Scalability of an Internet of Things Platform for Smart Water Management for Agriculture

Ivan Zyrianoff, Alexandre Heideker, Dener Silva, Carlos Kamienski
Federal University of the ABC
Santo André, Brazil
{ivan.zyrianoff,alexandre.heideker,dener.silva,cak}@ufabc.edu.br

Abstract–**The emergence of a new breed of smart applications requires middleware platforms that enable the rapid development of IoT-based solutions, which can be hosted partially in fog nodes, as well as in a traditional cloud datacenter. Currently, there is no scalable de facto open IoT platform but the European Commission is pushing FIWARE to fill this gap. We analyzed the performance of FIWARE under different platform configurations comparing fog/cloud and cloud-only scenarios for precision irrigation in smart farming. Our results reveal interesting and non-intuitive findings, such as that fog computing does not always improve the overall system performance and in some cases it even makes it worse. Also, the network between the farm and the cloud datacenter causes some unexpected differences between different scenarios.**

I. Introduction

The Internet of Things (IoT) [15] and the billions of sensors that will be deployed in the next decade [5], as well as novel associated technological breakthroughs, have been enabling the emergence of a new breed of smart applications and services for the benefic or our society in different domains or verticals, such as smart farming, smart cities, smart healthcare and smart industries. Building interoperable IoT services and applications requires a set of middleware components and system development, deployment and operation tools and platforms, In order to avoid developing extremely focused and vertical IoT applications not able to interact with other applications, common and generic middleware services used by different application domains become necessary.

The widespread availability of IoT-based applications requires adequate platforms for both development and operation phases. The former for releasing developers from the need of mastering different technologies outside their core business and that do not add value to the process [7]. The latter for freeing organizations from the need of deploying and testing customized platforms for supporting the operation of IoT-based applications. There are some IoT platforms available today, both open source and proprietary [4]. There is a tradeoff in the deployment of such platforms, which traditionally are hosted in cloud datacenters, but are slowly considered to be partially moved to edge infrastructure closer to the users, which is known as fog computing [2].

Although many IoT platforms exist today, the arena is not clear regarding the suitability of them for the different deployment scenarios for different smart applications. IoT tools and platforms must provide an end-to-end treatment for the data path, since data is generated by sensors, transmitted to the storage place, processed by smart algorithms, decisions are made, which fire actions that are finally forwarded to actuators as commands aimed at changing some configuration.

Also, scalability is a major concern for IoT platforms. It has been shown that different architectural choices of IoT platforms affect system scalability and that automatic real time decision-making is feasible in an environment composed of dozens of thousands of sensors continuously transmitting data [16]. Currently, there is no scalable de facto open IoT platform but the European Commission is pushing FIWARE to fill this gap.

The SWAMP project develops and assesses an IoT-based smart water management platform for precision irrigation in agriculture with a hands-on approach that focuses on pilots in Italy, Spain and Brazil [9]. The same underlying platform can be customized to different pilots considering different countries, climate, soil, and crops. The SWAMP platform may be implemented in a range of deployment configurations involving both cloud and fog environments.

In this paper, we analyze the performance of FIWARE under different platform configurations comparing fog/cloud and cloud-only scenarios for precision irrigation using one of the SWAMP pilots as the evaluation scenario. Experiments consisted simulated sensors sending messages to FIWARE, deployed both in a fog/cloud and a cloud-only configuration mode. We performed experiments with a large number of probes simultaneously sending messages to the platform, in order to verify and understand it scalability.

Security and privacy aspects are key components for making it possible to emergence of a true market for such IoT platforms for precision irrigation, such as SWAMP [8]. However, these aspects have not been evaluated here since they are outside the scope of this paper that focuses on performance and scalability of the FIWARE Platform when configured for the needs of SWAMP.

Our results reveal interesting and non-intuitive outcomes, such as, that fog computing does not always improve the overall system performance. In some cases, the addition of a fog processing nodes even proved to worsen the performance. An important and unexpected factor in the experiments was the impact of the network. Initially, we believed that the network would equally impact both configurations (fog/cloud and cloud-only). However, the impact was much higher in the fog/cloud configuration.

In the remainder of the paper, section II presents background and related work. Section III introduces SWAMP and the two scenarios for the MATOPIBA pilot. Section IV provides a detailed view of research design and methods. The key results are presented in section V, followed by discussion of lessons learned in section VI. Finally, section VII draws some conclusions and propose relevant future work.

II. Background and Related Work

This section introduces related background and related work.

A. *IoT and Fog Computing*

Fog computing is a fairly new paradigm aimed at dealing with challenges related to the huge amount of data that inevitably will be generated with the increasing utilization of IoT-based systems [2]. It solves serious problems such as decreasing latency from real time applications, decreasing data traffic between the network edge and core, and softening the processing burden of the cloud by performing load balancing. The metaphor comes from the fact that fog is a cloud but closer to the ground and to the people [13]. The fog is based on a highly virtualized platform that provides computing, storage and communication services between users and the datacenter where the cloud is hosted [1], by bringing services from the cloud closer to the users.

Since its inception the fog has been envisioned for operating together with the cloud for making it possible to implement IoT-based smart applications. In other words, fog and IoT are related yet independent concepts, as the fog can deal with a broader variety of applications and IoT does need the fog to come true.

A new technological trend that has been used for implementing the fog is container-based virtualization that provides a lightweight alternative to the traditional hypervisors [12]. Containers do not emulate the underlying hardware. Rather, the virtualized OS communicates directly with the host OS, which makes the appropriate calls to the real hardware. Lightweight virtualization minimizes the use of computational resources of the host machine, since there is no need to duplicate the operating system.

B. *The FIWARE Platform*

The FIWARE platform (fiware.org) has been attraction general attention for being a worldwide open source solution fostered and funded by the European Commission under Horizon 2020 program. It is comprised of a series of software components called Generic Enablers (GE) that perform functions needed in a different variety of IoT-based applications for smart societies, focusing in cities, farming, industry, healthcare and sustainability. GEs can be used to build different applications that exchange information through a REST API following the OMA NGSI standard (openmobilealliance.org/release/NGSI) , based on JSON. The central aspect of the FIWARE NGSI Context Management information model is the concept of entities and their attributes.

Among the many GEs available by the FIWARE, some are considered the key enablers for smart applications. Here we only introduce the ones cited and used in our paper.

- Orion Context Broker: Orion is a publish/subscribe context broker, which makes it the main FIWARE GE and the heart of the platform. Orion provides an interface where clients can register entities and their attributes as well as producers/consumers of those entities. Orion only stores the latest version of entity attributes and it needs to work together with other GEs or applications in order to maintain historical data. Orion is available in fiware-orion.readthedocs.io.
- IDAS and IoT Agents: IDAS is an implementation of the Device Backend Management GE that comprises many IoT Agents that map data coming from sensors and going to actuators to the FIWARE NGSI information model to be stored in Orion and further processed by other GEs or external applications. Since low power sensors do not possess computing, storage and transmission capabilities for speaking NGSI, they use different IoT protocols such as MQTT, CoAP, Ultralight or LoRaWAN transport, which are converted into/from NGSI by IoT Agents. IDAS is available in catalogue-server.fiware.org/enablers/backend-device-management-idas.
- STH Comet: Short Time Historic, also known as Comet, works with Orion storing entity data as a time series that can be further used by applications or other GEs. STH Comet is available in fiware-sth-comet.readthedocs.io.

The use of FIWARE by any organization or developer involves the installation of its GEs in an appropriate infrastructure where it can run, which might be standalone machines, public clouds or preferentially a private cloud using the OpenStack cloud manager.

FIWARE has been used as a computing platform for many IoT-based applications, such as farming and environment [14]. Also, many large scale projects use FIWARE for IoT in agriculture, such as IoF2020 (iof2020.eu), as well as startups such as Breeze (breeze-technologies.de), Hop Ubiquitous (hopu.eu) and Agricolus (agricolus.com).

C. *Related Work*

Scalability is a major concern for IoT platforms. Numbers vary but in general it is forecasted that in the beginning of the 2020's there will be about 30 billion connected devices [5]. It has been shown that different architectural choices of IoT platforms affect system scalability and that automatic real time decision-making is feasible in an environment composed of dozens of thousands of sensors continuously transmitting data [16].

The performance of FIWARE has been calling the attention its user community. A comprehensive study that proposes qualitative and quantitative metrics and evaluates the performance of various IoT platforms is presented by da Cruz et al. (2018) [4]. From 11 platforms analyzed by the qualitative approach, 5 were selected for the quantitative performance analysis, including FIWARE. However, since they adopted a generic approach, they did not go into the specifics of FIWARE -only a single Orion + STH platform configuration was considered – and they did not evaluate different infrastructures (as fog computing).

Martínez et al. (2016) [11] gives a detailed description of the architecture of a testbed of the FIWARE platform configured for the precision agriculture domain, which differs from our approach, because their test application connects directly to FIWARE using NGSI JSON interface, while we included an IoT Agent for MQTT using a scalable IoT sensor simulator for generating synthetic data. Lastly, Cardoso et al. (2017) [3] compared the performance of FIWARE and their own implementation of ETSI M2M, under different running conditions, which makes it difficult to understand the tradeoffs of each platform. In this paper, we differ from previous work by evaluating five configurations of FIWARE and focusing on the scenarios of smart irrigation in agriculture from the SWAMP project [9].

III. Smart Water Management Platform

The primary objective of the SWAMP project is to develop IoT based methods and approaches for smart water management in precision irrigation domain and to pilot them in four places, two pilots in Europe (Italy and Spain) and two pilots in Brazil [9], more information can be found in swamp-project.org. Also, it is aimed at improving precision irrigation by increasing the awareness of the condition of the crop, by monitoring the field based on crop status and environment, and to adjust the irrigation prescription map accordingly.

A. SWAMP Pilots

The four SWAMP pilots are based on the similar technical solutions and deal with different crops and have different primary goals.

1) CBEC Pilot (Bologna/Italy): the main objective of the Consorzio di Bonifica Emilia Centrale (CBEC) pilot is optimizing water distribution to the farms.
2) Intercrop Pilot (Cartagena/Spain): Intercrop Iberica addresses several challenges since production is in a dry area and its primary goal is making a rational use of water.
3) Guaspari Pilot (Espírito Santo do Pinhal / Brazil): The Guaspari Winery transfers the wine grape harvesting to the winter season (June-August) using irrigation techniques. The main goal for Guaspari is improving wine quality.
4) MATOPIBA Pilot (Barreiras/Brazil): The Rio das Pedras Farm is located in the MATOPIBA region, and irrigation is mostly performed by center pivots. The main pilot goal is to implement and evaluate a smart irrigation system based on Variable Rate Irrigation (VRI) for center pivots in soybean production and save energy used in irrigation.

The SWAMP project is developing a high-precision smart irrigation system concept for agriculture. The fundamental idea is to enable optimizations of irrigation, water distribution, and consumption based on a holistic analysis that collects information from all aspects of the system including even the natural water cycle and the cumulated knowledge related to growing particular plants. It results in savings to all parties as it guarantees the availability of water in situations where water supply is limited and also prevents over- and under-irrigation.

A fundamental idea in SWAMP is to facilitate the replication of water management systems built on top of its platform with minimum redesign and redevelopment. Different layers of the architecture have components that are more generic and thus less difficult to be ported to other pilots, whereas others are more application-specific and thus require new development efforts whenever a new pilot is conceived.

B. SWAMP MATOPIBA Scenario

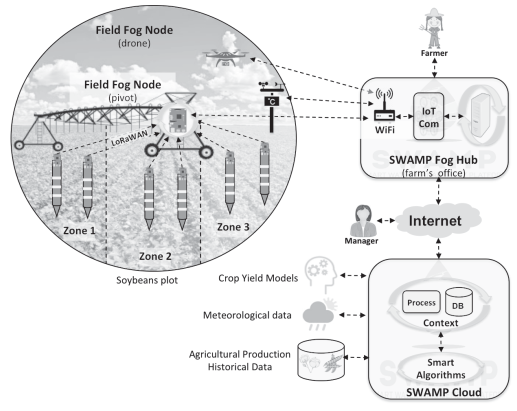
In this paper we will evaluate the performance of FIWARE when configured for the SWAMP MATOPIBA scenario depicted in **Fig. 1**, which represents the center-pivot with variable-rate irrigation pilot. The scenario captures both the farm and SWAMP Platform viewpoints and represents a future vision and not the current situation. A center pivot irrigates a circular agricultural plot of 100 hectares that alternates soybeans and cotton. The plot is further divided into different management zones, identified before the crop season and based on differences in the soil properties.

Two general types of sensors collect data for the SWAMP system: a) stationary sensors within the soil at the root system depth that continuously measure metrics such as temperature and moisture, and; b) drones acting as flying sensors equipped with thermal/multispectral cameras or working as data mules by collecting sensor data and transmitting to the farm's office. The center pivot is electrified to make it able to control the variable rate irrigation sprinklers (actuators) and thus it can also be equipped with processing and communication capabilities. No special energy harvesting is predicted for the sensing devices. We developed multiparametric probes for soil sensing, which include moisture, temperature and electrical conductivity sensors at three depths from the soil surface. Sampling and transmission rates will be adjusted from a few minutes up to hours, according to the time of the day and the application requirements. This way, the probe sensing electronics combined with ZigBee or LoRaWAN will be powered by extended lifespan batteries.

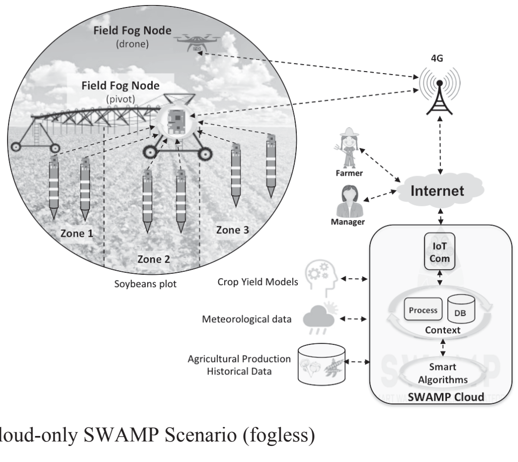
The key difference between Fig. 1(a) and Fig. 1(b) is that the former includes a distributed fog/cloud configuration of the platform whereas the latter is solely based on the cloud for hosting all software components. Depending on different variables - such as farmers interest, availability of infrastructure, intended responsiveness to delay, robustness to disconnections, and cost/effectiveness – one or the other scenario may be more or less appropriate. For the MATOPIBA pilot, the preferred solution is the fog-based one, for providing autonomous processing capacity to the farm as it is located in a region where Internet connections may suffer periods of instability. Anyway, we evaluate both approaches because the cloud-based one might be used as well.

In Fig. 1(a) both the center pivot and drones are deemed Field Fog Nodes (FFN) for the SWAMP architecture. Notably, the FFN at the center pivot acts as a sensor aggregation point and as a distributor of actuator commands received from the SWAMP system. FFNs communicate with the nearest fixed attachment point of the SWAMP platform, which usually is an on-premises Typically, when sensors are powered with LoRaWAN, the FFN will host the LoRaWAN Gateway and possibly the LoRaWAN Server and other supplementary features. The Fog Hub is located in the farm's office. SWAMP allows FFNs to communicate directly with the platform running in the cloud, whenever this is the preferred deployment choice. The FFNs send data directly to the SWAMP platform located in the Fog Hub at the farm's offices via different wireless technologies, which in Fig. 1(a) is WiFi. For this scenario, the functions performed by the SWAMP architecture are divided up into local fog and remote cloud components. Heavy processing, such as irrigation models and analytics using smart algorithms (i.e., machine learning), is performed in the cloud. External information is fed to the platform, such as crop yield models, meteorological data and historical data.

On the other hand, Fig. 1(b) depicts a scenario where no fog is available and data generated by sensors are sent directly to the cloud via a typical cellular technology that might be 3G/4G or even NB-IoT.



a) Fog-cloud SWAMP scenario



b) Cloud-only SWAMP Scenario (fogless)

Fig. 1. SWAMP MATOPIBA scenario; a) Fog-cloud; b) Cloud-only

IV. Research Design and Methods

A. Evaluation Scenario and Environment

One of the main goals of this paper is to compare fog/cloud and cloud-only based approaches for smart water management. Therefore, based on the SWAMP MATOPIBA pilot specification (Section 3), we designed a FIWARE-based IoT platform that involves obtaining sensor data values up to the point of they are transparently consumed by an application that can be deployed in the cloud and/or in the fog.

The software modules depicted in Fig.3 and Fig.4 are implemented by lightweight virtualization of Docker containers, including the FIWARE modules, whose container images can easily be obtained in that format. The other modules of this scenario are:

- SenSE: the Sensor Simulating Environment (SenSE), is an open-source large-scale IoT sensor data generator able to abstract real devices and to model different complex scenarios, such as smart farms [16]. The tool is a traffic workload generator that emulates a huge number of heterogeneous sensors representing tens of thousands of IoT sensors sending data simultaneously via a typical IoT protocol (e.g., MQTT). Although the sensors are synthetic, the traffic is real. In our scenarios, SenSE generates probe data and represents the fog field node that forward sensor data to the platform. The source code can be download from github.com/ivanzy/SenSE-Sensor-Simulation-Environment.
- Mosquitto: Eclipse Mosquitto is an open source MQTT message broker. Available in mosquitto.org.
- MongoDB: a document-oriented NoSQL database. Available in mongodb.com.
- Consumer: Consumer is a special purpose Express.js web application that subscribes in Orion and receives sensor data from the probes. When Orion sends a message to the consumer's API, a timestamp is recorded and is subtracted from the timestamp of the message, in such a way we obtain the elapsed time between the generation of the message until it reaches the Consumer.

The sequence of processing steps and data flow starting from the sensor data generation and ending at the consumer is shown in Fig. 2. This data flow is deployed in both fog and cloud-only scenarios, depicted in Fig.1.

1) SenSE generates sensor (probe) data and sends it to Mosquitto MQTT broker;
2) IoT Agent receives the probe messages from the MQTT broker, stores it in MongoDB, translates them to the NGSI format, and finally sends it to Orion through HTTP protocol;
3) Orion Context Broker receives NGSI data from the IoT Agent, updates entity values, stores them in MongoDB, and sends them to the subscribed applications through HTTP protocol and structured as NGSI;
4) Data is delivered to the consumer, which stores it and calculates the elapsed since the message was generated

in the probe (by SenSE), the consumer is subscribed to be notified from all probe sensors updates.

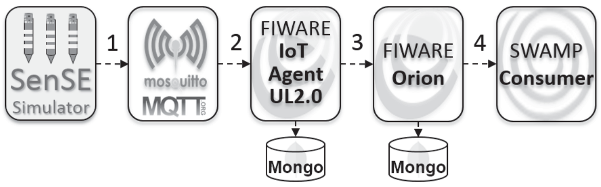


Fig. 2. Processing steps and data flow

In the cloud-only experiments (Fig.3) the sensor data is directed sent to the cloud, where all modules are in the same physical machine. Since both Orion and IoT Agent are in the same VM, both use the same MongoDB instance. In the fog experiments, the first two steps of processing occur in the fog node, which means that the MQTT broker and the IoT Agent run in the edge device, and the Orion and Consumer are executed in a VM in the cloud. This configuration needs a MongoDB instance in each machine, where the fog hosts the IoT Agent and cloud hosts Orion and the Consumer.

In order to fully understand the evaluation scenario, one needs to comprehend the data model of the probe messages. We adopted the Ultralight 2.0 (UL) protocol – a lightweight text based protocol aimed to constrained devices and communications where the bandwidth and device memory may be limited resources [6]. Each probe consists of a set of seven sensors of three different types:

- Three soil temperature sensors in different depths;
- Three soil moisture sensors in different depths;
- One soil electrical conductivity sensor.

Each probe sends a message every 10 minutes using the UL protocol over MQTT. The structure of each message is:

```
t1|v|t2|v|t3|v|m1|v|m2|v|m3|v|c|v|ts|v
```

where:

- t1, t2, t3: temperature sensors in different depths;
- m1, m2, m3: moisture sensors in different depths;
- c: electrical conductivity of the soil;
- ts: timestamp of message generation;
- v: value of the metric.

Fig.3 and Fig.4 show the two evaluation scenarios, cloud-only and fog/cloud, respectively. We performed the experiments in a lab testbed – all the VMs in the same LAN - and using a network emulator (WANem – available to download at wanem.sourceforge.net) to consider the impact of network parameters between the place where the data is generated (in the farm, where the sensors and fog nodes are located) and the place it is processed (the cloud, usually located in some datacenters placed in big cities). Although the case where the machines representing the fog and the cloud are in the same LAN does not adequately portray the reality, it is important to further understand the behavior of the IoT platform and what are the impacts of a constrained network.

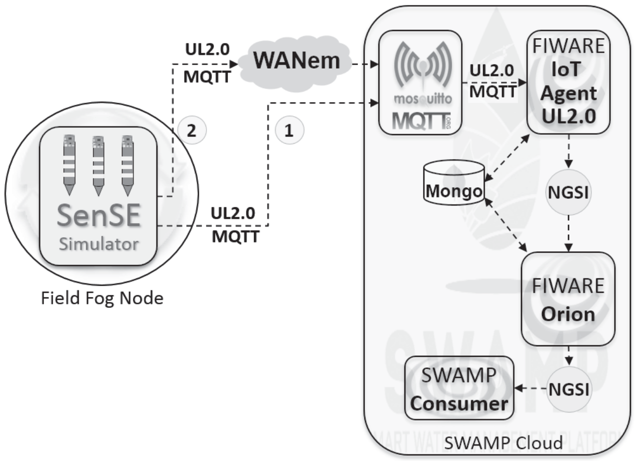


Fig. 3. Cloud/Only FIWARE-SWAMP Evaluation Scenarios – with and without WAN

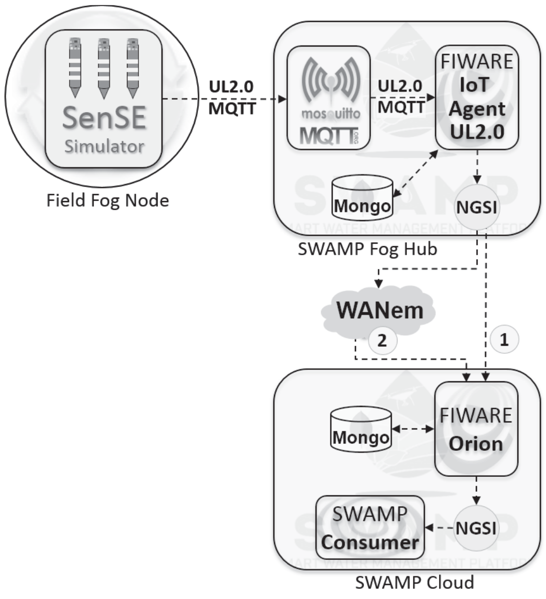


Fig. 4. Fog/Cloud FIWARE-SWAMP Evaluation Scenarios – with and without WAN

The experiments were performed in a private cloud environment implemented with OpenStack in a lab testbed. As the experiments were in our own private cloud, we could assure that they did not suffer from outside interference from other virtual machines running in the same physical servers. Both fog and cloud were implemented using virtual machines (VM) in OpenStack, though with very different configurations. We used the standard Amazon AWS VM configurations: cloud VM equivalent to a t2.medium instance (2vCPU - 4GB of RAM) and the fog VMs (both fog field node and fog hub) equivalent to a t2.small instance (1vCPU – 2 GB RAM). Our cloud was composed by 6 machines with the following configuration: Intel(R) Xeon(R) CPU E3-1240 V2 @ 3.40GHz - 8 cores and GB of RAM. Two different physical machines were used.

B. Metrics

There are two different metric used in our experiments:

- Average elapsed time: the average time elapsed since a sensor data point is generated in the probe until it reaches the consumer application. Basically, we are interested in the time elapsed since the data is collected until it is ready to be used by another application – such as a dashboard or an analytics module;
- System metrics: CPU and memory usage per Docker container, which allows us to observed each application, obtained by `docker stats` command every 5 seconds.

C. Experiments

Experiments consisted of SenSE simulating probes and sending messages to the IoT platform, running in the cloud-only configuration or in the fog/cloud configuration. We performed experiments with a large number of probes simultaneously sending messages to the platform, in order to verify and understand it scalability. Table I shows the factors and levels used in the performance analysis. Our experiments varied all factors with all levels, consisting of 16 different possibilities. Each experiment took 2 minutes and was replicated 30 times, totalizing 16 hours of running experiments. The asymptotic confidence intervals were calculated with a confidence level of 99%.

TABLE I. FACTORS AND LEVELS

Factor	**Level**
Number of Probes	1,000 – 5,000 – 10,000 – 15,000
Network Conditions	LAN – Emulated WAN
System Configuration	Fog - Cloud

To configure the emulated WAN, we tried to replicate a connection from a farm to a cloud. Therefore, we made a simple experiment in order to obtain the parameters used in the network emulator by pinging a public cloud using a 4G connection, and next we estimated the parameters as a 10 Mbps connection with 45ms of delay time and 5ms of jitter.

V. RESULTS

The results for the elapsed time in all configurations are depicted in Fig.5, wherein one can observe the effect of increasing the workload in the fog/cloud and cloud-only scenarios. The platform remained stable during almost all experiments, except for the ones performed with a workload of 15,000 probes - that we consider being a very high workload. The high confidence interval in those experiments reveals this instability.

Regarding the comparison between the experiments with the presence of the fog and the ones with only the cloud, it is essential to analyze networking issues. First we analyze those configurations without the network effects (no WANEm emulator), and next the impact of the WAN in the experiments. It is important to reinforce that experiments without considering the impact of the network are only for a baseline comparison and do not reflect real scenarios.

In the experiments using a local network, the two approaches had equivalent results, since neither the fog nor the cloud were overloaded. The difference between the two approaches –fog and cloud - is that in the fog case some software modules (Docker containers) were moved to another machine (representing the fog), there was no improvement in the overall performance of the system when the cloud was not overloaded. Therefore, our experiments revealed that performing load balancing between two different machines does not improve the performance of the system.

In the case where the cloud is overloaded (15,000 probes), distributing the processing to another machine (the fog) should have improved the overall performance of the system. However, the results show that there is an overlap between the confidence interval of the experiments with fog and cloud with 15000 probes, with the fog experiments having a slightly better performance. This occurs due to the limited computational resources of the machine that represents the edge device, so that when the workload increases the fog node becomes the bottleneck.

Experiments with the network emulator (WANem) had results that did not confirm our intuition. In the scenarios with a low workload (1,000 and 5,000 probes), the cloud approach has a significant better performance than the fog. This occurs because in the cloud experiments the network traffic was composed of small MQTT packets, which did not impose a major constrain to the network. However, in the fog experiments the MQTT traffic is only send to the fog, which sends a very verbose NGSI/HTTP traffic via the network to the cloud. This traffic suffers much more with the network constrains (latency and limited bandwidth).

Anyway, there is a clear trade-off between the cloud and fog approaches when processing becomes the bottleneck, instead of the network. The almost perfect balance occurs with experiments with 10,000 probes, in which both approaches achieved the same performance. For higher workloads, the fog has a slightly better performance than the cloud.

The system metrics – RAM and CPU usage - for the cloud-only experiments are depicted in Fig. 6 and Fig. 7, for the fog/cloud experiments are showed in Fig. 8 and Fig. 9. Since the workload in the system is very similar with and without the network, we opted to portrait only the results performed considering the network, since they illustrate the whole set of experiments. The experiments using the fog approach, there were two MongoDB instances - in the fog to help the IoT Agent and in the cloud connected to Orion - are identified as Mongo-fog and Mongo-cloud respectively.

Analyzing Fig. 6 and Fig. 8, it becomes clear that the container that demands more processing and it is the potential bottleneck of the system is MongoDB. Also, when comparing the cloud-only and fog/cloud results for the CPU usage, it is possible to infer that Orion imposes a higher workload in, since when the IoT Agent is on the fog node, MongoDB in the cloud has only an approximately 20% decrease of CPU usage.

Unlike the CPU usage, the memory usage has a stable behavior, even with the increase of the workload, as shown by Fig.7 for the cloud experiments, and in Fig. 9 for the fog experiments. However, there is also a decrease of approximately 20% of memory usage when the IoT-Agent is moved to the fog, with its own instance of MongoDB.

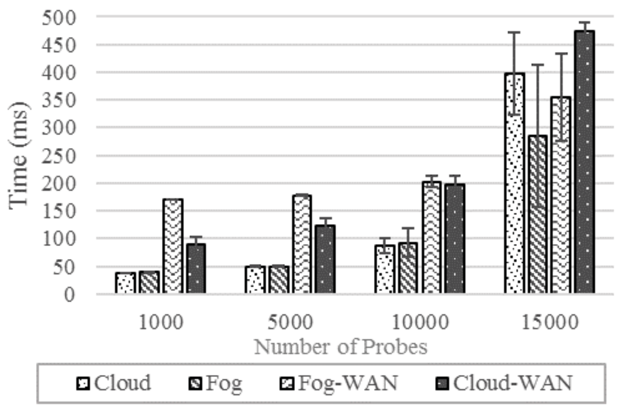


Fig. 5. Response time for the cloud and fog configurations

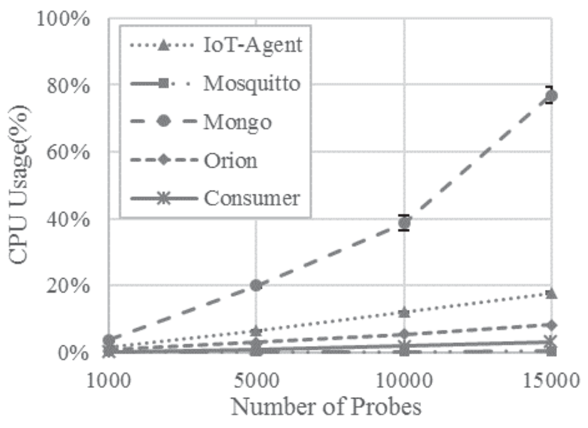


Fig. 6. CPU usage in the cloud-only experiments

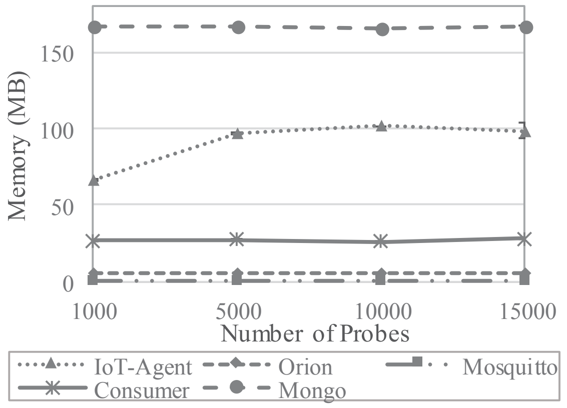


Fig. 7. Memory usage in the cloud-only experiments

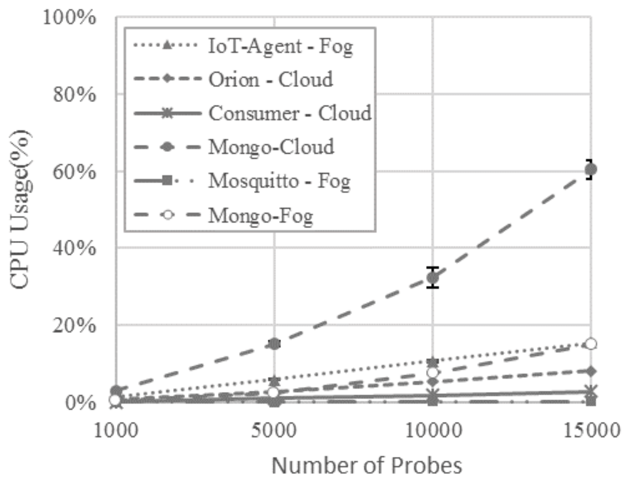


Fig. 8. CPU usage in the fog/cloud experiments

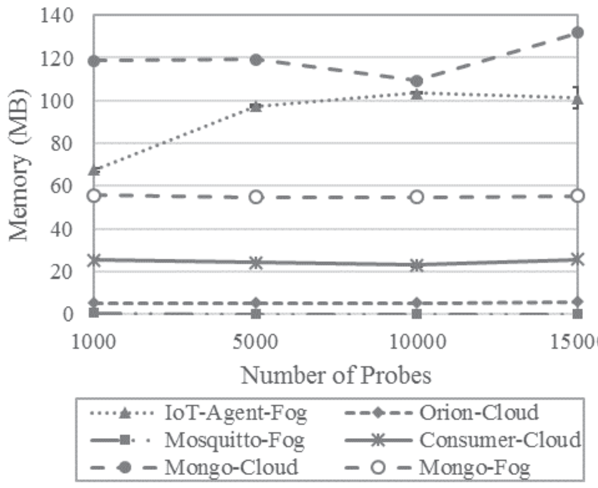


Fig. 9. Memory usage in the fog/cloud experiments

VI. Discussion

The experiments were successful in replicating a large-scale smart farm environment considering the number of sensors. Also, they showed that fog computing does not always improve the overall system performance. However, there are complex issues to evaluated in opting for a cloud or a fog approach beyond simply adding a machine in the edge of the network. There is a non-trivial trade-off between the given application, the available infrastructure and the system workload.

In the SWAMP scenarios, we expect a smaller number of sensors, due to costs and farm sizes, and therefore the experiments that better represent real case scenarios are the ones with 1,000 and 5,000 probes. In those cases, the addition of a fog processing node proved to be a downgrade regarding performance. Nevertheless, the fog has others benefits that compensate the lack of performance. In farms in the inward countryside of Brazil, when there is Internet connectivity, it is frequently unstable and limited, suffering from frequent disconnections. In this case, the fog node is responsible to store sensor data and keep a subset of services online when there is no connection to the Internet.

An important and unexpected factor in the experiments was the impact of the network. Initially, we believed that the network would equally impact both configurations fog and cloud). However, the impact was much higher in the fog configuration. This happens because the traffic of JSON/HTTP packets suffer more from network constrains than the lightweight UL 2.0 MQTT traffic. Therefore, when deploying a solution in a real environment, there is the need to analyze if the gain in processing part of the data in another device (the fog) will overcome the limitations imposed by the network. Another solution is to use a lightweight protocol when sending the data to the cloud form the fog.

This set of experiments created higher awareness about the scalability tradeoffs of the FIWARE IoT platform. A major concern is regarding the default database used by Orion and the IoT Agent (i.e. MongoDB) that was the major bottleneck of the experiments. It is a known fact that MongoDB does scale as well as other NoSQL databases, such as Cassandra [10].In the FIWARE documentation, there is no guide or configuration regarding using a different database for both software and it seems that they were hardcoded for working with MongoDB. FIWARE could potentially increase its scalability if it allows the of more efficient databases other than MongoDB. Another solution is to place MongoDB in a dedicated machine.

Although FIWARE has a scalability ceiling due to MongoDB, it is suitable for most of the IoT scenarios in smart farms, being capable to handle 10,000 probes without major performance issues. It is proven to be a reliable tool with several advantages. It provides a standard communication format between all services in a given system through the NGSI information model. FIWARE also provides a transparent way to applications – such as dashboards and analytics – to consume sensor data. The IoT Agent can handle data from different sensors and translate them to a standard NGSI and automatically send this data to Orion, whose API is very flexible and able to add new sensors on-the-fly. Orion also proved to be a reliable and useful tool to manage context information, as it stores the last data point of each attribute and provides a simple API to consult it. Also, by using the subscribe operation in Orion, applications do not need to perform polling in the interesting devices. Rather, when data is changed, Orion notifies the subscribed application.

VII. CONCLUSION

We analyzed the performance of FIWARE under different platform configurations comparing fog/cloud and cloud-only scenarios for precision irrigation using one of the SWAMP pilots as the evaluation scenario. Our results reveal non-intuitive outcomes, such as, that fog computing does not always improve the overall system performance. In some cases, the addition of a fog processing nodes even proved to worsen the performance. Also, the network between the farm and the cloud datacenter causes some unexpected differences between different scenarios. The results presented in this paper reveal the tip of the iceberg. In order to fully understand the tradeoffs involved in using IoT platforms and particularly FIWARE, we need to broaden our scope and perform experiments in different configurations, pilots and in scenarios beyond smart farming.

REFERENCES

[1] Aazam, M., Huh, E.-N., "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm," IEEE Potentials, vol. 35, no. 3, pp. 40–44, May/Jun. 2016.

[2] Bonomi, F. Milito, R., Zhu, J., Addepalli, S., "Fog Computing and its role in the Internet of Things", IEEE Workshop on Mobile Cloud Computing (MCC), pp. 13-16, 2012.

[3] Cardoso, J., Pereira, C., Aguiar, A., Morla, R., "Benchmarking IoT Middleware Platforms". IEEE 18th Intl Symposium on World of Wireless, Mobile and Multimedia Networks, WoWMoM 2017, April 2017.

[4] da Cruz, M.A., Rodrigues, J.J., Sangaiah, A.K., Al-Muhtadi, J., Korotaev, V., "Performance Evaluation of IoT Middleware, Journal of Network and Computer Applications, 109, pp.53-65., May 2018.

[5] Ericsson, "Cellular Networks for Massive IoT", White Paper, UEN 284 23-3278, January 2016, https://www.ericsson.com/res/docs/whitepapers /wp_iot.pdf, accessed in September 2018.

[6] FIWARE official website, Web: https://fiware-iotagent-ul.readthedocs.io/en/latest/usermanual/index.html

[7] Kamienski, C., Jentsch, M., Eisenhauer, M., Kiljander, J., Ferrera, E., Rosengren, P., Thestrup, J., Souto, E., Andrade, W., Sadok, D., "Application Development for the Internet of Things: A Context-Aware Mixed Criticality Systems Development Platform", Computer Communications, October 2016.

[8] Kamienski, C., Kleinschmidt, J., Soininen, J.P., Kolehmainen, K., Roffia, L., Visoli, M., Maia, R.F., Fernandes, S., "SWAMP: Smart Water Management Platform Overview and Security Challenges", 48th IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN'18), June 2018.

[9] Kamienski, C., Soininen, J.P., Taumberger, M., Fernandes, S., Toscano, A., Salmon, T., Filev, R. Torre, A., "SWAMP: an IoT-based Smart Water Management Platform for Precision Irrigation in Agriculture", Global IoT Summit 2018 (GIoTS'18), June 2018.

[10] KLEIN, J. et al. Performance Evaluation of NoSQL Databases : A Case Study Data bases Mapping the. p. 5–10, February, 2015.

[11] Martínez, R., Pastor, J.Á., Álvarez, B., Iborra, A., "A Testbed to Evaluate the FIWARE-based IoT Platform in the Domain of Precision Agriculture, Sensors, 16(11), November 2016.

[12] Morabito, R. Kjällman, J., Komu, M., "Hypervisors vs. Lightweight Virtualization: A performance Comparison", IEEE International Conference on Cloud Engineering, IC2E 2015, p. 386–393, 2015.

[13] Mukherjee, M., Shu, L., Wang, D., "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges", IEEE Communications Surveys & Tutorials, Third Quarter 2018.

[14] Rodriguez, M., Cuenca, L., Ortiz, A., "FIWARE Open Source Standard Platform in Smart Farming-A Review", Working Conference

[15] Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M., "Internet of Things for Smart Cities", IEEE Internet of Things Journal, 1(1), pp. 22-32, February 2014.

[16] Zyrianoff, I. Borelli, F., Biondi, G., Heideker, A., Kamienski, C., "Scalability of Real-Time IoT-based Applications for Smart Cities", IEEE Symposium on Computers and Communications (ISCC 2018), June 2018.