Monitoring "Grid-Cloud" model using Complex Event Processing (CEP)

Hassina LADOUR
Networks Division
Research Center on Scientific
and Technical Information (CERIST)
Algiers, Algeria
Email: ladour@dtri.cerist.dz

Aouaouche El-Maouhab
Networks Division
Research Center on Scientific
and Technical Information (CERIST)
Algiers, Algeria
Email: elmaouhab@dtri.cerist.dz

***Abstract*—Cloud Computing has been widely used by Customers to outsource IT services and business applications managed by a third-party called Cloud Provider in order to reduce their equipment management costs. The academic sector have also taken notice and are investigating how Cloud technology can provide results more quickly and at lower cost in order to deploy Grid services which benefit already from distributed environment like Grid Computing.**

Indeed, Grid Computing environment supports the large-scale distribution of all resources but faces problem of resource's increasing demands and resource's management caused by the growing of scientific computing applications. In this case, taking full advantage of Cloud technologies like enabling the use of Cloud-provided resources and exploiting infrastructure Cloud management system can simplify Grid service's resource deployment and allocation. However, it is necessary to have an adaptable monitoring solution to operate, analyze and adjust Cloud's resources to the specifications and requirements of Grid services including "Middleware" which allows them to run.

For this purpose, the work presented in this paper consists on the proposal of an integration "Grid-Cloud" model to deploy Grid services on Cloud IaaS-resource provided. And secondly, complete this model by developing a Framework based on new monitoring approach with "Complex Event Processing (CEP)".

***Index Terms*—Cloud Computing, Grid Computing, Infrastructure as Service (IaaS), Cloud Monitoring, Complex Event Processing (CEP).**

I. Introduction

The service model "*Infrastructure as a Service (IaaS)*" in Cloud Computing offers a flexible and easy way to rent virtual resources on demand. The main advantages of this model are flexibility and quickness to provide Cloud customers the computing resources they require as "pay-as-you-go[1]" model, providing cost based on actual consumption and leaving the management of the underlying Cloud infrastructure to the Cloud provider. Given these characteristics of "IaaS Clouds", the scientific community was also interested on the possibilities to benefit from the flexibility that can offer this type of model in the area of virtualization, outsourcing and dynamic resources allocation to deploy scientific services, research applications and high performance computing in lower cost.

[1]Principle of consumption payment in Cloud Computing.

Currently Grid Computing is used by scientific communities to provide strength calculation required for scientific applications. They can provide seemingly one single infrastructure of various resource providers, generally heterogeneous, allowing several projects to work together [1]. However the current Grid Computing's architecture faces two major problems: increasing resource demand caused by scientific applications calculation's growing demand and infrastructure resource allocation problem [2].

Therefore, researchers are focused on the possibility to integrate Grids and Clouds [3], in order to deploy, allocate or release resources dynamically and to react with the changes and demands caused by the quantity and size of jobs launched on Grids [4]. The researchers began by clarifying the differences and similarities of both paradigms Grid and Cloud [5], [6]. Then, they tried to introduce to the Grids, technologies used by Clouds namely virtualization [7]. Even more recently, they have embarked on a new idea which is to apply a model or an economic approach to the Grid systems to automatically allocate resources requested by Grid users from public Cloud infrastructures [4]. Other researchers see the Grids and Clouds systems as complementary and propose solutions of Grid services deployment over virtualized Cloud resources [8].

However, an another challenge is added to the Cloud Monitoring system to best manage the Cloud resources allocated to the integrated Grid services such as accessing and interpreting real-time monitored information. In this type of real-time monitoring, information is considered like event streams that require advanced processing mechanisms to find more sophisticated data flow models. In this context, Complex Event Processing (CEP) [9] represents a new technology for describing approaches to handle with event streams and produce other derivatives events. CEP technical are used to continuously collect, process, analyze data streams in real-time and produce results without delay, even when data arrive at very high rates [10].

In this paper we describe our proposal model "Grid-Cloud" that provides more flexible Grid services by exploiting dynamic allocation of Cloud IaaS-provided resources taking into account the specifications of Grid Computing like the Middleware. We complete the model by using CEP to build

978-1-5090-4580-8/16/$31.00

a flexible Framework for real-time monitoring of Cloud IaaS-provided resources used to run Grid services and for enabling an effective management of services, physical and virtual resources. The main purpose of this solution is to show that CEP technology can provide expressive query constructions to support complex scenarios and to discover relevant monitoring information.

II. Background

This section briefly reviews some Cloud, Grid and CEP fundamentals.

A. Grid Computing

A Grid Computing is an architecture allowing multiple communities to share computer resources. The central idea introduced by Grid Computing is the concept of Virtual Organization (VO) which means a group of users, institutions or organizations that have a common purpose in their using of Grid resources and services allowing simplified access to data and resources shared by the organization [11], [12].
To ensure execution and role of each component in the Grid and the collaboration between the different resources, a set of APIs, protocols and softwares are installed. This set is called "Middleware". Several Middlewares exist and are used by many Grid projects, most known of them are "Globus Toolkit (GT)" [13], "gLite" [14] and more recently "EMI[2]" [15].

B. Cloud Computing

Cloud Computing is an infrastructure of a set of virtualized resources (networks, servers, storage, applications) providing services to Customers according to the model "pay-per-use" in what guarantees are offered by the infrastructure provider by means of a "Service Level Agreement (SLA)" [16], [17], [18], [19].
The Cloud is consisting of three service models depending on the provided services: "Infrastructure as a Service (IaaS)" if provided services are VM machines and storage, "Platform as a Service (PaaS)" if provided services are application developing platforms, and "Service as a Service (SaaS)" if provided services are software applications.
According to its deployment, it can be public if it is accessible by general public usually through Internet, private if it is accessible by organization's members owning the Cloud, or hybrid if it is a combination of several public and/or private Clouds.

C. Cloud Monitoring

Cloud Monitoring is of major importance for both sides, Cloud Provider and Cloud Customer. It helps the Provider to control and manage the physical infrastructure by providing capacity and resource management. It indicates the Customer application performance in order to provide capacity and resource planning. It is also related to troubleshooting and security management [20].
The survey [21] gave a detailed study on Cloud Monitoring by analyzing and discussing the properties and the issues and describing current monitoring platforms for Cloud Computing (commercial and open source). According to this survey, Cloud Monitoring can be deployed in many aspects following Cloud's architecture and features (Layers, Abstraction levels, Service Models, Tests and metrics). The survey has also defined some properties required by Cloud Monitoring system and some related research issues. Most important are: *Scalability* (the capacity to cope with a large number of probes), *Elasticity* (the capacity to cope with dynamic changes of monitored entities), *Adaptability* (the capacity to adapt to varying computational and network loads in order not to be invasive), *Timeliness* (Availability of events on time for their intended use).

D. CEP

"Complex Event Processing (CEP)[3]" is a new technology introduced in the 90s by Professor David Luckham[4] to extract information in a complex message-based system in real-time using Event Patterns [9]. Events that logically correspond to Event Patterns are aggregated into Complex Events. A Complex Event is an abstraction of other events, meaning it occurs only if a considerable amount of other events occurs [10]. Event patterns are formal functions that can be applied to events, to derive other Complex Events [10].
Several open-source and commercial CEP solutions exist for offering CEP functionalities based on SQL-like processing languages to express patterns and rules. "Esper" [22] is an open-source library written in Java. It has the distinction to express Event Pattern's rules through "Event Processing Language (EPL) [23]" with built-in capabilities for analysis over time and event correlation.

III. "Grid-Cloud" integration model

Two approaches are proposed in the literature concerning the combining of both paradigms Grid and Cloud :

- **"Grid-over-Cloud"**: consists of using Cloud infrastructure system to construct and manage Grid infrastructure resources and services. For this purpose, "StratusLab project" [24] aims to provide a complete open Cloud distribution to supply a Grid Service Platform. "GaaS" [25] is another solution using this approach to provide Grid resources as PaaS model in the Cloud.
- **"Cloud-over-Grid"**: consists of using Grid Computing resources or Grid Middleware to construct Cloud infrastructure Systems. An implementation example of this approach is "Clever" [26], a solution that installs a specialized Cloud management system and a Virtual Machine Monitor over Grid Worker Nodes. "WNoDeS" [27] from INFN uses as well this approach. It uses the submission of Grid jobs for launching VM images to execute these jobs. "Nimbus[5]", is another example which consists of using

[2]European Middleware Initiative. http://www.eu-emi.eu/

[3]http://www.complexevents.com/

[4]http://www-ee.stanford.edu/~luckham/

[5]http://www.nimbusproject.org/

Globus toolkit Middelware to implement the Nimbus IaaS system.

We propose in this paper a "Grid-Cloud" model based on an integration approach to deploy Grid services on IaaS Cloud resources. Our model takes the approach "Grid-over-Cloud", treating Grid Services as a "Virtualized Platform" running over an IaaS Cloud infrastructure and follows the "Dynamic Grid Services" scenario published by EMI that describes the use of Cloud infrastructure to supply Grid services as service models (SaaS, PaaS, IaaS). The "Grid-Cloud" model provides an PaaS service model to Grid end-users with the possibility to access Virtualised Grid computing resources and services via both Grid and Cloud interfaces. It provides Grid site managers an IaaS service model exploiting fully the dynamic nature of IaaS Cloud System for flexible management of computational resources to existing Grid Sites running as "Virtualized Platform" taking into account the Grid Middleware.
The architecture of the "Grid-Cloud" model is composed of several abstraction layers to describe the supply of virtualized Grid platform by Cloud IaaS-provided resources. The three abstraction layers are (Figure 1):

A. "Infrastructure Layer"

The Infrastructure Layer is composed itself of three abstraction levels:

- **Storage level**: represents the entire infrastructure data definition such as VM and network templates, Cloud users information, hosts information stored in a dedicated database. In addition, a "VM Image Repository" is used representing a backup repertory to store VM disk images issues from virtual appliances to instantiate virtual machines.
- **Compute level**: this level is the basis of the Cloud infrastructure layer. It includes physical nodes to host virtual machines. Hypervisor is installed on each physical node to allow virtual machine management. Cloud VM Manager ensures resources allocation, hypervisor interfacing and VMs scheduling. Cloud APIs allow Cloud resource's access.
- **Network level**: the virtualized Grid platform should be visible to end-users outside of the organization running the Cloud infrastructure. For this purpose a public network linking Grid services on virtual machines is needed. Unlike the physical infrastructure which is visible to expert users (Cloud and Grid managers), a private network is sufficient.

B. "Service Layer"

We consider Grid services based on "EMI" Middleware in our model "Grid-Cloud". The Grid services architecture provided by "EMI" is organized in a distributed manner with centralized management while the computation functions are distributed among several sites (Figure 2). Those provided are [15]: Management services (User Interface(UI), Virtual Organization Management System(VOMS), Information System(IS), Wokload Management System(WMS)), Computing services (Computing Element(CE), Worker Nodes(WNs)) and Storage services (Storage Element(SE)). In our model, each of these services is instantiated on a Virtual Machine constituting together the virtualised Grid platform. Computing services can be instantiated many times depending on demand allowing the extension of new Grid sites to existing Grid sites.
To instantiate a new Grid service (eg. CE), we use a Virtual Appliance to create a VM disk image. Each of the VM disk image created is stored in the "VM image Repository" and then used on demand by the Grid manager to instantiate a new virtual machine .
Grid services based on "EMI" implement their own monitoring and accounting solutions to monitor services, jobs and Grid applications like "DGAS[6]", "HLRMon[7]" and more recently "APEL" [28], [29]. In our model, these existing monitoring solutions are re-integrated and accessible to Grid users on the virtualized Grid platform.

C. "User Layer"

We identify four different actors on the model:

- **Grid end-user**: is the end user of the Grid, he submits jobs on Worker Nodes and exploits the storage management capabilities of the Grid services. A Grid end-user belongs to one or more Virtual Organizations and must have a certificate to access Grid resources.
- **VO manager**: is an expert user responsible of a Virtual Organization within the Grid. His role is to provide all softwares needed for community application's implementation and negotiate for a number of Grid computing resources to make them available to community's members (VO users).
- **Grid Site manager**: is the user that has privileges to operate the Grid sites by providing the necessary technical support. From Cloud point of view, the Grid site manager is a regular user of IaaS Cloud, he accesses through the Cloud API (IaaS service model) to allocate resources (Virtual machines) requested by the VO Manager and negotiated with the Cloud Manager.
- **Cloud Manager**: is the supervisor who manages the entire set of material and physical resources of the IaaS Cloud used to maintain Grid services.

IV. The "MonCep" Framework

Several monitoring solutions exist for both type of Cloud Computing (commercial and open source) implement both high and low-level monitoring. The majority of commercial Clouds don't provide information on the low-level monitoring system used. At high level, they provide their users monitoring softwares to monitor applications, services and virtualized infrastructures [21].

Monitoring solutions in open source Clouds are based either on integrated set of APIs and tools or either on external open source monitoring platforms (eg. Nagios [30], Ganglia [31],

[6]Distribute Grid Accounting System. http://www.to.infn.it/dgas/

[7]http://www.italianGrid.it/Grid_operations/tools/accounting_portal/HLRmon

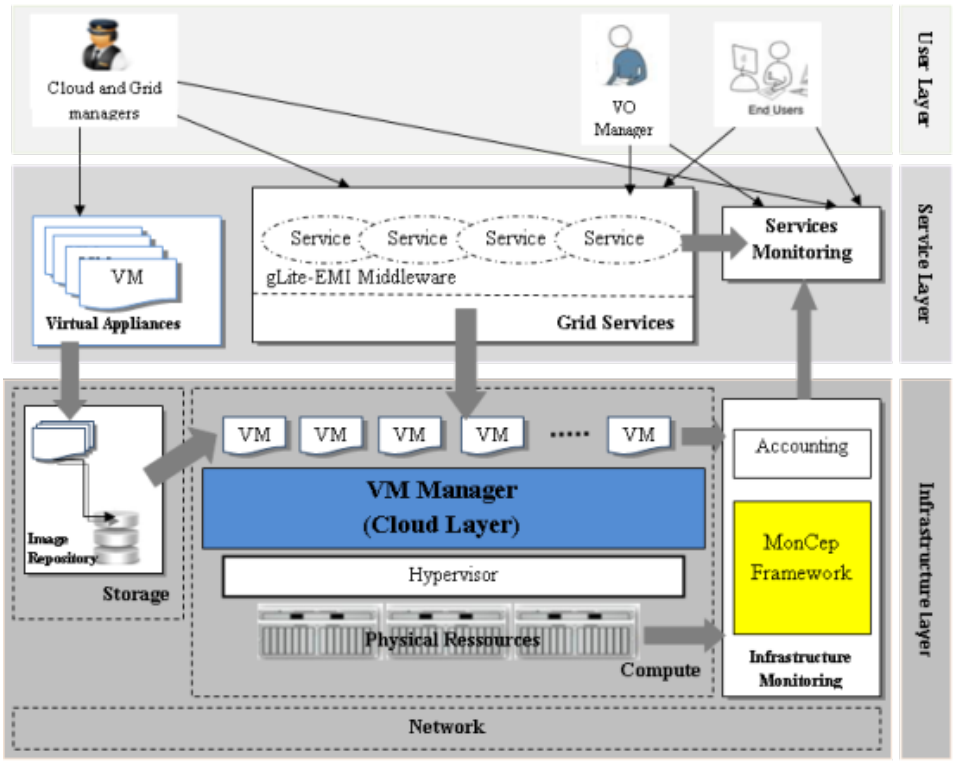

Fig. 1. The "Grid-Cloud" Model architecture

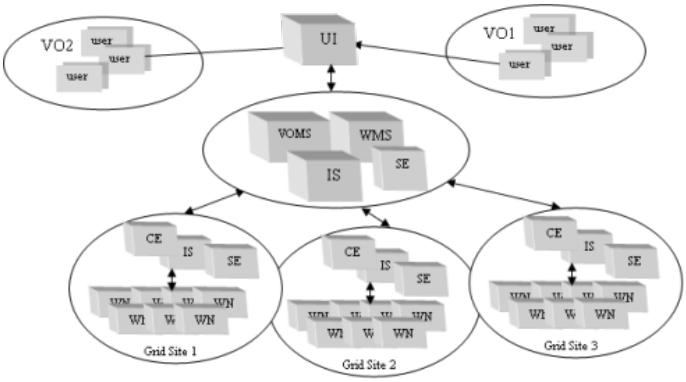

Fig. 2. EMI Grid architecture

etc.) or either on research projects solutions like [32], [33], [34], [35], [36] that give the ability to have a holistic view of the Cloud and to monitor at the same time, physical resources, virtual resources and applications. Most of these monitoring solutions have properties required by a Cloud Monitoring system and meet less or more the needs [20]. Nevertheless, most of the properties and their related issues have to be studied and implemented as improving Cloud Monitoring system with new techniques that provide information correlation and very fine grained measures [21].

We propose a monitoring Framework solution integrated on the infrastructure level of the model "Grid-Cloud" to monitor resource consumption of the virtualized Grid platform deployed on IaaS Cloud resources. The solution allows to provide correlation and real-time processing of monitored information according to the Grid Services requirements on the Cloud IaaS-provided resources. It is composed of modular components (Figure 3) based on CEP technology to provide filtering, correlation and fast processing of huge volume of data in real-time. By Applying CEP Engine to a monitoring system, it enables real-time processing of monitoring information flow including: Filtering relevant information, Information correlation from various sources and Using a powerful query language (eg. Event Processing Language (EPL)) to express sophisticated patterns. For these characteristics, the "MonCep" modules use CEP engines for processing monitoring events at different levels:

A. The "MonCep Sensor" module (Information gathering and filtering level)

Located as close as possible to the source of monitoring metrics (on each physical host on Cloud Infrastructure), the **"MonCep Sensor"** module allows gathering physical and virtual resource metrics by specific monitoring agents then filtering theme as XML events by using EPL queries to express filter and aggregation Patterns defined on a CEP Engine. Filter and aggregation patterns allow passing incoming events that meet a given condition of interest and filter out the others which do not meet this same condition. Conditions of interest are defined by administrators as needed. For example, the Cloud Manager may be interested in specific events whose values symbolizing monitoring metrics that

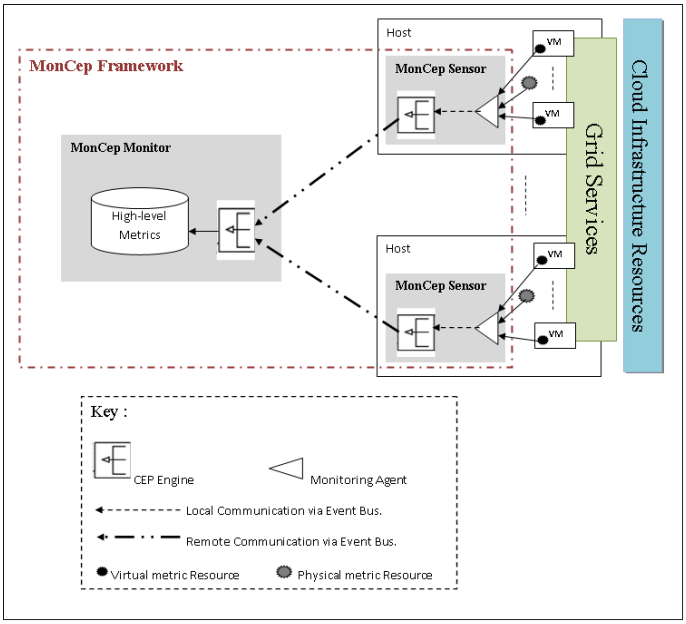

Fig. 3. The "MonCep" Framework architecture

exceed or fall a given threshold. Let us consider an example of *HostInfoMsg* stream of metrics measurement gathered from a host on the Cloud Infrastructure by the agent. The following EPL request that express Filter pattern, extract the Load Average from the stream of metrics in a 10 minute time window and report it once every 30 seconds but only when it falls below 1.6%:

```
Select HCUE.load_five from
HostInfoMsg.win:time(10 min) where
HCUE.load_five>    1.6 output last every 30
seconds
```

B. The "MonCep Monitor" module (information correlation level)

Located on a central site, the **"MonCep Monitor"** module allows the correlation of events generated by multiple sources (by Moncep Sensors on hosts) for the discovery of more complex and more relevant monitoring information by using EPL queries to express aggregation and correlation patterns defined on a dedicated CEP Engine. The complex and relevant monitoring events are then stored persistently in event Queues to be used for administrator's notification or resource scaling policies. Let us consider an example of *HostInfoMsg* stream of metrics measurement gathered by agents from Host1 and an another *HostInfoMsg* stream of metrics measurement gathered from Host2 on Cloud Infrastructure: Suppose that the Grid manager wants to compute the average memory of a virtual machine VMx (eg. running a Grid application x) on Host1 for the last 5 minutes and report it but only if it exceeds 80%. The following EPL rule on Moncep Sensor (Host1) allows to express this request:

```
Insert into AvgVmxMemInfo Select
avg(VRUE.VMx.vmem_util) as VMx.vmem_util
from HostInfoMsg.win:time(5 min) where
avg(VRUE.VMx.vmem_util) > 80"
```

And suppose that he wants also to report the CPU usage of a virtual machine VMy (running the database of that Grid application x) on Host2 for the last 5 minutes but only if it exceeds 97%. The following EPL rule on Moncep Sensor (Host2) allows to express this request:

```
Insert into VmyCpuInfo Select
VCUE.VMy.vcpu_util as VMy.vcpu_util
from HostInfoMsg.win:time(5 min) where
VCUE.VMy.vcpu_util> 97
```

Then the two previous event streams can be correlated to have a more complex event stream that indicates whether the Grid application x will have enough resources to run properly. The following EPL rule on MonCep Monitor is sufficient to create the required monitoring information stream by using correlation pattern that selects properties from both event streams (*AvgVmxMemInfo*, *VmyCpuInfo*) and returns events that contain information about memory utilization and CPU capacity but only if the condition required is met:

```
Select VMx.vmem_util, VMy.vcpu_util from
pattern [every (memUtil=AvgVmxMemInfo and
cpuUtil=VmyCpuInfo)]
```

The "MonCep" Framework modules exchange large number of events (between "MonCep Sensors" and "Moncep Monitor") and within the modules (between monitoring agent and CEP engine). Thus, we designed a local communication mechanism based on a local event bus between the monitoring agent and the CEP engine within the "MonCep Sensor". We designed a remote communication model between each "MonCep Sensor" and the "Moncep Monitor" based on a remote event bus that uses a queuing mechanism.

C. Notification level

We complete the MonCep Framework by a notification system to allow visualization and real-time interpretation of processed monitoring events according to several notification strategies like: Saving the resulting events before they are routed to their destination. That is useful to feed the destination in case of message reception failure or in order to process the events later. Using a Log System to detect problems in real-time. Distributing events on event bus according to the XML standard format which is suitable for processing and display-based systems. Distributing warn events to administrators in several strategies (Mail and Web).

V. Implementation and Experimentation

For the validation of the proposed "Grid-Cloud" model in a real environment, we use DZ e-Science Grid[8] with the management system of OpenNebula [37] IaaS Cloud.
The DZ e-Science Grid is the National Algerian Grid infrastructure which is participating to European Grid projects: EumedGrid[9], EumedGrid-Support and CHAIN-REDS[10]. It represents a suitable environment to validate our integration approach since it includes all the management, computing and storage services provided by EMI architecture. We choose OpenNebula IaaS Cloud for virtual machine management because of its maturity, open architecture permitting extensions and open-source Cloud delivery system to perform the test.

We deployed a Front-end machine containing the OpenNebula virtual machine management, Virtual Appliances, VM image repository, User and Machines Information database, VM and network templates. We started by the deployment of two physical nodes Host1 and Host2 (Cloud infrastructure) containing the KVM[11] virtualization hypervisor.
We performed above this environment different tests to instantiate several DZ e-Science GRID services. The first test was the migration of an existing Grid service on the Cloud infrastructure as a virtual machine. The second test was to instantiate 4 virtual machines using a Virtual Appliance as new Grid services for the extension of the existing Grid site. Table I summarizes features of each machine (physical and virtual) on the deployed environment "Grid-Cloud".

The "MonCep" Framework modules are being implemented using the Java programming language and on widely accepted standards:

- "Host sFlow" [38] monitoring agent for virtual and physical metrics measurement and "Gmond" from Ganglia [31] as monitoring metric collector from the monitored hosts.
- Esper Engine written in java for processing events by EPL queries.
- "WSO2 ESB" [39] an open-source Enterprise Service BUS written in java as well used as local bus communication between Esper Engine and Gmond agent. It allows message mediation of events as XML/SOAP messages through "synapse-esper" [40] mediator.
- JMS API[12] (Apache Active MQ[13]) for remote event dissemination between the Framework modules.

For validating our approach, we started with the deployment of the Framework modules on the "Grid-Cloud" environment. "MonCep Sensors" on Host1 and Host2, "Moncep Monitor" on Front-end machine.

VI. Evaluation and discussion

We observed during the virtualized Grid platform deployment over the Cloud infrastructure that the process time to instantiated virtual machines was the main factor. According to table II, the resources deployment time is different depending on VM image's size to be instantiate as VM and involves the physical hosts hosting these VMs. To improve and optimize the resource provisioning time, we were considering the following points:

1) Since instantiating a VM involves the image copy process from the storage infrastructure that is hosting the VM image repository to the dedicated physical host and assuming that a minimal VM image size of several tens of gigabytes takes a time in the order of hundreds of seconds, we used for each VM creation a unique Virtual Appliance with an operating linux system to create a consolidated Image disk with a minimal size (eg. 16GB).
2) The image size will be increased if needed using LVM[14] functionalities after being copied on the physical host.
3) We were using the same VM template and the same network configuration for each new VM creation.

For the "MonCep" Framework experiments, we executed a representative scenario in which we defined a set of EPL queries to express Grid service's resource usage on the Cloud infrastructure: R1, R2, R3, R4, R5 and R6 EPL queries that express filter and aggregation pattern executed by "MonCep Sensors" installed on Host1 and Host2. For example the query R1 (executed by the "MonCep Sensor" on Host1) express that the average memory consumed by the VM (VM2_grid) on Host1 exceeds 80%. R1 is formed in EPL like:

```
select avg(memUtil_gridVM2)
as avgMemUtil_gridVM2 from
Ganglia.win:length_batch(10) having
avg(memUtil_gridVM2) > 80
```

On the other hand, the query R6 (executed by the "Moncep Sensor" on Host2) express that the outgoing network traffic from the VM (VM1_grid) on Host2 exceeds 30 bytes/sec for a window length of 20. R6 is formed in EPL like:

```
select bytesOut_gridVM1
as bytesOut_gridVM1 from
Ganglia.win:length_batch(20) having
bytesOut_gridVM1 > 30
```

We defined also C1 and C2 EPL queries that express correlation patterns executed by "MonCep Monitor" on the Front-End machine. For example C2 is a correlation query which select metric values on the "MonCep Monitor" that met both R5 and R6 conditions in the same time. C2 is written in EPL like:

```
select * from pattern [every
(cpuUser=avgCpuUser_Host2 and
bytesOut=bytesOut_gridVM1)]
```

[8] http://www.Grid.arn.dz
[9] http://www.eumedGrid.eu
[10] http://www.chain-project.eu/
[11] Kernel Based Virtual Machine. http://www.linux-kvm.org/
[12] http://java.sun.com/products/jms/
[13] http://activemq.apache.org/
[14] GNU/Linuxs Logical Volume Manager

TABLE I
"GRID-CLOUD" MACHINES CHARACTERISTICS

Machine	Type	OS	CPU	Cores	Memory	Storage
Opennebula VM Management	Front-end	Linux Redhat Entreprise 6.4	Intel(R) Xeon(R) CPU 2.27GHz	8	10GB	2TB
host1	Host	Linux Redhat Entreprise 6.4	Intel(R) Xeon(R) CPU 2.27GHz	8	10GB	2TB
host2	Host	Linux Redhat Entreprise 6.4	Intel(R) Core(TM) Duo CPU 3.00GHz	4	4GB	300GB
Integrated Grid Service	VM	Linux Redhat Entreprise 6.4	-	1	1GB	50GB
Grid_VM1 ... Grid_VM4	VMs	Linux Redhat Entreprise 6.4	-	1	1GB	16GB

TABLE II
DEPLOYMENT PROCESSING TIME

VM Image	VM Instance	Storage	Host	Deployment time/secondes
Integrated Grid Service	Grid_VM	50GB	host1	562
Virtual Appliance	Grid_VM1, Grid_VM2	16GB	host1	170
Virtual Appliance	Grid_VM3, Grid_VM4	16GB	host2	2441

Then we evaluated the MonCep Framework module's performance according to several areas: monitoring data reduction, network traffic and resource usage.

A. Monitoring data reduction

The monitoring approach described in this paper attempts to show the monitored data amount's reduction through the filter processing level. This first level naturally provides a reduction of the number of messages by filter and aggregation queries. As for the correlation level, it provides most relevant monitoring information than data collected by monitoring agents.

During the tests, we have calculated the number of XML messages[15] collected by monitoring agents and the number of events generated by each EPL query running on each MonCep Sensor and MonCep Monitor (Table III) in an interval time of 2 hours.

We may observe according to the results of table III that the number of messages ($1882 + 2268$ messages) collected by agents on Host1 and Host2 was significantly reduced ($362 + 44 + 380$ events) after being filtered by EPL queries on the first level (MonCep Sensors), while EPL correlation queries (C1 and C2) allowed us to provide ($290 + 90$ events) more complex and relevant events.

B. Resource usage and network traffic

Ganglia were used during experiments to measure CPU utilization, memory consumption and network traffic for all the Framework modules. If we take the example of Moncep Monitor, the measurements indicated that the average CPU usage of the machine was around 10% (similar before and after running the module) and the number of machine processes was constant. The memory consumption of the machine before the execution was around 3.5GB. It was increased up to 4GB after the execution of the module.

We also analyzed the packets received and network traffic for the events being disseminated between the MonCep Sensors and the MonCep Monitor. The average receiving network traffic on the MonCep Monitor machine was around 8Kbytes/s before executing the scenario and it reached 12Kbytes/s after the execution. The average number of packets received was 10 packets/s, it reached 30 packets/s.

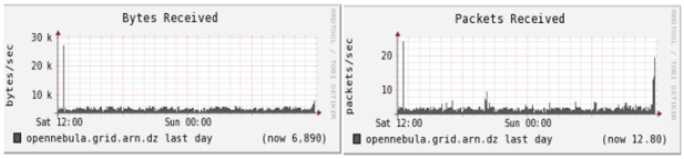

Fig. 4. Network Traffic of "opennebula.grid.arn.dz" machine before running the scenario.

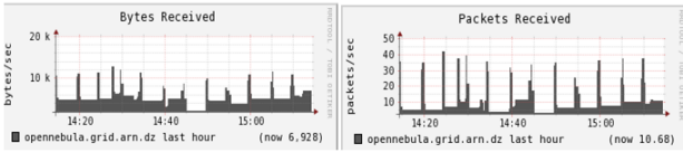

Fig. 5. Network Traffic of "opennebula.grid.arn.dz" machine during the scenario.

The experimental data clearly shows that the overhead added by the MonCep Framework modules is small. It shows also the feasibility of the solution and that the initial fixed objectives have been achieved: the amount of monitoring data reduction sent over the network and the discovery of complex and relevant monitoring information. These measures also show the lightness of the solution which is based on the integration of a set of standards and technology, especially as it supports the most important features of a Cloud Monitoring system [20]:

1) ***Scalability***: our solution is scalable in the sense that we can used a wide number of "MonCep Sensor" modules.

[15]One XML message collected by the agent is a set of monitoring metrics. Each metric is considered as event.

TABLE III
NUMBER OF GENERATED EVENTS

Machine / Queue of messages	Host1 (Moncep Sensor)	Host2 (MonCep Sensor)	Front-End (MonCep Monitor)
Ganglia_XML	1882 messages	2268 messages	-
avgMemUtil_GridVM2 (R1)	362 events	-	-
avgCpuIdle_Host2 (R2)	-	0 events	-
avgCpuIdle_Host1 (R3)	0 events	-	-
avgLoadOne_Host2 (R4)	-	0 events	-
avgCpuUser_Host2 (R5)	-	380 events	-
bytesOut_GridVM1 (R6)	44 events	-	-
correlation1 (C1)	-	-	290 events
correlation2 (C2)	-	-	39 events

We can also added many "MonCep Monitor" modules on different hosts depending on the number of "MonCep Sensor" to spread the load.

2) ***Elasticity***: each virtual or physical resource created or destroyed is detected by the agents of the Framework.
3) ***Adaptability and Lightness***: our solution is adaptable to grid services on the Cloud infrastructure and does not add significant or negative charge to the hosts.
4) ***Timeliness***: events detected by our solution are available in real-time.
5) ***Modularity***: the solution is modular and flexible.

VII. CONCLUSION

We presented in this paper an integration "Grid-Cloud" model that aims to introduce the Cloud management system within the Grid administrative domain as a private Cloud, while separating within the same Grid administrative domain, Grid services management responsibilities and Cloud resources management responsibilities. We completed the Model by the proposal of a new monitoring approach (MonCep Framework) based on "Complex Event Processing (CEP)" technology to provide correlation and monitored information's real-time processing as events by the discovery of "Patterns" according to the needs imposed by the execution of Grid services on Cloud IaaS-provided resources. We have shown through the implementation and testing of the monitoring solution on the "Grid-Cloud" environment the validation of the goals we have set: data monitoring reduction via information filtering and aggregation, discovery of complex and relevant monitoring information via information correlation in real-time and lightness of the solution.

Through the development of this work, we have identified several potential areas for the extension of the "Grid-Cloud" model and the "MonCep" Framwork. We plan to experiment the "Grid-Cloud" model on OpenStack [41] IaaS Cloud system. We will try to adapt the Framework for monitoring Grid services on Federated Clouds (Hybrid or heterogeneous Clouds). One of the goals of the Framework is to integrate correlation formulas that express Grid services performance metrics and Grid scientific application's quality of service such as availability and response-time to achieve SLA targets.

ACKNOWLEDGMENT

The authors would like to thank DZ e-Science GRID infrastructure and team.

REFERENCES

[1] H. Kloh, B. Schulze, A. Mury, and R. C. G. Pinto, "A scheduling model for workflows on grids and clouds," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM Trans, 2010.

[2] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in *International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEExplore, Nov 2008.

[3] B. Amedro, F. Baude, F. Huet, and E. Mathias, "Combining grid and cloud resources by use of middleware for spmd applications," in *2nd International Conference on Cloud Computing Technology and Science, Indianapolis, United States*, 2010.

[4] L. Rodero-Merino, E. Caron, A. Muresana, and L. Desprezt, "Using clouds to scale grid resources: An economic model," *Elsevier: Future Generation Computer Systems*, 2011.

[5] S. Jha, A. Merzky, and F. Geofrey, "Using clouds to provide grids higher-levels of abstraction and explicit support for usage modes," *Concurrency and Computation: Practice and Experience*, 2008.

[6] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, GCE '08*. IEEE Trans, 2008.

[7] M. Murphy and S. Goasguen, "Virtual organization clusters: Self-provisioned clouds on the grid," *Elsevier: Future Generation Computer Systems*, Oct 2010.

[8] C. Loomis, M. Airaj, M. Begin, E. Floros, S. Kenny, and D. Callaghan, "Stratuslab cloud distribution," *Research Report*, Jan 2012.

[9] D. Luckham and A. Wesley Professional, "Review of the power of events: An introduction to complex event processing in distributed enterprise systems," *ACM Trans*, Jan 2002.

[10] D. Lokhi Prasad, "Analyze and act on fast moving data: An introduction to complex event processing," MphasiS, Tech. Rep., Aug 2010.

[11] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Springer, 1998.

[12] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," in *Proceedings of the Conference on Cluster Computing and the Grid*. IEEE Conference publications, 2001.

[13] G. Alliance, *Official Documentation*, 2012, URL: http://www.globus.org/alliance [accessed: 2016-10-10].

[14] GLITE, *GLITE 3.2 User Guide*, 2012, URL: https://zenodo.org/record/6831/files/gLite-3-UserGuide.pdf [accessed: 2016-10-10].

[15] EMI, *EMI Architecture and Technology Developement Plan*, Apr 2013, URL: https://cds.cern.ch/record/1551298/ [accessed: 2016-10-10].

[16] L. Badger, T. Grance, R. Patt-Corner, and J. Voas, "Draft cloud computing synopsis and recommendations," National Institute of Standards and Technology, Technical Report 800-146, 2011.

[17] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology, Technical Report 15, 2009.

[18] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Elsevier: Future Generation Computer Systems*, vol. 25, 2009.

[19] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *SIGCOMM Computer Communications review*, 2009.

[20] G. Aceto, A. Botta, W. De Donato, and A. Pescapé, "Cloud monitoring: definitions, issues and future directions," *ACM Trans*, 2012.

[21] ——, "Cloud monitoring: A survey," *Computer Networks*, 2013, URL: http://dx.doi.org/10.1016/j.comnet.2013.04.001 [accessed: 2016-10-10].

[22] E. W. Site, "Esper," 2013, URL: http://www.espertech.com [accessed: 2016-10-10].

[23] E. Team and E. Inc, *Esper Reference version 4.10.0*, Sep 2013, URL: http://www.espertech.com/esper/release-4.10.0/esper-reference/html/index.html [accessed: 2016-10-10].

[24] C. Loomis, M. Airaj, M. Beegin, E. Floros, S. Kenny, and S. O'Callaghani, *European Research Activities in Cloud Computing*. Cambridge Scholars Publishing, Jan 2012, ch. StratusLab Cloud Distribution, pp. 260–282, book's authors: D.Petcu, J.Luis and V.Poletti.

[25] G. Barone, R. Bifulco, V. Boccia, D. Bottalico, R. Canonico, and L. Carracciuolo, "Gaas: Customized grids in the clouds," *Euro-Par 2012 Workshops*, pp. 577–586, 2012.

[26] F. Tusa, M. Paone, M. Villari, and A. Puliafito, "Clever: A cloud cross-computing platform leveraging grid resources," *IEEE Computer Society*, pp. 390–396, 2011, URL: http://dblp.uni-trier.de/db/conf/ucc/ucc2011.html [accessed: 2016-10-10].

[27] I. W. Site, "Worker nodes on demand service (wnodes)," 2014, URL: http://web.infn.it/wnodes [accessed: 2016-10-10].

[28] R. Byrom and al., "Apel: An implementation of grid accounting using r-gma," URL: http://www.academia.edu/17738300/APEL_An_implementation_of_Grid_accounting_using_R-GMA [accessed: 2016-10-10].

[29] C. Del Cano Novales and J. Gordon, "Apel cpu accounting in the egee/wlcg infrastructure," URL: https://indico.cern.ch/event/45477/contributions/1952735/attachments/946670/1342981/GDB-Accounting-Gordonv1.pdf [accessed: 2016-10-10].

[30] Nagios, "Nagios," URL: http://nagios.sourceforge.net/docs/nagioscore-3-en.pdf [accessed: 2016-10-10].

[31] M. Massie, B. Li, B. Nicholes, and V. Vuksan, *Monitoring with Ganglia*. Oreily, November 2012.

[32] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L. Vaquero, K. Nagin, and B. Rochwerger, "Monitoring service clouds in the future internet," *ACM Trans*, 2010.

[33] S. De Chaves, R. Brundo Uriarte, and C. Becker Westphall, "Toward an architecture for monitoring private clouds," *IEEE Communications Magazine*, December 2011.

[34] S. Padhy, D. Kreutz, A. Casimiro, and M. Lasige, "Trustworthy and resilient monitoring system for cloud infrastructures," *ACM Trans*, 2011.

[35] B. Konig, J. Alcaraz Calero, and J. Kirschnick, "Elastic monitoring framework for cloud infrastructures," *ACM Trans*, 2012.

[36] G. Katsaros, G. Kousiouris, G. Spyridon V, D. Kyriazis, A. Menychtas, and T. Varvarigou, "A self-adaptive hierarchical monitoring mechanism for clouds," *The Journal of Systems and Software*, 2012.

[37] Opennebula, "Opennebula 4.2 detailed features and functionality," May 2013, URL: http://opennebula.org/documentation:rel4.2:features [accessed: 2016-10-10].

[38] S. W. Site, "Host sflow," 2014, URL: http://host-sflow.sourceforge.net/ [accessed: 2016-10-10].

[39] WSO2, *WSO2 Enterprise Service Bus documentation version 4.7.0*, 2014, URL: https://docs.wso2.org/display/ESB481/Architecture [accessed: 2019-10-10].

[40] S.-F. Project, *Sci-Flex (Flexible Integration of SOA & CEP)*, 2008, URL: https://docs.wso2.org/display/ESB481/Architecture [accessed: 2016-10-10].

[41] Openstack, "Official site," URL: http://docs.openstack.org [accessed: 2016-10-10].