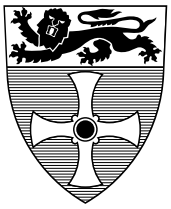
UNIVERSITY OF
NEWCASTLE



University of Newcastle upon Tyne

COMPUTING SCIENCE

Server allocation in Grid systems with on/off sources

J. Slegers, I. Mitrani, N. Thomas.

TECHNICAL REPORT SERIES

No. CS-TR-982 September, 2006

TECHNICAL REPORT SERIES

No. CS-TR-982 September, 2006

erver allocation in Grid systems with on/off sources

Joris Slegers, Isi Mitrani, and Nigel Thomas

Abstract

A system consisting of a number of servers, where demands of different types arrive in bursts (modelled by interrupted Poisson processes), is examined in the steady state. The problem is to decide how many servers to allocate to each job type, so as to minimize a cost function expressed in terms of average queue sizes. First, an exact analysis is provided for an isolated IP/M/n queue. The results are used to compute the optimal static server allocation policy. The latter is then compared to two heuristic policies which employ dynamic switching of servers from one queue to another (such switches take time and hence incur costs).

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

SLEGERS, J., MITRANI, I., THOMAS, N..

Server allocation in Grid systems with on/off sources
[By] J. Slegers, I. Mitrani, N. Thomas.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 2006.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, No. CS-TR-982)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE
Computing Science. Technical Report Series. CS-TR-982

Abstract

A system consisting of a number of servers, where demands of different types arrive in bursts (modelled by interrupted Poisson processes), is examined in the steady state. The problem is to decide how many servers to allocate to each job type, so as to minimize a cost function expressed in terms of average queue sizes. First, an exact analysis is provided for an isolated IP/M/n queue. The results are used to compute the optimal static server allocation policy. The latter is then compared to two heuristic policies which employ dynamic switching of servers from one queue to another (such switches take time and hence incur costs).

About the author

Isi Mitrani studied Mathematics at the Universities of Sofia and Moscow (Diploma, 1965), and Operations Research at the Technion, Haifa (MSc, 1967). He joined the University of Newcastle in 1968 as a Programming Advisor, became a Lecturer in 1969 (PhD, 1973), Reader in 1986 and Professor in 1991. He has held several visiting positions, including sabbatical years at INRIA (Le Chesnay) and Bell Laboratories (Murray Hill). His research interests are in the areas of probabilistic modelling, performance evaluation and optimization. Publications include 4 authored books, 3 edited books, more than 100 journal and conference papers and one patent.

Nigel Thomas joined the School of Computing Science in January 2004 from the University of Durham, where he had been a lecturer since 1998. His research interests lie in performance modelling, in particular Markov modelling through queueing theory and stochastic process algebra.
Nigel was awarded a PhD in 1997 and an MSc in 1991, both from the University of Newcastle upon Tyne.
Nigel is currently Director of Postgraduate Teaching within the School of Computing Science and Degree Programme Director and Chair of the Board of Examiners for the MSc and Diploma in Computing Science.

Joris Slegers received an M.Sc. in applied mathematics from the University of Twente (The Netherlands), with a specialisation in Fundamental Analysis. In October 2005 he joined the University of Newcastle as a Ph.D. student as part of the DOPCHE-project. His research is currently mainly in the area of preformance modelling and evaluation.

Suggested keywords

SERVER ALLOCATION,
DYNAMIC PROGRAMMING,
HEURISTICS,
GRID COMPUTING

Server Allocation in Grid Systems with On/Off Sources

Joris Slegers, Isi Mitrani and Nigel Thomas

School of Computing Science, Newcastle University, NE1 7RU
[j.a.l.slegers,isi.mitrani,nigel.thomas]@ncl.ac.uk

Abstract. A system consisting of a number of servers, where demands of different types arrive in bursts (modelled by interrupted Poisson processes), is examined in the steady state. The problem is to decide how many servers to allocate to each job type, so as to minimize a cost function expressed in terms of average queue sizes. First, an exact analysis is provided for an isolated $IP/M/n$ queue. The results are used to compute the optimal *static* server allocation policy. The latter is then compared to two heuristic policies which employ *dynamic* switching of servers from one queue to another (such switches take time and hence incur costs).

1 Introduction

Recent developments in distributed and grid computing have facilitated the hosting of service provisioning systems on clusters of computers. Users do not have to specify the server on which their requests (or 'jobs') are going to be executed. Rather, jobs of different types are submitted to a central dispatcher, which sends them for execution to one of the available servers. Typically, the job streams are bursty, i.e. they consist of alternating 'on' and 'off' periods during which demands of the corresponding type do and do not arrive.

In such an environment it is important, both to the users and the service provider, to have an efficient policy for allocating servers to the various job types. One may consider a static policy whereby a fixed number of servers is assigned to each job type, regardless of queue sizes or phases of arrival streams. Alternatively, the policy may be dynamic and allow servers to be reallocated from one type of service to another when the former becomes under-subscribed and the latter over-subscribed. However, each server reconfiguration takes time, and during it the server is not available to run jobs; hence, a dynamic policy must involve a careful calculation of possible gains and losses.

The purpose of this paper is to (i) provide a computational procedure for determining the optimal static allocation policy and (ii) suggest acceptable heuristic policies for dynamic server reconfiguration. In order to achieve (i), an exact solution is obtained for an isolated queue with n parallel servers and an on/off source. The dynamic heuristics are evaluated by simulation.

The problem described here has not, to our knowledge, been addressed before. Much of the server allocation literature deals with polling systems, where a single

server attends to several queues [2–6]. Even in those cases it has been found that the presence of non-zero switching times makes the optimal policy very difficult to characterize and necessitates the consideration of heuristics. The only general result for multiple servers concerns the case of Poisson arrivals and no switching times or costs: then the $c\mu$-rule is optimal, i.e. the best policy is to give absolute preemptive priority to the job type for which the product of holding cost and service rate is largest (Buyukkoc et al [1]).

A model similar to ours was examined by Palmer and Mitrani [10]; however, there all arrival processes were assumed to be Poisson; also, the static allocation was not done in an optimal manner. The novelty of the present study lies in the inclusion of on/off sources, the computation of the optimal static policy and the introduction of new dynamic heuristics.

The assumptions of model are stated in section 2. The analysis of the $IP/M/n$ queue, leading to the optimal static policy, is presented in section 3. Section 4 describes the dynamic heuristics, while section 5 shows the results of experiments comparing the performance of the different policies.

2 The model

The system contains N servers, each of which may be allocated to the service of any of M job types. There is a separate unbounded queue for each type. Jobs of type i arrive according to an independent interrupted Poisson process with on-periods distributed exponentially with mean $1/\xi_i$, off-periods distributed exponentially with mean $1/\eta_i$ and arrival rate during on-periods λ_i $(i = 1, 2, ..., M)$. The required service times for type i are distributed exponentially with mean $1/\mu_i$.

Any of queue i's servers may at any time be switched to queue j; the reconfiguration period, during which the server cannot serve jobs, is distributed exponentially with mean $1/\zeta_{i,j}$. If a service is preempted by the switch, it is eventually resumed from the point of interruption.

The cost of keeping a type i job in the system is c_i per unit time $(i = 1, 2, ..., M)$. These 'holding' costs reflect the relative importance, or willingness to wait, of the M job types. The system performance is measured by the total average cost, C, incurred per unit time:

$$C = \sum_{i=1}^{N} c_i L_i \ , \tag{1}$$

where L_i is the steady-state average number of type i jobs present. Those quantities depend, of course, on the server allocation policy.

In principle, it is possible to compute the optimal dynamic switching policy by treating the model as a Markov decision process and solving the corresponding dynamic programming equations. However, such a computation is tractable only for very small systems. What makes the problem difficult is the size of the state space one has to deal with. The system state at any point in time is described

by a quadruple, $S = (\mathbf{j}, \mathbf{n}, \mathbf{u}, \mathbf{m})$, where $\mathbf{j}$ is a vector whose ith element, j_i, is the number of jobs in queue i (including the jobs in service); $\mathbf{n}$ is a vector whose ith element, n_i, is the number of servers currently assigned to queue i; $\mathbf{u}$ is a vector whose ith element, u_i, is 0 if the ith arrival process is in an off-period, 1 if it is on; $\mathbf{m}$ is a matrix whose element $m_{i,k}$ is the number of servers currently being switched from queue i to queue k. The possible actions that the policy may take in each state are to do nothing or to initiate a switch of a server from queue i to queue k.

A numerical procedure to determine the optimal policy would involve truncating the queue sizes to some reasonable level, discretizing the time parameter through uniformization and then applying either policy improvement or value iterations (e.g., see [11, 12]). It is readily appreciated that the computational complexity of that task grows very quickly with the number of queues, M, the number of servers, N, and the truncation level. For that reason, we have concentrated on determining the optimal static allocation policy (which does not involve switching) and comparing its performance with that of some dynamic heuristics.

3 The $IP/M/n$ queue

If the server allocation is fixed, with n_i servers assigned to queue i ($n_1+n_2+\ldots+n_M = N$), then the M queues are independent of each other. Queue i behaves like an isolated $IP/M/n_i$ queue and may be analyzed as such. To simplify notation, the index i will be omitted in this section.

The state of the queue is described by the pair (j, u), where j is the number of jobs present and u is 0 if the arrival process is in an off-period, 1 if it is on. Let $p_{j,u}$ be the equilibrium probability of state (j, u). Also denote by μ_j the total service completion rate when there are j jobs present: $\mu_j = \min(j, n)\mu$.

The necessary and sufficient condition for stability is that the offered load is less than the number of servers:

$$\frac{\lambda\eta}{\mu(\xi+\eta)} < n \; . \tag{2}$$

When that condition is satisfied, the steady-state probabilities satisfy the following set of balance equations ($j = 0, 1, \ldots \; ; \; u = 0, 1$).

$$[\lambda u + \mu_j + \xi u + \eta(1-u)]p_{j,u} = \lambda u p_{j-1,u} + \mu_{j+1}p_{j+1,u}$$

$$+ [\xi(1-u) + \eta u]p_{j,1-u} \; , \tag{3}$$

where $p_{-1,u} = 0$ by definition.

This model can be solved numerically by treating it as a 'Markov-modulated queue'. The Markovian environment that influences the behaviour of the queue is the phase of its arrival process. Then one can compute performance measures by applying either the spectral expansion or the matrix-geometric solution method

(see [7, 9]). However, the present model is sufficiently simple to allow both an explicit exact analysis and an approximate solution in closed-form.

It is convenient to introduce the generating functions of the probabilities corresponding to off and on periods, respectively:

$$g_0(z) = \sum_{j=0}^{\infty} p_{j,0} z^j \;\; ; \;\; g_1(z) = \sum_{j=0}^{\infty} p_{j,1} z^j \; . \tag{4}$$

Then the balance equations (3) can be transformed into a set of two equations for $g_0(z)$ and $g_1(z)$.

$$[\eta z - n\mu(1-z)] g_0(z) = \xi z g_1(z) - \mu(1-z) P_0(z) \; , \tag{5}$$

$$\begin{aligned} &[\lambda z(1-z) - n\mu(1-z) + \xi z] g_1(z) \\ &\quad = \eta z g_0(z) - \mu(1-z) P_1(z) \; , \end{aligned} \tag{6}$$

where $P_0(z)$ and $P_1(z)$ are two polynomials involving 'boundary' probabilities (corresponding to states with state-dependent departure rates):

$$P_0(z) = \sum_{j=0}^{n-1} (n-j) p_{j,0} z^j \;\; ; \;\; P_1(z) = \sum_{j=0}^{n-1} (n-j) p_{j,1} z^j \; . \tag{7}$$

From equations (5) and (6), $g_0(z)$ and $g_1(z)$ can be expressed as rational functions whose numerators involve $P_0(z)$ and $P_1(z)$, and a common denominator, $d(z)$. The latter is quadratic and has two real zeros, z_1 and z_2, such that $0 < z_1 < 1$ and $1 < z_2 < \infty$.

The balance equations (3) for $j < n-1$ supply $2n-2$ equations for the coefficients of $P_0(z)$ and $P_1(z)$. An additional equation is provided by the normalizing condition $g_0(1) + g_1(1) = 1$. The final equation is obtained by observing that the generating functions are finite in the interior of the unit disc and therefore their numerators must vanish at $z = z_1$.

This determines all unknown probabilities, and hence the full distribution of the queueing process.

The average number of jobs present, L, is given by

$$L = g_0'(1) + g_1'(1) \; . \tag{8}$$

Thus, the procedure for determining the optimal static allocation of servers would use the solution described here to evaluate the cost function (1) for each feasible allocation and then choose the best one. A server allocation (n_1, n_2, ..., n_M), with $n_1 + n_2 + \ldots + n_M = N$, is feasible if the stability condition (2) is satisfied for every queue.

It can also be shown that the tail of the queue size distribution is geometric with parameter $1/z_2$ (see [8]). When the queue is heavily loaded, this leads to a very simple approximation for the average queue size:

$$L = \frac{1/z_2}{1 - 1/z_2} = \frac{1}{z_2 - 1} \; . \tag{9}$$

Using that approximation speeds up the search for the optimal static server allocation considerably.

4 Dynamic heuristics

It is to be expected that a dynamic allocation policy which reacts to changing queue sizes and switches servers when large disparities occur, can achieve lower costs than even the best static policy. However, because of the size of the state space, a computation of the optimal policy is impractical. Hence, our objective is to design heuristic dynamic policies and compare their performance with the optimal static policy.

Two heuristics will be examined. In both cases, switching decisions are made by taking into account the currently observed system state and estimating the costs that would be incurred over some subsequent period of time if (a) no action is taken, and (b) one or more servers are switched from queue j to queue i. The two policies differ by the way they estimate future costs.

The first policy will be referred to as the *Average Flow* heuristic. It ignores the on/off periods and treats queue i as a deterministic fluid which arrives at rate γ_i, given by

$$\gamma_i = \frac{\lambda_i \eta_i}{\xi_i + \eta_i} \ . \tag{10}$$

That fluid is consumed at rate $n_i \mu_i$, where n_i is the number of servers currently allocated to queue i.

Suppose that two queues, i and j, have current sizes k_i and k_j, and currently allocated numbers of servers n_i and n_j, respectively. If no further actions are taken, and both queues are stable (i.e., $\gamma_i < n_i \mu_i$ and $\gamma_j < n_j \mu_j$), then those fluid queues would decrement at constant rates and would empty in times $k_i/(n_i \mu_i - \gamma_i)$ and $k_j/(n_j \mu_j - \gamma_j)$, respectively. The total holding costs incurred would be proportional to the areas of the resulting triangles.

Hence, the *Average Flow* heuristic estimates the cost of taking no action with queues i and j as

$$C_0 = \frac{c_i k_i^2}{2(n_i \mu_i - \gamma_i)} + \frac{c_j k_j^2}{2(n_j \mu_j - \gamma_j)} \ . \tag{11}$$

On the other hand, if a decision is made to switch a server from queue j to queue i, and that switch takes time $1/\zeta$ (deterministic), then the service rate at queue j immediately reduces to $(n_j - 1)\mu_j$, while that at queue i remains the same for the duration of the switch and then increases to $(n_i + 1)\mu_i$. Assuming that queue i does not empty during the switch, its size at the point when the switch is completed would be equal to m_i, where

$$m_i = k_i - (n_i \mu_i - \gamma_i)/\zeta \ . \tag{12}$$

The total holding cost incurred in clearing both queues is estimated as

$$C_1 = \frac{c_i(k_i + m_i)}{2\zeta} + \frac{c_i m_i^2}{2((n_i + 1)\mu_i - \gamma_i)} + \frac{c_j k_j^2}{2((n_j - 1)\mu_j - \gamma_j)} \ . \tag{13}$$

At every arrival or departure event, the *Average Flow* heuristic evaluates C_0 and C_1 for every pair of queues i and j, where i is the queue where the arrival

occurred, or j is the queue where the departure occurred. If $C_1 < C_0$, a server is switched from queue j to queue i. If that inequality holds for more than one queue i, the switch is made to the queue for which the difference $C_0 - C_1$ is largest. If a contemplated switch would leave queue j potentially unstable (i.e., $(n_j - 1)\mu_j \leq \gamma_j$), then it is not made. If, at a decision instant, a server is in the process of being switched, it is counted as being already available at the destination queue.

The second policy will be referred to as the *On/Off* heuristic. When making allocation decisions, it assumes that that the current phase of each arrival processes, whether it is 'on' or 'off', will last forever. Again queue i is treated as a fluid, but the arrival rate is taken to be $\gamma_i = \lambda_i u_i$, where $u_i = 1$ if the queue i arrival process is in an on-period and $u_i = 0$ if it is in an off-period.

Switching decisions are made not only at arrival and departure instants, but also when an arrival process changes phase from 'on' to 'off' or vice versa. As well as evaluating the estimated costs C_0 and C_1, of doing nothing or switching one server from queue j to queue i, the *On/Off* heuristic evaluates the costs C_s, of switching s servers from queue j to queue i, for $s = 2, 3, \ldots, n_j$. This is necessary because a phase change can make a big difference to the arrival rate at a queue, requiring or releasing more than one server. When calculating C_s for $s > 1$, one could assume that all s servers become available at queue i after a switching interval of length $1/\zeta$. Alternatively, the assumption could be that the s switches complete at different times: the earliest after an interval $1/(s\zeta)$, the next after a subsequent interval $1/((s-1)\zeta)$, etc. The first alternative would be appropriate if the switching times are nearly constant, the second if they are exponentially distributed. In both cases, the costs are evaluated by adding together areas under linear segments.

As before, the switching decision that yields the largest cost reduction is taken. If no reduction is possible, or if all estimated costs are infinite (that can happen, for example, if all arrival processes are in an on-period and the corresponding arrival rates are greater than the available service rates), then no action is taken.

5 Results

To illustrate the behaviour of the dynamic heuristics, and compare their performance with that of the optimal static allocation policy, a cluster of servers was simulated. For completeness, a simpler static policy was also included in some comparisons. The latter is referred to as the ρ-rule: it allocates the servers to job types in proportion to costs, c_i, and offered loads, $\rho_i = \gamma_i/\mu_i$ (γ_i is given by (10)). The ρ-rule is what one would apply without the benefit of the analysis in section 3. Indeed, that was the static allocation used in [10].

The following parameters were kept fixed throughout.

Number of job types: $M = 2$.
Number of servers: $N = 20$.
Average required service times: $1/\mu_1 = 1/\mu_2 = 1$.

In the first experiment, the average 'on' and 'off' periods were equal at the two queues, and so were the average switching times: $1/\xi_i = 1/\eta_i = 100$, $i = 1, 2$; $1/\zeta_{i,j} = 1$, $i, j = 1, 2$. The two arrival rates were also equal, and were increased simultaneously. Some asymmetry was introduced by making type 2 jobs more expensive than type 1: the holding costs were $c_1 = 1$, $c_2 = 1.5$. The simulated time for each run was 10000 time units. Since each arrival process is 'on' for about half of that time, if $\lambda_1 = \lambda_2 = 10$, a total of about about 100000 jobs go through the system. (Note that the simulations were required only for the dynamic policies. The static ones could have been solved numerically, but since the simulation programs could easily be adapted to different policies, they were used in all cases.)

Figure 1 shows the average costs achieved by the two static and two dynamic policies, as the load increases. As expected, at light loads it does not matter very much which policy is adopted. However, differences start appearing at medium loads and become ever larger at heavy loads.

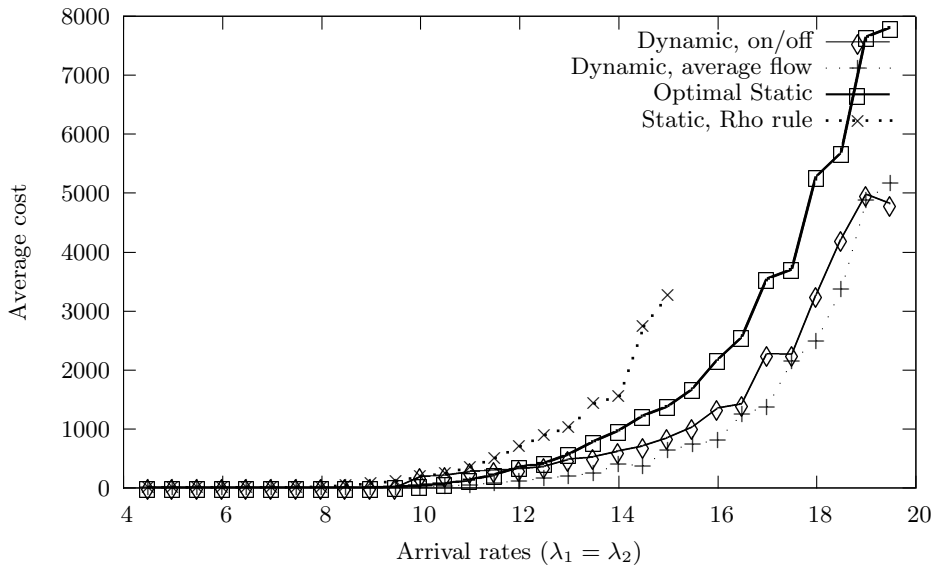


Fig. 1. Policy comparisons: increasing λ_1, λ_2

The ρ-rule has the worst performance. Its application results in 8 servers being allocated to type 1 and 12 servers to type 2 throughout. Hence, when $\lambda_1 \geq 16$, the system becomes unstable and in the long run incurs infinite cost. The optimal static policy allocates 9 servers to type 1 and 11 to type 2 for most of the range, changing to 10 and 10 when the arrival rates become greater than about 17. That policy achieves considerably lower costs than the ρ-rule, and remains stable for $\lambda_i < 20$. The benefits of the dynamic allocation policies become significant for $\lambda_i > 13$. Their performance is similar, although the *Average Flow*

policy appears to be consistently slightly better than *On/Off*. We point this out as an observed fact, but cannot explain it. Intuitively, one might have expected that a heuristic which takes into account the current state of the input stream would make better allocation decisions.

Figure 2 shows the effect of increasing the average switching time, in a reasonably lightly loaded system ($\lambda_1 = \lambda_2 = 10$). We observe that the cost of the *Average Flow* heuristic increases initially and then stops changing significantly: it performs slightly better than the optimal static policy (whose allocation does not change) when $1/\zeta < 4$ and then becomes slightly worse. On the other hand, the *On/Off* policy shows an initial steady improvement before its cost also stops changing significantly.

The static allocation policy based on the ρ-rule is not included in these and subsequent comparisons because its performance can only be worse than that of the optimal static policy.

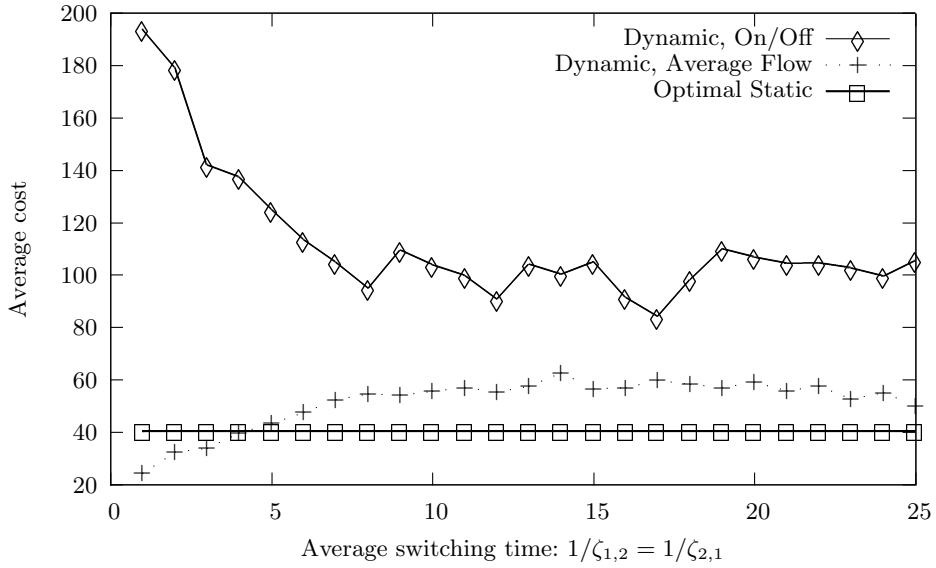


Fig. 2. Performance of different policies for increasing switching times

The last experiment involves a system with asymmetric traffic characteristics. Type 1 jobs arrive in a steam which is 'on' most of the time: $\xi_1 = 1/100000$, $\eta_1 = 1$, $\lambda_1 = 10$. That stream would need at least 10 servers in order to remain stable. Jobs of type 2 arrive in short bursts, with long intervals in between: $\xi_2 = 1/25$, $\eta_2 = 1/500$ (i.e., the arrival stream of type 2 is 'on' for less than 5% of the time). The arrival rate of type 2 during 'on' periods, λ_2, is increased from 20 to 120 in steps of 5. The switching costs are very small.

Figure 3 illustrates very clearly the benefits of using a dynamic allocation policy. Again, there is not much difference between the performance of the *Av-*

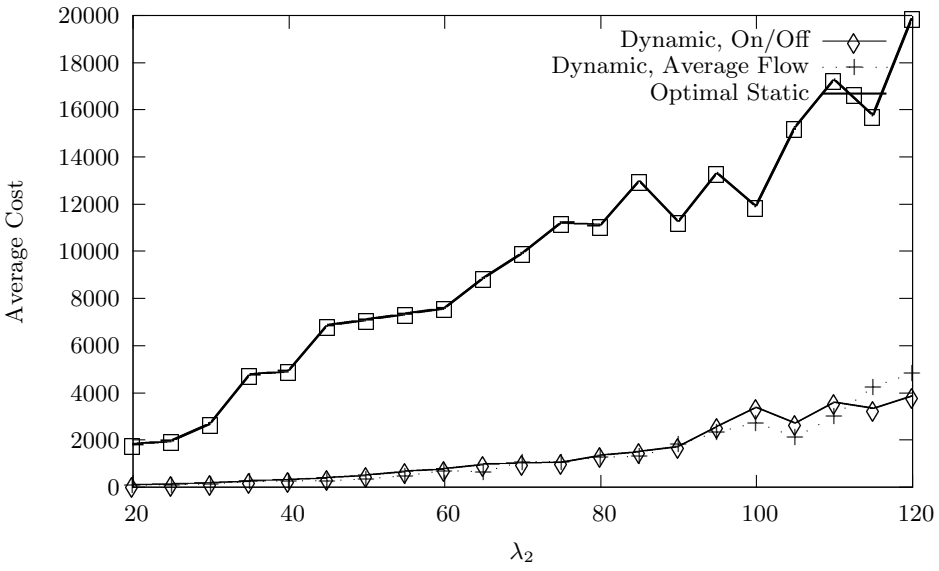


Fig. 3. One steady and one bursty source: increasing λ_2

erage Flow and the *On/Off* heuristics. However, the average cost achieved by either of them can be an order of magnitude lower that that of the optimal static policy.

6 Conclusions

We have addressed a resource allocation problem that is of considerable importance in the context of distributed computing and grid systems with heterogeneous, bursty demand streams. Under Markovian assumptions, the optimal static allocation policy can be determined quite simply, using the analytic solution provided in section 3. An even simpler approximate solution is also available.

Two dynamic heuristic policies were proposed and evaluated by simulation. When the system is heavily loaded, both achieve large savings in costs, compared to the best static policy.

Further work is required in several directions. First, it is clearly necessary to carry out more extensive evaluations and comparisons for different system configurations (including larger numbers of job types), cost structures and patterns of demand. Next, are there other, better dynamic heuristics that react more promptly and accurately to changes of state? Also, it would be desirable to implement the proposed heuristics in a real grid system and measure their performance. Finally, how would the resource allocation problem change if the servers, too, can be 'on' or 'off' (they may be subject to breakdowns and repairs, or for other reasons become unavailable from time to time)? These and other extensions will be tackled in the future.

Acknowledgements

This work was carried out in the framework of the collaborative project DOPCHE (Dynamic Operative Policies for Commercial Hosting Environments), funded by the UK Engineering and Physical Sciences Research Council under its E-Science programme. The support of the Network of Excellence EuroNGI, funded by the EU, is also acknowledged.

References

1. C. Buyukkoc, P. Varaiya and J. Walrand, "The $c\mu$-rule revisited", *Advances in Applied Probability*, 17, pp 237-238, 1985.
2. I. Duenyas and M.P. Van Oyen, "Heuristic Scheduling of Parallel Heterogeneous Queues with Set-Ups", *Technical Report 92-60*, Department of Industrial and Operations Engineering, University of Michigan, 1992.
3. I. Duenyas and M.P. Van Oyen, "Stochastic Scheduling of Parallel Queues with Set-Up Costs", *Queueing Systems Theory and Applications*, 19, pp 421-444, 1995.
4. G. Koole, "Assigning a Single Server to Inhomogeneous Queues with Switching Costs", *Theoretical Computer Science*, 182, pp 203-216, 1997.
5. G. Koole, "Structural Results for the Control of Queueing Systems using Event-Based Dynamic Programming", *Queueing Systems Theory and Applications*, 30, pp 323-339, 1998.
6. Z. Liu, P. Nain, and D. Towsley, "On Optimal Polling Policies", *Queueing Systems Theory and Applications*, 11, pp 59-83, 1992.
7. I. Mitrani and D. Mitra, "A spectral expansion method for random walks on semi-infinite strips", *IMACS Symposium on Iterative Methods in Linear Algebra*, Brussels, 1991.
8. I. Mitrani, "Approximate Solutions for Heavily Loaded Markov Modulated Queues", *Performance Evaluation*, 62, pp 117-131, 2005.
9. M.F. Neuts, *Matrix Geometric Solutions in Stochastic Models*, John Hopkins Press, 1981.
10. J. Palmer and I. Mitrani, "Optimal Server Allocation in Reconfigurable Clusters with Multiple Job Types", Journal of Parallel and Distributed Computing, 65/10, pp 1204-1211, 2005.
11. E. de Souza e Silva and H.R. Gail, "The Uniformization Method in Performability Analysis", in *Performability Modelling* (eds B.R. Haverkort, R. Marie, G. Rubino and K. Trivedi), Wiley, 2001.
12. H.C. Tijms, Stochastic Models, Wiley, New York, 1994.