Computing and Informatics, Vol. 37, 2018, 815–837, doi: 10.4149/cai_2018_4_815

PERFORMANCE MODELS FOR FROST PREDICTION IN PUBLIC CLOUD INFRASTRUCTURES

Lucas E. Iacono

ITIC Research Institute
Consejo Nacional de Investigaciones Científicas
y Técnicas (CONICET), Argentina
&
Facultad de Ingeniería, Universidad Nacional de Cuyo
Mendoza, Argentina
e-mail: liacono@uncu.edu.ar

José Luis Vázquez Poletti

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid
Madrid, Spain
e-mail: jlvazquez@fdi.ucm.es

Carlos García Garino

ITIC Research Institute
Facultad de Ingeniería, Universidad Nacional de Cuyo
Mendoza, Argentina
e-mail: cgarcia@itu.uncu.edu.ar

Ignacio Martín Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid
Madrid, Spain
e-mail: llorente@dacya.ucm.es

Abstract. Sensor Clouds have opened new opportunities for agricultural monitoring. These infrastructures use Wireless Sensor Networks (WSNs) to collect data on-field and Cloud Computing services to store and process them. Among other applications of Sensor Clouds, frost prevention is of special interest among grapevine producers in the Province of Mendoza – Argentina, since frost is one of the main causes of economic loss in the province. Currently, there is a wide offer of public cloud services that can be used in order to process data collected by Sensor Clouds. Therefore, there is a need for tools to determine which instance is the most appropriate in terms of execution time and economic costs for running frost prediction applications in an isolated or cluster way. In this paper, we develop models to estimate the performance of different Amazon EC2 instances for processing frosts prediction applications. Finally, we obtain results that show which is the best instance for processing these applications.

Keywords: Cloud computing, wireless sensor networks, frost prediction, virtual clusters, sensor clouds, Amazon EC2

1 INTRODUCTION

In the Province of Mendoza, region of Cuyo, Argentina, frost is one of the main causes of crop damage. This meteorological event causes damage in vineyards and fruit trees, which are the main agricultural products of the province. In some cases, like in the year 2013, frost damages affected up to 80 % of crops and resulted in the economic emergency of the region. Currently, there are different defense methods that can be used to minimize frost damage, the most commonly used are sprinklers, heaters and wind turbines [26].

Defense systems are activated by frost alarms, which are provided by Frost Alarm Systems (FAS). FAS perform on-field data acquisition and data management. Moreover, FAS ensure production quality and guarantee crops traceability. On the one hand, the on-field data acquisition process can be performed using traditional instruments like thermometers, weather stations or Wireless Sensor Networks (WSNs) [1, 23]. When making a comparison of traditional measurement instruments and weather stations, WSNs have the advantage of covering extensive areas with low cost devices called sensor nodes. This advantage is of special interest when studying frost due to the dependence of this phenomenon with terrain characteristics, like presence of weeds, trees or closeness to mountains. Sometimes frost occurrence has only been observed in a few hectares of the farm (such as those at the base of mountains) while it has not been observed in other hectares of the same farm (such as those surrounded by trees).

WSNs data management process includes data remote access, storage and data processing. This process can be reliably and easily performed using Cloud Computing technologies [2, 10, 12, 13, 19]. The use of Cloud Computing for data management allows to incorporate the benefits of this technology (data replication, fault

tolerance, and resources scalability, among others) to FAS. There are two main reasons for using public Clouds in order to process and store WSNs data. The first one is the large volume of data generated by WSNs. As an example, in the region of Cuyo there are up to 170 000 cultivated hectares that can be instrumented with at least one sensor node per hectare. Therefore, there are 170 000 potential sensors that can generate data, which must be processed and stored in a proper infrastructure. The second reason is the traffic bottleneck from the WSNs to an isolated private data center.

Nowadays, the offer of Public Cloud Services is wide (Google Compute Cloud, Microsoft Azure, Amazon and others). One of the top providers included in that offer is Amazon. The Elastic Compute Cloud (EC2) toolkit service [4] provides different types of virtual machines (also called instances) for data processing and storage. Due to the wide range of instances offered by Amazon, there is a need for tools to identify which of these instances has better performance in terms of execution time and economic cost when processing frost prediction applications. In addition, these tools can provide information about a better way (single or cluster) to run these instances in order to minimize economic costs and execution times. In this paper, we propose a set of models constructed from empirical data that can be used to estimate the performance and economic costs of Amazon EC2 instances, applied for processing frost prevention applications. The proposed models allow to predict which is the instance that can process more sensor nodes in a certain time when the instance is working isolated. Although there are other costs associated with the use of Amazon EC2 instances – like the data transfer ones – the target of our study is the economic costs for WSNs data processing, taking into account that they are more relevant than the data transfer ones.

This paper is structured as follows: Section 2 introduces Frost Monitoring Systems based in WSNs. Next, Section 3 discusses related works, while Section 4 describes our application for frost prediction. Then Section 5 presents our performance estimation models for each Amazon EC2 instance and the methodology used to build them. Section 6 discusses the performance of Amazon EC2 instances in a typical use case; and Section 7, the accuracy of our models in this typical use case. Section 8 presents experiments about the instances' performance when they are executed in virtual clusters. Finally, Section 9 concludes this paper and details future works.

2 FROST ALARM SYSTEMS BASED ON WSNS

In this section, we provide an introduction of technologies used to perform data acquisition and management in FAS based on WSNs.

2.1 Data Acquisition with WSN

As shown in Figure 1, sensor nodes are composed of a micro-controller, memory, different sensors (e.g. temperature and humidity), battery and a radio module.

Sensor nodes can be interconnected into networks called WSNs, interacting among them. WSNs are used to study the environment and to acquire different variables related to weather, like temperature, humidity, pressure, solar radiation and others.

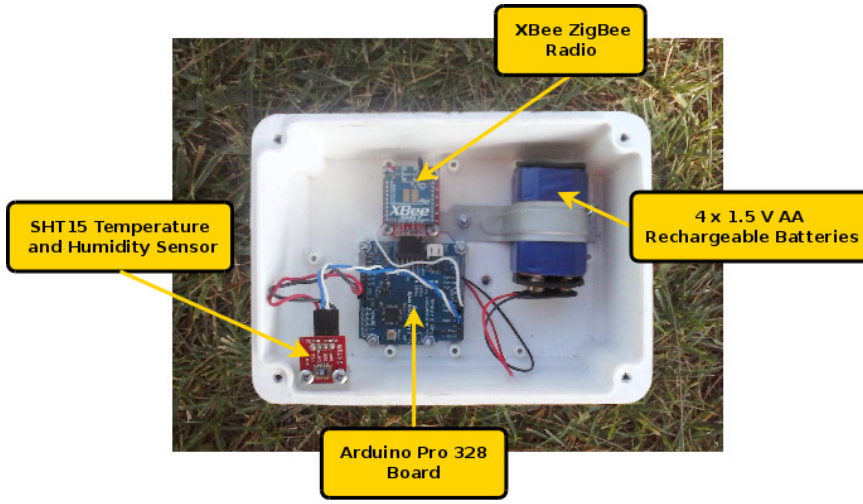


Figure 1. Sensor node based on Arduino Pro 328 board and ZigBee transceiver

Figure 2 illustrates a WSN for frost prediction that is deployed in a farm in the south of Mendoza, Argentina. In this WSN, data are acquired by source nodes and sent via ZigBee to a special node (known as sink). The sink node is connected to a personal computer (PC) or embedded system. The join of both sink node and PC is also called base station. This base station coordinates all operations of the WSN and transmits the information collected by sensor nodes to the final user, through the Internet.

WSN sensor nodes must meet requirements such as autonomy, low power consumption, low cost, robustness and reliability. Unlike traditional wireless networks, WSN nodes use communication protocols specifically designed for working with scarce energy and hardware resources. Some of these protocols are IEEE 802.15.4 [17], 6lowPAN [5] and ZigBee [29]. These ones are not compatible with TCP/IP networks, therefore, Base Station must include a gateway, which acts as translator between WSNs and the Internet communication protocols.

2.2 Data Management

Data collected by FAS through WSNs are used to provide solutions to frost prevention damage in crops [24, 27, 28]. Data management process starts when WSN data are sent to the Internet, then they are stored and processed into Public Clouds in order to obtain useful information for predicting frost. Next subsections cover the details of different technologies used to WSN data management.

Figure 2. WSN for frost prediction in Mendoza, Argentina

2.2.1 Traditional Technologies

Generally, the use of single machines like typical PCs or mainframes is enough to process low volumes of non-critical WSN data. A typical case of single machine use is when low volumes of data (in the order of kilobytes) are sent from the base station (which is deployed in the field) to remote machines. Data transmission can be achieved using different technologies like TCP sockets [11], RSS services [25] and others. Next, the outside-WSN machine stores the data and proceeds to run the processing application.

Although this technology is suitable for processing WSN data, it presents some problems for processing large volume of data, scaling to a large number of WSN nodes and ensuring availability 24 hours a day – 365 days a year. A possible solution to solve these issues is using powerful servers, mainframes and clusters in appropriate data center infrastructures. However, this solution generates prohibitive economic costs, at least for associations of small farmers. Since the use of traditional technologies is not always suitable, different authors propose the use of Cloud Computing services for processing WSN data [2, 10, 12, 13, 19].

2.2.2 Cloud Computing

Cloud Computing is a paradigm for application development and for the use of computing and storage resources [6]. Through the use of virtualization techniques and web services, hardware resources and applications can be dynamically provided to the user.

Foster et al. [7] define Cloud Computing as *"A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet"*. One of

the main advantages of Clouds is resources scalability. In this way, Clouds can solve the computational and storage requirements of the applications. Cloud providers offer their services according to three fundamental models which are described below.

Infrastructure as a Service (IaaS)

Where *"service"* means resource. Through infrastructure services, users can access to virtualized high-performance computing (HPC) resources (CPUs and storage devices, among others). The service provider delivers resources to a client in accordance with the specific requirement, such as CPU type and power, memory, storage, and operating system among other. Amazon EC2 [4] can be cited as an example of IaaS, being a set of Cloud services which allow to run applications on custom virtual machines (VM) deployed in Amazon data center servers. Amazon offers various types of VMs with different processing power and memory capability.

Platform as a Service (PaaS)

Where *"service"* means platform-level functionality. These services provide Application Programming Interfaces (APIs) and standard development kits (SDKs) in order to allow users to develop and implement their own applications for Clouds. Some examples of these platforms are Google App Engine [8] and Windows Azure [22].

Software as a Service (SaaS)

Where *"service"* means application. The SaaS Cloud providers deliver applications that can be accessed by an end user through an Internet connection and a standard web browser. Furthermore, the applications can be developed with Platform Services and executed with Infrastructure Services. Among other SaaS, we can cite Google Drive [9] and SAP Business Suite[1].

3 RELATED WORKS

In the last six years, different authors have proposed the use of Cloud technologies for managing WSNs resources. In [18] Lee et al. describe concepts of Cloud like virtualized resources, SaaS, pay-per-use price model; and apply them to create a Cloud infrastructure capable of integrating devices with sensing capabilities. This infrastructure is called Tangible Cloud and uses Amazon EC2 instances in order to process data from sensor nodes [19]. In the paper, the authors demonstrate that the platform solves (through resources scalability) the computational power requirements of environmental monitoring and modeling applications.

Another work proposed by Ahmed and Gregory [2] introduces an integration framework between WSN and Cloud Computing. The main objective of the proposed framework is to *"facilitate the shift of data from WSN to the Cloud Computing*

[1] `http://go.sap.com/solution/cloud.html`

environment". In addition, the authors suggest that the join of Cloud Computing with sensor networks allows the possibility of WSNs data storage in publics domains. Then different users and applications can access to the sensors' information, resulting in a better data usage.

Aneka is another platform to integrate WSN into Clouds [10]. Aneka uses resources of Private and Public Clouds in order to provide support to applications of smart environments, including health-care, transportation, monitoring and others.

Regarding the use of Clouds in agricultural environments, Hirafuji et al. [12] developed an Ambient Sensor Cloud System for High-throughput Phenotyping. This platform allows the storage and access to data collected by means of sensor nodes using Twitter Cloud services. The main goal of the system is to provide a simple and economical solution to solve the access and storage of large datasets from various sensor nodes. Hori et al. [13] present a commercial solution to storage and process WSNs data. The platform allows the integration with business management, production history, traceability and good agricultural practice systems provided as a SaaS model.

In [20] Mazurek and Fukuda present MASS, a library for multi-agent spatial simulation and parallelizing temperature prediction programs. The authors propose the use of MASS for the processing of sensor data *"on the fly"*. This library allows the parallelization of frost prediction models, which is necessary in order to minimize the execution times of frost prediction methods. MASS can work in Cloud Computing in multi-core instances and virtual clusters. The authors implement the library and a frost prediction method based on Artificial Neural Networks, Prediction Polynomials and Inverse Distance Weighting. Finally, they prove that MASS parallelism improves the performance of frost prediction methods by 55 %.

Dinh and Kim present in [21] an efficient interactive model designed for providing WSNs services to multiple applications on Sensor Clouds. The model proposed by Dinh and Kim has three main goals: providing on-demand WSN services to different applications at the same time, minimizing the number of requests sent to physical sensor nodes, and optimizing energy consumption on WSN nodes. This model could be applied in agriculture because one of the main issues of agricultural WSNs is the minimization of battery consumption on sensor nodes.

Based on the works studied in this section, it can be concluded that Cloud is a promising technology for solving the management and processing of data in WSN's based FAS. Although most of the studied works use Amazon EC2, to the best of our knowledge, there are no works oriented to model the performance and economic cost of Amazon EC2 instances when they are processing agricultural monitoring applications.

4 FROST PREDICTION APPLICATION

In this section, the frost prediction application (FPA) is introduced. The main objective of this application is to compute the minimum temperature reached during

the night. Then, according to this temperature value, frost occurrence on the farm can be predicted. The section is organized as follows. In Subsection 4.1, we present the method for frost prediction used in our application. Next, in Subsection 4.2 the application implementation is detailed.

4.1 Frost Prediction Method

The application was developed using the frost prediction method (FPM) belonging to Snyder and Melo-Abreu [26], which is based on Allen's equation [3]. The FPM predicts the minimum temperature that will occur in nights without both clouds and cold fronts. Therefore, it is only suitable to predict radiation frosts.

In order to carry out the prediction, the method takes temperature, humidity and dew point from days on which radiation frosts occurred. These days must belong to the month in which the prediction is performed (regardless the year). In this paper, we use data from a historical ten-year dataset. In addition, FPM needs the temperature, humidity and dew point recorded two hours after sunset in the prediction day.

Formally, the minimum temperature is calculated by the following linear regression (LR) equation:

$$T_p = s_T * T_o + s_D * D_0 + i \tag{1}$$

where T_p is the minimum temperature to be predicted. T_o is the temperature and D_0 dew point, both recorded the same day of the prediction two hours after sunset. Finally, i is the LR intercept, s_T temperature slope and s_D dew point slope. The values of s_T and i are calculated from the Equations (2) and (3), respectively.

$$s_T = \frac{\sum(T_{h0} - \bar{T}_{h0})(T_m - \bar{T}_m)}{\sum(T_{h0} - \bar{T}_{h0})^2}, \tag{2}$$

$$i = \frac{\sum T_m - s_T \sum T_{h0}}{n} \tag{3}$$

where T_{h0} are historical temperatures recorded two hours after sunset, T_m minimum temperatures that occurred in the night, and n is the number of historical data. Finally, $\bar{T}_{h0}$ and $\bar{T}_m$ account for the average data temperatures.

The slope s_D is calculated by using the Equation 4.

$$s_D = \frac{\sum(D_{h0} - \bar{D}_{h0})(R - \bar{R})}{\sum(D_{h0} - \bar{D}_{h0})^2} \tag{4}$$

where D_{h0} are historical dew points two hour after sunset and R the residuals. The parameters $\bar{D}_{h0}$ and $\bar{R}$ are the average of D_{h0} and R, respectively. Finally, the residual is calculated with the expression: $R = T_m - s_T * T_o + i$.

4.2 Application Implementation

In order to develop the FPA, we have implemented the Snyder and Melo-Abreu [26] FPM, using Java and MySQL. MySQL was used to store the data collected by sensor nodes and the results obtained after running the FPM. The application was executed using Amazon EC2 instances.

The integration of WSN data with Cloud infrastructures was performed with a WSN – Cloud integration platform called Sensor Cirrus [14, 15, 16]. Sensor Cirrus manages the WSN data using Cloud services and includes the developed FPA for data processing.

Figure 3 illustrates a scheme corresponding to the implemented FPA. The information collected by WSN sensors on the field is stored in a proper database, as it is seen in process (1). Next, in process (2), the application performs a query to catch a sample of fifty days in which the radiation frost has happened. This sample includes all the collected data (temperature, humidity, solar radiation, and wind speed, among others) by the WSNs. Then in process (3) the application retrieves only the FPM input data (T_o, D_o, etc.) from the sample of fifty days. Finally, in process (4) the FPM is executed on Amazon EC2 instances, providing the minimum temperature that would occur the following night (5). Process (4) can be performed using a single Amazon EC2 instance to process the FPM or using several instances working in parallel on a cluster of virtual machines.

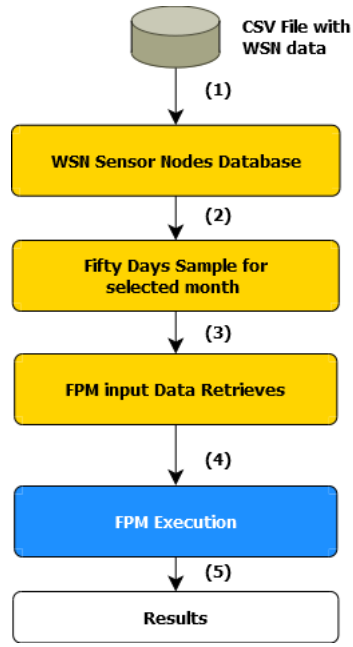


Figure 3. Frost prediction application

Figure 4 illustrates the processing of FPA on a virtual cluster. Within the FPA, processing tasks are distributed equitably on each virtual cluster node. Each job consists of processing a sensor node, therefore in each instance the same number of sensor nodes is processed. Hence, for processing 1 000 sensor nodes in a cluster of 4 instances, the master node of the cluster sends data from 250 sensor nodes to each slave node. Then those data are processed in order to determine the frost occurrence probability.

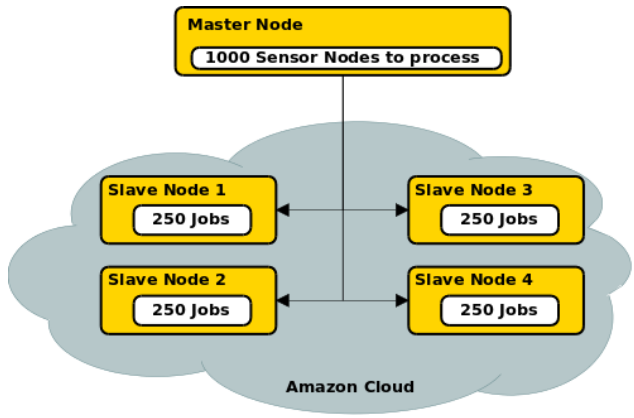


Figure 4. Processing of FPA on a cluster of Amazon EC2 instances

5 PERFORMANCE ESTIMATION MODELS

In this section, we present our models to estimate the performance of EC2 instances for processing FPA. The methodology used to construct the models is the following: first, we execute the FPA in each instance to obtain empirical results of performance metrics, specifically execution time and economic cost. Then we use polynomial expressions and empirical results in order to generate the performance models. Finally, we draw conclusions about the accuracy of the proposed models.

5.1 FPA Execution

The execution consists of running the FPA in different Amazon EC2 instances and measuring the execution time. In order to obtain the average value of the execution time, the procedure is repeated four times for different number of sensor nodes (from 10 to 1 000) in each instance. Finally, we use the average execution time and the Amazon's pricing list to calculate the economic cost required to execute the application.

Table 1 details the four instances to be modelled. Each row in Table 1 represents a different instance type, i.e., m1.small, m1.large, m1.xlarge and c3.xlarge. Each

column of the same table indicates the instance characteristics, i.e., number of virtual CPUs (vCPUs), Amazon EC2 Compute Unit (ECU), Memory (expressed in GBytes) and Instance Pricing. Regarding the Amazon's pricing model used in our work, we use the *"on demand"* pricing model. It is worthy of remark that in this paper we do not make an analysis of the accuracy of the minimum temperature predicted by the FPA. However, based on agronomists' experience, we can affirm that an error of $+/-1.5$ Celsius degrees is an acceptable error value to predict frost, and the used FPM meets this requirement.

Amazon EC2 Instance	vCPUs	ECU	Memory [GBytes]	Pricing *on demand* [US$]
m1.small	1	1	1.7	0.047
m1.large	2	4	7.5	0.190
m1.xlarge	4	8	15	0.379
c3.xlarge	4	14	7.5	0.239

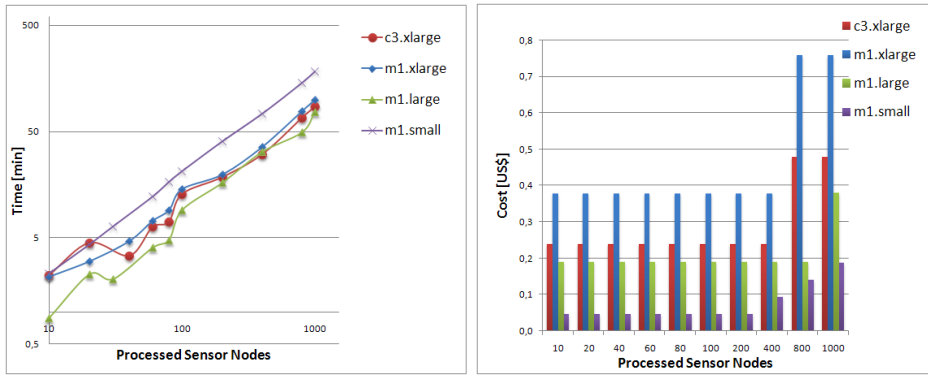
Table 1. Tested Amazon EC2 instances

The application execution allows to obtain empirical performance results in each EC2 instance. This execution was performed using the Screen window manager[2] in each tested Amazon EC2 instances. Screen multiplexes a physical terminal between several virtual terminals. Thanks to Screen, a process executed in a virtual terminal can be run completely and independently of any other terminal process executed in the same physical machine. In this work, we use Screen because it allows to avoid the influence of the SSH connection in the FPA execution, since we have noticed that the SSH connection affects the execution of the application. In some cases, SSH connection delays increase the execution time of frost prediction application; while in others, an interruption in SSH connection halts the application execution. Figure 5 illustrates the empirical results obtained from the FPA execution on each instance. Particularly, Figure 5 a) shows the execution time versus the number of processed sensor nodes, and Figure 5 b) details the economic cost versus the number of processed sensor nodes.

In Figure 5 a), it can be observed that m1.large is the instance that has achieved the shortest execution time for the frost prediction application. In other words, when processing up to 200 sensor nodes, the m1.large instance performance is noticeable. For more than that number of sensors, the m1.large performance becomes similar to the m1.xlarge and c3.xlarge performances. Furthermore, results show that for multiprocessor machines, like m1.large and c3.xlarge, the processing times decrease for 30 and 40 sensor nodes, respectively. Regarding the observed decrease, the one in m1.large instance is lower (about 9 % over the previous calculation) than the one in c3.xlarge instance (20 % compared to the previous point).

Analysis of hardware features of such instances (m1.large and c3.xlarge) shows that they have:

[2] `https://www.gnu.org/software/screen/`



a) Execution times b) Execution costs

Figure 5. Empirical results

1. two and four vCPUs, respectively, and
2. the same RAM memory (7.5 GBytes).

It can therefore be concluded that the decrease of processing times could be due to the load balancing between processors and the access to shared resources such as memory, and buses, among others.

Figure 5 b) shows the empirical economic costs. It can be verified that economic costs are the same from 10 to 400 nodes. Since Amazon sets the pricing of instances per hour of use, the cost is the same provided the processing time is less or equal than one hour. In a like manner, for processing times longer than one hour but shorter than two hours (for example 800 to 1000 nodes), the cost value is similarly doubled and so on.

5.2 Performance Estimation Models

In this subsection, we introduce the proposed models in order to estimate each instance performance when running the FPA in single way. These models are obtained through polynomials up to second degree of the form:

$$t = ax^2 + bx + c$$

where x is the number of sensor nodes processed and t is the estimated execution time. The values of the coefficients a, b and c for each scenario are shown in Table 2. These coefficients are obtained by statistical approximation using polynomials (up to second degree) applied to results obtained in Subsection 5.1.

In order to evaluate the proposed models we calculate the execution time and economic cost for each instance. Figure 6 shows a comparison of the empirical execution times and the ones predicted with our performance models for each tested

Amazon EC2 Instance	a	b	c
m1.small	1.84×10^{-6}	1.78×10^{-1}	1.80
m1.large	6.02×10^{-6}	6.50×10^{-2}	9.98×10^{-1}
m1.xlarge	1.40×10^{-5}	8.24×10^{-2}	2.25
c3.xlarge	1.66×10^{-5}	6.73×10^{-2}	2.59

Table 2. Coefficients of theoretical model of each instance

instance. Figure 7 illustrates the comparison between empirical and theoretical economic costs for the same instances.

Specifically, Figure 6 shows that execution times calculated through the proposed model differ in seconds or few minutes (depending on the instance) from the ones obtained through experiments. Then the proposed models are able predict the results with a reasonably good accuracy.

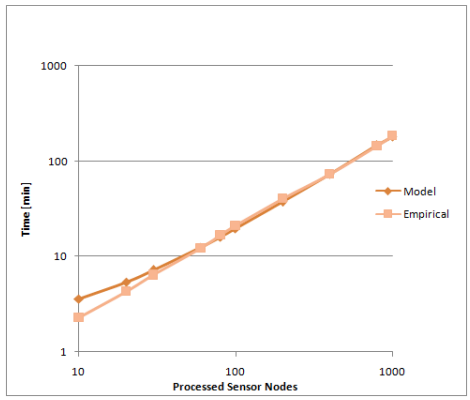
Regarding economic cost, a particular case is when execution times are close to an hour. In this situation, if the execution time calculated by the model is longer than one hour, the costs predicted by the model will be twice those empirical ones. This is because the price of EC2 instances is fixed per hour of use, as it is here in above explained. Likewise, if the model predicts less time than one hour, the corresponding calculated cost would be half of the empirical costs. However, when the proposed model is used, this situation does not happen, so we can say that the accuracy of the models regarding economic cost is suitable.

6 INSTANCE PERFORMANCE IN TYPICAL USE CASE

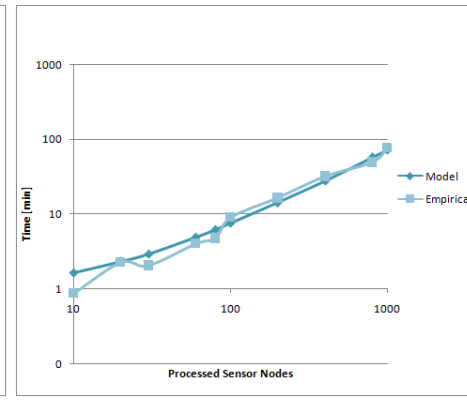
In order to select the instance with the best performance for running FPAs, we present in this section a comparison of a typical use case of frost prediction. The typical use case consists of WSNs deployed in different farms in the Province of Mendoza. The prediction is made for one day of July because this month belongs to the frost season, which begins in April and ends in October.

The FPA runs in a single instance of the Cloud and predicts the minimum temperature, allowing the agronomist's alert. Finally, the agronomist decides if the guard procedure against frosts must be conducted. Frost guard procedure consists in moving the staff to the farm and wait for the specialist's decision – based on data collected in real time – to activate proper defense systems, like heaters, sprinklers or wind turbines.

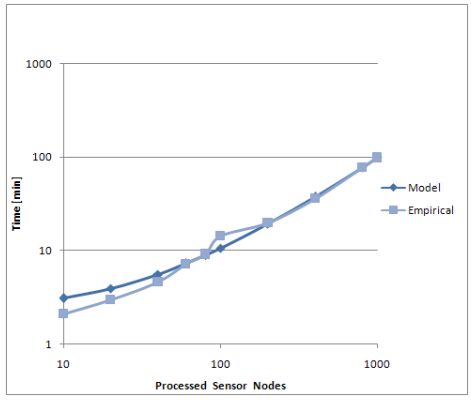
Data Processing time constraint is another aspect to take into consideration. Frost prediction application can only be launched after the measurement of T_0 temperature, which is obtained on the day of the prediction, specifically two hours after sunset. During the month of July in Mendoza province, the sunset takes place at 7.00 pm approximately, then T_0 temperature must be taken at 9.00 pm. Additionally, because of logistics, the frost defense system requires estate farm staff to be alerted before 10.00 pm. According to the above-mentioned reasons, the maximum execution time allowed for the application must be less or equal to one hour.



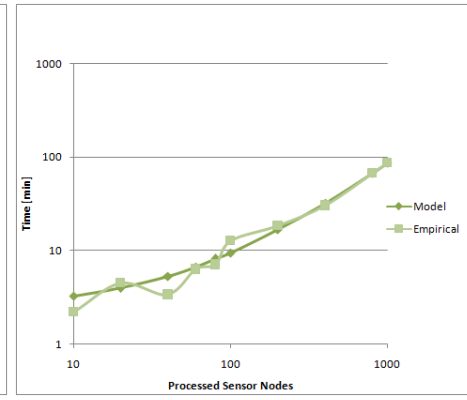
a) M1.small instance



b) M1.large instance



c) M1.xlarge instance



d) C3.xlarge instance

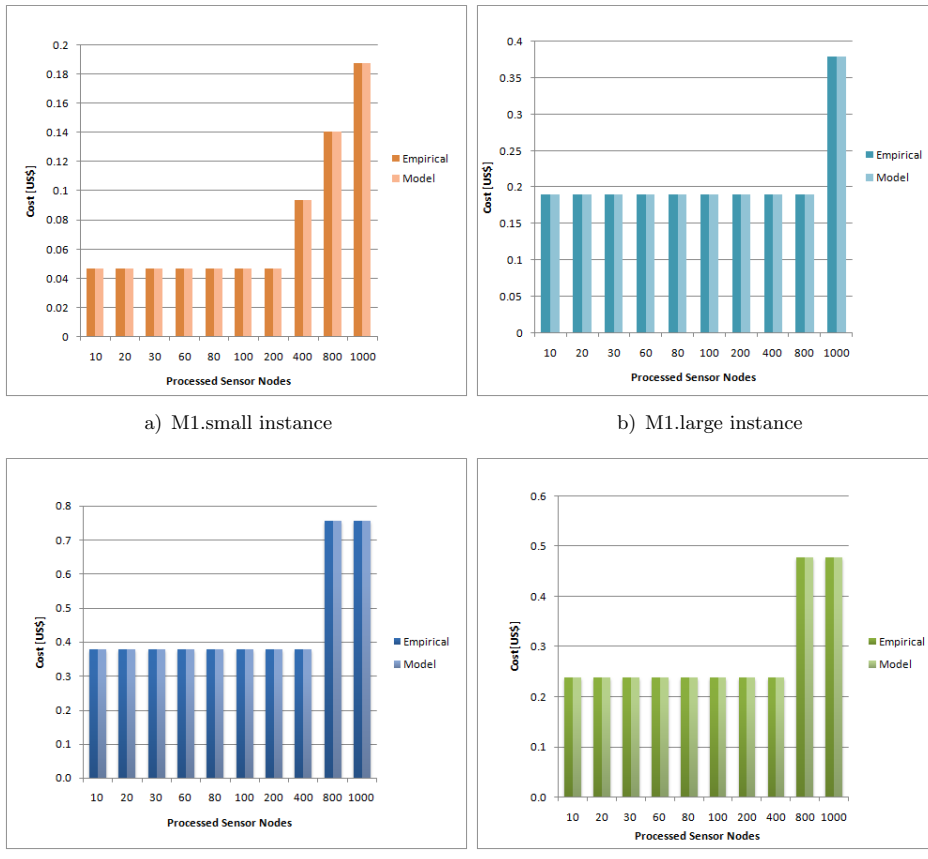
Figure 6. Execution times comparison in tested Amazon EC2 instances

Table 3 shows the execution cost and the number of nodes processed by each EC2 instance model, for one hour predicted by the proposed performance models.

Amazon EC2 Instance	Nodes	Economic Cost [US$]
m1.small	324	0.047
m1.large	841	0.190
m1.xlarge	632	0.379
c3.xlarge	722	0.239

Table 3. Processed sensor nodes in one hour predicted by performance models

Results demonstrate that the instance m1.large is the most suitable machine for this application type. The reason is because the m1.large is the machine that



a) M1.small instance

b) M1.large instance

c) M1.xlarge instance

d) C3.xlarge instance

Figure 7. Economic costs comparison in tested Amazon EC2 instances

can process the largest number of sensor nodes in one hour and its economic cost is smaller than those belonging to the m1.xlarge and c3.xlarge instances. Therefore, results presented in Table 3 suggest that m1.small instance can be used in parallel for processing more sensor nodes at a lower economic cost than what the m1.large machine is able to. Thus, the need to conduct more experiments in order to study the model accuracy of this instance in the typical use case becomes apparent.

7 ACCURACY OF MODELS FOR TYPICAL USE CASE

Execution times and costs are predicted by the proposed models through statistical approaches, hence it is necessary to perform experiments with the purpose of studying the accuracy of these models for a specific number of sensor nodes.

This section discusses the theoretical performance model accuracy of m1.small and m1.large instances for the typical use case presented in Section 6.

This section is organized as follows. Subsection 7.1 details the execution of the FPA for the number of sensor nodes that can be processed in each instance in one hour. Next, Subsection 7.2 presents a comparison between the execution times obtained in Subsection 7.1 and the ones predicted through theoretical models. The main objective of the comparison is to calculate the errors of the models for the typical use case. Then, based on these errors, we can correct the number of sensor nodes that can be processed in one hour. Finally, in Subsection 7.3 the FPA is executed for the corrected number of sensor nodes in each instance.

7.1 FPA Execution for Best Instances

This subsection details the results obtained through the execution of the FPA in m1.small and m1.large instances. These instances are selected because they can process more sensor nodes than the m1.xlarge and c3.xlarge instances, in an isolated way (m1.large) or parallel way (m1.small) at lower prices. The experiments are conducted in each instance (running single) for the number of sensor nodes that can be processed in one hour, predicted by theoretical performance models. In order to obtain the average value of execution time, the FPA is executed four times in each Amazon EC2 instance.

Table 4 presents the average execution times obtained by the execution of the FPA in each instance.

Amazon EC2 Instance	Nodes	Average Execution Time [min]
m1.small	324	48.77
m1.large	841	53.30

Table 4. Empirical execution times obtained for typical use case

7.2 Comparison of Models and Experimental Results

In order to determine the error of the proposed models, Table 5 shows a comparison between the empirical execution time and theoretical execution time obtained in the m1.small and m1.large instances. Column two details the number of sensor nodes processed in each instance, columns three and four show the execution times predicted with theoretical models and experiments, respectively. Finally, column five presents the error percentage observed between empirical results and theoretical models for each instance.

Experiments conducted in m1.small instance detailed in row two of Table 5 show that the error between the theoretical time and empirical time is 18.83 %. Results of experiments conducted in m1.large instance showed in row three of Table 5 present an error of 11.16 % between the empirical time and theoretical execution time.

Amazon EC2 Instance	Nodes	Theoretical Execution Time [min]	Empirical Execution Time [min]	Error [%]
m1.small	324	60	48.77	18.83
m1.large	841	60	53.30	11.16

Table 5. Comparison between empirical and predicted execution times for typical use case

7.3 Execution of FPA for Theoretical Corrected Results

Errors observed in our models allow the correction of the number of sensor nodes that can process each instance in one hour (typical use case). Table 6 shows the corrected number of sensor nodes that can be processed in m1.small and m1.large instances in a typical use case. Column two of Table 6 presents the predicted number of sensor nodes, Column three details the corrected number of sensor nodes and Column shows four errors used for correcting the number of sensor nodes.

Amazon EC2 Instance	Predicted Number of Sensor Nodes	Corrected Number of Sensor Nodes	Error [%]
m1.small	324	385	18.83
m1.large	841	934	11.16

Table 6. Correction of processed sensor nodes for typical use case

Once the number of sensor nodes to process in an hour is corrected, we conduct the execution of FPA for those values. Table 7 shows execution times obtained through experiments conducted in m1.small and m1.large instances for corrected sensor nodes.

Amazon EC2 Instance	Processed Sensor Nodes	Empirical Execution Times [min]
m1.small	385	54.88
m1.large	934	58.40

Table 7. Processed sensor nodes for typical use case

Row two of Table 7 presents the number of experiments conducted in m1.small instance. Results show that this machine can process 385 sensor nodes in 54.88 minutes, reason why we can affirm that theoretical models have been fixed correctly for typical use case in m1.small instance.

Row three of Table 7 details the execution times when FPA is executed considering the new number of nodes in m1.large instance. Like the m1.small instance, the obtained execution time (58.40 min) has validated the correction made in the theoretical model.

8 FPA EXECUTION IN VIRTUAL CLUSTERS

Experiments conducted in Subsection 7.3 determine the Amazon EC2 instance that can process more sensor nodes when working in an isolated way (m1.large). However, the above mentioned experiments suggest that m1.small instances can be used in parallel through a virtual cluster for processing larger number of sensor nodes at lower cost than the instance m1.large. In this section, we perform experiments in order to determine how many sensor nodes can be processed in m1.small instances when they are executed in parallel. The execution is conducted considering a virtual cluster for which the hour price is lower than a m1.large instance hour price.

The experimental methodology is the following: first, we consider the number of sensor nodes that can be processed by each instance in one hour, working in an isolated way. Then we implement a virtual cluster composed of the number of virtual machines whose total cost is less than 0.19 US$ (an m1.large instance's hour price). Next, the FPA is processed in each node of the virtual cluster for the number of sensor nodes that can process each slave node (instance) in an hour. Finally, the run time is measured in each execution. The experiment is achieved four times in the virtual cluster in order to obtain the average execution time.

Table 8 shows the number of sensor nodes processed using the m1.small instance on a virtual cluster. Second and third columns show the number of processed sensor nodes and slave nodes (instances) used in parallel way, respectively. Fourth column indicates the recorded execution time, and finally fifth column shows the resulting economic cost of FPA executed in the virtual cluster.

Amazon EC2 Instance	Sensor Nodes	Instances Executed in Parallel	Execution Time [min]	Economic Cost [US$]
m1.small	1 540	4	57	0.188

Table 8. Execution times of virtual cluster in a typical use case

Making a comparison with the results obtained for m1.large instance from Table 7, the results in Table 8 indicate that m1.small cluster can process 606 more sensor nodes than one m1.large instance in one hour at lower economic cost. This result is obtained using four m1.small instances in parallel way. Furthermore, the recorded average execution time is 57 minutes, which fulfills the time constraint requirement defined in the typical use case.

9 CONCLUSIONS AND FUTURE WORKS

In this paper, we have studied the use of Amazon EC2 instances for frost prediction. In the first place, we have presented an application developed to predict the occurrence of frost based on data collected on-field by Wireless Sensor Networks. This application has been used to generate performance models of different Amazon EC2 instances when they were applied to process frost prediction applications. The metrics used to evaluate the performance were execution time and economic cost.

In order to generate the models, we have conducted experiments in four test scenarios. Each scenario has corresponded to a different Amazon EC2 instance. The obtained model of each instance was compared with empirical data of the frost prediction application executions. From the results, we have concluded that the proposed models were suitable to estimate both the execution time and the economic cost. However, the proposed models have presented some problems when they have been used to predict economic cost and execution time in a specific number of sensor nodes. That was why we have performed more tests in a typical application case, allowing us to determine the models' errors. These experiments have been only performed for the instances with better performance working in a single (m1.large) or parallel way (m1.small).

Experiments conducted in a typical use case have showed that models' error were 18.83 % and 11.16 % for m1.small instance and m1.large instance, respectively. Once the error of each model has been determined, the number of sensor nodes that could be processed in the typical use case has been corrected. Then we have performed tests to validate if the corrected number of sensor nodes were fulfilling the time constraints of typical use case. Results have showed that the models were fixed correctly, then m1.small and m1.large instances were able to process 385 and 934 sensor nodes, respectively, in an hour.

A typical use case has been used to determine which instance working in an isolated way was more suitable for processing frost prediction applications. Results have showed that the m1.small instance and m1.large were the correct instances to use for WSNs made up by 140–385 and by 386–934 sensor nodes, respectively.

While the m1.xlarge and c3.xlarge were the instances with the best performance, we have not observed important differences in the performance when comparing them to the other tested instances. Moreover, if we also consider the m1.xlarge and c3.xlarge high costs, their use is not recommended for this type of applications.

The execution of frost prediction application in individual machines have allowed us to determine that for the case of WSNs – made up by more than 934 sensor nodes – multiple EC2 instances were to be used in parallel way to run the application. Hence, we have performed an experiment on a virtual cluster. The virtual cluster was composed of four m1.small instances. These instances have been selected because they were able to process more than 934 sensor nodes running in parallel way at a cost lower than the best instance running in single way (m1.large) in the typical use case.

Results of experiments on virtual clusters have showed that the m1.large cluster can process 1 540 sensor nodes in one hour. Consequently, taking into account the above mentioned number of WSNs sensor nodes, we can affirm that the m1.large cluster is more suitable because it allows the processing of data from more sensor nodes than one single m1.large instance.

In conclusion, this paper demonstrates that second degree polynomials are a simple and suitable way to estimate the performance of Amazon's EC2 instances. Regarding future works: first, we are going to improve the frost prediction application for parallel execution. Next, we will continue the validation of our models by study-

ing the processing of the frost prediction application on virtual clusters scheduled with algorithms based on meta-heuristics and managed by specific Cloud tools like Star Cluster. The purpose of these future experiments is to extend the proposed models to estimate not only how many machines are needed to optimize the execution time/economic cost relationship for frost prediction applications but also how these machines should be managed.

Acknowledgments

The authors acknowledge the financial support provided by the Argentinean Agency for R & D activities (ANPCyT) through projects PAE-PID 146 and PICT-2012-2731, and to the Spanish National Plan for Research, Development and Innovation for financial support provided by the project TIN2012-31518 (ServiceCloud). Lucas Iacono also thanks the Consejo Nacional de Investigaciones Científicas y Técnicas of Argentina (CONICET) for the postdoctoral scholarship, as well as Norberto Faraz for the source code provided for the frost prediction application, and finally we are grateful for the Bec.AR program.

REFERENCES

[1] ur Rehman, A.—Abbasi, A. Z.—Islam, N.—Shaikh, Z. A.: A Review of Wireless Sensors and Networks' Applications in Agriculture. Computer Standards and Interfaces, Vol. 36, 2014, No. 2, pp. 263–270.

[2] Ahmed, K.—Gregory, M.: Integrating Wireless Sensor Networks with Cloud Computing. Seventh International Conference on Mobile Ad-Hoc and Sensor Networks (2011 MSN), IEEE, 2011, pp. 364–366, doi: 10.1109/MSN.2011.86.

[3] Allen, C. C.: A Simplified Equation for Minimum Temperature Prediction. Monthly Weather Review, Vol. 85, 1957, No. 119–120.

[4] Amazon EC2: Amazon Web Services. 2012, available at: `http://aws.amazon.com/es/ec2/`.

[5] Bormann, C.—Mulligan, G.—Arkko, J.—Townsley, M.—Schumacher, C.: IPv6 over Low Power WPAN (6lowpan). IETF Working Group, 2006, available at: `http://datatracker.ietf.org/wg/6lowpan/charter/`.

[6] Buyya, R.—Yeo, C.—Venugopal, S.—Broberg, J.—Brandic, I.: Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5^{th} Utility. Future Generation Computer Systems. Vol. 25, 2009, No. 6, pp. 599–616.

[7] Foster, I.—Zhao, Y.—Raicu, I.—Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop 2008 (GCE '08), IEEE, 2008, pp. 1–10, doi: 10.1109/GCE.2008.4738445.

[8] Google: Google App Engine. 2013, available at: `https://developers.google.com/appengine/`.

[9] Google: Google Docs. 2013, available at: `https://drive.google.com/`.

[10] Gubbi, J.—Buyya, R.—Marusic, S.—Palaniswami, M.: Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. Future Generation Computer Systems, Vol. 29, 2013, No. 7, pp. 1645–1660.

[11] Higuera, J. E.—Polo, J.: IEEE 1451 Standard in 6LoWPAN Sensor Networks Using a Compact Physical-Layer Transducer Electronic Datasheet. IEEE Transactions on Instrumentation and Measurement, Vol. 60, 2011, No. 8, pp. 2751–2758, doi: 10.1109/TIM.2011.2129990.

[12] Hirafuji, M.—Yoichi, H.—Kiura, T.—Matsumoto, K.—Fukatsu, T.—Tanaka, O. et al.: Creating High-Performance/Low-Cost Ambient Sensor Cloud System Using OpenFS (Open Field Server) for High-Throughput Phenotyping. Proceedings of 2011 SICE Annual Conference (2011 SICE), IEEE, 2011, pp. 2090–2092.

[13] Hori, M.—Kawashima E.—Yamazaki, T.: Application of Cloud Computing to Agriculture and Prospects in Other Fields. Fujitsu Scientific and Technical Journal, Vol. 46, 2010, No. 4, pp. 446–454.

[14] Iacono, L.: Acceso Remoto a Redes de Sensores Inalámbricas Mediante Tecnologías de Computación Distribuída. Thesis Proposal, Facultad de Ingeniería, Universidad de Mendoza, 2013 (in Spanish).

[15] Iacono, L.: Sensor Cirrus. June 2014, `https://sites.google.com/site/sensorcirrus/`.

[16] Iacono, L.—García Garino, C.—Marianetti, O.—Párraga, C.: Wireless Sensor Networks: A Software as a Service Approach. In: García Garino, C., Printista, M. (Eds.): Prospective and Ongoing Projects, VI Latin American Symposium on High Performance Computing (HPCLatAm 2013), Mendoza, Argentina, 2013, pp. 184–195.

[17] IEEE: IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003), 2006, pp. 1–305.

[18] Lee, K.—Hughes, D.: System Architecture Directions for Tangible Cloud Computing. International Workshop on Information Security and Applications (IWISA 2010), Vol. 25, Qinhuangdao, China, 2010, doi: 10.1109/CDEE.2010.57.

[19] Lee, K.—Murray, D.—Hughes, D.—Joosen, W.: Extending Sensor Networks into the Cloud Using Amazon Web Services. 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA), IEEE, 2010, pp. 1–7, doi: 10.1109/NESEA.2010.5678063.

[20] Mazurek, E.—Fukuda, M.: A Parallelization of Orchard Temperature Predicting Programs. 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim), IEEE, 2011, pp. 179–184, doi: 10.1109/PACRIM.2011.6032889.

[21] Dinh, T.—Kim, Y.: An Efficient Interactive Model for On-Demand Sensing-as-a-Services of Sensor-Cloud. Sensors (Basel), Vol. 16, 2016, No. 7, Art. No. 992, 28 pp.

[22] Microsoft: Windows Azure. 2013, available at: `http://www.windowsazure.com/es-es/`.

[23] Oliveira, L. M. L.—Rodrigues, J. J. P. C.: Wireless Sensor Networks: A Survey on Environmental Monitoring. Journal of Communications, Vol. 6, 2011, No. 2, pp. 143–151, doi: 10.4304/jcm.6.2.143-151.

[24] Pierce, F. J.—Elliott, T. V.: Regional and On-Farm Wireless Sensor Networks for Agricultural Systems in Eastern Washington. Computers and Electronics in Agriculture, Vol. 61, 2008, No. 1, pp. 32–43, doi: 10.1016/j.compag.2007.05.007.

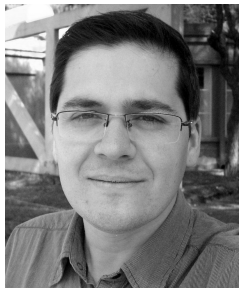
[25] Reddy, S.—Chen, G.—Fulkerson, B.—Kim, S. J.—Park, U.—Yau, N.—Cho, J.—Hansen, M.—Heidemann, J.: Sensor-Internet Share and Search: Enabling Collaboration of Citizen Scientists. Proceedings of the ACM Workshop on Data Sharing and Interoperability on the World-Wide Sensor Web, 2007, pp. 11–16.

[26] Snyder, R. L.—de Melo-Abreu, J. P.: Frost Protection: Fundamentals, Practice and Economics. Vol. 1, Food and Agriculture Organization of the United Nations (FAO), 2005.

[27] Yoo, S. E.—Kim, J. E.—Kim, T.—Ahn, S.—Sung, J.—Kim, D.: A2S: Automated Agriculture System Based on WSN. IEEE International Symposium on Consumer Electronics (ISCE 2007), 2007, pp. 1–5.

[28] Zerger, A.—Viscarra Rossel, R. A.—Swain, D. L.—Wark, T.—Handcock, R. N.—Doerr, V. A. J.—Bishop-Hurley, G. J.—Doerr, E. D.—Gibbons, P. G.—Lobsey, C.: Environmental Sensor Networks for Vegetation, Animal and Soil Sciences. International Journal of Applied Earth Observation and Geoinformation, Vol. 12, 2010, No. 5, pp. 303–316, doi: 10.1016/j.jag.2010.05.001.

[29] ZigBee Alliance: ZigBee Specification. ZigBee Document 053474r13, 2006, pp. 344–346.



Lucas E. Iacono received his Engineering degree in electronics and electrical from Universidad de Mendoza, Argentina, in 2007 and his Ph.D. from Universidad de Mendoza in 2015. Currently, he is Posdoctoral Fellow at the Consejo Nacional de Investigaciones Científicas y Técnicas of Argentina (CONICET). His research interests include internet of things technologies, particularly WSNs and cloud computing applied to frost prediction and environmental monitoring.

José Luis Vázquez Poletti is Tenure-Track Assistant Professor at UCM, from where he received his Ph.D. in computer architecture. His research interests include high-performance computing, cloud computing, and grid technology, focusing on their application to real life problems.

Carlos García Garino graduated in engineering at the University of Buenos Aires, Argentina in 1978 and received his Ph.D. degree from Universidad Politécnica de Cataluña, Barcelona, Spain in 1993. Currently he is Full Professor at the School of Engineering and Head of the ITIC Research Institute, Universidad Nacional de Cuyo, Argentina. His research interests include computational mechanics, computer networks, and distributed computing. He has more than 50 papers published in scientific journals and proceedings of international conferences carried out in Argentina, Brazil, Chile, Canada, Spain, France, Portugal, Belgium and Japan.

Ignacio Martín Llorente is co-founder and Director of OpenNebula, and Full Professor at UCM. He received his Ph.D. in computer architecture from UCM and an executive MBA from the Instituto de Empresa. His research interests include high-performance computing, virtualization, cloud computing, and grid technology. He is a senior member of IEEE.