UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MASTER IN COMPUTER SCIENCE
EVALUATION OF MONITORING TOOLS FOR CLOUD COMPUTING ENVIRONMENTS

By: Sandoval Briceño, Yosandra del Valle
Tutor: Mariela Curiel.
Co-tutor: Bastian Koller.
March 2013

ABSTRACT

Within the progress that has had the world of distributed computing, stands out the emergence of new paradigms as Cloud computing. Cloud computing is a model that integrates a number of computational resources to offer them as a service to users on demand, based on technologies such as grid computing, virtualization and Web 2.0. Monitoring in this computational model is a very complex issue, due to the particular characteristics present in this type of platform, such as abstraction of virtualized resources and loss of control over information by users, by relying on external providers. On the other hand, research on the topic is still scarce. This research work, that has been done within the context of the EU project OPTIMIS (Optimized Infrastructure Services), is about the problem of monitoring cloud environments, particularly for private clouds. The main objective of this research is to evaluate existing monitoring tools that can be used in cloud computing and propose an architecture for monitoring in such environments, which can be implemented in any private cloud and in particular in the monitoring component of OPTIMIS project. For the evaluation of the tools a three-stage methodology is proposed, which determines whether the tools fulfill the requirements for monitoring cloud environments. After studying a significant set of tools, we can conclude that Lagios and Nagios are two open source tools best suited for monitoring private clouds in IaaS (Infrastructure as a Service) and PaaS (Platform as Service) levels, respectively. With these two tools a monitoring architecture for OPTIMIS was proposed.

Keywords: cloud computing, services, virtualization, monitoring, monitoring tools, monitoring architecture, OPTIMIS.
ÍNDICE GENERAL

ABSTRACT ................................................................................................................................. 1

1. Introduction ................................................................................................................................. 3
   1.1. Problem definition .................................................................................................................. 3
   1.2. Objectives .............................................................................................................................. 3

2. Monitoring on Cloud Computing .............................................................................................. 3

3. Evaluation Methodology ............................................................................................................ 5
   3.1. Phase I: Basic Requirements ............................................................................................... 5
   3.2. Phase II: Analysis of Requirements for Monitoring on Clouds .............................................. 6
   3.3. Phase III: Evaluation of Resources consumption ................................................................. 7
       3.3.1. Aim of the Experiments .................................................................................................. 7
       3.3.2. Experimentation Platform .............................................................................................. 8
       3.3.3. Monitors for measuring the overhead at the Server ....................................................... 9
       3.3.4. Resources consumption ............................................................................................... 9
           3.3.4.1. CPU consumption .................................................................................................. 9
           3.3.4.2. Memory consumption .......................................................................................... 11
           3.3.4.3. Network Consumption ......................................................................................... 12

   4.1. Components of the Monitoring Architecture ....................................................................... 14
       4.1.1. Components for each Computing Node ........................................................................... 15
       4.1.2. Master Node Components ........................................................................................... 16
   4.2. Proposed Architecture and OPTIMIS ................................................................................... 17

5. Conclusions .............................................................................................................................. 17

6. Future Work ............................................................................................................................... 18

References ........................................................................................................................................ 19
1. Introduction

1.1. Problem definition

Cloud computing [1]–[3] is a new paradigm in which both new technologies and existing ones converge to offer as services all the capabilities of a computing system to different users [2]. These services can be accessed from any device with internet connection independently of their physical location. This new computing model has many benefits, posing some challenges such as its monitoring. The main difficulty is for developers and administrators [4], in connection with the abstraction level of the infrastructure (abstraction/unification of the resources is done through virtualization [5]). Most of the tools and APIs available used for monitoring resources have been developed for Grid [6] and Clusters [7] infrastructures or are not open source, such as monitoring tools created specifically for major public cloud providers (Amazon, Google y Microsoft, among others). For this reason, it is an interesting research problem, to evaluate in detail which tools are more suitable for this new computational model.

1.2. Objectives

Evaluate different monitoring tools available for distributed Systems, in order to determine their usefulness in cloud computing environments and give a recommendation about how they could be included (in whole or taking certain components) in a monitoring architecture for this type of environments, in particular in the OPTIMIS monitoring infrastructure.

2. Monitoring on Cloud Computing

Monitoring takes a significant role when it comes to ensuring the quality of service in cloud computing [8], since it provides indicators to verify the fulfilment of the SLAs [9]. The use of monitors also facilitates the development of mechanisms to increase the resource utilization adaptively; to detect problems on services; to optimize application deployment and
to discover the usage pattern of numerous end users, among other advantages. The monitoring is useful to user as well as infrastructure providers, allowing them to verify the performance of their own applications and/or services through the review of the state of the virtual machines.

Some challenges to be overcome in the developing of a monitoring system on cloud computing environments are presented below:

- **Abstraction/unification of the virtualized resources:** for developers and cloud administrators, the resources abstracted/unified through virtualization and some other levels of encapsulation make difficult the following up of the problems at their sources. Furthermore, the limited information supply by the provider about the infrastructure performance and resource utilisation may not have the necessary level of detail to understand what the current status of resources is.

- **Energy consumption:** this criterion is very important nowadays due to the introduction of new legal regulations related to environmental impact [10]. It is pursued to improve the utilization of hardware resources, to reduce costs, space and energy without a performance penalty on the cloud.

- **Monitoring Tools for specific environments:** most monitoring tools that have been developed for cloud environments have been created specifically for major public cloud providers (Amazon, Google y Microsoft, among others).

Since monitoring plays an important role in cloud computing environments, one of the objectives of this work is to find the adequate tools that can be inserted later in the monitoring component of OPTIMIS project [11], [12], considering the challenges or difficulties that are present in these environments.

The main goal of OPTIMIS is developing Tools to support the complete life cycle of IT services on cloud environments (development, deployment and runtime management) in order to optimize the resources used. OPTIMIS considers different types of scenarios based on both public and private clouds (“cloud bursting”, “multi-cloud” y “cloud federation”).
3. Evaluation Methodology

For the evaluation process, we propose three stages. The purpose of each phase is described below:

**Phase I:** consists in a preliminary assessment of several monitoring tools used in distributed systems which must meet certain basic selection requirements. The evaluation was made based primarily on the review of the documentation tools.

**Phase II:** consists in the evaluation in more detail of the tools selected on phase I, taking into account this time requirements more specific that any monitoring tool for cloud environments must meet, as well as requirements for platform (PaaS) and infrastructure (IaaS) level according to [13] y [14]. In this phase each tool was installed on a small cloud environment in order to test the main required characteristics.

**Phase III:** consists in taking the tools that are selected in the second phase in order to do a detailed empirical study about the resource consumption of each one. The idea is testing in a cloud environment a little more realistic with respect to their size. With the data obtained we fitted statistical models for explaining the influence of certain parameters on the resource consumption of the tools.

3.1. Phase I: Basic Requirements

In this first phase we established some basic requirements considering: particular requirements of OPTIMIS project; ideal characteristics in any monitoring tool for distributed systems and certain basic characteristics desirable for clouds. These requirements are listed below:

1. Must be open source, to be deployed on any cloud environment.
2. Must allow monitoring for virtual environments.
3. Must be capable of measuring performance of both applications and resources.
4. Must not limit the number of devices to be monitored.
5. Must require few hardware components for operation.
After reviewing the documentation tools (for: Nagios [15], Lattice [16], HypericHQ [17], Zenoss [18], GroundWork [19], Zabbix [20] and OpenNMS [21]) and evaluated which of them meets all the basic requirements required, we concluded that the tools Nagios, Lattice, Zenoss and HypericHQ look more suitable in this first phase for monitoring clouds environments, because they meet all the basic requirements established, as showed in detail in [22].

3.2. Phase II: Analysis of Requirements for Monitoring on Clouds

As shown on [22], the four tools mentioned above fulfil the characteristics of scalability, elasticity, adaptability and migration to be taken into account for monitoring clouds environments, according to [14]. Federation was not checked due to the limitation of not being able to tests the migration between different domains, thus remains as future work examining it.

Related to IaaS level, Lattice and Nagios are the most appropriate tools because they offer more options to retrieve performance metrics of different physical and virtual resources (Hosts); they are scalable and exhibit low utilization in most resources, except network consumption. HypericHQ could be also appropriate if the infrastructure where it is deployed has no limitations on the amount of memory available. On PaaS level, Nagios and HypericHQ are more suitable because they allow the collection and display of the monitoring information. They have an intuitive web interface, easy to use and from which the tool can be managed and all the information collected observed. They offer also a variety of reports online and as historical data. In Table 3-1 is the summary of the results for the second phase of the methodology. From the table, we can conclude quickly that the open source tool more complete for levels PaaS and IaaS is Nagios because it meets most of the requirements for both levels, has less impact on the use of two of the three resources considered (CPU and Memory) and does not limit the administration of monitoring parameters in comparison with HypericHQ. Additionally, Lattice is recommended. Although it does not meet most of the requirements for PaaS level, it is very suitable for IaaS level because it has a low impact on
resources utilization and it is developed for monitoring dynamic environments as the cloud. With Lattice and Nagios was the last phase of the evaluation.

<table>
<thead>
<tr>
<th>Main Characteristics</th>
<th>Nagios</th>
<th>HypericHQ</th>
<th>Lattice</th>
<th>Zenoss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Elasticity</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Migration</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Adaptability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Autonomy</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**IaaS Level**

| Support several types of resources | X | - | X | - |
| Impact on resource utilization    | X | - | X | - |

**PaaS Level**

| Data Collection of the infrastructure | X | X | X | X |
| Analysis of data collected          | X | X | - | X |
| GUIs                               | X | X | - | - |
| Report Generation                   | X | - | - | - |
| Viewing of historical data          | X | - | - | X |

**Legend:**
- X: meets expectations
- -: Does no meet expectations

| Table 3-1: Comparison of the tools selected on phase II. |

### 3.3. Phase III: Evaluation of Resources consumption

#### 3.3.1. Aim of the Experiments

The aim of this last phase is to evaluate the influence that the following parameters have on the resource consumption at the Server: The Monitoring Tool; the Number of Virtual Machines on which the Tools are executed simultaneously; and the changes in virtual machines’ configuration (as represented in the number of CPUs). The metrics to assess, or response variables, are related to the CPU utilisation (%CPU at the Server: %usr, user mode), memory use (% Mem at Server: %memused) and network consume (total bytes received by the Server). The configuration that the Cloud supports in determined by the number of virtual
machines running and by the number on CPUs in each virtual machine. The factors to move during the experimentation, as well as their levels, are the following:

- **Tool Type**: the type of monitoring tool being used. The levels of this factor are: None, Lattice and Nagios. By including “None”, it is possible to observe the infrastructure’s behaviour without the load of any monitoring tool, and then compare it with the behaviour under the load of the two Tools.

- **#VM**: Number of virtual machines in which the Tools’ clients (sensors/agents) are simultaneously executed. The levels of this factor are: 1, 5, 10, 15 and 20 virtual machines.

- **#CPU**: Number of CPUs that each virtual machine has. The levels of this factor are: 1, 2 and 4 CPUs.

### 3.3.2. Experimentation Platform

The tests were performed on a Cloud environment, which hosted up to 20 virtual machines for each level of factor “Monitoring Tool” with the same hardware configuration at the physical machine (Adelgunde), and the Server (Wagadugu) which was in charge of collecting the measurements sent by the sensors/agents of each Tool. The experimentation platform can be observed at Figure 3-1. In order to execute the instances of the virtual machines, the Xen [23] hypervisor was employed.

![Figure 3-1. Schematic view of the different VMs combinations used for each level of factor “Tool Type”](image-url)
3.3.3. Monitors for measuring the overhead at the Server

In order to take measures at the server, two tools were used: Sar [24], for collecting the CPU and Memory consumption data; and Wireshark [25], for measuring the network consumption. Additionally, scripts were developed and CRONs\(^1\) were used for executing the tools mentioned above (Sar and Wireshark) in synchronisation with the Tools’ agents.

3.3.4. Resources consumption

For the data analysis presented below, a co-variance model was used. In the following, indexes \(i = 1, 2, 3\) are assigned the Tool Types “None”, “Lattice” and “Nagios”, respectively. The indexes \(j = 1, 2, 3\) correspond to Configuration Types “1 CPU”, “2 CPUs” and “4 CPUs”, respectively. For the sake of parameter estimation, the effects corresponding to “None” (No Tool) and “1 CPU” were considered as baselines, that is to say, were given a value of zero\(^2\).

3.3.4.1. CPU consumption

In order to analyse the CPU consumption, at Figure 3-2 the behaviour of the tools in terms of the CPU percentage usage at the Server. The values of all three repetitions for each treatment are shown. The following co-variance model provides a good representation of the data collected for response variable \(y = \% \text{ CPU at the Server}:\)

\[
y_{ij,k} = \mu + \gamma_i + \lambda_j + \beta y x + \epsilon_{ij,k}
\]

Where,

\(y_{ij,k} = k\)-th observation of response variable, for Tool Type \(i\) and configuration \(\#CPU\) \(j\).

\(^1\) In the Unix system, cron is a background regular process administrator (daemon), which executes processes or scripts at regular intervals.

\(^2\) This assignment is standard when estimating the effects of categorical explanatory variables (factors), and receives the name “corner point constraint”. 

\[ \mu = \text{Global mean.} \]

\[ \gamma_i = \text{Effect due to Tool Type } i. \]

\[ \lambda_j = \text{Effect due to #CPU per virtual machine.} \]

\[ x = \text{Number of virtual machines (#VM).} \]

\[ \beta_{ij} = \text{Slope of the line expressing } y \text{ as a function of } x, \text{ for Tool Type } i \text{ and #CPU level } j. \]

\[ \epsilon_{ij,k} = \text{Error term.} \]

---

**Figure 3-2.** Plot of %CPU consumption at the Server. All three repetitions are shown for each treatment.

The model's fit is very good, as expressed in an R-squared\(^3\) statistics of 0.9971 (a perfect fit would amount to 1). On the basis of the fitted model, we can say that the rates of change \( \beta_{ij} \), of the CPU consumption with respect to #VM, are not the same for each combination of Tool Type \( i \) and configuration type \( j \), as shown in Table 3-2.

Observing Table 3-2, we conclude that Lattice exhibits a smaller increase in consumption per additional VM, for all types of configuration (#CPU). Factor #CPU shows no

---

\(^3\) R-ajustado: Es una medida relativa de qué tanto las variables de predicción incluidas en el modelo explican la variación de las observaciones. Sus valores se encuentran entre 0 y 1.
evidence of having influence on the CPU consumption rates of change, for Tool type “Lattice”.

<table>
<thead>
<tr>
<th>Tool Type</th>
<th>1 (1CPU)</th>
<th>2 (2CPUs)</th>
<th>3 (4CPUs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Ninguna)</td>
<td>0</td>
<td>0.0197</td>
<td>0.0254</td>
</tr>
<tr>
<td>2 (Lattice)</td>
<td>0.0929</td>
<td>0.0806</td>
<td>0.0773</td>
</tr>
<tr>
<td>3 (Nagios)</td>
<td>0.3359</td>
<td>0.3278</td>
<td>0.3393</td>
</tr>
</tbody>
</table>

Table 3-2. Rate of change of CPU consumption as a function of the number of Virtual Machines created.

The value in can be assumed equal to zero.

3.3.4.2. Memory consumption

Figure 3-3 helps us to have an idea of the kinds of interrelations among the response variable “Memory consumption” and the explanatory variables. All three repetitions for each treatment are plotted. The following co-variance model is useful for representing the data collected, and explain response variable $y = \% Mem$ at the Server:

$$y_{ij} = \mu + \gamma_i + \beta_i x + \epsilon_{ij}$$

Where,

$y_{ij} = j$-th response under Tool Type $i$.

$\mu = $ Global mean.

$\gamma_i = $ Effect due to Tool $i$.

$x = $ Number of virtual machines (#VM).

$\beta_i = $ Regression coefficient of %Mem on #VM, for Tool type $i$.

$\epsilon_{ij} = $ Error.

The fit of the model is very good, as expressed in an R-squared of 0.8585. On the basis of the fitted models, we can remark: for Lattice, no increase in %Mem at the Server was observed when increasing variable #VM. However, for Nagios such an increase is present, with a non-neglectable estimated value of 0.0126% additional consumption per VM, as shown in Table 3-3.
To summarize, Lattice produces in general a higher Memory consumption, which is consistent with the fact that Lattice is a tool implemented in Java, while Nagios is implemented in C. However, this relative advantage of Nagios vanishes as we increase variable #VM.

\[
\begin{array}{|c|c|}
\hline
\text{1 (Ninguna)} & 0 \\
\hline
\text{2 (Lattice)} & 0 \\
\text{3 (Nagios)} & 0.0126 \\
\hline
\end{array}
\]

Table 3-3. Tasa de crecimiento del consumo en Mem en el servidor como función del número de máquinas virtuales creadas. El valor en rojo indica que se puede asumir igual a cero.

### 3.3.4.3. Network Consumption

Finally, Figure 3-4 helps visualize the relation of response variable \( y = \) “Total bytes received at the Server”. It presents all three repetitions per treatment. As for response variables “CPU consumption” and “Memory Consumption”, it is possible to notice a linear relation between the response and variable #VM. The following co-variance model is proposed for explaining the response \( y = \) “Total bytes received at the Server”:

\[
y_{y,k} = \beta_y x + \varepsilon_{y,k}
\]

Where,
$y_{i,j,k} = k-th$ response observation, for Tool Type $i$ and type of CPU configuration $j$.

$x = \text{Number of virtual machines (#VM)}.$

$\beta_{ij} = \text{Slope of the line expressing } y \text{ as a function of } x, \text{ for Tool Type } i \text{ and } \#CPU \text{ configuration type } j.$

$\epsilon_{i,j,k} = \text{Error}.$

Figure 3-4. Plot of Network consumption at the Server. All three repetitions are shown for each treatment.

The R-squared for this model is approximately 1. The following remarks can be drawn from the model: The rate of change of “Total bytes” with respect to #VM is always greater for Nagios, and it stays practically unaltered by the type of configuration (#CPU). On the other hand, this rate of change varies for Lattice, when varying the configuration type (#CPU). The rates of change of “Total bytes received by the Server” with respect to #VM are shown in Table 3-4.

As a general conclusion, we have observed that both the %CPU and the Network consumptions at the Server are higher for Nagios. Lattice exhibits a moderately higher Memory consumption for virtual machines with 2 and 4 CPUs. Lattice presents an increase of the total bytes sent by the network when varying the architecture of the virtual machines (#CPU), but this total amount of bytes is always below those sent by Nagios.
This concludes the Tools overhead evaluation. In the following section we propose a monitoring architecture for Cloud Environments, in which the two tools that fulfil the pre-established requirements are deployed.


Considering the results of this investigation about the performance of monitoring tools (Nagios and Lattice) and taking into consideration the work done in [26], [27], we propose an architecture for monitoring in cloud computing environments.

4.1. Components of the Monitoring Architecture

This architecture is based on the “producer-consumer” concept, widely applied in monitoring systems. The Producers of monitoring information (Agents), collect data from sensors and the consumers (Collectors) read the monitoring data, as shown in Figure 4-1. This architecture consists of two components: Computing node and Master node. The Computing node consists of physical and virtual resources deployed on the cloud, while the Master node is responsible for managing the monitoring information generated on the Cloud by the Computing node. A cloud environment would have a master node and multiple Computing nodes. The master node can be made redundant in order to maintain the ability to monitor the computing nodes in case of failure, by creating one or several mirrors of it. The following describes the elements and functions that perform each of these components.

<table>
<thead>
<tr>
<th></th>
<th>1 (1CPU)</th>
<th>2 (2CPUs)</th>
<th>3 (4CPUs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Ninguna)</td>
<td>109.77</td>
<td>89.92</td>
<td>104.10</td>
</tr>
<tr>
<td>2 (Lattice)</td>
<td>31140</td>
<td>35340</td>
<td>43740</td>
</tr>
<tr>
<td>3 (Nagios)</td>
<td>150979.94</td>
<td>151444</td>
<td>151628.11</td>
</tr>
</tbody>
</table>

Table 3-4. Rate of change of total of bytes received at the Server as a function of the number of Virtual Machines created.
4.1.1. Components for each Computing Node

Each computing node is in charge of collecting performance information of resources that are running through different components. The components are listed below:

1. **Agents for monitoring specific components**: they are in charge of collecting monitoring data of physical and virtual infrastructure.
   - **Virtual Monitor Agent (VMA)**: running on virtual machines to get performance information from services and applications that are executing on a particular virtual machine. At this level it is proposed to use Lattice sensors. Lattice does not impose a high load on the running VMs; it can be easily adapted to monitor different parameters of the applications and presents a low network consumption in comparison with Nagios.
   - **Physical Monitor Agent (PMA)**: running on physical machine to monitor the underlying virtual infrastructure of the Computing nodes and basic resource consumption of the physical host. It is suggested to use also Lattice’s sensors at this level. Lattice has the facility to get information about the performance of the hypervisor,
as well as to monitor the CPU, memory and network usage of each virtual and physical machine deployed.

- **Supervisor Agent (SA):** running on the physical machine to monitor the state of the platform. At this level it is proposed to use Nagios plug-ins. Nagios can send alerts when it fails and recover critical infrastructure components. Nagios makes use of a plug-in system for testing specific infrastructure capabilities. Many plug-ins already exist, but if necessary new plug-ins can be developed in order to monitor further capabilities, which add scalability and flexibility to the monitoring system.

2. **Collector:** This component must retrieve periodically monitoring information gathered by agents of both deployed tools and send the received information to the Database.

3. **Local Database:** used to store the information collected by the Collector from the different agents.

4. **Data Reader:** used to read the monitoring data from the local database and send it to the Data receiver located in the Master node.

### 4.1.2. Master Node Components

The Master Node, whose function is to collect and process information from several computing nodes for later display via the GUI. The Master Node consists of the following components:

1. **Data Receiver:** receives the resources monitoring data from each Computing node. The data received is processed (formatted and categorized) to be sent to the Global Database and it is also determined which part of this data will be shown on real time in the GUI.

2. **Data Reader:** used to read the monitoring data from the Global Database and send it to the GUI, according to the requests configured in the GUI.

3. **Global Database:** to store resources monitoring data which are received by the Master node.

4. **GUI:** through it the system administrator can view the monitoring information of the whole cloud environment and can set the configuration monitors’ parameters.
4.2. Proposed Architecture and OPTIMIS

To implement the components of the proposed architecture in the Cloud environment provided by OPTIMIS project, an image for the virtual machine has to be created which contains Lattice’s sensors with all the parameters (e.g. measurement time) and monitors (e.g. CPU usage, memory usage, network usage, etc) that we need to use. This image will be used as base when deploying a new virtual machine on the cloud. In the physical machines Lattice sensors and Nagios plug-ins should be installed, the first to monitor the resource consumption of the virtual machine (hypervisor) and the second to monitor the general state of the platform. Finally, a machine (physical or virtual) is required that operates as a server that is in charged of executing the Collector of the monitoring data from the deployed agents. In the evaluation process of the feasibility of the proposed architecture, performed with the Active Reviews for Intermediate Design (ARID [28], [29]), it was determined that it is feasible to be implemented on a cloud because many of the components related with the Computing node are already implemented (using the features available for each tool). Still to be implemented are: all components of the Master node and the communication components between the Master node and the Computing node(s). The communication proposed between computing nodes and master node can be easily implemented through scripts (bash) o JAVA in order to not to impose additional requirements to the infrastructure.

5. Conclusions

The aim of this work was to evaluate monitoring tools for cloud environments, in particular private ones. For this end, a three-stage methodology was proposed. The idea was to evaluate a large number of tools, using to the full the documentation available in the first two stages, to verify essential requirements, and leaving the evaluation related to the performance, which is dependent on the platform, for the last stage. Finally, we propose a monitoring architecture that includes the tools analyzed. The analysis leads to the following conclusions:

- The requirements to be met by the tools used in a monitoring system are based on the type of business carried out by the organization and the particular characteristics that
they wish to cover. For this reason, in this work we evaluated the tool in terms of the requirements for the service levels IaaS and PaaS and the overhead that the tools will introduce on a cloud environment.

- There is no single solution to deploy a monitoring system different elements should be taken into consideration, which depend on the resources available and the needs of the organization that will use the monitoring system. In this work we considered to make use of existing tools, performing a detailed analysis to identify the optimal way in which they can be used on a cloud environment.

- After applying the evaluation methodology, it is concluded that the tools Nagios and Lattice could be implemented in a monitoring system for the levels considered (IaaS and PaaS).

- From the statistical analysis of resource consumption generated by each tool we got information for the deployment of these tools on a cloud environment, particularly in the monitoring architecture proposed in this work. It was concluded: since Nagios has high consumption of resources compared with Lattice, it is recommended for PaaS level, leaving Lattice for IaaS level.

- Considering the results of the evaluation methodology, a monitoring architecture was proposed that can be implemented in any private cloud and in particular in the monitoring component of OPTIMIS project. Therefore, it is suggested to use both tools combined into a hybrid solution that would use, on the one hand, the Nagios plug-in architecture for adding scalability and flexibility to the monitoring system, and on the other hand, the advantages of low impact of Lattice on monitored resources.

6. Future Work

The work done can be extended and supplemented in many ways. Here are some tasks that could be done:

- Evaluate the performance of the tools deployed among various organizations, to determine whether it is feasible to implement the proposed monitoring infrastructure in different domains.

- Consider incorporating SaaS level in the methodology.
• In relation to resource consumption a further study with respect to memory consumption by tools must be done.

• Additionally, it is very important to study the overhead of the tools considering other factors, among which are: a greater number of machines (physical and virtual), number of users requesting services from the cloud, other virtualization platforms, etc.

• In order to evaluate the results of the methodology, it is important to implement the monitoring architecture for cloud environments and evaluate the global solution proposed here with the combination of both tools.

• Finally, it would be interesting also to gather metrics on the energy consumption and temperature. This can be done by incorporating hardware sensors. This way, a better knowledge of infrastructure performance can be obtained from every possible perspective.

References


