A fog-based approach for distributed cloud testing: a VR-assist rental testing case study

Fatima Zahra Moutai*, Sara Hsaini, Salma Azzouzi and My El Hassan Charaf

Faculty of Science,
Ibn Tofail University,
Kenitra, Morocco
Email: fz.moutai@gmail.com
Email: hsaini.sara@gmail.com
Email: salma.azzouzi@gmail.com
Email: charaf@gmail.com
*Corresponding author

Abstract: Cloud computing offers the opportunity to improve productivity and reduce costs. However, this technology faces high latency problems. In this context, a decentralised cloud paradigm has been introduced in order to bring geographically cloud services close to users. Moreover, the growth of personal devices has stimulated cloud services with further requirements such as compliance (conformance) with a standard or high traffic volume. To cope with these problems, we need assessments to test for example how these requirements can affect the performance degradation in such environments and how we can enhance the distributed cloud architecture in order to perform the end-users objectives. The main contribution in this paper is to suggest a distributed testing approach derived from 'fog computing' design and leveraging its advantages especially optimising response time requirement. To this end, we present a comprehensive survey on distributed cloud architectures, and then we formally suggest a new architecture for testing in such environments. Finally, we illustrate our approach through a case study.

Keywords: distributed testing; distributed cloud; edge; core; internet of things; IoT.

Reference to this paper should be made as follows: Moutai, F.Z., Hsaini, S., Azzouzi, S. and Charaf, M.E.H. (2022) 'A fog-based approach for distributed cloud testing: a VR-assist rental testing case study', *Int. J. Powertrains*, Vol. 11, No. 1, pp.1–18.

Biographical notes: Fatima Zahra Moutai is a PhD candidate in Computer Science at the Laboratory of Informatics, Systems and Optimization 'ISO-LAB' at the Faculty of Science-Ibn Tofail University-Kenitra, Morocco. Her research interests include essentially: distributed testing, problems of controllability, observability and synchronisation as well as distributed testing in the cloud computing field. She has been a member of the communication committee of previous editions of the International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS) as well as a member of the technical program committee of many international conferences in the area of control and computer science.

Sara Hsaini is a PhD candidate in Computer Science at the Laboratory of Informatics, Systems and Optimization 'ISO-LAB' at the Faculty of Science-Ibn Tofail University-Kenitra, Morocco. Her research interests include essentially: distributed testing, problems of controllability, observability and synchronisation, timing constraints in distributed testing and big data. She is a member of the technical program committee of many international conferences in the field of control and computer science. She has been also a member of the communication committee of previous editions of the International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS).

Salma Azzouzi received the State Engineer degree in Computer Science from the National Institute of Statistics and Applied Economics (INSEA), Morocco, in 2006 and later, she received her PhD in Computer Science from the IBN TOFAIL University, Morocco. She is currently a Professor in the Faculty of Science and a member of Laboratory of Informatics, Systems and Optimization 'ISO-LAB' at the Ibn Tofail University. Her research interests include essentially: distributed testing, problems of controllability/observability, control, timing constraints in distributed testing, and cloud computing. She is the guest editor of the journal: *Statistics, Optimization & Information Computing*. She has been the chairman or co-chair at many research conferences.

My El Hassan Charaf received a state engineering degree in Computer Science in 2004. Later, he received his PhD in Computer Science from the Ibn Tofail University in 2013. He is currently a Professor of Computer Science and a member of the Laboratory of Informatics, Systems and Optimization 'ISO-LAB' at the Faculty of Science-Ibn Tofail University-Kenitra, Morocco. His research interests include: distributed testing, control, distributed artificial intelligence, software testing and big data. He is the guest editor of the journal: *International Journal of Modelling, Identification and Control*. He has been the chairman or co-chair at many research conferences.

This paper is a revised and expanded version of a paper entitled 'Testing in distributed cloud: a case study' presented at International Symposium on Advanced Electrical & Communication Technologies (ISAECT2019), Italy, Rome, 27–29 November 2019.

1 Introduction

Over past few years, cloud computing has been considered one of the most advanced paradigms that have made computing resources publicly available on the internet. However, even though cloud computing has benefited from advances in communication technologies, in particular the increase in bandwidth, there is still a major effort to be made to improve the latency due to real-time services and high traffic volumes, if a large number of users simultaneously request cloud services at the same location.

In the context of internet of things (IoT), for example, cloud computing offers important resources for improving the IoT infrastructure, such as computational power and storage capacity. However, this paradigm has some drawbacks induced in the intercommunication between an IoT device and the cloud. Currently, the most common approach is to centralise the processing of data in the cloud on a single site in order to reduce costs and ensure high application security. However, with the huge amount of data

collected from these devices spread around the world. This approach remains inefficient especially if the data is sensitive and requires quick response time.

In general, data generated from IoT devices can be processed at three locations: cloud, network, or device itself. If data is processed in the cloud, it will take a lot of time to be analysed. Thus, it is better to process it either on the network or in the device. Both the technologies bring intelligence and data to analytic platforms located close to the source of data generation. To this end, many concepts and techniques have been introduced to drive a new generation of distributed clouds such as fog computing, edge cloud computing (ECC) or mobile edge cloud (MEC) (Nguyen et al., 2018). In fact, such distributed clouds refer to micro-clouds where we can make computation, storage and networking instead of the main cloud. For instance, the idea of ECC is to bring the cloud services or part of them to the edge network located in the geographical proximity of the users in order to reduce latency, network congestion and ensure quality of service (Shi et al., 2016). Moreover, Fog computing is a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum OpenFog Consortium (2017). Its means essentially, extending computing to the edge rather than hosting and working from a centralised cloud. This facilitates processing data locally in smart devices and smoothes the computation operations, storage, and networking services between end devices and data centres. To this end, a new layer is added in the IoT architecture between IoT devices and the cloud to ensure the service quality (Moysiadis et al., 2018).

On the other hand, testing the service quality in a distributed context is a complex and expensive task, and testing in a distributed cloud environment is still more difficult and need to set high procedure requirements for deployment of such systems.

Nowadays, many works has been devoted to cloud testing. Researchers have realised the potential of cloud testing to address and reduce the problem of high testing costs.

In fact, the cloud makes it possible to develop and support expensive test infrastructures and to exploit on-demand resource configuration for performance testing. Therefore, the main contribution in this paper is to suggest a distributed testing architecture derived from Fog Computing design and leveraging its advantages such as optimising response time requirements. Furthermore, we propose a VR-assist rental testing case study to illustrate our approach.

As novelties, we aim to optimise the test process by reducing the messages exchanged in the cloud especially; we minimise number of coordination messages exchanged between distributed testing components. Furthermore, connected objects require the transfer of data directly to the cloud for processing. However, the cost of the data transfer between the endpoints (things) and the cloud can be significant. For this purpose, we suggest handling data closer to the endpoints and ensuring coordination of multiple endpoint services in a consistent manner. Finally, we suggest to group ports of communications according to regions in the cloud. The test is made then at each region accordingly and the coordination is ensured between regions.

This article is organised as follows: Section 2 gives some preliminaries. In Section 3, we present the distributed cloud for IoT architecture. In Section 4, we introduce some related works. Section 5 presents our architecture for distributed testing in the cloud. In Section 6, we present a case study to illustrate our approach. Finally, we provide a discussion in Section 7. The last section gives conclusions and identifies future works.

2 Preliminaries

2.1 Internet of things

The IoT refers to the different physical devices that are now connected to the internet, collecting and sharing data. These devices have a unique identifier and take advantage of cloud computing. Nowadays, IoT manufacturers and application developers are starting to discover the benefits of performing calculation and analysis on these devices instead of the cloud. This approach reduces the latency for critical applications and avoids dependency on the cloud which helps to manage the huge amounts of data generated by these devices efficiently.

2.2 IoT and cloud computing

Cloud computing offers significant resources to IoT networks, such as computational power and storage capacity. However, this paradigm has certain drawbacks such as the latency induced in the intercommunication between an IoT device and the cloud. The two paradigms of cloud and IoT have seen a rapid evolution. These concepts are very different from each other and their characteristics are often complementary. In fact, cloud can offer a solution for IoT service management for implementing applications and services that exploit the things or the data produced by the device. On the other hand, cloud can benefit from IoT by extending its scope to deal with real world things in a more distributed and dynamic manner (Mora et al., 2019).

3 Distributed cloud for IoT architecture

3.1 Fog vs. edge computing

Fog computing is a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing path (OpenFog Consortium, 2017). In this context, we extend computing to the edge rather than hosting and working from a centralised cloud. This facilitates the local processing of data in smart devices and smoothes the operation of compute, storage, and networking services between end devices and cloud computing data centres. In the IoT context, a new layer is added in the IoT architecture between IoT devices and the cloud as illustrated in Figure 1 (Moysiadis et al., 2018).

The word ‘edge’ has several meanings. A common usage of the term refers to the edge network as opposed to the core network, with equipment such as edge routers, base stations, and home gateways (Chiang et al., 2017). Edge computing architectures place servers, applications, or small clouds at the edge as shown in Figure 2.

3.2 Brief comparison

Fog computing and edge computing are two fairly similar infrastructures. In fact, fog computing has a hierarchical and flat architecture with several layers forming a network, while edge computing relies on separate nodes that do not form a network. Moreover, fog computing has been referred to as edge computing while edge computing is supposed to

resolve problems by storing data close to the ground. In other words, it stores data in local computers and storage devices, rather than routing all the information through a centralised data centre in the cloud. Table 1 presents a comparative study of fog and edge computing (Francis, 2018).

Figure 1 Fog computing architecture (see online version for colours)

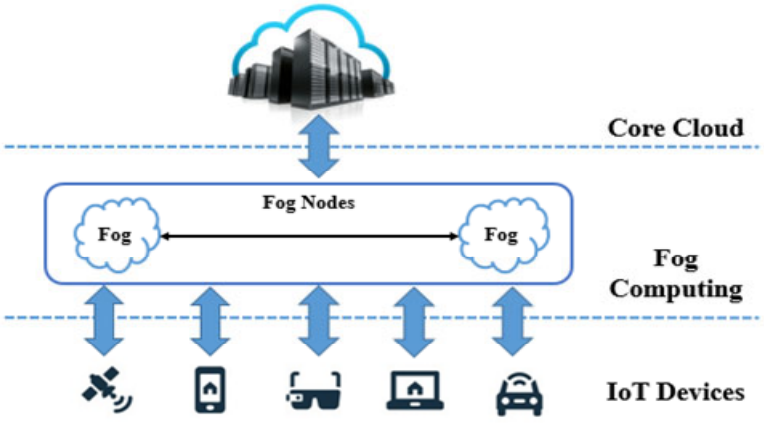
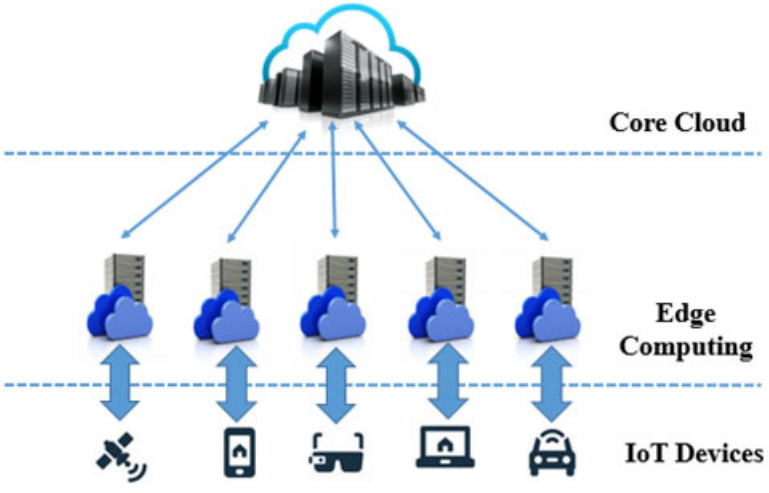


Figure 2 Edge computing architecture (see online version for colours)



By referring (Nguyen et al., 2018), we give a review of the main benefits of distributed cloud to provide services:

- *Location transparency*: A cloud service is provided from any cloud node, that is, an edge cloud, a regional cloud, a core cloud, or combination of them.
- *High real-time*: A cloud service on the distributed cloud environment can be provided from edge cloud located in the proximity of the cloud service customers, so that it can guarantee the QoS of high real-time applications.

- *High mobility*: The path of a cloud service is changed, but cloud service is still provided from a main cloud (i.e., core cloud).
- *Load detachability*: When a cloud service customer requests a cloud service to a core cloud on the distributed cloud environment, the edge clouds and the regional clouds between the core cloud and the cloud service customer can be aware of the cloud service and accomplish the cloud service instead of the core cloud.

Table 1 Fog vs. edge computing.

Fog computing	*Edge computing*
Deployed at the local premises of mobile users	Deployed as a traditional data centre with extended capabilities
Virtualised device with build-in data storage, computing, and communication facility	Uses an edge server similar to a traditional data centre server
Can be adapted from existing system components	It is completely built as new system or a mini cloud data centre
Energy consumption of fog is less than cloud services, but overhead is high compared to cloud	Edge uses less resources than the cloud and initial overhead to build is high compared to cloud
No central entity controlling the fog cloud	Edge server using cloud technologies and virtualisation used to control edge components
May not be controlled by network operators, uses an ad-hoc distribution	Allows the mobile network operators to improve existing services with edge
Has own security and load balancing	Limited security, distributed load balancing

3.3 Distributed cloud architecture

The distributed cloud consists of a core cloud, a regional cloud and an edge cloud, as shown in Figure 3:

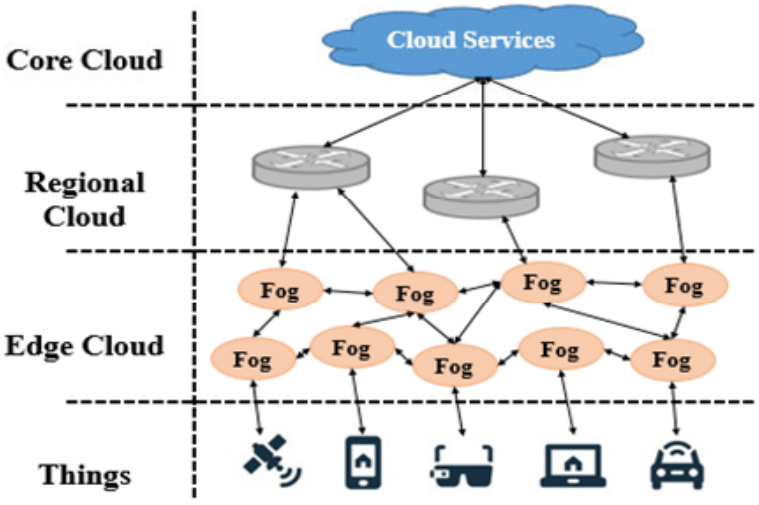
- A core cloud is the heart of this paradigm, it has the capabilities of all conventional cloud work, including management and provision of cloud services/data and it contains all cloud services.
- A regional cloud ensures communication by sharing information between different layers.
- An edge cloud is located close to the customers, it communicates directly with the users. In what follows, we consider the fog computing as example of edge cloud layer.

In general, a core cloud has the largest scale and exists on the upper layer of the distributed cloud. A regional cloud is smaller and exists on the middle layer of the distributed cloud and has proxy capabilities to support high mobility of devices, as well as self-deploying cloud service images and caching of data.

The edge cloud has the smallest scale and exists on the lower layer of the distributed cloud. Generally, the edge cloud exists in the proximity of users having the ability to know the device, service, location, preferences, etc., and then deploy and deliver real-time cloud services (Nguyen et al., 2018).

Finally, all these components can be managed using diverse administrative domains as their starting points are different.

Figure 3 Distributed cloud architecture (see online version for colours)



3.4 Testing in the distributed cloud context

Cloud computing and distributed systems are closely related. Cloud technology adds new organisational and technological ways to create infrastructure for distributed systems (Tómasson, 2011). Moreover, testing distributed systems is still more difficult due to coordination, synchronisation and security issues.

We focus in the following on conformance testing that is a type of functional testing of a black box nature where the source of the implementation under test (IUT) is unknown and its behaviour is checked with respect to a specification.

The massive body of research in testing distributed systems – especially distributed cloud systems – has been done in last few years. In what follows, we outline some of these research proposals as well as some of our works I the field.

4 Literature review

Oriol and Ullah (2010) discuss and propose a testing tool for distributing software testing over multiple computers in a cloud, for performance gain in running the tests. Cloud9 (Ciortea et al., 2010) is a platform for automatic testing which parallelises symbolic execution, but still poorly scalable in test automation technique for cloud infrastructures.

Sudharson and Prabha (2020) suggest an efficient model for identifying the faults in the software products using improved EM algorithm using Gaussian function. The improved EM function includes k-means algorithm together to perform the task.

By using their designed test bench, the authors in Esfandyari et al. (2017) show that all control logics of the drive motor management software (DMMS) can be evaluated and this will reduce the development costs and time immensely.

King and Ganti (2010) describe a testing service in the cloud called test-support-as-a-service (TSaaS). They aim to use the processing capabilities provided by the cloud to improve self-testing processes. The TSaas provides cloud customers automated test operations for remotely hosted cloud services.

The work (Tómasson, 2011) investigates the challenges of software testing in elastic cloud computing environments and the applicability of the Jata test framework for testing distributed cloud applications. Furthermore, Felemban et al. (2013) describe the design requirements for mobile cloud computing (MCC) architecture. The novelty of their architecture is to integrate a cloudlet and base station subsystem that can meet application-level quality of service requirements and allow mobile resource provisioning close to the user.

In Shi et al. (2016), the authors introduce a new simulation framework, so-called ECSim++, caching communication or energy efficiency related issues in ECC. The work D-Cloud (Felemban et al., 2013) for testing of cloud applications enables automation of system configuration and the ability to run test cases simultaneously with the ability to emulate hardware faults.

Aymen and Mahmoudi (2019) examine ways to develop a better understanding of how EVs can achieve energy use optimisation and be connected with a smart city. The study is based on an original idea that would be useful in informing transport operators of how to improve and embrace better EV technologies in the context of smart cities. The proposed approach is based on vehicles' and buildings' communication to share some special information related to the vehicles' status and to the road conditions.

In Mahmoudi et al. (2019), the authors introduce a new learning technique concept based on cloud experience exchange between electric vehicles. They aim to build a better EV experience in power management through cloud sharing and definitely cut with conventional architecture that may have reached its boundaries.

Mora et al. (2019) describe a computational model of a multilayer architecture for increasing the performance of devices using the MCC paradigm. The main novelty of this work lies in defining a comprehensive model where all the available computing platforms along the network layers are involved to perform the outsourcing of the application workload.

In this paper, we aim to improve our distributed testing architecture to deal with distributed cloud. Therefore, we extends previous works done in the field where we suggest to use some artificial intelligence features to avoid problems of coordination in distributed platforms (Charaf et al., 2013, 2014, 2017; Azzouzi et al., 2015,2020; Moutai et al., 2019a, 2019b, 2020).

In the next section, we review the common distributed testing issues and we explain formally our model over a distributed cloud environment. Afterwards, we give a case study to illustrate our approach.

5 Our approach for distributed cloud testing

5.1 Architecture

The basic idea of distributed cloud testing architecture is to coordinate communication between fog-testers and the IUT. An IUT is the implementation of the distributed application to be tested in the core cloud. It can be considered as a 'black-box'; its

behaviour is known only by interactions through its interfaces with the environment or other systems.

Figure 4 Distributed cloud testing architecture (see online version for colours)

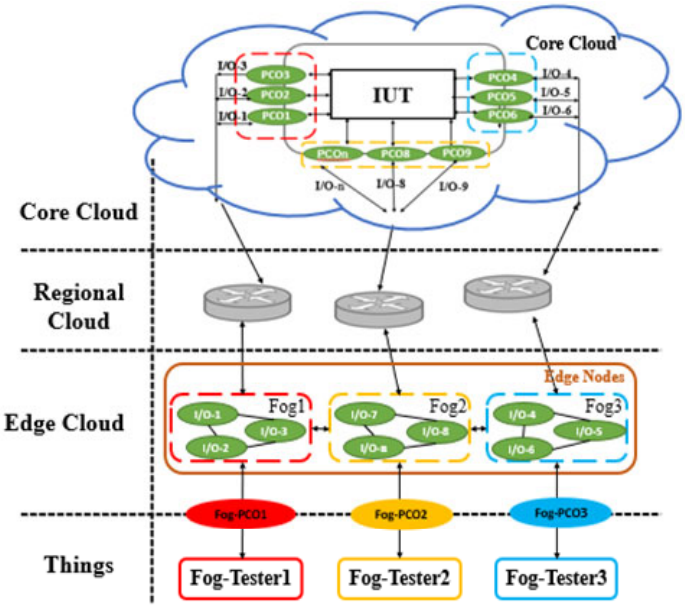


Figure 4 gives an abstract view of distributed cloud testing architecture where each tester (fog-tester) interacts with the IUT using two kinds of ports:

- Fog-PCO: ensures communication between a fog-tester and the edge cloud.
- PCO: ensures communication between the edge cloud and the IUT. Therefore, the I/O data exchanged through PCO(s) that are close geographically are grouped by regions in the regional cloud.

5.2 Modelling by automaton

To approach the testing process in a formal way, the specification and the IUT must be modelled using the same concepts. The specification of the behaviour of a distributed application is described by an automaton with n-port (FSM: finite state machine) (Gill, 1962) defining inputs and the results expected at each PCO.

Definition 1: A multi-port FSM with n ports (np-FSM) is defined by $A = (Q, q_0, Act, T)$ with:

- Q is a finite set of states
- $q_0 \in Q$ is the initial state
- $Act = \Sigma \cup \Gamma$, is a countable set of actions with:
  - a $\Sigma = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$ where Σ_i is a finite set of inputs of ports i, $\Sigma_i \cap \Sigma_j = \emptyset$ for $i \neq j$ and i, j = 1, 2, …, n and $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_n$

b $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\}$ where Γ_i is a finite set of outputs of ports i, $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$ and i, j = 1, 2, …, n and $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_n$.

- T is a finite set of transitions and $T \subseteq Q \times Act \times Q$.

Definition 2: The transition T is a tuple (q_1, σ, q_2) where:

- $q_1, q_2 \in Q$ are the source and destination states
- $\sigma \in Act$, is a sending of an input x (figured as $!x_i$) or a reception of an output y (figured as $?y_j$).

A transition T (q1, σ, q2) means that is possible to move from state q_1 to state q_2 with action $\sigma \in Act$.

Example 1: Figure 5 gives an example of a 3p-FSM with: $Q = \{q_0, q_1, q_2, q_3\}$, q_0 the initial state, $\Sigma_1 = \{a_1, a_2\}$, $\Sigma_2 = \{b_1\}$, $\Sigma_3 = \{c_1\}$, and $\Gamma_1 = \{x_1, x_2, x_3, x_4\}$, $\Gamma_2 = \{y_1, y_2, y_3, y_4\}$, $\Gamma_3 = \{z_1, z_2, z_3, z_4\}$.

ε indicates that no message is received at the corresponding port.

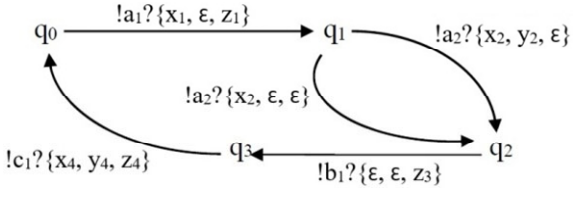
Definition 3: A global test sequence (GTS) of an np-FSM is a sequence in the form: $!x_1 ?y_1 !x_2 ?y_2 \ldots !x_t ?y_t$ where:

- For i = 1, …, t, x_i belongs to $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_n$ with $\Sigma_i \cap \Sigma_j = \emptyset$ and $!x_i$ denotes sending the message xi to IUT.
- For i = 1, …, t, $y_i \in \Gamma_k$ and for each port k, $|Y_i \cap \Gamma_k| \leq 1$ and $?y_i$: Denotes the reception of messages belonging to the Y_i from the IUT.

Example 2: An example of a GTS of 3p-FSM illustrated in Figure 5 is:

$$!a_1 ?\{x_1, \varepsilon, z_1\} !a_2 ?\{x_2, y_2, \varepsilon\} !b_1 ?\{\varepsilon, y_3, z_3\} !c_1 ?\{x_4, y_4, z_4\} \quad (1)$$

Figure 5 Example of 3p-FSM



5.3 Common distributed test problems

As explained in distributed cloud testing architecture (Figure 4), all Fog-testers work in parallel and independently. The projections of (1) on ports alphabets are required to get the test sequence related to each fog-tester. In this case, many problems influencing fault detection during the conformance testing process arise if there is no coordination between distributed test components (Rafiq and Cacciari, 2003):

- Observability problems: The observability may be defined as the capability of the test system to determine the output events and the order in which they take place at

corresponding edge. It arises when the tester at port (i) cannot receive an output from IUT responding to a specific input and cannot decide when to start and stop waiting.

- Synchronisation problems: It may be defined as the capability of the test system to realise input events at corresponding fog in a given order. It arises when the test system cannot guarantee that IUT will receive an input of port p(i) before an input of port p(i+1).

To resolve these problems, we propose in Azzouzi et al. (2020) an algorithm to generate local test sequences (LTS) from the GTS. As shown in the obtained LTS, some coordination messages are added to the projections to avoid previous problems when using the GTS.

$$\begin{cases} W_1 = !a_1 ?x_1 ?S^3 !O^2 !a_2 ?x_2 !S^2 ?O^3 ?x_4 \\ W_2 = ?O^1 ?y_2 ?S^1 !O^3 !b_1 ?y_3 !S^3 ?y_4 \\ W_3 = ?z_1 !S^1 ?O^2 ?z_3 ?S^2 !O^1 !c_1 ?z_4 \end{cases} \quad (2)$$

We notice two kinds of coordination messages:

- $!S^{t1,tr}$ ($!O^{t1,tr}$ resp.) the sending of a synchronisation message (observation message resp.) to the testers t_1, …, t_r.
- $?S^t$ ($?O^t$ resp.) the receipt of a synchronisation message (observation message resp.) from the tester t.

As the generated LTS sequences encapsulate the information needed to control the test execution, we can use these local sequences to detect faults in distributed cloud testing. Therefore, the FOG-testers can observe inputs and outputs locally at each FOG-PCO port. Furthermore, the fog-testers will be able to guarantee the correct execution of the global test by exchanging synchronisation and observability messages introduced in the LTS sequences.

The emphasis of recent works is to minimise the use of external coordination message exchanges among testers or to identify conditions on a given FSM under which controllability and observability problems can be overcome without using external coordination messages. Therefore, we suggest in this paper a fog computing-based architecture to optimise the number of exchanged messages between various components of the distributed test platform. For this purpose, the testers do not exchange messages directly with each other but communicate via their fogs.

As the PCO's of the IUT are the sources of the controllability problems at the IUT level, each tester will execute its LTS as follows:

- If these PCO's ports belong to the same region then the problem is solved at the fog level. In fact, the I/O data of all these fog-testers belong to the same fog and when a fog-tester needs to apply a new entry then all the necessary information for this sending will be available in the same fog.
- Otherwise, if the ports do not belong to the same region then their I/O data will be located in different fogs. In this case: To apply an entry without encountering controllability problems, each fog-tester need to obtain necessary information on other fogs using the communication channels installed by default in the distributed cloud.

In this case, the test process is carried out as follows:

- For each message 'xi' sent to the IUT or a coordination message, the fog-tester supports the process of sending this message.
- If 'xi' is an expected message from the IUT or a coordination message, the fog-tester waits for this message. If no message is received, or if the received message is not expected, the fog-tester returns a verdict fail.
- If the fog-tester reaches the end of its local test sequence, then it gives a verdict accept (accepted).
- If all fog-testers return a verdict accept, then the test system ends the test with a global verdict accept.

6 Distributed cloud testing: case study

6.1 Overview

With the advent of IoT, the concept of smart cities has become popular (Santos et al., 2018). The aim of smart cities is to improve the quality of citizens' life by making the city more adaptive and efficient. The scope covering this new mode of city management includes in particular: e-services, e-administrations, public infrastructure and transport (Aymen and Mahmoudi, 2019). Moreover, The future of smart cities lies also in modelling cities in 3D using virtual reality (VR) technology in many fields (Lv et al., 2016).

6.2 VR-assist rental service

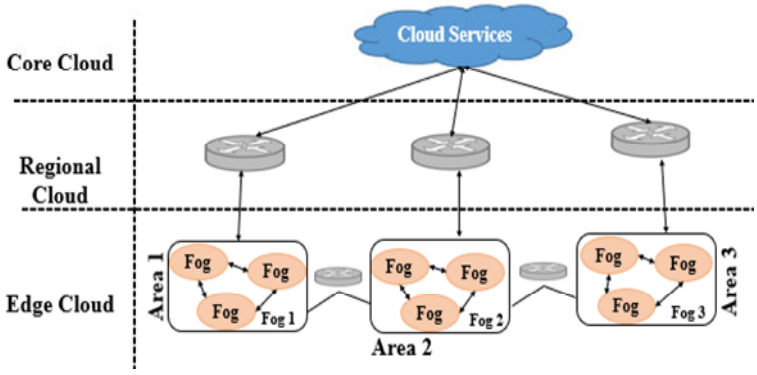
We present in this section a VR technology example: A VR-assist rental service.

Figure 6 VR-assist rental description (see online version for colours)

VR-assist rental service is an application for finding an apartment/house to rent. The service allows viewing a range of properties without a need to visit the actual house (Nguyen et al., 2018). As shown in Figure 6, the apartments/houses are therefore grouped into areas containing apartments/houses located in the same geographic area to simplify the search for users.

While the service is deployed in a centralised cloud, all the databases are then stored in the core cloud. By the way, this can cause the network congestion if many users try to access the service at the same time. To face this problem, it could be efficient if the service is deployed on a distributed cloud by distributing geography-based data to edge cloud. In this case, each Fog will handle a part of the database according to the houses located in the same area as shown in Figure 7 (Nguyen et al., 2018).

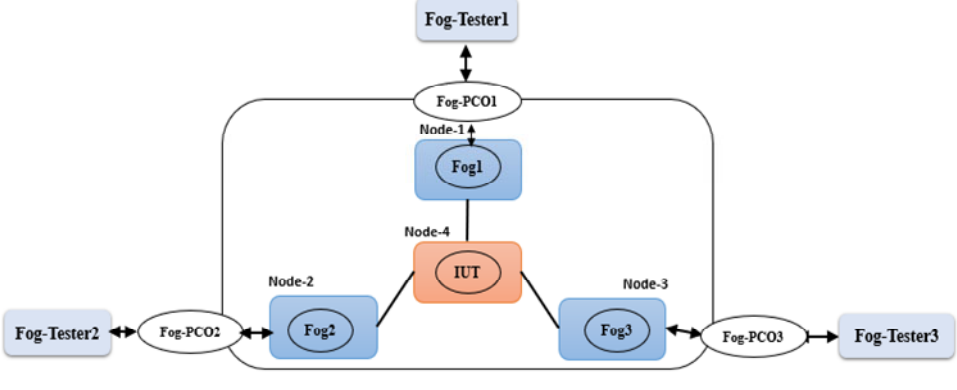
Figure 7 VR-assist rental service implementation (see online version for colours)



6.3 VR-assist rental testing architecture

The application is deployed in a distributed cloud and the experiment was executed on a cluster of four nodes. One node handles the core cloud, and the three other nodes handle the different Fogs of the distributed cloud.

Figure 8 VR-assist rental service testing architecture (see online version for colours)



As shown in Figure 8, The IUT is the implementation of the distributed cloud application (VR-assist rental service) and each Fog-Tester$_i$ observes the behaviour of a Node-i and interacts with IUT through a port of control Fog-PCO$_i$.

6.4 Test execution

To illustrate the test execution of the IUT, we refer to the GTS [equation (1) given in the previous Section 5] in order to describe a simple scenario of sending inputs to the IUT (VR-assist rental service) and the expected outputs as follows:

Figure 9 VR-assist rental service GTS

$!a_1\{x_1, \varepsilon, z_1\}$

$!a_1$	Ask house in area$_3$
$?x_1$	Data not found
$?z_1$	Sends matched data

$!a_2?\{x_2, y_2, \varepsilon\}$

$!a_2$	Ask house in area$_2$
$?x_2$	Data not found
$?y_2$	Sends matched data

$!b_1\{\varepsilon, \varepsilon, z_3\}$

$!b_1$	Ask house in area$_3$
$?z_3$	Data not found

$!c_1?\{x_4, y_4, z_4\}$

$!c_1$	Ask house in area$_3$
$?x_4$	Data not found
$?y_4$	Data not found
$?z_4$	Sends matched data

Therefore, we will get the following GTS as given in (3):

$$\begin{aligned}
&!\text{Ask house in area3}?\{\text{Data not found}, \varepsilon, \text{Sends matched data}\}\\
&!\text{Ask house inarea2}?\{\text{Data not found}, \text{Sends matched data}, \varepsilon\}\\
&!\text{Ask house in area3}?\{\varepsilon, \text{Data not found}, \text{Sends matched data}\}\\
&!\text{Ask house in area3}?\{\text{Data not found}, \text{Data not found}, \text{Sends matched data}\}
\end{aligned} \tag{3}$$

The projections of (3) on different ports are required to get the test sequence related to each fog-tester. The projections W_{p1}, W_{p2}, W_{p3} of the GTS are:

$$\begin{cases}
W_{p1} = !\text{Askhouse in area3}?\ \text{Data not found}!\ \text{Ask house in area2}?\\
\qquad \text{Data not found}?\ \text{Da tan ot found}\\
W_{p2} = ?\text{Sends matched data}!\ \text{Ask house in area3}?\ \text{Data not found}\\
W_{p3} = ?\text{Sends matched data}?\ \text{Sends matched data}!\text{Ask house in area3}?\\
\qquad \text{Sends matched data}
\end{cases} \tag{4}$$

If each Fog-Tester$_i$ executes the projection W_{pi} related to its corresponding port Fog-PCO$_i$, then some observation and synchronisation problems occur. As explained previously in Section 5.3, we face for example a synchronisation problem while the 'Fog-Tester1'sends the message 'a2:Ask house in area2' to IUT before the third tester 'Fog-Tester3' receives 'z1: Sends matched data' from the IUT (Figure 10).

Figure 10 A faulty test execution

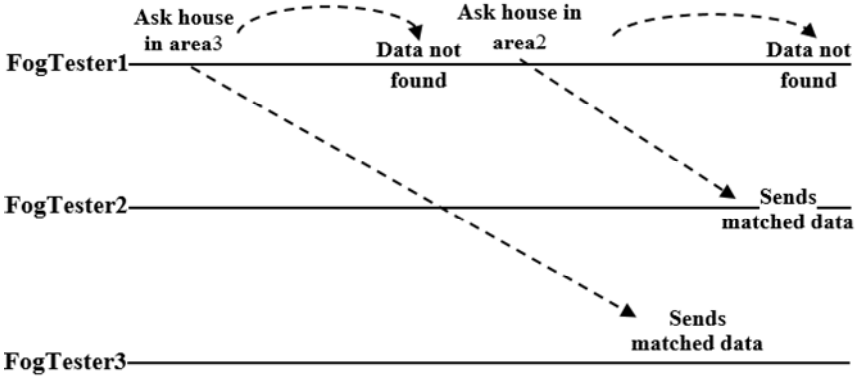
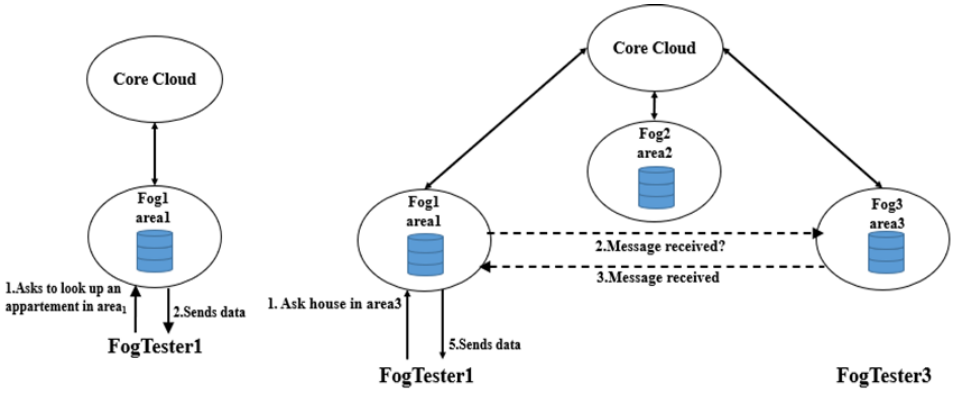


Figure 11 Optimisation of messages exchanged between fog-testers (see online version for colours)



(a) (b)

In this scenario, the execution of the projections by the fog-testers is not conform with the specification given in (3), where the message 'a2:Ask house in area2' should be sent only if all messages due to the sending of 'a1: Ask house in area3' by fog-tester1 are received from the IUT.

To resolve these coordination problems, we introduce as presented above some synchronisation and observation messages into the LTS by applying the algorithms given in Azzouzi et al. (2020) and Hsaini et al. (2018). So, the LTS become as follows:

$$\begin{cases} W_1 = !\text{Askhouse in area3?}\ \text{Data not found}?S3!O2!\text{Ask house in area2?} \\ \qquad \text{Data not found}!S2?O3?\text{Data not found} \\ W_2 = ?O1?\text{Sends matched data}?S1!O3!\text{Ask house in area3?} \\ \qquad \text{Data not found}!S3?\text{Data not found} \\ W_3 = ?\text{Sends matched data}!S1?O2?\text{Sends matched data}?S2!O1! \\ \qquad \text{Ask house in area3?}\ \text{Sends matched data} \end{cases} \tag{5}$$

As a major contribution, we suggest in this paper to use the fog computing concepts in order to minimise the exchange of coordination messages. Hence, the fog-testers will exchange S and O messages only if they belong to different areas (Figure 11). In other words, there will be two scenarios:

- If the flats belong to the same area then the fog-tester will find the required information in the same fog area [Figure 11(a)].
- If the flats do not belong to the same area [Figure 11(b)], then the corresponding fog will use the distributed cloud communication channel to retrieve the required information from the other fog-testers concerned by the controllability message.

Finally, by executing many tests according the new LTS, we get favourable results which prove how conformance can be assessed and that problems influencing fault detection during the testing process can be overcome by using the fog computing paradigm.

7 Discussion and concluding remarks

Cloud computing has benefited from advances in communication technologies. However, testing such platforms requires a major effort to improve the latency especially if a large number of users request simultaneously cloud services at the same location. In this paper, we suggest to optimise the test architecture of such platforms by using fog paradigm. Therefore, we suggest (as shown in Figure 4) to group all the PCOs that carry the same inputs/outputs (I/O) in a single fog in order to reduce the bandwidth consumption and improve network performance. These are some concluding remarks:

- *Minimising communication channels*: Based on the principles of fog computing, all testers will have direct communication with the fog they are connected to. If a tester needs to communicate with another tester, the communication is then performed through the channels of the fog computing already installed. Therefore, we no longer need the multicast channel for exchanging coordination messages between testers.
- *Optimisation of the number of coordination messages*: In fact, we suggest in this paper to overcome coordination issues between testers but a potential challenge relates to the massive number of messages exchanged to perform coordination between testers. The idea as presented in our architecture (Figure 4) suggests minimising the number of testers which will reduce considerably the messages exchanged between testers.
- *Data transfer cost reduction*: The cost of data transfer between the endpoints (things) and the cloud can be significant. The idea is to group data (fog's data) according to

the geographical situation. For this purpose, we suggest handling data closer to the endpoints. Once the fogs have been constructed, we assign to each fog a specific tester (fog-tester) in order to observe the I/O exchanges and by the way to ensure coordination of multiple endpoint services in a consistent manner.

8 Conclusions

As a conclusion, current testing researches in distributed clouds remain insufficient despite the developments made to overcome testing issues in these environments. Thus, this work can be used to support research in this area in order to explore different types of failures that can occur within distributed cloud infrastructures.

In this paper, we propose a new architecture for conformance testing of a distributed implementation in the cloud (IUT). Then, we review some issues that occurred while testing in these distributed environments. Afterwards, we show how we can deal with these problems by exchanging coordination messages between testers and the IUT. Finally, we give an explanation of our model using an advanced test case.

As prospects, we intend to use heuristics and combinatorial optimisation to choose PCOs to be grouped in a fog. We plan also to extend our model to cover performance as well as security testing in distributed clouds.

References

Aymen, F. and Mahmoudi, C. (2019) 'A novel energy optimization approach for electrical vehicles' in a smart city', *Energies*, Vol. 12, No. 5, p.929.

Azzouzi, S., Benattou, M. and Charaf, M.E.H. (2015) 'A temporal agent based approach for testing open distributed systems', *Computer Standards & Interfaces*, Vol. 40, No. 40, pp.23–33.

Azzouzi, S., Hsaini, S. and Charaf, M.E.H. (2020) 'A synchronized test control execution model of distributed systems', *International Journal of Grid and High Performance Computing (IJGHPC)*, Vol. 12, No. 1, pp.1–17.

Charaf, M.E.H. and Azzouzi, S. (2017) 'A colored Petri-net model for control execution of distributed systems', *The 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, pp.0277–0282.

Charaf, M.E.H., Benattou, M. and Azzouzi, S. (2013) 'An expert system based architecture for testing distributed systems', *International Journal of Innovative Computing, Information & Control: IJICIC*, Vol. 9, No. 9, pp.3779–3799.

Charaf, M.E.H., Benattou, M. and Azzouzi, S. (2014) 'A JESS AGENT based architecture for testing distributed systems', *J. Inf. Sci. Eng.*, Vol. 30, No. 5, pp.1619–1634.

Chiang, M., Ha, S., Chih-Lin, I., Risso, F. and Zhang, T. (2017) 'Clarifying fog computing and networking: 10 questions and answers', *IEEE Communications Magazine*, Vol. 55, No. 4, pp.18–20.

Ciortea, L., Zamfir, C., Bucur, S., Chipounov, V. and Candea, G. (2010) 'Cloud9: a software testing service', *ACM SIGOPS Operating Systems Review*, Vol. 43, No. 4, pp.5–10.

Esfandyari, M.J., Esfahanian, M.M., Yazdi, M.R.H., Esfahanian, V., Nehzati, H. and Salehi, A. (2017) 'Hardware-in-the-loop simulation for verification of the drive motor management software in a series hybrid electric bus', *International Journal of Powertrains (IJPT)*, Vol. 6, No. 2, pp.184–199.

Felemban, M., Basalamah, S. and Ghafoor, A.(2013) 'A distributed cloud architecture for mobile multimedia services', *IEEE Network*, Vol. 27, No. 5, pp.20–27.

Francis, T. (2018) 'A comparison of cloud execution mechanisms fog, edge, and clone cloud computing', *International Journal of Electrical & Computer Engineering*, Vol. 8, No. 6, pp.4646–4653.

Gill, A. (1962) *Introduction to the Theory of Finite-State Machines*, McGraw-Hill Education, New York.

Hsaini, S., Azzouzi, S. and Charaf, M.E.H. (2018) 'Testing rules for Mapreduce fameworks', *IEEE 5th International Congress on Information Science and Technology (CiSt)*, IEEE, October, pp.94–98.

King, T.M. and Ganti, A.S. (2010) 'Migrating autonomic self-testing to the cloud', *The Third International Conference on Software Testing, Verification, and Validation Workshops*, IEEE, pp.438–443.

Lv, Z., Yin, T., Zhang, X., Song, H. and Chen, G. (2016) 'Virtual reality smart city based on WebVRGIS', *IEEE Internet of Things Journal*, Vol. 3, No. 6, pp.1015–1024.

Mahmoudi, C., Flah, A. and Sbita, L. (2019) 'Smart database concept for power management in an electrical vehicle', in *International Journal of Power Electronics and Drive System (IJPEDS)*, March, Vol. 10, No. 1, pp.160–169.

Mora, H., Mora Gimeno, F.J., Signes-Pont, M.T. and Volckaert, B. (2019) 'Multilayer architecture model for mobile cloud computing paradigm', *Complexity*, Vol. 2019, pp.3951495–3951508.

Moutai, F.Z., Hsaini, S., Azzouzi, S. and Charaf, M.E.H. (2019a) 'Security testing approach for IaaS Infrastructure', *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, pp.1–5.

Moutai, F.Z., Hsaini, S., Azzouzi, S. and Charaf, M.E.H. (2019b) 'Testing distributed cloud: a case study', *International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, IEEE, p.1–5.

Moutai, F.Z., Tajioue, M.A., Oualla, A. and Azzouzi, S. (2020) 'Towards distributed systems testing in cloud environment', *IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, IEEE, pp.1–6.

Moysiadis, V., Sarigiannidis, P. and Moscholios, I. (2018) 'Towards distributed data management in fog computing', *Wireless Communications and Mobile Computing*, Vol. 2018, pp.7597686–7597700.

Nguyen, T.D., Kim, Y., Pham, X.Q., Nguyen, T.D. and Huh, E.N. (2018) 'Mobile services meet distributed cloud: benefits, applications, and challenges', *Mobile Computing: Technology and Application*, p.39, IntechOpen, Rijeka.

OpenFog Consortium (2017) *OpenFog Reference Architecture for fog Computing* [online] https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf (accessed 5 January 2020).

Oriol, M. and Ullah, F. (2010) 'Yeti on the cloud', *Third International Conference on Software Testing, Verification, and Validation Workshops*, IEEE, pp.434–437.

Rafiq, O. and Cacciari, L. (2003) 'Coordination algorithm for distributed testing', *The Journal of Supercomputing*, Vol. 24, No. 2, pp.203–211.

Santos, J., Leroux, P.,Wauters, T.,Volckaert, B. and De Turck, F. (2018) 'Anomaly detection for smart city applications over 5g low power wide area networks', in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, pp.1–9.

Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016) 'Edge computing: vision and challenges', *IEEE Internet of Things Journal*, Vol. 3, No. 5, pp.637–646.

Sudharson, D. and Prabha, D. (2020) 'Improved EM algorithm in software reliability growth models', in *International Journal of Powertrains (IJPT)*, Vol. 9, No. 3, pp.186–199.

Tómasson, H. (2011) *Distributed Testing of Cloud Applications Using the Jata Test Framework*, PhD thesis, Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, Iceland.