expected penalty) goes down.

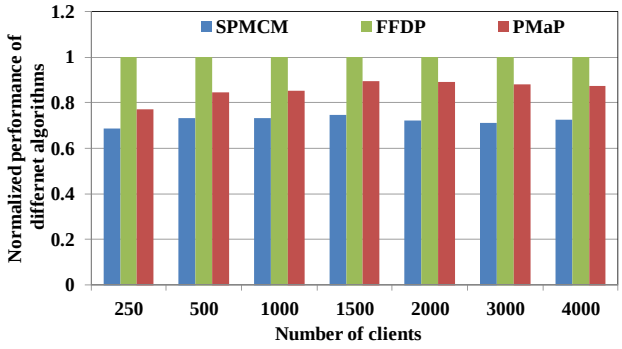

Figure 2. Normalized cost of the datacenter for different algorithms

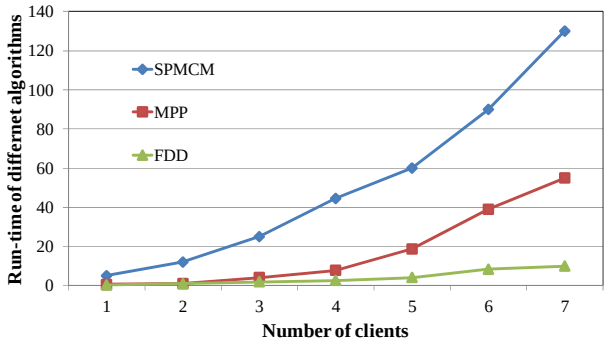

Figure 3. Run-time of SPMCM on 2.8GHZ E5550 server from Intel for different number of clients

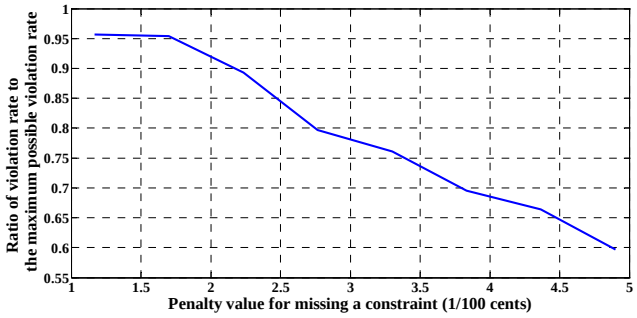

Figure 4. Ratio of expected percentage of the response time constraint's violation to the maximum allowed percentage of violation

VII. CONCLUSION

In this paper we considered VM placement to minimize the power and migration cost in a cloud system. Soft SLA constraints on response time were considered for the clients in this system. We proposed an algorithm based on convex optimization method and dynamic programming. Simulation results demonstrated the effectiveness of our algorithm with respect to a lower bound and other well known solutions. Based on the results of this paper, it can be seen that considering the SLA with effective VM placement can help to minimize the operational cost in the cloud computing system. Considering more general queuing theory models can increase the effectiveness of this work for real world workloads. Moreover, decreasing the complexity of the VM placement problem and experimental evaluation of this work can be done in the future.

REFERENCES

[1] L. A. Barroso and U. Hölzle, The Case for Energy-Proportional Computing, IEEE Computer, 2007.

[2] L. A. Barroso and U. Holzle. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Morgan & Claypool Publishers, 2009.

[3] D. Meisner, B. Gold, and T. Wenisch, PowerNap: eliminating server idle power, in Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2009.

[4] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang and X. Zhu. No "power" struggles: Coordinated multi-level power management for the datacenter. ACM SIGPLAN Notices 43(3). 2008.

[5] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In Proceedings of the 2008 conference on Power aware computing and systems (HotPower'08). 2008.

[6] X. Wang and Y. Wang. Co-con: Coordinated control of power and application performance for virtualized server clusters. Proceeding of the IEEE 17th International Workshop on Quality of Service (IWQoS). 2009.

[7] C. Tang, M. Steinder, M. Spreitzer and G. Pacifici. A scalable application placement controller for enterprise datacenters. Proceeding of 16th International World Wide Web Conference, WWW. 2007.

[8] T. Kimbrel, M. Steinder, M. Sviridenko and A. Tantawi. Dynamic application placement under service and memory constraints. Proceeding of Int'l Workshop on Efficient and Experimental Algorithms. 2005.

[9] A. Verrna, P. Ahuja and A. Neogi. pMapper: Power and migration cost aware application placement in virtualized systems. Proceeding of ACM/IFIP/USENIX 9th International Middleware Conference. 2008.

[10] A. Beloglazov and R. Buyya. Energy efficient resource management in virtualized cloud datacenters. Proceeding of 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid). 2010.

[11] I. Goiri, J.O. Fitó, F. Julià, R. Nou, J.L. Berral, J. Guitart, and J. Torres. Multifaceted resource management for dealing with heterogeneous workloads in virtualized data centers. Proceeding of IEEE/ACM International Conference on Grid Computing (GRID). 2010.

[12] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. Capacity Leasing in Cloud Systems using the OpenNebula Engine. Workshop on Cloud Computing and its Applications. 2008.

[13] R. Buyya, Y S. Chee, S. Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. IEEE International Conference on High Performance Computing and Communications. 2008.

[14] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In Workshop on Power Aware Computing and Systems (HotPower '08). 2008.

[15] Z. Liu, M. S. Squillante and J. L. Wolf. On maximizing service-level-agreement profits. Proceeding of Third ACM Conference on E-Commerce. 2001.

[16] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir and M. Martonosi. Capping the brown energy consumption of internet services at low cost. Proceeding of 2010 International Conference on Green Computing (Green Comp). 2010.

[17] D. Ardagna, B. Panicucci, M. Trubian, L. Zhang, Energy-Aware Autonomic Resource Allocation in Multi-Tier Virtualized Environments. IEEE Transactions on Services Computing. 2010.

[18] L. Zhang and D. Ardagna. SLA based profit optimization in autonomic computing systems. Proceedings of the Second Int. Conf. on Service Oriented Computing. 2004.

[19] A. Chandra, W. Gongt and P. Shenoy. Dynamic resource allocation for shared clusters using online measurements. ACM SIGMETRICS. 2003.

[20] H. Goudarzi and M. Pedram. Maximizing profit in the cloud computing system via resource allocation. Proc. Of international workhop on Datacenter Performance. 2011.

[21] H. Goudarzi and M. Pedram. Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems. Proc. of the IEEE Cloud. 2011.

[22] S. Martello and P. Toth. Knapsack Problems: Algorithms and Computer Implementations. Wiley. 1990.

[23] http://ark.intel.com/

[24] http://aws.amazon.com/ec2/#pricing