Multi-Objective Optimization Approach for Task Scheduling in Fog Computing

Conference Paper · July 2019

DOI: 10.1109/ICABCD.2019.8851038

4 authors:

Mxolisi Mtshali
Council for Scientific and Industrial Research, South Africa
5 PUBLICATIONS **25** CITATIONS
SEE PROFILE

Matthew Adigun
University of Zululand
212 PUBLICATIONS **819** CITATIONS
SEE PROFILE

Sabelo Dlamini
University of Cape Town
31 PUBLICATIONS **193** CITATIONS
SEE PROFILE

Pragasen Mudali
University of Zululand
59 PUBLICATIONS **174** CITATIONS
SEE PROFILE

Multi-Objective Optimization Approach for Task Scheduling in Fog Computing

Abstract— **The Fog computing paradigm allow applications to be processed at the edge of a network. This paradigm is designed to mitigate high latency and the burden of task requests sent to centralized cloud servers by end devices. Fog computing permits different portions of applications to be scheduled to Fog nodes available at the edge. These Fog nodes offer cloud processing services and have appeared as a feasible technique for real time applications. However, scheduling a task among available Fog nodes must be effective, meaning it must not over consume available resources because of limited resources at the edge. Consuming extra amount of energy than available on the Fog nodes can lead to network breakdown or application failure which is not acceptable for real-time applications. Therefore, to address this challenge, this paper presents an application scheduling technique based on virtualization technology to find an efficient algorithm that can optimize energy consumption and average delay of real-time applications in Fog computing networks. This is achieved by implementing four task scheduling policies in a Fog node scheduler to assess their performance and efficiency. Simulations were conducted using the iFogSim tool and the results demonstrate that the FCFS scheduling policy achieved improvement in energy consumption by 11 %, average task delay 7.78 %, 4.4 % network usage and execution time 15.1 % better than other algorithms.**

Keywords—Internet of Things, Fog Computing, Cloud Computing, iFogSim, Task Scheduling.

I. INTRODUCTION

Cloud network architectures offers robust computing functionality to terminal nodes and has acted as a de-facto solution for the past decades [1]. However, since the Internet of Things (IoT) has emerged, more technological transformation has occurred and cloud computing began facing challenges. These challenges include real-time application requirements such as low-latency, location-awareness and mobility support [2]. In short, achieving IoT application requirements is deemed to be theoretically tedious for conventional cloud network architectures [3].

Therefore, due to the previously stated facts from literature, there is a necessity to manage the amount of resources on each Fog nodes (FNs) to optimize the resource utilization and enforce fairness among FNs. In [4], the authors highlights that resource management can be task-scheduled based on either virtualisation or containerisation. Based on above considerations, it is necessary to study resource management policies based on virtualization to consider fairness, computation and energy availability on FNs.

In this paper, we conduct an evaluation study of scheduling policies to optimize performance and energy with fairness. Fairness is considered as optimizing the load of tasks processed on each FNs. This objective is crucial since it can impact the overall performance of the network. In addition, we provide testing of the optimization problem solution by designing a Fog-Computing-based use case scenario and implement a scheduling policy that gives the optimal solution

The rest of paper is organized as follows: Section II, presents related work in resource scheduling in Fog computing applications, section III presents approach of scheduling algorithm in fog computing, section IV presents experimental setups and simulation results; Section V presents conclusions and future work.

II. LITERATURE AND RELATED WORK

In this section, we start by briefly introducing the three-tier architecture of Fog Computing model. Moreover, we provide a descriptive analysis of the overall work related to resource scheduling of task processes among all fog nodes in cooperative fog computing system. Finally, we discuss the future direction and extension that our work will be presenting in terms of optimization problems.

A. Literature on Fog Computing Architecture

Fig. 1 illustrates Fog Computing architecture. Fog computing is described as the extension of cloud computing capabilities towards the edge of the local network. This is done by introducing an intermediary computing layer between cloud core-servers, and terminal nodes. This intermediary layer is composed of deployed micro-datacenters known as Fog Nodes (FNs).

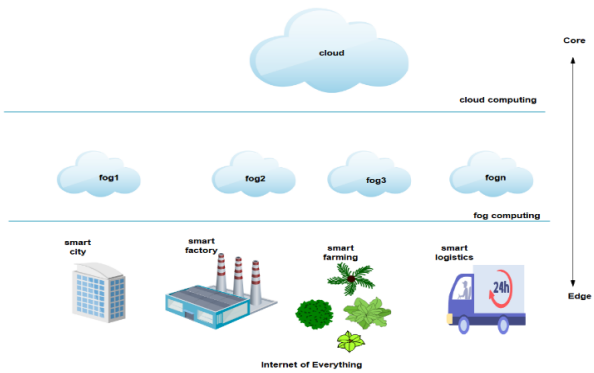

Fig. 1:Fog Computing Architecture and Use Cases.

In the architecture illustrated in Fig. 1, each FN is presented as a highly virtualized computing system, analogous to a light-weight cloud server and is equipped with facilities for storage, compute and network [5]. The main functionality of these FNs is to offer storage and computational services at the edge more proximal to terminal nodes. However, since FNs are remotely located from cloud core-servers, they have limited physical resources in terms of storage, memory and processors [6].

B. Related work

In Fog Computing, resource scheduling is a very attractive topic to researchers yet very challenging. It alludes to making decisions concerning where and how to share available computing resources for processing. This will help to achieve fairness among shared resources. Yi et al [7] summarizes new opportunities and challenges in fog computing, and discusses techniques related to issues of resource management, QoS, interfacing, security and privacy.

Klas [8] introduces and explores different types of edge computing implementations. Furthermore, descriptively details their distinct characteristics with standards they support for recommendations. The newest type of edge computing paradigm mostly motivated by internet of things is Fog computing. Most recent studies in [9],[10] and [11], have focus more on illustrating possible Fog computing architectures designs.

Furthermore, since fog computing is relatively new, there are few studies focusing on resource optimization using scheduling technique. Jalali et al [12] provide two energy consumption minimization models, flow-based and time-based for both shared and unshared network equipment, respectively. To validate and evaluate their proposed models, they perform experiments based on energy consumption of a service provided by nano data Centre's(nDCs) and centralized data centers (DCs). In their findings, they state that nDCs can complement centralized DCs and they are energy saving. Specifically, to IoT based applications.

Zhu et al [13] propose a novel fog computing model which aims to optimize execution time and energy consumption. This model uses offloading policy to effectively bring the fog computing power closer to the mobile user. The model is composed of two types of nodes, remote cloud nodes and local cloud nodes, attached to wireless access infrastructure. In their findings they found that this proposed method optimizes execution time and energy consumption. However, they did not consider performance metric such as latency and network usage for overall application sense-actuate process.

Liu et al [14] propose to optimize energy consumption and performance delay of mobile devices (MDs) by using offloading method. They derived analytical results of objective metric by assuming three types queueing models at MDs, explicit consideration of wireless channel, fog and central cloud. From the results, they used integer programming models (IPMs) algorithms. The results show their scheme outperforms over existing schemes evaluated. However, authors did not consider evaluating fog-based only scheduling methods, since fog was argued to be efficient than cloud. Furthermore, they did not consider performance metric such as execution time and network usage.

Natesha and Guddeti [15] propose First-Fit Decreasing heuristic approach to solve the challenge of selecting a fog node that is suitable to host the IoT applications. This placement problem goal is to reduce application latency and energy consumption for efficient usage of fog node resource. The validation of the algorithm was evaluated with two placement approaches; either applications are deployed in Fog-Cloud interplay or in Cloud only. However, this method was only tested to be efficient on fog vs cloud. They did not consider performance evaluation of scheduling algorithms in the fog-only deployment.

Kabirzadeh et al [16], [17] present various existing scheduling algorithms and propose a hyper-heuristic algorithm for evaluation in fog computing. These several scheduling algorithms includes namely, FCFS (First Come First Served), PSO (Particle Swarm Optimization), ACO (Ant Colony Optimization Algorithm), SA (Simulated Annealing Algorithm:) and GA (Genetic Algorithm). However, not all algorithms which are alternatives of FCFS have been evaluated. Therefore, we intend to investigate classical scheduling algorithms that have been applied in centralized cloud architectures.

III. SYSTEM MODELLING AND USE CASE

In order to realize full capabilities of Fog computing, in this section we discuss specific scheduling algorithms that can be used in optimizing the system performance on either fog or cloud-related case studies. Furthermore, we present formulas utilized to calculate impact of scheduling algorithms on performance of cloud or fog computing data centers. The performance metric includes energy consumption, execution time and network usage. Finally, we model a Fog computing related case study of video surveillance, and description of its configurations

A. Selected Algorithms

The algorithms evaluated in this work are selected based on work done in [18]. They are selected because they have been successfully applied in cloud computing linear programming problem which is what this work aims to address, but now in a distributed fog computing network. These algorithms are as follows:

First Come First Serve (FCFS) is the most basic scheduling algorithm. It processes tasks based on their arrival order. It does not consider any of task length or size, it takes the first task in the ready queue. The second algorithm is Shortest Job First (SJF), it is a version of FCFS algorithm, adding the feature of task processing based on shortest length of task and queuing them for execution in an ascending order. Therefore, this particular algorithm does the sorting first.

The third scheduling algorithm is Round Robin, it is designed to distribute equal processing time among tasks. Each task is given equal time of accessing resources. This time is known as time quantum (usually 10-100 milliseconds). The final algorithm evaluated is Generalized-Priority; it assigns fixed priorities to each and every task and the scheduler arranges the tasks in the ready queue based on their priority (score) and executes the ones with the higher priority. The prioritization in this work is based on the execution time and the job length.

B. Evaluation Metrics

The comparative analysis of performance metrics for task scheduling on distributed fog computing is based on energy consumption, network usage and execution time. The performance metrics are discussed below:

Energy Consumption: refers to the overall energy consumed by the system. This energy is used by any component interacted from sensor to actuator loop. It can be mathematical presented by eq. 1

$$Energy = CEC + (CT - LUUT) * HLU \quad (1)$$

Where *CEC* represents Current energy consumption, *CT* presenting the current time, *LUUT* presenting the last utilization update time and *HLU* presenting the host last utilization.

Execution time: refers to the time the tuple spends utilizing fog node resources. It can be mathematical presented by eq. 2 as follows

$$Execution\ Time = CT - SST \quad (2)$$

Where *CT* represents the current time and *SST* denotes the simulation start time.

Network Utilization: refers to amount of task request sent and received by a network. It can be mathematically presented by eq. 3 as follows

$$Network\ Utilization = \frac{(TL*TS)}{MST} \quad (3)$$

Where *TL* and *TS* denotes the total latency and total size of the tuple, respectively. The *MST* denotes the maximum simulation time.

C. Case Study Video Surveillance

The physical configuration tabulated in Table I is the case study of the topology illustrated in of Fig. 2.

TABLE I: CONFIGURATION OF FOG DEVICES

DEVICE TYPE	CPU GHz	RAM(GB)	POWER(W)
Cloud VM	3.0	4	107.339(M) 83.433(I)
Wi-Fi gateway	3.0	4	107.339(M) 83.433(I)
Smart Camera	1.6	1	83.53 (M) 82.44 (I)
ISP gateway	3.0	4	107.339(M) 83.433(I)

In this case the choice of the configuration values in Table I. is based on the minimum requirement of video surveillance in real-world scenarios. This is due to the fact that video quality is important, thus we do not want poor quality video and the storage needs to be efficient for continuous storage of captured video streams. The model topology is shown below in Fig. 2.

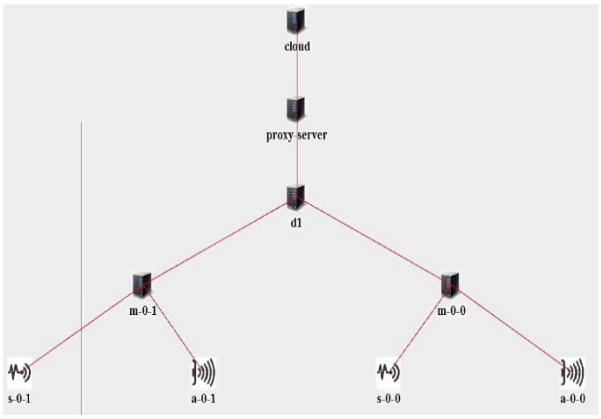

Fig. 2: Fog Computing Application Model in iFogSim.

In the case study, we used linear modelling to indicate that each process is dependent to another as illustrated in Fig. 2. The d1 in the topology refers to a fog device. While s-0-1 and a-0-1 referring to live stream fed to smart cameras and pan–tilt–zoom (PTZ) actuation to object detection. Lastly we have m-0-1 representing smart cameras physical infrastructure.

In Fig. 3 we illustrate the method applied for video surveillance application. In order to elaborate how the resource management can be beneficial to real-time application,we provide a linear approach of how modules in the video surveillance case study co-operate. The descriptive analysis of each module application function are provided below.

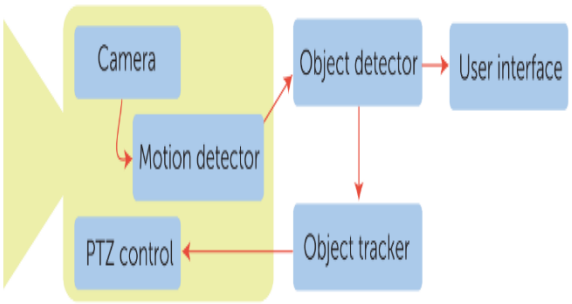

Fig. 3: Video Surveillence Application Module.

In Fig. 3, the camera module takes video streams and feed them to motion detector module. The motion detector module filters incoming video stream and push the filtered video streams to object detector. The object detector module then identifies object detected by calculating coordinates and the push them to the object tracker.

The object tracker computes the desirable coordinates and optimizes PTZ of all cameras for clear capturing of object. In this case study we assume that motion detector and PTZ control modules are always placed in the cameras. The user interface is on terminal devices, and continuously sends fractions of application live video streams containing each tracked object to the surveillance operators/ managers.

In addition, Table II below outlines the configuration of each application module component in the Video Surveillance Application.

TABLE II:VALUES FOR MONITORING APPLICATION

Tuple Type	CPU Length	N/W length
raw_video_stream	2000	20 000
motion_video_stream	500	2000
detected_object	1000	100
ptz_params	100	100

The choice of the configuration values in Table II above were selected based on the minimal requirements of existing video surveillance systems since the goal is to simulate a realistic case study.

Table III below outlines latencies configuration between the source and destination nodes. This is how the node pairing communication is achieved.

TABLE III :NETWORK LINK PARAMETERS

SOURCE	DESTINATION	LATENY, MS
sensor	wi-fi gateway	2
wi-fi gateway	isp gateway	2
isp gateway	cloud dc	100

In Table III, each sensor connects to Wi-Fi gateway and also connects to ISP gateway as illustrated in Fig. 2. The Cloud and ISP gateway connection is mostly utilized for historical storage on the cloud or for nod-delay sensitive data. The configuration of end sensor devices are outlined in Table IV below, the average time for communication between application modules are as shown.

Table IV: SENSOR CONFIGURATIONS

CPU Length	NW Length	Average Inter-arrival Time
1000 million instructions	20 000 bytes	5 ms

In summary, we have provided a descriptive analysis of a video surveillance case study in this section. Moreover, we presented the application modelling and types of scheduling methods that can be applied to optimise performance of the case study. All the configurations were presented and were related to minimum requirements of a real-life case study of video surveillance.

IV. RESULTS AND DISCUSSION

In this section, we simulate a case study in Fog computing. We evaluate the impact of scheduling policies on objective metric such as energy consumption, network utilization, application latency, and execution time.

A. *Datasets and Procedures*

The experiments were conducted on Windows 10, with the following specification: Intel(R) Core (TM) i7 processor 3.20GHz x12, 40 GB RAM, using iFogSim simulation tool [19]. iFogSim tool is for simulating fog computing related network scenarios and has been used in [20], [21], and [22]. The iFogSim classes are related to CloudSim tool [23]. The simulations were done for 3 state areas and mobile devices as in (Area, Mobile) = (1,1), (1,2), (1,3) for different network topology configurations, namely: Config 1, Config 2 and Config 3 respectively. Each topology configuration was tested under the impact of four scheduling policies implemented, namely, FCFS, SJF, GP, and RR. The impact of scheduling policies on the workflow of application task scheduling focuses on energy consumption, application latency, execution time and network usage.

B. *Result Evaluaton*

In Fig. 4 below we illustrate the impact of scheduling policies on energy consumption for various configurations. The energy consumption is based on formula (1).

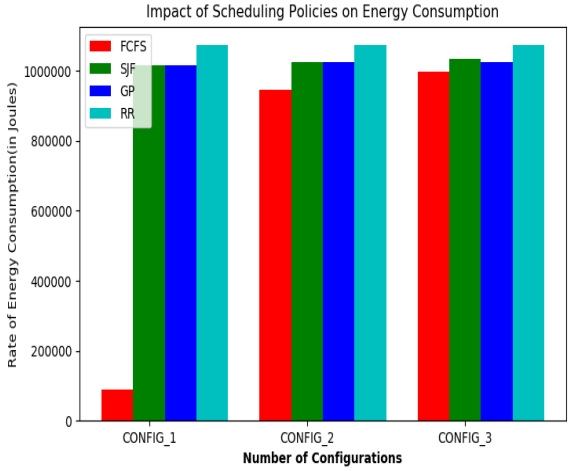

Fig. 4: Average Energy Consumed for Different Policies.

The observation shows that as the number of surveillance cameras monitoring the area increase, the number of fog devices needed also increases, so is the total energy consumed by the system. Hence, from Fig. 4, the FCFS is optimal than the three mentioned algorithms. The energy consumed by RR is slightly larger, while SJF and GP consumed the same amount of energy. In Fig. 5 below we illustrate the impact of scheduling policies on average delay of application modules for various configurations. The delay of application is dependent to the number of hops from sensing to resource efficient node.

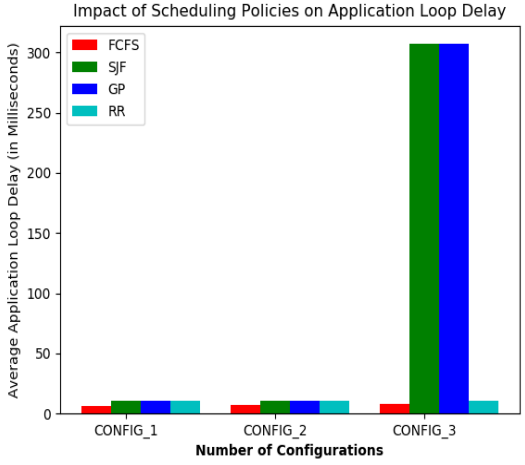

Fig. 5: Average Delay of Application for Different Policies.

Fig. 5 above outlined the average processing latency from point of sensing to actuating in a control loop. From this observation we could clearly see that the highest delay is when there are a greater number of surveillance cameras. This shows that increase in task results in more processing and need more fog nodes and more stretching to occur on the loop control searching for efficient node.

In Fig. 6 below we illustrate the impact of scheduling policies on average execution time for various configurations. The measured results are based on formula (2).

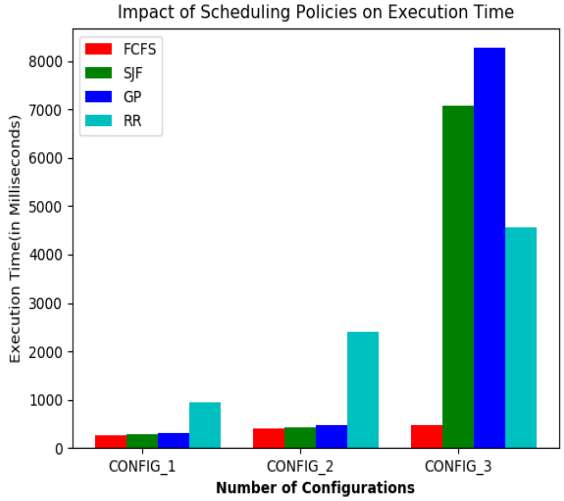

Fig. 6: Average Execution Time for Different Policies.

Fig. 6 above shows that FCFS appears to be slightly optimal than SJF and GP for first and second configuration. However, for configuration three, where there's 1 area monitored by three cameras. The RR appears to be more optimal than SJF, and GP. While remaining less optimal than the FCFS. This could be the fact that RR gives equal amount of time to access resource available for every available task. Thus, the amount of execution is quite fair than SJF, and GP.

In Fig. 6 below we illustrate the impact of scheduling policies on average network utilization for various configurations. The measured results are based on formula (3).

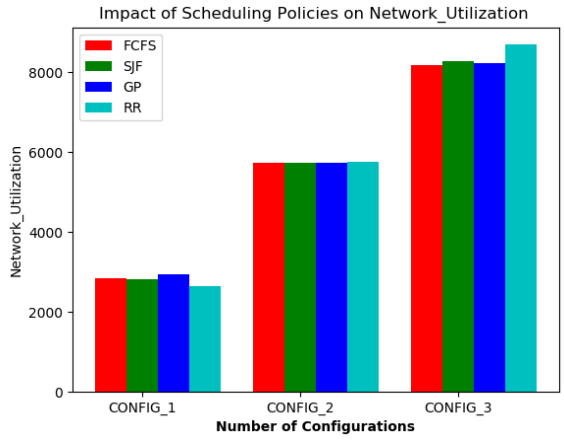

Fig. 7: Average Network Usage for Different Policies.

The illustration of network usage shown in Fig. 7 outlines that as the number of cameras per area monitored increase, more fog nodes are needed, so is the amount of network resource used increases. As shown in the illustration on the first network topology configuration, the RR is optimal than all the algorithms. While the SJF appears to be optimal than FCFS and GP. Furthermore, based on formula (3) which is used to calculate the amount of network usage, it is clearly that total tuple size and total latency have a huge impact. Another observation is that the FCFS algorithm remains optimal in overall performance.

In summary, the results show that FCFS overall performance on average is better than SJF, GP and RR. The performance metric includes Application delay, Network usage, execution time and Energy consumption. The possible reason for FCFS scheduling policy to be optimal than other scheduling policies in terms of energy could be that, the FCFS algorithm only process what is given to it regardless of size or length. Thus, it does not consume more energy while sorting or arranging tasks. While on the other side the SJF, and GP needs to do the sorting and arrangement according to length of task and size of task respectively.

CONCLUSION AND FUTURE WORK

In this paper we investigated resource scheduling problem in fog computing environment. Several cloud computing based scheduling policies have been evaluated on the fog computing environment. The evaluation of these scheduling policies is based on multi-objective metric, namely, energy consumption, execution time, network consumption and network consumption. The multi-objective problem optimization was investigated under FCFS, SJF, GP, and RR scheduling policy. The results show that on average the FCFS performance improved by 11% energy consumption, 15% execution time, 7.78 % average delay and 4.4 % network usage than SJF, GP and RR. Therefore, FCFS is the most optimal scheduling policy which minimizes energy consumed for computing of task and service execution of task.

In our future work, we will look at the impact of heuristics, hyper-heuristics, meta-heuristics and hybrid-heuristics scheduling algorithm(s) on performance metrics such as energy consumption, service latency and cost. Furthermore, a testbed design base on Fog computing will be provided as show case.

REFERENCES

[1] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair Task Offloading among Fog Nodes in Fog Computing Networks," 2018 IEEE Int. Conf. Commun., pp. 1–6, 2018.

[2] K. Bierzynski, A. Escobar, and M. Eberl, "Cloud, fog and edge: Cooperation for the future?" 2017 2nd Int. Conf. Fog Mob. Edge Comput. FMEC 2017, pp. 62–67, 2017.

[3] M. Mukherjee, L. Shu, and D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," IEEE Commun. Surv. Tutorials, no. c, pp. 1–1, 2018.

[4] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, "A Comparative Study of Containers and Virtual Machines in Big Data Environment," 2018.

[5] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog Computing: Towards Minimizing Delay in the Internet of Things," Proc. - 2017 IEEE 1st Int. Conf. Edge Comput. EDGE 2017, pp. 17–24, 2017.

[6] T. Wauters, B. Volckaert, and F. De Turck, "Fog Computing: Enabling the Management and Orchestration of Smart City Applications in," 2018.

[7] S. Yi, C. Li, and Q. Li, "A Survey of Fog Computing: Concepts, Applications and Issues," 2015.

[8] G. I. Klas, “Fog Computing and Mobile Edge Cloud Gain Momentum Open Fog Consortium, ETSI MEC and Cloudlets,” pp. 1–14, 2015.
[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” Proc. first Ed. MCC Work. Mob. cloud Comput. - MCC ’12, no. March, p. 13, 2012.
[10] W. Shi and S. Dustdar, “The Promise of Edge Computing,” Computer (Long. Beach. Calif)., vol. 49, no. 5, pp. 78–81, 2016.
[11] M. A. Nadeem and M. A. Saeed, “Fog computing: An emerging paradigm,” 2016 6th Int. Conf. Innov. Comput. Technol. INTECH 2016, no. August 2016, pp. 83–86, 2017.
[12] F. Jalali et al., “Fog Computing May Help to Save Energy in Cloud Computing,” IEEE J. Sel. Areas Commun., vol. 34, no. 5, pp. 1728–1739, 2016.
[13] Q. Zhu, B. Si, F. Yang, and Y. Ma, “Task Offloading Decision in Fog Computing System,” pp. 59–68, 2017.
[14] L. Liu, Z. Chang, S. Member, X. Guo, S. Mao, and S. Member, “Multiobjective Optimization for Computation Offloading in Fog Computing,” IEEE Internet Things J., vol. 5, no. 1, pp. 283–294, 2018.
[15] B. V Natesha, R. Mohana, and R. Guddeti, “Heuristic-based IoT Application Modules Placement in the Fog-Cloud Computing Environment,” 2018 IEEE/ACM Int. Conf. Util. Cloud Comput. Companion (UCC Companion), vol. 1, no. 6, pp. 24–25, 2018.
[16] D. Rahbari and M. Nickray, “Scheduling of Fog Networks with Optimized Knapsack by Symbiotic Organisms Search.”
[17] S. Kabirzadeh, D. Rahbari, and M. Nickray, “A hyper heuristic algorithm for scheduling of fog networks,” Conf. Open Innov. Assoc. Fruct, pp. 148–155, 2018.
[18] H. G. Tani et al., “Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data to cite this version: HAL Id: hal-01443713 Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big,” 2018.
[19] 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, 2017.
[20] J. P. D. Comput, R. Mahmud, S. Narayana, and K. Ramamohanarao, “Quality of Experience (QoE) -aware placement of applications in Fog computing environments,” J. Parallel Distrib. Comput., 2018.
[21] M. I. Naas, J. Boukhobza, P. R. Parvedy, and L. Lemarchand, “An Extension to iFogSim to Enable the Design of Data Placement Strategies,” 2018 IEEE 2nd Int. Conf. Fog Edge Comput., pp. 1–8, 2018.
[22] M. I. Naas, J. Boukhobza, P. R. Parvedy, and L. Lemarchand, “An Extension to iFogSim to Enable the Design of Data Placement Strategies,” no. May, 2018.
[23] A. Pardo, J. A. Fisteus, and C. D. Kloos, “A distributed collaborative system for flexible learning content production and management,” J. Res. Pract. Inf. Technol., vol. 44, no. 2, pp. 203–221, 2012.