Context-aware Task Scheduling Algorithm for the Fog Paradigm - A Model Proposal

Celestino Barros
Faculty of Science and Technology
University of Cape Verde
Praia, Cabo Verde
celestino.barros@docente.unicv.edu.cv

Vítor Rocio
INESC TEC and Open University of Portugal
Vila Real, Portugal
vitor.rocio@uab.pt

André Sousa
Critical TechWorks
Porto, Portugal
asousa@roundstone.pt

Hugo Paredes
INESC TEC and University of Trás-os-Montes and Alto Douro
Vila Real, Portugal
hparedes@utad.pt

Olavo Teixeira
Faculty of Science and Technology
University of Cape Verde
Praia, Cabo Verde
olavo.teixeira@docente.unicv.edu.cv

Abstract - **Task scheduling in fog paradigm is very complex and, in the literature, according to the author's knowledge there are still few studies. In the cloud architecture, it is widely studied, and, in much research, it is approached from the perspective of service providers. Trying to bring innovative contributions in these areas, in this paper, we propose a solution to the context-aware task-scheduling problem for fog paradigm. In our proposal, different context parameters are normalized through Min-Max normalization, requisition priorities are defined through the application of the Multiple Linear Regression (MLR) technique and scheduling is performed using Multi-Objective Non-Linear Programming Optimization (MONLIP) technique. The results obtained from simulations in the iFogSim toolkit, show that our proposal performs better compared to the non-context-aware proposals.**

Keywords - Context awareness; fog computing paradigm; task scheduling; scheduling in fog paradigm.

1. INTRODUCTION

The growth of mobile devices and the evolution of the Internet of Things (IoT) has stimulated the growth of devices connected to the Internet. This growth tends to increase significantly. On the other hand, several of these devices run applications that require part of the processing to run in large, centralized datacenters known as cloud. However, due to centralization and physical distance from end user´s devices, it causes an increase in communication latencies and harms applications that require real-time responses. Different techniques that minimize cloud processing by adopting local processing strategies and that allow solving cloud limitations have been proposed. One such technique is the use of the fog computing paradigm [1].

According to [2], many of the task scheduling algorithms in the cloud architecture and fog paradigm found in the literature do not describe how the priority is defined, do not explain the method used to prioritize tasks, nor do they define the prioritization of tasks based on context information, many defend the perspective of service providers. Others are applied in grouped tasks to decrease execution time. Some optimize only QoS. Others explore only some contexts. They also claim that they allow solving many problems. However, some aspects such as the use of context in task scheduling, prioritization of context sensitive tasks, consideration of energy restriction in task scheduling, preservation of network signal strength, preservation of QoS, reduction of average waiting time and QoE optimization can be explored and/or improved.

This paper has as its main objective the conception of context-aware task scheduling algorithms for fog computing paradigm. Aiming to achieve the main objective, some specific objectives were defined: to contextualize concepts such as fog computing, task scheduling and context-aware; identify the main contexts that can be considered in any mobile computing environment; justify the choice of contexts for the domain of our problem; standardize the different context parameters through the use of Min-Max normalization; define the priorities of the requests through the application of the MLR technique and optimize the scheduling through the use of the MONLIP technique.

As the main objective of this article is to design a proposal that has as its scope the creation of knowledge in the form of techniques, methods, models and theory, the most appropriate methodology is Design Science Research. It is organized in five sections: In the first section, we introduce the paper, in the second, we deal with the contextualization of the subject, in the third, we describe the contexts envisaged, the model, and the proposed architecture. In the fourth, the performance of the proposed model is evaluated, and comparison is made with non-context aware proposals and in the fifth, the conclusion of the paper is made.

2. THEORETICAL BACKGROUND

Mobile computing provides users with several utilities, allows portability, supports applications of various interests, and has several limitations such as scarcity of resources, reduced battery life, among others. In recent years, several architectures have been proposed to solve these limitations, being cloud computing one of them. Despite the advantages, in some situations it is not beneficial to use cloud. It is centralized and consequently the processing is done in concentrated data centers, for optimization of energy and communications costs. To solve these inconveniences, several paradigms have been proposed. Among them is fog computing, which aims to make the services offered by the cloud available at the edge of the network [3].

In this section, we contextualize and discuss concepts such as fog computing, context-aware and design of task scheduling algorithms.

2.1. Fog computing

In the opinion of [3], fog computing is a new paradigm that aims to overcome the limitations of cloud by providing services at the edge of the network. In [4], it is broadly defined, and emphasis is given to some characteristics such as geographical distribution, predominance of wireless access, heterogeneity, distributed environment, among others.

As reported in [5], fog has grown a lot since its adoption in late 2012 by Cisco. It disappoints as an integrated solution to extend the capabilities of the cloud to the edge of the network and allow answers to the inconveniences of the classic centralized model.

According to [6], is: “A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum.”

In the opinion of [7], it is a non-trivial extension of the cloud because it provides processing, storage, and networking services near the edge of an enterprise network. Its main characteristics are its proximity to end users, its dense geographic distribution, and its mobility support.

As reported in [8], fog computing from the perspective of mobile computing, aims to provide a cloud-like facility. However, lighter, closer to the users of mobile devices, it can serve these users through direct connection, shorter, compared to the cloud connection. They also advance, that as fog can be implemented locally, it can provide personalized and location-committed services, which are more desirable for mobile users [8].

From our perspective fog computing, converges in terms of approach as per definitions described in this section. That is, it combines data processing and storage in the cloud and/or at the edge of the network, it has the capacity to, when necessary, search for additional computational resources in the cloud, availability of services that allow greater geographical distribution of the system/application and low latency. It consists of three layers as illustrated in Figure 1.

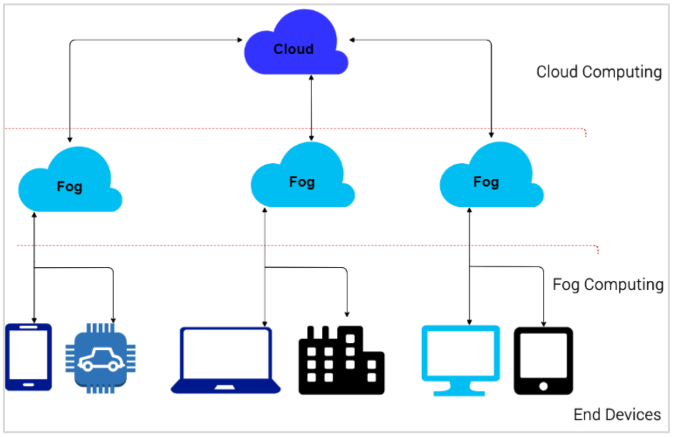


Figure 1. Fog architecture outlook

2.2. Context-aware and Fog Computing

In mobile computing, the context of a user is very dynamic. When using applications in that environment, the behavior of that application must be customized to the current situation of the user. To promote an effective use of context, many authors provide context settings and categorizations. In [9], a definition, the context categories and context sensitive applications are made available. Information and services, information marking with context and automatic service execution methodology are still presented as well as the survey of the state of the art regarding context-aware computing.

Bazire and Brézillon in [10], reviewed 150 definitions of context, in different areas of research and concluded that the creation of a single definition is a hard and probably impossible effort since it varies with the scientific area and depends mainly on the field in which it is being applied. However, they define the context as a set of constraints that influence the behavior relating to a given task [10].

The context definition most used today, even in other fields, as in the operationalization was given by [11], [12]:

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between user and an application, including the user and applications themselves” [12] (p. 45).

In mobile computing, context refers to the processing environment, user environment, physical environment, relevant for the interaction between a user and an application, including the user and the applications themselves [1].

When mobile devices communicate with cloud, they face high network latency and high transmission power consumption [13]. They also advance that in fog paradigm, mobile devices send tasks to fog nodes to be processed and returned the result. This process reduces, among others, the transmission delay and power consumption of the mobile device. Due to the lower capacity of the fog nodes when compared to cloud, the tasks, which cannot be executed in the fog, are sent to be executed in cloud [13], as illustrated in the three-tier architecture shown in Figure 1.

2.3. Design of scheduling algorithms

According to [14], scheduling is the allocation of resources needed to execute a task. It assumes as a fundamental process to improve the reliability and flexibility of the systems. It requires advanced algorithms capable of choosing the most appropriate resource to perform the task. It also advances that in its design, we must consider some constraints such as cost of tasks, dependencies between tasks and the location. It also guarantees that scheduling decisions can be static - where decisions about scheduling are made during the compilation. Or dynamic - where information about the state of the task flow is used at a given time during execution for the scheduling decisions. It is the best approach because it allows several problems to have solutions that can be represented by a search tree. However, these problems are computationally demanding, require a strategy of parallelization and dynamic load balancing [14].

3. PROPOSED MODEL AND ARCHITECTURE

In the following subsections, we describe the envisaged contexts, we illustrate, and we discuss the model and architecture of the proposed solution.

3.1. Contexts envisaged and assumptions

We assume that an appropriate code offloading technique (e.g., MAUI defined in [15], COMET presented in [16], among others) is being run on mobile devices to make the best decision as to whether to offload codes and which fog nodes [17].

We consider that a discharged request includes the maximum delay allowed to execute the application, the battery level, and the values of the network signal strength of the device. We also assume, as in [8] that fog provides greater computing capabilities than mobile devices and can extract the contexts associated with the requests and make the scheduling decision accordingly.

Musumba and Nyongesa in [1], is defined the main contexts that can be explored in any mobile computing environment as: network connection; available processors; battery level; location; network bandwidth; network traffic; leased of Virtual Machines (VM) and application QoS requirements.

In our domain of the problem, the contexts of the service providers because we do not know them were ignored. In addition, after downloading the tasks in the fog nodes, it becomes unnecessary to consider the processors of the mobile device. The location of the device also does not affect the scheduling, as well as network traffic and bandwidth that are the same for all users. Based on these criteria we take into consideration three context parameters: battery level, signal-to-noise network interference ratio (SIN) and application QoS.

3.2. Proposed MODEL

The fog nodes, with our proposal activated, consists of three units: *Context Information Retrieval Unit*, comprises an architecture, as defined in [18]. It retrieves context information (C_i) from each request ($r \in R$). The recovered context information is forwarded to the *Context-Aware Task Prioritization Unit*, which estimates the value of the context priority (P_r) for each individual request $r \in R$ and routes it to the *QoE and Context-Aware Scheduler Unit*, which schedules tasks to be executed in VMs so that QoE is optimized.

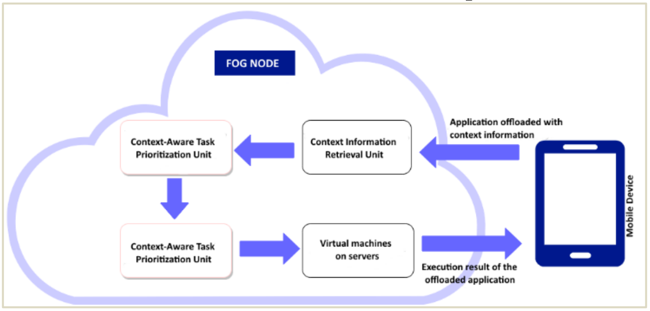

FOG NODE
Context-Aware Task Prioritization Unit
Context Information Retrieval Unit
Application offloaded with context information
Context-Aware Task Prioritization Unit
Virtual machines on servers
Execution result of the offloaded application
Mobile Device

Figure 2: Proposed model.

Figure 2 shows the different units of the proposed model. The relevant notations used in the different units are listed in Table1.

Symbols	Definition
C	Set of all context parameters
α_i	Number of different types of abstract levels of a context parameter, $C_i \in C$
γ_i	Logical difference between two abstract levels $C_i \in C$
η_i	Normalized range of values $C_i \in C$
L_i	Set of all concrete levels of $C_i \in C$ at instant t
θ_i	Set of biased values $C_i \in C$
F	Set of all possible combinations of one element of each L_i
M	Multi set of minimum values $\forall \mu \epsilon F$
R	Set of request application execution at a given instant t
V	Set of virtual machines available at a given instant t
β	Set of multiple linear regression coefficients
P_r	Priority of any request $r \epsilon R$
$\psi_{r,v}$	Time required to execute a request $r \epsilon R$ in a virtual machine $v \epsilon V$
I_r	Number of interruptions of a request $r \epsilon R$

Table 1: Notations and definitions of the proposed model.

We assume that some VMs are already created with different configurations, which allows minimizing the overloads related to the processes of creation and elimination of VMs, as claimed in [19] and [20]. The optimal provision of VM for application requisitions and their energy-efficient allocations are outside the scope of this paper. However, they were further developed in [21] and [22].

3.3. Architecture of the proposed model

The fact that the context parameters associated with a request are heterogeneous makes it difficult to explore the context information in the scheduling. To solve this problem, in Han, Kamber and Jian [23], a context heterogeneity resolver was proposed, which processes several parameters, in a normalized interval, through Min-Max normalization, where each request is prioritized based on its context values.

The following subsections define the architecture of the proposed model, starting with the Context-Aware Task Prioritization Unit.

3.3.1. Context-aware Task Prioritization Unit

This unit is composed by: Context Repository, which stores context information of current and previously received tasks and Context Forecasting Unit, exploits the context information at a given time and feeds the Forecast Table. Thus, we manage to eliminate the heterogeneity of the context information in the feeding of the forecast table.

The Forecast Table provides a data set for the MLR analysis that aims to define the priority of requests.

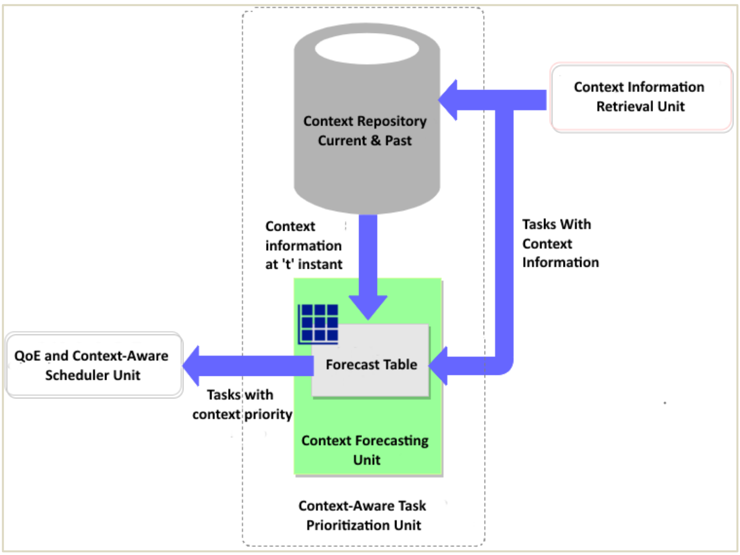

Context Repository Current & Past
Context Information Retrieval Unit
Context information at 't' instant
Tasks With Context Information
QoE and Context-Aware Scheduler Unit
Forecast Table
Tasks with context priority
Context Forecasting Unit
Context-Aware Task Prioritization Unit

Figure 3: Context-Aware Task Prioritization Unit architecture of the proposed model.

Figure 3 shows the architecture of the Context-Aware Task Prioritization Unit of the proposed model.

3.3.2. Creating the Context Priority Forecast Table

To solve problems of heterogeneity of context information, based on Min-Max standardization, a model was designed that feeds the forecast table. The context repository provides context information from tasks previously received at a given time *t*. $\forall C_i \in C$, all the context information is normalized in relation to the extremes, the values are concentrated in a range between 0 and 1. This approach allows minimizing the heterogeneity of the different context parameters in terms of values and units. We define the normalized interval, η_i for a context parameter, $C_i \in C$ as:

$$\eta_i = \frac{\max(C_i) - \min(C_i)}{\max(C_i)}. \qquad (1)$$

Within this normalized range $C_i \in C$, we assume there exist α_i abstract levels. These abstract levels represent the qualitative measurement of the corresponding context parameter. They allow the variations of values of a given context parameter to be classified internally. The logical differences between two consecutive abstract levels, γ_i, of a context $C_i \in C$ are defined as:

$$\gamma_i = \frac{\eta_i}{\alpha_i}. \qquad (2)$$

The numerical representation of this abstraction comprises a concrete set of levels, L_i for any $C_i \in C$. This approach transforms qualitative measurement into quantitative, compatible for further calculations. L_i set is defined as:

$$L_i = \left\{x : x = \frac{\min(C_i)}{\max(C_i)} + z\gamma_i\right\}, \forall C_i \in C. \qquad (3)$$

Where z = 0,1, 2,…,$(\alpha_i - 1)$.

Calculating the set of Cartesian products, the combinatorial set of all context levels is defined from the different context parameters. This Cartesian product is formulated as indicated in equation 4.

$$\mathrm{F} = \prod_{i=1}^{|C|} \mathrm{L}_i. \qquad (4)$$

All possible combinations of the different context parameters of a request($r \in R$) are determined.

A multi-set M is also created with the minimum values of each combination of F, defined according to equation 5.

$$\mathrm{M} = \{\mathrm{q} : \mathrm{q} = \min(\mu)\}, \ \forall \mu \in \mathrm{F}. \qquad (5)$$

It identifies the bottleneck of all possible combinations of the context of a request $r \in R$. This bottleneck parameter influences the prioritization of symbolic combinations.

In addition to the bottleneck parameters, the bias values, associated with the different levels of the various context parameters, used in the prioritization, also influence the prioritization.

This bias value allows emphasizing the other parameters. It is a one-to-one mapping between the elements of L_i and the biased set, θ_i for a given context parameter, $C_i \in C$.

Be, $\theta_{i,j}$ refers to the minimum bias value of C_i, at its j^{th} level associated with a candidate combination in F. For mapping, θ_i to Li, $\theta_{i,0}$, it is assumed, $\forall C_i \in C$.

$\forall C_i \in C, \theta_{i,j}$ must satisfy the following condition:

$$\theta_{i,j} * max(q \in M) < \theta_{i,j+1} * min(q \in M). \qquad (6)$$

The priorities of these combinations are defined so that the context information of any request can be mapped over itself to predict the priority of that request.

The priority calculation is performed using the relevant bias values $\forall C_i \in C$ and its bottleneck context parameter. We assume that $\delta_i\ (\mu_k)$ defines the context priority $\mu_k \in F$ biased in $C_i \in C$. $\delta_i\ (\mu_k)$ is represented according to equation 7.

$$\delta_i\ (\mu_k) = \frac{\theta_{ij} * q_k}{\sum q}, \forall q \in M. \qquad (7)$$

Where, $0 \le k \le (|F| - 1)$ and $q_k \in M$ is associated with μ_k.

The priority estimated, $\hat{\delta}_i(\mu_k)$ of μ_k is calculated using equation 8.

$$\hat{\delta}_i(\mu_k) = \sum_{C_i \in C} \delta_i\ (\mu_k). \qquad (8)$$

μ_k and $\hat{\delta}(\mu_k)$, $\forall \mu_k \in F$ will feed the forecast table with a set of data. This forecast table data works as hypothetical data to predict the priority of any queued request.

3.3.3. Predicting Context Priority

MLR is one of the multivariate statistical methods whose main concern is to establish the relationships between several independent or predictor variables and a dependent variable or criteria [24]. By identifying how these multiple independent variables relate to the dependent variable, information about the independent variables can be used to make accurate and powerful forecasts [24]. Consider m observations of a set of p number of the predictor variable *Xs* and a criterion variable Y associated with them. The MLR model, which fits this scenario, is defined according to equation 9.

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_j x_{ij} + \cdots + \beta_p x_{ip} \in_i, \qquad (9)$$

Where, for the i^{th} observation, y_i= Y e x_{ji}=X_j,$\forall X_j \in X$. The MLR coefficient, β_0, is the interception of the population and β_j, is the variation of Y in a unit of variation in X_j, $\forall X_j \in X\ e\ 1 \le j \le p$, keeping the other independent variables constant, ϵ_i is the random or unexplainable error associated with the i^{th} observation. The estimated values of the MLR coefficients are calculated using the known values in equation (8) of each observation and solved algebraically. Estimating β = $\{\beta_0, \beta_1, \ldots, \beta_p\}$, $\forall \beta_0 \in \beta$ and σ_ϵ^2 (error of the variance), the adjusted regression line, which predicts Y for any unknown observation, is expressed according to equation 10:

$$\hat{y}_i = b_0 + b_1 x_{i1} + \cdots + b x_{ij} + \cdots + b_p x_{ip} \in_i \qquad (10)$$

Where, $\hat{y}_i$ is the predicted value for any unknown observation, b_j is the sample estimate of $\beta_j\ \forall \beta_j \in \beta$.

Instead of calculating regressions for each predictor variable individually, MLR uses information from all independent variables simultaneously to predict a single criteria variable. As a result, it is naturally faster than other multivariate analysis methods [25].

We map the context forecast table to the MLR model, considering each tuple of the forecast table as an observation of an MLR data set where, the set of independent variables, X = C$_i$, $\forall C_i \in C$ of each combination $\mu \in F$, the dependent variable, Y = $\hat{\delta}_i(\mu_k)$ is the expected priority of any unknown request,, $P_r = \hat{y}_i$.

For any request $r \in R$, the lower the P_r value, the higher the priority. The context priority forecast table is created through the quantitative values $\forall C_i \in C$ and the context parameters that are independent from each other.

After the context priority is defined, the context information of this task is saved in the context repository.

3.4. Optimization of application scheduling

In order to optimize QoE, the proposed scheduler explores the context priority (P_r) of the request $r \in R$ and its estimated execution time duration (T_r,v) to define the scheduling of this request $r \in R$ in an VM, $v \in V$. If due to the limited number of VM's and the low priority, a request $r \in R$ cannot be scaled in a scheduling interval, it must be scaled in the next intervals. In our proposal, the execution of tasks with lower priority is interrupted due to the arrival of tasks with higher priority. Considering that this preemption can provide indefinite waiting for the execution of a request $r \in R$, we also explore the number of scheduling intervals (I_r), in which a scheduling request is postponed since its arrival, to avoid the starvation condition.

To optimize QoE, the MONLP technique was used to schedule the requisitions.

According to [26], the MONLP technique is generally applied to optimization problems where there are more than one nonlinear and conflicting objective function (OF) to be optimized simultaneously. It has some elements: set of nonlinear objective functions that must be optimized; decision variables that constitute the domain in which each OF must operate; restrictions that delimit the search space.

In sections 3.4.1 and 3.4.2, the objective functions and restrictions for optimized application scheduling are defined.

3.4.1. Definition of the objective function

One of the objectives of this paper consists of scheduling requisitions, $r \in R$ in VM, $v \in V$, to optimize the QoE for all the requisitions in a certain scheduling interval.

If the execution of all requisitions, $r \in R$ are not preemptive and that all VMs, $v \in V$ are created in the fog, the OF is defined according to equation 11.

$$min_{r,v} \sum_{r \in R} \sum_{v \in V} \frac{P_r * \psi_{r,v}}{I_r} \quad (11)$$

And it is subjected to some restrictions, (inequation (12) to (17)).

This equation indicates that the QoE of all user application requests can be optimized by minimizing the sum of their execution times. It also considers the priority execution of the tasks with higher priorities by minimizing the sum of the priorities of all requests, since the lower the result, the higher the priority obtained. Moreover, the sum of the inverse values of (Ir), $\forall r \in R$ shows that the requisitions, in which their scheduling has been postponed in each interval, will have higher priority to be scaled in the current intervals, thus mitigating the starvation situation.

3.4.2. Definition of restrictions

For the optimized scheduling, we defined the following restrictions:

- Capacity restriction, is presented as the inequation 12

$$\sum_{v \in V} \eta_v \leq \eta \quad (12)$$

Where, η_v represents the size of the virtual machine and η is the total capacity of the fog node.

- Restriction of VM allocation is presented according to inequation 13.

$$\sum_{v \in V} \boldsymbol{K_{v,r}} \leq \mathbf{1}, \forall \boldsymbol{v} \in \boldsymbol{V}, \forall \boldsymbol{r} \in \boldsymbol{R} \quad (13)$$

Where, $K_{v,r}$ is a Boolean variable, which is equal to one, if a VM $v \in V$ is affected to a request $r \in R$; otherwise, it is zero.

- Restriction of scheduling of requisitions is written according to the inequation 14.

$$\sum_{r \in R} \boldsymbol{\chi_{r,v}} \leq \mathbf{1}, \forall \boldsymbol{r} \in \boldsymbol{R}, \forall \boldsymbol{v} \in \boldsymbol{V}. \quad (14)$$

Where, $\chi_{r,v}$ is a Boolean variable, is equal to one if a request $r \in R$ is scheduled in a VM $v \in V$; otherwise, it is zero.

- QoS restriction is represented according to inequation 15.

$$\boldsymbol{\psi_{r,v}} + \boldsymbol{Q_r} \leq \boldsymbol{T_r}, \forall \boldsymbol{r} \in \boldsymbol{R}. \quad (15)$$

Where, $\psi_{r,v}$ represents the time needed to execute a request $r \in R$ in a VM $v \in V$, Q_r is the waiting time in the requisition queue r. e T_r corresponds to the QoS of the application.

- Restriction of energy consumption, is presented according to the inequation 16:

$$\boldsymbol{B_r} \geq \boldsymbol{B_{th}}. \quad (16)$$

Where, B_r represents the battery level of the end-user device associated with a request, $r \in R$ and B_{th} indicates the minimum battery level, so that the requesting device remains on until the end of the execution.

- Restriction of the quality of the signal, is written according to the inequation 17:

$$\boldsymbol{SIN_r} \geq \boldsymbol{SIN_{th}}, \quad (17)$$

Where, SIN_r represents the signal strength associated with a request $r \in R$ and SIN_{th} specifies the minimum signal strength required for submitting a request.

4. Performance evaluation of the proposed model

The simulation environment of the proposed model was developed in the iFogSim simulation toolkit and models a fog consisting of three hosts with parameters as shown in Table 2.

Parameters	Values
Amount of MV	1 ~ 15
Disk capacity of each VM	10,000 MB
Memory of each VM	512 MB
Application QoS	6 ~ 15 s
Request size	1000 ~ 2000 MI
B_{th}	15%
SIN_{th}	15 dB
$\alpha_{Bateria}$	3
α_{sin}	2
α_{QoS}	4
Simulation Duration	500 s

Table 2: Simulation parameters

4.1. Proposed model analysis performance metrics

The following performance metrics have been used to compare our scheduling proposal (with static and dynamic Escalation Interval (IE)) with non-context-aware scheduling approaches such as: FCFS, SJF and QoS-based:

- *Percentage of successful request execution:* is calculated by the ratio between the number of tasks that preserve the different context parameters and the total number of requested tasks. The higher this percentage is, the greater the quantity of tasks that preserve the different context parameters.
- *Average waiting time for a task*: it is the time elapsed from its arrival to its availability in a VM. *Qr* indicates the waiting time of a request $r \in R$. The average waiting time of a task is calculated according to equation 18.

$$\widehat{\boldsymbol{Q}} = \frac{\mathbf{1}}{|\boldsymbol{R}|} \sum_{r \in R} (\boldsymbol{Q_r}) \quad (18)$$

where R represents all requests received during the simulation period. The lower value of $\hat{Q}$ the better the performance.

Quality of Experience (QoE): is the degree of overall satisfaction of users with the use of a product or service. It can be improved through the difference between the maximum time allowed to obtain the response (QoS requirement of the application), Tr and the response time of a request.

We calculate the average QoE of all requests R issued during the simulation as shown in equation 19.

$$\boldsymbol{QoE} = \frac{\mathbf{1}}{|\boldsymbol{R}|}\sum_{\boldsymbol{r}\in\boldsymbol{R}}\sum_{\boldsymbol{v}\in\boldsymbol{V}}(\boldsymbol{T_r} - (\boldsymbol{\psi_{r,v}} + \boldsymbol{Q_r})) \qquad (19)$$

The higher this value, the better the system's ability to optimize user QoE.

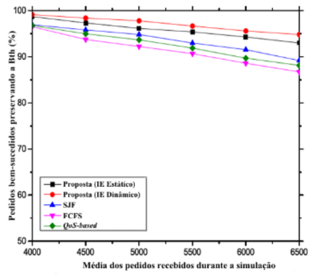
4.2. *Results and discussions*

A detailed analysis of the simulation file allows us to conclude that the success rate in the context-aware scheduling is higher in comparison to the non-context-aware scheduling.

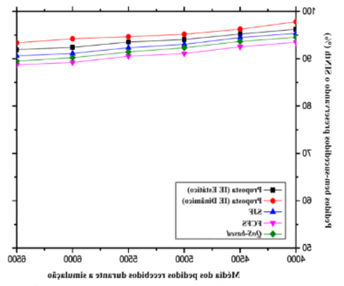
In the following, we present and discuss the variation of successfully executed requests in relation to the increase in requests, VMs and QoS requirements of the application.

4.2.1. Impacts of increased requests

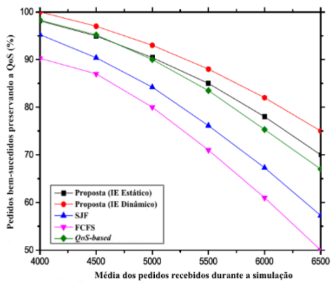
The percentages of successfully executed requests preserving the different context parameters (keeping constant the amount of VMs), are presented in graphs 1, 2 and 3. And shows that by as you increase the average number of requests received, decrease the successful requests that preserve the different context parameters.



Graph 2: Successfully executed tasks preserving battery level in relation to increasing requests.

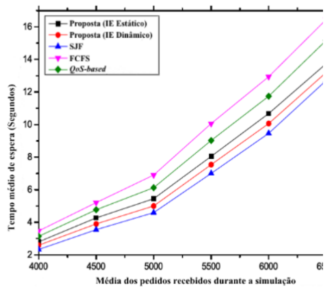


Graph 3: Tasks successfully executed preserving the signal level in relation to the increase requests.

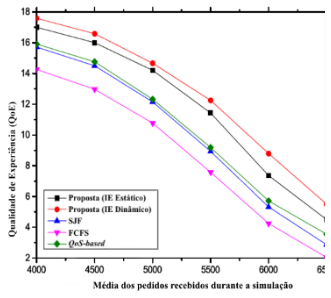


Graph 4: Successfully executed tasks preserving the QoS level in relation to the increase in the number of requests.

In relation to average waiting time and QoE of users we may conclude that as they increase the average number of requests received, they increase the average waiting time (Graph 4). As they increase the average number of requests received, they decrease the QoE of users (Graph 5).



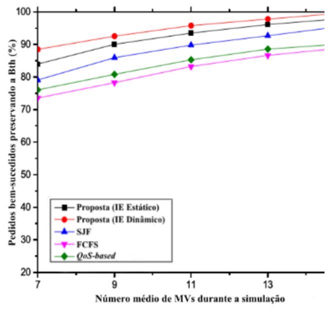
Graph 5: Average waiting time in relation to the increase in the number of requests.



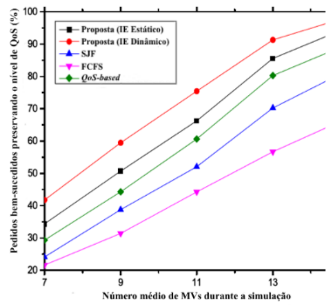
Graph 6: Users' QoE in relation to the increase in the number of requests.

4.2.2. Impacts of the increase of VMs

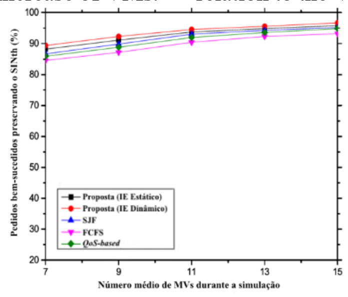
As the average number of VMs increases, so do the successful requests that preserve the different context parameters



Graph 7: Requests executed preserving the battery level in relation to the increase of VMs.



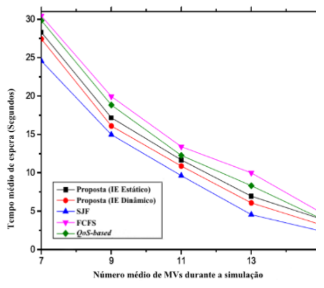
Graph 8: Requests executed preserving the SIN level in relation to the VM increase.



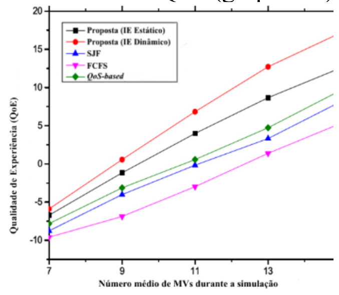
Graph 9: Executed requests preserving the QoS in relation to the VM increase.

As for the waiting time and QoE in relation to the VM increase, according to graphs 9 and 10, they present theoretically proven scenarios:

- increase average number of VMs, decrease average waiting time (graphs 9).
- increase average VMs, increase users' QoE (graphs 10)



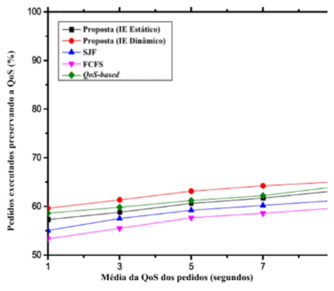
Graph 10: Average waiting time in relation to the VM increase.



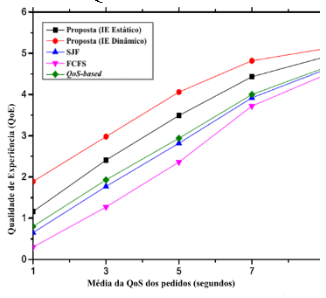
Graph 11: User QoE in relation to VM increase.

4.2.3. Impacts of increasing the average of the application's QoS requirement

In graphs 11 and 12 the impact of the QoS requirement of requests on the performance of the proposed model is illustrated.
In graph 11 we can observe that INCREASE the average of application QoS of a request, INCREASE the executed requests while preserving QoS.
In graph 12 we can observe that INCREASE the average QoS of a request application, INCREASE the QoE of users.



Graph 12: Executed requests regarding the increase of QoS of the application.



Graph 13: User QoE in relation to increased application QoS.

In conclusion to this experience, we state without a doubt that the proposed context-aware scheduling strategy is efficient in terms of the number of tasks that are scheduled and that satisfy QoS, reduction of response time, optimization of QoE, reduction of average waiting time, among others, in comparison with non-context-aware scheduling approaches. Considering the reasoned observations presented, we can state that our scheduling proposal has conditions to be implemented in practical environments.

5. CONCLUSIONS

The main objective of this paper is to define a context-aware task scheduling architecture for the fog computing paradigm. As presented, the proposed architecture is efficient enough to prioritize tasks regardless of the heterogeneity of their contextual information. In addition, it can optimize the QoE of end users and enable win-win situations in terms of resource utilization and runtime.

To achieve the main objective, the following piecemeal objectives were achieved:

- We defined some concepts such as: fog computing paradigm; context-aware and task scheduling. We intend to contextualize the main theories and concepts related to this paper.
- We have identified the different categories and types of context in mobile computing as such: Context of the user's device, where the types of contexts are highlighted: battery level; available processors, location; Network context is another categories, where the types of contexts are highlighted such as: network connectivity, bandwidth, network traffic; another is application runtime context, where we indicate the type of context: QoS of the application; Service provider context, where we indicate the type of context leased virtual machines.
- In our domain of the problem, we take into consideration three context parameters: battery level, signal-to-noise network interference ratio (SIN) and application QoS.
- We propose a context-aware mobile application task scheduler architecture for fog computing paradigm, and we present an illustrative example that helps visualize the functionalities of the proposed architecture. The proposed architecture uses Min-Max normalization, to normalize the different context parameters and solve the problem of heterogeneity of device and application contexts. To define the priority of the context of the requests, the MLR analysis was used, which allows the availability of a set of hypothetical data. The optimal scheduling of requests to optimize QoE, considering the various constraints at the application and service levels was solved by using the MONLP technique.
- The validation of the model and the proposed architecture was done based on simulations performed in the iFogSim toolkit, which enabled us to make experimental and theoretical comparisons of our context aware proposal for both static and dynamic scheduling intervals with non-context aware schedulers (FCFS, SJF, QoS-based) based on the following metrics: request success rate; average waiting time and users' QoE.

All proposed objectives were achieved. We consider several context parameters. However, others can still be pondered to analyze their influences on the scheduling. In addition, we plan to implement our proposal in real fog environment or in the iFogSim simulator, analyze the performance and compare the results with the other proposals of escalation not context-aware like: First Come First Served, Shortest Job First and QoS based Priority Scheduling. Based on some metrics of the evaluation such as: percentage of successful requests execution, average waiting time and QoE of users in relation to the increase of requests, VMs and QoS requirements of the application.

REFERENCES

[1] G. W. Musumba and H. O. Nyongesa, "Context awareness in mobile computing: A review," *Int. J. Mach. Learn. Appl.*, vol. 2, no. 1, May 2013. DOI: 10.4102/ijmla.v2i1.5. URL: https://ijmla.net/index.php/ijmla/article/view/5.

[2] C. Barros, V. Rocio, A. Sousa, and H. Paredes, "Survey on Job Scheduling in Cloud-Fog Architecture," in *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, 2020, pp. 1–7.

[3] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, Nov. 2014, pp. 117–122. DOI: 10.1109/ATNAC.2014.7020884. URL: http://ieeexplore.ieee.org/document/7020884/.

[4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008. DOI: 10.1145/1496091.1496100. URL: https://doi.org/10.1145/1496091.1496100.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16. DOI: 10.1145/2342509.2342513. URL: https://doi.org/10.1145/2342509.2342513.

[6] OpenFog, "OpenFog Reference Architecture for Fog Computing," *OpenFog*, 2017. https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf (accessed Feb. 24, 2020). URL: https://www.iiconsortium.org/pdf/OpenFog_Reference_Arch

itecture_2_09_17.pdf.

[7] 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," *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017. DOI: 10.1002/spe.2509. URL: http://doi.wiley.com/10.1002/spe.2509.

[8] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. We, and L. Sun, "A View of Fog Computing from Networking Perspective," Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.01509. URL: http://arxiv.org/abs/1602.01509.

[9] A. K. Dey, D. Salber, G. D. Abowd, and M. Futakawa, "The Conference Assistant: combining context-awareness with wearable computing," in *Digest of Papers. Third International Symposium on Wearable Computers*, 1999, pp. 21–28. DOI: 10.1109/ISWC.1999.806639. URL: http://ieeexplore.ieee.org/document/806639/.

[10] M. Bazire and P. Brézillon, "Understanding Context Before Using It," in *Modeling and Using Context*, A. Dey, B. Kokinov, D. Leake, and R. Turner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 29–40. DOI: 10.1007/11508373_3. URL: http://link.springer.com/10.1007/11508373_3.

[11] A. K. Dey, "Understanding and Using Context, Personal and Ubiquitous Computing, Vol. 5," vol. 5, pp. 4–7, 2001.

[12] A. K. Dey, "Understanding and Using Context," *Pers. Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Feb. 2001. DOI: 10.1007/s007790170019. URL: https://doi.org/10.1007/s007790170019.

[13] Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, "Resource Efficient Mobile Computing Using Cloudlet Infrastructure," in *2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks*, Dec. 2013, pp. 373–377. DOI: 10.1109/MSN.2013.75. URL: http://ieeexplore.ieee.org/document/6726359/.

[14] P. Swaroop, "Cost Based Job Scheduling in Fog Computing," (Ph.D), Delhi Technological University, 2019. URL: http://dspace.dtu.ac.in:8080/jspui/handle/repository/16722.

[15] E. Cuervo *et al.*, "MAUI: Making Smartphones Last Longer with Code Offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, 2010, pp. 49–62. DOI: 10.1145/1814433.1814441. URL: https://doi.org/10.1145/1814433.1814441.

[16] M. Gordon, D. Jamshidi, S. Mahlke, Z. Mao, and X. Chen, "COMET: code offload by migrating execution transparently," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, 2012, pp. 93–106.

[17] F. Berg, F. Dürr, and K. Rothermel, "Increasing the Efficiency and Responsiveness of Mobile Applications with Preemptable Code Offloading," in *2014 IEEE International Conference on Mobile Services*, 2014, pp. 76–83. DOI: 10.1109/MobServ.2014.20.

[18] H. J. La and S. D. Kim, "A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services," in *2010 IEEE 3rd International Conference on Cloud Computing*, Jul. 2010, pp. 466–473. DOI: 10.1109/CLOUD.2010.78. URL: http://ieeexplore.ieee.org/document/5557960/.

[19] A. K. Das, T. Adhikary, M. A. Razzaque, and Choong Seon Hong, "An intelligent approach for virtual machine and QoS provisioning in cloud computing," in *The International Conference on Information Networking 2013 (ICOIN)*, Jan. 2013, pp. 462–467. DOI: 10.1109/ICOIN.2013.6496423. URL: http://ieeexplore.ieee.org/document/6496423/.

[20] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-Aware Fog Service Placement," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, May 2017, pp. 89–96. DOI: 10.1109/ICFEC.2017.12. URL: http://ieeexplore.ieee.org/document/8014364/.

[21] T. Li, Y. Liu, L. Gao, and A. Liu, "A Cooperative-Based Model for Smart-Sensing Tasks in Fog Computing," *IEEE Access*, vol. 5, pp. 21296–21311, 2017. DOI: 10.1109/ACCESS.2017.2756826. URL: http://ieeexplore.ieee.org/document/8049440/.

[22] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2094–2106, Jun. 2018. DOI: 10.1109/JIOT.2018.2823000. URL: https://ieeexplore.ieee.org/document/8331084/.

[23] J. Han, M. Kamber, and P. Jian, *Data Mining: Concepts and Techniques*. Elsevier, 2012. DOI: 10.1016/C2009-0-61819-5. URL: https://linkinghub.elsevier.com/retrieve/pii/C20090618195.

[24] G. P. Quinn and M. J. Keough, *Experimental Design and Data Analysis for Biologists*. Cambridge University Press, 2002. DOI: 10.1017/CBO9780511806384. URL: https://www.cambridge.org/core/product/identifier/9780511806384/type/book.

[25] C. Mertler, *Advanced and Multivariate Statistical Methods: Practical Application and Interpretation*, 4th Editio. Pyrczak Publishing, 2009.

[26] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12. Boston, MA: Springer, Boston, MA, 1998. DOI: 10.1007/978-1-4615-5563-6. URL: http://link.springer.com/10.1007/978-1-4615-5563-6.