Multi-dimensional SLA-based Resource Allocation for Multi-tier Cloud Computing Systems

Hadi Goudarzi and Massoud Pedram
University of Southern California, Los Angeles, CA 90089
{hgoudarz,pedram}@usc.edu

Abstract—**With increasing demand for computing and memory, distributed computing systems have attracted a lot of attention. Resource allocation is one of the most important challenges in the distributed systems specially when the clients have Service Level Agreements (SLAs) and the total profit in the system depends on how the system can meet these SLAs. In this paper, an SLA-based resource allocation problem for multi-tier applications in the cloud computing is considered. An upper bound on the total profit is provided and an algorithm based on force-directed search is proposed to solve the problem. The processing, memory requirement, and communication resources are considered as three dimensions in which optimization is performed. Simulation results demonstrate the effectiveness of the proposed heuristic algorithm.**

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, SaaS (Software as a Service), 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 also been increasing rapidly. Examples include digital services and functions required by the various industrial sectors, ranging from manufacturing to housing, from transportation to banking. Such a dramatic increase in the computing demand requires a scalable and dependable information technology (IT) infrastructure comprising of servers, storage, networks, physical facilities, Electrical Grid, IT workforce, and billions of dollars in capital expenditure and operational cost to name a few.

Modern internet applications are complex software implemented on multi-tier architectures [6]. Each tier provides a defined service to the next tiers and uses services from the previous tiers. The problem of resource allocation for multi-tier applications is harder than that for single tier applications because tiers are not homogenous and a performance bottleneck in one tier can decrease the overall profit (gain from meeting a service level agreement, SLA) even if the other tiers have acceptable service quality.

The IT infrastructure provided by the datacenter owners/operators must meet various SLAs established with the clients. The SLAs include compute power, memory/storage space, network bandwidth, 24-7 availability, data security, and so on. Infrastructure providers often end up over provisioning their resources in order to meet the clients' SLAs. Such over provisioning may increase the cost incurred on the datacenters in terms of both the electrical energy cost and the carbon emission footprint. Therefore optimal provisioning of the resources is imperative in order to reduce the cost incurred on the datacenter operators as well as their environmental impact. The problem of optimal resource provisioning is challenging due to the diversity present in the client applications being hosted and the SLAs. For example: some client applications may be compute-intensive while others may be memory intensive, some applications may run well together while others do not, etc. We focus on the SLA based resource allocation in the cloud computing system. There are two types of SLA contracts. For the Gold SLA class, response time is guaranteed and if this constraint is violated, the cloud provider pays a penalty. For the Bronze SLA class, each client has a defined utility function based on its response time. We used terms Gold and Bronze (as opposed to Silver) to show the large difference between these two SLA classes.

The IT infrastructure provided by the large datacenter owners is often geographically distributed. This helps reduce the peak power demand of the datacenters on the local power grid, results in higher fault tolerance and more reliable operation of the IT infrastructure, and even, reduced cost of ownership. A datacenter comprises of thousands to tens of thousands of server machines, working in tandem to provide services to the clients, see for example [1][2]. In such a large computing system, energy efficiency can be maximized through system-wide resource allocation and server consolidation, this in spite of non-energy-proportional characteristics of current server machines [3]. Clients in 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" [4]. Servers are modeled based on these three capabilities: computational, memory, and networking bandwidth. Cost of operation of active servers is related to the degree of their use of these resources [5].

The remainder of the paper is organized as follows. Related work is discussed next. In section III, the system model and problem formulation are presented. The optimization problem and an upper bound on the profit are given in sections IV and V. The proposed algorithm is presented in section VI. Simulation results and conclusions are given in the sections VII and VIII.

II. RELATED WORK

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

An analytical model for multi-tier internet services is presented in [6]. Processor sharing model is used to derive the average response time of the applications based on mean-value analysis. The solution presented does not have a closed-form because of complex modeling but the accuracy of the solution is demonstrated. Srikantaiah et al. [5] presented energy aware consolidation to decrease the total energy consumption of the cloud computing system. The authors presented an experimental method to model the energy consumption of the servers based on the CPU and disk utilization. Based on this method, a simple heuristic to consolidate the processing works in the cloud computing system is presented.

Multi-dimensional resource allocation for single tier applications in the cloud computing system is presented in [7]. SLA model based on the response time of the applications is considered to model the profit optimization problem. This problem is solved with generating an initial solution and using local optimization techniques. Tang et al. [8] presents a dynamic resource provisioning technique for the case of very large number of servers and application sizes. The proposed heuristic solution for this NP-hard problem is focused on the scalability aspects of the solution. Virtualization management policies are presented in [9] to handle the performance, efficiency and stability of a server system. The results show that effective dynamic resource management can greatly reduce the operation cost of the system and improve the stability of the applications.

In [11], Zhang and Ardagna extend the early work of [10] and present a problem statement with clients that have discrete utility functions. The authors propose a heuristic to solve the problem of assigning different client classes to different servers to maximize the total profit. Ardagna et al. [12] extend this work to profit (revenue) optimization for continuous utility functions in a multi-tier virtualized environment. The authors use a complex model for energy calculation to increase the accuracy and solve the problem by generating a feasible solution and improving it by local search. The availability of servers is considered as a new constraint to the problem in [13]. A heuristic to maximize the profit via decreasing the energy consumption in cloud systems is presented in [14], where an adaptive search based on turning servers on or off is proposed. Resource allocation for tasks with fixed memory, disc and processing requirements is presented in [15]. An approximation algorithm for this problem that is proved to be NP-complete is presented. References [16]-[18] use mathematical or economics-based models to formulate the profit optimization problem.

In this paper, we assume that servers are characterized by their maximum capacity in three *dimensions*: processing power, memory usage, and communication bandwidth. We then focus on multi-dimensional resource allocation in datacenters while guaranteeing SLAs for clients with applications that require *multiple tiers* (stages) of service to complete. Our paper provides the following key features and contributions:

- A multi-dimensional resource allocation scheme to consider communication resources for multi-tier applications that incur large inter-server communications.
- A closed-form formula for calculating the average response time of a client's request for multi-tier applications.
- A unified framework for handling both strong (Gold class) and weak (Bronze class) SLAs.
- A provable upper bound on the total profit in the system for given clients, resources and SLAs.
- A simultaneous server consolidation and client-to-server assignment method based on force-directed search method to maximize the total profit accrued by the system.

III. System Model

An SLA-based multi-dimensional resource allocation scheme for the multi-tier services in a cloud computing system (e.g., a hosting datacenter) is presented to optimize the total expected profit. The profit is composed of the expected gain due to satisfying the SLA contracts of the clients, the expected penalty for violating the contracts, and the energy cost (in dollars) of serving clients' requests. In this paper, we use terms "applications" and "clients" interchangeably.

A cloud computing system comprises of potentially heterogeneous servers chosen from a set of known *server types*. Servers of a given type are modeled by their processing (rate of computational work performed by the server), communication (available bandwidth), and main memory capacities as well as their operational expense (cost) which is directly related to their average power consumption. We assume that the local (or networked) secondary (hard disc) storage is not a system bottleneck. The operational cost of a server is modeled as a constant power cost plus another variable power cost which is linearly related to the utilization of the server in the processing domain. Note that the power cost of communication resources in a datacenter is amortized over all servers and switching gear, assumed to be relatively independent of the clients' workload, and hence, not incorporated in the equation for power cost of a server. We assume that the cloud computing system has a central manager that has information about clients and servers.

Each client is identified by a unique id, represented by index i. Each server is similarly identified by a unique id, denoted by index j. There are often a set of application tiers that an application needs to complete. For each tier, requests of the application are distributed among some of the available servers. Each ON server is assigned to exactly one of these application tiers. This means that if a server is assigned to some tier, it can only serve the requests on that specified tier. Each application has a constraint on memory allocation in each tier. This means that a constant amount of memory should be allocated to the ith client in each server that serves a portion of the client's requests in tier t. No hard constraints are imposed on the processing and communication resource allocations but these allocations determine the system profit.

Table I. Notation and Definitions

Symbol	***Definition***
λ_i	Predicted average request rate of the ith client
λ_i^{max}	Agreed average request rate of the ith client per SLA
λ_i^{EPT}	Expected Profitability threshold on average request rate of the ith client
cc_i	Client class of the ith client
sc_j	Server class type of the jth server
$U_i^b(R_i)$ U_i^g	Utility function value as a function of the response time for each request in the Bronze SLA class
R_i^g, u_i^g, f_i^g	Contract response time target, utility and penalty values for each request in the Gold SLA class
a_i^b	Rate of increasing utility by decreasing the average response time for the ith client in the Gold SLA class
$s_{ij}^{pf,t}, s_{ij}^{cf,t}$ $s_i^{pb,t}, s_i^{cb,t}$	Average processing and communication service times of the jth server for requests of the ith client in the tth tier for forward and backward directions (for communication, it is independent of server type)
m_i^t	Required memory for the tth tier of the ith client
C_j^p, C_j^c, C_j^m	Total processing, communication and memory capacities of the jth server
$p_i^t = 1 - q_i^t$	Probability that requests of ith client do not go to the next tier and instead go back to the previous tier
P_j^0	Constant power consumption of the jth server operation. It is related to sc_j
P_j^p	Power consumption of the jth server in terms of the processing resource utilization related to the $sc(j)$
T_e	Duration of the decision epoch in seconds
C_p	Cost of energy consumption
x_j	A pseudo-Boolean integer variable to determine if the jth server is ON (1) or OFF (0)
y_j^t	A pseudo-Boolean integer variable to determine if the jth server is assigned to the tth tier (1) or not (0)
α_{ij}^t	Portion of the ith client's requests that are in the tth tier and served by the jth server
$\phi_{ij}^{pf,t}, \phi_{ij}^{cf,t}$ $\phi_{ij}^{pb,t}, \phi_{ij}^{cb,t}$ $\phi_{ij}^{m,t}$	Portion of processing, communication and memory resources of the jth server that is allocated to the tth application tier of the ith client (forward (f) or backward (b) directions)

To increase paper readability, Table I presents key symbols used throughout this paper along with their definitions.

A. Multi-tier service model

Consider the ith client with an ordered set of application tiers, T_i. This ordered set is a subset of the available tiers in the cloud computing system. This is a simplified model taken from [6]. The inter-arrival time of requests for the ith client is assumed to follow an exponential distribution with rate parameter λ_i. In addition, in each level of the application tier, the client's requests are distributed among a number of servers. For each tier, there is a probability p_i^t that the requests do not go to the next application tier and instead return to the previous tier. Therefore there are requests moving in two different directions: forward and backward. Although the backward requests are served by the servers that previously served those requests in the forward direction, because the backward streams of requests may have different service times, they are put in different queues. In this model, the requests in the backward direction go to the previous tier with probability of one.

Figure 1 shows an example of a multi-tier system with 3 tiers. D* represent the request dispatchers in different tiers. In this figure, the solid lines represent forward requests while the dash lines show backward requests. For this case, the ordered subset of tiers is $\{u,v,w\}$, which is a subset of T_i by fixing *order*(*u*) < *order*(*v*) < *order*(*w*). Also the probabilities are related to the specific client and the selected subset of tiers used by it.

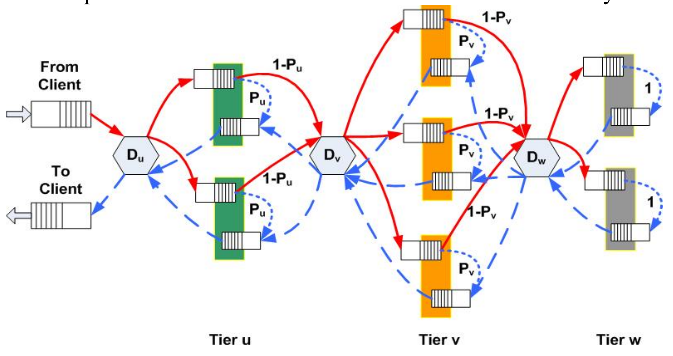

Figure 1. An example of a client with three application tiers.

The processing and communication queues in each server are assumed to be in series. This allows pipelining between processing and communication processes in the servers. In addition, we consider *generalized processor sharing* (GPS) at each queue since GPS approximates the scheduling policy used by most operating systems, e.g., weighted fair queuing and the CPU time sharing in Linux.

Based on the queuing theory, the output of each queue with this characteristic has an exponential distribution with a mean value of $1/(\mu-\lambda)$. The *average response time* of the requests in the queues of a resource (the μ parameter) can be calculated from dividing the percentage of allocated resource (ϕ_{ij}^*) by the average service time of the client's request on that resource (s_{ij}^*) multiplied by the processing or communication capacity of the resource as the case maybe. The *average arrival rate* of the requests for a resource (the λ parameter) can in turn be calculated by multiplying the average arrival rate of the requests by the probability of receiving requests in a specific tier (calculated from probabilities, $p_i^t = 1 - q_i^t$), and the probability of assigning the requests to the server (α_{ij}^t).

We use f and b to denote forward and backward directions whereas p and c denote the processing and communication parts of the application. With this notation, the average response time in the forward direction of processing and communication queues for the ith client in tier t is calculated as follows:

$$R_i^{pf,t} = \sum_j \alpha_{ij}^t \left(\frac{1}{C_j^p \phi_{ij}^{pf,t}/s_{ij}^{pf,t} - \alpha_{ij}^t Q_i^t \lambda_i} \right) \tag{1}$$

$$R_i^{cf,t} = \sum_j \alpha_{ij}^t \left(\frac{1}{C_j^c \phi_{ij}^{cf,t}/s_i^{cf,t} - \alpha_{ij}^t q_i^t Q_i^t \lambda_i} \right) \tag{2}$$

where Q_i^t is the probability of receiving a request in tier t, which is related to the product of the probability of moving in the forward direction in the previous tiers:

$$Q_i^t = \prod_{l \in T_i, order(l) < order(t)} q_i^l \tag{3}$$

In calculating average response time for the communication queue in the forward direction, the average arrival rate for this queue is similar to the (1) multiplied by q_i^t. This is because only requests that are going to be served in the next tier are served in the communication queue of the forward direction of a tier. The average response times for processing and communication queues in the backward direction, which are omitted for brevity, are calculated from equations similar to (1) and (2).

To make the paper more readable, we use M_{ij}^t and Λ_{ij}^t to denote four-element vectors of the service rate and arrival rate of the ith client assigned to the jth server for different directions and different queues.

$$\mathrm{M}_{ij}^t = [\frac{\phi_{ij}^{pf,t}}{s_{ij}^{pf,t}}; \frac{\phi_{ij}^{cf,t}}{s_{ij}^{cf,t}}; \frac{\phi_{ij}^{pb,t}}{s_{ij}^{pb,t}}; \frac{\phi_{ij}^{cb,t}}{s_{ij}^{cb,t}}] \tag{4}$$

$$\Lambda_{ij}^t = Q_i^t \alpha_{ij}^t \lambda_i [1; q_i^t; 1; 1] \tag{5}$$

In the remainder of this paper the kth element of M_{ij}^t and Λ_{ij}^t to will be denoted by $\mathrm{M}_{ij}^{t,k}$ and $\Lambda_{ij}^{t,k}$, respectively.

Based on the GPS technique and model presented above, the average response time of the ith client is calculated as follows:

$$R_i = \sum_{t \in T_i} Q_i^t R_i^t \tag{6}$$

where

$$R_i^t = R_i^{pf,t} + q_i^t R_i^{cf,t} + R_i^{pb,t} + R_i^{cb,t} \tag{7}$$

B. SLA model for this system

Service Level Agreement (SLA) is an important consideration in the system. There are different kinds of SLA in literature but we adopt two classes of SLA's for this paper: (i) average response time guaranteed SLA; and (ii) SLA that has a price pre request based on the response time. The arrival rate of the requests is a random process, which may not even be stationary. If the cloud manager reacts to these changes by limiting the arrival rate of the clients, it is possible to violate the SLA constraints or pay large penalties during busy times. It is also possible that the cloud manager conservatively plans for the maximum arrival rate of the clients, which in turn leads to resource overbooking and increase in the cost of operation.

In this paper, two models of the SLA are considered: (i) the *Gold SLA* class, which specifies an average response time target, a maximum arrival rate for the client's requests, a *utility* (reward) value for each serviced request (regardless of its response time), and a *penalty* if the average request response time is missed; and (ii) the *Bronze SLA* class, which specifies a maximum arrival rate and a *utility function* that specifies a profit per request based on the response time. The arrival rate of a client in the Gold SLA class is determined by a *probability distribution function* (PDF) dynamically profiled and predicted by a cloud-level monitor. This PDF is used to determine the proper amount of resources to allocate to the servers in this SLA class based on the penalty value set in the SLA for exceeding the response time bound. The expected client utility (per request) for the Gold SLA class is calculated as follows:

$$U_i^g = \left(u_i^g - \left(1 - CDF(\lambda_i^{EPT})\right) f_i^g\right) \tag{8}$$

The first term in the parentheses is the constant price that the user pays for the service whereas the second term is the expected penalty for violating the constraint in the SLA. Based on the resources provided to the client, it is possible to calculate an *expected profitability threshold* (EPT) for the request arrival rate i.e., the maximum arrival rate of a client's requests that can satisfy the average response time constraint in the Gold SLA class. Probability of violating the contract average response time constraint is $[1 - CDF(\lambda_i^{EPT})]$ where CDF denotes the *cumulative distribution function* of the predicted average arrival rate of the ith client.

The utility function for the Bronze SLA class is a non-increasing function of the average response time. It is possible that the average response time is higher than a predicted average response time, i.e., there is no guarantee for the response time in this SLA class. We use the predicted average response time (based on the most probable average inter-arrival rate) as the model response time for the user associated with this SLA class.

C. Resource management problem

The goal of the resource management problem is to maximize the total profit for serving the clients. In this system, the decision making interval (called a *decision epoch* from here on) can be defined based on dynamic parameters in the system. In particular, the frequency of changes in the request rates of the clients affects the acceptable decision time. This is because the solution found by the presented algorithm is acceptable only as long as the client behaviors remain stationary during the decision epoch. Although some small changes in the parameters can be effectively tracked and responded to by proper reaction of *request dispatchers*, large changes cannot be handled by the local managers.

In the following, the resource allocation problem in each decision epoch is presented and a solution is presented. However, we do not discuss the characterization and prediction of clients' behavior and dynamic changes in system parameters as these issues fall outside the scope of the present paper.

IV. Problem Formulation

The profit maximization problem is formulated below.

$$Max \sum_{i=1}^{n} \lambda_i^{max} T_e\, U_i - C_p \sum_j \left[x_j P_j^0 + P_j^p \sum_t y_j^t \sum_i \left(\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}\right)\right] T_e \tag{9}$$

Subject to:

$$\sum_t y_j^t \le 1, \qquad \forall j \tag{10}$$

$$x_j \ge \sum_t y_j^t \sum_i \alpha_{ij}^t, \qquad \forall j \tag{11}$$

$$\sum_t y_j^t \sum_i (\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}) \le 1, \qquad \forall j \tag{12}$$

$$\sum_t y_j^t \sum_i (\phi_{ij}^{cf,t} + \phi_{ij}^{cb,t}) \le 1, \qquad \forall j \tag{13}$$

$$\sum_t y_j^t \sum_i (\phi_{ij}^{m,t}) \le 1, \qquad \forall j \tag{14}$$

$$\sum_j \alpha_{ij}^t = 1, \qquad \forall i,t \tag{15}$$

$$\mathrm{M}_{ij}^t \ge \Lambda_{ij}^t, \qquad \forall i,j,t \tag{16}$$

$$\phi_{ij}^{m,t} = z_{ij}^t\, m_i^t / C_j^m, \qquad \forall i,j,t \tag{17}$$

$$z_{ij}^t \ge \alpha_{ij}^t, z_{ij}^t \le \sum_j \alpha_{ij}^t + \alpha_{ij}^t - \varepsilon, \qquad \forall i,j,t \tag{18}$$

$$x_j \in \{0,1\}, y_j^t \in \{0,1\}, z_{ij}^t \in \{0,1\}, \qquad \forall i,j,t \tag{19}$$

$$\mathrm{M}_{ij}^t \ge 0, \phi_{ij}^{m,t} \ge 0, \alpha_{ij}^t \ge 0, \qquad \forall i,j,t \tag{20}$$

with addition of equations (6)-(8). Parameter ε denotes a very small positive value.

In this problem, α_{ij}^t, ϕ_{ij}, x_j and y_j^t are the optimization parameters (cf. Table I for their definitions) whereas the other parameters are constant or functions of the optimization variables. In the objective function, the first part is the summation of the client's utilities. If a client has opted for the Gold SLA class, the utility is calculated from (8); otherwise, the Bronze utility function is used to calculate the utility. The second part of the objective function is the operation cost of the servers. The total power consumption of the servers is calculated by adding the fixed power consumption of the ON servers and variable (utilization-dependent) power consumption. Multiplying the total power consumption by the duration of the epoch produces the energy consumption. Clearly, the average price of a KWh of electrical energy can be used to convert the energy consumption to the operational cost in dollars.

Constraint (10) forces the servers to select only one of the tiers whereas constraint (11) determines the ON servers based on the allocated resources. Constraints (12), (13) and (14) are used to limit the summation of the processing, communication and memory resources in the servers. Constraint (15) ensures that *all* requests generated by a client during a decision epoch are served in the servers. Constraint (16) shows the lower limit of the processing and communication resources in the servers if the allocated client uses the Bronze SLA contract. Constraint (17) determines the amount of the memory allocated to the assigned clients. Assigned clients are determined by a pseudo-Boolean parameter, z_{ij}^t. If α_{ij}^t is not zero, the value of z_{ij}^t is set to one based on the first inequality in (18); otherwise the value of z_{ij}^t is zero as seen from the second inequality in (18). Finally, constraints (19) and (20) specify domains of the variables.

It can be seen that the problem formulation is a mixed integer non-linear programming. This problem cannot be solved by any commercial software because of the large input size (numbers of the clients and servers are large.) An upper bound for the profit is introduced in the next section. A heuristic solution for this problem inspired from the force-directed scheduling is presented in section VI.

V. An Upper-Bound on the Total Profit

The profit maximization is a hard problem and it is very time consuming (it is intractable in the general case) to solve the problem even with linear relaxation of some key system parameters because the number of constraints is too many to use classical optimization methods. In particular, similar problems in the literature (for only Bronze SLA class) [12] are proved to be neither convex nor concave after linear relaxation. More precisely, it can be shown that even if the number and types of the ON servers are known in advance and the utility functions (for the Bronze SLA class) are estimated with continuous decreasing utility functions of the response times, the objective function is neither convex nor concave i.e., the Hessian matrix is not positive definite or negative definite. Therefore, the problem cannot be solved with the convex optimization methods.

We start by using familiar relaxation techniques for finding approximate solutions to bin packing and Knapsack problems [19] to find an upper bound on the total profit. The formulation below describes the profit upper-bound problem statement:

$$Max \sum_{i=1}^{n} \lambda_i^{max} T_e U_i - C_p \sum_j \left[x_j P_j^0 + P_j^p \sum_t \sum_i (\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t})\right] T_e \tag{21}$$

Subject to:

$$x_j = \sum_t \sum_i \left(\phi_{ij}^{cf,t} + \phi_{ij}^{cb,t}\right), \tag{22}$$

$$\phi_{ij,t}^{pf} + \phi_{ij,t}^{pb} \le 1, \qquad \forall i,j,t \tag{23}$$

$$\phi_{ij,t}^{cf} + \phi_{ij,t}^{cb} \le 1, \qquad \forall i,j,t \tag{24}$$

$$m_{i,t}/C_j^m \leq 1, \qquad \forall i,j,t \tag{25}$$

with addition of constraints (15), (16) and (20). Variables of optimization are α_{ij}^t and ϕ_{ij}^*. Notice that the capacity constraints (12), (13) and (14) and the constraint of only one tier for each server (10) have been relaxed. As a result one can solve the profit maximization problem for each client independently of others and simply sum up the results of best profits for all clients to get the best system profit.

One of the pseudo-Boolean variables in the original problem statement is x_j, which converts the problem to a bin packing like problem. To simplify the profit upper-bound problem formulation, we consider x_j as a continuous variable calculated by equation (22). Note that the resulting power cost is a lower bound on the actual power cost of servers in the system. Even with this relaxation, it can be shown that the Hessian matrix of the objective function in (21) is not negative definite or positive definite, and therefore, it is not possible to use the convex optimization method for this problem. To address this last difficulty, α_{ij}^t is fixed in order to make the problem a concave optimization problem with respect to $\phi_{ij}^{p,t}$ and $\phi_{ij}^{c,t}$.

To solve this new problem (with fixed α_{ij}^t) by using the Karush-Kuhn-Tucker (KKT) conditions, we need to obtain the derivatives of the profit with respect to the optimization parameters. Taking this derivative for the Bronze class is straight forward and omitted here for brevity. The derivation of the profit function (*Pr*) with respect to $\phi_{ij,t}^{pf}$ for a client in the Gold class is given below:

$$\frac{\partial Pr}{\partial \phi_{ij,t}^{pf}} = T_e f_i^g \lambda_i^{max} \frac{\partial CDF(\lambda_i)}{\partial \lambda_i} \frac{\partial \lambda_i^{EPT}}{\partial \phi_{ij}^{pf,t}} - T_e C_p P_j^p \tag{26}$$

Note that all calculations are done in $\lambda_i = \lambda_i^{EPT}$.From the definition of λ_i^{EPT}, $\partial \lambda_i^{EPT} / \partial \phi_{ij,t}^{p}$ is calculated as:

$$\frac{\partial \lambda_i^{EPT}}{\partial \phi_{ij}^{pf,t}} = \frac{Q_i^t \alpha_{ij}^t / s_{ij}^{pf,t} \left(\mathrm{M}_{ij}^{t,1} - \Lambda_{ij}^{t,1}\right)^2}{\sum_{t \in T_i} Q_i^t \left[\left(R_i^{pf,t}\right)^2 + q_I^t \left(R_i^{cf,t}\right)^2 + \left(R_i^{pb,t}\right)^2 + \left(R_i^{cb,t}\right)^2\right]} \tag{27}$$

where λ_i in Λ_{ij}^t is set to λ_i^{EPT} for this calculation. The other derivatives of profit have the same form as (26) except that the superscripts are appropriately modified.

This equation shows that, the total profit is more sensitive to the amount of resources allocated to a client that imposes a higher penalty value for violating its response time constraint.

Details of the solution of this optimization using KKT conditions omitted for brevity.

The complete solution of problem (21) in case of the Bronze SLA class can be found by applying *dynamic programming* (DP) as explained next. The solution of the problem for constant α_{ij}^t and for each server type is calculated applying KKT conditions. Using this solution, the partial profit of assigning an α_{ij}^t portion of the ith client's requests from tier t to the jth server is calculated. Then the DP method is used to find the best case of assigning the client's requests to the servers so that constraint (15) for each tier is satisfied. Since we are dealing with profit maximization for one client at a time, there is no need to use the whole set of servers for each DP calculation for each client in each tier; Instead we can use the small number of server types to find the best request distribution rates. The pseudo code for upper-bound calculation algorithm is given in Figure 2.

In case of the Gold SLA class, because the derivatives of the profit with respect to optimization parameters include all of the optimization parameters in a server, it is not possible to find the solution of the problem in one step. Instead, iteration on the solution found by the KKT conditions is used to reach an acceptable γ_{ij}^t for the servers. More precisely, we start with the γ_{ij}^t value obtained for the Bronze SLA class and perform iterations (using numerical techniques [21]) on the solution of the problem calculated applying KKT conditions. These iterations continue until the profits of two consecutive iterations are nearly the same.

```
Algorithm Profit_UB_Calc (i)
Stable = 0 and Profit = 0;
While (Stable == 0){
// Find Optimal Resource Allocation for each server type
  For (k = 1 to number of server types){
          For (t ∈ Ti){
          For ($\alpha_{ik}^t$ = 1/granularity of alpha to 1)
              Find resource shares from KKT conditions ;}}
// Find the Best way to combine resources
  For (t ∈ Ti){
      X = granularity of alpha;
      Y = number of server types * constant;
      For (y =1 to Y){
          For (x = 1 to X){
              D[x,y]=-infinity;
              For (z = 1 to x){//portion of request assignment
                   D[x,y]=max(D[x,y],D[x-1,y-z]+partial profit
                   from alloc (k=div(y,constant) and $\alpha_{ik}^t$=z));}
              D[x,y]=max(D[x,y], D[x-1,y]);}}
      Back track to find the best solution from D[X,Y];}
  IF (class client type is Gold){
      Find EPT arrival rate and Calculate eqn (26) for each
      server;
      IF (no changes from previous step)   Stable =1;}
  Else
      Stable =1;}
Profit = total profit for client i;
```

Figure 2. Pseudo code for calculating a profit upper bound.

As can be seen, all of the dependencies among clients and tiers are removed with relaxation of the capacity and tier allocation to servers. This upper bound on total profit is used as the "golden result" against which the results of our proposed solution are compared. In addition, we use a technique inspired from this profit upper bound calculation to generate an initial solution to the original problem statement as detailed below.

VI. Profit Maximization Solution

In this section, a heuristic, called *force-directed resource assignment* or FRA, to find a solution for the optimization problem in (9) is presented. In this heuristic algorithm, an initial solution based on the solution given for the profit upper bound problem is generated. Next, distribution rates are fixed and resource sharing is improved by a local optimization step. Finally a resource consolidation technique, inspired by the force-directed scheduling, which is one of the most important scheduling techniques in the high-level synthesis [20], is applied to consolidate resources, determine the active (ON) servers and further optimize the resource assignment.

A. Initial Solution

We start by pointing out that the order of resource assignment to the clients and tiers affects the quality of the solution especially when the total computation and communication resources in the system are only just enough to meet the client's requirements.

A greedy technique to rank the clients and the application tiers for each client is used to determine the order of resource assignment processing in our constructive approach. For each client, the following equation is used as its ranking metric:

$$m_i = \sum_j \sum_{t \in T_i} \left(\frac{1}{s_{ij}^{pf,t}} + \frac{q_i^t}{s_{ij}^{cf,t}} + \frac{1}{s_{ij}^{pb,t}} + \frac{1}{s_{ij}^{cb,t}}\right) \tag{28}$$

Clients are ordered in non-decreasing order of this metric and processed in that order (going from low to high metric values.) This allows us to assign resources to the clients that need more resources (client's requests give rise to fairly low service rates) or the number of available resources is lower (client's requests can be served only on a relatively small number of available servers.)

For the selected client, tiers are ordered using a similar metric:

$$n_i^t = \sum_j \left(\frac{1}{s_{ij}^{pf,t}} + \frac{q_i^t}{s_{ij}^{cf,t}} + \frac{1}{s_{ij}^{pb,t}} + \frac{1}{s_{ij}^{cb,t}}\right) \tag{29}$$

For example after selecting a client for resource allocation, the required tiers for this client are ordered based on the summation of the available servers multiplied by the service rate of that server for that specific tier (ordered from low to high.)

After selecting a client and a tier (consider the ith client and tier $t \in T_i$), the solution proposed in section V is used to assign resources to the client in the tier in question. For this assignment, some of the servers are already turned on (due to assignment of previously processed clients) and they have a selected tier. Also some of these ON servers have unassigned resources. To consider these changes, servers that are assigned to other tiers ($y_j^t = 0$) are removed from the resource pool. Also constraints in (23), (24) and (25) are replaced by the following constraints.

$$\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t} \le 1 - \phi_j^p, \qquad \forall i,j,t \tag{30}$$

$$\phi_{ij}^{cf,t} + \phi_{ij}^{cb,t} \le 1 - \phi_j^c, \qquad \forall i,j,t \tag{31}$$

$$C_j^m / m_i^t \le 1 - \phi_j^m, \qquad \forall i,j,t \tag{32}$$

where ϕ_j^p, ϕ_j^c and ϕ_j^m denote the previously-committed portion of the processing, communication and memory resources to the jth server, respectively.

Solution to this problem is the same as the upper bound profit solution for the ith client and tth tier with following exceptions. (i) All ON servers and some of the OFF servers from each server type are used for the DP method; and (iii) A small value is added to the profit of allocating resources from ON servers to prefer these allocations in the DP method in case of a tie.

Resource allocation in this constructive approach is not final because some servers are turned ON and resource can indeed be allocated better. By solving the problem of resource adjustment for each server, the solution is optimal by considering a fixed client to server allocation. This procedure is called *Adjust_ResourceShares*(α_{ij}^t) in the pseudo code.

B. Resource Consolidation using Force-Directed Search

To search the solution space, a method inspired by the force-directed search is used. This search technique is not only a local neighborhood search but also acts like steepest ascent. This characteristic makes this searching technique less dependent to the initial solution.

This algorithm is based on defined forces between servers and clients. A client that has the highest *force difference* toward a new server (difference of forces toward a new server and the server that the client is already assigned to) is picked and if the required server is available, the load replacement is done. After this replacement, forces are updated and the new maximum force differential client-to-server assignment is made. This algorithm continues until there are no positive force differentials for any clients. Because the total profit in this system is not monotonically increasing, the best solution is saved in each step. Figure 3 shows the pseudo code of this technique.

```
Algorithm Resource_Consolidate ()
// Search the solution space to find better profit
TP = total profit;
Initialize the forces between clients and servers;
// calculate force differentials
$D_{i,j\to k}^{\alpha,t} = F_{ik}^{\alpha,t} - F_{ij}^{\alpha,t}$ ; $\forall j,i,t,\alpha$
ΔF = 1;
While (ΔF > 0) {
        ΔF = max ($D_{ij\to k}^{\alpha,t}$); // client i and $\alpha$
        j  = selected source server;
        k  = selected destination server type;
        g  = selected destination server;
        If (ΔF is toward an ON server in server type k){
            g = find the least busy server in k, assigned to tier t;
            If (lower bound constraints satisfied) goto Re-Assign;
            Else goto skip Re-Assign;}
        Else If (ΔF is toward an OFF server in server type k){
            g = find an OFF server in k;
            If (found an OFF server) goto Re-Assign;
            Else goto skip Re-Assign;}
        Else If (ΔF is toward a server serving client i) goto Re-Assign;
Re-Assign:      Re-assign $\alpha$ portion of the requests to g from j;
                Update force related to j, g and client i;
                P = total profit;
                If (P>TP) TP = P; save the state;}
Skip Re-Assign:Update the move limit; }
```

Figure 3. Pseudo code for resource consolidation.

In this search technique, a definition of force is used that is based on the partial profit gained from allocating each portion of the clients' request in a specific tier to a server with specific type. For example, if there are K differnet server types that can execute tier t of the applications, the force toward each server type (for the ith client) is calculated according to (33) for the Gold SLA class and according to (34) for the Bronze SLA class:

$$F_{ik}^{\alpha,t} = \frac{\alpha_{ij}^t}{|T_i|} \lambda_i^{max} f_i^g CDF(\lambda_i^{EPT}) - C_p P_j^p (\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}) \tag{33}$$

$$F_{ik}^{\alpha,t} = -a_i^b \lambda_i Q_i^t \alpha_i \left(R_i^{pf,t} + q_i^t R_i^{cf,t} + R_i^{pb,t} + R_i^{cb,t}\right) - C_p P_j^p (\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}) \tag{34}$$

where k denotes the server type k and ϕ's are the results of the optimal resource allocation problem. Also λ_i^{EPT} is the expected profitability threshold on λ_i based on the new resource allocation. To account for the cost of turning on a server that is off, $T_e C_p P_j^0 (\phi_{ij}^{cf,t} + \phi_{ij}^{cb,t})$ must be subtracted from these forces. These values show the partial improvement in the total profit if a portion of the clients' requests in tier t is assigned to a server from a specific server type.

For each client, forces toward servers having some resources allocated to that client are calculated from other formulas to keep different parts of the application together. This is because splitting a client's requests among servers reduces the total profit in case of equal resources. Also some time, merging parts of a client's request increase the total profit even without increasing the resources used. Details are omitted for brevity.

Based on these forces, the client replacement and re-assignment of the resources are done. In each step, the highest force differential is picked. If the selected destination server is not one of the servers that the client is already assigned to and is an ON server, among all the ON servers assigned to the selected tier on the selected server type, the one with the lowest utilization is picked. If there is any available server to pick, the re-assignment is done only if the available resources on that server satisfy the lower bound constraints on the required resource shares.

After replacement, forces for the selected client are updated. Also forces that are related to the selected source and destination servers are updated. To limit the number of tries in the algorithm and avoid loops and lockouts, we apply the following rules:

- After re-assigning a portion of a client's request, forces toward destination of this re-assignment are updated as a weighted average of the expected partial profit and resulting partial profit.
- The re-assignment of a portion of the client's requests to a server type is locked after some re-assignment in order to avoid loops.
- For a server with utilization less than a threshold, clients are rewarded to leave the server (i.e., there will be less force to keep the clients on the server) so that we can eventually turn off the server.
- We limit the number of re-assignments to control the complexity of the search method.

VII. Simulation Results

In this section we present the simulation results of the proposed solution to evaluate its effectiveness. The number of server types varies between 2 and 10. For each server type, an arbitrary number of servers exist as explained below. We consider 10 to 100 clients in the system and for each client the processing power and memory capacity requirements are set with random variables to model clients with different requirements. Ten different application tiers in the system are considered. The number of application tiers for each client is selected randomly to be between 3 and 5 and the probabilities of going forward or backward in the corresponding tier graph are randomly set with average of 80% going forward and 20% backward for each tier. Service times for clients with different application tiers on different server types are also modeled with random variables. Each client is assumed to have Gold or Bronze SLA class with probability of 50%.

To model the PDF for the arrival rate of the client requests in the Gold SLA classes, we used linear function between zero and the maximum arrival rates.

The power dissipation cost of different server types is determined as the random variables. The mean of these random variables is set based on the real experimental results. Also for the memory capacity of the server types, random variables based on actual available servers are considered. The processing and communication capacities of the server types are selected arbitrarily from the actual available servers such as Intel Xeon processors and Gigabyte communication ports.

We used two different *scenarios* in terms of number of servers. In the first scenario (low server to client ratio), the average number of servers is $5n$ where n denotes the number of clients. In the second scenario (high server to client ratio), the average number of servers is set to $10n$. Recall that from the distribution of tiers per client, there are (on average) $4n$ client-tiers in both scenarios system.

The baseline method used to compare with the results of the proposed force-directed resource assignment algorithm (called FRA) is an iterative method (IM) based on fixing the resource shares and optimizing the task distribution rates and then fixing the distribution rates and optimizing the resource shares. This is similar to the iterative improvement approach of [12] and [13]. We also compare our results with the upper bound solution (UB) of section V. In particular, FRA/UB and FRA/IM columns in

Table II. Quality of the final solution

Client count, n	First Scenario (low server count – high workload)		Second Scenario (high server count = low workload)	
	FRA/UB	FRA/IM	FRA/UB	FRA/IM
10	58%	114%	80%	140%
20	54%	116%	71%	117%
30	57%	120%	69%	139%
40	52%	117%	76%	142%
50	57%	109%	76%	110%
60	55%	107%	80%	109%
70	53%	107%	77%	120%
80	50%	113%	76%	108%
90	49%	115%	77%	107%
100	49%	110%	84%	105%

Table II reports the quality (expected total system profit) of our solution with respect to the upper bound and iterative method solution, respectively.

As can be seen, the quality of our solution compared to the upper bound is quite different for the first and second scenarios. This is because the upper bound solution does not capture the resource availability in the system and only finds the best possible profit in case of no competition among clients to reserve the resources. Also this table shows that FRA generates a better solution with respect to the iterative improvement method.

To show the effectiveness of the server consolidation technique that is proposed in this paper, an initial solution based on the proposed initial solution with the constraint of $\alpha_{ij}^{t} \leq 0.2$ is generated and the force-directed search is used to find the final solution. Trace of the execution for this case is shown in Figure 4. In this case, number of client is set to 50 and the first scenario is used for the number of servers.

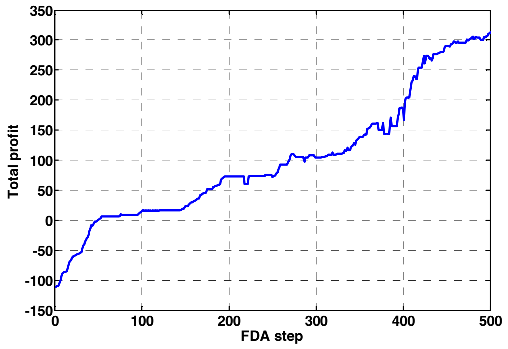

Figure 4. An example of Resource Consolidation Improvement in case of a bad initial solution.

The upper bound profit in this case is 2000. It can be seen that the resource consolidation method based on the force-directed search increases the total profit saliently because the initial solution used for this case is not a good initial solution. Also some decrease in the total power can be seen which is because of the nature of the force-directed search.

Figure 5 shows the average run time of the proposed heuristic for different number of clients and different scenarios. Although the average number of servers for the second scenario is double this number for the first scenario, the run time does not increase a lot because the force-directed search is based on the server types not the actual servers. It can be seen that in case of having an average of 400 client-tiers and 1000 servers, the solution is found in less than 1.5 minutes which is acceptable for cases with decision epoch in order of half an hour.

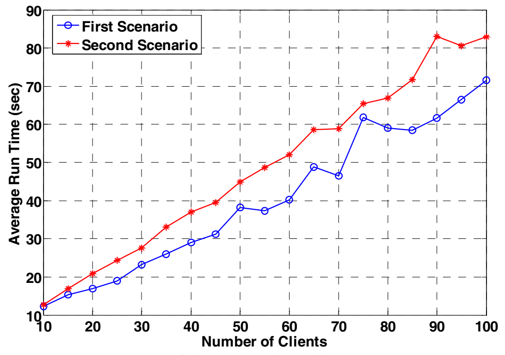

Figure 5. Average run time of the FRA algorithm on 2.8GHZ E5550 server from Intel for different number of clients.

To show the characteristic of the proposed solution, Figure 6 shows the average ratio of $\lambda_i^{EPT}/\lambda_i^{max}$ for the clients with the Gold SLA class for different ratio of f_i^g/u_i^g. As it is expected, the ratio of the EPT arrival rate is increased to compensate increase in the penalty value. For some penalty values, the EPT arrival rate is more than the contract arrival rate which is because of iterative nature of the resource allocation in case of the Gold SLA class.

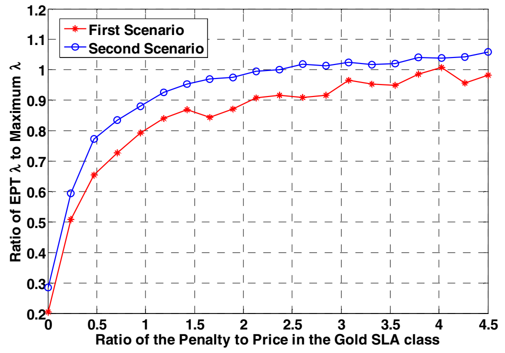

Figure 6. Ratio of the EPT inter-arrival rate to the maximum inter-arrival rate for different penalty values for Gold SLA class.

Figure 7 shows the average utilization factor of the servers in case of different P_j^p/P_j^0 values. Lowering the value of P_j^p/P_j^0 means that the idle energy cost has a bigger portion in the total energy cost in case of full utilization which may be resulted in more consolidation in servers and less ON servers.

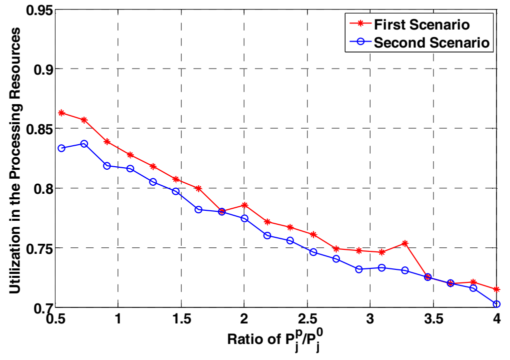

Figure 7. Utilization of the servers for different P_j^p/P_j^0 values.

VIII. Conclusion

In this paper, we considered the problem of the resource allocation to optimize the total profit gained from the SLA contracts and lost from operational cost. The model based on the multi-tier applications is presented and the guarantee based SLA is used to model the profit in the system. An upper bound on the profit of each client is found by relaxing the capacity constraint of the servers between different clients. A solution based on generating an initial solution inspired from the upper bound and a resource consolidation technique based on the force-directed search is proposed. The quality of the solution is compared to the upper bound solution and the iterative improvement approach proposed in the previous work.

Acknowledgement: This work was sponsored in part by a grant from the National Science Foundation.

References

[1] M. Armbrust, A. Fox, et al. A view of cloud computing. *Communications of the ACM* 53(4), 2010.

[2] R. Buyya. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. *The 9th IEEE/ACM Symposium on Cluster Computing and the Grid*, 2009.

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

[4] 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*, Chicago, Illinois, USA, 2008.

[5] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. *Power Aware Computing and Systems*, San Diego, USA, December 2008.

[6] B. Urgaonkar, G. Pacifici, P. Shenoy,M. Spreitzer, and A. Tantawi. An analytical model for multi-tier Internet services and its applications, *ACM International Conference on Measurement and Modeling of Computer Systems*, 2005.

[7] H. Goudarzi and M. Pedram, "Maximizing profit in the cloud computing system via resource allocation," *Int'l workshop on Data Center Performance,* Minneapolis, MN, Jun. 2011.

[8] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifci. A scalable application placement controller for enterprise data centers. *Int'l Conf. on WWW*, 2007.

[9] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan, K. Schwan. vManage: loosely coupled platform and virtualization management in data centers. *Int'l Conf. on Autonomic Computing*, 2009.

[10] Z. Liu, M. S. Squillante and J. L. Wolf. On maximizing service-level-agreement profits. *The Third ACM Conference on Electronic Commerce*, 2001.

[11] L. Zhang and D. Ardagna. SLA based profit optimization in autonomic computing systems. *The Second Int. Conf. on Service Oriented Computing*, November 2004.

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

[13] B. Addis, D. Ardagna, B. Panicucci, Z. Li. Autonomic Management of Cloud Service Centers with Availability Guarantees. *IEEE 3rd International Conference on Cloud Computing*, July 2010.

[14] M. Mazzucco, D. Dyachuk, R. Deters. Maximizing Cloud Providers' Revenues via Energy Aware Allocation Policies. *IEEE 3rd International Conference on Cloud Computing*, July 2010.

[15] F. Chang, J. Ren, R. Viswanathan. Optimal Resource Allocation in Clouds. IEEE 3rd International Conference on Cloud Computing, July 2010.

[16] C. Santos, X. Zhu, and H. Crowder. A mathematical optimization approach for resource allocation in large scale clusters. *Technical Report HPL-2002-64, HP Labs*, March 2002.

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

[18] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat and R. P. Doyle. Managing energy and server resources in hosting centers. *ACM SOSP*, 2001.

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

[20] P.G. Paulin, J.P. Knight. Force-directed scheduling for the behavioral synthesis of ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.8, no.6, pp.661-679, Jun 1989.

[21] J.Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.