OPTIMIZING REVENUE
Service Provisioning Systems with QoS Contracts

J. Palmer, I. Mitrani, M. Mazzucco
School of Computing Science, Newcastle University, NE1 7RU, UK

P. McKee, M. Fisher
BT Group, Adastral Park, Ipswich, IP5 3RE, UK

Keywords: Quality of service, Revenue maximization, Server allocation, Admission policies, $M/M/N/K$ queue.

Abstract: We consider the problem of how best to structure and control a distributed computer system containing many processors, subject to Quality of Service contracts. Services of different types are offered, with different charges for running jobs and penalties for failing to meet the QoS requirements. The aim is to choose the number of servers allocated to each service type, and the admission criteria for jobs of that type, so as to maximize the total average revenue per unit time. The performance of a fast allocation heuristic is evaluated.

1 INTRODUCTION

The context for this work is a service provisioning system where a cluster of resources (servers) is employed to offer different services to a community of users. The immediate motivation came from the world of web services, but other multi-class hosting environments would fall in the same framework. With each service type is associated a *service level agreement* (SLA), formalizing the obligations of the users and the provider. In particular, a user agrees to pay a certain amount for each accepted and completed job, while the provider agrees to pay a penalty whenever the response time (or waiting time) of a job exceeds a certain bound. It is then the provider's responsibility to decide how to allocate the available resources, and when to accept jobs, in order to make the system as profitable as possible. That, in general terms, is the problem that we wish to address.

In order to do that, it is necessary to have a quantitative model of user demand, service provision and admission policy. We use, as a basic building block, the $M/M/N/K$ queueing model, augmented with the economic parameters of charges and penalties. The aim of this paper is to propose and evaluate efficient and easily implementable policies for resource allocation and job admission.

The approach we have adopted includes mathematical analysis, numerical solutions and also experimentation with a real hosting environment where different web services are requested and deployed.

The economic issues arising in multi-server, multi-class systems with QoS contracts based on response times or waiting times do not appear to have been studied before. Huberman et al (Huberman et al., 2005) describe a pricing structure for service provision which ensures truthful reporting of QoS by providers. However, that paper assumes that the probability of providing a particular level of QoS is fixed and known, without being specific about how QoS is measured. The distribution of response or waiting times is not considered. Rajkumar *et al* (Rajkumar et al., 1997) consider a resource allocation model for QoS management, where application needs may include timeliness, reliability, security and other application specific requirements. The model is described in terms of a *utility function* to be maximised. This model is extended in (Ghosh et al., 2003) and (Hansen et al., 2004). Such multi-dimensional QoS is beyond the scope of this paper. However, although the model described by Rajkumar *et al* allows for variation in job computation time and frequency of application requests, once again the distribution of the response times and/or waiting times is not considered.

The model assumptions are described in section 2. The revenue analysis is carried out in section 3. Numerical results and observations gathered from a working web service hosting system are presented in

Palmer J., Mitrani I., Mazzucco M., McKee P. and Fisher M. (2007).
OPTIMIZING REVENUE - Service Provisioning Systems with QoS Contracts.
In *Proceedings of the Second International Conference on e-Business*, pages 187-191
DOI: 10.5220/0002111501870191

section 4. Section 5 contains a summary and conclusions.

2 THE MODEL

The system consists of N identical servers, which may be used to serve jobs belonging to m different types. However, once allocated to a type of service, a server remains dedicated to jobs of that type only. In other words, a static and non-sharing server allocation policy is employed: n_i servers are assigned to jobs of type i ($n_1+n_2+\ldots+n_m=N$). Such a policy may deliberately take the decision to deny service to one or more job types (this will certainly happen if the number of services exceeds the number of servers).

Jobs of type i arrive according to an independent Poisson process with rate λ_i, and join a separate queue. Their required service times are distributed exponentially with mean $1/\mu_i$. An admission policy controlled by a set of thresholds is in operation: if there are K_i jobs of type i present in the system (waiting and in service), then incoming type i jobs are not accepted and are lost ($i=1,2,...,m$).

For the purposes of this model, the *quality of service* experienced by an accepted job is measured either in terms of its response time, W (the interval between the job's arrival and completion), or in terms of its waiting time, w (excluding the service time). Whichever the chosen measure, it is mentioned explicitly in a service level agreement between the provider and the users. We assume that each such contract would include the following three clauses:

1. Charge: For each accepted and completed job of type i a user shall pay a charge of c_i (in practice this may be proportional to the average length of type i jobs).
2. Obligation: The response time, W_i (or waiting time, w_i), of an accepted job of type i shall not exceed q_i.
3. Penalty: For each accepted job of type i whose response time (or waiting time) exceeds q_i, the provider shall pay to the user a penalty of r_i.

Thus, in this model, service type i is characterized by its 'demand parameters' (λ_i,μ_i), and its 'economic parameters', namely the triple

$$(c_i,q_i,r_i)=(charge,obligation,penalty) \quad (1)$$

Within the control of the provider are the server allocations, n_i, and the admission thresholds, K_i. The objective is to choose those allocations and thresholds so as to maximize the total average revenue earned per unit time in the steady state. A stationary regime always exists for a bounded queue, but if $K_i=\infty$ for some i, then the corresponding demand parameters must satisfy $\lambda_i<n_i\mu_i$ in order that the queue be stable.

Note that, although we make no assumptions about the relative magnitudes of the charge and penalty parameters, the more interesting case is where the latter is at least as large as the former: $c_i\le r_i$. Otherwise one could guarantee a positive revenue by accepting all jobs of type i, regardless of the load and of the obligation made.

3 REVENUE EVALUATION

We concentrate first on the subsystem associated with service i, for a given set of demand and economic parameters, and fixed allocation n_i and threshold K_i. That subsystem behaves like an $M/M/n_i/K_i$ queue (see, for example, (Mitrani, 1998)).

Denote by V_i the average revenue earned from type i jobs per unit time in the steady state. If the QoS measure is the response time, then V_i is given by

$$V_i=\lambda_i\sum_{j=0}^{K_i-1}p_{i,j}[c_i-r_iP(W_{i,j}>q_i)]\,, \quad (2)$$

where $p_{i,j}$ is the stationary probability that there are j jobs of type i in the $M/M/n_i/K_i$ queue, and $W_{i,j}$ is the response time of a type i job which finds, on arrival, j other type i jobs present.

If the QoS measure is the waiting time, then V_i is given by a similar expression, with $P(W_{i,j}>q_i)$ being replaced by $P(w_{i,j}>q_i)$ (where $w_{i,j}$ is the waiting time of a type i job which finds, on arrival, j other type i jobs present).

The stationary distribution of the number of type i jobs present is found by solving the balance and normalizing equations. Similarly, the probabilities $P(W_{i,j}>q_i)$ can be evaluated by computing the distribution function of a convolution of the right number of exponential distributions. This can be done in closed form.

When the computation of V_i is done for different sets of parameter values, it becomes clear that it is a unimodal function of K_i. That is, it has a single maximum, which may be at $K_i=\infty$ for lightly loaded systems. We do not have a mathematical proof of this proposition, but have verified it in numerous numerical experiments. That observation implies that one can search for the optimal admission threshold by evaluating V_i for consecutive values of K_i, stopping either when V_i starts decreasing or, if that does not happen, when the increase becomes smaller than some ε. Such searches are typically very fast.

The second problem under consideration is to maximize the total average profit per unit time, V, earned from all the m different service types:

$$V = V_1 + V_2 + ... + V_m \tag{3}$$

The server allocations vector, $(n_1, n_2, \ldots, n_m)$ (satisfying $n_1 + n_2 + + n_m = N$), and the admission thresholds vector, $(K_1, K_2, \ldots, K_m)$, must be chosen so as to maximize V.

A simple and intuitively sensible policy is to allocate the servers roughly in proportion to the offered load, $\rho_i = \lambda_i/\mu_i$, and to the service charge, c_i, for each type. In other words, set

$$n_i = \left\lfloor N \frac{\rho_i c_i}{\sum_{j=1}^{m} \rho_j c_j} + 0.5 \right\rfloor (i = 1, ..., m-1) \ ;$$

$$n_m = N - \sum_{i=1}^{m-1} n_i \tag{4}$$

(adding 0.5 and truncating is the round-off operation).

The corresponding vector of optimal admission thresholds is determined as described above, at a computational cost on the order of $O(m)$. This will be referred to as the 'heuristic' policy. Note that it may yield an allocation of $n_i = 0$ and an admission threshold $K_i = 0$ for some service types. If that is undesirable for reasons other than revenue, the allocations can be adjusted appropriately.

4 NUMERICAL AND EMPIRICAL RESULTS

Several experiments were carried out, aiming to evaluate the benefits of determining the optimal system configuration. To reduce the number of variables, the following features were held fixed:

- The QoS measure is the response time, W.
- The obligations undertaken by the provider are that jobs will complete within twice their average required service times, i.e. $q_i = 2/\mu_i$.
- All penalties are equal to the corresponding charges: $r_i = c_i$ (i.e., if the response time exceeds the obligation, users get their money back).

The first experiment concerns a 20-server system offering two services ($N = 20,\ m = 2$). The 21 possible server allocations (n_1, n_2) are evaluated and compared, for three different pairs of arrival rates. In each case, the total offered load is $\rho_1 + \rho_2 = 15.0$, which means that the 20-server system is 75% loaded.

The average service times and job charges are $1/\mu_1 = 1/\mu_2 = 1.0$ and $c_1 = c_2 = 100.0$, respectively.

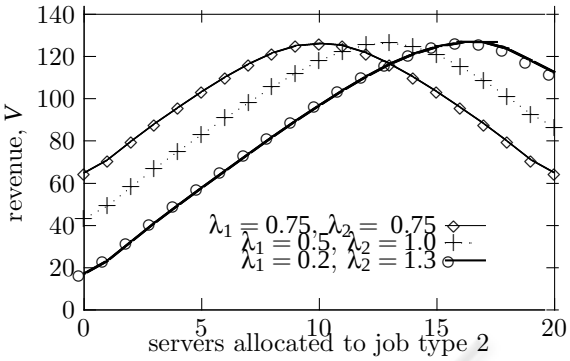

Figure 1: Maximum revenue earned for different server allocations: $N = 20, m = 2, c_i = r_i = 100, \mu_i = 0.1$.

For each server allocation, the optimal pair of admission thresholds is determined and used.

Figure 1 shows the total revenue earned, $V = V_1 + V_2$, as a function of n_2. When the two arrival rates are equal, $\lambda_1 = \lambda_2 = 7.5$, the demand is symmetric and so the optimal allocation is $n_1 = n_2 = 10$. The optimal admission thresholds are $K_1 = K_2 = 19$. For asymmetric demands, as one might expect, it is better to allocate more servers to the more heavily loaded service. Thus, if $\lambda_1 = 5$ and $\lambda_2 = 10$, the optimal allocation is $n_1 = 7,\ n_2 = 13$, with admission thresholds $K_1 = 14$, $K_2 = 24$. Lastly, when $\lambda_1 = 2$ and $\lambda_2 = 13$, the optimal allocation is $n_1 = 4,\ n_2 = 16$, with admission thresholds $K_1 = 9$, $K_2 = 28$.

It is worth noting that the optimal server allocations are quite close to those suggested by the heuristic described in the last section.

The aim of the next experiment is to evaluate the quality of the heuristic allocation policy. A 20-server system with 2 types of service was subjected to fluctuating demand controlled by a single parameter, λ. During a period of time of length 1000, jobs of type 1 and 2 arrive at rates $\lambda_1 = \lambda$ and $\lambda_2 = 10\lambda$, respectively. Then, during the next period of length 1000, the arrival rates are $\lambda_1 = 10\lambda$ and $\lambda_2 = \lambda$, respectively; and so on. The average service times for the two types are equal, $1/\mu_1 = 1/\mu_2 = 0.8$, as are the charges, $c_1 = c_2 = 100$. In addition, a third policy which uses the same server allocations as the heuristic, but does not restrict admissions (i.e., $K_1 = K_2 = \infty$), is included in the comparison. In all cases, it is assumed that, at the beginning of every new period, the demand parameters become known instantaneously, so that the server allocations and admission thresholds can be computed and applied during that period. In order to avoid the question of whether the system reaches steady state during each period, the comparisons were done by simulation.

In figure 2, the total revenue earned per unit time by the three policies is plotted against the offered load

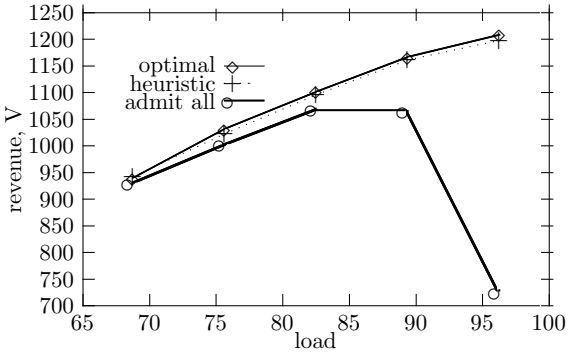

Figure 2: Policy comparisons: revenue as function of load: $N = 20, \mu_i = 0.8, c_i = r_i = 100$.

(which is equal to $11\lambda/\mu_1$). The near-optimality of the heuristic is rather remarkable. In contrast, the revenues earned by the unrestricted admission policy increase more slowly, and then drop sharply as the load becomes heavy. This example demonstrates that, by itself, a sensible server allocation is not enough; to yield good results, it should be accompanied by a sensible admission policy.

Realization of a Hosting System

A middleware platform for the deployment and use of web services was designed and implemented, with the aim of providing a real-life environment in which to study various QoS policies. The architecture is message-based and asynchronous.

Client requests for services arrive at a controller which collects statistics, estimates parameters and implements the server allocation and job admission policies. The controller makes reconfiguration decisions (e.g., server reallocations or changes of admission thresholds) at intervals of specified length. There is a handler associated with each service type, whose responsibilities include (a) deployment of the service (fetched from the code store) if not already deployed, (b) queueing of accepted jobs, if necessary, and (c) passing to the controller all necessary statistics.

An experiment similar to the one illustrated in figure 1 was carried out using a cluster of 20 computers. Two service types were deployed, and streams of requests were generated. Note that this was not an emulation of the model, but a real implementation. In the real system, messages passed between client, controller, handler and server are subject to network delays and processing overheads, which cannot be controlled. Also, it could not be guaranteed that the computers were dedicated to these tasks; there could be random demands from other users.

The three pairs of arrival rates used in this experiment were the same as in figure 1: $\lambda_1 = \lambda_2 = 0.75$; $\lambda_1 = 0.5$, $\lambda_2 = 1.0$; and $\lambda_1 = 0.2$, $\lambda_2 = 1.3$. The average service times were also the same: $1/\mu_1 = 1/\mu_2 = 10$. However, because of the factors mentioned above, one should not expect a precise match between the numerical predictions and the observations of the real system. In figure 3, the total revenues earned are plotted against n_2, the number of servers allocated to service 2. Each point in the figure corresponds to a separate run of the system, during which about 1000 jobs of each type arrived and were completed. These plots have the same general characteristics as the ones in figure 1. The maximum achievable revenue is again about 120 per unit time, and the server allocations that achieve it are the same, with one exception. In the case of $\lambda_1 = 0.5$, $\lambda_2 = 1.0$, the real system earned its highest revenue for $n_1 = 10,\ n_2 = 10$, whereas the numerical calculations suggested $n_1 = 7,\ n_2 = 13$. However, the differences in revenues are not large.

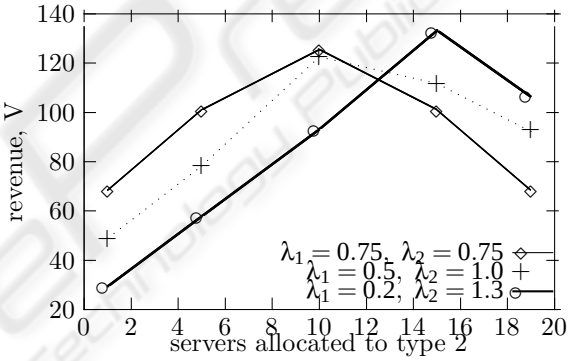

Figure 3: Observed revenues for different server allocations: $N = 20, m = 2, c_i = r_i = 100$.

5 CONCLUSIONS

The contribution of this paper is to introduce quantitative methods to a previously unexplored area, namely the market in computer services. We have demonstrated that policy decisions such as server allocations and admission thresholds can have a significant effect on the revenue earned. Moreover, those decisions are affected by the contractual obligations between clients and provider in relation to quality of service.

When the numbers of services offered and servers available are too large for an exhaustive search of the optimal policy, a simple heuristic is proposed. Experimentation suggests that it is close to optimal.

ACKNOWLEDGEMENTS

This work was carried out as part of the research project QOSP (Quality Of Service Provisioning),

funded by British Telecom. It was also supported by the European Union Network of Excellence EuroNGI (Next Generation Internet).

REFERENCES

Ghosh, S. et al. (2003). Scalable resource allocation for multi-processor qos optimization. In *The International Conference on Distributed Computing Systems*.

Hansen, J. et al. (2004). Resource management of highly configurable tasks. In *IPDPS '04, The 18th International Parallel and Distributed Processing Symposium*, pages 116–123.

Huberman, B. A. et al. (2005). Ensuring trust in one time exchanges: solving the qos problem. *Netnomics*, 7:27–37.

Mitrani, I. (1998). *Probabilistic Modelling*. Cambridge University Press.

Rajkumar, R. et al. (1997). A resource allocation model for qos management. In *RTSS '97, The 18th IEEE Real-Time Systems Symposium*, pages 298–307.