An Approach to Enable Cloud Service Providers to Arrange IaaS, PaaS, and SaaS Using External Virtualization Infrastructures

Antonio Celesti, Francesco Tusa, Massimo Villari and Antonio Puliafito
Dept. of Mathematics, Faculty of Engineering, University of Messina
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {acelesti, ftusa, mvillari, apuliafito}@unime.it

***Abstract*—Nowadays, the cloud computing ecosystem is more and more distributed and heterogeneous. Cloud service providers begin to build their services using cloud-based services offered by other service providers. This raises several issues due to integration between services and provider themselves. In this paper, we propose a practice addressing such a concern in a "Vertical Supply Chain" scenario of distributed clouds.**

***Keywords*-Cloud Computing; Vertical Supply Chain; Information Retrieval; Cloud Management.**

I. Introduction

Today, cloud computing represents more and more a tempting business opportunity for ICT operators of increasing their revenues. The success of cloud computing is due to the fact that it offers new business opportunities for both service providers and their clients (e.g., organizations, universities, ICT societies, other clouds, mobile and desktop end-users, etcetera), by means of architectures for delivering Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

A possible classification of cloud architectures can be made according to a three-tier stack [1] including from the bottom: Virtual Machine Manager (VMM), Virtual Infrastructure Manager (VIM), and Cloud Manager. The VMM provides an abstraction for the above layer. It can be a hypervisor (e.g., Xen, KVM, VMware, Virtual Box, Virtual PC, etcetera) running on top of the Operating System (OS) of each physical server composing the cloud's datacenter, and enables the capability of deploying Virtual Machines (VMs). The VIM acts as a dynamic orchestrator of physical server running hypervisor. It is responsible to arrange IaaS defining which VMs have to be instantiated and in which physical server. The main purpose of this layer is to setup VMs regardless of the underlying VMM. Examples of open source solutions are CLEVER [2], OpenNebula [3]. In the end, the CM implements the business logic of a cloud service provider also addressing security and Quality of Service (QoS). It is responsible to arrange "specific" IaaS, PaaS, or SaaS, related to a target business, on top of "generic" IaaS provided by the underlying layer (an open source example is Claudia [4]). Moreover, currently, also hybrid solutions are available implementing both the VIM and CM layers, i.e., Nimbus [5], Eucalyptus [6],

Commonly, these three layers are implemented within an administrative domain in a monolithic configuration and are managed by a single cloud service provider, nevertheless, as stated by Thomas Bittman of Gartner, the evolution of the cloud computing market is switching from a "Monolithic" stage 1 , where cloud providers are based on isolated proprietary architectures to a "Vertical Supply Chain" stage 2, where cloud providers are able to leverage cloud services from other providers in order to arrange services in a distributed environment.

The "Vertical Supply Chain" scenario consists of several cooperating small and medium hybrid clouds. This scenario involves several issues such as cloud discovery, autonomic management, security, QoS, service management, and information retrieval. In this paper, we consider a possible scenario of "Vertical Supply Chain" where a cloud service provider is able to arrange its own services using the IaaS(s) provided by other clouds. More specifically, we investigate an approach based on the eXtensible Resource Identifier (XRI) [7] technology enabling a cloud service provider to manage the whole composed services retrieving information about them from the IaaS(s) on which they are deployed.

The paper is organized as follows: in Section II, we propose a scenario of "Vertical Supply Chain" also highlighting the involved issues. In Section III, we introduce XRI, the technology that in our opinion addresses information retrieval in "Vertical Supply Chain" scenarios. A concrete use case adopting XRI is presented in Section IV, also highlighting, in Section V, experimental results. Conclusions are summarized in Section VI.

II. How to Enable a Cloud Service Provider to Use External Cloud Infrastructure Providers

In order to simplify cloud-based service deployment, clouds and their clients need an easy-to-use comprehensive mechanism that allows the description, installation, configuration, and management of complex multi-tiered services.

In general, we could say that the CM implements the business logic of a Cloud Service Provider. Regarding the Service Provider 1 of the Figure 1, it is responsible to arrange a "specific" IaaS, PaaS, or SaaS, on top of "generic" IaaS(s) provided by the underlying layer. The computing

978-0-7695-4461-8/11 $26.00 © 2011 IEEE
DOI 10.1109/SERVICES.2011.92

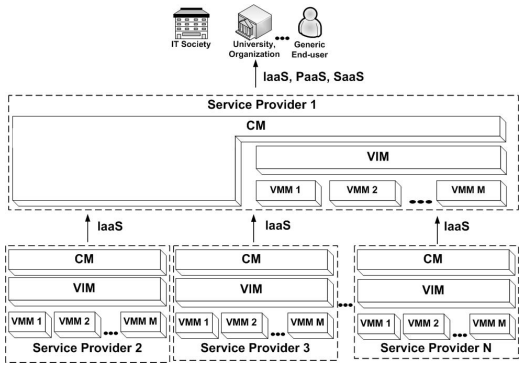

Figure 1. Three-tier stack in a "Vertical Supply Chain" scenario.

infrastructure acting at the VIM layer may belong to the Service Provider itself or could be offered by external Infrastructure Providers (as in the Figure). In order to achieve a "Vertical Supply Chain", we generalized the three-tier stack for meeting the requirement of a business cloud ecosystem in which the Cloud Service Providers "cooperate" on the basis of business agreements.

In Figure 1 we depict a Cloud Service Provider (Service Provider 1) aiming to supply IaaS, PaaS, and SaaS to its clients employing the computing infrastructure made available by its datacenter and/or multiple external Cloud Service Providers (Service Providers 2, 3, ..., N): as the Figure shows, in order to enable high scalability (and efficiency) of the offered services, Service Provider 1, which directly supplies services to the Clients, exploits, in turn, the IaaS(s) made available from Service Providers 2, 3, ..., N (e.g., by mean of web services gaining resources as needed).

In the scenario pointed out in Figure 1, requests for service allocation received from Service Provider 1 are processed and eventually forwarded to Service Providers 2, 3, ..., N, where are caught at CM layer (which acts as interface between the "world" and the infrastructure) and sent to the underlying VIM layer. This latter will offer the ability of deploying VMs on which the IaaS, PaaS, and SaaS will be executed according to a dynamic and scalable fashion: each VIM, in order to satisfy the service allocation coming from the overlying CM, will exploit the capabilities of specific hypervisors acting at VMM layer, thus creating an abstraction for the physical datacenter(s).

The scenario we are considering might be used to address situations in which a Cloud Service Provider acting both at CM and VIM layers has expired its computational capabilities, and it is not able to satisfy further service requests anymore. Such a scenario is depicted in Figure 2.

Provider A acts both at CM and VIM layers, so that it offers IaaS, PaaS, and SaaS. It wants to build a SaaS on top of a IaaS, but it does not have enough resources to arrange the VMs composing the required IaaS. Thus, in order to satisfy the SaaS allocation requests, it contacts respectively Cloud Infrastructure Provider B and C. Therefore the SaaS will be built on top of the IaaS provided by its own Infrastructure Provider using also the IaaS(s) provided respectively by cloud infrastructure provider B and C. Nevertheless, this leads to a complex management of the resources because the running services will be deployed on independent computing infrastructures.

The scenario we are pointing out is more complex than a traditional "monolithic" one, so that a smart, improved coordination system is needed for enabling the cooperation of the different involved Service Providers. In this new cloud perspective, together with the traditional internal monitoring mechanisms of each Service Provider, the service metering information became important both for accounting, billing and service placement decisions among the involved Service Providers: through this approach, each Service Provider is able to achieve an efficient use of the hardware resources, minimizing costs consequently.

More specifically, as depicted in Figure 2, when the hardware resources of one Service Provider are insufficient, or it is necessary to re-arrange the service distribution for meeting the Service Level Agreement (SLA) scheduled with the Clients, the CM middleware exploits the resources coming from different VIM middleware belonging to other different Service Providers.

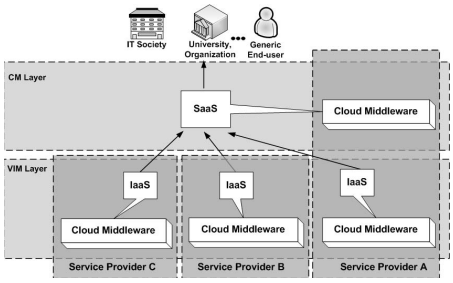

Figure 2. Use case of service composition in a "Vertical Supply Chain" scenario.

In order to achieve and implement this scenario, some issues have to be addressed and some new features have to be included within the VIM middleware. The CM middleware, in fact, must be able to interact with the underlying VIM middleware regardless of their implementation, in order to monitor the state of the computing infrastructure and consequently send specific commands to them. In particular:

- it needs a specific mechanism for tracing the state of the requested services and maintain information about where they are deployed (i.e. the specific VIM middleware on which a service is deployed);
- it should be able to access the VIM middleware services using a "common interface" that allows to send generic commands to the VIM (e.g. start, stop and destroy

VMs).

- In order to gain knowledge about the resource allocation state of each VIM middleware, a unified method for describing such an information, regardless the VIM implementation, has to be employed.

A solution able to fulfill these requirements in a cloud scenario does not exist yet and finding it is not trivial at all. Section IV describes our proposal for enabling the cooperation of the different Service Providers.

III. The XRI Technology

In this Section, after a brief description the XRI technology, we motivate how it can help to address *aaS information retrieval in "Vertical Supply Chain" scenarios.

A. XRI and XRDS

The XRI protocol provides a standard syntax for identifying entities, regardless any particular concrete representation. The XRI system is similar to DNS, including a set of hierarchical XRI authorities but more powerful because it is compatible with any URN domain. XRI enable organization to logically organize entities building XRI tree. According to the XRI terminology, each entity in the tree is named authority. An example is xri://@XYZ*(xri://ABC*development) indicating that the development branch of the XYZ organization is managed by the ABC organization through cross-reference between the XRI naming system of XYZ and the one of ABC.

An authority is resolved by means of an XRDS document representing a simple, extensible XML resource description format standard describing the features of any URI, IRI, or XRI-identified entity in a manner that can be consumed by any XML-aware system. Each XRDS describes which types of information are associated to an authority and the way in which they can be obtained. Using HTTP, XRI resolution involves two phases: authority resolution which is the phase required to resolve a XRI into a XRDS document from an XRI Authority Resolution Server (ARS), and Service End-Point Selection which is the phase of selection of the SEP server (e.g., web service, service provider, web application) returning the data describing the entity in a given context. The same SEP server can also return different data of the same authority.

B. Why Does XRI suit to Cloud Computing?

XRI meets the requirements of cloud computing in a "Vertical Supply Chain" scenario because it can be adopted to develop a seamless mechanism for retrieve data regarding composed *aaS. As XRI is compatible with IRI naming systems, there is not the need to use a unique global naming system, even though this would be possible. This feature allows clouds to manage their own XRI naming systems, mapping them on the global DNS maintaining the compatibility with the existing naming systems. Moreover, with XRI, a cloud can keep one tree representing IaaS, PaaS, and SaaS. In addition, such a technology can be used for both identify and resolve VMs and whole *aaS by means of the resolution of XRI authorities. For example the cloud service provider may need to retrieve three types of information about an authority representing a VM, resolving it in three different ways. In the first way the VM has to be resolved by means of general data (e.g., CPU, memory, kernel, operating system), in the second way the VM has to be resolved by means of real time performance data (e.g., amount of used CPU and memory used), in the third way instead the VM has to be resolved by means of real time data regarding an internal running application (e.g., the percentage of processed data). Such a situation can be addressed by mean of three different XRDs inside the XRDS document corresponding to the VM, authority, each one pointing to a target SEP server.

IV. Service Arrangement Using XRI

In this Section, after a description on how the XRI technology may be used for addressing the issues debated in II, using the XRI authority resolution server provided by the OpenXRI project [7], we are going to simulate the XRI operations needed in a Vertical Supply Chain scenario, evaluating their performance.

A. Use Case Overview for Highlighting the XRI use

In order to clarify the ideas on how XRI can help the management of Vertical Supply Chain scenarios, we are going to consider a possible concrete business scenario involving a Service Provider A and External Service Providers able to leas resources. Let us suppose that a Cloud Service Provider (SP) named *"Video Cloud SP"* supplies a SaaS (e.g. video transcoding, video streaming, etcetera), building it on top of an IaaS consisting of several VMs running within its virtualization infrastructure, managed by a VIM middleware. Using the offered SaaS application, a Client will be able to exploit an high-level performance transcoding service because the Video Cloud SP (behind the scenes) parallelizes the video transcoding process, distributing the video source data to be processed among different VMs.

Let us suppose that the Video Cloud SP runs out of its virtualization resources: in order to go on providing video transcoding SaaS, it may exploit a Vertical Supply Chain using the IaaS(s) offered by external Service Providers. As depicted in Figure 3, the video transcoding SaaS is built on top of three different IaaS: the first one is provided by the Virtualization Infrastructure of Video Cloud SP itself, whereas the second one and the third one are respectively supplied by the Virtualization Infrastructures of Providers B and C. This implies that, besides the information about its own IaaS, the Video Cloud SP must be also aware about the status of the other IaaS(s). E.g., for each IaaS, it has to know

how many VMs have been arranged and the percentage of completed tasks within each of them.

B. *aaS Data Retrieval Using XRI

The Video Cloud SP needs a method for describing the logical organization of its own running services and the different instances: this task can be accomplished using the XRI tree.

As depicted in Figure 3, in order to logically manage their services and retrieve associated information, Video Cloud SP, Provider B and Provider C use their own XRI Authority Resolution Servers and several SEP. More specifically, Video Cloud SP is represented by the XRI authority *xri://@VideoCloudSP*, whereas at the underlying level are mounted several authorities including **v_trans* and **v_str* respectively representing video transcoding and video streaming services. Moreover, considering **v_trans*

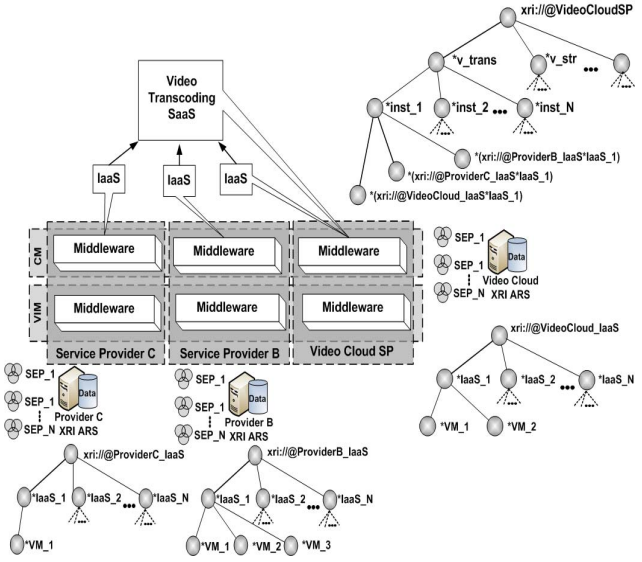

Figure 3. Video Cloud Service Provider using External Virtualization Infrastructures

the underlying mounted authorities identify the different instances for that service. In turn, each instance may be built on top of one or more IaaS(s). For example, the instance represented by the XRI authority **inst_1* is arranged on three different IaaS(s): one hosted in the Virtualization Infrastructure of the Video Cloud SP itself, whereas the others are placed in the external Service Providers B and C.

As for the SaaS(s) of Video Cloud SP, Provider B, Provider C, and Video Cloud Itself, logically organize their IaaS(s) mapping them on their own XRI trees. The Video Cloud SP, using its own XRI ARS, builds another XRI tree, with *xri://VideoCloud_IaaS* authority root, representing its IaaS(s). At the underlying level are mounted the authorities representing the different IaaS instances, e.g. the authority *IaaS_1* describes the used resources, and the underlying mounted authorities **VM_1* and **VM_2* represents the VMs on which the IaaS instance is deployed. More specifically, the paths to resolve the two VMs respectively are: *xri://VideoCloud_IaaS*VM_1* and *xri://VideoCloud_IaaS*VM_2*. Similarly, Providers B and C logically organize their own IaaS instances using their own XRI ARSs: e.g., in the representation the IaaS instance of Provider B *xri://ProviderB_IaaS*IaaS_1* is deployed on three different VMs represented by the XRI authorities:

- *xri://ProviderB_IaaS*IaaS_1*VM_1*
- *xri://ProviderB_IaaS*IaaS_1*VM_2*
- *xri://ProviderB_IaaS*IaaS_1*VM_3*

In the same way, considering Provider C, the authority *xri://ProviderC_IaaS*IaaS*VM_1*, represents the VMs on which *xri://ProviderC_IaaS*IaaS_1* is deployed.

The advantage of using XRI is that it allows one to create an abstraction from the implementation of the CM and VIM layers. Considering the aforementioned video transcoding SaaS(s) the Video Cloud SP, at CM layer, needs a method to manage the instance identified by the authority *xri://@VideoCloudSP*inst_1*. More specifically, the Video Cloud SP has to know the IaaS(s) on which the instance of the video transcoding SaaS is deployed (both its own IaaS and the IaaS(s) supplied by other Providers). Thanks to XRI, this is possible using cross-reference mechanisms, mounting under the *xri://@VideoCloudSP*inst_1* the cross-reference authorities: **(xri://VideoCloud_IaaS)*, **(xri://ProviderB_IaaS*IaaS_1)* and **(xri://ProviderC_IaaS*IaaS_1)*. In this way, resolving these cross-reference authorities by means of appropriate SEPs, the Video Cloud SP is able to know the VMs where each IaaS is deployed, retrieving for each VM information such as the IP address, the percentage of used CPU, the amount of used memory, the amount of used internal storage, and the status of the running process associated to the SaaS (i.e., in our example the percentage of completed transcoded fragment of video). Thanks to such an information, the Video Cloud SP will be able to build and update in real time the web graphical interface for the end-user of the video trascoding SaaS.

V. Cross-Mounting Performance Evaluation

In this Section we are going to analyze the experimental results collected from a set of simulation we performed using the ARS made available from the open source project OpenXRI [7]. Our idea was to reproduce an environment able to fit the Vertical Supply Chain scenario we discussed, considering different arrangements of the XRI authorities, even reproducing “extreme” functioning conditions.

The series of tests executed (50 runs for each simulation, performed using the JMeter tool) guarantee a wide coverage of possible results. The confidence interval (at 95%) depicted in all graphs indicates the goodness of our analysis. Our

testbed has been deployed inside a system running a Redhat Enterprise Linux AS Release 3.0.8 operating system having the following hardware features: Blade LS21 AMD Opteron Biprocessor Dual Core 2218 2.6GHz 8GB RAM.

To perform this analysis and estimate the workload of OpenXRI ARS 1.2.1 we submitted cross-mounting requests using JMeter and evaluated the Response Time of the system expressed in *msecs*. We have measured the time interval between the request phase to the system at Ts and the response time at Tr taking place in the receiving phase. In our graphs we reported the total time spent to accomplish each task: $Tt = Ts + Tr$. The exchange of requests and responses is measured in a local network (LAN, without any Internet connection), since the measurements are not affected from the network communication parameters (e.g., throughput, delays, jitter, etcetera).

All the collected results have been statistically analyzed and presented schematically in Figures 4 and 5. These latter consider the response time of cross-mounting tasks in both wide and deep XRI tree structures. Figure 4 depicts the response time of authority cross-mounting tasks considering the wide tree structure. On the x-axis we have represented the number of nodes, whereas on the y-axis we have represented the response time expressed in milliseconds. For this experiment, we have response times that ranges from about 1 second to less of 2 seconds considering 10, 100, and 1000 authorities, a result very good considering cloud computing environments.

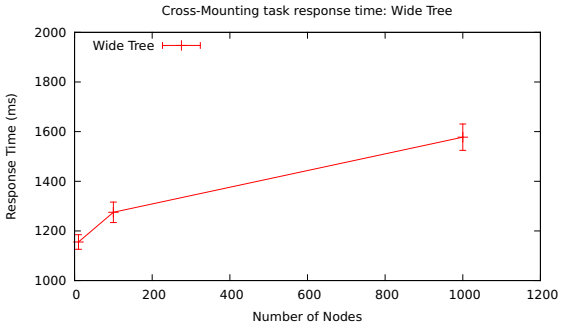

Figure 4. Cross-Mounting task response time: "Wide Tree".

In the end, Figure 5 depicts the response time trend of authority cross-mounting tasks considering the deep tree structure. On the x-axis we have represented the number of levels, whereas on the y-axis we have represented the response time expressed in milliseconds. In such an experiment, considering 10, 20, and 30 tree levels, we can observe how the response time is indeed negligible: less of 0.2 seconds.

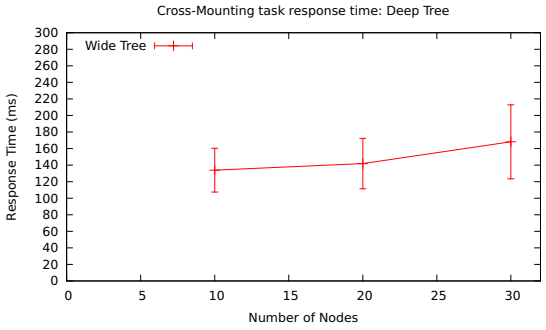

Figure 5. Cross-Mounting task response time: "Deep Tree".

For the aforementioned reasons we can state that cross-mounting tasks are not dependent from the structure of the tree (wide or deep).

VI. Conclusions and Remarks

In this paper we tackled the integration between heterogeneous clouds which build their services using the services supplied by other clouds. We proposed a practice enabling a cloud to manage their composed services retrieving data about the IaaS(s) on which such services are deployed. Moreover, considering the OpenXRI ARS several experiments have been made. Information retrieval about composed *aaS is only one of the possible issues in the "Vertical Supply Chain" topic. We hope we succeed to stimulate your interest in further contributing about this topic.

References

[1] 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, September 2009.

[2] F. Tusa, M. Paone, M. Villari, and A. Puliafito., "CLEVER: A CLoud-Enabled Virtual EnviRonment," in *IEEE ISCC '10. Riccione*, June 2010.

[3] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," in *IEEE HPCC '09*, pp. 59–68, June 2009.

[4] MORFEO Claudia, http://claudia.morfeo-project.org/wiki/index.php/Main_Page.

[5] 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.

[6] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *IEEE CCGRID '09*, pp. 124–131, May 2009.

[7] OpenXRI Project, XRI applications and libraries, http://www.openxri.org/.