To appear in *Large Scale Network-Centric Computing Systems*,
Editors: H. Sarbazi-Azad and A. Zomaya, Wiley Series on Parallel and Distributed Computing, Jul. 2013

Profit-Maximizing Resource Allocation for Multi-tier Cloud Computing Systems under Service Level Agreements

Hadi Goudarzi and Massoud Pedram

University of Southern California

Department of Electrical Engineering

Los Angeles CA, USA

Abstract— Pervasive use of cloud computing and resulting rise in the number of hosting datacenters (which provide platform or software services to clients who do not have the means to set up and operate their own facilities) have brought forth many challenges including energy cost, peak power dissipation, cooling, carbon emission, etc. With power consumption becoming an increasingly important issue for the operation and maintenance of the hosting datacenters, corporate and business owners are becoming increasingly concerned. Furthermore provisioning resources in a cost-optimal manner so as to meet different criteria such as response time, throughput, network bandwidth, etc. has become a critical challenge. The goal of this chapter is to provide a review of resource provisioning and power management strategies that optimize server-related energy costs in a datacenter while meeting the stipulated service level agreements in terms of throughput and response time.

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 communal levels, encompassing business, commerce, education, manufacturing, and communication services. At personal level, the wide scale presence of online banking, e-commerce, SaaS (Software as a Service), social networking and so on 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 various industries, ranging from manufacturing to housing, and from transportation to banking. Such a dramatic increase in the computing resources requires a scalable and dependable IT (information technology) infrastructure comprising of servers, storage, network bandwidth, physical infrastructure, Electrical Grid, personnel and billions of dollars in capital expenditure and operational cost to name a few.

Datacenters are the backbone of today's IT infrastructure. The reach of datacenters spans a broad range of application areas from energy production and distribution, complex weather modeling and prediction, manufacturing, transportation, entertainment and even social networking. There is a critical need to continue to improve efficiency in all these sectors by accelerated use of computing technologies, which inevitably requires increasing the size and scope of datacenters. However, datacenters themselves are now faced with a major impediment of power consumption. To put their energy consumption in perspective, the peak power demand from all existing datacenters in the USA is estimated at 7 Gigawatts (2006 data) [1]. Power consumption of datacenters will soon match or exceed many other energy-intensive industries such air transportation.

Apart from the total energy consumption, another critical component is the peak power; the peak load on the power grid from datacenters is currently estimated to be approximately 7 Gigawatts (GW), equivalent to the output of about 15 base-load power plants. This load is increasing as shipments of high-end servers used in datacenters (e.g., blade servers) are increasing at a 20-30 percent CAGR.

The environmental impact of datacenters is estimated to be 116.2 million metric tons of CO2. Realizing the perils of spiraling carbon emissions growth, the Intergovernmental Panel on Climate Change (IPCC) has called for an overall greenhouse gas emissions reduction of 60-80 percent—below levels from 2000—by 2050 to avoid significant environmental damage. It is believed that the ICT industry can enable a large portion of that reduction. By providing actors with the information necessary to make better decisions about energy consumption, ICT solutions can reduce the carbon footprint of human activity while improving quality of life. ICT-enabled solutions may cut annual CO2 emissions in the U.S. by 13-22 percent from business-as-usual projections for 2020, which translates to gross energy

and fuel savings of $140-240 billion dollars [2]. To enable a sustainable ICT ecosystem both economically and environmentally, it is imperative that datacenters implement efficient methods to minimize their energy use, thus reducing their ecological footprint and becoming more "green."

A significant fraction of the datacenter power consumption is due to resource over-provisioning. For instance, a recent EPA report [3] predicts that if datacenter resources are managed with state-of-the-art solutions, the power consumption in 2011 can be reduced from 10 Gigawatts to below 5 Gigawatts. These solutions require well-provisioned servers in datacenters. More precisely, today's datacenters tend to be provisioned for near-peak performance since typical service level agreements (SLAs) between clients and hosting datacenters discourage the development of significant performance bottlenecks during peak utilization periods. In order to achieve peak performance with minimum power dissipation under given SLAs, we need to design computing systems with the least energy consumption per instruction.

Optimal provisioning allows datacenter operators to use only the minimum resources needed to perform the tasks arriving at the datacenter. Optimal provisioning is complicated by the fact that over time datacenter resources become heterogeneous, even if a datacenter is initially provisioned with homogeneous resources. For instance, replacing non-operational servers or adding a new rack of servers to accommodate demand typically leads to installing new resource types that reflect the advances in current state-of-the-art in server design.

System-wide power management is another huge challenge in the datacenter. First, restrictions on availability of power and large power consumption of the IT equipment make the problem of datacenter power management a very difficult one to cope with. Second, the physical infrastructure (e.g., the power backup and distribution system and the computer room air conditioning, or CRAC for short, systems) tends to account for over one third of total datacenter power and capital costs [4]-[6]. Third, the peak instantaneous power consumption must be controlled. The reason for capping power dissipation in the datacenters is the capacity limitation of the power delivery network and CRAC units in the datacenter facility. Fourth, power budgets in datacenters exist in different granularities: Datacenter, cluster, rack or even servers. A difficulty in the power capping is the distributed nature of power consumption in the datacenter. For example, if there is a power budget for a rack in the datacenter, the problem is how to allocate this budget to different servers and how to control this budget in a distributed fashion. Finally, another goal is to reduce the total power consumption. A big portion of the datacenter operational cost is the cost of electrical energy purchased from the utility companies. A trade-off exists between power consumption and performance of the system and the power manager should consider this trade-off carefully. For example, if the supply voltage level and clock frequency of a CPU are reduced, the average power consumption (and even energy needed to execute a given task) is reduced, but the total computation time is increased.

In general, datacenters serve different (sometimes independent) applications or serve the same application for different clients. If a physical server is dedicated for each application, the number of applications that datacenter can support will be limited to the number of physical servers in the datacenter. Also, allocating one application to each server can be energy inefficient because of energy usage pattern of the applications.

Virtualization technology creates an application-hosting environment that provides independence between applications that share a physical machine together [7]. Nowadays, computing systems rely heavily on this technology. Virtualization technology provides a new way to improve the power efficiency of the datacenters: consolidation. Consolidation means assigning more than one Virtual Machines (VM) to a physical server. As a result, some of the servers can be turned off and power consumption of the computing system decreases. This is because servers consume more than 60% of their peak power in the idle state and turning off a server improves the power efficiency in the system. Again the technique involves performance-power tradeoff. More precisely, if workloads are consolidated on servers, performance of the consolidated VMs (virtual machines) may decrease because of the reduction in the available physical resources (CPU, memory, I/O bandwidth) but the power efficiency will improve because fewer servers will be used to service the VMs.

The IT infrastructure provided by the datacenter owners/operators must meet various service level agreements (SLAs) established with the clients. The SLAs include compute power, storage space, network bandwidth, availability and security, etc. 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 the electrical energy bill. Therefore optimal provisioning of the resources is imperative in order to reduce the cost incurred on the datacenter operators.

The IT infrastructure provided by large datacenter owners/operators is often *geographically distributed*. This helps with reducing the peak power demand of the datacenters on the local power grid, allow for more fault tolerance and reliable operation of the IT infrastructure, and even, reduced cost of ownership. A datacenter however comprises of thousands to tens of thousands of server machines, working in tandem to provide services to the clients, see for example [8] and [9]. 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 [10]. More precisely, compute servers suffer from significant imbalances in their energy proportionality. In particular, they consume 80% of the peak power even at 20% utilization [11]. Hence, it is insufficient to consider energy cost of operating a datacenter based on the number of active servers; in fact, it is critical to consider the energy cost as a function of server utilization.

This chapter starts off by providing a review of the important approaches and techniques for addressing the power and performance management problems in a datacenter. The review is by no means comprehensive, but aims to present some key approaches and results. Next, a problem related to maximization of the profit in a hosting datacenter or cloud computing system is discussed and an efficient solution is proposed. The chapter is thus organized as follows. A review of some of the approaches and techniques addressing power management and performance management is presented in the section II and III. System model for the profit maximization problem is presented in section IV. The problem formulation is presented in section V. The proposed solution is presented in section V. Simulation results and conclusions are given in the sections VI and VII.

II. Review of Datacenter Power Management Techniques

Various approaches for power management in datacenter have been presented in the literature. Power management is strongly linked with job (VM) management (including job migration and consolidation) in a datacenter. As a result, most of the prior work [12]-[20] considers power management along with job management. Some researchers use control theoretic approaches [12]-[14], [19] and [21] while others rely on heuristic approaches for power management [11] and [15]-[17]. Form another perspective, some of the prior work focuses on average power reduction [11] and [15], another group focus on the power capping problem [13] and [15], yet others consider both of these issues [12] and [20].

Two main techniques in saving energy consumption for non-peak workload conditions are (1) voltage and frequency scaling of CPUs in servers and (2) turning on and off servers. Individual and coordinated voltage and frequency scaling as well as turn on/off policy are compared with each other in terms of the resulting power saving potential [22]. According to this study, voltage and frequency scaling policy results 18% power saving subject to meeting the performance constraints whereas the policy that considers turning on/off servers produces 42% power saving. At the same time, 60% saving in power consumption can be achieved for a policy with coordinated voltage and frequency scaling along with turning servers on/off based on the workload intensity.

In another work [12], Raghavendra et al. present a power management solution that coordinates different power saving approaches in order to avoid interference among these approaches. More precisely, the authors propose a hierarchical approach for power management. At the lowest level, an efficiency controller optimizes the average power consumption in each server by adjusting the P-state of the CPU to match the future demand. Moreover, a server manager is used to prevent (thermal) power budget violation in the servers. At the top-most level, enclosure and group managers are used to control the (thermal) power in blade enclosures/racks and at the datacenter level, respectively. These controllers are designed in a coordinated fashion to avoid negative interference in the power and performance domains. A VM

controller is deployed to decrease the average power consumption (operational cost) in the datacenter by consolidating the VM's into as few servers as possible, and subsequently, turning off any unused servers. Current server utilization information is used as the input to this controller, which decides about the VM to server assignments so as to decrease the power consumption at the rack or datacenter levels. Moreover, information about approximate power budget caps and any power violations in the previous decision making interval is used to implement an effective VM to server assignment. The authors use a greedy bin-packing heuristic as an approximation of the optimal solution in VM controller.

To evaluate their proposed architecture, a trace-based simulation approach for datacenter is used which utilizes real-world enterprise traces for server simulation. Simulation results show that most of the power saving comes from VM controller in case of low to medium utilization of servers while increasing the server utilization makes the local power optimizer more important.

Srikantaiah et al. in [11] present an energy aware consolidation technique for a cloud computing system. The authors report the performance, energy consumption, and resource utilization of the servers under different resource usages. Each server is considered to have two resource dimensions: disk and CPU. They notice that server utilization and performance are affected by consolidation in a non-trivial manner because of the contention for shared resources among consolidated applications. Based on these observations, they recommend doing consolidation carefully so as not to over-utilize servers in one resource dimension. The task placement problem, which is a bin packing problem by considering two dimensional resources in the servers, is discussed and a greedy solution is proposed for that problem.

Wang and Wang in [13] present coordinated control of power and application performance for virtualized server clusters. The authors present a coordinated approach based on feedback control theory to manage the (peak) power consumption and performance of the VMs in a server cluster. The proposed architecture includes a cluster-level power control loop and a performance control loop for every VM. Cluster-level power control loop measures the total power consumption of the cluster and, based on the difference of this value from the target power consumption, calculates the CPU frequency for each server and sends the frequency value to the DVFS controller in physical machines. For each VM, there is a performance controller to ensure satisfying a response time constraint. Based on the value of the average response time, this control loop adjusts the CPU resources allocated to a VM by changing its share from the CPU cycles. Moreover, a cluster-level resource coordinator is used to migrate the VMs if they are assigned to machines without sufficient resource. This migration is implemented by first-fit greedy heuristic to find the first physical host for VMs.

III. REVIEW OF DATACENTER PERFORMANCE MANAGEMENT TECHNIQUES

Different definitions for datacenter performance have been used in the published work. Whenever we consider a private datacenter, maximizing the number of serviced applications in each time slot can be used to capture the datacenter performance [24]. This is especially true when all ready jobs can be executed in the given time slot by some server in the datacenter. For example, in Google's datacenters, the maximum number of serviced web queries in a unit of time denotes the datacenter performance. For these cases, the energy cost is proportional to the datacenter performance (energy proportionality is achieved in the aggregate.) In public datacenters (i.e., the hosting datacenters or public cloud providers), performance can be defined as the total profit accrued by the datacenter owner by serving every client's requests [34]. For example, in the Amazon EC2 public cloud, the profit that the cloud provider can get by serving the clients while meeting their SLAs determines the system performance.

A datacenter's performance is strongly influenced by the resource management approach used in the datacenter. Job scheduling and job-to-server assignment (or virtual machine management and placement in virtualized datacenters) are two of the important techniques used in resource management to optimize performance, energy consumption, and profit in datacenters. These problems have extensively been considered in the literature. Some of the published work ([20] and [23]-[26]) describe industrial platforms and solutions whereas others ([27]-[31]) represent solutions presented by academia. From a different viewpoint, some of the publications ([23]-[26] and [32]) focus on performance while others ([27]-[31]) consider both performance and power dissipation.

In the following paragraphs, we explain a representative work due Verna et al [28]. The work presents a framework, called pMapper, and an algorithm for solving power and migration cost aware application placement in a virtualized datacenter. pMapper relies on an application placement controller with the objective of minimizing the power dissipation and migration cost subject to a performance requirement.

Various actions in the pMapper are categorized as: (i) soft actions like VM re-sizing, (ii) hard actions such as Dynamic Voltage Frequency Scaling (DVFS), and (iii) server consolidation actions. These actions are implemented by different parts of the implemented middleware. There is a performance manager, which has a global view of the applications and their SLAs and commands the soft actions. A power manager issues hard actions whereas a migration manager triggers consolidation decisions in coordination with a virtualization manager. These managers communicate with an arbitrator as the global decision making part of the system to set the VM sizes and find a good placement of applications onto the servers based on the inputs of different managers. In particular, an efficient algorithm is presented for placing applications on the physical servers so as to minimize the total system cost assuming fixed VM sizes. Servers are ordered based on their power efficiency figures and a first-fit decreasing heuristic is used to place the applications on the servers starting with the most power-efficient server.

To provide an acceptable quality of service or to meet negotiated service level agreement with the clients, datacenter performance should be estimated accurately. Different approaches have been proposed for performance modeling in datacenters. Queuing theoretic models are used in some works including [33]-[41]. Other work use learning-based modeling [40] or control theory-based modeling [21] approaches. The average response time [34] and instantaneous response time [37] have been used to specify service level agreements. Similarly, hard deadlines [37] and utility functions (as a function of response time) [33] have been used to set SLAs. Energy consumption and power management are considered in some of the published works along with performance management [36] [40] - [44].

In the following paragraphs, a SLA-based resource management algorithm based on the queuing (proposed by Zhang and Ardagna in [34]) is explained in some detail. The authors have focused on a theoretical method to model performance and used it for profit optimization problem in an autonomic computing system. In the considered system, a centralized dispatch and control unit distributes users' requests to various servers and determines the scheduling policy.

More precisely, an enterprise computing system with a number of heterogeneous server clusters is considered. Each server cluster comprises of multiple servers of the same type. Each server offers only one type of resource, i.e., computing resource (memory or I/O bandwidth are not considered). For each client, SLA is defined based on a utility function. This utility function, which is a discrete function, determines the price that client pays based on the average response time that he/she receives for its tasks. Generalized processor sharing is used as scheduling policy for each server. Using these models, an optimization problem to maximize the profit in the system is introduced. Profit is defined as the difference between the revenue gained from utility functions (defined based on the clients' SLAs) and the operational cost of the system (which is energy consumption cost of the active servers.) This problem is formulated as a mixed integer programming problem. It is shown that the multi-choice Knapsack problem is reducible to the aforesaid problem and hence the problem is NP-hard. .

Properties of the profit optimization problem are analyzed and an optimization algorithm is presented for this problem. A fixed-point iteration procedure is employed to determine the optimal scheduling policy while a network flow problem is solved to determine the request forwarding policy between different servers. A simulation environment is used to evaluate the performance of the proposed solution. Comparison with the proportional sharing mechanism shows 25% energy reduction using the proposed solution for the case of a fixed number of active servers.

Ardagna et al. [35] extend the works in [34] to multi-tier application placement considering SLAs based on response time and present an optimization technique to optimize the profit. Local search, fixed point iteration, and greedy heuristics are deployed in the proposed solution.

IV. System Model of Multi-Tier application Placement Problem

Modern internet applications are complex software implemented on multi-tier architectures [39]. 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.

In this section, a 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 chapter, 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 i^{th} 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.

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

Table I. Notation and Definitions

Symbol	***Definition***
λ_i	Predicted average request rate of the i[th] client
λ_i^{max}	Agreed average request rate of the i[th] client per SLA
λ_i^{EPT}	Expected Profitability threshold on average request rate of the i[th] client
cc_i	Client class of the i[th] client
sc_j	Server class type of the j[th] 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 pe alty values for each request in the Gold SLA class
a_i^b	Rate of increasing utility by decreasing the average response time for the i[th] 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 j[th] server for requests of the i[th] client in the t[th] tier for forward and backward directions (for communication, it is independent of server type)
m_i^t	Required memory for the t[th] tier of the i[th] client
C_j^p, C_j^c, C_j^m	Total processing, communication and memory capacities of the j[th] server
$p_i^t = 1 - q_i^t$	Probability that requests of i[th] client do not go to the next tier and instead go back to the previous tier
P_j^0	Constant power consumption of the j[th] server operation. It is related to sc_j
P_j^p	Power consumption of the j[th] 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 j[th] server is ON (1) or OFF (0)
y_j^t	A pseudo-Boolean integer variable to determine if the j[th] server is assigned to the t[th] tier (1) or not (0)
α_{ij}^t	Portion of the i[th] client's requests that are in the t[th] tier and served by the j[th] 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 j[th] server that is allocated to the t[th] application tier of the i[th] client (forward (f) or backward (b) directions)

A. Multi-tier service model

Consider the i[th] 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 [39]. The inter-arrival time of requests for the i[th] 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 those servers that previously served the 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.

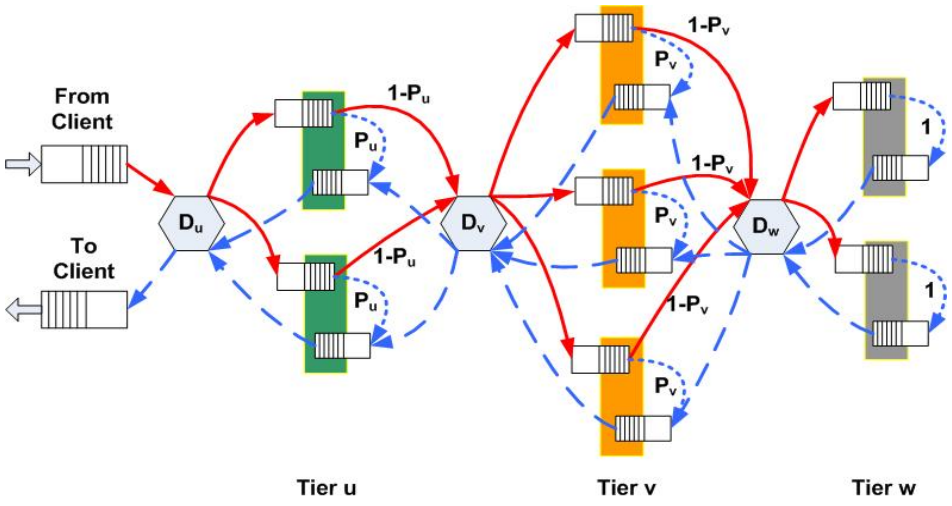

Figure 1. An example of a client with three application tiers.

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 i[th] client in tier t is calculated as follows:

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

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

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:

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

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 improve readablity, we use M_{ij}^t and Λ_{ij}^t to denote four-element vectors of the service rate and arrival rate of the i[th] client assigned to the j[th] server for different directions and different queues.

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

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

In the remainder of this chapter the k[th] 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 i[th] 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: (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 chapter, 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:

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

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$[1 - CDF\left(\lambda_i^{EPT}\right)]$ 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 writeup.

V. PROFIT MAXIMIZATION IN A HOSTING DATACENTER

A. Problem Formulation

The profit maximization problem in a hosting datacenter is formulated next.

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

Subject to:

$$\sum_t y_j^t \leq 1, \qquad \forall j \quad (10)$$

$$x_j \geq \sum_t y_j^t \sum_i \alpha_{ij}^t, \qquad \forall j \quad (11)$$

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

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

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

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

$$\mathrm{M}_{ij}^t \geq \Lambda_{ij}^t, \qquad \forall i,j,t \quad (16)$$

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

$$z_{ij}^t \geq \alpha_{ij}^t, z_{ij}^t \leq \sum_j \alpha_{ij}^t + \alpha_{ij}^t - \varepsilon, \qquad \forall i,j,t \quad (18)$$

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

$$\mathrm{M}_{ij}^t \geq 0, \phi_{ij}^{m,t} \geq 0, \alpha_{ij}^t \geq 0, \qquad \forall i,j,t \quad (20)$$

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 problem, which cannot be solved by any commercial software because of the huge input size (in terms of the number of clients and servers.) A heuristic solution for this problem inspired from the force-directed scheduling is presented in section V.

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, a constructive initial solution is generated. To generate the initial solution, clients are processed based on a greedy order and resources are assigned to them one by one. 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 search, which is one of the most important scheduling techniques in the high-level synthesis [46], is applied to consolidate resources, determine the active (ON) servers and further optimize the resource assignment. The pseudo code for this heuristic is shown in Figure 2.

```
Algorithm Force_directed_resource_assignment ()
// Find an initial solution
m_i = metric for ranking clients;
For (index = 1:number of clients) {
      i = argmax (M_i);
      n_i^t = metric for ranking tiers in client i;
          For (temp = 1 to |T_i|) {
              t = argmax (N_t);
              Initial_Assingmnent (i);
              Update resource availability;}
      Remove client I from m_i;}
// Update resource shares
For i = 1 to num_servers {
      (M_ij^t) =Adjust_ResourceShares(α_ij^t);}
// Force-Directed Search
Resource_Consolidate ();
P = total profit;
```

Figure 2. Pseudo code for the overall resource allocation heuristic.

B. Initial Solution

To generate the initial solution, a constructive approach is used. In this constructive approach, resource allocation for clients is done one by one. It means that clients are picked based on an order, resources are allocated to that client and resource availabilities are updated. The goal of resource allocation part in this approach is to maximize the profit that can be earned from each client considering resource availability. This process continues until all clients are processed. Details of this constructive approach are presented in the following paragraphs.

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:

$$m_i = \sum_j \sum_{t \in T_i} \left(\frac{1}{s_{ij}^{pf,t}} + \frac{q_i^t}{s_{ij}^{cf,t}} + \frac{1}{s_{ij}^{pb,t}} + \frac{1}{s_{ij}^{cb,t}}\right) \tag{21}$$

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:

$$n_i^t = \sum_j \left(\frac{1}{s_{ij}^{pf,t}} + \frac{q_i^t}{s_{ij}^{cf,t}} + \frac{1}{s_{ij}^{pb,t}} + \frac{1}{s_{ij}^{cb,t}}\right) \quad (22)$$

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.)

To generate the initial solution, Clients are picked based on the ranking metric in (21). The picked client is assigned to available servers to optimize the total profit earned from serving the client. After finding the solution, resource availabilities are updated and the next client is picked for the next assignment. The formulation below describes the profit maximization problem for a picked client (ith client).

$$Max \quad \lambda_i^{max} T_e U_i - C_p \sum_j \left[P_j^0 \sum_t (\phi_{ij}^{cf,t} + \phi_{ij}^{cb,t}) + P_j^p \sum_t (\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}) \right] T_e \quad (23)$$

Subject to:

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

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

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

with addition of constraints (10), (15), (16) and (20) with consideration of previous tier-server assignments.

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.

To eliminate the effect of fixed power consumption of servers on the complexity of the problem, we replaced the constant energy cost with a cost linearly proportional to the utilization of the server in communication domain in (23). Even with this relaxation, it can be shown that the Hessian matrix of the objective function in (23) 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 $\phi_{ij}^{c,t}$. This means that we examine different assignment parameters (α_{ij}^t) for different servers and the complete solution can be found by selecting the best set of assignment and allocation parameters.

To solve the problem with fixed α_{ij}^t 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:

$$\frac{\partial Pr}{\partial \phi_{ij,t}^{pf}} = T_e f_i^g \lambda_i^{max} \frac{\partial CDF(\lambda_i)}{\partial \lambda_i} \frac{\partial \lambda_i^{EPT}}{\partial \phi_{ij}^{pf,t}} - T_e C_p P_j^p \tag{24}$$

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:

$$\frac{\partial \lambda_i^{EPT}}{\partial \phi_{ij}^{pf,t}} = \frac{Q_i^t \alpha_{ij}^t / s_{ij}^{pf,t} \left(\mathrm{M}_{ij}^{t,1} - \Lambda_{ij}^{t,1}\right)^2}{\sum_{t \in T_i} Q_i^t \left[\left(R_i^{pf,t}\right)^2 + q_I^t \left(R_i^{cf,t}\right)^2 + \left(R_i^{pb,t}\right)^2 + \left(R_i^{cb,t}\right)^2\right]} \tag{25}$$

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 (25) 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.

The complete solution of problem (23) 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 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. To control the complexity of dynamic programming we used only up to five inactive servers from each server type and up to five active servers from each server type with least resource usage up to that point.

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 [47]) on the solution of the problem calculated applying KKT conditions. These iterations continue until the profits of two consecutive iterations are nearly the same.

The pseudo code for this step for ith client is shown in Figure 3.

```
Algorithm Initial_Assingmnent (i)
Stable = 0 and Profit = 0;
While (Stable == 0){
// Find Optimal Resource Allocation for each server type
For (t ∈ T_i){
        For (servers considered for resource assignment to tier t){
        For (α_ik^t = 1/granularity of alpha to 1)
            Find resource shares from KKT conditions ;}}
// Find the Best way to combine resources
  For (t ∈ T_i (picked based on the ranking metric)){
     X = granularity of alpha;
     Y = number of server considered for resource assignment to tier t;
     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 (y^th server and α_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 (24) for each server;
     IF (no changes from previous step)   Stable =1;}
  Else
     Stable =1;}
```

Figure 3. Pseudo code for initial solution.

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 assignment. This procedure is called *Adjust_ResourceShares*(α_{ij}^t) in the pseudo code.

C. 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 a steepest ascent method. 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 4 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 α
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 α 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 4. 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 i[th] client) is calculated according to (26) for the Gold SLA class and according to (27) for the Bronze SLA class:

$$F_{ik}^{\alpha,t} = \frac{\alpha_{ij}^{t}}{|T_i|}\lambda_i^{max} f_i^{g} CDF\left(\lambda_i^{EPT}\right) - C_p P_j^{p}(\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}) \quad (26)$$

$$F_{ik}^{\alpha,t} = -a_i^{b}\lambda_i Q_i^{t}\alpha_i\left(R_i^{pf,t} + q_i^{t}R_i^{cf,t} + R_i^{pb,t} + R_i^{cb,t}\right) - C_p P_j^{p}(\phi_{ij}^{pf,t} + \phi_{ij}^{pb,t}) \quad (27)$$

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 depart from 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.

VI. Simulation Results

In this section we present the simulation results of the proposed solution to evaluate its effectiveness. The number of server types is varied between 2 and 10. For each server type, an arbitrary, but fixed, 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 as random variables with known probability distribution functions (PDFs) to model clients with different computational and memory 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 moving in the corresponding tier graph are randomly set with an average of 80% for going forward and 20% going backward. 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 use a linear function between zero and the maximum arrival rates.

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

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

The proposed force-directed resource assignment algorithm (called FRA) is against a baseline iterative method (IM), which first fixes the resource shares and optimizes the task distribution rates, and subsequently, uses the derived distribution rates and then optimizes the resource shares. This is similar to the iterative improvement approach of [34] and [35]. We also compare our results with the upper bound solution (UB) that can be found from relaxing the capacity constraints in the problem. In particular, FRA/UB and FRA/IM columns in Table II report the quality (expected total system profit) of our solution with respect to the upper bound solution and the IM solution, respectively.

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%

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 this upper bound solution does not worry about the resource

availability in the system and only finds the best possible profit for the case of no competition among clients to reserve the resources. Also this table shows that FRA generates a better solution with respect to the IM method.

To show the effectiveness of the server consolidation technique proposed above, a solution based on the proposed initial solution with the constraint of $\alpha_{ij}^{t} \leq 0.2$ is generated, and then, the force-directed search is used to find the final solution. Trace of the execution for this setup is shown in Figure 5. Here the numbers of clients and servers are set to 50 and 250, respectively. This corresponds to the first (low server to client) ratio scenario.

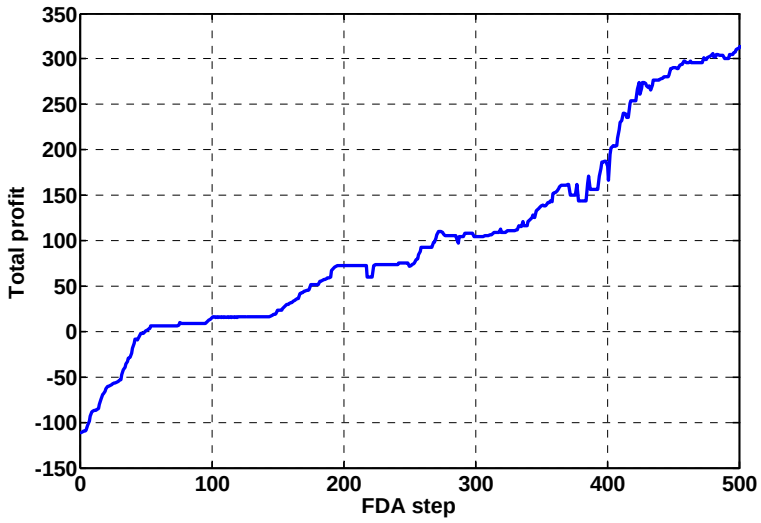

Figure 5. An example of Resource Consolidation Improvement in case of a poor initial solution.

The upper bound profit in this case is 2000. It can be seen that the resource consolidation method based on the force-directed search gradually increases the total profit. Also some decrease in the total power dissipation can be observed, which is due to the nature of the force-directed search.

Figure 6 shows the average run time of the proposed heuristic for different number of clients and servers. 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 lengths in the order of half an hour.

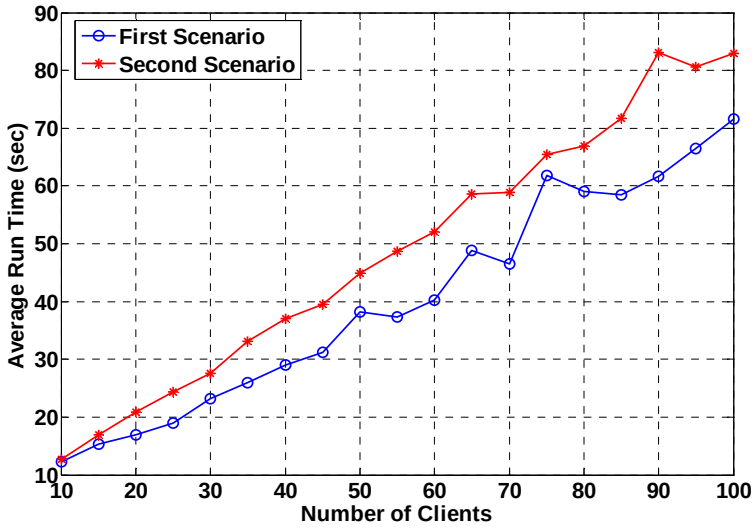

Figure 6. Average run time of FRA algorithm on 2.8GHZ E5550 server (Intel) for different number of clients.

To show the characteristic of the proposed solution, Figure 7 shows the average ratio of $\lambda_i^{EPT}/\lambda_i^{max}$ for the clients with the Gold SLA class for different ratios of f_i^g/u_i^g. As 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 due to the iterative nature of the resource allocation in the case of the Gold SLA.

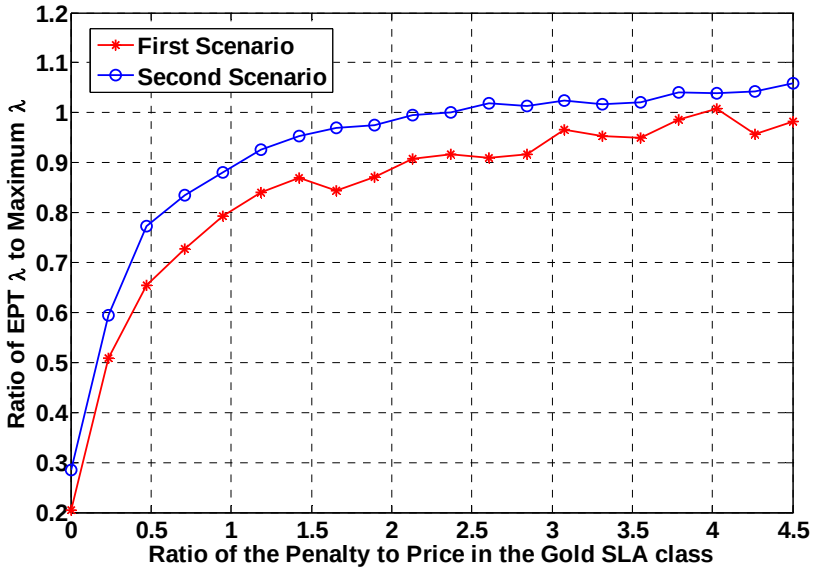

Figure 7. Ratio of EPT inter-arrival rate to maximum inter-arrival rate for different penalty values for Gold SLA class.

Figure 8 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 accounts for a bigger portion of the total energy cost in case of full utilization, which tends to result in more consolidation in servers and thus fewer ON servers.

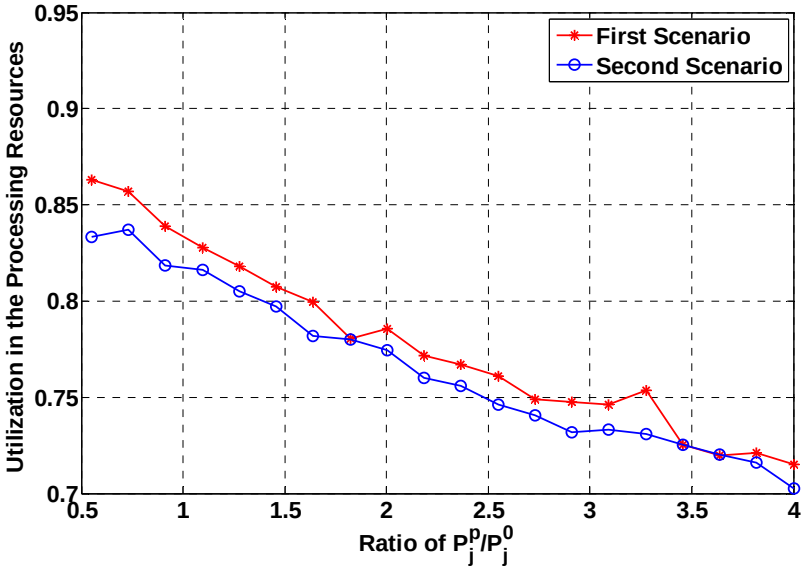

Figure 8. Utilization of the servers for different P_j^p/P_j^0 values.

VII. CONCLUSION

In this chapter, we reviewed important approaches and techniques for addressing the power and performance management problems in hosting datacenters. Then, We considered the problem of the resource allocation to optimize the total profit gained from the SLA contracts. A model based on the multi-tier applications was presented and the guarantee-based SLA was used to model the profit in the system. A solution based on generating a constructive initial solution and a resource consolidation technique based on the force-directed search was subsequently presented. The quality of the solution is compared to the upper bound solution found by relaxing the resource capacity constraints and the iterative improvement approach proposed in the previous work.

REFERENCES

[1] ENERGY STAR. 2007. Report to Congress on Server and Datacenter Energy Efficiency Public Law 109-431, Washington, D.C.: U.S.Environmental Protection Agency.

[2] The Climate Group on behalf of the Global eSustainability Initiative (GeSI), "SMART 2020: Enabling the low carbon economy in the information age," 2008.

[3] http://www.energystar.gov/index.cfm?c=prod_development.server efficiency#epa. Report to congress on server and datacenter energy efficiency. Retrieved Oct 2009.

[4] D, Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," 14th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems, 2009.

[5] S. Pelley, D. Meisner, T. F. Wenisch, and J, VanGilder, "Understanding and abstracting total datacenter power," Workshop on Energy-Efficient Design, 2009.

[6] EPA Conf. on "Enterprise Servers and Datacenters: Opportunities for Energy Efficiency," Lawrence Berkeley National Laboratory. 2006. Available at: http://hightech.lbl.gov/DCTraining/presentations.html.

[7] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield. Xen and the art of virtualization. Presented at 19th ACM Symposium on Operating Systems Principles. 2003.

[8] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia. A view of cloud computing. Commun ACM 53(4), pp. 50-58., 2010.

[9] 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.

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

[11] 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.

[12] 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), pp. 48-59. 2008.

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

[14] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang and N. Gautam. Managing server energy and operational costs in hosting centers. Presented at SIGMETRICS 2005. 2005.

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

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

[17] R. Nathuji and K. Schwan. VirtualPower: Coordinated power management in virtualized enterprise systems. Operating Systems Review 41(6), pp. 265-78. 2007.

[18] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang and Y. Chen. Greencloud: A new architecture for green datacenter. Presented at 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session, ICAC-INDST'09, June 15, 2009 - June 15. 2009.

[19] Y. Wang, X. Wang, M. Chen and X. Zhu. Power-efficient response time guarantees for virtualized enterprise servers. Presented at 2008 IEEE 29th Real-Time Systems Symposium. 2008.

[20] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan and K. Schwan. Vmanage: Loosely coupled platform and virtualization management in datacenters. Presented at 6th International Conference on Autonomic Computing, ICAC'09, June 15, 2009 - June 19. 2009.

[21] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. Presented at 2008 International Conference on Autonomic Computing (ICAC '08). 2008.

[22] E.N. Elnozahy, M. Kistler, and R. Rajamony, “Energy-Efficient Server Clusters,” Proc. 2nd Workshop Power-Aware Computing Systems, LNCS 2325, Springer, 2003.

[23] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. SIGARCH Comput. Archit. News 35, 2 (June 2007), 13-23.

[24] C. Tang, M. Steinder, M. Spreitzer and G. Pacifici. A scalable application placement controller for enterprise datacenters. Presented at 16th International World Wide Web Conference, WWW2007, May 8, 2007 - May 12. 2007.

[25] T. Kimbrel, M. Steinder, M. Sviridenko and A. Tantawi. Dynamic application placement under service and memory constraints. Presented at Proceedings. 2005.

[26] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko and A. Tantawi. Dynamic placement for clustered web applications. Presented at 15th International Conference on World Wide Web, WWW'06, May 23, 2006 - May 26. 2006.

[27] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat and R. P. Doyle. Managing energy and server resources in hosting centers. Presented at 18th ACM Symposium on Operating Systems Principles (SOSP'01), October 21, 2001 - October 24. 2002.

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

[29] I. Goiri, F. Julia, R. Nou, J. L. Berral, J. Guitart and J. Torres. Energy-aware scheduling in virtualized datacenters. Presented at 2010 IEEE International Conference on Cluster Computing (CLUSTER 2010). 2010.

[30] B. Sotomayor, R. S. Montero, I. M. Llorente and I. Foster. Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput. 13(5), pp. 14-22. 2009.

[31] M. Mazzucco, D. Dyachuk and R. Deters. Maximizing cloud providers' revenues via energy aware allocation policies. Presented at 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD 2010). 2010.

[32] F. Chang, J. Ren and R. Viswanathan. Optimal resource allocation in clouds. Presented at 3rd IEEE International Conference on Cloud Computing, CLOUD 2010, July 5, 2010 - July 10. 2010.

[33] A. Chandra, W. Gongt and P. Shenoy. Dynamic resource allocation for shared data centers using online measurements. Presented at International Conference on Measurement and Modeling of Computer Systems ACM SIGMETRICS 2003.

[34] L. Zhang and D. Ardagna. SLA based profit optimization in autonomic computing systems. Presented at ICSOC '04: Proceedings of the Second International Conference on Service Oriented Computing, November 15, 2004 - November 19. 2004.

[35] D. Ardagna, M. Trubian and L. Zhang. SLA based resource allocation policies in autonomic environments. Journal of Parallel and Distributed Computing 67(3), pp. 259-270. 2007.

[36] D. Ardagna, B. Panicucci, M. Trubian, L. Zhang, Energy-Aware Autonomic Resource Allocation in Multi-Tier Virtualized Environments. IEEE Transactions on Services Computing, vol. 99, no. PrePrints, , 2010.

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

[38] B. Addis, D. Ardagna, B. Panicucci and L. Zhang. Autonomic management of cloud service centers with availability guarantees. Presented at 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD 2010).

[39] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer and A. Tantawi. An analytical model for multi-tier internet services and its applications. Presented at SIGMETRICS 2005: International Conference on Measurement and Modeling of Computer Systems, June 6, 2005 - June 10. 2005.

[40] X. Wang, Z. Du, Y. Chen and S. Li. Virtualization-based autonomic resource management for multi-tier web applications in shared data center. J. Syst. Software 81(9), pp. 1591-608. 2008.

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

[42] G. Tesauro, N. K. Jong, R. Das and M. N. Bennani. A hybrid reinforcement learning approach to autonomic resource allocation. Presented at 3rd International Conference on Autonomic Computing. 2006.

[43] A. Gadafi, D. Hagimont, L. Broto and J. Pierson. Autonomic energy management of multi-tier clustered applications. Presented at 2009 10th IEEE/ACM International Conference on Grid Computing, Grid 2009, October 13, 2009 - October 15. 2009.

[44] M. Pedram and I. Hwang. Power and performance modeling in a virtualized server system. Presented at 2010 39th International Conference on Parallel Processing Workshops (ICPPW). 2010.

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

[46] 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.

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