Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 296 (2013) 199–210

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Energy-aware Management of Customer Streams

K. Aidarov[1]

National University Kazakhstan

P. Ezhilchelvan[2]

Newcastle University

I. Mitrani[3]

Newcastle University

Abstract

Customers submit streams of jobs of different types for execution at a service center. The number of jobs in each stream and the rate of their submission are specified. A service level agreement indicates the charge paid by the customer, the quality of service promised by the provider and the penalty to be paid by the latter if the QoS requirement is not met. To save energy, servers may be powered up and down dynamically. The objective is to maximize the revenues received while minimizing the penalties paid and the energy consumption costs of the servers used. To that end, heuristic policies are proposed for making decisions about stream admissions and server activation and deactivation. Those policies are motivated by queueing models. The results of several simulation experiments are described.

Keywords: QoS requirements, quality of service

1 Introduction

This paper addresses an optimization problem arising in the market for computer services. A service provider employs a cluster of servers in order to offer a number of different services to a community of users. The users pay for having their jobs run, but demand in turn a certain quality of service. More precisely, a user wishes to submit a specified number of jobs of a given type, at a specified rate (jobs per

[1] E-mail: kanat.aydarov@kaznu.kz

[2] E-mail: paul.ezhilchelvan@ncl.ac.uk

[3] E-mail: isi.mitrani@ncl.ac.uk

http://dx.doi.org/10.1016/j.entcs.2013.07.013

second); such a collection is referred to as a 'stream'. In some applications, the jobs within a stream are generated and submitted automatically, e.g. by sensors detecting events of particular type.

There is a charge for running a stream (which may depend on the type), and a QoS guarantee on the part of the provider: the average waiting time of all jobs in the stream will not exceed a given bound. If that obligation is not met, the provider pays a specified penalty to the user.

In addition, there is an energy consumption cost associated with each running server. The provider thus faces a difficult trade-off: energy costs are minimized by keeping the number of servers powered up as low as possible, while revenues are maximized by using as many servers as possible in order to be able to accept incoming streams and avoid paying penalties. It is therefore desirable to employ dynamic policies for deciding when to power servers on and off and whether to admit incoming streams or not. Moreover, those policies should react appropriately to changes in demand. Note that even when there is sufficient service capacity to serve a stream, it may be advisable to reject it if the likelihood of paying a large penalty is high enough.

We design and evaluate stream admission and server allocation heuristics that aim to maximize the average profit obtained (revenues minus costs) per unit time. They are based on queueing models of system behaviour. Under reasonably realistic assumptions, those models are intractable and it is necessary to use approximations. That approach is justifiable because the policies that emerge yield significantly larger profits than the default policy of 'keep all available servers powered up and admit all streams'. The proposed heuristics are readily implementable and can be used in real systems.

There is an extensive literature on both server allocation and energy-saving topics. However, the particular problem considered here does not appear to have been studied before. Perhaps the most closely related work is by Mazzucco et al [9,10,11] and Mitrani [13]. The concepts of charge, obligation and penalty were defined in [9] and were applied to individual jobs (rather than to streams). The notion of a user stream was introduced in [10]. That paper examined the allocation of servers between different types of users, but did not consider the costs of providing the servers and the desirability of dynamically powering them up and down. The latter aspects were studied in [13] and [11], without addressing the economics of user streams and the admission policies associated with them.

More distantly related are works by Chase et al [3], Villela et al [14], Levy et al [7], and Liu et al [8], who consider various server allocation policies. Chandra et al [4], Kanodia and Knightly [6], Bennani and Menascé [2] and Chen et al [5] examine certain aspects of resource allocation and admission control in systems where the QoS criterion is related to waiting or response time. Those studies do not consider the issues associated with dynamic policies and profit maximization.

The system model and the associated service-level agreements are described in section 2. The mathematical analysis and the resulting heuristic policies for stream admissions and server activation/deactivation and are presented in section

3. Some simulation experiments where the heuristics are evaluated and compared to the default policy are reported in section 4. A summary and comments on future research directions are given in the conclusion.

2 The model

The provider has a cluster of N servers which can be used to serve user jobs. A user request is referred to as a 'stream'; it consists of a number of jobs, submitted at a given rate over a period of time. These streams may be of m different types, with different demand characteristics. More precisely, a stream of type i ($i = 1, 2, ..., m$) consists of k_i jobs, submitted at the rate λ_i jobs per second. If a stream is accepted, all jobs in it will be executed. The service times of type i jobs are i.i.d. random variables with mean and second moment b_i and $M_{2,i}$ respectively.

Streams of type i arrive at the rate of γ_i (the arrival instant of a stream is the moment when the first of its jobs is submitted). Hence, if there is no admissions policy and all incoming streams are accepted, the total offered load would be equal to

$$\rho = \sum_{i=1}^{m} \gamma_i k_i b_i \ . \tag{1}$$

In that case, the system would be stable if $\rho < N$. Of course, the presence of an admissions policy invalidates that requirement.

The *quality of service* experienced by an accepted stream of type i is measured by the observed average waiting time, W_i:

$$W_i = \frac{1}{k_i} \sum_{j=1}^{k_i} w_j \ , \tag{2}$$

where w_j is the waiting time of the jth job in the stream (the interval between its submission and the start of its service). One could also decide to measure the quality of service by the observed average response time, taking the job lengths into account.

N. B. It is worth emphasizing that the right-hand side of (2) is a random variable; its value depends on every job that belongs to the stream. Hence, even if all interarrival and service times are distributed exponentially, one would have to include quite a lot of past history into the state descriptor in order to make the process Markov. This remark explains why some of the approximations that follow are really unavoidable.

Each service-level agreement includes the following three clauses:

- Charge: For each accepted stream of type i, the user shall pay a charge of r_i (this would normally depend on the number of jobs in the stream, k_i, their average service time b_i and submission rate, λ_i).
- Obligation: The observed average waiting time, W_i, of an accepted stream of type i shall not exceed q_i.

- Penalty: For each accepted stream of type i whose W_i exceeds q_i, the provider shall pay to the user a penalty of p_i.

So, in addition to their traffic characteristics, streams of type i have 'economic parameters', namely the triple (r_i, q_i, p_i), $i = 1, 2, \ldots, m$.

All jobs submitted by all active streams join a common FIFO queue and can be served by any of the currently operative servers. Each of the latter incurs a cost of c per unit time. The provider must make dynamic decisions about whether or not to accept incoming streams and how many servers to employ. More precisely, if at a stream arrival instant there are n operative servers, the following actions may be considered: reject the new stream; accept the new stream but do not power up any new servers; accept the new stream and power up one new server; . . .; accept the new stream and power up $N - n$ new servers.

These possibilities will be labeled -1, 0, 1, . . ., $N - n$, respectively.

If, at a stream completion instant, there are n operative servers, the possible actions are: leave the servers as they are; power down one server; power down two servers; . . .; power down n servers.

These possibilities are labeled 0, 1, . . ., n, respectively.

The times taken to power servers up or down are assumed to be small compared to the lifetime of a stream and will be neglected. Also, it is assumed that the intervals between consecutive policy decision instants are large compared to individual job interarrival and service times. That is, enough jobs arrive and are served during such an interval to enable the system to be treated as having reached steady state.

The performance of the system is measured by the average profit, R, received per unit time. This is given by

$$R = \sum_{i=1}^{m} a_i [r_i - p_i P(W_i > q_i)] - cS \ , \tag{3}$$

where a_i is the average number of type i streams that are accepted into the system per unit time; $P(W_i > q_i)$ is the probability that the observed average waiting time of a type i stream, (2), exceeds the obligation q_i; c is the cost of running one server for one unit of time; S is the average number of servers that are powered on. The objective of the management policy is to maximize the value of R.

3 Policies

Consider the system state when a stream of type i arrives. Suppose that L_j streams of type j are currently active ($j = 1, 2, ..., m$), and n servers are operative. If decision -1 is taken (i.e., the new stream is rejected), then nothing changes to affect the system's future, so that decision can be assigned value 0.

On the other hand, if decision s is taken, for $s \geq 0$ (i.e., the new stream is accepted and s new servers are powered up, $s = 0, 1, ..., N - n$), then the following changes will occur: (a) L_i will increase by 1; (b) a revenue of r_i will be received; (c) a potential penalty of p_i may be payable; (d) the new servers will incur running costs. (The new stream may also influence the likelihoods of paying penalties for the existing active streams; this is a second-order effect which will be ignored.)

The value of (d) can be estimated by remarking that, in the absence of long waiting times, the lifetime of a stream of type i is roughly k_i/λ_i. Hence, the running costs of the new servers are approximately sck_i/λ_i.

To assess the value of (c), we need to estimate the probability that the average waiting time of the jobs in the new stream will exceed the obligation q_i, given that there will be $n+s$ operative servers. To do that, we model the system as a $GI/G/n+s$ queue with arrival rate, average service time and offered load given by

$$\lambda = \sum_{j=1}^{m} L_j \lambda_j \ \ , \quad b = \frac{1}{\lambda} \sum_{j=1}^{m} L_j \lambda_j b_j \ \ , \quad \rho = \lambda b \tag{4}$$

(remember that L_i has increased by 1). Note that the arrival rate and offered load appearing in (4) depend on the currently active streams. This is different from the long-term offered load in (1).

Although there is no exact solution for the average waiting time, β, in the $GI/G/n+s$ queue, an acceptable approximation is provided by an appropriate scaling of the corresponding $M/M/n+s$ result (see Whitt, [15]):

$$\beta = \frac{ca^2 + cs^2}{2} w_{M/M/n+s} \ , \tag{5}$$

where $w_{M/M/n+s}$ is the average waiting time in the Markovian $M/M/n+s$ queue with the above parameters, and ca^2 and cs^2 are the squared coefficients of variation of the interarrival intervals and service times, respectively. We shall approximate cs^2 by saying that a job starting service is of type i with probability λ_i/λ and averaging over the job types:

$$cs^2 = \frac{1}{\lambda b^2} \sum_{i=1}^{m} L_i \lambda_i M_{2,i} - 1 \ . \tag{6}$$

The value of ca^2 will be taken as 1 (i.e., assume that the arrival processes of both streams and jobs within a stream are reasonably close to Poisson). That assumption is not essential, but if it is not made, some mechanism of estimating ca^2 would have to be provided.

The average waiting time in the $M/M/n+s$ queue is given by the well-known Erlang-C formula (or Erlang delay formula, e.g., see [12]). The appropriate expressions are as follows:

$$w_{M/M/n+s} = \frac{b}{n+s-\rho} P(J \geq n+s) \ , \tag{7}$$

where J is the number of jobs present, and so $P(J \geq n+s)$ is the probability that an incoming job will have to wait. That probability is given by

$$P(J \geq n+s) = \frac{(n+s)\rho^{n+s}}{(n+s)!(n+s-\rho)} p_0 \,, \tag{8}$$

with p_0 being the probability of an empty system:

$$p_0 = \left[\frac{(n+s)\rho^{n+s}}{(n+s)!(n+s-\rho)} + \sum_{j=0}^{n+s-1} \frac{\rho^j}{j!} \right]^{-1} . \tag{9}$$

When the system is heavily loaded, the waiting time in the $GI/G/n_i$ queue is approximately exponentially distributed (see [15]). Since the variance of the exponential distribution is equal to the mean, the waiting time variance can also be approximated by (5). Hence, the observed average waiting time of a stream of type i, which according to (2) involves the sum of k_i waiting times, can be treated as being approximately normally distributed with mean β and variance β/k_i. That approximation appeals to the central limit theorem and ignores the dependencies between individual waiting times.

Based on the normal approximation, the probability that the observed average waiting time, W_i, exceeds the obligation, q_i, can be estimated as

$$P(W_i > q_i) = 1 - \Phi\left(\frac{q_i - \beta}{\sqrt{\frac{\beta}{k_i}}} \right) , \tag{10}$$

where β is given by (5), and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution (mean 0 and variance 1). That function can be computed very accurately by means of a rational approximation (see [1]).

If $\rho \geq n+s$ (violating the stability condition), then it is natural to set $\beta = \infty$ and $P(W_i > q_i) = 1$.

The quality of the approximation (10) will depend on how well the implied assumptions are satisfied, namely the load is heavy and there is a large number of jobs per stream (the second of these conditions also ensures that any dependencies between the waiting times within a stream can be neglected). On the other hand, if the system is lightly loaded, then it is not so important to come up with a clever admission policy; all incoming streams would be admitted.

Thus, the value of decision s, $v(s)$, can be assessed as

$$v(s) = r_i - p_i P(W_i > q_i) - \frac{sck_i}{\lambda_i} . \tag{11}$$

The policy we propose to apply at arrival instances is to evaluate $v(s)$ for $s = 0, 1, \ldots, N-n$, and if any of those values are positive, choose the decision that yields the largest value. If all are negative, choose decision -1. That policy will be referred to as the ‘Current State’ heuristic.

When a stream completes, a sensible policy is to power down as many servers as were powered up when that stream arrived.

The performance of the Current State heuristic will be compared against the 'default' policy which keeps all servers permanently powered up and accepts all incoming streams.

Although the computations involved in applying the Current State heuristic are by no means excessive, in some circumstances it may be desirable to avoid them. Therefore, we include in the comparisons a 'Simple' heuristic which ignores the possible penalties and just ensures that the number of active servers exceeds the current total offered load (including that of the new stream).

More precisely, the Simple heuristic works as follows: compute the new offered load ρ according to 4, with L_i incremented by 1. Accept the incoming stream if there is a non-negative integer s such that $N \geq n+s > \rho$; the number of new servers powered up is equal to the smallest such s. If that is impossible, i.e. if $N \leq \rho$, then the stream is rejected.

4 Simulation results

To compare the performance of the Current State, Simple and Default policies, a number of simulation experiments were carried out. In all cases, the profits achieved by the three policies in a system where the total number of servers is $N = 40$, are plotted against the stream arrival rate. Each point in the graphs corresponds to a simulation run where more than a million jobs arrive and are served. Each run is divided into 10 portions and the observations corresponding to those portions form a sample for the purpose of computing confidence intervals.

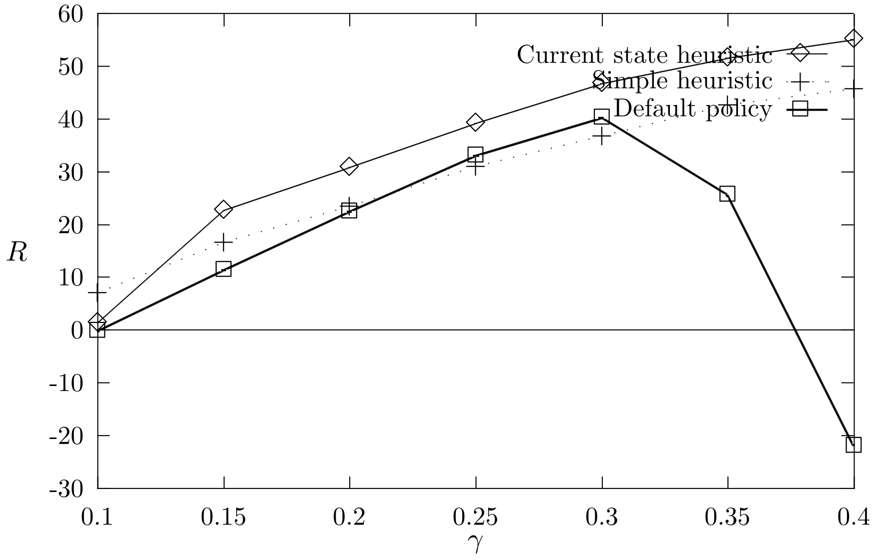

Fig. 1. Policy comparison: single stream type, high penalties

In the first experiment, all streams are of the same type. Each consists of 100 jobs, arriving at the rate of $\lambda = 0.9$ and requiring an average service time of $b = 1$, with squared coefficient of variation $cs^2 = 1$. The power consumption cost for one server is $c = 0.5$ per second. The revenue and penalty per stream are $r = p = 200$, while the obligation is $q = 1$ (in other words, if the average waiting time over the 100 jobs is larger than the average service time, the user will get his money back).

Note that, if all streams are accepted, the average offered load would be $100\gamma b$, so the system would saturate at $\gamma = 40/100 = 0.4$. We vary γ in the range (0.1,0.4).

Figure 1 shows that, in this case, the Current State heuristic produces consistently higher profits than the other policies. The simple heuristic also performs quite well, roughly keeping up with the Current State heuristic but yielding approximately 20% lower profits. The important point to note is that the profits of both these heuristics grow steadily with the offered load. On the other hand, the default policy yields negative profits at very low and very high loads. This is because in those situations the costs of running the servers and paying penalties are greater than the revenues received.

The confidence intervals for all three policies are largest at low traffic rates, where fewer jobs go through the system. However, for the vast majority of points, the half-width of the 90% confidence interval is less than 10% of the observed mean value; in many cases it is less 5%. These remarks apply to all subsequent figures too.

Intuitively, reducing the penalty for not meeting the obligation should reduce the importance of making good policy decisions. This is indeed the case, as is shown in figure 2. The set-up here is the same as in figure 1, except that now the penalty is half as large as the revenue: $p = r/2 = 100$.

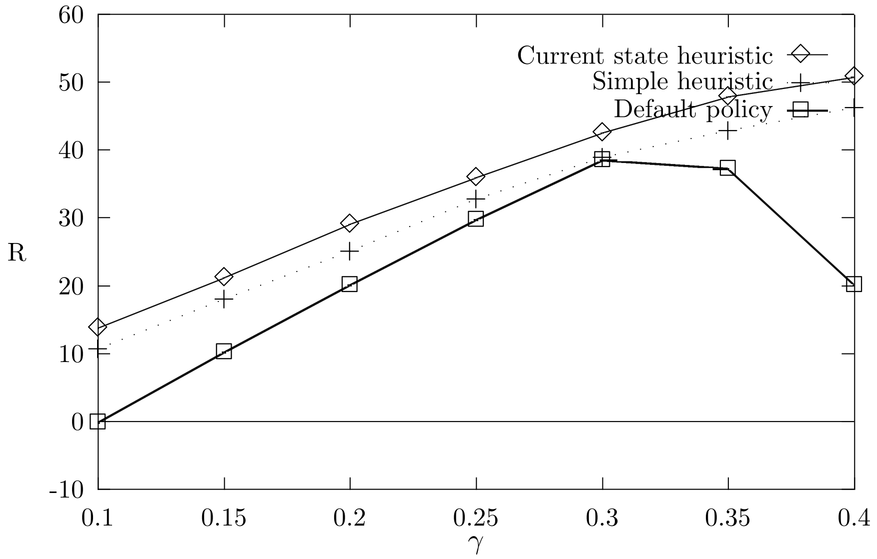

Fig. 2. Policy comparison: single stream type, low penalties

We observe that the performance of the simple heuristic is closer to that of the current state heuristic. The default policy also performs better than before, producing non-negative profits throughout

In the next experiment, the same 40 servers are available to serve two stream types, with the following parameters:

	Type 1	Type 2
k	100	100
λ	0.9	0.4
b	1	4
r	200	600
q	1	4
p	400	1200

Thus, jobs of type 2 are longer but are submitted less frequently than those of type 1. In both cases, the waiting time obligation is equal to the average service time. Type 1 streams pay 3 times higher charges; this time both penalties are twice as large as the corresponding charges. The running cost for one server is again $c = 0.5$ per second.

The squared coefficient of variation, cs^2, is now computed according to (6), with $M_{2,1} = 2b_1^2$ and $M_{2,2} = 2b_2^2$.

The arrival rates of both stream types are increased in a fixed proportion: 65% of all incoming streams are of type 1 and 35% are of type 2. If the overall stream arrival rate is γ, the offered load is $65\gamma + 35 * 4\gamma = 205\gamma$. Hence, if all streams are accepted, the saturation point would be $\gamma = 40/205$.

Figure 3 compares the profits of the two heuristics and the default policy when γ is varied in the range (0.025,0.2).

Again we observe that both the Current State and the simple heuristic yield profits that increase with the offered load, while the default policy can produce highly negative profits at low and high loads. The profits of the simple heuristic are within 15% or less of those of Current State.

As was the case with a single stream type, reducing the penalties should favour the simple heuristic and the default policy. This is illustrated in figure 4, where all other parameters are the same as in figure 3, but the penalties for both stream types are half as large as the respective revenues: $p_1 = r_1/2 = 100$; $p_2 = r_2/2 = 300$. The simple heuristic now performs as well as the current state heuristic (even outperforms it slightly in one case, although the difference is not statistically significant). The default policy also performs significantly better than before as the load increases.

Finally, let us put our heuristics to a more stringent test. We consider a rather extreme example with a single stream type. The server costs are twice as high as those in figure 1. They are now equal to $c = 1$, so that the cost of one server employed during the lifetime of a stream is just under half of the revenue obtained

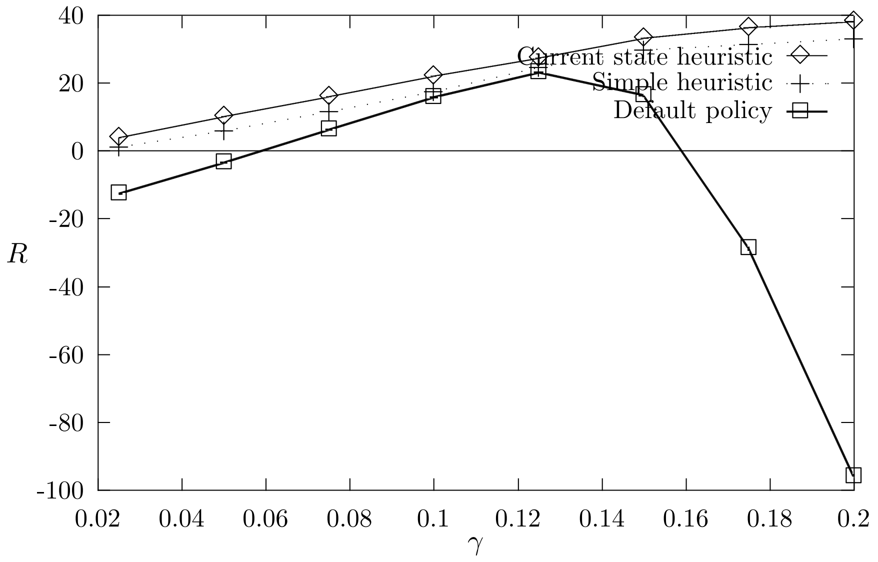

Fig. 3. Policy comparison: two stream types; high penalties

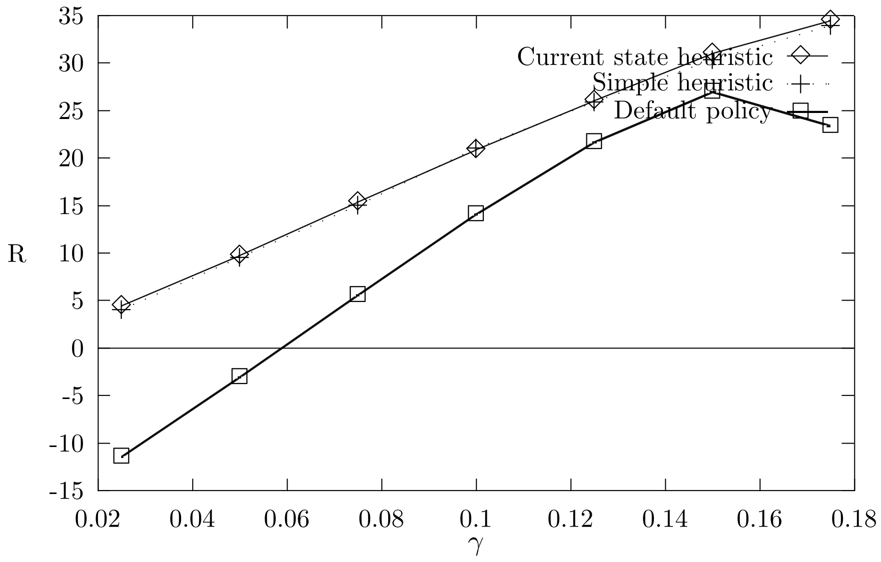

Fig. 4. Policy comparison: two stream types; low penalties

from that stream. The penalty for failing to meet the obligation is now four times as high as the revenue: $p = 4r = 800$. The other parameters are as in figure 1. The aim of this experiment is not to examine a realistic set-up, but to see how the heuristics behave in an unforgiving environment where making the wrong decisions on stream admissions and server allocations is very costly.

The results are shown in figure 5.

We see that the Current State heuristic copes well in these extreme circumstances and achieves similar profits to those in figure 1. This is quite encouraging. On the other hand, the simple heuristic is not really adequate. For most of the traffic range,

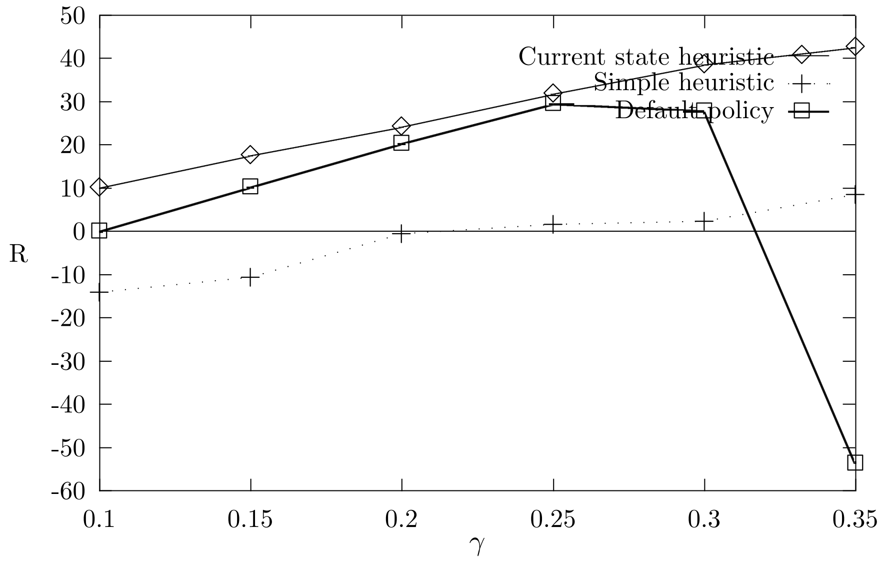

Fig. 5. Policy comparison: very high server costs and penalties

it would be better to use the default policy than the simple heuristic. Only at heavy loads is the situation reversed.

5 Conclusions

We have examined the problem of maximizing profits in a service provisioning environment where demand comes in the form of customer streams and where QoS requirements and energy costs are conflicting factors. An approximate queueing analysis has enabled us to propose easily implementable heuristic policies for making intelligent decisions about stream admissions and server activations and deactivations. The Current State heuristic, in particular, yields significantly higher profits than the default policy of keeping all servers powered on and accepting all incoming streams. The simple heuristic is much easier to implement and also performs well, except in extreme circumstances when both the penalties and the server costs are very high.

It is clearly necessary to carry out a more extensive programme of experimentation, exploring the behaviour of the heuristics under different demand conditions and economic regimes. It would also be desirable to implement these heuristics in a functioning system and observe their behaviour under real-life offered loads. In addition, one may wish to consider the unit costs of powering servers up and down.

In practice, powering a server on and off is not instantaneous but takes an interval of time during which the server cannot serve jobs. Introducing such delays into the queueing model would complicate the solution considerably, if not render it intractable. However, the simulation could be adapted to incorporate these set-up times without too much difficulty.

A different operational model might dedicate servers to streams of a particular type. Then there would be a separate queue for each stream type. As well as deciding how many servers should be powered up, one would have to decide how many of the active servers should be allocated to each queue. In addition, there could be a cost associated with switching an operative server from one queue to another. That would also be an interesting topic of future research.

Acknowledgements

This work was carried out in the context of a collaboration between Kazakh National University and Newcastle University. Financial support was also provided by the EPSRC project SIDE, which investigates the optimal use of computer clouds for stream processing.

References

[1] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions*, 10th printing, National Bureau of Standards, Washington, DC, 1972.

[2] M.N. Bennani and D. Menascé, “Resource allocation for autonomic data centers using analytic performance methods”, Procs., 2nd IEEE Conf. on Autonomic Computing (ICAC-05), pp 229-240, 2005.

[3] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat and R.P. Doyle, “Managing energy and server resources in hosting centers”, Procs 18th ACM symposium on Operating systems principles, NY, 2001.

[4] A. Chandra, W. Gong and P. Shenoy, “Dynamic resourse allocation for shared data centers using online measurements”, Procs., 11th ACM/IEEE Int. Workshop on Quality of Service (IWQoS), pp 381-400, 2003.

[5] X. Chen, P. Mohapatra and H. Chen, “An admission control scheme for predictable server response time for web accesses”,Procs., WWW10, Hong Kong, 2001.

[6] V. Kanodia and E.W. Knightly, “Multi-class latency bounded web services”, Procs., 8th Int. Workshop on Quality of Service (IWQOS 2000), pp 231-239, 2000.

[7] R. Levy, J. Nagarajarao, G. Pacifici, A. Spreitzer, A. Tantawi and A. Youssef, “Performance management for cluster based web services”, 8th Int. Symp. on Integrated Network Management, 2003.

[8] Z. Liu, M. Squillante and J.L. Wolf, “On maximizing service-level-agreement profits”, Procs., EC’01, Tampa, 2001.

[9] M. Mazzucco, I. Mitrani, J. Palmer, M. Fisher and P. McKee, “Web Service Hosting and Revenue Maximization”, Procs., 5th European Conf. on Web Services (ECOWS’07), Hale, 2007.

[10] M. Mazzucco, I. Mitrani, M. Fisher and P. McKee, “Allocation and Admission Policies for Service Streams”, Procs. MASCOTS’08, Baltimore, pp 155-162, 2008.

[11] M. Mazzucco, M. Vasar and M. Dumas, “Squeezing out the Cloud via Profit-Maximizing Resource Allocation Policies”, submitted for publication.

[12] I. Mitrani, *Probabilistic Modelling*, Cambridge University Press, 1998.

[13] I. Mitrani, “Managing Performance and Power Consumption in a Server Farm”, Annals of Operations Research, DOI:10.1007/s10479-011-0932-1, 2011.

[14] D. Villela, P. Pradhan and D. Rubinstein, “Provisioning servers in the application tier for e-commerce systems”, *ACM Trans on Internet Technology*, Vol. 7, No. 1, 2007.

[15] W. Whitt, “Approximations for the GI/G/m Queue”, *Production and Operations Management*, Vol. 2, No. 2, pp. 114-161, 1993.