

FIRST STEPS IN CREATING A METHODOLOGY TO DEVELOP AND TEST SCHEDULING POLICIES FOR INTERNET OF THINGS

Paula Vergehet
Esteban Mocskos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Pabellón 1, Ciudad Universitaria
CABA, C1428EGA, ARGENTINA

ABSTRACT

Internet of Things (IoT) refers to a paradigm in which all objects can send information and collaborate with their computing resources through the Internet. The combination of Fog and Cloud Computing defines a distributed system composed of heterogeneous resources interconnected by different communication technologies. Despite its theoretical capacity, using these computational resources poses a challenge to distributed applications and scheduling policies. In this work, we show the initial steps in developing a tool to support the creation of scheduling policies combining simulation and validation. We show the details to be considered when selecting and configuring the different layers of software. To evaluate the proposal, we use a segmentation method in both platforms and a theoretical model to predict the total compute time. Our results show that both simulation and validation platforms agree in the obtained results which also can be explained in terms of a theoretical model.

1 INTRODUCTION

Internet of Things (IoT) refers to a paradigm in which any object can be part of the Internet. From mobile phones and sensors to smart clothes, these devices can communicate or be contacted to extract information and collaborate to perform a determined task Aazam and Huh (2016). Cloud Computing emerged to be an alternative to supply computer power and storage providing computing resources following a *pay as you go* model. Notwithstanding, the expected growth of data generated by the increasing amount of connected devices suggests that the centralized cloud data centers would suffer from network collapse. Fog Computing was recently proposed in Bonomi et al. (2012) to increase the computation being performed near the network edge using smart devices. Fog/Edge Computing aims to decrease bandwidth usage keeping the computation near the source of data and avoiding the movement of information to the cloud data centers.

The combination of Fog and Cloud Computing defines a distributed system composed of heterogeneous resources interconnected by different network technologies with diverse bandwidth, computer resources, storage, and latency. Despite its theoretical capacity, taking advantage of the computational resources included in this large-scale platform poses a challenge to scheduling policies, resource discovery and interchange of status information Chen et al. (2017).

One of the fundamental characteristics of this ecosystem is the need to reduce heterogeneity in protocols and policies. This large distributed system has to be accessible by different software applications, hardware solutions, and middlewares.

In this work, we show the first steps creating a platform to support the design and development of scheduling algorithms in Fog/Edge environments combining simulation and validation. The ultimate objective is providing a tool to allow the exploration of new ideas, their testing using different conditions and

scenarios, and the deployment of experiments in a controlled tested to establish their working conditions, advantages, and limitations. The study of heuristics to obtain better scheduling policies in distributed systems has a long and rich history but also new proposals can be easily found and are being developed by the community. For example, Luo et al. (2019) and Anglano et al. (2019) proposes new scheduling policies in Fog/Edge/Cloud environments, but both works use simulation tools with no further validation in real hardware.

As was recently mentioned in Svorobej et al. (2019), the validation of results obtained by simulation tools is a challenge in Fog/Cloud scenarios. Combining several tools to create a more useful one is the object of the hybrid simulation field, which has also a long history in different fields of application. Tolk et al. (2018) present a review which starts discussing the definition of hybrid simulation and gives an in-depth discussion of this field. Our proposal aims to create a platform which can combine simulation and validation in an easy way. The user starts the exploration using a simulation tool in which different configurations, parameters, and scenarios can be tested. Once this stage is finished, our proposal manages the deployment in the validation platform and supports the recovering of the results. In this way, any user can program a new scheduling policy, put it under test in the simulation world and then validate it using a large scale distributed facility with real hardware.

The rest of this paper is organized as follows: in section 2, we introduce the facility used to perform the validation stage, including its network topology, the available hardware and some measurements of its network behavior. The simulation engine is presented in section 3, while in section 4 we present the methodology and the link between both platforms. Section 5 shows the results obtained using simulation and validation in real hardware. Finally, in section 6 we draw some conclusions.

2 EXPERIMENTAL PLATFORM: FIT/IoT-LAB

Validation stage is usually one of the most critical parts in any scientific development. When designing scheduling policies for heterogeneous distributed systems like the ones defined in the Fog/Edge paradigm, the creation, management, and deployment of a distributed infrastructure could become extremely difficult or even impossible. Notwithstanding that using real hardware to test a proposal in scheduling policies could provide more insight and robustness about its behavior and properties.

FIT IoT-LAB testbed offers an open access multi-user scientific tool to support design, development, tuning, and experimentation related to IoT, see Adjih et al. (2015). FIT IoT-LAB testbeds are located at six different sites across France enabling the use of more than 2700 wireless sensors nodes (see in Figure 1).

To perform an experiment using the FIT IoT-LAB's resources, it is necessary to reserve them beforehand. The reservation (*slice* in terms of FIT IoT-LAB's terminology) can be obtained to access the resources immediately (based on their availability) or for a specific time and date in the future. The resources to use in an experiment can be selected based on their id, architecture, mobility, and site, providing a high level of traceability and reproducibility. In the rest of this section, we present the hardware and software characteristics of FIT IoT-LAB including the results of performing monitoring of the network infrastructure to evaluate its stability.

2.1 Network Infrastructure

The available infrastructure provided by FIT IoT-LAB includes a wide range of sensors, robots and computing resources distributed in different sites. An experiment could use resources from only one site which are connected by a local switch or can be composed of distributed resources among two or more sites. Figure 1 shows the current configuration of FIT IoT-LAB: different types and amount of resources are installed in each site, all of them are connected locally and, by one gateway per site, to the external resources.

The connectivity of the different sites is provided by RENATER which also supports the access of researchers worldwide through the different advance networks in each country. Each link in Figure 1 has 10 Gbit/s, but this backbone is increased with parallel links to provide more bandwidth like between

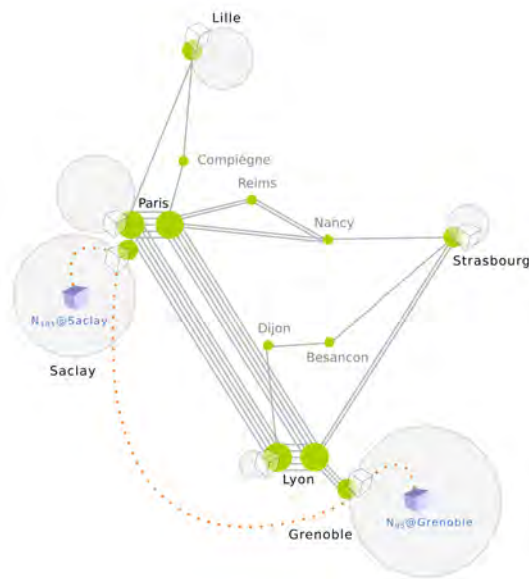


Figure 1: FIT IoT-LAB infrastructure: six sites across France provide IoT resources. These sites are interconnected using RENATER. All the base links support 10 Gbit Ethernet and some can reach 100 Gbit/s like Paris 1 - Paris 2 and Paris - Lyon.

Paris and Lyon. For example, the sensor $N_{101}@Saclay}$ needs to send data to the actuator $N_{95}@Grenoble}$. Then, $N_{101}@Saclay}$ sends a request to start a connection through the gateway in Saclay (represented by the transparent cube in the figure). The connection between the gateways in Saclay and Grenoble uses the RENATER's infrastructure (based on TCP/IP). Finally, the gateway in Grenoble forwards the request to $N_{95}@Grenoble}$ and establishes the connection delivering the data.

The monitoring and control of the experiments can be performed using usual networks tools and there is a local NFS-based file system which is available for some of the devices to share files in the same site. Each site has its own file system which is not shared among different sites, i.e. as a user, you need to copy files between different sites manually.

As our objective is to create a tool that links simulation and validation, the dynamic behavior of the network infrastructure should be evaluated. We test the stability of local network infrastructure and intersite communications to quantify the real latency that an application could experience when running in FIT IoT-LAB. We set up an experiment to monitor the state and response time (i.e. RTT) of the connections between all the six sites and their internal behavior. We request resources in each site and monitor the RTT during the first 10 minutes of each hour for five days. Two scenarios are considered: intrasite (i.e. only local connectivity) and intersite (i.e. connectivity among resources in different sites). In the first case, only one hop is necessary to reach the destination while for the second scenario, the routes have more than five hops including both gateways in the source and destination sites.

Tables 1 and 2 present the results of monitoring measurements for internal traffic (i.e. intrasite). During the experiment performed in June 2017 (Table 1), all the sites present a local average RTT between 1.5 ms to 2.0 ms. The difference in the behavior of the sites is in the stability of the measurements, Paris and Saclay present a similar standard deviation, while Strasbourg has higher deviation than the two previous sites and Grenoble presents a very unstable behavior with the highest values of standard deviation.

The next experiment was performed during November 2018 after a reconfiguration of FIT IoT-LAB network configuration update. In Table 2, all the sites present a lower average RTT in the range between 0.8 ms to 1.3 ms. The stability also improved, only Grenoble presents a strong variability during the first day and a small one during the fourth day. The more stable site is Strasbourg followed by Saclay and Paris.

Table 1: Experiment 06/2017: Average Round Trip Time (RTT) of near 14000 observations for intrasite connectivity during June 2017 (expressed in ms).

Hours	Grenoble		Paris		Saclay		Strasbourg	
	avg	sd	avg	sd	avg	sd	avg	sd
0-23	1.602	6.203	1.560	0.401	1.539	0.379	1.577	0.74
24-47	1.820	12.460	1.560	0.390	1.538	0.378	1.544	0.723
48-71	1.852	15.925	1.555	0.386	1.569	0.479	1.523	0.151
72-95	1.926	9.945	1.573	1.708	1.546	0.382	1.519	0.151
96-119	1.621	6.706	1.564	0.491	1.551	0.491	1.537	0.147

Table 2: Experiment 11/2018: Average Round Trip Time (RTT) of near 14000 observations for intrasite connectivity during November 2018 (expressed in ms).

Hours	Grenoble		Paris		Saclay		Strasbourg	
	avg	sd	avg	sd	avg	sd	avg	sd
0-23	1.164	15.724	1.247	0.287	1.233	0.147	1.280	0.094
24-47	0.933	0.227	1.250	0.468	1.245	0.128	1.274	0.097
48-71	0.845	0.457	1.248	0.232	1.237	0.141	1.273	0.097
72-95	0.855	1.125	1.246	0.346	1.250	0.103	1.276	0.092
96-119	0.932	0.254	1.243	0.237	1.239	0.203	1.279	0.099

Figure 2 shows the measurements of network behavior from Saclay and Strasbourg to the rest of the sites. Although five days of data were recorded and analyzed, two representative days are included in this figure to improve legibility. The figure includes data from two experiments: June 2017 (Experiment 06/2017 and November 2018 (Experiment 11/2018), but the intersite behavior does not show changes in these two experiments. The minimum value for RTT is related to the number of hops contained in the route to the destination. As Paris is near Saclay (see Figure 1), Figure 2(a) and Figure 2(c) show the lowest value for RTT between them. To reach Lille from Saclay it is needed to hop through Paris, which explains the increment in RTT while Grenoble and Strasbourg need more hops with higher RTT. The traffic from Strasbourg to the rest of the sites is presented in Figure 2(b) and Figure 2(d). Strasbourg is the more remote site in terms of network connectivity, it is needed several hops to reach any other site. This is reflected in the high values of RTT which are considerably higher than Saclay. Paris is the nearest site in terms of RTT with the rest of sites near it. The path Strasbourg to Lille presents the highest RTT value from all the analyzed data.

The analysis of the combination of local and intersite traffic patterns regarding expected latency, network instability which jointly with the availability of resources guided us to select which site fits better the requirements for each experiment. These considerations should be managed by a tool to help the user define the experimental conditions coordinating simulation and validation. For example, Strasbourg presents a more stable traffic pattern intrasite, especially for our latest measurements, but using Strasbourg for intersite experiments increments the noise and latency involved.

2.2 Hardware Infrastructure

One of the main objectives of FIT IoT-LAB is providing a platform to develop and test applications, modules, and protocols for small wireless sensor devices and heterogeneous communicating objects. This facility has a large number of distributed resources, which backbone is based on three types of devices specifically designed for FIT IoT-LAB:

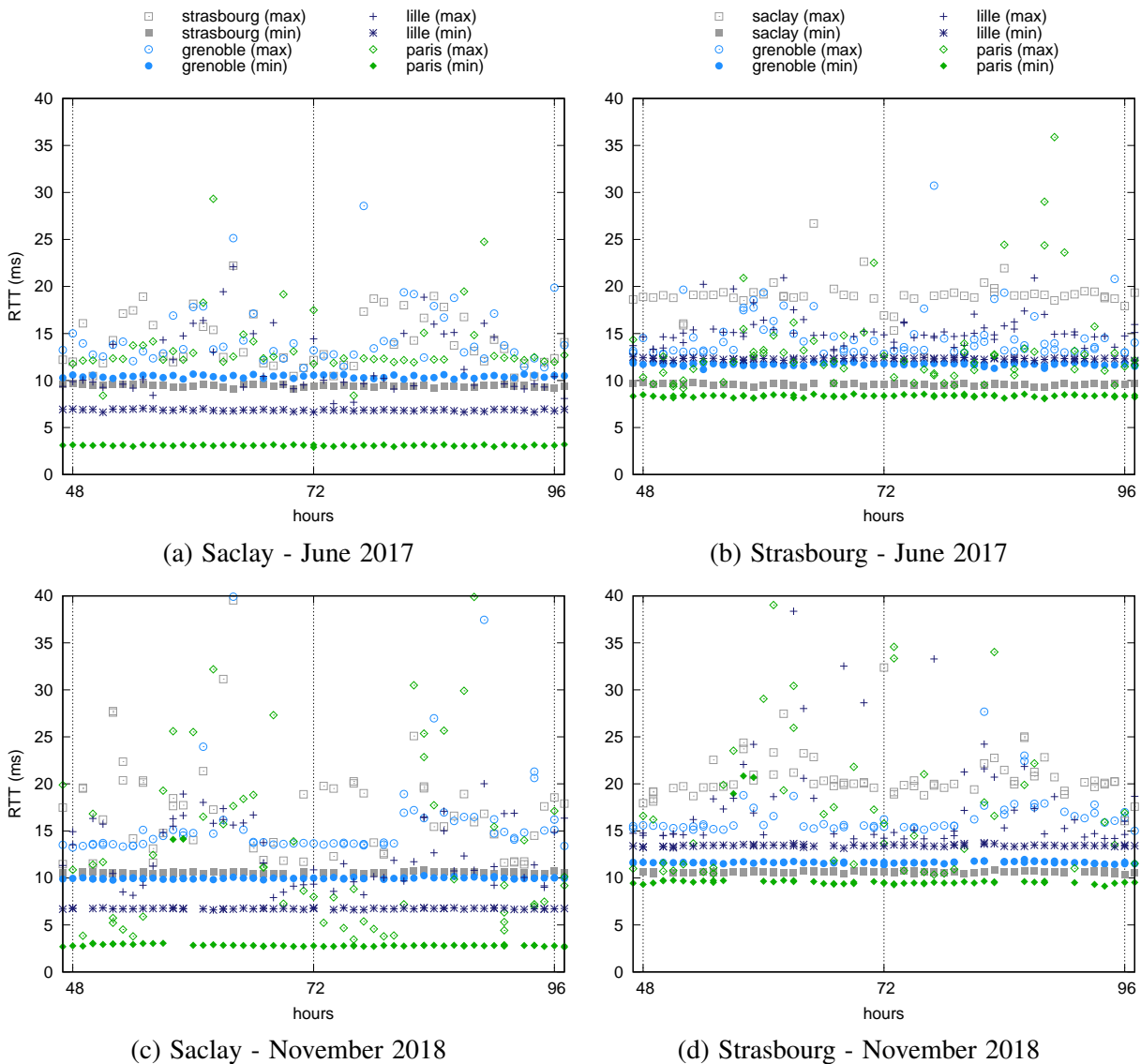


Figure 2: Monitoring of RTT between Saclay and Strasbourg with the rest of the available sites during the first 10 minutes of each hour during seven days (only third and fourth day are shown to improve legibility). The experiments in (a) and (b) correspond to June 2017 (Experiment 06/2017), while the experiments in (c) and (d) were performed during November 2018 (Experiment 11/2018).

- WSN430 Node: based on MSP430F1611 MCU and communication with a 802.15.4 PHY Layer (800 MHz or 2.4 GHz).
- M3 Node: based on STM32F103REY MCU and communication with a 802.15.4 PHY Layer (2.4 GHz)
- A8 Node: based on TI SITARA AM3505 (ARM Cortex A8) allows to run Linux. This node embeds also an M3 Node with 802.15.4 comm.

The first two types of devices only support a lightweight operating system (like FreeRTOS, Contiki, or Riot) while the A8 Node can execute a standard Linux distribution. A8 Nodes consists of a single core Texas Instruments AM3517 / AM3505 processor executing at 600 MHz with 256 MB of RAM. Additionally,

FIT IoT-LAB has a wide range of commercial boards installed including Zigduino, BBC micro:bit, and Zolertia Firefly.

3 SIMULATION PLATFORM: SIMGRID

The next step to build a develop-test-validate framework for scheduling algorithms in IoT is to select an efficient and versatile simulation tool to support the initial exploration. There are several proposals that could be used: CloudSim by Calheiros et al. (2011), Simgrid by Casanova (2001), Batsim by Dutot et al. (2017), iFogSim by Gupta et al. (2017), and pysimgrid by Sukhoroslov et al. (2018).

The main characteristics needed are:

- *Processing time*: the tool needs to support executing jobs with different complexity (in terms of computing power needed) considering heterogeneous computing resources.
- *Network traffic*: the sending and receiving of data and commands use the network infrastructure to travel from the source node (i.e. scheduler in our case) to destination (i.e. worker). The simulation tool has to consider the latency and bandwidth of the link to incorporate the effects of latency and congestion. Moreover, usually wireless with different flavors is one of the preferred ways to communicate IoT devices.

The event-based simulator Cloudsim and iFogSim families are focused on the evaluation of virtualization techniques or distributed applications development in Fog/Cloud environments, but these tools use high-level abstract models for communication and computing, and lack of support for wireless communication. These tools are implemented in Java and to adding new modules or models to them imposes using this technology.

Simgrid is also an event-based simulation engine which is widely used worldwide having an active and large community of users. In spite of lacking support for wireless communication, the communication and computing models are more sophisticated and versatile than other tools. Simgrid is natively programmed in C/C++ but it has a clear API that can be used to interact with it from other languages. Pysimgrid is a Python-based framework to support interacting with SimGrid and Batsim is a new layer on top of SimGrid focused on scheduling, but currently, this project is under heavy development.

Our proposal is designed to be flexible to use different simulation tools, but for this initial prototype, we need to adopt one simulation platform. Based on the state of maturity, our previous experience with the tool, the size of the community and versatility of the solution, we select SimGrid as the simulation engine and pysimgrid to interact with it taking advantage of Python.

3.1 Hardware and Network Infrastructure

As we already mentioned, the simulation tool needs to support realistic models for heterogeneous devices and communication network. Simgrid allows configuring of the network topology using an input file in which the topology, bandwidth, and latency for each link are defined. The computing power of all the resources is also specified in this configuration file. Simgrid uses these values to compute the time needed to complete a job based on the declared requirements of the task and the resources used to solve it. SimGrid has several communication models to choose when computing the time required for a message to reach its destination. In these models, latency and bandwidth are combined with the message size and the considered conditions of the network infrastructure to compute the communication time.

For any scenario, compute power and network details need to be defined. Moreover, when using an experimental platform as a validation stage for simulation results, both properties have to be carefully selected to match the real hardware and network topology (i.e. FIT IoT-LAB) with the simulation tool.

The network parameters of the validation platform can be extracted from the measurements presented in Figure 2 and Tables 1 and 2. The computing power of the available devices needs to be precisely defined in the configuration of the simulation tool in terms of FLOPS. For example, Aroca and Gonçalves (2012)

estimate the peak performance of an ARM Cortex-A8 in 33 MFLOPS and an average of 25 MFLOPS, while in Frlinger et al. (2011), the authors obtain 66.7 MFLOPS but only 40 MFLOPS when running the Linpack benchmark. Rajovic et al. (2014) conclude that the Cortex-A8 can deliver near 60 MFLOPS at most, which is similar to the value found by Cloutier et al. (2016).

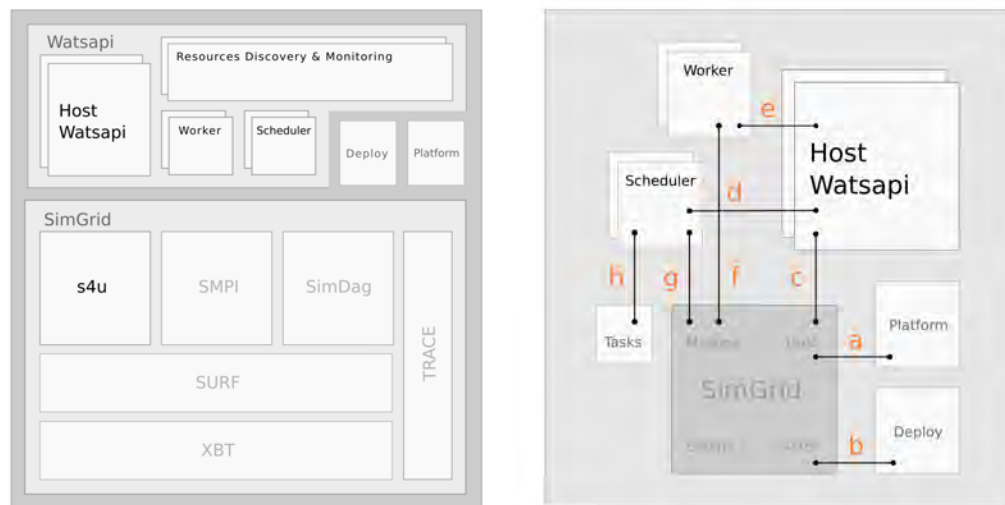
This wide range of possibilities neglects the selection of a single value representing the behavior, even for only one type of computing resource. In our case, the procedure to select this value is based on executing a *representative* task. Based on the performed measurements, we estimate 60 MFLOPS as the computing power of the available Cortex-A8.

4 SIMULATION MODEL DEVELOPMENT

The creation of a platform to develop, test and validate scheduling policies in Fog/Edge/Cloud environments requires not only access to the validation platform (in our case, FIT IoT-LAB) and the simulation engine but also software modules to support their use.

As a first step in creating the glue between these technologies and resources, we present Watsapi, which is a software tool created in the top of Simgrid and aimed to offer a simulation platform to execute simulations in Fog/Edge environments which then can be easily deployed in the validation platform provided by FIT IoT-LAB.

Watsapi uses the Simgrid’ module s4u to manage communication between entities, which is a recent replacement for the older MSG interface. Figure 3(a) presents the current software modules developed to support the simulation of scheduling policies while Figure 3(b) shows the relationship between all the software modules. The main entity is the Host Watsapi which can be specialized to become a scheduler or a worker. The tool is also prepared to add resource discovery and monitoring support, but to simplify this first prototype and its analysis, we disabled this functionality for this work.



(a) Software modules used and developed.

(b) Interactions between entities.

Figure 3: Modules involved in the definition of the tool which combines the simulation engine with the validation platform.

Host Watsapi can be used as a scheduler (allowing the definition of multiple schedulers in a scenario) or a worker. It is also possible to use a scheduler as a worker too, but as the validation platform only has single core resources, assigning multiple roles to a node could impact negatively on the results. In

terms of SimGrid, each of these entities is an Actor executing at one Host Watsapi. We use the registration mechanism to assign the functions to each entity which are defined in the configuration file.

The standard use scenario for this tool can be briefly defined as: each scheduler has a set of tasks to execute and a list of computing resources. It uses a scheduling policy to select the resource (i.e. worker) to send the task to be served. The workers are initially waiting to execute a task. They start serving the first task to arrive and, in the case of receiving more, the new tasks are queued until they finish the current one.

The interactions defined in Figure 3(b) are: a) The network topology, link bandwidth and latency, and type and amount of nodes are described in a config file (i.e. platform) which is then read by Simgrid to define the scenario. b) The functions to be executed in each node and variables to monitor are specified in another configuration file (i.e. deploy), which is also an input for Simgrid. The functions for each host are defined in this file, which then are assigned to objects of the class `s4u::Actor`. c) Simgrid's host class is extended to support new functionalities to become HostWatsapi class. This new class supports new specific logging details and managing of known computing resources. In Fog/Edge environment, on-the-fly clustering is one possible strategy to deal with complex tasks, the knowledge of near resources could be necessary for scheduling, but also for workers to offload tasks in case of need. d) The Scheduler is an extension to HostWatsapi. It needs to maintain the list of pending tasks, available resources, and tasks currently being solved in remote resources. To be able to communicate with other entities, this class has to define a Simgrid communication port (i.e. mailbox). e) The Worker usually waits for a task to compute. f) To receive one task, a worker uses the communication infrastructure provided by Simgrid. It starts waiting until a message is received in its mailbox. Once the task is received, it continues busy until its end. g) A scheduler uses the Simgrid communication layer to send the tasks to a worker and to receive the results. h) The list of tasks to be served is managed by a container class implemented.

The relationship between simulation and validation stages is introduced in Figure 4, which corresponds to two consecutive time steps in the dynamics of the involved entities during simulation and validation.

In these figures, some of the events are marked in both stages to show how they are linked: I) One of the nodes in this example is a scheduler, it has to match tasks to solve and available computing resources. II) The tasks to be solved are received by the scheduler and stored in memory. III) The scheduler has information about the state of the available resources. IV) Based on the resource information and the scheduling policy, the node selects a computing resource to send the task to be executed. V) The computing resources have a list of pending tasks. VI) After some time (Δt), a node finishes the execution of a task and sends the results to the scheduler. In the case of the simulation, the node uses Simgrid infrastructure to send it, while in the validation platform, the shared file system is used to communicate the results. VII) The scheduler detects the end of a task and moves it from the pending to the ended list. VIII) The resources information is updated by the scheduler.

This methodology enables the development and testing of scheduling policies for heterogeneous scenarios in terms of computing resources and network communication infrastructure. In the next section, we propose an example to test this proposal and evaluate its behavior.

5 RESULTS

The next step is proposing a scenario to demonstrate the potentiality of this proposal. The image segmentation procedure is an extended methodology which takes an image or set of images and generates a representation which should be easier to process or analyze. In the case of an IoT scenario, a set of devices could be installed to process images or other sensor data, for example the Array of Things project, which is being deployed in Chicago (US), aims to install a large number of devices with a set of sensors (including video camera) and computing resources to process images in the edge. It is easy to imagine a large number of similar projects relaying in this kind of resources to process images or other complex data in the edge trying to detect objects or situations of interest without the need to transfer the data to a centralized data center. In the case of the array of things project, the images have to be processed in each node due to privacy concerns.

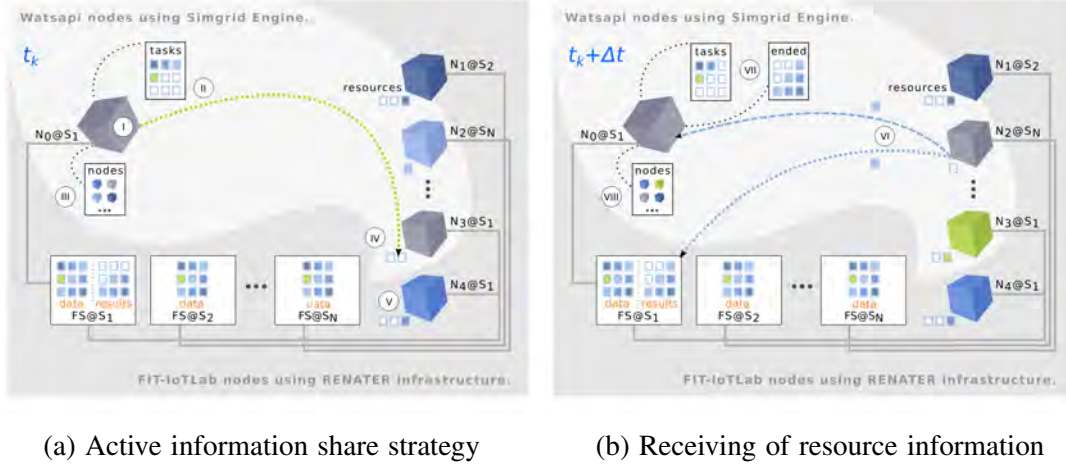


Figure 4: Implementation in simulation and validation platforms. At time t_k the scheduler has some pending tasks and available resources. It sends a task to N_3 , one of the available resources. After some time (represented by Δt), N_2 sends the results of a previously assigned task with a message in the the simulation platform, while in the validation platform it uses an available local file system to communicate it.

Image segmentation is widely used in many diverse areas like pattern and text recognition Lyu et al. (2005), document image processing Farrahi Moghaddam and Cheriet (2012), crack concrete structures detection Talab et al. (2016), and medical image Manikandan et al. (2014). One of the methods established for this objective is the method by Otsu (1979). It can be applied independently to each frame belonging to a sequence (i.e. defines an embarrassingly parallel workload).

We apply this method to a dataset based on Le Ballon Rouge sequence (Lamorisse 1956) previously processed to convert it to grayscale and modified with Gaussian noise. This dataset has 4520 frames, each one with 480 pixels \times 360 pixels as is exemplified in Figure 5. Figure 5(a) is the original frame in grayscale, while in Figure 5(b) Gaussian noise is added and Figure 5(c) represents the result of the segmentation procedure in black and white. We select 100 frames to decrease the amount of time needed to execute these tests in the hardware platform and the disk space needed to store it. We propose two scenarios to execute the test:

- **Intrasite:** the scheduler and all the computing resources are located on the same site. In this case, all the entities experience the same latency.
- **Intersite:** the scheduler and the computing resources are in different sites: the scheduler is in one site and it uses resources located in two other sites. The entities have different access time depending on the site they are deployed.

In both scenarios, we solve the workload composed of 100 frames using from one to ten computing devices. The selection of resources is performed using Random and Round-Robin scheduling policies. Intrasite experiments are performed in Grenoble while the intersite are based on Grenoble, Saclay, and Strasbourg. As all the frames in the workload can be processed independently, the compute time for all the dataset can be obtained by the following equation:

$$\text{Expected_Time}(n) = \frac{\text{single_time} \cdot \text{tasks_count}}{n} + \text{tasks_count} \cdot (t_1 + t_2) - (t_1 + t_2) \quad (1)$$

where n is the number of workers collaborating to solve the workload, single_time is compute time for a single frame, task_count is the number of frame to be processed, t_1 corresponds to the scheduling overhead and t_2 is the communication overhead.

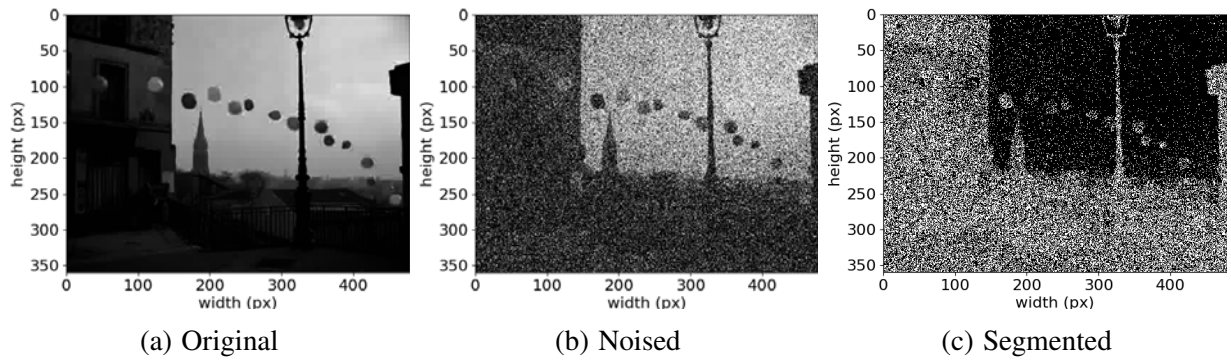


Figure 5: One of the frames of Le Ballon Rouge sequence: (a) is the original in grayscale, while (b) is modified with Gaussian noise and (c) is the result of the segmentation method in black and white.

For each number of workers and scheduling policy, the experiments in the validation platform are repeated ten times to keep the use of FIT IoT-LAB under a reasonable limit. In the case of the simulation stage, each case is repeated 1000 times.

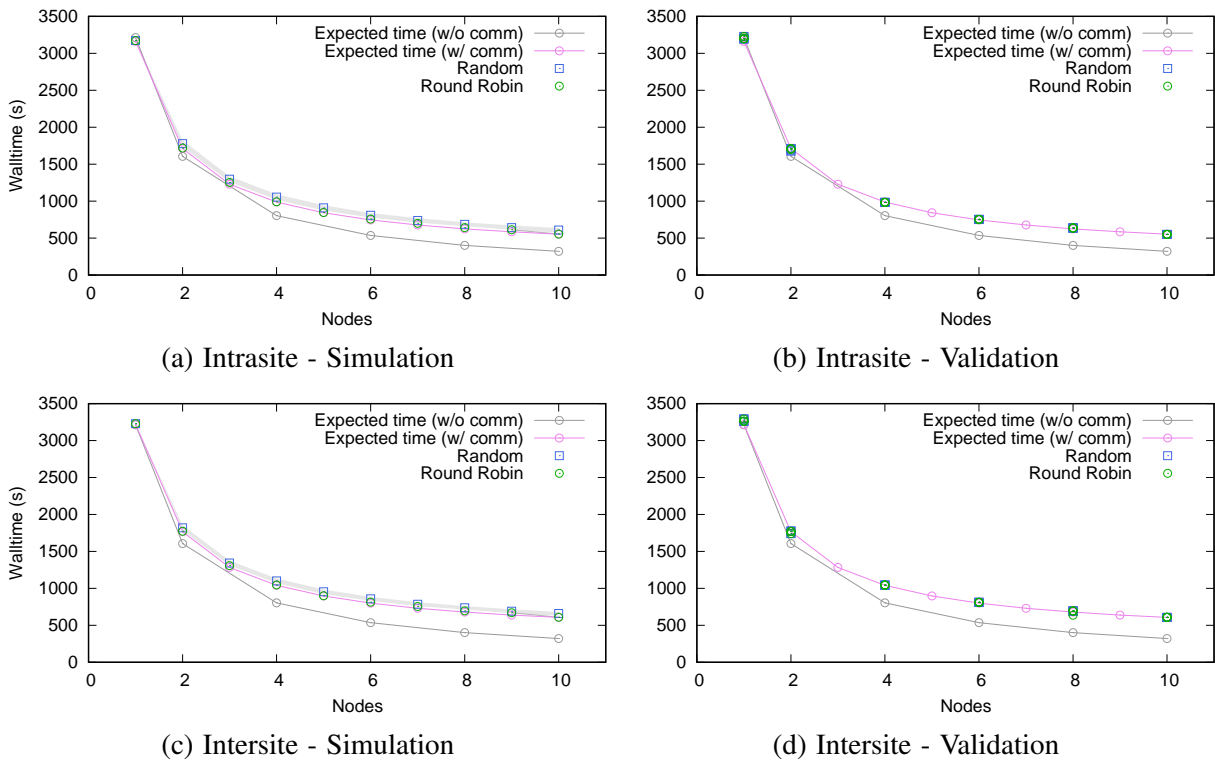


Figure 6: Intrasite experiments: total time needed to process 100 frames with the Otsu segmentation method using an increasing number of computing resources in the same site: (a) and (c) present the results of executing in the simulation platform, while (b) and (d) show the results in real hardware. The continuous lines corresponds to the model presented in Equation 1 with and without considering overheads.

Figure 6 shows the total amount of time to solve the workload using an increasing number of computing resources in simulation and validation platforms. As can be seen, both platforms reasonably agree with the total time needed to solve the workload. On the other hand, the predicted computing time obtained

using Equation 1 without considering overheads (labeled Expected_time(w/o comm) in the figure) shows strong differences with both platforms especially when the number of involved resources increases. As the scenario of Figures 6(a) and 6(b) has very low latency, we can unify the overhead setting $t_1 = 2.65$ s and keeping $t_2 = 0$. With this improvement, the predicted value (labeled Expected_Time (w/comm) in the figures) fits with the obtained results in both platforms.

Figures 6(c) and 6(d) present the results for solving the same dataset in a scenario in which the scheduler and the resources are in different sites (intersite experiment). Both platforms agree in the time needed to complete the workload. Once again, not considering the overhead in Equation 1 leads to a disagreement with the measurements and simulation. In this case, as the entities are deployed in different sites, the communication overhead (t_2) needs to be incorporated. With a value for $t_2 = 0.55$ s, the line labeled Expected_Time (w/comm) is obtained agreeing with measurements in both platforms.

6 CONCLUSIONS

In the Internet of Things (IoT) paradigm, all the devices can send data and collaborate with their computing resources. Developing applications, protocols, and algorithms to take advantage of the complexity emerged in the combination of devices and communication characteristics is not a trivial task.

In this work, we present an initial effort to support the development of scheduling policies in Fog/Edge environments. We show the details about the implementation of the simulation platform based on Simgrid and a validation platform which uses the resources available in FIT IoT-LAB open distributed facility. We show how both platforms can be linked based on measurements of network behavior and computing performance of the available resources.

To evaluate our proposal, we segment images using the Otsu method with an increasing number of resources in both simulation and validation platforms. A theoretical model explains the obtained results in terms of the communication and scheduling overhead. We configure two scenarios: one in which all the computing resources and the scheduler share the same site and another in which they are deployed in different ones increasing the impact of latency. The results show that this methodology is promising to support the development and testing of new strategies focused on the complex system defined in the Fog/Edge/Cloud environment.

ACKNOWLEDGMENTS

This work is supported by Universidad de Buenos Aires (UBACyT 20020130200096BA), Consejo Nacional de Investigaciones Científicas y Técnicas (PIO13320150100020CO), and Agencia Nacional de Promoción de Ciencia y Técnica (PICT-2015-2761 and PICT-2015-0370). The authors thanks the access to FIT IoT-LAB infrastructure and the support received by its team while performing the experiments.

REFERENCES

- Aazam, M., and E. N. Huh. 2016. "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm". *IEEE Potentials* 35(3):40–44.
- Adjih, C., E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. 2015. "FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed". In *Proceedings of IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 459–464: Institute of Electrical and Electronics Engineers, Inc.
- Anglano, C., M. Canonico, and M. Guazzone. 2019. "WQR-UD: An Online Scheduling Algorithm for FemtoClouds". In *Proceedings of the 12th EAI International Conference on Performance Evaluation Methodologies and Tools*, 179–182. New York, NY, USA: Association for Computing Machinery.
- Aroca, R. V., and L. M. G. Gonçalves. 2012. "Towards Green Data Centers: A Comparison of x86 and ARM Architectures Power Efficiency". *Journal of Parallel and Distributed Computing* 72(12):1770–1780.
- Bonomi, F., R. Milito, J. Zhu, and S. Addepalli. 2012. "Fog Computing and Its Role in the Internet of Things". In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12*, 13–16. New York, NY, USA: Association for Computing Machinery.

- Calheiros, R. N., R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. 2011. "CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms". *Software: Practice & Experience* 41(1):23–50.
- Casanova, H. 2001. "Simgrid: A Toolkit for the Simulation of Application Scheduling". In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, 430–437. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Inc.
- Chen, Y. C., Y. C. Chang, C. H. Chen, Y. S. Lin, J. L. Chen, and Y. Y. Chang. 2017. "Cloud-Fog Computing for Information-Centric Internet-of-Things Applications". In *Proceedings of the International Conference on Applied System Innovation (ICASI)*, 637–640. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Inc.
- Cloutier, M. F., C. Paradis, and V. M. Weaver. 2016. "A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement". *Electronics* 5(4):61.
- Dutot, P.-F., M. Mercier, M. Poquet, and O. Richard. 2017. "Batsim: A Realistic Language-Independent Resources and Jobs Management Systems Simulator". In *Job Scheduling Strategies for Parallel Processing*, edited by N. Desai and W. Cirne, Volume 10353 of *Lecture Notes in Computer Science*, 178–197. Cham, Switzerland: Springer International Publishing.
- Farrahi Moghaddam, R., and M. Cheriet. 2012. "AdOtsu: An Adaptive and Parameterless Generalization of Otsu's Method for Document Image Binarization". *Pattern Recognition* 45(6):2419 – 2431.
- Fürlinger, K., C. Klausecker, and D. Kranzlmüller. 2011. "Towards Energy Efficient Parallel Computing on Consumer Electronic Devices". In *Information and Communication on Technology for the Fight against Global Warming*, edited by D. Kranzlmüller and A. M. Toja, 1–9. Berlin, Germany: Springer Berlin Heidelberg.
- Gupta, H., A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya. 2017. "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments". *Software: Practice & Experience* 47(9):1275–1296.
- Lamorisse, A. 1956. "Le Ballon Rouge (The Red Balloon)". Classic French fantasy comedy-drama filmed in the Ménilmontant neighbourhood of Paris. Freely available at <http://publicdomainmovie.net/movie/le-ballon-rouge-the-red-balloon>.
- Luo, J., L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, and H. Luo. 2019. "Container-based Fog Computing Architecture and Energy-balancing Scheduling Algorithm for Energy IoT". *Future Generation Computer Systems* 97:50 – 60.
- Lyu, M. R., J. Song, and M. Cai. 2005. "A Comprehensive Method for Multilingual Video Text Detection, Localization, and Extraction". *IEEE Transactions on Circuits and Systems for Video Technology* 15(2):243–255.
- Manikandan, S., K. Ramar, M. W. Iruthayarajan, and K. Srinivasagan. 2014. "Multilevel Thresholding for Segmentation of Medical Brain Images using Real Coded Genetic Algorithm". *Measurement* 47:558–568.
- Otsu, N. 1979. "A Threshold Selection Method from Gray-Level Histograms". *IEEE Transactions on Systems, Man, and Cybernetics* 9(1):62–66.
- Rajovic, N., A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez. 2014. "Tibidabo: Making the Case for an ARM-based HPC System". *Future Generation Computer Systems* 36:322 – 334.
- Sukhoroslov, O., A. Nazarenko, and R. Aleksandrov. 2018. "An Experimental Study of Scheduling Algorithms for Many-Task Applications". *Journal of Supercomputing*:1–15.
- Svorobej, S., P. Takako Endo, M. Bendeche, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn. 2019. "Simulating Fog and Edge Computing Scenarios: An Overview and Research Challenges". *Future Internet* 11(3):55.
- Talab, A. M. A., Z. Huang, F. Xi, and L. HaiMing. 2016. "Detection Crack in Image using Otsu Method and Multiple Filtering in Image Processing Techniques". *Optik* 127(3):1030 – 1033.
- Tolk, A., E. H. Page, and S. Mittal. 2018. "Hybrid Simulation for Cyber Physical Systems: State of the Art and a Literature Review". In *Proceedings of the Annual Simulation Symposium, ANSS '18*, 10:1–10:12. San Diego, CA, USA: Society for Computer Simulation (SCS) International.

AUTHOR BIOGRAPHIES

PAULA VERGHELET is a Teaching Assistant at the Computer Science Department, Universidad de Buenos Aires. She holds a Licentiate degree in Computer Science (equivalent to Msc) specialized in distributed systems. She is author of several contributions in the field of High Performance Computing focused on distribution of resource information. She can be contacted in the email address pverghelet@dc.uba.ar.

ESTEBAN MOCOSKOS is an Associate Professor at the Computer Science Department, School of Sciences, Universidad de Buenos Aires (UBA), Argentina. He is also researcher at the Center for Computer Simulation for Technological Applications (CSC-CONICET). He holds a PhD in Computer Science from UBA. His research interests include the areas of distributed systems, computer networks and protocols, parallel programming and applications. He can be contacted in emocskos@dc.uba.ar.