Middleware for a CLEVER Use of Virtual Resources in Federated Clouds

Francesco Tusa, Maurizio Paone, Massimo Villari and Antonio Puliafito
Università degli Studi di Messina, Facoltà di Ingegneria
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {ftusa,mpaone,mvillari,apuliafito}@unime.it

***Abstract*—Nowadays Cloud Computing is becoming an interesting distributed computation infrastructure able to strongly leverage the concept of Virtualization of physical resources. This paper deals with the opportunity for managing Virtualization Infrastructures in Federated scenarios. In particular, the middleware we are introducing presents several features enabling an useful and easy management of private/hybrid clouds and provides simple and easily accessible interfaces to interact with different "interconnected" clouds. In that context, one of the main challenges it is necessary to address is the capability that systems need to interact together, maintaining separated their own domains and the own administration policies. A Cloud middleware has been designed, we named it CLEVER, and this paper describes the architecture in each part. Several UML schemas highlight the relevant complexity of our new architecture. In the current status of the work, a primitive prototype, integrating some features of the whole architecture, has been developed as far software classes implementing the basic functionalities.**

***Keywords*-Cross Cloud Computing; XMPP; Fault Tolerance; Virtual Infrastructure Management; clusters.**

I. INTRODUCTION

Cloud computing is generally considered as one of the more challenging topic in the Information Technology (IT) world, although it is not always fully clear what its potentialities are and which are all the involved implications. Many definitions of cloud computing are presented and many scenarios exist in literature. In [1], Ian Foster describes 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. Until now, such trend has brought the steady rising of hundreds of independent, heterogeneous cloud providers managed by private subjects yielding various services to their clients.

In order to provide a flexible use of resources, cloud computing delegates its functionalities in a virtual context, allowing to treat traditional hardware resources like a pool of virtual ones. In addition virtualization enables the ability of migrating resources, regardless of the underlying real physical infrastructure. Peter Mell and Tim Grance from the Computer Security Division of the National Institute of Standards and Technology (NIST) are initiating important government efforts to shape essential components in the broader cloud arena[2]. They identified clouds provide services at three different levels: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS):

- *SaaS*: Software as a Service represents the capability given to the consumer in using provider's applications running on a cloud infrastructure and accessible from various client devices through a thin client interface such as a Web browser.
- *PaaS*: Platform as a Service represents the capability given to the consumer in order to deploy his own application onto the cloud infrastructure using programming languages and tools supported by the provider (i.e Java, Python, .Net).
- *IaaS* represents the capability given to the consumer for renting processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure.

Our work is aimed to define new functionalities that focus on the lower level of services that is *IaaS*. Service Provides are customers of such infrastructures (SPs, i.e., Ebay, Facebook, Twitter, etc.); they need to easily deploy and execute their services. Owners of these infrastructures are commonly named Infrastructure Providers (IPs) or Cloud Providers (i.e. public clouds: Amazon EC2, Google, Saleforce, Microsoft Azure, RightScale, etc.). We consider the cloud as constellations of hundreds of independent, heterogeneous, private/hybrid clouds. Currently many business operators have predicted that the process toward an interoperable federated cloud scenario will begin shortly. In [3], the evolution of the cloud computing market is hypothesized in three subsequent phases:

- *Monolithic* (now), cloud services are based on proprietary architectures - islands of cloud services delivered by megaproviders (this is what Amazon EC2, Google, Salesforce and Microsoft Azure look like today);
- *Vertical Supply Chain* , over time, some cloud providers will leverage cloud services from other providers. The clouds will be proprietary islands yet, but the ecosystem building will start;
- *Horizontal Federation*, smaller, medium, and large

 ISBN: 978-1-61208-106-9

providers will federate horizontally themselves to gain: economies of scale, an efficient use of their assets, and an enlargement of their capabilities.

This paper aims to describe our architecture able to make up an interoperable heterogeneous cloud middleware that accomplishes the *Horizontal Federation*. The main challenge of interoperable clouds that needs to overcome is the opportunity that federated heterogeneous systems have to interact together, maintaining separated their own domains and administration policies. The architecture we designed is characterized by a main skeleton, in which it is possible to add more functionalities, using a flexible approach plug-in based. The diagrams we present in the next sections show several details of our design choices. The complexity of the system is rather high, this is due to the number of constrains and issues we are trying to face. The main motivation to design a new cloud middleware has been given from the lack in literature of such a middleware. The overall motivations to propose CLEVER (CLoud-Enabled Virtual EnviRonment) have been provided in our recent work [4]. That work also provides an useful and detailed description of pros and cons in CLEVER against the other existing infrastructures.

The paper is organized as follows. In Section II, we provide a brief description about the new Cloud stack. In the same section we briefly explore the current state-of-the-art in Cloud Computing and existing middleware implementations. This section critically analyses the features of such cloud middlewares and motivates the need to design and implement a new one. Section III introduces CLEVER as the *Virtual Infrastructure Manager* and explains the functional and non functional requirements that it tries to meet. Section IV provides an overview of the CLEVER's features, which are then deeply discussed in Section V where also a logical description of each middleware module is reported together with a brief description of our prototype implementation (see Sec. VI). Section VII concludes the paper.

II. Background and Related Works

The work we are describing deals with the technology necessary to make up a cloud infrastructure whatever level it is: *IaaS, PaaS and SaaS*. This technology is the well-known as *Virtualization*. Cloud Computing strongly exploits the concept of virtualization of physical resources (hosted in IaaS), through the Instantiation of Virtual Machines (VMs). During our dissertation we named these VMs as Virtual Environments (VEs), a more general term useful to describe other virtual containers (i.e., Java Virtual Containers). These VEs can be seen as the smaller part of cloud systems: "atoms", in which IT Services (i.e., PaaS or SaaS) are confined into. CLEVER is a middleware able to manage VEs in cross cloud environments. This middleware should accomplish an important core of a general framework able to address Federation among sites, guaranteeing Fault Tolerance, Easy Configuration (Zero-conf), Accounting and Billing, Performance (SLAs), Security, etc.

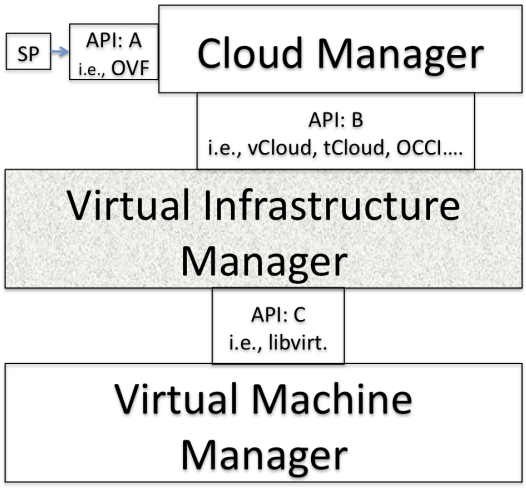

Figure 1. Cloud Management Stack

In order to identify the main components constituting a cloud and better explain the federation idea on which our work is based, we are considering the internal architecture of each cloud as the three-layered stack [5] presented schematically in Fig. 1. Starting from the bottom, we can identify: *Virtual Machine Manager, (VMM)*, *Virtual Infrastructure (VI) Manager* and *Cloud Manager, (CM)*. The Virtual Machine Manager is the layer in which the hardware virtualization is accomplished. It hides the physical characteristics of a computing platform from SPs, instead showing another abstract computing platform. The software that controls the virtualization used to be called a "control program" at its origins, but nowadays the term *hypervisor* is preferred. The main hypervisors currently used to virtualize hardware resources are: Xen [6], KVM [7], VMware [8], VirtualBox [9], Microsoft Hyper-V [10], Oracle VM [11], IBM POWER Hypervisor (PR/SM) [12], Apple Parallel Server [13], etc. For example one type of a VMM can be a physical machine with the Xen hypervisor para-virtualizer [14] controlling it (in this case the VM are Xen domains), whereas another type can be a machine with the necessary software to host KVM (full-virtualizer [15]), and so on. The VI manager is a fundamental component of private/hybrid clouds acting as a dynamic orchestrator of VEs, which automates VEs setup, deployment and management, regardless of the underlying level. The Cloud Manager layer is instead able to transform the existing infrastructure into a cloud, providing cloud-like interfaces and higher-level functionalities for security, contextualization and VM disk image management.

Recently, many Virtual Infrastructure Managers are appearing in literature, some of them can be listed as follows: OpenNebula [16], OpenQRM [17], Nimbus [18], Eucalyptus [19], etc.

 ISBN: 978-1-61208-106-9

The architectures listed above were conceived considering two main philosophies:

- Apply the virtualization model to simplify the management of proprietary datacenters.
- Reorganize the previous distributed computation middleware for accomplishing an easy management of virtualization capabilities.

The early case represents a simpler way to organizes hardware resources hosted in local and private datacenters that need a tedious and complex management (i.e., OpenQRM and Eucalyptus). In the latter, we have infrastructures in which the original focus is modified to be adapted for clouds; for instance rearranging of GRID infrastructure as well as Open Nebula and Nimbus. For instance OpenNEbula (ONE) can be considered as evolution of a Grid Manager aimed to Virtualization Systems. Thanks to the European Project RESERVOIR [20], ONE is including some key concepts we are trying to address in our work. Nimbus was originated by an adaption of Globus (GRID), in fact it is currently deployed into a Globus 4.0.x Java container.

Nowadays it is time for new cloud architectures able to deal with new cloud requirements and constrains. Furthermore these architectures must to be conceived from the scratch, that is the core of the system has to include at the beginning many more basic capabilities necessary for Cloud Computing environments. CLEVER was designed keeping mind these goals, adding as much as possible and at the early stage many functionalities. In our system, Cloud Manager layers have to offer the following opportunities: Cloud Federation, Composability, and Orchestration of resources, distributed virtual resources management, inter-cloud security, inter-cloud business model, inter-cloud networking, inter-cloud monitoring, etc. Even at Virtual Infrastructure Manager layers it is necessary to add these following features in order to satisfy the CM layers needs: local virtual resources management, load-balancing, fault tolerance, intra-cloud security, intra-cloud business models, intra-cloud networking, intra-cloud monitoring, etc. To drawing new cloud systems, it is important to initially take into account concepts of ”inter” and ”intra” requirements and capabilities. In the interoperability arena, Application Programming Interface (APIs) cover a strong role, as shown by Fig. 1 (rectangle shapes in between layers: A, B and C). APIs have to guarantee the interoperability among Clouds (see section II-B) in Private, Public and Hybrid scenarios.

Business and trading may represent the engine of Cloud development under many perspectives, Section II-A highlights these concepts.

In the direction of an open [21] and advanced cloud framework, NASA is working on an ambitious project: OpenStack [22]. On their web portal, they stated: *The goal of OpenStack is to allow any organization to create and offer cloud computing capabilities using open source software running on standard hardware. OpenStack Compute is software for automatically creating and managing large groups of virtual private servers. OpenStack Storage is software for creating redundant, scalable object storage using clusters of commodity servers to store terabytes or even petabytes of data*. These sentences remark as our overall assumptions and motivations, which led us to develop CLEVER, are completely in line with the new Cloud researching context.

In our final considerations, the figured world can appear rather complex and hard to face but the virtualization environments might greatly simplify the necessary effort. Specifically all hypervisors leverage the new virtualization capabilities offered by hardware architectures (i.e., Intel, AMD, etc.), thus allowing to limit their diversity. These features are strongly exploited by cloud operators because they increase the system performances. But at the same time thus determines the reduction of the differentiation among hypervisors, hence we can foresee a great challenge in the Virtual Computing (or Virtual Cloud). We can remark the concept considering the *virtualization world* as the simplest way to identify and regroup a subset of functionalities that it is necessary to deal within a Cloud. Since the formalization of such requirements, constrains, and capabilities might not be particularly tough to accomplish.

A. Commercial Cloud Infrastructures and their businesses.

Cloud Computing is finding a wide consensus among IT operators, because behind its computation model it is possible to advise a real business model [23]. One of the main reason of the Grid Computing failure, seen under the economic perspectives, was the not applicability of any business model on it. Currently in the IT world we are assisting to a progressive mixing of meaningful efforts among operators, scientific communities and organizations for standards (as well: DTMF, IEEE, IETF, ITU, etc.) to enforce this new distributed computation infrastructure: Cloud Computing. We can state that it is not possible to draw clear boundaries among commercial, academic, opensource and legacy cloud platforms. Several platforms born in academia have migrated to commercial purposes (Nimbus, Eucaliptus, OpeNebula), others from legacy to OpenSource (OpenQRM) to consolidate and advertise the work done, and finally a few of them were totally born as OpenSource (OpenStack) or commercial (Citrix, VMware, Telefonica, IBM, HP, etc.) approaches. The evolution we are noting is a movement of lower layers toward the upper layers of the Cloud stack. For instance hypervisor as Vmware and Xen is providing a new cloud middleware able to address all the issues in entire distributed datacenters (VMM to VIM: VMware vSphere, XEN Cloud Platform). VIMs are looking at Cloud Management.

In this process, we think the commercial IT world might be far to define an interoperable framework, whereas the current VIM middleware should change the original focus to be ready for the new challenges. A platform ready to

face these challenges might be the OpenStack framework, it promises an interesting contribution, but it is a young architecture yet. The CLEVER infrastructure might also provide a valid support in that direction.

The standardization rate of proposals is also showing an huge interest of IT stockholders, the next section will provide a brief enlightenment on that.

B. Cloud Standards in a Nutshel: APIs

The concepts of interoperability need to be enforced at Cloud Manager level, but the VIMs need to provide Application Program Interfaces (APIs) enabling the full control and monitoring of cross-cloud virtual resources. The federation among Clouds can be perpetrated if each Cloud Operator can dynamically join the federation [24], interact with federated clouds in trustiness [25],[26] and finally exchange data with their partners [27]. Since the federation can exist if there is a high level infrastructure that allows this aggregation. Furthermore it is not possible to spread computation among several IPs, without an adequate intercommunication protocol among parties. Fig. 1 highlights this concept. In the Cloud stack it is necessary to include APIs among each layer that allow to cross-correlate more heterogeneous infrastructures. The working group in DMTF Standards is defining the Cloud Incubator Initiative for Cloud Management Standards. It was formed to address management interoperability for Cloud Systems [28]. The DTMF organization has began the initiatives in Virtualization environments with Open Virtual Format (OVF, see the fig., API: A).The format (OVF) represents an early descriptor able to define the customer requirements, in terms of number of VEs to instantiate; memory, CPUs, storage and etc. to allocate and so on. This format permits the interaction from IaaSs and their costumers (SPs).

Inside DTMF many IT companies are also trying to add standards in the cloud particularly useful for their businesses. For instance VMware has introduced in DMTF the VCloud[29] API, while Telefonica their TCloud [30] API. Both proposals should guarantee a set of new parameters useful for the cross interaction. In this way, many more functionalities should be addressed, such as: load balancing, fault tolerance (VMs replication), network configuration, firewalling policies, etc. The main feature of such standards is the adoption of OVF as the base, to meet the SPs requirements and constrains, without an heavy translation among different descriptor files. Another example of API standardization process is the Open Cloud Computing Interface (OCCI) standard [31]. The OCCI standard, is proposed inside the Open Grid Forum (OGF), it should guarantee an exposition of VIM capabilities, as depicted in Fig. 1 (API: B). But it does not appear to be nor particularly flexible neither oriented to SPs requirements. In this case it needs a translation from OVF. All the standards presented above are based on XML in order to guarantee the portability and interoperability in heterogeneous systems.

In the following subsection we briefly describe the main VIMs. Subsequently, we provide an in-depth description of our CLEVER architecture.

C. Related Works

As we introduced in Section II and stated in [5], cloud management can be performed at the lowest stack layer of Fig. 1 as *Virtual Infrastructure Management*.

The project OpenQRM [17] is an open-source platform for enabling flexible management of computing infrastructures. It is able to implement a cloud with several features that allows the automatic deployment of services. It supports different virtualization technologies and format conversion during migration. This means VEs (appliances in the OpenQRM terminology) can not only easily move from physical to virtual (and back), but they can also be migrated from different virtualization technologies, even transforming the server image. OpenQRM is able to grant a complete monitor of systems and services by means of the Nagios tool [32], which maps the entire openQRM network and creates (or updates) its corresponding configuration (i.e., all systems and available services). Finally, OpenQRM addresses the concepts related to High Availability (HA) systems: virtualization is exploited to allow users to achieve services failover without wasting all the computing resources (e.g. using stand-by systems).

OpenNebula [16] is an open and flexible tool to build a Cloud computing environment. OpenNebula can be primarily used as a virtualization tool to manage virtual infrastructures in a data-center or cluster, which is usually referred as Private Cloud. Only the more recent versions of OpenNebula are trying to supports Hybrid Cloud to combine local infrastructure with public cloud-based infrastructure, enabling highly scalable hosting environments. OpenNebula also supports Public Clouds by providing Cloud interfaces to expose its functionalities for virtual machine, storage and network management.

Still looking at the stack of Fig. 1, other middlewares work at an higher level than the VI Manager (High-level Management) and albeit they provide high-level features (external interfaces, security and contextualization) their VI management capabilities are limited and lack VI management features: this type of cloud middlewares include Globus Nimbus [18] and Eucalyptus [19].

Nimbus [18] is an open source toolkit that allows to turn a set of computing resources into an Iaas cloud. Nimbus comes with a component called workspace-control, installed on each node, used to start, stop and pause VMs, implements VM image reconstruction and management, securely connects the VMs to the network, and delivers contextualization. Nimbus's workspace-control tools work with Xen and KVM but only the Xen version is distributed. Nimbus provides

 ISBN: 978-1-61208-106-9

interfaces to VM management functions based on the WSRF set of protocols. There is also an alternative implementation exploiting Amazon EC2 WSDL.

Eucalyptus [19] is an open-source cloud-computing framework that uses the computational and storage infrastructures commonly available at academic research groups to provide a platform that is modular and open to experimental instrumentation and study. Eucalyptus addresses several crucial cloud computing questions, including VM instance scheduling, cloud computing administrative interfaces, construction of virtual networks, definition and execution of service level agreements (cloud/user and cloud/cloud), and cloud computing user interfaces.

III. The CLEVER architecture

CLEVER aims to provide *Virtual Infrastructure Management* services and suitable interfaces at the *High-level Management* layer to enable the integration of high-level features such as Public Cloud Interfaces, Contextualization, Security and Dynamic Resources provisioning.

Looking at the middleware implementations, which act as *High-level Cloud Manager* [18], [19], it can be said that their architecture lacks modularity: it could be a difficult task to change these cloud middleware for integrating new features or modifying the existing ones. CLEVER instead intends granting an higher scalability, modularity and flexibility exploiting the plug-ins concept. This means that other features can be easily added to the middleware just introducing new plug-ins or modules within its architecture without upsetting the organization.

Furthermore, analysing the current existing middleware [17], [16], which deal with the *Virtual Infrastructure Management*, we retain that some new features could be added within their implementation in order to achieve a system able to grant high modularity, scalability and fault tolerance. Our idea of cloud middleware, in fact, finds in the terms flexibility and scalability its key-concepts, leading to an architecture designed to satisfy the following requirements: 1) *persistent communication* among middleware entities; 2) *transparency* respect to "user" requests; 3) *fault tolerance* against crashes of both physical hosts and single software modules; 4) heavy *modular design* (e.g., monitoring operations, managing of hypervisor and managing of VEs images will be performed by specific plug-ins, according to different OS, different hypervisor technologies, etc.); 5) *scalability* and *simplicity* when new resources have to be added, organized in new hosts (within the same cluster) or in new clusters (within the same cloud); 6) automatic and optimal *system workload balancing* by means of dynamic VEs allocation and live VEs migration.

Looking at Figure 2, we believe the existing solutions lack a cloud Virtual Infrastructure able to implement all the characteristics of each row. The big black dot in the cell specifies the feature that the middleware has. CLEVER has

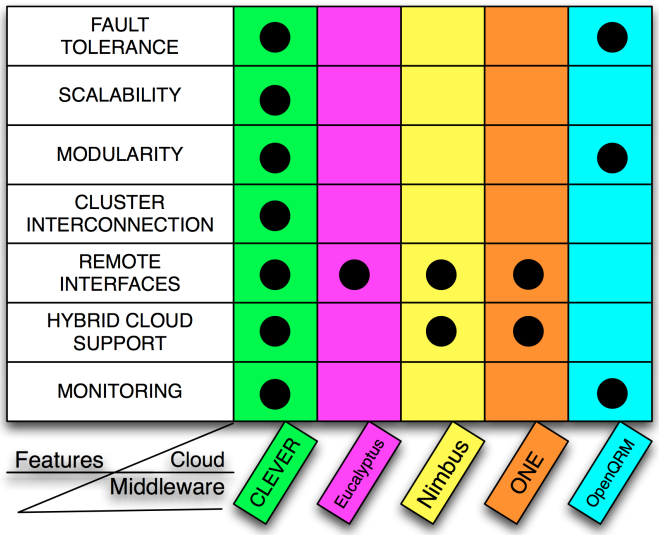

Figure 2. A comparison of CLEVER features VS other Cloud middleware implementations

all the features listed, in fact CLEVER is able to manage in a flexible way both physical infrastructures composed of several hosts within a cluster and physical infrastructures composed of different "interconnected" clusters. This task is performed ensuring fault tolerance while operations are executed, exploiting particular methods which allow the dynamic activation of recovery mechanisms when a crash occurs. Furthermore, due to its pluggable architecture, CLEVER is able to provide simple and accessible interfaces that could be used to implement the concept of hybrid cloud. Finally, it is also ready to interact with other different cloud technologies supposing that their communication protocol or interfaces are known.

IV. CLEVER Reference Scenario

Our reference scenario consists of a set of physical hardware resources (i.e., a cluster) where VEs are dynamically created and executed on the hosts considering their workload, data location and several other parameters. The basic operations our middleware should perform refer to: 1) Monitoring the VEs behavior and performance, in terms of CPU, memory and storage usage; 2) Managing the VEs, providing functions to destroy, shut-down, migrate and network setting; 3) Managing the VEs images, i.e., images discovery, file transfer and uploading.

Considering the concepts stated in [4] and looking at Fig. 1, such features, usually implemented in the *Virtual Infrastructure Management* layer, can be further analyzed and arranged on two different sub-layers: *Host Management* (lower) and *Cluster Management* (higher).

Grounding the design of the middleware on such logical subdivision and taking into account the satisfaction of all

 ISBN: 978-1-61208-106-9

the above mentioned requirements, the simplest approach to design our middleware is based on the architecture schema depicted in Fig. 3, which shows a cluster of n nodes (also an interconnection of clusters could be analyzed) each containing a *host level* management module (Host Manager). A single node may also include a *cluster level* management module (Cluster Manager). All these entities interact exchanging information by means of the *Communication System* based on the XMPP. The set of data necessary to enable the middleware functioning is stored within a specific *Database* deployed in a distributed fashion.

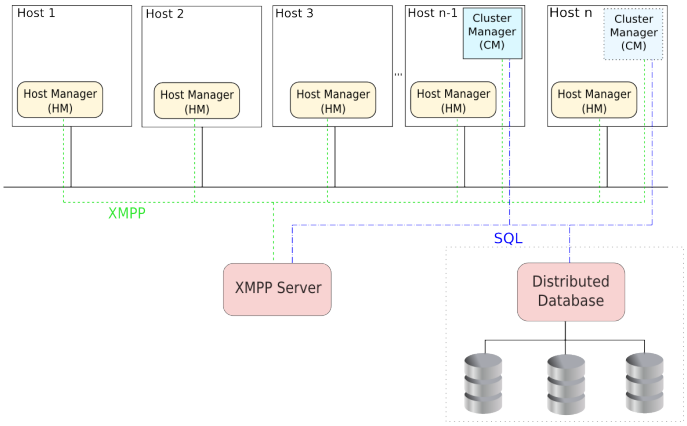

Figure 3. CLEVER reference Scenario representation

Figure 3 shows the main components of the CLEVER architecture, which can be split into two logical categories: software agents (typical of the architecture itself) and the tools they exploit. To the former set belong both *Host Manager* and *Cluster Manager*:

- **Host manager (HM)** performs the operations needed to monitor the physical resources and the instantiated VEs; moreover, it runs the VEs on the physical hosts (downloading the VE image) and performs the migration of VEs (more precisely, it performs the low level aspects of this operation). To carry out these functions it must communicate with the hypervisor, hosts' OS and distributed file-system on which the VE images are stored. This interaction must be performed using a plug-ins paradigm.
- **Cluster Manager (CM)** acts as an interface between the clients (software entities, which can exploit the cloud) and the HM agents. CM receives commands from the clients, performs operations on the HM agents (or on the database) and finally sends information to the clients. It also performs the management of VE images (uploading, discover, etc.) and the monitoring of the overall state of the cluster (resource usage, VEs state, etc.). Following our idea, at least one CM has to be deployed on each cluster but, in order to ensure higher fault tolerance, many of them should exist. A master CM will exist in active state while the other ones will remain in a monitoring state, although client messages are listened whatever operation is performed.

Regarding the tools such middleware components exploit, we can identify the *Distributed Database* and the *XMPP Server*:

- **Distributed Database** is merely the database containing the overall set of information related to the middleware (e.g. the current state of the VEs or data related to the connection existing on the Communication System). Since the database could represent a centralized point of failure, it has to be developed according to a well structured approach, for enabling fault tolerance features. The best way to achieve such features consists of using a Distributed Database
- **XMPP Server** is the "channel" used to enable the interaction among the middleware components. In order to grant the satisfaction of our requirements, it is able to offer: decentralization (i.e., no central master server should exist: such capability in native on the XMPP) in a way similar to a p2p communication system for granting fault-tolerance and scalability when new hosts are added in the infrastructure; flexibility to maintain system interoperability; security based on the use of channel encryption. Since the XMPP Server also could exploit the distributed database to work, the solution enables an high fault tolerance level and allows system status recovery if a crash occurs.

V. ARCHITECTURE DESIGN

In this Section, we will provide further details about the internal structure of the two main software components deployed on the cluster's hosts (already pointed out in Fig. 3) analyzing the Host Manager and the Cluster Manager. Moreover, the main results of our design process will be presented using the UML description.

Figure 4 shows the internal organization of such CLEVER components and how they are deployed on the reference scenario already introduced.

The left part of the Figure points out the Host Manager. As previously stated, such component lies in the lower part of the VI layer of the stack and interacts with the OS of each host of the cluster. The main modules composing the Host Manager are described below.

- **Monitor**: Provides resource usage monitoring for each host. The information are organized and made available to the HM coordinator.
- **Hypervisor Interface**: is the middleware back-end to the hypervisor running on the host's OS. Different virtualization technologies could be employed using different plug-ins structure has to be developed.
- **Image Manager**: supplies to the Hypervisor Interfaces the needed VE disk-image corresponding to a specific

 ISBN: 978-1-61208-106-9

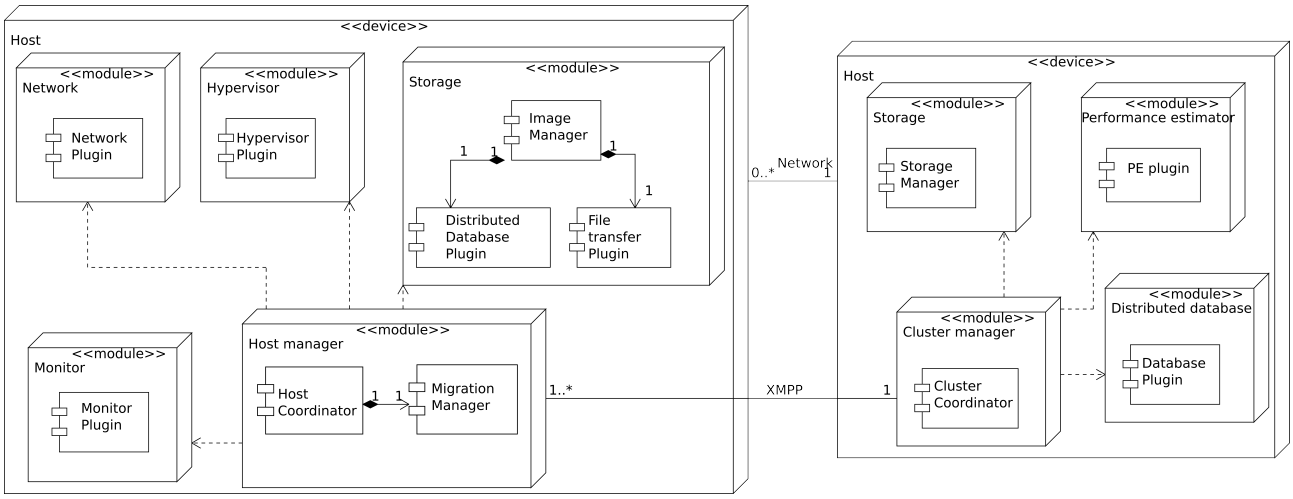

Figure 4. CLEVER deployment diagram

VE. Different plug-ins associated could be associated to different data access/transfer methods.

- **Network Manager**: Gathers information about the host network state. Manages host network (OS level) according to the guidelines provided by the HM Coordinator: dynamically creates network bridges, routing and firewalling rules.

The Cluster Manager, depicted in the right part of Figure 4, lies in the higher part of the same stack layer of the Host Manager and coordinates all the entities of the middleware (i.e. the HMs). Its internal composition is described in the following.

- **Database Manager**: interacts with the database employed to store information needed to the cluster handling. Database Manager must maintain the data strictly related to the cluster state.
- **Performance Estimator**: Analysis of the set of data collected from the Coordinator, in order to compute and provide a probable trend estimation of the collected measures.
- **Image Manager**: manages both registrations and uploads within the Cluster Storage System of the VEs disk-images. The Storage Manager is used to perform the registration process of such files and manage the internal cluster distributed file system.

As the Figure points out, both Host Manager and Cluster Manager include a specific module named respectively Host Coordinator and Cluster Coordinator. More specifically, the Host Coordinator manages the communication between the Host Manager internal modules while the Cluster Coordinator performs the same task for the Cluster Manager modules.

Furthermore, the Host Coordinator and Cluster Coordinator, exploiting the XMPP connection, allows the middleware functioning by exchanging messages in a chat-like fashion: as previously introduced, both Host Manager and Cluster Manager(s) will attend a XMPP chat session for enabling operation of resource monitoring, VEs allocation. The middleware back-end to the XMPP is represented by the Host Coordinator and the Cluster Coordinator.

As will be better explained in the following, within each component of the middleware, each module has been designed according to a well-structured plugin fashion: this allows a given module to be independent from the underlying technologies of the hosts on which it is running, since the best plugin will be employed and linked (dynamically) to the corresponding module.

Figure 5, considering the reference scenario described above, presents the whole use case diagram of the middleware, highlighting the main offered services and involved actors. The latter include Host Coordinator, Cluster Coordinator, Operating System, Hypervisor, Client and (eventually) Other Clusters. Starting from such diagram which describes the middleware specification, the design process has been refined using both activity diagrams, sequence diagrams and other use case diagrams. The final result of such work has been the definition of the whole system regarding the classes (referring to the object oriented software development) of each module on the CLEVER components.

VI. Prototype Implementation

In the current status of the work, a primitive prototype, integrating some features of the whole architecture, has been developed as far software classes implementing the basic functionalities of the Host Manager and Cluster Manager have been written using the Java programming language, allowing middleware components interaction by means of the XMPP protocol: the management of all the problems related to identification, access, monitoring and control of hardware resources in the Cloud Environment have been thus addressed in the current implementation.

 ISBN: 978-1-61208-106-9

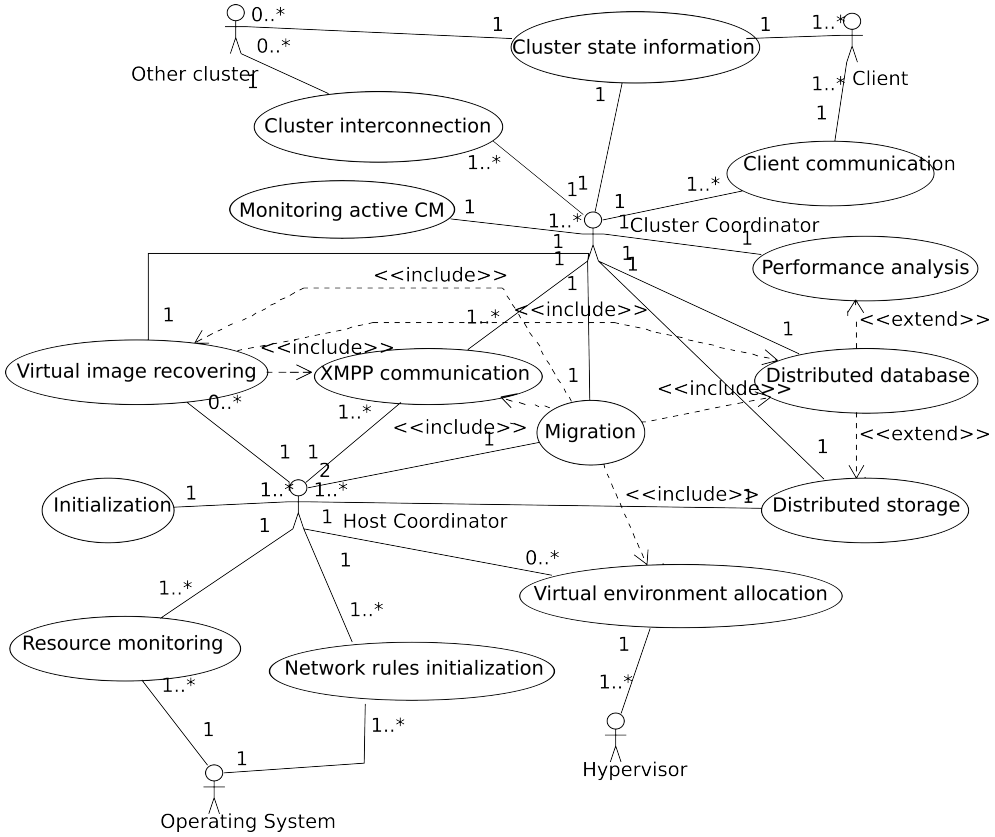

Figure 5. CLEVER general use case diagram

As previously introduced, using the UML description we achieved a deep description of the middleware behavior. The design process has lead to the definition of a series of class, packaged according to the deployment diagram of Fig. 4, each referred to a different module of the middleware components: currently, our set class consists of approximately 150 hundreds of classes and 50% of them have been fully implemented.

Figure 6 depicts the class diagram of the Hypervisor Interface module: we would like to underline that in our current implementation, each module of both the HM and the CM has been developed as a self-contained entity running into a different OS process. Such approach ensures an higher fault-tolerance level (since a failure of a single process/module will be isolated) and increases the modularity of the whole system. A drawback of such approach is related to the high level of complexity introduced: since each module within the middleware components represents a different process, a specific (interprocess) communication method has to be employed to ensure the interaction between all the modules. According to the aforementioned plugin approach, our communication method is based on a particular plugin referring to Apache ActiveMQ [33] and Java Message Service.

According to the description reported in the Section III, both the Host Manager and the Cluster Manager have been implemented including a Java class which act as XMPP client exploiting the Smack [34] set of libraries. By means of such software module, our middleware components are able to communicate each other exploiting the XMPP facilities provided by the Ejabber XMPP server [35]. The latter has been chosen due to its distributed and fault tollerance features.

In order to manage VEs allocation, our implementation of the HM components, includes a specific software module whose aim refers to the interaction with the hypervisor running on the Host OS: such software practically implements a plugin for the Hypervisor Interface of the Host Manager, linking the Libvirt [36] set of libraries. Using the API offered by libvirt, the plugin is able to start create and start virtual environment on several hypervisor.

VII. Conclusions and Future Works

In this work, we described the design principles and the preliminary prototype implementation of our cloud middleware named CLEVER: unlike similar works existing in the literature, CLEVER provides both *Virtual Infrastructure Management* services and suitable interfaces at the *High-level Management* layer to enable the integration of Public

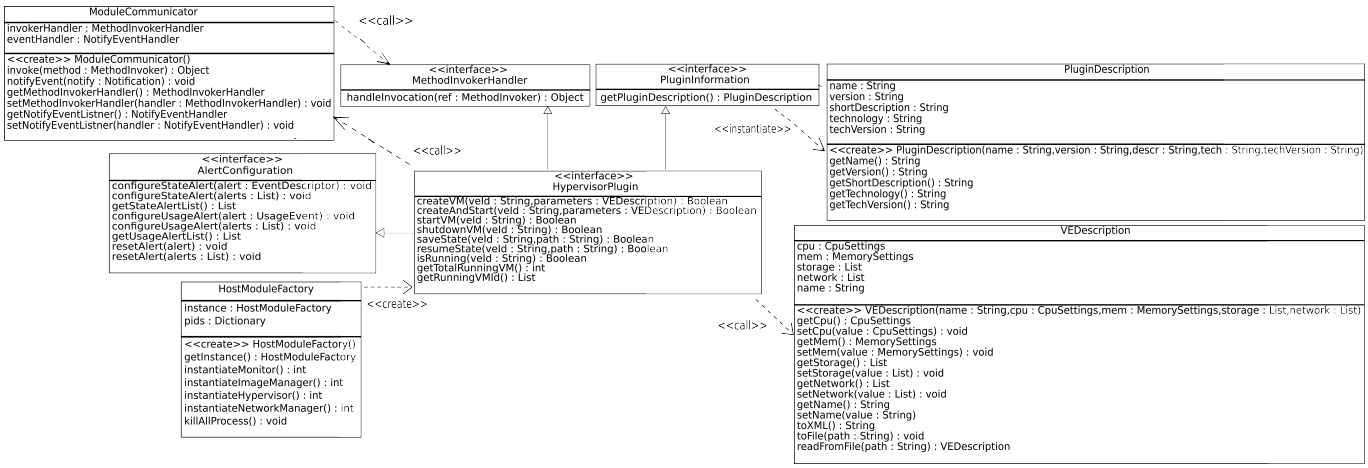

Figure 6. CLEVER: example of the hypervisor interface class diagram

Cloud Interfaces, Contextualization, Security and Dynamic Resources provisioning within the cloud infrastructure.

Furthermore, thanks to its pluggable design, CLEVER grants scalability, modularity, flexibility and fault tolerance. We are working to further extend the middleware functionalities according to the reference UML model described in this paper. Moreover, a set of tests is being executed to obtain a comprehensive set of experimental results to deeply evaluate the behavior of the middleware and its performance.

VIII. ACKNOWLEDGEMENTS.

The research leading to the results presented in this paper has received funding from the European Union's seventh framework programme (FP7 2007-2013) Project RESERVOIR under grant agreement number 215605.

REFERENCES

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, pp. 1–10, 2008.

[2] National Institute of Science and Technology. NIST Definition of Cloud Computing; http://csrc.nist.gov/groups/SNS/cloud-computing/ July 2010.

[3] T. Bittman, "The evolution of the cloud computing market," *Gartner Blog Network,http://blogs.gartner.com/thomas_bittman/2008/11/03/the-evolution-of-the-cloud-computing-market/*, November 2008.

[4] F. Tusa, M. Paone, M. Villari, and A. Puliafito., "CLEVER: A CLoud-Enabled Virtual EnviRonment," in *15th IEEE Symposium on Computers and CommunicationsS Computing and Communications, 2010. ISCC '10. Riccione*, June 2010.

[5] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *Internet Computing, IEEE*, vol. 13, pp. 14–22, Sept.-Oct. 2009.

[6] Xen Hypervisor - Leading open source hypervisor for servers. http://www.xen.org/ August 2010.

[7] KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware. http://www.linux-kvm.org August 2010.

[8] Virtualization: the essential catalyst for enabling the transition to secure cloud computing. http://www.vmware.com/it/ August 2010.

[9] x86 virtualization software package developed by Sun Microsystems. http://www.virtualbox.org/ August 2010.

[10] Microsoft Hyper-V Server 2008 R2, the stand-alone product that provides a reliable and optimized virtualization solution. http://www.microsoft.com/hyper-v-server/en/us/default.aspx August 2010.

[11] Oracle VM, ther virtualization software that fully supports both Oracle and non-Oracle applications, and delivers more efficient performance. http://www.oracle.com/us/technologies/virtualization/oraclevm/ August 2010.

[12] The POWER Hypervisor, the abstraction layer between the hardware and firmware and the operating system instances for GX host channel adapter (HCA) implementations. http://publib.boulder.ibm.com/infocenter/powersys/v3r1m5/ August 2010.

[13] The Mac: Real versatility through virtualization. http://www.apple.com/business/solutions/it/virtualization.html August 2010.

[14] Paravirtualization. The virtualization technique that presents a software interface to virtual machines that is similar but not identical to that of the underlying hardware. http://en.wikipedia.org/wiki/Paravirtualization July 2010.

[15] Full Virtualization. The virtualization technique used to provide a certain kind of virtual machine environment, that is a complete simulation of the underlying hardware. http://en.wikipedia.org/wiki/Full-virtualization July 2010.

 ISBN: 978-1-61208-106-9

[16] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," in *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on*, pp. 59–68, June 2009.

[17] OpenQRM official site: http://www.openqrm.com July 2010.

[18] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the Use of Cloud Computing for Scientific Workflows," in *SWBES 2008, Indianapolis*, December 2008.

[19] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pp. 124–131, May 2009.

[20] B. R. J. Caceres, R. Montero, "Reservoir: An architecture for services,," The first issue of the RESERVOIR Architecture document. http://www.reservoir-fp7.eu/twiki/pub/Reservoir/Year1 Deliverables/080531-ReservoirArchitectureSpec-1.0.PDF, June 2008.

[21] Open Cloud Manifasto, dedicated to the belief that the cloud should be open, http://www.opencloudmanifesto.org/ July 2010.

[22] The Open Source, Open Standards Cloud, Innovative, open source cloud computing software for building reliable cloud infrastructure. http://openstack.org/ July 2010.

[23] Sun Microsystems, Take your business to a Higher Level - Sun cloud computing technology scales your infrastructure to take advantage of new business opportunities, guide, April 2009.

[24] A. Celesti, M. Villari, and A. Puliafito, "Ecosystem Of Cloud Naming Systems: An Approach For The Management And Integration Of Independent Cloud Name Spaces.," in *The 9th IEEE International Symposium on Network Computing and Applications (IEEE NCA10), Boston, USA*, July 2010.

[25] A. Celesti, F. Tusa, M. Villari, and A. Puliafito., "Three-phase Cross-cloud Federation Model: The Cloud Sso Authentication.," in *The 2nd International Conference on Advances in Future Internet (AFIN 2010), Venice, Italy*, July 2010.

[26] E. Bertino, F. Paci, R. Ferrini, and N. Shang, "Privacy-preserving digital identity management for cloud computing," *Computer*, vol. 32, pp. 21–27, March 2009.

[27] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How To Enhance Cloud Architectures To Enable Cross-federation.," in *The 3rd IEEE International Conference on Cloud Computing (IEEE Cloud 2010), Miami, Florida, USA*, July 2010.

[28] DMTF Cloud Management Standards Workgroup Formed to Address Management Interoperability for Cloud Systems. http://www.dmtf.org/about/cloud-incubator July 2010.

[29] VMware vCloud Cloud Computing For Any Application, Any Customer. http://www.vmware.com/products/vcloud/ July 2010.

[30] Telefonica I+D releases the TCloud API for cloud Computing interoperability. April 2010 http://www.tpd.com.br/en/News

[31] Open Cloud Computing Interface WG (OCCI-WG) http://www.ogf.org/ July 2010.

[32] Nagios, The Industry Standard in IT Infrastructure Monitoring: http://www.nagios.org/ July 2010.

[33] Apache ActiveMQ: Open source messaging and Integration Patterns provider, http://activemq.apache.org/ July 2010.

[34] Smack client API, the Open Source XMPP (Jabber) client library for instant messaging and presence. http://www.igniterealtime.org/projects/smack/ July 2010.

[35] Ejabberd, the Erlang Jabber/XMPP daemon, http://www.ejabberd.im/ July 2010.

[36] Libvirt API http://libvirt.org/ July 2010.

 ISBN: 978-1-61208-106-9