RC23052 (W0401-006) January 5, 2004
Computer Science

IBM Research Report

SLA Based Profit Optimization in Web Systems

Li Zhang
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Danilo Ardagna
Politecnico di Milano
Eipartimento di Elettronica e Informazione
Via Ponzio 34/5
20133 Milano
Italy

SLA Based Profit Optimization in Web Systems

Li Zhang
IBM
T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
zhangli@us.ibm.com

Danilo Ardagna
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Via Ponzio 34/5, 20133 Milano, Italy
ardagna@elet.polimi.it

ABSTRACT

With the rapid growth of eBusiness, the Web services are becoming a commodity. To reduce the management cost for the IT infrastructure, companies often outsource their IT services to third party service providers. Large service centers have been set up to provide services to many customers by sharing the IT resources. This leads to the efficient use of resources and a reduction of the operating cost. The service provider and their customers often negotiate utility based Service Level Agreements (SLAs) to determine the cost and penalty based on the achieved performance level. The system is based on a centralized controller which can control the request volumes at various servers and the scheduling policy at each server. The controller can also decide to turn ON or OFF servers depending on the system load. This paper designs a resource allocation scheduler for such web environments so as to maximize the profits associated with multiple class SLAs. We consider a realistic utility function for the profits, which depends on the level of achieved service quality in a non-linear way. We show that the overall problem is NP-hard, and develop meta-heuristic solutions based on the tabu-search algorithm. Experimental results are presented to show the benefits of our approach.

Keywords

Resource Allocation, Quality of Service, Utility Function, SLA Optimization, Load Balancing

1. INTRODUCTION

With the rapid growth of eBusiness, the Web services are becoming a commodity. To reduce the management cost for the IT infrastructure, companies often outsource their IT services to third party service providers. Large service centers have been set up to provide services to many customers by sharing the IT resources. This leads to the efficient use of resources and a reduction of the operating cost.

The service provider and their customers often negotiate utility based Service Level Agreements (SLAs) to determine the cost and penalty based on the achieved performance level. The service provider need to manage its resource to maximize its profits. Utility based optimization approaches are commonly used for providing load balancing and obtain the optimal trade-off among job classes for Quality of Service levels. The utility functions some times are used as guidelines and provide only the trend at high level.

This paper designs a resource allocation scheduler for Web service environments. The scheduling policy is designed to maximize the revenue while balancing the cost (or energy) of using the resources. The overall profit (utility) includes the revenues and penalties incurred when Quality of Service guarantees are satisfied or violated. The revenue depends on the QoS levels in a discrete fashion. It is naturally used in the contracts between customers and the service providers. The customer would pay a per request fee for the volume of requests served. The per request fee depends on the level of performance experienced by the user. The better the achieved performance, the higher the revenue gained per request for the service provider. The system is based on a centralized controller which can establish the request volumes at various servers and the scheduling policy at each server. The controller can also decide to turn ON or OFF servers depending on the system load.

We show that the overall problem is NP-hard. We further develop meta-heuristic solutions based on the tabu-search algorithm. The neighborhood exploration is based on a fixed-point iteration, which requires solving a new network allocation flow problem. We prove structural properties of problem solutions and apply these insights to guide search algorithm. Experimental results are presented to show the benefits of our approach.

The rest of the paper is organized as follows. Section 2 introduces the overall system model. The optimization problem formulation is presented in Section 3. Section 4 analyzes the structural properties of the optimization problem. The structural properties are the basis for the search algorithms in Section 5. Experimental results in Section 6 demonstrates the quality and efficiency of our solutions.

2. THE SYSTEM

We consider the service system to be a distributed computer system consisting of M heterogeneous clusters of servers hosting N different e-commerce web sites. Each cluster is built from a number of homogeneous machines. There are totally K classes of request streams. Each class of request can be served by a collection of servers. For simplicity assume that each class of request is associated with a single web site. Let $A_{i,k}$ be the indicator function that assigns requests (and sites) to clusters: $A_{i,k}$ equals 1 if class k request can be executed by server i, 0 otherwise.

The architecture comprises of a request dispatcher in front of the clusters to assign the incoming requests to individual servers in the cluster. The controller can also establish the scheduling policy at each server. Each server has a Generalized Processor Sharing (GPS) scheduler. The allocation weights for each class can be set by the controller. The controller can also turn OFF and ON individual server inside a clusters in order to reduce the overall cost.

For each class k requests, a step-wise utility function is defined to specify the per request revenue (or penalty) incurred when the corresponding average response time assumes a given value. Fig-

WWW2004, May 17–22, 2004, New York, NY USA.
ACM xxx.xxx.

ure 1 shows, as an example, the plot of an utility function. Intuitively, customers are willing to pay a higher rate per request when their requests are served with lower response times. We observe the discontinuity in the function in Figure 1. As we will discuss in the next sections, this discontinuity in the cost function and the discrete nature of the problem make the optimization problem NP-hard. In the literature the load balancing problem with SLA profits was faced considering always continuous convex and differentiable cost functions (see for example [7],[10],[15],[11]). Considering step-wise functions of the mean response time is more intuitive from the customer point of view and are currently adopted [4].

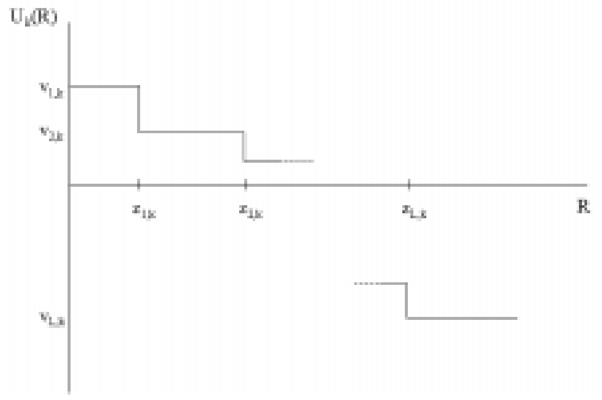

Figure 1: Utility Function

Each data center is modeled by a queueing network composed of a set of multi-class single-server queues and a multi-class infinite-server queues. The former represents the collection of servers within heterogeneous clusters. The infinite-server queues represent the client-based delays, or think times, between the server completion of one request and the arrival of the subsequent request within a Web session (see Figure 2).

User sessions begins with a class k request arriving to the data center from an exogenous source with rate λ_k. Upon completion the request either returns to the system as a class k' request with probability $p_{k,k'}$ or it completes with probability $1 - \sum_{l=1}^{K} p_{k,l}$. Let Λ_k denote the aggregate rate of arrivals for class k requests $\Lambda_k = \sum_{k'=1}^{K} \Lambda_{k'} p_{k',k} + \lambda_k$.

In next sections the following notation will be adopted:

M_i	:=	number of homogeneous server within cluster i;
y_i	:=	number of cluster i servers ON;
C_i	:=	capacity of a single server in cluster i;
$\mathcal{C}$	:=	overall computing capacity in the time scheduling period under consideration;
μ_k	:=	service rate for class k jobs at a server of capacity 1;
$\lambda_{i,k}$	:=	load at cluster i for class k jobs;
$\lambda_{i,m,k}$	:=	load at server m in cluster i for class k jobs;
$\phi_{i,m,k}$	:=	scheduling GPS parameter for class k jobs at server m within cluster i;
$R_{i,m,k}$	:=	response time for class k jobs at server m in cluster i;
$U_k(R)$	:=	utility step-wise function for class k jobs;
L	:=	number of thresholds for utility functions.
c_i	:=	cost associated with turning on a server in cluster i;

The routing matrix $[A_{i,k}]$ is used to assign private servers to individual Web site, for dedicated e-commerce transaction servers, or to limit the number of Web sites assigned to clusters for caching issues ([15]). Often the $A_{i,k}$ bounds become from the solution of long term provisioning optimization problems (the solution is evaluated for example once a week [15]). Our problem can also consider short term provisioning and the optimization algorithm is executed more frequently (several times every hour). Note that c_i, the cost associated with turning on a server in cluster i, is a function of inter-scheduler time. In practice, $c_i \propto C_i$, if power is the main cost associated with turning a server on. In the following, we will assume that the first K-1 job classes are associated with SLA, and class K is the best effort class.

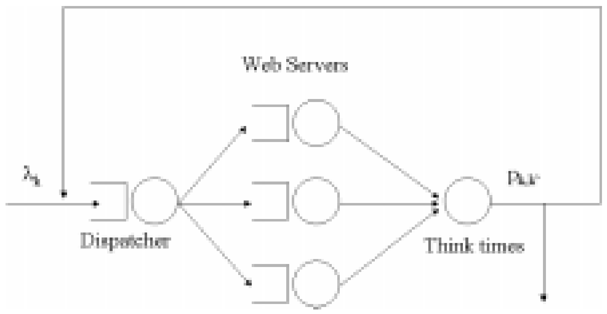

Figure 2: Network Queue Model

The analysis of multi-class queueing system is notoriously difficult. We use the GPS bounding technique in [16] to approximate the queueing system. Under GPS the server capacity devoted to class k requests at time t (if any) is $C_i \phi_{i,m,k} / \sum_{k' \in \mathcal{K}(t)} \phi_{i,m,k'}$, where $\mathcal{K}(t)$ is the set of classes with waiting requests on server m at cluster i at time t. Requests within each class on every server are executed either in a First-Come First-Serve (FCFS) or a Processor Sharing (PS) manner. Under FCFS, we assume that the service requirements for class k requests at cluster i have an exponential distribution with mean $(C_i \mu_k)^{-1}$, whereas under PS service requirements of class k requests at cluster i follow a general distribution with mean $(C_i \mu_k)^{-1}$, including heavy-tail distributions. Since many Web servers exploit the local operating system mechanisms for scheduling work within a class, the assumption of PS within each class is reasonable for a wide range of Web servers found in practice ([10]). In the approximation each multi-class single-server queue associated with server m in cluster i is decomposed into multiple single-class single-server queues with capacity greater than or equal to $C_i \phi_{i,m,k}$. The response times evaluated in the isolated per-class queues are then upper bounds on the corresponding measures in the original system.

3. OPTIMIZATION PROBLEM

Given the system model in Section 2 we formulate the cost optimization problem below. We would like to maximize the overall profit by controlling the request routing and the processor sharing scheduling policies.

$$\max \qquad \sum_{i=1}^{M} \left(\sum_{m=1}^{y_i} \sum_{k=1}^{K-1} U_k(R_{i,m,k}) \lambda_{i,m,k} - c_i y_i \right) \qquad (1)$$

$$\sum_{m=1}^{y_i} \lambda_{i,m,k} = \lambda_{i,k} \quad (2)$$

$$\sum_{i=1}^{M} \lambda_{i,k} = \Lambda_k \quad (3)$$

$$\lambda_{i,k} = 0 \qquad \text{if } A_{i,k} = 0 \quad (4)$$

$$\lambda_{i,k} \geq 0 \qquad \text{if } A_{i,k} = 1 \wedge y_i > 0 \quad (5)$$

$$\sum_{k=1}^{K-1} \phi_{i,m,k} \leq 1 \quad (6)$$

$$R_{i,m,k} = \frac{1}{C_i \mu_k \phi_{i,m,k} - \lambda_{i,m,k}}; \qquad y_i > 0; \quad (7)$$
$$\lambda_{i,m,k} < C_i \mu_k \phi_{i,m,k}$$

$$U_k(R_{i,m,k}) = \sum_{l=1}^{L} v_{l,k} \text{SAT}_{l,k}(R_{i,m,k}) \quad (8)$$

$$\text{SAT}_{l,k}(R_{i,m,k}) = 1 \Leftrightarrow z_{l-1,k} < R_{i,m,k} \leq z_{l,k};$$
$$z_0 = 0; \; z_{L+1} = \infty$$

$$\sum_{l=1}^{L} \text{SAT}_{l,k}(R_{i,m,k}) = 1 \quad (9)$$

$$y_i \in [0, M_i]; \qquad y_i \text{ integral}$$

Equation (2) entails that the traffic assigned to individual servers in a cluster equals the overall load assigned to the cluster. Equation (3) defines the overall load of class-k jobs as a function of exogenous arrivals and feed-back probability. Note that in autonomic computing systems the exogenous arrival usually is a prediction of the arrival rate for the current inter-scheduler period. Equations (4-5) assign sites requests to clusters according to $A_{i,k}$ constrains and the status of the cluster (servers ON or OFF). Finally equations (6-8) express GPS parameters scheduling bounds and utility functions in terms of response times (the condition $\lambda_{i,m,k} < C_i\mu_k\phi_{i,m,k}$ guarantees that resources are not saturated). Equation (9) guarantees that at each server, each job class is assigned to one SLA level. Here y_i, $\lambda_{i,m,k}$ and $\phi_{i,m,k}$ are decision variables and overall we have a Mixed Integer Programming problem.

In the following section, fixing some variables, the problem will be reduced to a multi-choice binary knapsack (MKP) and an NP-hard network flow resource allocation problem, so the overall problem is NP-hard. The problem is solved by implementing a tabu-search algorithm; the evaluation of the neighborhood is based on a fixed point iteration of MKP and network flow resource allocation problems. The optimization technique will be discussed in Section 5.

4. ANALYSIS OF THE PROBLEM

Let's first fix the number of servers ON in a cluster. The problem of finding the best routing and scheduling parameters $\lambda_{i,m,k}$ and $\phi_{i,m,k}$, in order to maximize revenue at a single server becomes a multiple-choice binary knapsack problem and a network flow resource allocation problem.

The Multi-choice Binary Knapsack Problem (MKP) is a variant of the classical Knapsack Problem. Let there be n groups of items. Group l has n_l items. Each item of the group has a particular value and it requires resources. The objective of the MKP is to pick exactly one item from each group for maximum total value of the collected items, subject to the resource constraint of the knapsack. In mathematical notation, let $c_{l,k}$ be the value of the k-th item in l-th group, $w_{l,k}$ the resource requirement and W the resource bound of the knapsack. Then the problem is:

$$\max \quad \sum_{l=1}^{n} \sum_{k=1}^{n_l} c_{l,k} x_{l,k}$$
$$\sum_{l=1}^{n} \sum_{k=1}^{n_l} w_{l,k} x_{l,k} \leq W$$
$$\sum_{k=1}^{n_l} x_{l,k} = 1 \qquad x_{l,k} \in \{0,1\}$$

If the number of server ON and the load at each server are fixed, then in order to maximize the objective function one can maximize revenues at single servers obtaining $\sum_{i=1}^{M} \overline{y_i}$ sub-problems:

$$\max \sum_{k=1}^{K-1} \sum_{l=1}^{L} v_{l,k} \overline{\lambda}_{i,m,k} \text{SAT}_{l,k}(R_{i,m,k}(\phi_{i,m,k}))$$

$$\text{SAT}_{l,k}(R_{i,m,k}(\phi_{i,m,k})) = 1 \Leftrightarrow$$
$$z_{l-1,k} < \frac{1}{C_i\mu_k\phi_{i,m,k} - \overline{\lambda}_{i,m,k}} \leq z_{l,k} \Leftrightarrow$$
$$\frac{1}{z_{l,k}} \leq C_i\mu_k\phi_{i,m,k} - \overline{\lambda}_{i,m,k} < \frac{1}{z_{l-1,k}}$$
$$\sum_{k=1}^{K-1} \phi_{i,m,k} \leq 1$$
$$\sum_{l=1}^{L} \text{SAT}_{l,k}(R_{i,m,k}(\phi_{i,m,k})) = 1$$

where $\phi_{i,m,k}$ are the decision variables with $i = 1,..,M$, $m = 1,..,\overline{y}_i$ and $k = 1,..,K-1$. In order to satisfy constrains and save resources for scheduling the previous inequality can be satisfied setting:

$$\text{SAT}_{l,k} = 1 \Leftrightarrow \phi_{i,m,k} = \frac{1}{z_{l,k}C_i\mu_k} + \frac{\overline{\lambda}_{i,m,k}}{C_i\mu_k}$$

which corresponds in selecting one interval of the utility function (see Figure 1) when the response time equals to the upper bound of the interval. We now have the MKP problem where parameters are defined by:

$$c_{l,k} = v_{l,k}\overline{\lambda}_{i,m,k}$$
$$w_{l,k} = \frac{1}{z_{l,k}C_i\mu_k} \quad (10)$$
$$W = 1 - \sum_{k=1}^{K-1} \frac{\overline{\lambda}_{i,m,k}}{C_i\mu_k} \quad (11)$$
$$x_{l,k} = \text{SAT}_{l,k}(R_{i,m,k}(\phi_{i,m,k}))$$

and all groups have size L, the number of thresholds of utility functions. Note that equation (11) corresponds to the system equilibrium condition defined in equation (8), while the bound that keep selecting one item from every groups in the MKP formulation corresponds to select one interval of the utility function for each SLA request job class.

Now consider a directed network consisting of nodes $\mathcal{V}$ and direct arcs $\mathcal{A}$. The arcs $a_{v_1v_2} \in \mathcal{A}$ carry flow $f_{v_1v_2}$, from node $v_1 \in \mathcal{V}$ to node $v_2 \in \mathcal{V}$. The flow is a real variable that is constrained to be bounded below by a constant $l_{v_1v_2}$ and above by a constant $u_{v_1 v_2}$. Let be a single source node $s \in \mathcal{V}$, satisfying $\sum_{a_{sv_2}} f_{sv_2} - \sum_{a_{v_1 s}} f_{v_1 s} = \mathcal{R} > 0$. This value $\mathcal{R}$, the net outflow from the source, is a constant and represents the amount of resource available to be allocated. There are n sinks node $v_2 \in \mathcal{N} \subseteq V$ which have the property that their net inflow $\sum_{a_{v_1 v_2}} f_{v_1 v_2} - \sum_{a_{v_2 v_3}} f_{v_2 v_3} > 0$. All other nodes v_2 are transshipment nodes that satisfy $\sum_{a_{v_1 v_2}} f_{v_1 v_2} - \sum_{a_{v_2 v_3}} f_{v_2 v_3} = 0$. A function $F_{v_2}(x)$ is associated with each sink node v_2 and the optimization problem is:

$$\max \sum_{v_2 \in \mathcal{N}} F_{v_2}\Big(\sum_{a_{v_1 v_2}} f_{v_1 v_2} - \sum_{a_{v_2 v_3}} f_{v_2 v_3}\Big)$$
$$\sum_{a_{sv_2}} f_{sv2} - \sum_{a_{v_1 s}} f_{v1s} = \mathcal{R}$$
$$l_{v_1 v_2} \le f_{v_1 v_2} \le u_{v_1 v_2} \quad \forall v_1, v_2 \in \mathcal{V}$$

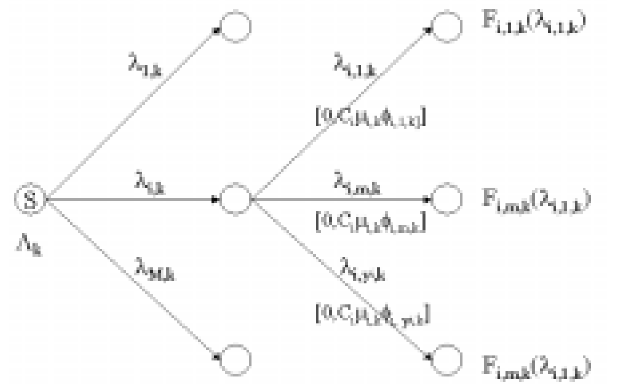
S
Λ_k
$\lambda_{1,k}$
$\lambda_{i,k}$
$\lambda_{M,k}$
$\lambda_{i,1,k}$
$[0, C_i\mu_{i,k}\phi_{i,1,k}]$
$\lambda_{i,m,k}$
$[0, C_i\mu_{i,k}\phi_{i,m,k}]$
$\lambda_{i,y,k}$
$[0, C_i\mu_{i,k}\phi_{i,y,k}]$
$F_{i,1,k}(\lambda_{i,1,k})$
$F_{i,m,k}(\lambda_{i,1,k})$
$F_{i,m,k}(\lambda_{i,1,k})$

Figure 3: Network Flow Model

If the number of server ON and the scheduling policy at each server are fixed, then in order to maximize the objective function one can establish the load at each server and solve the following $K-1$ sub-problems:

$$\begin{aligned}
\max \quad & \textstyle\sum_{l=1}^{L} v_{l,k} \lambda_{i,m,k} \mathrm{SAT}_{l,k}(R_{i,m,k}(\lambda_{i,k})) \\
& \textstyle\sum_{m=1}^{\overline{y}_i} \lambda_{i,m,k} = \lambda_{i,k} \\
& \textstyle\sum_{i=1}^{M} \lambda_{i,k} = \Lambda_k \qquad (12)\\
& \lambda_{i,k} = 0 \quad \text{if } A_{i,k} = 0 \\
& \lambda_{i,k} \ge 0 \quad \text{if } A_{i,k} = 1 \wedge \overline{y_i} > 0 \\
& \mathrm{SAT}_{l,k}(R_{i,m,k}(\lambda_{i,k})) = 1 \Leftrightarrow \\
& z_{l-1,k} < \tfrac{1}{C_i \mu_k \overline{\phi}_{i,m,k} - \lambda_{i,m,k}} \le z_{l,k} \Leftrightarrow \\
& C_i \mu_k \overline{\phi}_{i,m,k} - \tfrac{1}{z_{l-1,k}} < \lambda_{i,m,k} \le C_i \mu_k \overline{\phi}_{i,m,k} - \tfrac{1}{z_{l,k}} \\
& \textstyle\sum_{l=1}^{L} \mathrm{SAT}_{l,k}(R_{i,m,k}(\lambda_{i,m,k})) = 1
\end{aligned}$$

where $\lambda_{i,m,k}$ are the decision variables with $i = 1, .., M$, $m = 1, .., \overline{y}_i$ and $k = 1, .., K-1$. This problem is a special case of the network flow resource allocation problem where the overall flow is defined by equation (12), and the network is shown in Figure 3. A plot of sink cost functions is shown in Figure 4. The cost function is discontinue and intervals where the function is linear are defined by:

$$(f_{l-1}^m, f_l^m] = \left(C_i \mu_k \overline{\phi}_{i,m,k} - \frac{1}{z_{l-1,k}}, C_i \mu_k \overline{\phi}_{i,m,k} - \frac{1}{z_{l,k}} \right]. \quad (13)$$

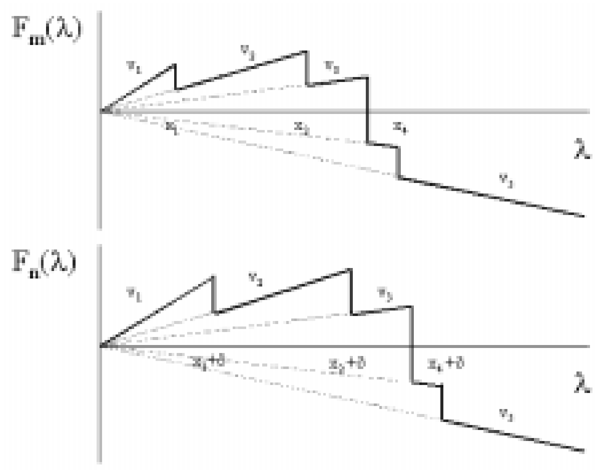

Figure 4: Sink Cost Function

The interval bounds are a function of the scheduling policy, the capacity of servers and of utility function thresholds. So different sink nodes are characterized by different cost functions, with corresponding intervals characterized by the same slope. Note also that the slopes are decreasing moving from left to right and a sink function can obtain is maximum in any interval (the maximum of a sink function depends on slopes and on widths of intervals). If we consider different cost functions these are translated by $\delta = -C_i \mu_k \overline{\phi}_{i,m,k}$ (see Figure 3). Note also that if a server has limited capacity (i.e., $C_i \mu_k \overline{\phi}_{i,m,k} - 1/z_{l,k} < 0$) some intervals may be missing.

Since sink nodes cost functions are neither convex, nor differentiable, the network flow resource allocation problem is NP-hard([9]). Nevertheless, the following properties of the optimum solution can be proved.

THEOREM 1. *In the optimum solution of a flow problem at most one server is assigned to a request rate different from a sink function upper edge interval.*

PROOF. If we relax equation (12) bound, the optimization problem has a trivial optimum that assigns to each server the load which corresponds to individual optimum of sink functions. (Note that each sink node corresponds to a server). Let's assume that in the optimum solution two servers are assigned to a load λ', λ'' which do not coincide to upper edges of corresponding intervals. Let v' and v'' be the corresponding slopes and assume $v' > v''$. Then the solution can be improved by increasing the load λ' assigned to the first server while decreasing λ'' of the second one in order to satisfy the flow equation (12) until λ' coincide with the upper edge of the interval; this contradicts the hypothesis that the solution is optimal. □

This property basically says that in the optimum solution there is only one λ *free assignment*.

THEOREM 2. *In the optimum solution of a flow problem the λ free assignment is in the interval which corresponds to the lowest level of performance.*

PROOF. Let's consider an optimum solution, let be λ', the free assignment v' the slope of the corresponding interval and assume that there is an assignment λ'' which coincide to an upper edge of an interval such that the corresponding slope v'' is greater that v'.

Then the solution can be improved increasing λ' while decreasing λ'' in order to satisfy the flow equation (12); this contradicts the hypothesis that the solution is optimal. $\square$

5. OPTIMIZATION TECHNIQUE

In the previous section we have shown that the optimization problem is NP-hard. We now provide structural properties of the problem. Based one the insights, we develop a tabu-search algorithm in order to find a quasi-optimal solution. The neighborhood exploration is based on a fixed point iteration of the MKP to determine the optimal scheduling and network flow problems to determine optimal routing. The next section will discuss the fixed point iteration procedure. Section 5.2 will describe the tabu-search algorithm implementation.

5.1 Fixed Point Iteration Procedure

The solution of the MKP problem is based on the HEU heuristic described in [2] which provides solution on average equal to 94% of the optimum. On the other hand, the literature does not provide any results for the network flow allocation problem and we have developed a local search approach in order to find a solution. The local search performs two dual moves in order to improve the initial solution:

- Increase by a unit the performance level of servers with low performance allocating their load to servers with higher performance level.
- Decrease by a unit the performance level of servers with high performance allocating their load to servers with lower performance level.

Based on the properties described in theorem 1 and 2, at each move, load of the server with higher slope is increased and the load at the server with the lowest slope is decreased in order to satisfy flow equation (12).

The local search starts increasing GPS parameters determined by the MKP solution. GPS parameters are modified in order to allow more degrees of freedom performing the two moves in the second phase. We have implemented two approaches to increase GPS parameters which derive two different fixed point iteration procedures:

- At each server GPS parameters are increased, until their sum equals to 1, fairly among job classes assigned to the server.
- At each server only one job class GPS parameter is increased (again until sum locally equals to 1). The job class is chosen by separately solving a flow problem for each SLA class and selecting the class that gives the best improvement in the solution of the flow problem.

In order to evaluate the quality of the solution, results of the two approaches are compared with results of an exhaustive search algorithm. Here we only need to compare the performance of the algorithms for a fixed set of servers. Therefore, the cost associated with servers is neglected and only revenues are considered. Results are obtained randomly generating service rates μ_k, overall class request rate Λ_k and setting $A_{i,k} = 1$. The number of thresholds varied between 5 and 9, the utility functions have fixed thresholds (proportional to the service time $1/\mu_k$ and the cost scheme is also proportional to the service time [14]. The data center utilization varies between 0.2 and 0.8. Table 5.1 shows proportional coefficients for thresholds and costs. Note that independent of the number of steps adopted, the revenue associated with the last step

Table 1: Utility Function Proportionality Coefficients

Thresholds	Costs
1.5	1000
5	250
10	150
20	100
50	50
100	30
1000	20
2000	10
5000	5
10000	-10^9

Table 2: Fixed Point Iteration Results

	Av. err%	Max err%	Av. impr%	Max impr%
FPI1	15.73%	66.63%	131.75%	423.98%
FPI2	15.40%	69.16%	134.44%	413.37%
FPI 1+2	12.21%	66.17%	145.71%	423.98%

equals -10^9, in this way we are guaranteed that the last thresholds is never violated.

Tests consider three job classes and a data center with alternatively three and four servers. The exhaustive search is very time consuming, when the load is light (0.2 utilization of the overall capacity of the data center) with 5 servers the solution of a single problem requires a day. Overall 2600 tests were run and results are almost independent by the number of thresholds of utility functions. Results are shown in Table 5.1, and are compared also with revenues of the proportional assignment scheme which employs:

$$\lambda_{i,m,k} = \Lambda_k \frac{C_i \mu_k A_{i,k}}{\sum_{l=1}^{M} C_l \mu_k A_{l,k}}$$

$$\phi_{i,m,k} = \frac{\lambda_{i,m,k}/\mu_k}{\sum_{l=1}^{K} \lambda_{i,m,l}/\mu_l}$$

Note that this proportional allocation scheme is a natural way to assign the traffic and server capacity. It is provably the best load balancing scheme in terms of stability regions and it is used as a benchmark in the SLA profits maximization literature ([10]).

Table 5.1 reports also results of a third heuristic which for each problem instance takes the maximum between fixed point iteration 1 and 2 solutions (this is the implementation that has been adopted in the tabu-search). In this way both the average error and the average improvement can be enhanced.

Exhaustive search results shows that the optimum solution is very different from the proportional assignment scheme since instead of balancing the load among servers of clusters, often favors the use of dedicated servers for job classes. This can be explained considering that dedicated servers give better performance. Let's consider as an example two different servers with the same capacity and two job classes. If the two job classes has, for the sake of simplicity, the same arrival rate λ and the same service time $1/\mu$ the proportional assignment scheme has a response time $R_1 = R_2 = 1/(0.5(\mu - \lambda))$ twice than the dedicated server scheme $R_1 = R_2 = 1/(\mu - \lambda)$.

Plot in Figure 5 shows the trace of the execution for the two fixed point iteration procedures for a system with 2 clusters, each cluster having 60 servers, and totally 3 job classes. The plot shows that the fixed point iterations converges very quickly (usually less than 30 iterations), execution time is about 3 seconds.

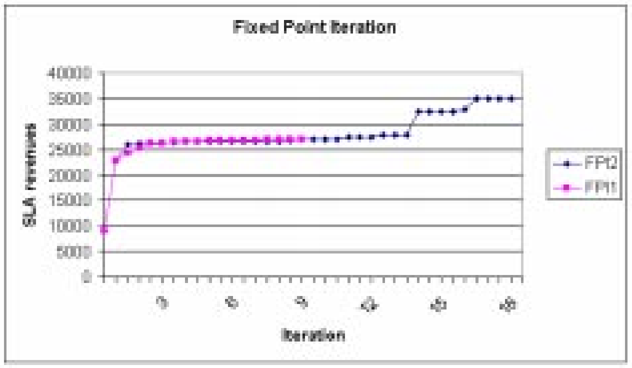

Figure 5: Fixed Point Iteration Execution Trace

5.2 Tabu Search Algorithm

The overall optimization problem (1) is solved by implementing a tabu-search algorithm. The neighborhood of the current solution is defined by two moves which increase and decrease alternatively the number of servers ON at each cluster. The evaluation of each move requires a fixed point iteration of the MKP and network flow allocation problems in order to obtain $\lambda_{i,m,k}$ $\phi_{i,m,k}$ values. The fixed point iteration discussed in the previous section converges very quickly. But we can not guarantee it converges to a global optimal solution. For this reason in the evaluation of a move the fixed point iteration is executed twice. The first execution considers as initial solution the solution obtained applying the proportional assignment schema, the second execution, vice versa, tries to take advantage of the current solution routing and scheduling policies in the following way:

- If the move turns ON a server then for the new server we apply the scheduling policy of the cluster bottleneck; the new server and the bottleneck server share equally the load. For the other servers in the cluster the routing and scheduling policy remains the same as the previous iteration.
- If the move turns OFF a server then its load is assigned to others servers in the same cluster proportionally according to servers' spare capacity, the scheduling policy is unmodified. That is the scheduling of the previous iteration is applied.

The execution that take advantage of current scheduling and routing assignment is more efficient and leads to the next current solution in almost 70% of cases. The neighborhood of a solution is defined by all solutions that can be obtained by applying these moves to all clusters. The search is guided by a tabu-search meta-heuristic in which only the short-term memory mechanism has been implemented. Since the neighborhood has almost the same size during the algorithm execution, the tabu list has a static size proportional to the number of clusters in the system.

In order to obtain good results, we faced the problem of finding a high quality initial configuration for servers at the data center. The number of servers ON y_i are the main variables of the problem, since they affect performance and cost function. On the other hand, $\lambda_{i,m,k}$ and $\phi_{i,m,k}$ affect only performance and can be considered fine tuning variables while, at high level, the performance of a Web site mainly depends on the number of servers adopted. In order to find a good initial solution for the tabu-search algorithm we have evaluated three different estimates for the number of server ON.

In the first estimate we evaluate the cost function at cluster level and enumerate job class performance level at various clusters. The

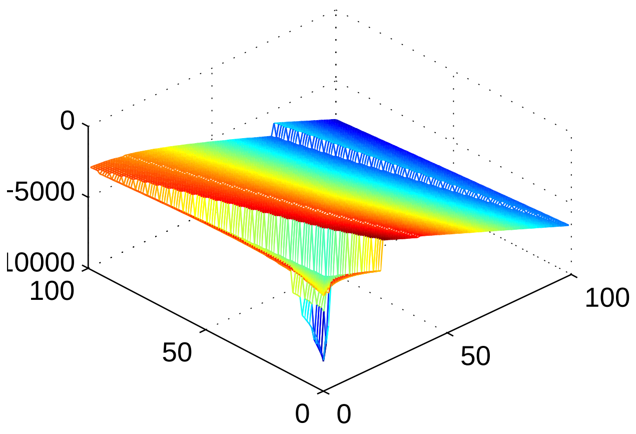

Figure 6: Plot of equation (1) as a function of the number of servers ON

cost function (1) is non-convex and non-differentiable and presents abrupt steps in correspondence of individual step of utility functions. As an example, consider the plot of Figure 6. The plot shows the trend of equation (1) as a function of the number of server ON, considering two clusters and two job classes, assigning higher priority to the most cost advantage job class and assigning requests to cluster according to their overall capacity (that is the load is assigned according to $C_i y_i/\mathcal{C}$ ratio and is equally balanced among different servers in the same cluster). The cost function initially decrease when servers are turned ON since for high load few servers can not satisfy SLA and there is the cost associated with server ON. When the number of servers is increased and SLA are satisfied the cost function jumps abruptly and the global maximum is always associated with satisfying a SLA for one set of job classes. Asymptotically when the number of server ON is increased the cost function decrease as the plane of equation $-\sum_{i=1}^{M} c_i y_i = costant$ since when SLA are satisfied at the maximum level for each class of jobs, performance are improved but increasing the number of servers is not of advantage for the Service Provider; the cost function is dominated by the cost associated with servers ON.

The estimate is obtained by considering a different cost function that evaluates SLA revenue at cluster level instead of at server level as in equation (1); let's consider the following problem:

$$\max \quad \sum_{i=1}^{M}\left(\sum_{k=1}^{K-1} U_k(R_{i,k})\lambda_{i,k} - c_i y_i\right) \tag{14}$$

$$\sum_{m=1}^{y_i} \lambda_{i,m,k} = \lambda_{i,k}$$

$$\sum_{i=1}^{M} \lambda_{i,k} = \Lambda_k \lambda_{i,k} = 0 \qquad \text{if } A_{i,k} = 0$$

$$\lambda_{i,k} \geq 0 \qquad \text{if } A_{i,k} = 1 \wedge y_i > 0$$

$$y_i \in [0, M_i]; \qquad y_i \text{integral}$$

$$\sum_{k=1}^{K-1} \phi_{i,m,k} \leq 1$$

$$R_{i,m,k} = \frac{1}{C_i \mu_k \phi_{i,m,k} - \lambda_{i,m,k}}; \qquad y_i > 0; \lambda_{i,m,k} < C_i \mu_k \phi_{i,m,k} \tag{15}$$

$$R_{i,k} = \frac{\sum_{m=1}^{y_i} \lambda_{i,m,k} R_{i,m,k}}{\lambda_{i,k}}$$

$$U_k(R_{i,k}) = \sum_{l=1}^{L} v_{l,k} \mathrm{SAT}_{l,k}(R_{i,k})$$

$$\mathrm{SAT}_{l,k}(R_{i,k}) = 1 \quad \Leftrightarrow \quad z_{l-1,k} < R_{i,k} \leq z_{l,k}; \qquad (16)$$

$$z_0 = 0; \; z_{L+1} = \infty \qquad (17)$$

$$\sum_{l=1}^{L} \mathrm{SAT}_{l,k}(R_{i,k}) = 1 \qquad (18)$$

SLAs are evaluated considering the average response time for class k jobs at cluster level, besides $\phi_{i,m,k}$ and $\lambda_{i,m,k}$ are fixed:

$$\phi_{i,m,k} = \frac{1}{K-1} \qquad (19)$$

$$\lambda_{i,k} = \Lambda_k \frac{C_i y_i}{\mathcal{C}} \qquad (20)$$

$$\lambda_{i,m,k} = \frac{\lambda_{i,k}}{y_i} \qquad (21)$$

which corresponds to adopt Processor Sharing scheduling policy (19), assign load to clusters proportionally to their capacity (20) and balancing the load in each cluster among individual servers (21). $\mathcal{C} = \sum_{i=1}^{M} C_i y_i$ is the overall computing capacity available at the data center in the scheduling time interval under consideration. Then it is easy to evaluate response times $R_{i,m,k}$ and $R_{i,k}$:

$$R_{i,m,k} = \frac{1}{\frac{C_i \mu_k}{K-1} - \frac{1}{y_i}\Lambda_k \frac{C_i y_i}{\mathcal{C}}} = \frac{(K-1)\mathcal{C}}{C_i(\mu_k \mathcal{C} - (K-1)\Lambda_k)}$$

$$R_{i,k} = \frac{\sum_{i=1}^{M} \lambda_{i,,m,k} R_{i,m,k}}{\lambda_{i,k}} = \frac{(K-1)\mathcal{C}}{C_i(\mu_k \mathcal{C} - (K-1)\Lambda_k)}$$

Note that with positions (19-21) response times depend only on the overall computing capacity of the site, that is $\mathcal{C} = \sum_{i=1}^{M} C_i y_i$, and servers in the same cluster have the same performance.

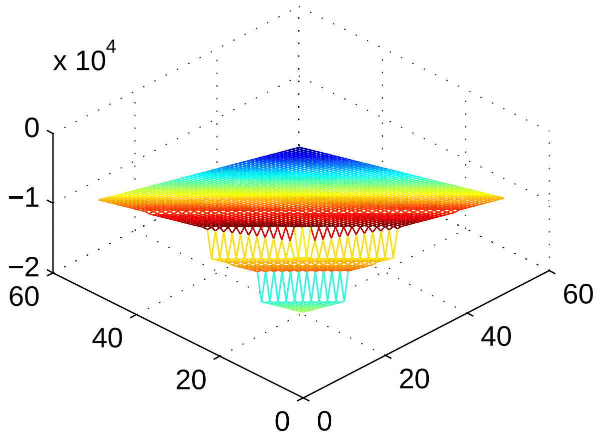

Figure 7: Plot of equation (14) with homogeneous clusters as a function of the number of servers ON

Figure 7 is an example of a plot of equation (14) as a function of the number of servers ON when only two clusters and two job classes are considered. Note that plots are smoother of the previous case but show the same behavior, that is initially decreasing, asymptotically trend a plane and the global maximum correspond to satisfying a SLA of one job class.

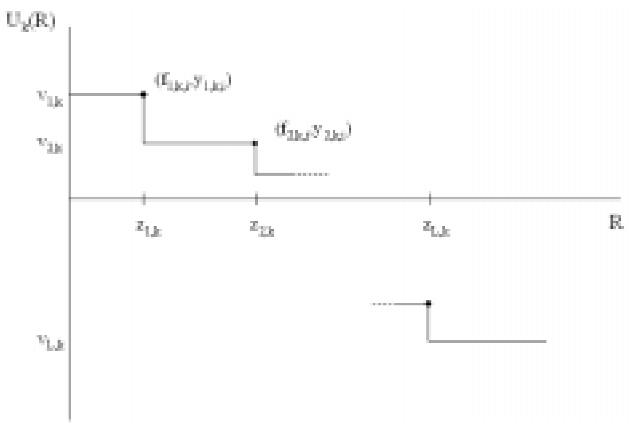

Figure 8: Utility Function, Multiple Class Dedicated Servers

In particular the plot in Figure 7 consider two homogeneous clusters and the two job classes have the same throughput and utility function. Under these assumptions the cost function behaves like a staircase and each step corresponds to a straight line in the y_1, y_2 plane. Straight line equations can be determined considering all of the possible combinations of thresholds for the two class of jobs and the two cluster that is:

$$\overline{z}_{l-1,k} < R_{i,k} \leq \overline{z}_{l,k}; \qquad i = 1, \ldots, M; k = 1, \ldots, K-1$$

It is easy to find that this set of equation gives:

$$\frac{C_i \overline{z}_{l-1,k}(K-1)\Lambda_k}{C_i \overline{z}_{l-1,k}\mu_k - K + 1} < \sum_{i=1}^{M} C_i y_i \leq \frac{C_i \overline{z}_{l,k}(K-1)\Lambda_k}{C_i \overline{z}_{l,k}\mu_k - K + 1}. \qquad (22)$$

Since the system has homogeneous servers and the two job classes have the same throughput and utility functions, the system of inequalities reduce to a single equation $\sum_{i=1}^{M} C_i y_i = const$. If clusters are heterogeneous or job classes have different utility functions or throughput then the staircase degenerates and the plot is less regular but the optimum can be anyway found considering the solution of the set of inequalities (22). So in order to evaluate the initial number of server under these hypotheses of scheduling and load $L^{M(K-1)}$ problems like the following have to be solved:

$$\min \sum_{i=1}^{M} c_i y_i \qquad (23)$$

$$\overline{z}_{l-1,k} < R_{i,k} \leq \overline{z}_{l,k}; \qquad i = 1, \ldots, M; \;\; k = 1, \ldots, K-1$$

$$y_i \in [0, M_i]; \qquad y_i \text{ integral}$$

The minimization problem originate from the fact that given response time bounds from the set of inequalities (22), the aim is to reduce the number of server. Problem (23) can be reduced to the following LPI problem:

$$\min \textstyle\sum_{i=1}^{M} c_i y_i \qquad (24)$$

$$\frac{C_i \overline{z}_{l-1,k}(K-1)\Lambda_k}{C_i \overline{z}_{l-1,k}\mu_k - K + 1} < \sum_{i=1}^{M} C_i y_i \leq \frac{C_i \overline{z}_{l,k}(K-1)\Lambda_k}{C_i \overline{z}_{l,k}\mu_k - K + 1} \qquad (25)$$

Note that in the set of constrains (25), the sum $\sum_{i=1}^{M} C_i y_i$ does not depend on k, then the set of constrains can be replaced by a single inequality taking the maximum of the left inequality and minimum of the right one. Besides, since we have a minimization problem only the left inequalities have to be satisfied, then the problem simply becomes:

$$\min \quad \sum_{i=1}^{M} c_i y_i \tag{26}$$

$$\sum_{i=1}^{M} C_i y_i \quad > \quad \max_{k=1,\ldots,K-1} \frac{C_i \overline{z}_{l-1,k}(K-1)\Lambda_k}{C_i \overline{z}_{l-1,k}\mu_k - K + 1}. \tag{27}$$

The y_i variables are bounded, for each i, $y_i \leq Y$, $Y = \max M_i$, with the change of variable $x_i = Y - y_i$ and considering that $\min f(x)$ is equivalent to $\max -f(x)$. Therefore, problem (26) becomes a knapsack problem. There are $L^{M(K-1)}$ problems originate from the combination of bounds in (25). These cases correspond to all of the possible assignments of job response times for K classes at cluster i to SLA levels. The best choice is obtained by inspecting the solutions of the $L^{M(K-1)}$ problems. Note that all of the set of combination has to be enumerated. But each combination does not necessarily require the solution of a new knapsack problem. If the number of job class is large the enumeration of knapsack problems can be intractable, in such situation the number of class can be reduced by clustering.

The second method identifies the initial solution by a greedy algorithm that assigns dedicated servers to job classes, exploiting the structure of the optimum solution discovered from exhaustive search results. The greedy procedure includes two steps. In the first step the optimum performance level assignment to job classes is identified. That is, the combination of performance level of job classes that maximizes revenues is found (if job class k is assigned to the performance level $\overline{l}$ then the revenue associated with the class is simply $v_{\overline{l},k}\Lambda_k$). The performance level assignment is identified by enumerating all of the possible performance level combinations of job classes and evaluating the overall capacity of a cluster as $C_i M_i$. This is essentially modeling a set of servers as a single server of capacity proportional to the number of servers, which implies an over-estimate of data center capacity. The second phase iteratively tries to assign job classes to dedicated servers according to the performance levels identified in the first phase. if the capacity available at the data center is not sufficient then the performance level of the class that corresponds to the minimum loss in revenue (that is the class k such that $(v_{\overline{l},k} - v_{\overline{l}+1,k})\Lambda_k$ is minimized) is decreased.

The third method finds the initial solution by assigning again dedicated servers to job classes but the assignment is identified by the solution of a multiple-choice multiple-dimension knapsack problem (MMKP). A multiple-dimension knapsack problem is one kind of knapsack where the resources are multi-dimensional, i.e. there are multiple resource constrains for the knapsack. The MKKP problem is a combination of the MKP problem presented in Section 4 and a multiple-dimension knapsack. Formally let there be n groups of items, group l has n_l items, let $c_{l,k}$ be the value of the k-th item in l-th group, $w_{l,k,i}$ the amount of resource i required by the k item in the l group and W_i the amount of the i resource. Then the problem is:

$$\max \quad \sum_{l=1}^{n}\sum_{k=1}^{n_l} c_{l,k}x_{l,k} \sum_{l=1}^{n}\sum_{k=1}^{n_l} w_{l,k,i}x_{l,k} \quad \leq \quad W_i$$

$$\sum_{k=1}^{n_l} x_{l,k} \quad = \quad 1, \qquad x_{l,k} \; \in \; \{0,1\}.$$

In this case we assume to assign to job classes dedicated servers inside each cluster which share the load assigned to the cluster. Let's denote with $y_{i,k}$ the number of servers dedicated to job class k at cluster i. Class load is assigned to cluster proportionally to their capacity, i.e.,

$$\lambda_{i,k} = \frac{C_i A_{i,k}}{\sum_{i=1}^{M} C_i A_{i,k}}\Lambda_k.$$

Now let's consider a very simple system constituted by a single class and single cluster. Let C be the capacity of servers and c the cost associated with servers in status ON. If load is balanced among servers in the clusters then the response time is given by:

$$R = \frac{1}{C\mu - \lambda/y}; \qquad \lambda < C\mu y \tag{28}$$

and the cost function becomes:

$$\max \quad U(R)\lambda - cy$$

If we consider the utility function in Figure 1, it is easy to see that the optimum values of y can be found by considering discontinuity points of the utility function since in the same interval the response time is smaller and potentially the number of server ON is greater but the overall revenue is the same. So each discontinuity point could be characterized by a couple $y_l = \lceil \frac{\lambda}{C\mu - 1/z_l} \rceil$, f_l, where the latter is the value of the cost function obtained with $y = y_l$. The optimum number of server ON could be so evaluated simply by inspection. Now consider the general system under study but assign job classes to dedicated servers. Then the response time is given by the following equation:

$$R_{i,k} = \frac{1}{C_i\mu_k - \lambda_{i,k}/y_{i,k}}; \qquad \lambda_{i,k} < C_i\mu_k y_{i,k} \tag{29}$$

Considering each job class k and cluster i then the set $\{(f_{l,k,i}, y_{l,k,i})\}$ can be determined (see Figure 8) as discussed previously. Let be $x_{l,k,i} = 1$ if class k is assigned to cluster i and the corresponding SLA is l, and $x_{l,k,i} = 0$ otherwise. Then we can consider the problem:

$$\max \sum_{i=1}^{M}\sum_{k=1}^{K-1}\sum_{l=1}^{L} f_{l,k,i}x_{l,k,i}$$

$$\sum_{k=1}^{K-1}\sum_{l=1}^{L} y_{l,k,i}x_{l,k,i} \leq M_i \tag{30}$$

$$\sum_{l=1}^{L} x_{l,k,i} = 1, \qquad x_{l,k,i} \in \{0,1\}, \tag{31}$$

where the set of constrains (30) implies that at most M_i servers in each cluster are assigned and constrains (31) assign each job class to a single cluster and to exactly one SLA level. Also this MKKP problem has been solved implementing HEU heuristic proposed in [2]. Next section will show results that can be obtained by adopting the three different initial solutions.

6. EXPERIMENTAL RESULTS

In this section we present experimental results to illustrate the effectiveness of our approach. The number of job classes has been varied between 3 and 6; the number of clusters has been varied between 2 and 10 and overall data centers with 200 servers have been considered. Service times were random generated and for each test case the load was increased in a way that the utilization of data center resources varied between 0.2 and 0.8.

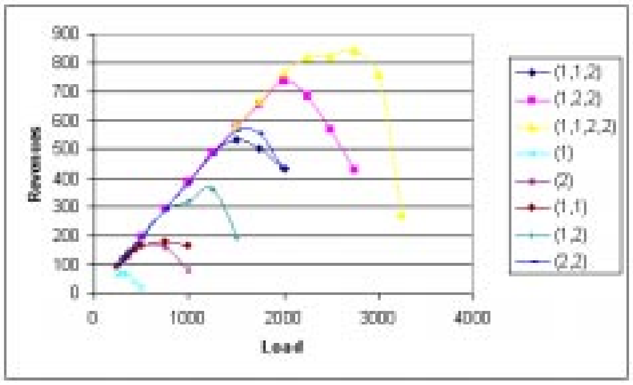

Figure 9: Revenue for Different Data Center Configurations

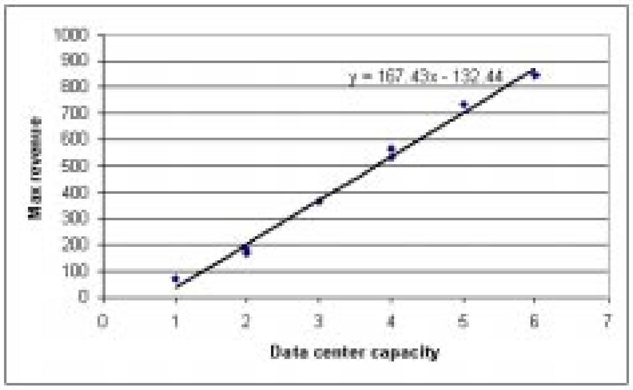

Figure 10: Maximum Revenue vs. Data Center Capacity Plot

We begin with a simple exercise to decide the cost associated with servers. The cost of a unit capacity server has been evaluated considering the revenues obtained. Plots in Figure 9 shows revenues evaluated by an exhaustive search algorithm applied on data centers with various capacities and configurations for increasing load (numbers in parenthesis specify the capacity of servers adopted). It is interesting to note that revenues increase almost linearly but start decreasing after a maximum that is obtained when the data center utilization is about 0.5-0.6. After the maximum, job classes are assigned to lower levels of performance and the increasing load implies a loss in revenues instead of a potential benefit for the Service Provider. Figure 10 shows that the maximum revenue grows linearly with data center capacity with coefficient almost equal to 160. In our tests we used 120 as unit capacity cost; sensitivity analyses showed that a $\pm 20\%$ variation of unit cost coefficient implies on average a $\pm 15\%$ variation of our results.

Plots in Figure 11 show the trace of execution of the tabu search algorithm for a system with 2 clusters. The first cluster has 60 servers with capacity 1. And the second cluster has 60 servers with capacity 2. There are 3 classes of jobs. Clusters are shared among the three job classes, i.e., $A_{i,k} = 1$. Plots show that the tabu-search approach is efficient since solutions can be improved after the analysis of worsening ones. Usually the initial solution obtained with the HEU heuristic gives better performance in terms both of initial and final solution when the load is high. When the load is light, better results can be obtained by the greedy approach. The first method proposed to determine the initial solution gives the best results only in 14% of cases. The second best solution identified by the algorithm usually differs from the best one by at most one server. Sometimes, the second best solution uses the same number of servers as the identified optimal but adopt different scheduling and routing policies.

An estimate of the quality of our solution is obtained by comparing our results with results of an exhaustive search algorithm. Tests considered data centers with two clusters shared by 3 job classes for increasing loads. In order to keep the analysis tractable a cluster with y_i servers ON was modeled as a single server with capacity $C_i y_i$. Results are quite good since with this approximation in the exhaustive search performance of a single cluster are y_i times better than y_i servers, the average error was about 30% varying between 5-70%, the error increases with the utilization of the data center since the number of servers adopted in the solution also increases and the inaccuracy of the estimation grows. In order to compare our results with the adoption of the proportional assignment schema the number of servers that has to be turned ON is evaluated as the number of servers that keeps the utilization of the data center equals to 0.6, according to greedy solutions which adopt utilization thresholds in resource allocation control as described in [7] and [1] (we empirically verified that applying proportional assignment, SLA are optimized when resource utilization is approximately 0.6). Considering this scenario our approach improves SLA revenues of one order of magnitude since for the same load our controller is able to reduce the number of servers ON, furthermore inspecting solutions we can identify islands of servers which share the load inside clusters but in general the load is not balanced among all of the servers of a cluster.

7. RELATED WORK

Recently, the problem of maximization of SLA revenues in shared data center environments has attracted vast attention by the research community. Considering the high variability of system load ([7]) the goal is to design a self-managing infrastructure that allocates shared resources to third party sites hosted at the data center, while maximizing revenues from SLA contracts. Main components of these architectures ([14]) typically are a requests classifier, which identify requests from different sites and estimates requests service times, a predictor, which from load history forecasts future system load conditions, and a controller which assigns system resources to requests trying to satisfy SLA bounds while maximizing the provider's revenue. Overviews of self-managing infrastructures can be found in [3, 7, 13, 6].

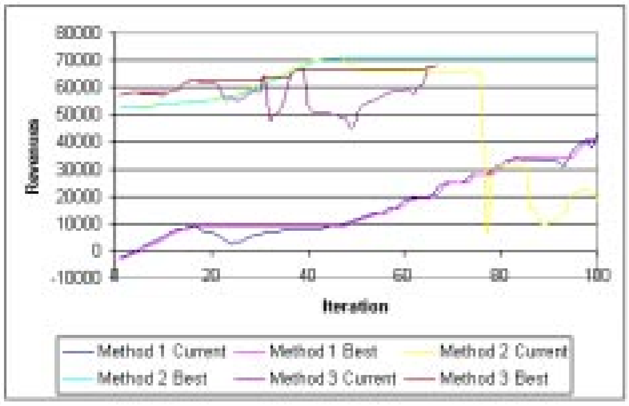

Figure 11: Execution Trace of the Tabu-search Algorithm for Different Initial Solutions

The problem of maximization of SLAs can be formulated as the dual problem of minimization of system response times and maximization of throughput as in [15]. That work proposes a static algorithm which assigns sites to overlapping servers executed once a week, on long term predictions basis, while a dynamic algorithm implements a real time dispatcher and assigns incoming requests to servers considering short term load forecasts. In [12] continuous yield functions are introduced and the problem of maximization of yield is formulated as a scheduling problem. The work proposes a greedy scheduling algorithm while incoming requests are assigned to servers according to the queue length. The effectiveness of the overall approach is verified by simulation. The authors in [6] considered the dual problem of minimizing customers' discontent function considering an online estimate of service time requirements and response times. The optimal GPS scheduling is identified by Lagrange multipliers.

In [10], the authors proposed an analytical formulation of the problem to maximize the multi-class SLA in heterogeneous web clusters considering the tail distribution of the requests response times. The problem is solved by a fixed point iteration which converges to the global optimum of the system (the cost function is concave continuous and differentiable) but the number of servers in every cluster is fixed independently by load condition. The control variables are the GPS parameters at each cluster and the frequency of requests assigned to different clusters. The load is balanced in each cluster. The problem of minimization of the cost associated with servers is considered in [7], the main costs associated with the use of resources is energy consumption and a greedy resource allocation algorithm is proposed which reconfigures cluster farms on the basis of servers utilization. Finally, in [5] is presented a study which estimates benefits of resource multiplexing of on-demand data centers environment with respect to the granularity of the control, that is the spatial allocation granularity (the resource unit allocated to customer classes, i.e. one server) and the temporal allocation granularity (i.e. the inter-scheduling time of the controller). The study proposes also an analysis which relates the temporal granularity with the accurateness of the predictor.

8. CONCLUSIONS

We proposed an allocation controller for web data center environments which maximizes the profits associated with multi-class Service Levels Agreements. The cost model consists a class of realistic utility functions which include revenues and penalties incurred depending on the achieved level of performance and the cost associated with servers. The revenue function depends on the achieved level of service as well as the request volume. The per request revenue decreases for lower level of user experienced response time. The overall optimization problem is NP-hard. We show structral properties of the optimal solution, and use these insights to guide our meta-heuristic procedure based on the tabu-search algorithm. Experimental results show that revenues that can be obtained with a proportional assignment schema can be significantly improved and we verified the quality of our solution with exhaustive search. Future work will consider the problem of maximization of SLA profits in multi-tiers systems and the model will be extended in order to include in the cost model also the tail distribution of response times.

9. REFERENCES

[1] Abdelzaher, T. F., Shin, T.,F., Bhatti, N. 2002. *Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach*. IEEE Transactions on Parallel and Distributed Systems. 13, 1, 80-96.

[2] Akbar, M. M., Manning, E.,G., Shoja, G., C., Khan, S. 2001. *Heuristic solution for the Multiple-Choice Multiple-Dimension Knapsack problem*. Conference on Computational Science, San Francisco, USA.

[3] Appleby, K., Fakhoury, S., Fong, L., Goldszmidth, G., Kalantar, M., Krishnakumar, S., Pazel, D. P., Pershing, J., Rochwerger, B. 2001. *Oceano- SLA Based Management of a Computing Utility*. In Proc. of the IFIP/IEEE Symposium on Integrated Network Management, 855-868.

[4] Boutilier, C., Das, R., Kephart, G. Tesauro, G. Walsh W. 2003. *Cooperative Negotiation in Autonomic Systems using Incremental Utility Elicitation*. To appear, Uncertainty in Artificial Intelligence.

[5] Chandra, A., Goyal, P., Shenoy, P. 2003. *Quantifying the Benefits of Resource Multiplexing in On-Demand Data Centers*. In Proc. of the First Workshop on Algorithms and Architectures for Self-Managing Systems, San Diego, CA.

[6] Chandra, A., Gong, W., Shenoy, P. 2003.*Dynamic Resource Allocation for Shared Data Centers Using Online Measurements*. In Proc. of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Poster Session.

[7] Chase , J. S., Anderson, D. C. 2001 *Managing energy and server resources in hosting centers*. In Proc. of the eighteenth ACM symposium on Operating systems principles, 103-116.

[8] Chen., X., Mohapathra, P, Chen, H. 2001. *An Admission Control Scheme for Predictable Server Response Time for Web Access*. In Proc. of WWW 2001 Conference, 545-554.

[9] Kim, D., Pardalos, P.M. 1999. *Dynamic Slope Scaling and Trust Interval Techinques for Solving Concave Piecewise Linear Network Flow Problems*. Networks 35, 3, 216-222.

[10] Liu, Z., Squillante, M. S., Wolf, J. 2001 *On maximizing service-level-agreement profits*. In Proc. of the 3rd ACM conference on Electronic Commerce, 213-223.

[11] Liu, Z., Squillante, M. S., Wolf, J. 2002. *Optimal Resource Management in e-Business Environments with Strict Quality-of-Service Performance Guarantees*. IEEE Conference on Decision and Control.

[12] Shen, K., Tang, H., Yang, T. 2002. *A Flexible QoS Framework for Cluster-based Network Services*. citeseer.nj.nec.com/485133.html.

[13] Urgaonkar, B., Shenoy, P., Roscoe, T. 2002. *Resource Overbooking and Application Profiling in Shared Hosting Platforms*. ACM SIGOPS Operating Systems Review, 36, 239-254.

[14] Verma, A., Ghosal, S. 2003. *On Admission Control for Profit Maximization of Networked Service Providers*. In Proc. of WWW 2003 Conference, 128-137.

[15] Wolf, J., Yu, P. S. 2001. *On balancing the load in a clustered web farm*. ACM Transactions on Internet Technology, 1,2, 231-261.

[16] Zhang, Z. L., Towsley, D., Kurose, J. 1995. *Statistical analysis of the generalized processor sharing scheduling discipline*. IEEE Journal on Selected Areas in Communications, 13,6, 1071-1080.