# Resource Allocation Mechanism for a Fog-Cloud Infrastructure

Rodrigo A. C. da Silva, Nelson L. S. da Fonseca
Institute of Computing
State University of Campinas, Brazil
Email: rodrigo@lrc.ic.unicamp.br, nfonseca@ic.unicamp.br

***Abstract*—Fog computing brings the cloud close to the users, reducing latency and allowing the deployment of new delay-sensitive applications. Fogs and clouds can work cooperatively to improve service delivery to the end users. An essential aspect of a fog-cloud system is the decision-making process on where to allocate resources to run the tasks of an application. This paper introduces a novel mechanism named Gaussian Process Regression for Fog-Cloud Allocation (GPRFCA) for resource allocation in infrastructure composed of cooperative fogs and clouds. The GPRFCA mechanism employs a Gaussian Process Regression to predict future demands in order to avoid blocking of requests, especially delay-sensitive ones. Results show that the GPRFCA mechanism reduces energy consumption, blocking as well as latency.**

## I. Introduction

The number of devices connected to the Internet, such as smartphones, is expected to grow two to three orders of magnitude in the near future [1], considerably increasing computation and communication demands. Cloud computing has been employed to cope with such demands due to the scalability and elasticity of service provided. However, several new applications, such as augmented reality and those used by connected vehicles, require near real-time processing, which cannot be provided by distant data centers in the cloud.

A new paradigm has been proposed to bring computing, storage and network resources close to the users, known as fog computing [1]. Fog computing will support the quality of service requirements of real-time applications by reducing latency and avoiding transfer of data and code through the Internet to cloud data centers. By bringing cloud services to the network edge, fog computing will complement cloud computing. Fog computing will not replace cloud computing since fog resources are limited.

The Openfog Consortium has proposed a reference architecture [2] composed of several layers. The number of layers and their position depend on the users' needs. One of the possibilities to implement fog nodes is to use mini data centers known as cloudlets [3]. Solutions for managing this infrastructure employ scheduling and resource provisioning mechanisms [4], [5]. A typical management issue in a fog-cloud infrastructure is where a workload should be processed. Such a decision must consider resource availability on fog nodes and the latency for processing the workload in the fog as well as the latency for processing in the cloud.

This paper proposes the Gaussian Process Regression for Fog-Cloud Allocation (GPRFCA) mechanism to answer the question of where to run the tasks of an application. The infrastructure considered is composed of a fog layer and the cloud. Users submit requests to the fog nodes and the workload can be executed in the fog, in the cloud or partially executed in the fog and in the cloud. The GPRFCA mechanism decides where to schedule a workload to be processed taking into account the resource availability and the latency overhead.

To improve the energy efficiency of fog nodes, reducing costs for the provider, the Power Usage Effectiveness (PUE) metric is considered in the decision. Since the PUE metric of data centers is higher than the PUE in the fog, the mechanism decreases the energy consumption by decreasing the demand on cloud data centers.

To optimize the utilization of limited fog resources, a Gaussian Process Regression is employed to predict the arrival of future requests based on the history of arrivals. Such prediction helps the provisioning of resources to future requests, especially the real time ones which can only be processed in the fog, in an attempt to reduce blocking and improve overall utilization.

None of existing work considered the history of previous requests for resource allocation. Results derived by simulation show that the employment of the proposed solution balances the workload between the fog and the cloud, providing a good trade-off between energy consumption, latency, and blocking.

The remaining of this paper is organized as follows. Section II reviews related work. Section III describes the considered system model. Section IV introduces the Gaussian Process Regression for Fog-Cloud Allocation mechanism. Section V describes the evaluation of GPRFCA. Finally, Section VI concludes the paper and suggests future directions.

## II. Related work

This section reviews existing proposals for scheduling and resource allocation in fog-cloud systems with emphasis on those dealing with energy efficiency. Fog services demand management to meet users' needs while ensuring efficient resource usage. In this context, Masip-Bruin et al. [6] proposed that fogs and clouds should be organized in a fog-to-cloud (F2C) layered architecture which coordinates cloud, fog nodes, and users. This architecture speeds up tasks by enabling parallel execution of workload on the available layers.

Solutions for resource allocation using the F2C architecture were presented in [7] and [8]. In [7], a solution to support the QoS requirements of applications in cloud-fog scenarios was proposed. The solution attempts to reduce latency by favoring the allocation of resources on fogs rather than on clouds, considering an architecture composed of two fog layers and the cloud. The authors showed that the employment of multiple fog layers reduces the overall latency. The proposal [8] employs the same architecture and introduces resource allocation based on the knapsack problem to minimize latency as well as balancing the workload to reduce the energy consumption.

An architecture to unify fog nodes and clouds in 5G networks is proposed in [9]. This architecture coordinates fog nodes with the infrastructure by using a controller to centralize control and orchestration, allocating both fog, cloud, and network resources. Such an architecture allows the deployment of Network Functions Virtualization and Internet of Things (IoT) services over a 5G network. However, no mechanism is introduced for the allocation of resources in the fog nodes.

Concrete mechanisms for resource allocation are presented in [10], [11], [12]. Sarkar et al. [10] analyzed the employment of fog computing in the context of IoT. Results show that the greater the workload executed on the fog, the lower is the energy consumption, especially those related to computation and storage. The study in [11] attempts to achieve a trade-off between energy consumption and delay. An optimization problem and an approximate solution were proposed with this aim. By considering the power consumption of fog devices, cloud servers and the delay to reach the cloud, the authors showed that an interplay of cloud and fog can help to improve performance and energy consumption.

A simulator for fog computing, named iFogSim, was introduced in [12]. iFogSim allows the simulation of IoT devices, fog and cloud computing environments. Two use cases were presented: a delay-sensitive online game and an intelligent surveillance system. Results of cloud-only and cloud-fog allocations were analyzed, showing that latency and network utilization can be reduced when fog is used.

These papers propose solutions for the provisioning of resources in fog computing considering different aspects, but none of them explored prediction to manage the load on fog nodes while mitigating latency and energy consumption. This paper presents a resource allocation mechanism that explores the interplay of fog and cloud computing to guarantee quality of service as well as to improve the performance of the system.

## III. System model

The OpenFog Consortium has recently released guidelines to the design of fogs in its reference architecture document [2], which suggests that security, scalability, and reliability of fog nodes should be emphasized. One interesting aspect described in the reference architecture is the deployment of fog and cloud nodes in several layers. The system model considered in this paper has three layers, as shown in Figure 1: user devices, fog and cloud. The cloud is represented as a data center with

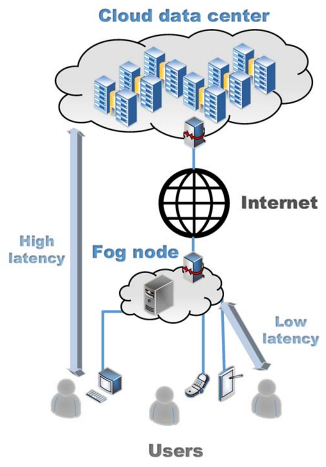


Fig. 1: Cloud and fog architecture.

plenty of physical servers, while the fog node is modeled as a cloudlet [3], capable of hosting virtual machines at a single hop distance from the final users. The third layer is composed of users devices and it is directly connected to the fog.

This system receives requests in the form of modularized applications. Each application contains one or more tasks and involves sensors and actuators. Sensors gather data either from the environment or from the users while actuators interact with users and perform actions. The tasks of an application perform specific processing of the application and receive data from a sensor or from another task. Resources are allocated to a task by instantiating a virtual machine for each task. The instantiation can be either in a fog node or in a cloud node.

This work considers two classes of tasks: tasks which virtual machines (VMs) must be instantiated in the fog and those which can be instantiated either in the cloud or in the fog. The former are delay-sensitive tasks that demand a quick response to the users, while the latter are tasks with flexible response time. Applications can have multiple tasks and they can be of different classes resulting, for example, in a request having part of its VMs served in the cloud and another part in the fog node. There are agents in the cloud and in the fog which decide on which physical server a virtual machine should be instantiated. If resource cannot be allocated, the request is blocked.

## IV. Proposed Mechanism

This section introduces the Gaussian Process Regression for Fog-Cloud Allocation (GPRFCA) mechanism. It analyzes the history of previously submitted requests for predicting future arrivals of virtual machines with strict latency requirements. Using this prediction, the GPRFCA mechanism can reserve resources in the fog node for future requests which must be executed in the fog, assigning tasks which are not delay-sensitive to the cloud. In this way, blocking can be reduced and the utilization of fog nodes increased.

Algorithm 1 presents the GPRFCA mechanism. CPU and RAM memory are considered allocable resources by the mechanism. Five parameters are given as input:

$\mathcal{H}$, $availableMIPS$, $availableRAM$, $mipsPerVM$, and $ramPerVM$. $\mathcal{H}$ is the set containing the historical dataset of the number of VMs arrived which could only be placed in the fog, i.e., $\mathcal{H} = h_1, h_2, \ldots, h_n$, where each $h_i$, $1 \leq i \leq n$, is an integer with the number of previous requested VMs. In other words, $h_1$ is the number of VMs that were submitted in the first minute, $h_2$ the number of submissions VMs in the second minute, and so forth. The time elapsed between sequential arrivals $h_i$ and $h_{i+1}$, $1 \leq i \leq n-1$ is 1 minute. All VMs require the same processing and memory resources. The mechanism, however, is not limited by this assumption and it can also handle heterogeneous demands. The parameters $availableMIPS$ and $availableRAM$ indicate the sum of the available number of instructions per second and the number of bytes of RAM memory in all servers in the fog, respectively. $mipsPerVM$ and $ramPerVM$ represent the requested CPU and RAM memory capacities, respectively, by each VM.

```
Input: $\mathcal{H}$, $availableMIPS$, $availableRAM$,
       $mipsPerVM$, $ramPerVM$
Output: Maximum number of virtual machines to be
        accepted in fog
1 $vmsMIPS \leftarrow \lceil availableMIPS/mipsPerVM \rceil$;
2 $vmsRAM \leftarrow \lceil availableRAM/ramPerVM \rceil$;
3 $numberVMs \leftarrow min(vmsMIPS, vmsRAM)$;
4 $futureVMs \leftarrow gaussianProcess(\mathcal{H})$;
5 return $max(0, futureVMs - numberVMs)$
```

**Algorithm 1:** Gaussian Process Regression for Fog-Cloud Allocation algorithm.

The algorithm starts by calculating the number of virtual machines $numberVMs$ that the fog node can still accommodate, considering CPU and memory (Lines 1 to 3). The next step is to invoke the Gaussian Process regression, Line 4, giving $\mathcal{H}$ as input, i.e., the historical dataset with the number of VM arrivals which could only be instantiated in the fog. The method $gaussianProcess$ performs the Gaussian process prediction, giving as output the predicted number of VMs, $futureVMs$, which can only be instantiated in the fog in the next interval. If the $numberVMs$ value is greater than the $futureVMs$ value, it is expected that the future occupation of the fog node will be sufficient to accommodate the future requests, then the output is $futureVMs - numberVMs$, otherwise it is zero, being the output the number of additional virtual machines that the fog will be able to accommodate.

The number of VMs arrived is used to update the dataset, given as input to the algorithm, which output is denoted by $N$. Then, the fog hosts all VMs which have strict latency requirements as well as up to $N$ additional VMs which could be hosted in the cloud; the remaining virtual machines are allocated in the cloud. The idea behind this procedure is twofold. The first is to ensure that delay-sensitive tasks are processed in the fog and the allocation of VMs will not prevent the acceptance of future requests. The second is to host a high number of virtual machines in the fog rather than in the cloud, minimizing overall latency.

The $gaussianProcess$ function receives as input a time series to make prediction based on the Gaussian Process Regression (GPR), explained next. Considering the observed values $X(t_i)$, $i = 0, 1, \ldots, n-1$ at $n$ time instants $t_i$, then $X(t_i) = f(t_i) + \varepsilon_i$, where $f(t_i)$ is a mapping function and $\varepsilon_i$ is an independent Gaussian noise with zero mean and variance $\sigma^2$. To make prediction, it is assumed that $f(t_i) \sim GP(m(t), k(t_i, t_j; \theta))$, where $GP$ is a stochastic Gaussian process with mean $m(t)$ and covariance function $k(t_i, t_j; \theta)$ with hyperparameters $\theta$. The decision of the covariance function depends on the actual covariance of the data. Previous analysis of cloud data centers [13] identified that the virtual machine arrival process can be modeled as a self-similar process. Furthermore, the work in [14], which employed GPR to predict traffic in real networks, identified that the rational quadratic (RQ) covariance function $k_{RQ}$ is suitable for modeling self-similar series. The covariance function $k_{RQ}$ was employed in this work and is given by:

TABLE I: Infrastructure configuration and virtual machine instance description.

| | PUE | Server specification | Delay to users |
|---|---|---|---|
| Fog | 1.0 | 2 servers Hp Proliant Dl380 G7<br>3.06 GHz Xeon X5675 processor<br>1 core, 4 Gb RAM | 10ms |
| Cloud | 1.4 | 100 servers Hp Proliant Dl380 G7<br>3.06 GHz Xeon X5675 processor<br>2 cores, 8 Gb RAM | 100ms |
| Virtual Machine | | 1000 MIPS, 256 Mb RAM | |

$$k_{RQ}(r; l, \alpha) = s^2 \cdot \left(1 + \frac{r^2}{2\alpha l^2}\right)^{-\alpha}$$

Three hyperparameters are used, namely the variance $s$, length-scale parameter $l$ and magnitude parameter $\alpha$. For self-similar series, $\alpha = 1 - H$ , where $H$ is the Hurst parameter of the series [14]. These hyperparameters are optimized by a local search algorithm that searches their values in an interval between 0 and 100 for $s$, 0 and four times the maximum value in $X(t_i)$ for $l$, and 0 and 0.5 for $\alpha$.

## V. Performance evaluation

In this section, the effectiveness of the GPRFCA mechanism is assessed.

### A. Simulation settings

The iFogSim simulator [12] was employed to simulate the scenario described in Section III, while the Gaussian Process Regression was implemented with gptools Python package [15]. Results were obtained by 30 different executions for each point in the graphs and using a 95% confidence interval derived by the independent replication method.

The cloud data center is modeled as a set of physical servers. As suggested in [16], mini data centers employed as fog nodes have similar hardware than that of servers in a cloud data center, but the number of physical machines is much smaller. The hardware configuration of servers and the virtual machine requirements are displayed in Table I.

Each task runs in a virtual machine hosted either in the fog or in the cloud. The mapping of a VM onto a physical server in the fog node or in the cloud data center is performed by a VM placement algorithm which chooses the first available server in sequential order to host a task, therefore consolidating the workload on few machines to save energy. If all servers are fully utilized and cannot host a VM, the task is blocked by the VM placement algorithm.

## B. *Evaluated policies*

To properly assess the GPRFCA mechanism, two other policies were simulated: cloudwards and fogwards [12]. Whenever a task can be hosted either in the data center or in a fog node, cloudwards always places it on the cloud while fogwards always places it on the fog if there are available resources, otherwise it places the task in the cloud. Since plenty of resources are available in the cloud, allocation in the data center is virtually always possible, and cloudwards does not send tasks to the fog unless it has strict latency requirements. For all policies, including the GPRFCA, tasks that require low latency are always created in the fog node when resources are available, otherwise the entire application request is blocked.

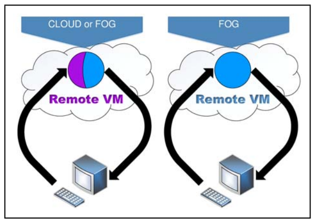
## C. *Workload*

The present work considers two applications: Remote VM and Augmented reality. Remote VM (Figure 2a) is a simple application in which a user access a terminal (thin client) and interacts with a remote operational system. User's actions are processed in a virtual machine and results are sent to the user's screen. Simulated scenarios have two classes of requests: requests with one task which can only be placed in the fog, and requests that can be deployed in the cloud.

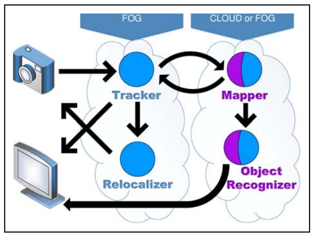
In augmented reality (Figure 2b), virtual objects are overlaid on an image acquired from a camera and then rendered on a display. The model used in this paper was presented in [17] and it is composed of six tasks: video source (a sensor fetching images from a camera), renderer (screen acting as an actuator), tracker (analysis of video frames), mapper (map generation and refinement), relocalizer (relocation of camera position) and object recognizer (localization of known objects). Video source and renderer are built on devices and do not require virtual machines. Tracker and relocalizer must be hosted on a fog node, and the mapper and object recognizer can be hosted either in the fog or in the cloud. The flow of exchanged data is shown in Figure 2b.

Requests are periodically submitted by users to the fog and their arrival is modeled as self similar series [13]. The algorithm in [18] is used to generate the number of requests per minute. Different series were created using different parameters for simulating different loads. The input for the generator in [18] is the values of the mean and standard deviation of the series as well as the Hurst parameter $H$. The employed scenarios as well as their parameters are shown in Table II, all employing $H = 0.7$.

All virtual machines deployed to satisfy the same request start processing at the same time and their lifetime is 5 minutes. Results were obtained for each application separately. In the simulation of remote VM application, half of the



(a) Remote VM



(b) Augmented reality

Fig. 2: Simulated applications. Each circle represents a task: blue represents exclusive fog tasks, while blue and purple tasks can be hosted either in fog or cloud. Arrows represent communication between tasks.

TABLE II: Simulated scenarios and parameters employed in the generator [18]. The standard deviation (SD) is half the value of the mean arrival rate.

| Scenario | Mean | SD | Scenario | Mean | SD |
|---|---|---|---|---|---|
| m2 | 2 | 1 | m12 | 12 | 6 |
| m4 | 4 | 2 | m14 | 14 | 7 |
| m6 | 6 | 3 | m16 | 16 | 8 |
| m8 | 8 | 4 | m18 | 18 | 9 |
| m10 | 10 | 5 | m20 | 20 | 10 |

requests must be hosted in the fog and the other half can be hosted either in the fog or in the cloud. For the augmented reality application such a distinction is not necessary since it is already formed by tasks of different classes.

## D. *Energy consumption model*

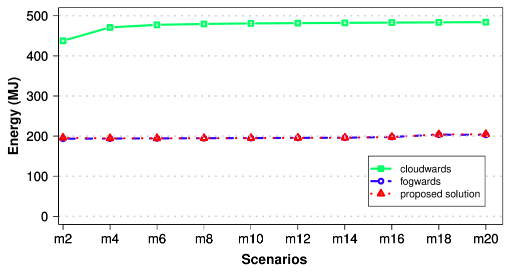
The energy consumed in the fog node and in the cloud data center is accounted as the sum of the consumption produced by their physical servers, which depends on the CPU usage. CPUs fully utilized lead to the highest server energy consumption, but idle servers consume about 70% of this value. A linear model is employed to model intermediary values of consumption [19]. The energy consumption model in this paper is based on a real benchmark of SPEC power, which measured power for different CPU usage levels [1]. These values are interpolated to calculate the energy consumption, considering current load imposed by the hosted VMs. Reference values for the employed server are presented in Table III.

As suggested in [16], cloudlets present similar hardware to those of cloud data centers and the energy model to estimate the consumption of their physical servers can be the same. However, the fog does not employ the same cooling and network equipment. Therefore, the PUE values differ; it is 1.4 for the cloud and 1.0 for the fog.

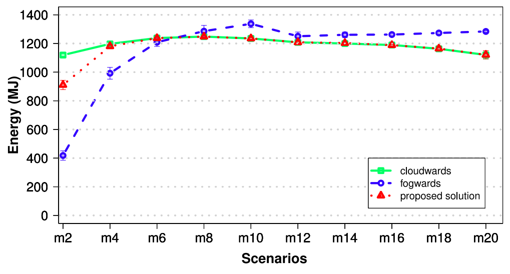
[1]https://www.spec.org/power_ssj2008/

TABLE III: Power consumption of cloud and fog server

| **CPU Load (%)** | **0** | **10** | **20** | **30** | **40** | **50** |
|---|---|---|---|---|---|---|
| **Consumption (W)** | 52.3 | 93.6 | 106 | 116 | 126 | 136 |
| **CPU Load (%)** | **60** | **70** | **80** | **90** | **100** | |
| **Consumption (W)** | 147 | 163 | 180 | 199 | 222 | |

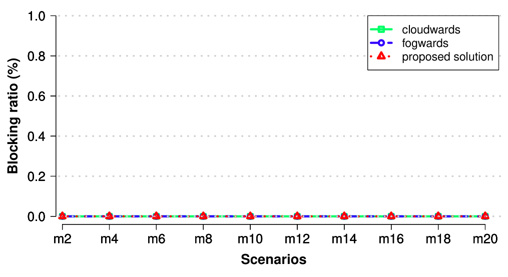


(a) Remote VM

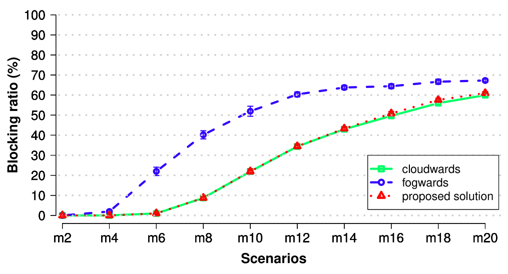


(b) Augmented reality

Fig. 3: Total energy for different applications.



(a) Remote VM



(b) Augmented reality

Fig. 4: Blocking ratio.

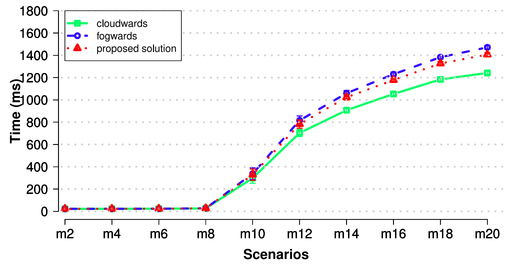
### *E. Numerical results*

Energy consumption, blocking ratio and latency are the metrics analyzed in this paper. The energy consumption was accounted for the fog and cloud infrastructures, considering the PUE metric for the cooling cost. Figure 3 shows the results for the energy consumption. For the remote VM application, fogwards produces the lowest energy consumption. Remote VM application requires only one virtual machine per request, thus it hardly causes an overload of resources in the fog, allowing the GPRFCA mechanism to obtain energy efficiency similar to that of fogwards. Augmented reality application has a different behaviour, since each request contains four virtual machines. Under low arrival rates, the fog manages to host several VMs, thus fogwards policy produces the lowest energy consumption. However, starting from the m8 scenario, the system becomes overloaded, forcing VMs to be hosted on the cloud. In these cases, the fogwards policy overloads the fog, increasing blocking. The GPRFCA mechanism, however, produces similar results to those of cloudwards.

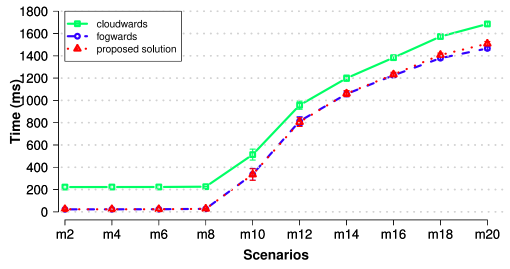
Figure 4 shows the blocking ratio, which is the ratio between the number of applications that were not instantiated and the number of submitted requests. Blocking happens whenever a virtual machine does not have its requirements satisfied due to the lack of physical resources in the fog. Remote VM produces no blocking due to the low demand of virtual machines. In contrast, augmented reality application, which submits four VMs per request, produces great blocking. Cloudwards produces lower blocking than does fogwards since it forwards all possible VMs to the cloud, where resources are abundant. Blocking under cloudwards occurs due to VMs which must be placed in the fog and there is no physical resources available for that. In fogwards, on the other hand, blocking happens because the fog node is overloaded with tasks that are not sensitive to delays. The GPRFCA mechanism tends to forward virtual machines to the edge, saving resources for future requests, which decreases the load on the fog node. As a consequence, it produces blocking ratio similar to that produced by cloudwards, while maintaining the energy consumption at a reasonable level.

Figure 5 presents the values for latency in the Remote VM application for tasks with strict latency requirements and also flexible tasks. The latency is the average time elapsed between an action of a user at the remote terminal (a pressed key or mouse movement) and rendering its results on the user's screen. Fogwards produces the smallest latency values since VMs are closer to end users. The GPRFCA mechanism produces latency values between those produced by the two other policies, but values are closer to those given by fogwards. Under load conditions equal or greater to that of the scenario m14, the latency is higher than 1 second, which can jeopardize the application performance. High latency values indicate that more resources in the fog should be deployed.

The advantage of employing the GPRFCA mechanism can be summarized as follows. The energy consumption is always between those produced by cloudwards and fogwards and it is always close to the lowest produced value by these two mechanisms. GPRFCA tries to produce high utilization in fog nodes and consequently reduces the energy consumption and blocking ratio. The GPRFCA mechanism does not increase the latency of requests which have VMs that can be placed



(a) Strict latency virtual machines which can be placed only on the fog



(b) Flexible virtual machines which can be placed either on the fog or on the cloud

Fig. 5: Latency for Remote VM application.

in the cloud. Moreover, it produces low latency for tasks that must be hosted in the fog.

## VI. Conclusion

This paper proposed a novel mechanism for the provisioning of resources in a fog and cloud environment named Gaussian Process Regression for Fog-Cloud Allocation mechanism, which aims at reducing the overall energy consumption while supporting latency requirements. Gaussian process regression was employed to estimate the demand of future arrivals so that fog nodes can host future requests, preventing blocking of new requests. Results obtained show that the GPRFCA mechanism maintains the energy consumption at acceptable levels and avoids overloading the fog reducing the blocking of requests. Latency was also kept at reasonable levels and decreased for applications with flexible latency requirements. The GPRFCA mechanism manages to take advantage of the good features of both cloudwards and fogwards mechanisms.

Multiple coordinated fog and cloud layers with different latencies are an asset of fog architectures according to the OpenFog Consortium [2]. The consideration of multiple levels of fog nodes and the evaluation of energy consumption of end user devices are suggested as future work. Considering mobility of user devices is also a promising direction in the design of resource provisioning in fog computing.

## Acknowledgment

The authors would like to thank Coordination for the Improvement of Higher Education Personnel (CAPES), CNPq and FAPESP (proc no 15/24494-8) for the financial support.

## References

[1] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. Cham: Springer International Publishing, 2014, pp. 169–186.

[2] “OpenFog Reference Architecture,” Online, 2017, available at https://www.openfogconsortium.org/ra/ [Accessed: 24/05/2017].

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[4] R. A. C. da Silva and N. L. S. da Fonseca, “Topology-aware virtual machine placement in data centers,” *Journal of Grid Computing*, pp. 1–16, 2015.

[5] ——, “Energy-aware migration of groups of virtual machines in distributed data centers,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[6] X. Masip-Bruin, E. Marn-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, “Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud (f2c) computing systems,” *Wireless Communications*, 2016.

[7] V. B. C. Souza, W. Ramrez, X. Masip-Bruin, E. Marn-Tordera, G. Ren, and G. Tashakor, “Handling service allocation in combined fog-cloud scenarios,” in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–5.

[8] V. B. C. Souza, X. Masip-Bruin, E. Marn-Tordera, W. Ramrez, and S. Snchez, “Towards distributed service allocation in fog-to-cloud (f2c) scenarios,” in *2016 IEEE Global Conference on Communications (GLOBECOM)*, Dec 2016, pp. 1–6.

[9] R. Vilalta *et al.*, “Telcofog: A unified flexible fog and cloud computing architecture for 5g networks,” *IEEE Communications Magazine*, vol. 55, no. 8, pp. 36–43, 2017.

[10] S. Sarkar, S. Chatterjee, and S. Misra, “Assessment of the suitability of fog computing in the context of internet of things,” *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[11] R. Deng, R. Lu, C. Lai, and T. H. Luan, “Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing,” in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 3909–3914.

[12] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, “ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments,” *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[13] Y. Han, J. Chan, and C. Leckie, “Analysing virtual machine usage in cloud computing,” in *Services (SERVICES), 2013 IEEE Ninth World Congress on*, June 2013, pp. 370–377.

[14] A. Bayati, V. Asghari, K. Nguyen, and M. Cheriet, “Gaussian process regression based traffic modeling and prediction in high-speed networks,” in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7.

[15] M. Chilenski *et al.*, “Improved profile fitting and quantification of uncertainty in experimental measurements of impurity transport coefficients using gaussian process regression,” *Nuclear Fusion*, vol. 55, no. 2, p. 023012, 2015.

[16] X. Sun, N. Ansari, and Q. Fan, “Green energy aware avatar migration strategy in green cloudlet networks,” in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov 2015, pp. 139–146.

[17] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, “Cloudlets: Bringing the cloud to the mobile user,” in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, ser. MCS ’12. New York, NY, USA: ACM, 2012, pp. 29–36.

[18] V. Paxson, “Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic,” *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 5, pp. 5–18, Oct. 1997.

[19] A. Beloglazov and R. Buyya, “Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers,” *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.