On Virtual Network Functions' Placement in Future Distributed Edge Cloud

Farah Slim, Fabrice Guillemin, Yassine Hadjadj-Aoul

On Virtual Network Functions' Placement in Future Distributed Edge Cloud

Farah Slim
Orange Lab. (France)
Email: farah1.slim@orange.com

Fabrice Guillemin
Orange Lab. (France)
Email: fabrice.guillemin@orange.com

Yassine Hadjadj-Aoul
University of Rennes 1 (France)
Email: yhadjadj@irisa.fr

***Abstract*—Network Function Virtualization (NFV) is changing the way networks are managed, by providing more scalability and flexibility. However, to meet stringent real-time constraints, some network functions have to be hosted close to end users, which incites Network Operators (NOs) to install well dimensioned data centers at the edge of the network. In this framework, the contributions of this paper are twofold. First, we proposed an analytical model for the blocking analysis in a multidimensional cloud system, which was validated using discrete events' simulations. Second, we conducted a comparative analysis of the most popular placement's strategies. The proposed model, as well as the comparative study, reveal practical insights into the performance evaluation of resource allocation and capacity planning for distributed edge cloud with limited capacities.**

***Keywords*—Edge cloud; placement; NFV; blocking; Openstack.**

I. Introduction

In the recent few years, cloud and virtualization have enabled the emergence of the virtualization of network functions (NFV) allowing network operators to decouple network functions from proprietary hardware appliances so that such functions can run in software [2]. The virtualization of network functions is certainly a major revolution that will impact not only the architecture of networks but also Network Operators (NOs) infrastructures. In fact, to meet stringent real-time constraints[6], some network functions have to be hosted close to end users (e.g. Radio Access Network (RAN) functions).

These latency requirements lead to the development of geographically distributed mini data centers, also referred to as cloudlets [4], at the edge of the network (i.e., typically at Points of Presence (PoPs) level). These edge data centers have rather small capacities in terms of storage, compute and networking resources when compared against huge centralized data centers deployed, for instance, by Google[1] or Amazon [5].

All these radical changes in NOs' infrastructures raise many new issues (especially in terms of resource allocation), which have so far not been considered in the cloud literature. Traditionally, resources in cloud platforms are considered as to be infinite and request blocking is most of the time ignored when evaluating resources' allocation algorithms, precisely because of this infinite capacity assumption [8]. However, if we assume that the NO's infrastructure will very likely be composed of small data centers with limited capacities, and deployed at the edge of network, the congestion of such a system may occur, notably if the demand is sufficiently high and exceeds what the infrastructure can handle at a given time.

Analysis of blocking in cloud edge data centers requires relevant models comprising a large number of parameters such as several types of requests at the cloud system and heterogeneous resources of the system. To the best of our knowledge, these features all together are not available in any of the existing models of blocking analysis in cloud systems. It is noteworthy that most papers in the cloud literature have instead focused on resource allocation algorithms.

If the demand for resources were perfectly known in advance, the problem can be formulated as an ILP problem, whose resolution can last for hours, notably because of the multidimensional nature of the system [1]. In order to overcome the difficulties raised by the multidimensional nature of the problem, several solutions have been proposed in the literature, in particular via the introduction of some heuristics reducing the complexity of the problem.

In this context, the First Fit-Decreasing (FFD) heuristic consists of ordering the items (VMs) and the bins (PMs) by size. Starting with the first bin, it iterates over the items and places an item in a bin when possible. Once the first bin is filled, it proceeds to the second bin from the ordered list, repeating the same process until acceptance or rejection of the request. When there is only one type of resource, then FFD can easily be applied. But, generally the placement problem is constrained by more than a single resource (CPU, memory, disk). Classically, to generalize FFD to a multidimensional scenario, the multidimensional vector of PM capacities is mapped onto a single scalar, referred to as metric.

The metric used by the cloud management platform Openstack is considered as a score to select suitable hosts and calculated as:

$$score_{host} = \eta_1 * w_1 + \eta_2 * w_2 + ... + \eta_N * w_N,$$ [2]

where w_n is the normalized value of resource n (i..e, the ratio of the amount of the resource currently available to the total capacity of the resource) and η_n is a weight (a.k.a. multiplier)

[1] https://www.google.com/about/datacenters/inside/locations/index.html

[2] http://docs.openstack.org/developer/nova/filterscheduler.html

associated with this resource. Another metric, referred to as $sand_{volume}$ is defined in [9] as:

$$sand_{volume} = \frac{1}{1-w_{cpu}} \cdot \frac{1}{1-w_{mem}} \cdot \frac{1}{1-w_{net}},$$

where w_{cpu}, w_{net} and w_{mem} are the corresponding normalized utilization ratio of the corresponding resource.

The authors in [7] introduced another metric refereed to as dot product, which is the scalar product of resource requirements vector of the VM to be placed and the resource utilization vector of each PM. The PM with the lower dot product is chosen. The idea motivating this heuristic is to place the VM in a complementary PM. The intuition is that a small dot product indicates a large angle between the VM vector and the resource utilization vector of the PM. This is equivalent to say that a large angle means that VM and PM are complementary.

In this paper, we propose a model for estimating blocking in a system of distributed data centers. The organization of this paper is as follows. In Section II, we present our model and analyze the blocking in a multidimensional cloud system. In Section III, we, first, numerically validate our model. Then, we present a comparative analysis of the performance of several resources' allocation algorithms. Finally, the paper concludes in Section IV with a summary recapping the main contributions of the paper.

II. Analytical Model

We analyze in this section blocking in a multidimensional cloud system with multi-class arrival request.

A. Model Settings

We consider a data center composed of N servers. Each server comprises J types of resources (CPU, RAM , disk, bandwidth , etc.). We assume that all servers are identical. The capacity of a server is denoted by c_j for resource j. The data center capacity is then Nc_j in resource j.

The data center accommodates resource requests of K classes. The demand in resource j of a class k request is denoted by A_j^k. We assume that requests of class k arrive according to a Poisson process with rate λ_k. The mean holding time of resources by a request of class k is denoted by $1/\mu_k$. To simplify the analysis we assume that the nominal request A_j^k has integer values. Moreover, we assume that the greatest common divisor of A_j^k for $k = 1, \ldots, K$ and fixed j is equal to 1.

The load of the system in resource j is

$$\rho_j = \frac{1}{Nc_j} \sum_{k=1}^{K} \rho_k A_j^k,$$

where $\rho_k = \lambda_k/\mu_k$. The model under consideration is illustrated in Figure 1

In practice, upon the arrival of a request, the scheduler, aware of the occupation of different servers in the data center, forwards the request to one of the available servers that can accommodate the request. The selection of the server is made according to a given algorithm. If the requested amount of resources is not available in all servers, the request is then blocked.

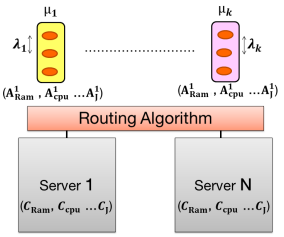

Fig. 1. Model settings

Due to the fragmentation of the resources among servers, there is a potential loss of efficiency. For a given request, the claimed amount of resources may be globally available but since the resources are fragmented it may happen that none of the server can accommodate the request.

The routing algorithm may have a major impact on the performance of the system. To study the impact of the routing algorithm on the global blocking of the system, we consider that the N servers are grouped into a unique big server, since there is no loss. Then we analyze the blocking for this system in order to evaluate subsequently the efficiency of different routing algorithms by comparing global blocking rates.

B. Blocking in a grouped data center: One Big Server

If we assume that the N servers can be grouped into a unique big Server, with capacity Nc_j for resource j, then we obtain a classical blocking system with heterogeneous resources. If we consider the system in equilibrium, let $\mathbf{n} = (n_1, \ldots, n_K)$ denotes the occupation of the server when there are n_k customers of class k in the system. The resource constraints translate into

$$\sum_{k=1}^{K} n_k A_j^k \leq Nc_j$$

for $j = 1, \ldots, J$.

The probability of being in state $\mathbf{n}$ is

$$\mathbb{P}(n) = \frac{1}{\mathcal{G}} \prod_{k=1}^{K} \frac{\rho_k^{n_k}}{n_k!},$$

where $\mathcal{G}$ is the normalizing constant given by

$$\mathcal{G} = \sum_{\mathbf{n} \in \mathcal{S}} \prod_{k=1}^{K} \frac{\rho_k^{n_k}}{n_k!},$$

the state space $\mathcal{S}$ being defined by

$$\mathcal{S} = \left\{ \mathbf{n} \in \mathbb{N}^K : \sum_{k=1}^{K} n_k A_j^k \leq Nc_j,\ j = 1, \ldots, J \right\}.$$

The state space $\mathcal{S}$ is delimited by J hyper-planes. We easily note that if $A_k^j \leq A_k^{j'}$ and $C_j \geq C_{j'}$ then the condition

$\sum_{k=1}^{K} n_n A_j^k$ is dummy. In the following we assume that this situation does not occur. Otherwise, we have to consider smaller number of resources $J' < J$ but the analysis is similar.

In the following we assume that N is large and we take N as a scaling factor. In other words, we replace λ_k by $N\lambda_k$ for the arrival rate of class k customers. On the basis of classical asymptotic methods developed in the context of circuit-switched for large networks (see fro instance [3]), we deduce the following estimates for the blocking probability β_k for class k customers under 3 different load conditions:

- **Underload conditions** If $\sum_{k=1} \rho_k A_j^k < c_j$, for large N

$$\beta_k = \frac{1}{\sqrt{2\pi N}} \sum_{j, A_j^k \neq 0} \frac{e^{-N.I_j}}{\sqrt{\Gamma_j}} \frac{1 - e^{-y_j A_j^k}}{1 - e^{1 - e^{-y_j}}} \left[1 + O\left(\frac{1}{\sqrt{N}}\right)\right]$$

with

$$\begin{aligned} I_j &= \sum_k \rho_k \left(1 - e^{-A_j^k y_j}\right) - C_j y_j, \\ \Gamma_j &= \sum_k \rho_k A_j^{k\,2} e^{-A_j^k y_j} \end{aligned}$$

and $y_j < 0$, $j = 1, \ldots, J$ being the solution to the linear system

$$\sum_{k=1}^{K} \rho_k A_j^k e^{-A_j^k y_j} = C_j, j = 1, \ldots, J; \qquad (1)$$

- **Critical conditions:** If $\sum_{k=1} \rho_k A_j^k = c_j$,

$$\beta_k = \frac{1}{\sqrt{2\pi N}} \sum_{j, A_j^k \neq 0} \frac{A_j^k}{\sqrt{\Gamma_j}} \left[1 + O\left(\frac{1}{\sqrt{N}}\right)\right];$$

- **Overload conditions:** When $\sum_{k=1} \rho_k A_j^k > c_j$ and the linear system (1) has positive solutions,

$$\lim_{N \to \infty} \beta_k = 1 - \prod_{j, A_j^k \neq 0} e^{-y_j A_j^k}.$$

III. Simulations Results

A. Numerical Validation of the proposed model

We propose in this section a quantitative evaluation of the blocking probability estimation obtained with our model. For comparison, results are provided via simulation where we consider a two-dimensional system (RAM and CPU resources) and two classes of requests. One class is composed of requests with small requirements in terms of CPU and RAM (referred to as mice). The requests of the second class (referred to as elephants) have high requirements in terms of CPU and RAM.

The arrival process of class 1 requests is assumed to be Poisson with rate λ_1. A class 1 customer requires c_1 units of CPU and r_1 units of RAM. The holding time of resources is assumed to be exponential with mean $1/\mu_1$. Similarly, class 2 requests arrive according to a Poisson process with rate λ_2, require c_2 units of CPU and r_2 units of RAM, and hold the resources for exponentially distributed duration with mean $1/\mu_2$.

We consider a data center with two identical servers with capacity C in terms of CPU and R in terms of RAM. We assume that the two servers are grouped into a unique big server with capacity $2R$ in terms of RAM and $2C$ in terms of CPU. We obtain via simulation blocking probabilities for both mice and elephant classes. Simulation results are averaged to obtain confidence intervals with a 95% confidence level.

Figure 2 displays the blocking probability versus traffic intensity under three different regimes: underload, critical and overload. Results illustrate the good accuracy of the proposed analytical model through blocking probability estimation for both classes and under the different regimes.

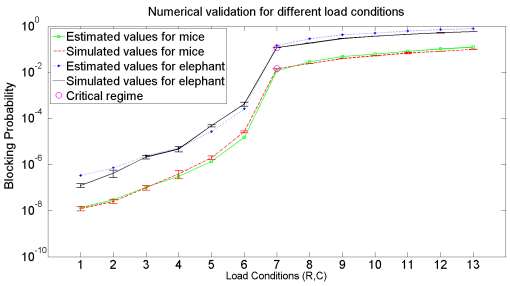

Fig. 2. Numerical Validation Of the Model .

B. Evaluation of resource allocation algorithms

In this section, we evaluate via simulation the efficiency of the various routing algorithms presented in Section I by comparing the blocking probabilities values to those obtained in the case where we consider servers grouped into one big server. Depending on the number of arrival classes, we have studied two scenarios.

1) Two arrival Classes: The system we have first considered is a data center composed of 2 servers with identical capacities and 2 arrival classes of requests referred to as Mice and Elephants as in the previous section.

a) Simulations Settings: To study the performance of the system, we have considered the overloaded conditions. We have also considered the underloaded as well as the critical regime; the results are qualitatively the same. Parameter values for overloaded conditions are given in Table I. The loads are $\rho_{CPU} = 1.1607$ and $\rho_{RAM} = 1.2600$.

TABLE I
Parameter values for overloaded conditions.

parameter	value
$(\lambda_1, \mu_1, c_1, r_1)$	(70,1,2,3)
$(\lambda_2, \mu_2, c_2, r_2)$	(30,1,17,35)
(C, R)	(280,500)

b) Simulations Results: The blocking probabilities for the two classes of requests and the various algorithms are given in Figure 3. We verify that the blocking rates for elephants are greater than those for mice but the blocking rates are not

significantly different from one algorithm to the other; there are variations but not by an order of magnitude.

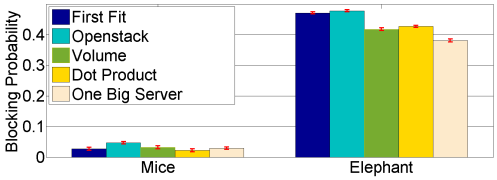

Fig. 3. Blocking Rates in an Overloaded System .

To further analyze the system in terms of blocking, we compare the system with two servers against one big server with capacity equal to the sum of the capacities of the two servers. We observe that the blocking rates are sightly different but of the same order of magnitude. These observations hold for all the simulation experiments we have performed for this kind of system. Hence, to qualitatively analyze the system with two servers of capacity C, it suffices to consider a system with a unique server of capacity $2C$.

This means that the fragmentation of resources into various servers has no impact as long as individual requests are small when compared to server capacities. This is a key fact for qualitatively estimate blocking in cloud platforms because the analysis of large multidimensional systems can be analyzed by using classical methods used in the context of circuit switched networks as we proposed in Section 1.

2) Four arrival Classes System:

a) Simulations Settings: Based on class specification of a popular cloud platform, we have defined 4 arrival types of requests with different resource requirements as shown in Table II. Arrival rate as well as holding times shown in Table II are set based on the proportion of each class in the system.

We have performed extensive simulations under different load conditions; the results are qualitatively the same. For the sake of conciseness, we report in the paper only simulation results for the underloaded conditions. We have varied the number of servers composing the system ($N = 2, ..., 10$) while conserving the same load conditions; the system is underloaded with load values $\rho_{CPU} = 0.9444$ and $\rho_{RAM} = 0.8897$.

TABLE II
PARAMETERS OF ARRIVAL CLASSES.

	A	B	C	D
RAM requirement (GB)	4.096	8.192	16.384	32.768
CPU requirement (cores)	1	2	4	8
Proportion	67%	22%	8%	3%

b) Simulations Results: Figure 4 displays the blocking probabilities for each class under the underloaded conditions. As in the previous scenario, we can verify that there is no significant difference between the various algorithms. We also note that as before, comparing these values against those obtained via simulation of a big unique server validates our proposed model in the sense that the blocking rates are similar. This opens the door to the analysis of a system composed of many servers by considering a unique big server whatever be the resource allocation algorithm in the multi-server system. Such an analysis is sufficient for dimensioning purposes; in practice only a rough estimates of blocking rates are sufficient.

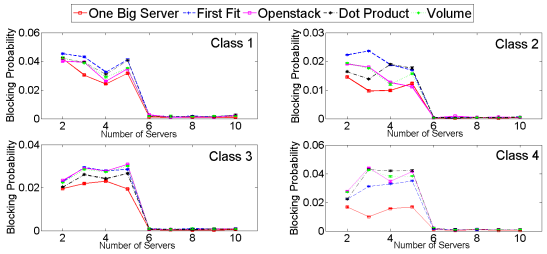

Fig. 4. Blocking Rates in an Underloaded System .

IV. CONCLUSION

We have evaluated resource allocation performance in cloud systems with finite capacity by paying special attention to blocking of requests in a probabilistic context. The key observation of this paper is that with regard to request blocking there is no noticeable difference between the various placement algorithms so far considered in the literature. In fact, we show that blocking rates are similar to that obtained when considering a global data center with a capacity equal to the sum of servers capacities. This model is able to accurately estimate the blocking probability in multidimensional cloud systems.

It turns out that if we have several data centers disseminated in the network and if it is possible to know their occupancy upon each request, then everything happens as if the network had a unique big data center with a unique server with capacity equal to the sum of all capacities. This gives a means of estimating request blocking at a network scale and eventually a simple method of dimensioning a system of data centers for a given demand.

REFERENCES

[1] Dhaval Bonde. Techniques for virtual machine placement in clouds. *Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, Mumbai.*, 2010.

[2] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[3] Ravi R Mazumdar. *Performance modeling, stochastic networks and statistical multiplexing*. Morgan & Claypool, 2013.

[4] Ranjan Pal, Sung-Han Lin, and Leana Golubchik. The cloudlet bazaar dynamic markets for the small cloud. *arXiv preprint arXiv:1704.00845*, 2017.

[5] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[6] Philipp Schulz, Maximilian Matthe, Henrik Klessig, Meryem Simsek, et al. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, 2017.

[7] Aameek Singh, Madhukar Korupolu, and Dushmanta Mohapatra. Server-storage virtualization: Integration and load balancing in data centers. *In Proceedings of the ACM/IEEE conference on Supercomputing*, 2008.

[8] Jiayi Song and Roch A Guérin. Pricing and bidding strategies for cloud computing spot instances. 2017.

[9] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and gray-box strategies for virtual machine migration. *Proc. of the 4th USENIX conference on Networked systems design and implementation*, 2007.