Ezhilchelvan P, Mitrani I.

[Optimal provision of multiple service types.](#)

In: IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS). 2016, London, UK: IEEE Computer Society.

Copyright:

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

This is the authors' accepted manuscript of a paper that was presented at IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) conference, held 19-21 September 2016, London, UK.

DOI link to article:

<http://dx.doi.org/10.1109/MASCOTS.2016.16>

Date deposited:

08/12/2016

Optimal provision of multiple service types

Paul Ezhilchelvan and Isi Mitrani

School of Computing Science, Newcastle University, NE1 7RU, U.K.

e-mail: paul.ezhilchelvan@ncl.ac.uk, isi.mitrani@ncl.ac.uk

Abstract—Services of different types are provided to paying customers on servers hired from a cloud. Different virtual machines can share a server, subject to one or more resource constraints. Incoming jobs whose resource requirements cannot be satisfied are lost. The objective is to maximize the long-term average profit per unit time. A single-server model is analyzed exactly and the results provide approximations for the system with n servers. The latter is also solved exactly when the servers are dedicated and when the VMs can migrate instantaneously. Numerical examples and comparisons with simulations are presented.

I. INTRODUCTION

This paper is concerned with the provision of services of different types, with different patterns of demand, resource requirements and revenue streams. The service provider hires servers from a Cloud, incurring certain costs. To run a job of a given type, a Virtual Machine (VM) of that type is instantiated on one of the servers. The resource availability on a server is bounded, so that whether a VM can be allocated to it or not, depends both on the type of the new job and on the numbers and types of the other jobs already running. When an incoming job cannot be started on any of the servers, it is rejected and the revenue that it would bring is lost.

The problem is to decide how many servers to hire so as to maximize the average long-term profit (revenues minus costs) per unit time. To that end, we examine first a model of a single server with either a single shared resource or multiple shared resources. The exact solution of that model, which is known, is then used to provide accurate estimates for the profit achieved by n servers and hence for purposes of optimization.

Since servers are hired, the numbers involved are not considered to be large. They tend to be on the order of tens, rather than the thousands that are typically available in a service center.

Two other models are solved exactly: (i) groups of servers, each of which is dedicated to a particular job type; (ii) VMs that can be moved (migrated) from server to server and packed efficiently according to some simple algorithm. The first of these underestimates the achievable profit, while the second overestimates it, but only slightly.

We assume that the demand parameters are given, and the system reaches steady state during a period where those parameters remain fixed. In practice, the hiring policies would have to be supplemented by some monitoring and parameter estimation technique that would detect when the traffic parameters change. Such techniques exist (see below).

The resource sharing and optimization problems described here have not, to our knowledge, been addressed before in

the context of server sharing by multiple job types. There has been quite a lot of work on server allocation with a single job type. Perhaps the closest to the present study is the paper by Ezhilchelvan and Mitrani [4], where it was found that dynamic allocation policies do not bring significant benefits over static ones. The trade-off between performance and energy consumption, again for a single job type, was examined by Mazzucco et al. [8], [9], using models and empirical observations. Their focus, and also that of Bodík et al. [2], is on estimating the traffic and reacting to changes in the parameters.

The studies by Wood et al [19], Singh et al [14], Weijia et al [18], and Arzuaga and Kaeli [1], assume a given set of jobs currently present in the system, together with their resource requirements, and aim to allocate the corresponding VMs so as to minimize the number of servers and satisfy certain performance constraints. There are similar works concerned with power management (eg. Tang et al [15] and Moore et al [11]). None of these papers take into account the processes of job arrivals and services.

The single server models and their solutions are described in section II. The profit maximization problem is introduced and solved in section III. The case of dedicated servers, together with some numerical results, is also presented here. Section IV covers the model with moveable VMs and packing, while some conclusions and directions for future work are summarized in section V.

II. MODEL OF A SINGLE SERVER

A server may be shared by VMs of K different types, numbered $1, 2, \dots, K$. The service provided by a VM of type i during its lifetime is referred to as a ‘job of type i ’. Jobs of type i arrive in an independent Poisson stream with rate λ_i . Their service times may be general IID random variables with mean $1/\mu_i$ ($i = 1, 2, \dots, K$).

Assume, to begin with, that the resource requirement of a VM is measured by a single number. More precisely, a job of type i consumes b_i units of resource. In order that the jobs running in parallel do not interfere with each other unduly, an upper bound B is imposed on the total amount of resource used by the jobs in the server. An incoming job that would cause that bound to be exceeded is rejected and is lost.

This model is of a type introduced and solved some decades ago in connection with circuit-switching networks. There, a number of trunks are allocated to calls of different types (e.g., see Ross [13]). The product-form solution was shown to be insensitive to the service time distribution.

The state of the server is described by the integer vector $\mathbf{j} = (j_1, j_2, \dots, j_K)$, where j_i is the number of jobs of type i in progress. Denote by $S(K, B)$ the set of admissible state vectors. The resource restriction implies that this set is defined by

$$S(K, B) = \left\{ \mathbf{j} : \mathbf{j} \geq 0, \sum_{i=1}^K j_i b_i \leq B \right\}. \quad (1)$$

The dependence of $S(K, B)$ on the individual resource requirements b_i is left implicit in order to keep the notation simple.

Let $\pi(\mathbf{j})$ be the steady-state probability that the server is in state $\mathbf{j}$. These probabilities are given by

$$\pi(\mathbf{j}) = \frac{1}{G(K, B)} \prod_{i=1}^K \frac{\rho_i^{j_i}}{j_i!}; \quad \mathbf{j} \in S(K, B), \quad (2)$$

where $\rho_i = \lambda_i / \mu_i$ is the offered load of type i . The normalizing constant $G(K, B)$ is chosen so that the sum of all probabilities is 1. That is,

$$G(K, B) = \sum_{\mathbf{j} \in S(K, B)} \prod_{i=1}^K \frac{\rho_i^{j_i}}{j_i!}. \quad (3)$$

Computing the normalization constant $G(K, B)$ can be a non-trivial task. A simple way to accomplish it is to use recursion. Let $m_i = \lfloor B/b_i \rfloor$ be the largest possible number of type i jobs that can be admitted into the server. Consider a particular job type, say type K , and note that if there are j jobs of type K present, the amount of resource they use is jb_K , leaving $B - jb_K$ for the other job types. Hence, we can write

$$G(K, B) = \sum_{j=0}^{m_K} \frac{\rho_K^j}{j!} G(K-1, B - jb_K). \quad (4)$$

The reduced normalization constants in the right-hand side of (4) are defined by (3) with one fewer job type and appropriately reduced state space. The recursion terminates if either $K = 0$, with $G(0, B) = 1$, or if $S(K, B)$ contains only the vector $\mathbf{j} = \mathbf{0}$ (i.e., the only feasible state is the one where the server is empty), again with $G(K, B) = 1$.

Another algorithm for computing $G(K, B)$ was proposed by Kaufman and Roberts [7], [12]. It is more difficult to implement but may be more efficient. The size of the state space faced by these algorithms is the main barrier to tackling problems with very large numbers of servers.

The performance measures of interest in this model are the probabilities, α_i , that an incoming job of type i is rejected ($i = 1, 2, \dots, K$). To determine those probabilities, note that a job of type i is *accepted* in all states $\mathbf{j}$ such that the resource currently used does not exceed $B - b_i$. The sum of the corresponding state probabilities is given by,

$$1 - \alpha_i = \frac{G(K, B - b_i)}{G(K, B)}, \quad i = 1, 2, \dots, K. \quad (5)$$

Thus, the performance measures can be computed by the same recursive procedure that evaluates $G(K, B)$.

A. Multiple resources

Suppose now that there are several critical resources, numbered $1, 2, \dots, L$. These may include CPU, memory, disks, communication bandwidth, etc. There is a bound, B_k , on the total amount of resource k that may be allocated to VMs. Each job of type i uses an amount $b_{i,k}$ of resource k . All other assumptions of the model are as before.

Denoting by $\mathbf{B} = (B_1, B_2, \dots, B_L)$ the vector of bounds, and by $\mathbf{b}_i = (b_{i,1}, b_{i,2}, \dots, b_{i,L})$ the vector of requirements for a job of type i , we can define the new state space as

$$S(K, \mathbf{B}) = \left\{ \mathbf{j} : \mathbf{j} \geq 0, \sum_{i=1}^K j_i \mathbf{b}_i \leq \mathbf{B} \right\}, \quad (6)$$

where the bounding inequalities must be satisfied element by element, for all elements of the corresponding vectors.

Having made that change, the main results continue to hold. The steady-state probabilities are of the form

$$\pi(\mathbf{j}) = \frac{1}{G(K, \mathbf{B})} \prod_{i=1}^K \frac{\rho_i^{j_i}}{j_i!}, \quad (7)$$

as can be verified again by checking that the local balance equations are satisfied. The normalizing constant is given by

$$G(K, \mathbf{B}) = \sum_{\mathbf{j} \in S(K, \mathbf{B})} \prod_{i=1}^K \frac{\rho_i^{j_i}}{j_i!}. \quad (8)$$

That constant can be computed by the recursive procedure

$$G(K, \mathbf{B}) = \sum_{j=0}^{m_K} \frac{\rho_K^j}{j!} G(K-1, \mathbf{B} - j\mathbf{b}_K), \quad (9)$$

where m_K , the largest number of type K jobs that can be admitted, is now equal to $\min(\lfloor B_1/b_{K,1} \rfloor, \lfloor B_2/b_{K,2} \rfloor, \dots, \lfloor B_L/b_{K,L} \rfloor)$. The boundary conditions are $G(0, \mathbf{B}) = 1$ and $G(K, \mathbf{B}) = 1$ if $S(K, \mathbf{B})$ contains only the vector $\mathbf{j} = \mathbf{0}$.

The probabilities, α_i , that an incoming job of type i is rejected, are obtained from

$$1 - \alpha_i = \frac{G(K, \mathbf{B} - \mathbf{b}_i)}{G(K, \mathbf{B})}, \quad i = 1, 2, \dots, K. \quad (10)$$

In general, increasing the number of resources whose usage is bounded leads to an increase in the number of constraining inequalities in the right-hand side of (6). This, in turn, is likely to decrease the number of admissible states. Consequently, the computational complexity of the procedure for determining the normalization constant and the performance measures is likely to decrease, rather than increase, when L increases.

III. OPTIMAL NUMBER OF SERVERS

Suppose that each accepted job of type i brings in a revenue of r_i . Each hired server incurs a cost of c per unit time. How many servers should be hired, given the characteristics of the demand (i.e., the arrival rates λ_i and the average service times $1/\mu_i$, $i = 1, 2, \dots, K$)?

Consider a system with n identical servers numbered $1, 2, \dots, n$. One possible mechanism for allocating an incoming job of type i is to assign it to the server with the lowest index where the job can be accepted. If none of the servers has room, then the job is rejected and is lost. This allocation policy will be referred to as ‘First-Fit-on-Arrival’, or FFoA.

Assume for now that, once a VM has been allocated to a server, it cannot be moved to another server. Later we shall consider moveable VMs.

Let $\beta_{i,n}$ be the steady-state probability that an incoming job of type i is rejected. Denote the vector of those probabilities by $\beta = (\beta_{1,n}, \beta_{2,n}, \dots, \beta_{K,n})$. The long-run average profit that the n servers achieve per unit time is given by

$$R(n, \beta) = \sum_{i=1}^K \lambda_i r_i (1 - \beta_{i,n}) - cn. \quad (11)$$

This function of n has been shown, in other contexts, to have a single maximum. In particular, in the special case of the single-class Erlang model, it has been proved that the rejection probability is convex in n , implying that $R(n, \beta)$ is concave. If we accept this single maximum conjecture, then the optimal value of n can be computed quite simply, by evaluating $R(n, \beta)$ for $n = 1, 2, \dots$, and stopping as soon as the profit ceases to increase. In fact, in practice one may not need to carry out a full search but would proceed incrementally. If there are n servers currently hired and the traffic monitor suggests that the offered loads have increased, evaluate the expected profit for $n+1, n+2, \dots$; if the offered loads have decreased, try $n-1, n-2, \dots$.

However, we still have the problem of determining $\beta_{i,n}$. The system state is now described by n vectors $\mathbf{j}_s$, where $j_{i,s}$ is the number of jobs of type i at server s ($s = 1, 2, \dots, n$). An exact solution, which would require finding the joint distribution of those n vectors, appears to be intractable. The closed-form solution of the previous section no longer applies because the servers are not independent of each other. We therefore propose approximate expressions for $\beta_{i,n}$ that are sufficiently accurate for purposes of optimization.

For simplicity, we shall concentrate on the single resource case where the capacity bound, and individual requirements, are expressed as single numbers. The generalization to multiple resources is quite straightforward and proceeds along the lines described in subsection II-A.

Compare the present system of n servers, each with a resource capacity B , with a hypothetical system consisting of a single server whose total resource capacity is nB . It is subjected to the same offered loads, $\rho = (\rho_1, \rho_2, \dots, \rho_K)$. That single-server system is roughly equivalent to the n -server one, but it makes a more efficient use of resource capacity and therefore tends to reject fewer jobs. Hence, the rejection probability for type i , α_i , in the single-server system, is likely to be an under-estimate, $\beta_{i,n}^u$, for the rejection probability for type i in the n -server system. Introducing a notation for α_i

where the dependence on offered loads and resource capacity is explicit, we write

$$\beta_{i,n}^u = \alpha_i(\rho, nB) ; \quad i = 1, 2, \dots, K. \quad (12)$$

Another estimate for $\beta_{i,n}$ is obtained by noting that an incoming job of type i tries to join server s only when server $s-1$ cannot accept it ($s = 2, 3, \dots, n$). Therefore, if $\sigma_{i,s}$ is the offered load of type i at server s , we may write

$$\sigma_{i,1} = \rho_i ; \quad \sigma_{i,s+1} = \sigma_{i,s} \alpha_i(\sigma_s, B) ; \quad s = 1, 2, \dots, n, \quad (13)$$

where σ_s is the vector $(\sigma_{1,s}, \sigma_{2,s}, \dots, \sigma_{K,s})$. These expressions are based on approximating the arrival processes into servers $2, 3, \dots, n$ as Poisson streams.

Then, treating the n servers as independent of each other, we get a second estimate, $\beta_{i,n}^v$, for the probability that an incoming job of type i is rejected by all n servers:

$$\beta_{i,n}^v = \prod_{s=1}^n \alpha_i(\sigma_s, B) ; \quad i = 1, 2, \dots, K. \quad (14)$$

This is also likely to be an underestimate because the conditional probability of a rejection at server $s+1$, given that there was no room at server s , may be expected to exceed the corresponding steady-state probability.

Faced with two possible underestimates, it is reasonable to take the larger rejection probability for each job type:

$$\beta_{i,n} = \max(\beta_{i,n}^u, \beta_{i,n}^v) ; \quad i = 1, 2, \dots, K. \quad (15)$$

These values are substituted in the right-hand side of equation (11) in order to estimate the profit achieved by the n -server system.

It should be pointed out that, although the tendency is for equations (12) and (14) to produce underestimates, that does not necessarily happen in all cases and for all job types. Some of the values of $\beta_{i,n}$ turn out occasionally to be overestimates. However, it appears that the rejection probability vectors produced by equation (15), and the resulting estimates of the achieved profit, are remarkably accurate. They can be used quite reliably in determining the optimal number of servers.

To illustrate and quantify the above results, consider an example system with three job classes, 1, 2 and 3, or ‘small’, ‘medium’ and ‘large’. The individual resource requirements of the three classes are $b_1 = 1$, $b_2 = 3$ and $b_3 = 5$, while the bound on resource usage per server is $B = 8$. Thus, a server can accommodate without interference up to 1 large and 1 medium job, or 1 large and 3 small jobs, or 2 medium and 2 small jobs, or 1 medium and 5 small jobs, or 8 small jobs.

The above numbers are motivated by similarities with the T2 family of VM instances offered by the Amazon EC2 (Elastic Computing Cloud) service (see [20]). The resource that is being shared and bounded in this context is vCPU (virtual CPU). That is, a total of 8 virtual CPUs are available on each server.

Larger jobs arrive less frequently but tend to stay longer than smaller ones. More precisely, the arrival rates for the different classes are $\lambda_1 = 6$, $\lambda_2 = 2$ and $\lambda_3 = 1$. The corresponding

Fig. 1. Estimated and simulated profit for different numbers of servers
 $K = 3$, $B = 8$, $\mathbf{b} = (1, 3, 5)$, $\boldsymbol{\lambda} = (6, 2, 1)$, $\boldsymbol{\mu} = (1, 1, 0.5)$, $\mathbf{r} = (1, 5, 10)$, $c = 3$

average residence times are $1/\mu_1 = 1$, $1/\mu_2 = 1$ and $1/\mu_3 = 2$. Thus the offered loads vector is $\boldsymbol{\rho} = (6, 2, 2)$.

Large jobs bring twice as much revenue as medium ones, which bring five times as much as small ones: $r_1 = 1$, $r_2 = 5$ and $r_3 = 10$. The cost of a server per unit time is $c = 3$, which means that one large job on its own makes a moderate profit (it occupies the server for 2 time units), one medium job makes a smaller profit, whereas 3 small jobs just cover the cost.

Figure 1 shows the average long-term profit achieved per unit time, R , as a function of the number of servers, N . One of the plots in the figure represents the numerical implementation of our model estimates. The other plot was obtained by simulating the arrival and departure processes, counting the number of rejections of different types and using the ratios of rejected jobs to incoming jobs as estimates of $\beta_{i,n}$ to be substituted into equation (11). Each point of the second plot was the result of a simulation run in which about 100000 jobs of all types went through the system. The runs were divided into 10 portions each, for the purpose of computing the 99% confidence intervals.

The figure confirms that the profit curve has a single maximum. This was to be expected. More surprising is the accuracy of the model estimates. The simulation estimates can be accepted as a basis for comparison, given the narrowness of their confidence intervals. Not only does the model predict correctly the optimal number of servers, but the value of the predicted optimal profit is well within the confidence interval of the simulated value.

This high accuracy of the model estimates is due to the ‘max’ operation in the right-hand side of (15). If either $\beta_{i,n}^u$ ($i = 1, 2, \dots, K$) on their own, or $\beta_{i,n}^v$ ($i = 1, 2, \dots, K$) on their own, had been taken as estimates of the rejection prob-

abilities, the distance between model and simulation would have been greater.

It is important to examine the predictive ability of the model under a variety of loading conditions. We have compared the computed and simulated optimal profit in the three-class example, as the total arrival rate, λ , increases. The ratios of class i arrival rates to the total are kept fixed, equal to the ones in figure 1: $\lambda_1/\lambda = 6/9$, $\lambda_2/\lambda = 2/9$, $\lambda_3/\lambda = 1/9$. The other parameters also keep their previous values.

The results of the comparison are displayed in figure 2. Each maximum profit point is obtained by evaluating the average long-term profit for $N = 1, 2, \dots$, using the model in one plot and simulation in the other, and stopping as soon as the profit ceases to increase.

The model predictions are almost indistinguishable from those of the simulations. Only when $\lambda = 25$ and $\lambda = 30$ are the former just outside the 95% confidence intervals of the latter.

The corresponding optimal numbers of servers, n^* , are shown in table 1.

TABLE I
OPTIMAL NUMBERS OF SERVERS

λ	5	10	15	20	25	30
Model n^*	2	4	7	9	11	12
Simulated n^*	2	4	6	8	10	12

There is a difference of one server on three occasions, but those differences have very small effects on the achievable profits.

Fig. 2. Optimal profit for different total arrival rates
 $K = 3$, $B = 8$, $\mathbf{b} = (1, 3, 5)$, $\boldsymbol{\lambda} = \lambda(6/9, 2/9, 1/9)$, $\mathbf{r} = (1, 5, 10)$, $c = 3$

A. Dedicated servers

Rather than share servers between VMs of different classes, one may decide to dedicate n_1 servers to type 1, n_2 servers to type 2, ..., n_K servers to type K . Each server can accommodate a maximum of m_i jobs of type i (in the case of a single resource this is given by $m_i = \lfloor B/b_i \rfloor$). Hence, there is room in the system for a total of $n_i m_i$ jobs of type i , $i = 1, 2, \dots, K$.

In this set-up the number of servers to hire can be optimized separately for each class, ignoring the others. The probability, $\beta_i(n_i)$, that an incoming job of type i will be rejected, is the Erlang loss probability

$$\beta_i(n_i) = \frac{\rho_i^{n_i m_i}}{(n_i m_i)!} \left[\sum_{j=0}^{n_i m_i} \frac{\rho_i^j}{j!} \right]^{-1}. \quad (16)$$

There is no approximation involved here.

The long-run average profit that the n_i servers dedicated to class i achieve per unit time is given by

$$R_i(n_i) = \lambda_i r_i [1 - \beta_i(n_i)] - c n_i. \quad (17)$$

This is a concave function and is easily maximized with respect to n_i .

Having performed the optimization for each class, the total number of servers hired, n , and the total profit obtained, $R(n)$, are given by

$$n = \sum_{i=1}^K n_i ; \quad R(n) = \sum_{i=1}^K R_i(n_i). \quad (18)$$

In order to compare the profitability of dedicated servers with that of the shared ones considered earlier, we have evaluated the optimal configurations of the two systems in

the context of the 3-class example of the previous section. In figure 3, the optimal profits achieved by the shared and the dedicated servers are plotted against the total arrival rate. Again, the individual arrival rates of the three job types are increased in fixed proportions.

It is clear that, for this pattern of demand, dedicated servers are significantly less profitable than shared ones. This is due to the fact that the available resources are used less efficiently. For example, type 3 jobs have a resource requirement of $b_3 = 5$, which means that the servers dedicated to type 3 can accommodate just one job each. When the servers are shared among the three types, a server can accept one job of type 3 and one of type 2, or one of type 3 and three of type 1.

We have tried quite hard, but without success, to construct examples where it would be advantageous to use dedicated servers. It appears that, when there are multiple job types, an optimized collection of shared servers is always better than one of dedicated servers. We have no proof of this, but the intuitive argument of more efficient utilization of resources is appealing.

IV. MOVEABLE VIRTUAL MACHINES

Consider now the possibility of migrating VMs from server to server. Assume that such moves can be carried out instantaneously (e.g., see [6], [5]). That assumption will allow us to derive an exact solution for the model with n servers and K job classes. In reality migration is not instantaneous, but the delays associated with it are often acceptable (see [17]). Also, the exact solution corresponding to instantaneous migrations provides a performance bound for the model without migrations.

The advantage of migrating VMs is that, as jobs depart and release resources, those remaining can be packed down, thus

Fig. 3. Shared and dedicated servers: increasing total load
 $K = 3$, $B = 8$, $\mathbf{b} = (1, 3, 5)$, $\boldsymbol{\lambda} = \lambda(6/9, 2/9, 1/9)$, $\mathbf{r} = (1, 5, 10)$, $c = 3$

making better use of servers and reducing the probability of rejection. How to do this optimally, i.e. how to place a given set of jobs into the smallest possible number of servers, is an instance of the Bin-Packing problem, which is known to be NP-hard. However, there are simple heuristic allocations such as First-Fit-Decreasing (FFD), that have been shown to be quite close to optimal (see [3]).

In the case of a single shared resource, the job types should be numbered in ascending order of their requirements: $b_1 \leq b_2 \leq \dots \leq b_K$. When the numbers of jobs present in the system are $(j_1, j_2, \dots, j_K)$, the FFD packing algorithm works as follows:

- 1) Allocate the jobs of type K first. If $j_K < m_K$, where $m_K = \lfloor B/b_K \rfloor$, then all of them are placed in server 1; otherwise, if $s m_K \leq j_K < (s+1)m_K$, then the first s servers receive m_K jobs each and the $s+1$ st gets the rest.
- 2) Reduce the available resource in each server by the amount reserved so far; allocate the jobs of type $K-1$ as in step 1, subject to the new resource bounds.
- 3) Repeat step 2 for job types $K-2, K-3, \dots, 1$.

The FFD algorithm is applied at every arrival and departure instant. Incoming jobs that cannot be placed into any of the servers *after repacking*, are rejected and are lost.

One of the consequences of packing is that, given the vector $\mathbf{j} = (j_1, j_2, \dots, j_K)$ specifying the total numbers of jobs of various types present in the system, the numbers, $j_{i,s}$, of type i jobs present in server s ($i = 1, 2, \dots, K$, $s = 1, 2, \dots, n$) are determined uniquely. Hence, the vector $\mathbf{j}$ describes the system state fully.

Denote by $S(n, K, B)$ the set of admissible states for a system with n servers, K job types and resource bound B per

server. The evolution of the system state is a Markov process with instantaneous transitions from state $\mathbf{j}$ to state $\mathbf{j} + \mathbf{e}_i$ with rate λ_i and state $\mathbf{j}$ to state $\mathbf{j} - \mathbf{e}_i$ with rate $j_i \mu_i$. Note that these are the same transitions, and the same rates, that governed the single-server process in section II. The difference now is in the size and composition of the state space.

Repeating the arguments in section II, we conclude that the steady-state probability, $\pi(\mathbf{j})$, that the system is in state $\mathbf{j}$, is given by expression of the same form as (2):

$$\pi(\mathbf{j}) = \frac{1}{G(n, K, B)} \prod_{i=1}^K \frac{\rho_i^{j_i}}{j_i!} ; \quad \mathbf{j} \in S(n, K, B) . \quad (19)$$

The new normalization constant, $G(n, K, B)$, depends, like the new state space, on n as well as on K and B . The value of $G(n, K, B)$ can again be computed recursively, but the algorithm is a little more complicated and requires a different notation.

Let $S(n, K, \mathbf{B})$ be the set of admissible state vectors $\mathbf{j}$ when the amounts of resource available at servers $1, 2, \dots, n$ are given by the vector $\mathbf{B} = (B_1, B_2, \dots, B_n)$. Denote by $G(n, K, \mathbf{B})$ the corresponding normalization constant:

$$G(n, K, \mathbf{B}) = \sum_{\mathbf{j} \in S(n, K, \mathbf{B})} \prod_{i=1}^K \frac{\rho_i^{j_i}}{j_i!} . \quad (20)$$

Let $m_i(\mathbf{B})$ be the largest possible number of type i jobs that can be accepted into the system when the resource availability is given by the vector $\mathbf{B}$. If $j \leq m_i(\mathbf{B})$ jobs of type i are to be allocated, let $\mathbf{d}_i^{(j)} = (d_{i,1}^{(j)}, d_{i,2}^{(j)}, \dots, d_{i,n}^{(j)})$ be the amounts of resource that will be used in servers $1, 2, \dots, n$ under the FFD

packing algorithm. Then we can write a recurrence relation similar to (4):

$$G(n, K, \mathbf{B}) = \sum_{j=0}^{m_K(\mathbf{B})} \frac{\rho_K^j}{j!} G(n, K-1, \mathbf{B} - \mathbf{d}_K^{(j)}) . \quad (21)$$

The terminating conditions are $G(n, 0, \mathbf{B}) = 1$ and $G(n, K, \mathbf{B}) = 1$ if the corresponding state space $S(n, K, \mathbf{B})$ contains only the vector $\mathbf{j} = \mathbf{0}$.

The desired normalization constant, $G(n, K, B)$, is equal to $G(n, K, \mathbf{B}_0)$, where $\mathbf{B}_0 = (B, B, \dots, B)$ (i.e., all servers are fully available).

The states in which an incoming job of type i would be accepted are those in which there is at least one server where the currently available resource is at least b_i . Since lower-numbered servers are packed before higher-numbered ones, if there are any such servers then server n is one of them. Consequently, the states in which an incoming job of type i is accepted are precisely those where the currently available resource in server n is at least b_i . Therefore, the steady state probability, $1 - \beta_{i,n}$, that an incoming job of type i is accepted is given by

$$1 - \beta_{i,n} = \frac{G(n, K, \mathbf{B}_0 - b_i \mathbf{e}_n)}{G(n, K, \mathbf{B}_0)} , \quad i = 1, 2, \dots, K , \quad (22)$$

where $\mathbf{e}_n$ is the n -vector whose n 'th element is 1 and all others are 0.

One can now use expression (11) to evaluate the long-term average profit, R , achieved by the n servers per unit time.

When there is more than one resource to be shared, the job packing problem becomes multidimensional. Several heuristic algorithms of varying complexity exist (e.g., see [5]). For our purpose, it does not really matter how jobs are packed, as long as the following properties are satisfied: (i) the algorithm is fast enough so that it can be applied at every arrival and departure instant; (ii) the state vector $\mathbf{j}$ uniquely determines the numbers $j_{i,s}$ of type i jobs at server s . One can then write equations similar to (20) - (22) and use them to compute the achievable profit.

Intuitively, the packing of jobs should lead to a more efficient use of servers, lower rejection probabilities and higher profits. Hence, the exact solution of the n -server system with packed jobs should provide an upper bound for the achievable profit in the system without packing.

In fact, it turns out that this upper bound is also an excellent approximation. As an illustration, figure 4 shows the average long-term profit R as a function of n , for the FFoA policy without packing (estimated and simulated values, plus 90% confidence intervals), and the FFD policy with packing (exact solution). The parameter values are as shown in the figure caption. The resource requirements and bound correspond to the M3 family of VM instances offered by the Amazon EC2 service (see [20]). There are now four job types (adding an 'extra-large' type) and bigger servers with a vCPU resource bound of 16.

The figure confirms that the packing of VMs leads to higher profits. However, the advantage gained is very marginal. The

three plots in figure 4 are remarkably close to each other. In particular, they all indicate the same optimal number of servers. It seems that allocating incoming jobs to the first server that has room for them, and then leaving them in place, has a similar effect to packing.

To emphasize the above observations, in figure 5 we have plotted the optimal achievable profit under the FFoA policy without packing (estimated and simulated, with 90% confidence intervals), and the FFD packing policy (exact), for increasing total arrival rate. The individual arrival rates of the four job types are kept in fixed proportions.

The three plots are so close as to be visually indistinguishable.

The corresponding optimal numbers of servers, n^* , are shown in table 2.

TABLE II
OPTIMAL NUMBERS OF SERVERS

λ	5	10	15	20	25	30	35
FFoA n^*	2	3	5	6	7	9	10
FFD n^*	2	3	5	6	7	8	10
Sim n^*	2	3	5	6	7	9	10

The only disagreement between the two policies is a 1-server difference in the predicted optima for $\lambda = 30$. The effect on the achievable profit is very small.

In fact, this experiment (and others that we have carried out), strongly suggests that one need not bother with packing. The best use for the results in this section is to provide a simpler approximation of the achievable profit (which is also a tight upper bound), for purposes of optimization.

V. CONCLUSIONS

We have provided easily implementable expressions for computing the expected profit in an n -server system where multiple job types share bounded resources. These expressions enable the evaluation of the optimal number of servers. The results are exact in the cases of a single server, a number of dedicated servers, or a number of shared servers where VMs are packed at arrival and departure instants. Empirical and simulation results have shown that (a) the estimations used to evaluate the profit of the FFoA policy without packing are accurate, and (b) packing yields very minimal improvements in profits.

Our methodology could, in principle, be used by a service center operator in order to decide how many servers to keep powered on. The limits to applicability in that area would come from the size of the state space: service centers tend to house thousands of servers, which might make the computation of the normalizing constant difficult.

A remaining open problem is to find an efficient way of computing the exact solution of the n -server model under the FFoA policy without packing. We do not know whether a product-form solution exists or not. However, it has to be said that this is more of theoretical interest than practical, since

Fig. 4. Packed and not packed VMs: increasing number of servers

$K = 4$, $B = 16$, $\mathbf{b} = (1, 2, 4, 8)$, $\boldsymbol{\lambda} = (6, 4, 2, 1)$, $\boldsymbol{\mu} = (1, 1, 0.5, 0.5)$, $\mathbf{r} = (1, 3, 6, 10)$, $c = 3$

Fig. 5. Optimal profit: increasing demand

$K = 4$, $B = 16$, $\mathbf{b} = (1, 2, 4, 8)$, $\boldsymbol{\lambda} = \lambda(6/13, 4/13, 2/13, 1/13)$, $\boldsymbol{\mu} = (1, 1, 0.5, 0.5)$, $\mathbf{r} = (1, 3, 6, 10)$, $c = 3$

the estimates and bounds that we have provided are highly accurate.

On the other hand, there is a related topic on which we have not touched, but which deserves attention: instead of rejecting jobs that cannot be accommodated on arrival, they might be queued according to their type. Different priorities may be assigned to those queues. Such a set-up would lead to a multi-class, multi-server priority queueing model with bounded shared resources. There are no existing results in that area (not even for the case of a single shared server), but it

would be an interesting topic for future research.

REFERENCES

- [1] E. Arzuaga and D.R. Kaeli, "Quantifying Load Imbalance on Virtualized Enterprise Servers", Procs. First Joint WOSP/SIPEW Int. Conf. on Performance Engineering (WOSP/SIPEW'10), pp. 235-242, 2010.
- [2] P. Bodík, R. Griffith, C. Sutton, A. Fox, M. Jordan and D. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters", Conf. on Hot Topics in Cloud Computing (HotCloud'09), Berkeley, CA, USA, 2009.
- [3] E.G. Coffman, J. Csirik, and G. Woeginger, *Approximate Solutions to Bin Packing Problems*, Oxford University Press, 2002.

- [4] P. Ezhilchelvan and I. Mitrani, "Static and Dynamic Hosting of Cloud Servers", *Computer Performance Engineering LNCS 9272*, Eds. M. Beltran, W. Knottenbelt and J. Bradley, Springer, 2015.
- [5] E. Feller, L. Rilling and C. Morin, "Energy-Aware Ant Colony Based Workload Placement in Clouds", 12th IEEE/ACM Int. Conf. on Grid Computing (GRID'2011), pp. 26-33, 2011.
- [6] C. Ghribi, M. Hadji and D. Zeglache, "Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms", 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 671-678, 2013.
- [7] J.S. Kaufman, "Blocking in a shared resource environment", *IEEE Trans. Commun.*, 29, pp. 1474-1481, 1981.
- [8] M. Mazzucco, D. Dyachuk, and M. Dikaiakos, "Profit-aware server allocation for green internet services", *IEEE Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 277-284, 2010.
- [9] M. Mazzucco, M. Vasar, and M. Dumas, "Squeezing out the cloud via profit-maximizing resource allocation policies", *IEEE Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 19-28, 2012.
- [10] I. Mitrani, *Probabilistic Modelling*, Cambridge University Press, 1998.
- [11] J. Moore, J. Chase, P. Ranganathan and R. Sharma, "Making Scheduling 'Cool': Temperature-aware Workload Placement in Data Centers", *Procs. USENIX Annual Technical Conference (ATEC'05)*, pp. 61-74, 2005.
- [12] J.W. Roberts, "A service system with heterogeneous user requirement", in *Performance of Data Communications Systems and Their Applications*, (Ed. G. Pujolle), North-Holland, pp. 423-431, 1981.
- [13] K.W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*, Springer-Verlag, 1995.
- [14] A. Singh, M. Korupolu and D. Mohapatra, "Server-storage virtualization: Integration and load balancing in data centers", *Procs. 2008 ACM/IEEE Conference on Supercomputing (SC'08)*, pp. 53:1-53:12, 2008.
- [15] Q. Tang, S.K.S. Gupta, and G. Varsamopoulos, "Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach", *IEEE Trans on Parallel and Distributed Systems*, 19, 11, pp. 1458-1472, 2008.
- [16] R. Urgaonkar, U. C. Kozat, K. Igarashi and M. J. Neely, "Dynamic Resource Allocation and Power Management in Virtualized Data Centers", *IEEE/IFIP NOMS 2010*, Osaka, Japan, 2010.
- [17] W. Voorsluys, J. Broberg, S. Venugopal and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation", *LNCS*, vol. 5931, pp. 254-265, 2009.
- [18] S. Weijia, X. Zhen, C. Qi and L. Haipeng, "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing", *IEEE Transactions on Computers*, 63, 11, pp. 2647-2660, 2014.
- [19] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration", *Procs. 4th USENIX Conference on Networked Systems Design and Implementation (NSDI'07)*, pp. 229-242, 2007.
- [20] <https://aws.amazon.com/ec2/instance-types/>, Amazon Web Services, 2016.