UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

UPCommons

Portal del coneixement obert de la UPC

<http://upcommons.upc.edu/e-prints>

This is a post-peer-review, pre-copy edit version of an article published in *Computing (Vienna/New York)*. The final authenticated version is available online at: <http://dx.doi.org/10.1007/s00607-020-00867-w>.

Published paper: Xhafa, F.; Aly, A.; Juan, A. Allocation of applications to Fog resources via semantic clustering techniques: with scenarios from intelligent transportation systems. "Computing (Vienna/New York)", Març 2021, vol. 103, núm. 3, p. 361-378. doi:[10.1007/s00607-020-00867-w](https://doi.org/10.1007/s00607-020-00867-w)

URL d'aquest document a UPCommons E-prints:

<https://upcommons.upc.edu/handle/2117/340699>

Allocation of Applications to Fog Resources via Semantic Clustering Techniques

With Scenarios from Intelligent Transportation Systems

Fatos Xhafa · Alhassan Aly · Angel A. Juan

Received: date / Accepted: date

Abstract The fast development in IoT and Cloud technologies has propelled the emergence of a variety of computing paradigms, among which Fog and Edge computing are salient computing technologies. Such new paradigms are opening up new opportunities to implement novel application scenarios, not possible before, by supporting features of mobility, edge intelligence and end-user support. This, however, comes with new computing challenges. One such challenge is the allocation of applications to Fog and Edge nodes. Indeed, for some application scenarios larger computing capacity might be needed. Therefore, due to co-existence of computing devices of different computing granularity, techniques for grouping up and clustering resources into virtual nodes of larger computing capacity are required. In this paper we present some clustering techniques for creating virtual computing nodes from Fog/Edge nodes by combining semantic description of resources with semantic clustering techniques. Then, we use such clusters for optimal allocation (via heuristics and Liner Programming) of applications to virtual computing nodes. Simulation results are reported to support the feasibility of the model and efficacy of the proposed approach. First Fit Heuristic Algorithm (FFHA) outperformed ILP method for medium and large size instances. Likewise, FFHA performed

Fatos Xhafa
Universitat Politècnica de Catalunya
Barcelona, Spain
E-mail: fatos.xhafa@upc.edu
**Corresponding author*

Alhassan Aly
Universitat Politècnica de Catalunya
Barcelona, Spain
E-mail: alhassan.ali@estudiantat.upc.edu

Angel A. Juan
Universitat Oberta de Catalunya
Barcelona, Spain
E-mail: ajuanp@uoc.edu

more consistently than ILP on various experimental setting. Finally, the results showed that the proposed clustering techniques deliver relatively fast response times, while enabling the service of a larger number of applications, with more demanding requirements.

Keywords Fog computing · Optimization · Allocation · Clustering · Semantic computing · Smart logistics · Intelligent Transportation Systems

1 Introduction

A number of computing paradigms and technologies have emerged after IoT and Cloud computing. Indeed, the view of IoT Cloud, in which IoT devices are directly connected to Cloud platforms and Data centers has shifted to a *continuum* view, where thereby various computing layers sit in between of IoT and Cloud. These layers referred to as Edge and Fog computing aim to retain the processing of data close to the source where the data is generated. Besides significantly alleviating the computing burden to the Cloud, such layers enable faster round-trip time processing and better support to end users. Most importantly, by processing data close to data sources and to end-users, real time decision making can be designed to support a variety of use cases. This is made possible by achieving the computing and data fabric view of Edge computing [26], where different computing granularity devices can collaboratively support both real time processing and data storage requirements.

Fig. 1: Cloud-to-thing continuum ecosystems

Fig. 1 depicts the Cloud-to-thing continuum concept. In the figure, IoT layer is deployed within smart cities, smart vehicles (transportation, logistics, etc.), smart fabrics, smart homes, etc., which are connected to Fog servers. Processing can be done at Fog servers and reports send back to individual users,

or it can be done at a more thorough level at enterprise servers / Cloud data center for decision making and end-to-end management. One particularity of Cloud-to-thing continuum is that it provides support to *mobility* requirements. Indeed, as can be seen in Fig. 1, for instance, there are nodes corresponding to transportation fleets and VANETs.

It should be noted that Cloud-to-thing continuum infrastructure supports both real time processing and in-depth processing of IoT data. Real time processing enables, in particular, the detection of anomalies or abnormal event [1, 2, 6, 27] regarding transportation fleet, etc., which would require immediate intervention by managers and local stakeholders. Further, massive offline processing enables computing of advanced descriptive, predictive and prescriptive analytics to support decision making and business intelligence and big data solutions [19]. As can be seen from Fig. 1, Cloud-to-thing-continuum enables the implementation of a plethora of application scenarios from smart homes [11] and smart grids [25] to smart logistics [12] and intelligent transportation systems [18, 29]. Indeed, with computing layers of Fog and Edge computing, the premise is that latency and round trip time can be drastically reduced.

In case of intelligent transportation systems, a new generation is being envisaged and deployed based on OBU–On Board Units– and RSU –Road Site Units– technologies in road transport networks [8, 16, 22]. The information gathered by OBU in vehicles and RSU units in roads enables information gathering and can be used to evaluate in real time the traffic density and to predict it for short term periods as well as to compute more accurate maps of road traffics density. Through cooperative sensing and collective intelligence improved traffic efficiency can be improved [13].

The implementation of applications in the Cloud-to-thing-continuum, however, comes with a number of challenges, among which we could distinguish the high degree of heterogeneity of resources, the variability in computing in the *continuum*, the scale and reliability. In this context, the allocation of computing resources becomes more complex than in traditional distributed systems. Many applications may require more computing resources than single Fog/Edge nodes can provide. In this paper, we address this issue by using semantic clustering of Fog resources into virtual nodes of larger computing capacity, to which applications can be allocated. Semantic categorization of resources enables a coherent clustering of resources and thus building clusters both statically and dynamically to cope with application requirements. Then, such clusters are used as input information to optimization procedures (heuristic methods and Linear Programming) to optimize the allocation. Feasibility and efficacy of the proposed approach is evaluated *via* simulation.

The rest of the paper is structured as follows. In Sect. 2, we present the Cloud/Fog model and node clustering. Some application scenarios are given in Sect. 3. In Sect. 4 we present methods for optimizing task/application allocation to clusters of nodes. Then, in Sect. 5, we report experimental results obtained via simulation of the Fog/Edge model and application profile. We end the paper in Sect. 6 with some conclusions and future work.

2 Modelling of C2F environment and semantic clustering

We consider the architecture depicted in Fig. 2. Fog nodes could be composed of one or more layers, however in our implementation we consider just one layer between the Cloud and the edge devices.

Fig. 2: Architecture overview

As shown in Fig. 2 we first have a Cloud layer which contains the Cloud infrastructure, then the Fog layer containing the Fog nodes, which are connected to the Cloud through the Internet. The connections between the Cloud and the Fog nodes pass through several hops. The physical distance between the Cloud data centers, and the Fog nodes affects data transfer rate, since it increases both latency and potential packet loss. Fog nodes are directly connected to the edge devices, i.e. one hop away, which makes the data transfer faster, more stable and reliable.

Cloud node has abundant memory and processing power, while Fog nodes are rather limited. The edge devices in our system will just have one application each, that they need to run using certain specifications.

We describe the Cloud in Fig. 3 (left) as a system that consists mainly of a processing component, memory component, management and operating systems. The memory incorporates containers as virtualization technology for application delivery and execution. Similarly, Fog nodes as shown in Fig. 3 (right) consists of the same building blocks of the Cloud, but with far less memory and computing power. Fog nodes may benefit greatly from containers because of their lightweight and resource-friendly structure [15].

Fig. 3: Cloud architecture (left); Fog node architecture (right)

2.1 Semantic Categorization of Edge Devices, Clustering and Offloading

The benefit of creating groups of devices in Edge lies in exploiting capabilities of several devices to simultaneously and collaboratively carry out a computing intensive task for more efficiency or because the Edge devices individually cannot complete the task under the efficiency requirements, latency, etc. The devices that are part of a group can share network computing resources among themselves to meet the requirements of the tasks/applications. To do this, we first propose to apply techniques of *semantic categorization of resources* at the Fog/Edge layer. Through semantic characterization, for example through RDF, the information of Edge devices such as their computational capacity (CPU, RAM, bandwidth) and storage capacity, the specificity of each resource (heterogeneity) and *enriched with contextual information* such as location, information on mobility, security, trust, etc.

Virtual clusters Clusters of devices in Edge, referred to as virtual nodes, can be formed based on semantic information of machine-readable format according to several criteria: (1) similarity of the devices (homogeneity); (2) device location; (3) computational and reliability performance analytics of a device and (4) device behavior analytics in terms of security, privacy and trust [9, 7]. The first criterion enables clustering of similar devices in terms of their computational characteristics in order to avoid disparity between devices and that some of them could become bottlenecks in the collaborative performance of a computational task. The second criterion allows to obtain device clusters “close to” an end-user or to a data source and thus explore the “user-proximity” or “data-locality” depending on the requirements of the application. The third criterion can be taken into account when assigning tasks to groups simply to distribute the computing load. Finally, the fourth criterion would be applicable when it is necessary for processing sensitive data.

Static clusters: We can calculate *a priori* static clusters, for example using the semantic similarity criterion (*semantic clustering*) or clustering based on the location (*geo-clustering*), which can be useful for certain types of applications whose requirements are known *a priori*, for example to process data from a

stream of fixed cameras on a highway. However, static clusters would not be suitable for all types of applications and scenarios, because finding *a priori* a suitable cluster size that fits various applications and scenarios is very difficult. In particular, static clusters cannot incorporate the current state of the device workload at run-time.

Dynamic clusters: To satisfy the computational demands of the applications without knowing *a priori* their requirements, that is, at run-time, it is necessary to create clusters of devices dynamically and taking into account the computational load of the devices. To do this, it is proposed to formulate optimization problems that calculate clusters of Edge devices to be assigned to computational tasks (either workflow or bag-of-tasks) in order to optimize the execution of the tasks and optimally use the Edge resources.

2.2 Offloading algorithms

Offloading algorithms is a class of algorithms that allow to delegate or offload the computation of tasks between parts of the same system or between different computer systems. The reasons for offloading the computation from an entity/system $\mathcal{A}$ to an entity/system $\mathcal{B}$ can be diverse (one or more at a time) but all of them try to take advantage of the $\mathcal{B}$ resources or features:

1. To perform $\mathcal{A}$ tasks in a parallel or distributed way. In this case, the aim is to increase resource utilization and system performance.
2. To relieve the computational burden of $\mathcal{A}$ so that $\mathcal{A}$ can achieve greater throughput and a shorter completion time/shorter response time.
3. Because $\mathcal{A}$ cannot meet the requirements of a task or application, which $\mathcal{B}$ can meet, such as real-time response/processing, memory capacity, disk storage, security and privacy requirements, or because we need information that is in $\mathcal{B}$ (e.g. $\mathcal{B}$ is a Cloud that hosts a database of human faces).
4. So that $\mathcal{A}$ can achieve large-scale computing (in dataset size, in number of devices, transactions, end users, etc.)
5. So that $\mathcal{A}$ can consume less resources, for example, to save energy.

Offloading algorithms arise very naturally in a Cloud-to-thing continuum systems [20,21] (see also Sect. 3).

3 Application scenarios

3.1 Road safety

In this application scenario the aim is to process a flow of car sensory data to detect areas of a road in poor condition (*potholes*). The detection of such areas of a road is of interest for several reasons: keeping the road in good

Fig. 4: Anomaly detection of car's wheels speed (left). Heat Map of San Francisco, USA (right)

condition, increasing the safety of road users, avoiding accidents (especially motorcyclists), avoiding damage to cars that cross potholes inadvertently, etc.

To that end, data from the CAN bus of a vehicle or data collected from an OBU can be used. The CAN bus connects different electronic control units within a vehicle and provides valuable information on vehicle state, tracking driver behavior, etc. In particular, it provides information on the speeds (*rpm*) of the car's wheels (see Fig. 4 (left) for an exemplification). The CAN bus data contains a lot of information but precisely the rotational information of each wheel is useful to spot potholes by detecting abnormal speeds in any of the wheels: when a car goes through a pothole, the rotational speed of the car's wheel(s) is significantly altered. In addition, this data stream could arrive combined with GPS data of the geographical position of the car.

This application fits into the scenario of *event stream processing at Fog/Edge* assuming an Edge infrastructure of RSU-Road Side Units for data collection and pre-processing, and Fog nodes for anomaly detection processing (see [27] for more details.)

3.2 Prediction of the volume of traffic by images of mobile crowd-sensing

This application scenario aims to process a data flow of images sent by smartphones (*mobile crowd-sensing*) to predict the volume of traffic in real time in large cities. Traffic jams in large cities constitute a social problem and have significant economic costs (see Fig. 4 (right) for an exemplification). An alternative to calculating the congestion map of a city in real time and to make the prediction of traffic volume is to apply crowd-sensing techniques.

Specifically, thousands of drivers can send images of the state of the roads in a city in real time, forming a stream of images and corresponding positions whose real-time processing can provide valuable information on the current state and predict traffic in the city. The premise is in the large number of images that allows to achieve the prediction with high precision. This problem is defined as predicting real-time spatial information from data collected by mobile IoT sensors.

Therefore mobile crowd-sensing constitutes an alternative to the costly deployment and maintenance of video-camera infrastructures throughout a city, especially in large cities where the cost of deployment and maintenance can be very high. However, crowd-sensing has intrinsic problems and challenges:

- Driver participation is voluntary and mass participation is not guaranteed.
- The quality of the images may be poor or just *fake*, which could affect the accuracy of the prediction.
- The images can reveal information about the license plates of the cars and the information received must be anonymized.

The first issue of volunteer participation of drivers can be addressed by implementing (in Fog/Edge) participation incentive schemes (a problem of interest to Vehicular Networks: [24,14]). Similarly, the second point can be addressed by implementing a classifier based on an image quality model of *image features* that allow deducing the contribution of an image in predicting traffic volume (low-quality images simply would not enter the system) and thus to be able to implement an efficient classifier of the images. Finally, for the third problem, we can apply the state of the art to anonymize the license plates of cars in the images. Such steps can be achieved in a Fog/Edge architecture as the one proposed in this paper. The reader is referred to [23] for more details.

4 Optimization of task/application allocation

The responsibility of the solver in our system, is to perform the mapping between the tasks in an application to Fog nodes within a cluster. Its function is to find which Fog node is going to execute which tasks of an application. Thus, mapping could be considered as an optimization problem. The problem could be described as a variant to the bin-packing problem [4], where we have a number of items that we want to place in a number of bins, with a common capacity and minimize the number of used bins. Unlike the traditional problem, we will have different capacities to represent the memory of the Fog nodes. We first apply ILP –Integer Linear Programming– to get the optimum solution, then we solve the problem using a First Fit Heuristic Algorithm (FFHA) [28] approach to decrease the run-time of the optimization.

The objective is minimizing both the number of Fog nodes used, and the processing time by allocating as many tasks to Fog nodes with higher CPU as possible. We formalize the problem as follows.

Objective:

$$\min \sum_{f \in F} f_{p_t} \times f_{used}$$

where:

- f : Fog node.

- F : Set of Fog nodes.
- f_{p_t} : processing time of node f .
- f_used : equals 1 if node f is used, 0 otherwise.

subject to:

- constraint 1:

$$\forall t \in T : \\ \sum_{f \in F} x_{tf} = 1$$

- constraint 2:

$$\forall f \in F : \\ \sum_{t \in T} t_size \times x_{tf} \leq f_{RAM} \times f$$

where:

- x_{tf} : equals 1 if task t is placed in Fog node f , 0 otherwise.
- t_size : Size of task t
- f_{ram} : RAM capacity of f .

Constraints 1 and 2, make sure that a task can only exist in one Fog node, and that the sum of task sizes cannot exceed the memory capacity of the Fog node they are placed in respectively.

For the ILP method, we use the *PuLP* Python package to solve the optimization problem. *PuLP* can generate Linear Programming (LP) files and call GLPK [17], COIN CLP/CBC [5], CPLEX [3], and GUROBI [10] to solve linear problems. PuLP uses Coin-OR Branch and Cut (CBC) as its default solver, which is an open-source mixed integer programming solver in C++.

For the FFHA, we define our own algorithm to solve our specific problem, where we have a variable number to represent Fog node's RAM (see pseudo-code of the FFHA Alg. 1).

Algorithm 1 First Fit Heuristic Algorithm (FFHA)

```

for  $Fogs = 1, 2, \dots$  do
  for  $index = 1, 2, \dots, inRange\ FogSizes.length$  do
    Assign  $taskSize$  in  $sizes$  to  $sizes[index][1]$ 
    if  $filledFogselectedFogSize \dots$  then
      if  $taskSize \leq (FogSize - filledFogSize)$  then
         $filledFog + taskSize$ 
      else
        break
      end if
    end if
  end for
  Return assignment results
end for

```

Fig. 5: Simulation with 5 Fog nodes and 10 Edge devices

As mentioned before, every time we run the simulation, the Fog nodes and the edge devices are generated in different positions with random attributes. Fig. 5 shows an instance of our simulation with 5 Fog nodes and 10 edge devices, where the nodes are represented using color coded circles. The Cloud node in red, the Fog nodes in blue and the edge devices in green. As we can see, all Fog nodes are connected to the Cloud node, while the lines between the edge devices and the Fog nodes represent if a device is in the range of a Fog node. The actual links are determined after the aforementioned calculations, to get the optimal connections. The connections between Fog nodes and other Fog nodes show that a neighbor cluster was formed. In this particular case, it happened between nodes **f1** and **f3**. As we can see edge devices **d1**, **d3**, **d9**, and **d10** did not fall in the range of any of the Fog nodes, therefore they connected directly to the Cloud.

5 Experimental evaluation

To assess the performance of our implementation, we benchmark the overall process by running the simulation with different sets of numbers for the Fog nodes and the devices. Since the semantic reasoning and clustering directly affects the overall all performance, we can test for how the system scales by increasing the number of the Fog nodes and edge devices. Moreover, we compare

between the two solvers we used by focusing on the differences in run-times, and the application response times obtained from various possible scenarios:

- The application is directly sent to the Cloud
- The application is sent to the Cloud through a Fog node
- The application is sent to a single Fog node
- The application is sent to an In-range cluster
- The application is sent to a neighbor cluster

We evaluate the utility of the two proposed clustering techniques, by comparing the response times obtained from each method.

5.1 Experimental setting and results

All tests were executed on a single machine with the following specifications:

Operating system: Windows 10 Pro; **CPU:** Intel Core i7-4710HQ @ 2.5 Ghz, base Clock Speed up to 3.5 Ghz Turbo Speed, 6MB L3 cache; **Memory:** 16GB DDR3 @ 1600 MHz, Dual channel and **Storage:** 500 GB NAND Sata SSD

Since the simulation generates random attributes for the Fog nodes, the edge devices and the connections, we have to run the simulation for number of times and calculate the means of the values to acquire decisive results.

On one hand, for bench-marking and comparison between the run-times of the solvers, we ran the simulation using different sets of numbers for the Fog nodes and edge devices. For each set of numbers, we ran the simulation 50 times with the ILP solver, and 50 times with the FFHA. On the other hand, for the response time analysis, we run the simulation with the same number of Fog nodes and edge devices 50 times. In that manner, we focus on how the clustering techniques affect the response times.

5.2 System performance

To evaluate the optimization techniques used to solve the task allocation problem, we focus on the response times of the applications in different scenarios, and we measure the average run-time for all the scenarios. We ran the simulation with the following sets of parameters 50 times with the ILP solver and 50 times with the FFHA.

- **5** Fog nodes with 100, 250 and 400 edge devices
- **10** Fog nodes with 100, 250 and 400 edge devices
- **20** Fog nodes with 100, 250 and 400 edge devices

As expected, the results show that the ILP solver achieves slightly better response times than that of the FFHA, in the case of running the simulation with 5 Fog nodes and 100, 250, and 400 devices as shown in Fig. 6.

When running the simulation with 10 Fog nodes and the same sets of devices, the results show that in some cases, the FFHA gets better response times compared to the ILP solver as shown in Fig. 7.

(a) 5 Fogs-100 devices

(b) 5 Fogs-250 devices

(c) 5 Fogs-400 devices

Fig. 6: Response times for 5 Fogs and 100, 250, 400 devices (ILP - FFHA)

Fig. 8 shows the results when running the simulation with 20 Fog nodes. We can see that the results are similar to running the simulation with 10 Fog nodes. Where the ILP solver performs slightly better than the FFHA, yet the difference is not significant.

Overall, regarding the application response times, given the different methods of sending the application, the results gathered from running the simulation with the aforementioned sets of numbers for the Fog nodes and the devices, we can deduce that the ILP solver has a slight advantage over the FFHA. However, the difference is rather minor. That is due to the fact that we perform the memory checks, as an initial filter, then the solvers optimize where the tasks are processed to get the least possible response time.

(a) 10 Fogs-100 devices

(b) 10 Fogs-250 devices

(c) 10 Fogs-400 devices

Fig. 7: Response times for 10 Fogs and 100, 250, 400 devices (ILP - FFHA)

In contrast, when reviewing the run-times of the simulations, there is a substantial difference between the performance of the ILP solver and the FFHA, where the FFHA performs much faster than the ILP solver as shown in Fig. 9.

On the one hand, increasing linearly the number of edge devices affects the run-time. On the other hand, we can see an exponential increase in run-time when increasing the number of Fog nodes, which is not surprising, since this increases the number combinations when forming the clusters.

To measure the consistency of the optimization techniques, we calculate the standard deviation, of both the response times and the run-times by conducting another experiment, using an instance size of 10 Fog nodes and 100 devices, and 100 iterations. Additionally, we calculated the number of Fog nodes in the clusters from the same instance, to give insight on the size of the clusters.

(a) 20 Fogs-100 devices

(b) 20 Fogs-250 devices

(c) 20 Fogs-400 devices

Fig. 8: Response times for 20 Fogs and 100, 250, 400 devices (ILP - FFHA)

Fig. 9: Run-times (ILP vs FFHA)

Tables 1 and 2 show the average response times and run-times with their corresponding Standard Deviations (STD), when running the simulation using the ILP solver and the FFHA respectively.

Table 1: ILP - Average and Standard deviations of response times & run-time

ILP		
	Avg response times	Response times STD
Cloud direct	17.1991722	8.71893098
Cloud through Fog	2.18383078	0.23542268
Single Fog	0.43523073	0.10711002
In-range cluster	1.09988548	0.21959514
Neighbor cluster	0.88225381	0.21159424
	Avg run-time	Run-time STD
	3.82993240	0.81419498

Table 2: FFHA - Average and Standard deviations of response times & run-time

FFHA		
	Avg response times	Response times STD
Cloud direct	18.2210832	10.5720824
Cloud through Fog	2.19942492	0.26223306
Single Fog	0.45561651	0.11327142
In-range cluster	1.09464597	0.19415273
Neighbor cluster	0.93392979	0.22757457
	Avg run-time	Run-time STD
	0.22792500	0.09110830

When comparing the run-times STD, we can see that the FFHA behaves more consistently than the ILP solver, with less deviation.

Table 3 presents the average number of Fog nodes in both the in-range clusters and the neighbor clusters. We can see that the numbers are marginally higher when using the FFHA, which indicates that when using the ILP solver, we obtain clusters containing less Fog nodes. Conclusively, the results reflect our choice of Fog node and application properties.

Table 3: Average cluster size

Cluster size		
Type	ILP	FFHA
In-range cluster	2.713035699	2.912991401
Neighbor cluster	3.294254702	3.50243713

5.3 Utility of Fog clustering

We focus mainly on two aspects to evaluate the utility of our proposed clustering techniques. First, we analyze the overall response times gathered from the different methods of sending the application, regardless of the optimization technique. Next, we consider the number of devices being served by the Fog layer when no clustering technique is applied, using only in-range clustering, and using both in-range and neighbor clustering.

Fig. 10: Average response times of different methods of sending applications

Fig. 10 shows the difference in application response times when sending the application using different methods. As we can see, the difference between sending the application directly to the Cloud and utilizing the Fog paradigm is huge, which is expected due to the physical distance and number of hops between the devices and the Cloud. Moreover the data rate of the connection is much slower compared to the ones between the Fog nodes and the devices. Processing an application in a single Fog node provides the fastest response time, since no task allocation optimization takes place, and also because that application is just being sent over one link.

Our proposed clustering techniques deliver relatively fast response times, while enabling the service of a larger number of applications, with more demanding requirements. Fig. 11, presents the percentages of applications being served regarding the different scenarios of how the applications are being sent and processed; directly to the Cloud, Cloud through a Fog node, a single Fog node, in-range clusters and neighbor clusters. We obtained the percentages shown in Fig. 11, by running the simulation using 10 Fog nodes and 100 edge devices for 100 iterations, without any clustering techniques, with only the In-range clustering, and with both In-range and neighbor clustering. The results

show a significant decrease in applications being sent to the Cloud when applying the clustering techniques, which in turn results in faster response times overall.

Fig. 11: Percentage of applications sent using different methods

6 Conclusions and future work

In this paper we have presented and evaluated optimization methods, which use semantic clustering of nodes, for allocation of tasks and allocations to Fog nodes in a Cloud-Fog computing environment. Our implementation shows that there is a massive gain in response times when utilizing the Fog computing paradigm, compared to just relying on the Cloud. After analyzing the results, it became apparent that the application response times are fundamentally faster when utilizing Fog nodes or clusters. While it is faster to process an application in a single Fog node, than to divide its tasks among several nodes within a cluster, some applications are more demanding and will require more resources than those of a single Fog node. Although the response times are slower when processing an application in a cluster, our implementation addresses the issue of processing more demanding applications in the Fog layer. Regarding the scalability of our implementation, one notable observation is that the system starts experiencing crashes when simulating with more than 30 Fog nodes and 2000 edge devices.

In our future work we would like to simulate more application profiles as well as to use features of real-life applications scenarios, from application profiling sources.

Acknowledgements This work is supported by Research Project, “Efficient & Sustainable Transport Systems in Smart Cities: Internet of Things, Transport Analytics, and Agile Algorithms” (TransAnalytics) PID2019-111100RB-C21/AEI/ 10.13039/501100011033, Ministerio de Ciencia e Innovación, Spain

References

1. Ahmad, A., Lavin, A., Purdy, S. and Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* Volume 262, 2017, Pages 134-147.
2. Carletti, M., Masiero, Ch., Beghi, A. and Susto, G.A. A deep learning approach for anomaly detection with industrial time series data: a refrigerators manufacturing case study. *Procedia Manufacturing*, Volume 38, 2019, Pages 233-240.
3. Ceselli, A., Fiore, M., Premoli, M. and Secci, S. Optimized assignment patterns in Mobile Edge Cloud networks. *Comput. Oper. Res.* 106: 246-259 (2019)
4. Coffman Jr. E.G., Csirik J., Galambos G., Martello S. and Vigo D. Bin Packing Approximation Algorithms: Survey and Classification. In: Pardalos P., Du DZ., Graham R. (eds) *Handbook of Combinatorial Optimization*. Springer, New York, NY. 2013 https://doi.org/10.1007/978-1-4419-7997-1_35
5. COIN-OR (Common Infrastructure for Operations Research), <http://www.coin-or.org>
6. Corizzo, R., Ceci, M., Japkowicz, N. Anomaly Detection and Repair for Accurate Predictions in Geo-distributed Big Data. *Big Data Research*, Volume 16, 2019, Pages 18-35.
7. Ferraris, D., Gago, M. C. F., López, J. A trust-by-design framework for the internet of things. In *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018, Paris, France, 2018*, Pages 1-4.
8. Fouchal, H., Bourdy, E., Wilhelm, G., Ayaida, M. A validation tool for cooperative intelligent transport systems. *J. Comput. Sci.* 22: 283-288, 2017.
9. Gago, M. C. F., Moyano, F., López, J. Modelling trust dynamics in the internet of things. *Inf. Sci.*, 396:72-82, 2017.
10. Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2020, <http://www.gurobi.com>
11. Hawley-Hague, H., Boulton, E., Hall, A., Pfeiffer, K., Todd, Ch. Older adults perceptions of technologies aimed at falls prevention, detection or monitoring: A systematic review. *International Journal of Medical Informatics* Volume 83, Issue 6, Pages 416-426, 2014.
12. Faulin, J., Grasman, S.E, A. Juan, A.A., Patrick Hirsch, P. *Sustainable Transportation and Smart Logistics. Decision-Making Models and Solutions*. Elsevier, 2019.
13. Ferreira, J., Alam, M., Fernandes, B., Silva, L., Almeida, J., Moura, L., Costa, R., Iovino, G., Cordivola, E. Cooperative sensing for improved traffic efficiency: The highway field trial. *Comput. Networks* 143: 82-97, 2018.
14. He, Z., Cao, J., Liu, X. (2015). High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Hong Kong, 2015*, Pages 2542-2550.
15. Oracle Linux, Linux Containers, 2019, <https://docs.oracle.com/en/operating-systems/oracle-linux/6/adminsg/ol.about.containers.html>
16. Mfenjou, M.L., Abba Ari, A., Njoya, A.N., Fots, D.J., Kolyang, M., Abdou, W., Spies, F. Control points deployment in an Intelligent Transportation System for monitoring inter-urban network roadway. *Journal of King Saud University - Computer and Information Sciences*, Available online 21 October 2019, In Press. <https://doi.org/10.1016/j.jksuci.2019.10.005>
17. Makhorin, A. GLPK (GNU Linear Programming Kit), <http://www.gnu.org/software/glpk/glpk.html>
18. Hussain, M., Alam, M.S, Beg. S. Vehicular Fog Computing-Planning and Design. *Procedia Computer Science*, Volume 167, 2020, Pages 2570-2580
19. Onggo, B.S., Mustafee, N., Smart, A., Juan, A.A. and Molloy, O. Symbiotic simulation system: hybrid systems model meets big data analytics. In *Proceedings of the 2018 Winter Simulation Conference (WSC 18)*. IEEE Press, 13581369, 2018.
20. Sánchez, L., Muñoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutiérrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis, E., and Pfisterer, D. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217-238, 2014.
21. Santa, J., Fernández, P. J., Sanchez-Iborra, R., Murillo, J. O., and Skarmeta, A. F. Offloading positioning onto network edge. *Wireless Communications and Mobile Computing*, Volume 2018 7868796:1-7868796:13, 2018.
22. Santa, J., Bernal-Escobedo, L., Sanchez-Iborra, R. On-board unit to connect personal mobility vehicles to the IoT. *FNC/MobiSPC 2020*: 173-180

23. Shinkuma, R., Takagi, R., Inagaki, Y., Oki, E., Xhafa, F. Incentive Mechanism for Mobile Crowdsensing in Spatial Information Prediction Using Machine Learning. *AINA* 2020: 792-803
24. Wang, X., Wu, W., and Qi, D. Mobility-aware participant recruitment for vehicle-based mobile crowdsensing. *IEEE Trans. Vehicular Technology*, 67(5):4415–4426, 2018.
25. Wang, X., Ahn, S.H. Real-time prediction and anomaly detection of electrical load in a residential community. *Applied Energy*, Volume 259, 2020.
26. Xhafa, F. The Vision of Edges of Internet as a Compute Fabric. Chapter 1, In *Advances in Edge Computing: Massive Parallel Processing and Applications. Book Series: Advances in Parallel Computing Series*. IOS Press, 2020
27. Xhafa, F., Kilic, B. and Krause, P. Evaluation of IoT stream processing at edge computing layer for semantic data enrichment Future Generation Computer Systems, volume 105, pages 730-736, 2020. DOI: 10.1016/j.future.2019.12.031
28. Yesodha, R. and Amudha, T. A Comparative Study On Heuristic Procedures To Solve Bin Packing Problems, *International Journal in Foundation of Computer Science '—&' Technology (IJFCST)*, Volume 2, 37-49, 2012, doi = 10.5121/ijfcst.2012.2603
29. Zhang, H, Lu, X. Vehicle communication network in intelligent transportation system based on Internet of Things. Special Issue on Internet of Things and Augmented Reality in the age of 5G. *Computer Communications*, Volume 160, 2020, Pages 779-789