J Internet Serv Appl (2011) 2:191–205
DOI 10.1007/s13174-011-0038-7

SI: CLOUD COMPUTING

An XRI naming system for dynamic and federated clouds: a performance analysis

Antonio Celesti · Massimo Villari · Antonio Puliafito

Received: 19 November 2010 / Accepted: 19 September 2011 / Published online: 15 October 2011

Abstract Cloud platforms are dynamic, self-optimizing, continuously changing environments where resources can be composed with other ones in order to provide many types of services to their users, e.g., companies, governments, organizations, and desktop/mobile clients. In order to enable cloud platforms to manage and control their assets, they need to name, identify, and resolve their virtual resources in different operating contexts. In such a scenario, naming, resource location, and information retrieval raise several issues regarding name space management. This paper aims to propose a standard practice for the implementation of a cloud naming system based on the eXtensible Resource Identifier (XRI) technology. More specifically, by means of the development of a Cloud Name Space Management (CNSM) front-end interacting with the OpenXRI architecture, we investigate its performance simulating typical cloud name space management tasks.

Keywords Cloud computing · Cloud name space · Cloud naming system · Cloud federation · XRI · XRDS

1 Introduction

The continuous evolution of cloud computing is allowing to small and medium companies to become more and more competitive on the market [1]. Nowadays, cloud providers supply many kinds of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to their users, e.g., desktop/mobile clients, companies, governments, organizations, and other clouds. Such services can be arranged composing and orchestrating several Virtual Environments (VEs) or Virtual Machines (VMs) through Virtual Machine Monitors commonly known as hypervisors which are spread over a network and orchestrated by Cloud Manager (CM) platforms [2, 40].

The overwhelming innovation of cloud computing is that cloud platforms can react to events internally rearranging the VMs composing their services pushing down management costs, and the interesting thing is that cloud users are not aware of changes, continuing to use their services without interruptions according to a priori Service Level Agreements (SLAs). For example, when a physical server hosting a hypervisor runs out or is damaged, the cloud can decide to move or "migrate" one or more VMs into another server of the same cloud's datacenter acting as virtualization infrastructure. Further migrations can be triggered for many other reasons including power saving, service optimization, business strategy, SLA violation, security, and so far. Another business model which can take place in federated scenarios might be the renting of VMs from a cloud to another. In order to clarify such a concept let us consider two clouds: cloud A and cloud B. Cloud A rents on-demand VMs, instead cloud B needs VMs, so that cloud A rents a VM to cloud B. In this case there is not a real migration, in fact the VM continues to be placed in the virtualization infrastructure of cloud A, but at the same time it is also logically considered part of cloud B.

A cloud computing scenario involves no just cloud-based services and VMs, but also other cloud entities such as physical appliances and cloud users. All these entities need to

A. Celesti (✉) · M. Villari · A. Puliafito
Department of Mathematics, Faculty of Engineering, University of Messina, Contrada di Dio, S. Agata, 98166 Messina, Italy
e-mail: acelesti@unime.it

M. Villari
e-mail: mvillari@unime.it

A. Puliafito
e-mail: apuliafito@unime.it

be named and represented both in human-readable and in machine-readable way. Moreover, they also need to be represented with appropriate data according to a given execution context. For example, as a VM needs to be identified by a name, clouds themselves, cloud administrators, etc. could be interested to resolve that name retrieving either data concerning general information on the VM (e.g., CPU, memory, kernel, operating system, virtualization format version), data regarding the performance of the VM (e.g., CPU and memory used), or by means of Single-Sign-On (SSO) authentication service (e.g., using an Identity Provider (IdP) asserting the trustiness of the VM when it migrate from a place to another in order to avoid identity theft), and many others. This scenario becomes more complex if we consider the fact that these entities might hold one or more names and identifiers also with different levels of abstraction; for the aforementioned concerns the management and integration of cloud name spaces can be difficult.

In order to discourage a possible evolving scenario where each cloud could develop its own proprietary cloud naming systems with compatibility problems in the interaction among different cloud name spaces, this paper aims to propose a standard approach for the designing of a seamless cloud naming system able to manage and integrate independent cloud name spaces also in federated scenarios. To achieve this goal, we propose a practice of cloud naming system architecture using the eXtensible Resource Identify (XRI) [3, 4] technology designed by OASIS [5] and the open source libraries developed by the OpenXRI project [6]. More specifically, we developed a Cloud Name Space Management (CNSM) front-end interacting with the OpenXRI Authority Resolution Server (ARS) 1.2.1 (i.e., similar to the DNS name server). The aim of the front-end is to add new utilities for the management of cloud name spaces in order to enable cloud platforms to manage their own name spaces by means of SOAP web services. Particularly, we focused both on the movement of names inside the same cloud name space and on cross-mounting operations in a federated scenario where the reference of a name placed in a cloud name space is mounted into another cloud name space.

The paper is organized as follows: Sect. 2 describes the state of the art of naming systems in distributed and ubiquitous computing environments. Section 3 provides an analysis regarding cloud name spaces. In Sect. 4, we provide an overview of the XRI technology motivating how, it seems to be one of the most viable solutions to address naming problems. In Sect. 5, we present a practice of XRI architecture for the management of cloud name spaces, also debating how the process of resolution works. An analysis of the performance regarding typical management tasks performed by cloud provider on its XRI-based cloud naming system is presented in Sect. 6. Conclusions are summarized in Sect. 7.

2 Related work and background

Cloud computing is generally considered as one of the more challenging research field in the ICT world. It mixes aspects of Utility Computing, Grid Computing, Internet Computing, Autonomic computing and Green computing [31, 32]. Many authors are trying to describe what it exactly means, in terms of Utility Computing (as the Electricity Model, see [23]), its Economics and Benefits, and what are its Obstacles and Opportunities as the TOP 10 list reports in [21, 22]. Cloud, combined with statistical multiplexing, should increase resources utilization compared to traditional data centers, offering services below the costs of medium-sized-datacenters and still making good profits (see [24]). In such new emerging environments, even though naming and resource location raise several issues, there have not been many related works in literature yet regarding naming systems managing cloud name spaces, and DNS is still erroneously considered the "panacea for all ills." In fact, DNS presents some problems: it is host centric, unsuitable for complex data and services location, and it is not suited to heterogeneous environments. Possible improvements might come from the naming system works in high-dynamic, heterogeneous, and ubiquitous environments. An alternative to DNS is presented in [45]. The authors propose a Uniform Resource Name System (URNS), a decentralized solution providing a dynamic and fast resource location system for the resolution of miscellaneous services. Nevertheless, the work lacks of an exhaustive resource description mechanism. With regard to naming system in ubiquitous computing, in [30] the authors propose a naming system framework for smart space environments. The framework aims to integrate P2P independent cloud naming systems with the DNS, but appears unfitted to be exported in other environments. In addition it aims to localize and identify an entity that moves from a smart space to another using as description mechanism the little exhaustive DNS resource records. A hybrid naming system that combines DNS and Distributed Hash Table (DHT) is presented in [29]. The authors adopt a set of gateways executing a dynamic DNS name delegation between DNS resolver and DHT node.

An interesting survey among different technologies for the Resource Discovery in Grid Environments has been done in [28]. The authors presented a valuable comparison among the P2P protocols ranging from Napster, Gnutella, CAN to Chord. It is interesting to notice the punctual evaluation (even taking into account the complexity of each one) of these protocols and their applicability in Grids. They mentioned that one of the main constrains in Grid is the scalability. Some of the protocols reported above are not really fully decentralized. (i.e., Napster) whereas others do not guarantee the operating in heterogeneous Grid environments. Other evaluations were conducted in [38] and [33]. Their assessments are about the possibility to use in Grid consolidated protocols for the Resource Discovery (RD) tasks. However, many solutions adopted in Grid ([41]) along with the *advanced DHT* usage (see [35]), are not suitable in clouds at all. We can affirm that the level of heterogeneity in Clouds is higher of any Grid infrastructure. For that reason, we cannot consider solutions embraced in Grid, but we have to look at solutions widely used in a distributed system as the Internet (i.e., DNS approach). In our point of view, concepts of systems heterogeneity and federation mechanisms need to be taken into account. Whether we consider the recent convergence of Web SSO systems in the Internet, in the last years, we assisted to a wide use of OpenID [7]. It is considered as one of the widely digital identity protocol used for making *Federation* among web services. Providers that adopted such a technique range from AOL, BBC, Google, IBM, MySpace, Orange, PayPal, VeriSign, LiveJournal, to Yahoo [8]. The new version of OpenID, 2.0 was released to overcome some big issues [9]. The way for improving it is to implement several causals existing in the XRI Standard Specification [3, 4].

We can assume the XRI standard as a step over of the DNS protocol. All enterprises may continue in using their internal systems for cataloging resources and services, as LDAP, Active Directory (AD), owned database, etc.; all these protocols are based on DNS. Our idea is to have an alternative to DNS, a kind of *advanced* DNS protocol, that is, XRI, compliant with URI/URL approach able to overcome DNS limitations, also in terms of its representativeness. We can state that XRI might represent a useful abstraction of what already exists in the Internet. In particular, we remind the XRI syntax and resolution infrastructure was designed explicitly for Internet-scale digital identity, and we are adopting it for enriching exchanged information in much more complex cloud scenarios maintaining its basic philosophy indispensable for the *Federated Digital Identity* management.

Regarding naming, name resolution, and service location in federated cloud environments, in our previous work [27], we highlighted the involved issues both debating a cloud name space analysis and proposing a generic theoretical cloud naming framework for the management of cloud name spaces. The cloud federation is a scenario where clouds establish a relationship in order to benefit new business advantages [25], for example, renting single VM or whole cloud services to other clouds, or when a cloud runs out of its computational and storage capabilities or when a cloud needs a service which is not able to allocate. In [26], considering such a cloud naming framework and several use-cases of the European Reservoir Project [10], we performed an analysis of the problems that such use-cases raise regarding the management of cloud name spaces, also debating how the aforementioned cloud naming framework could be adopted to manage naming and service resolution. As possible representation of the cloud naming framework, we chose XRI [3] and the eXtensible Resource Descriptor Sequence (XRDS) [4] technologies which are also the focus of this work. The aim of this paper is to evaluate the performance of several operational tasks of an OpenXRI Authority Resolution Server (ARS) managing an emulated cloud name space.

3 Cloud name space issues

In this section, we provide a generic analysis of the cloud name space, motivating the emerging need of a seamless cloud naming system.

3.1 Cloud name space analysis

In a highly dynamic federated cloud environment, the design of a cloud naming system could be very hard. First of all, it is not clear which the involved entities and the virtual contexts are. Moreover, in federated environments, a cloud naming system should be able to manage frequent name alteration and name space integration. The following analysis aims to clarify such concerns. Despite the internal cloud structure, we think cloud entities have many logical representations in various contexts. In addition, there are many abstract, structured entities. These entities are characterized by a high-level of dynamism: allocations, changes, and deallocations of VMs may occur frequently. Moreover, these entities may have one or more logical representations in one or more contexts. But which are the entities involved in cloud computing? In order to describe such entities, we introduce the generalized concept of *Cloud Named Entity* (CNE). A CNE is a generic entity indicated by a name or an identifier which may refer both to real/abstract and/or simple/structured entity. As depicted in Fig. 1, examples of CNE may be a cloud itself, a cloud federation, a virtualization infrastructure, a server running a hypervisor, a VM, a cloud service, or cloud users including companies, governments, universities, cloud technicians, and desktop/mobile clients.

In our abstraction, we assume that a CNE is associated to one or more identifiers. As a CNE is subject to frequent changes holding different representations in various *Cloud Contexts* (CCNTXs), the user-centric identity model [20] seems to be the most convenient approach. We define CCNTX as an execution environment where a CNE is represented by one or more identifiers. In this work, we assume a CNE is represented by one or more *CCNTX Resolver Server(s)*, which are servers returning data or services associated to a CNE in a given CCNTX. Figure 2 depicts an example of CNE associated with six identities within four CCNTXs. The target CNE holds identity 1, 2 inside CCNTX A, identity 3 inside CCNTX B, identity 4 inside CCNTX C, and identity 5, 6 inside CCNTX D. We define a *Cloud Naming System (CNS)* a system that maps one or more identifiers

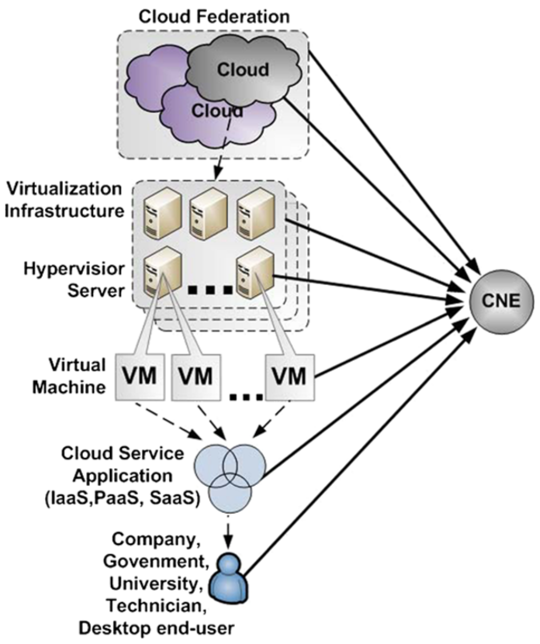
Cloud Federation
Cloud
Cloud
Virtualization Infrastructure
Hypervisior Server
Virtual Machine
VM
VM
VM
CNE
Cloud Service Application (IaaS,PaaS, SaaS)
Company, Govenment, University, Technician, Desktop end-user

Fig. 1 Examples of generic CNEs

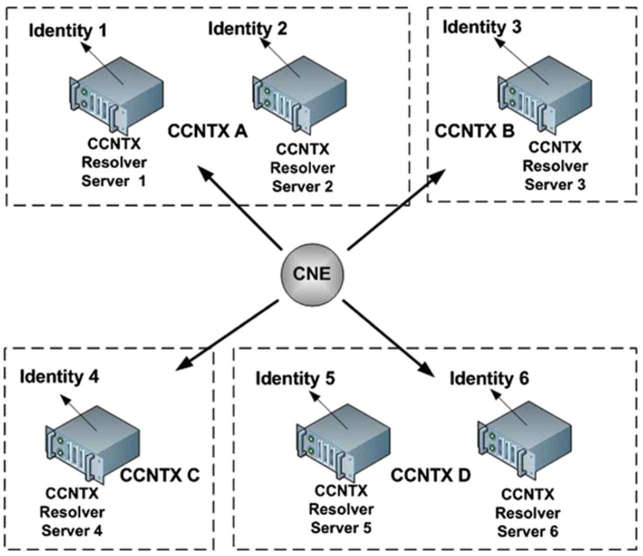

Fig. 2 Examples of a generic CNE associated to several CCNTXs

to a CNE. A CNS consists of a set of CNEs, an independent cloud name space, and a mapping between them. A cloud name space is a definition of cloud domain names. Instead, a name or identifier is a label used to identify a CNE. A client resolver which needs to identify a CNE in a given CCNTX performs a resolution task. Resolution is the function of referencing an identifier to a set of data or services describing the CNE in several CCNTXs.

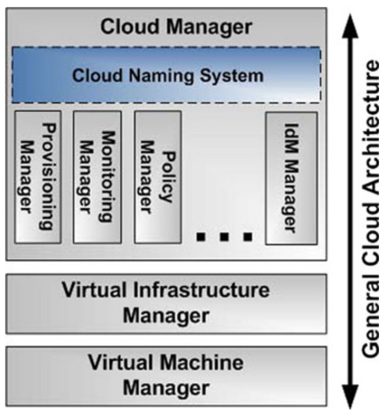

Fig. 3 Cloud naming system in a generic cloud architecture

3.2 The need of a seamless cloud naming system

The cloud name space management raises new challenges concerning: *compatibility*, *scalability*, *extensibility*, *description of CNEs*, *name recycling*, *noncorrelation*, and *name space integration*. As depicted in Fig. 3, the solution to the problem should be a CNS which takes place inside the highest level of a generic three-layer cloud architecture (from the bottom: the Virtual Machine Manager layer, the Virtual Infrastructure Manager layer, and the Cloud Manager layer) [40] compliant with the major existing cloud platforms. As the Cloud Manager layer is responsible for high-level tasks, we think the CNS should provide naming and information retrieval support to monitoring, QoS, security, identity management, federation, billing, and many other tasks. For example, considering a CNE name representing a VM, for monitoring purposes the CNE name should be resolved with the corresponding performance data. Instead, for security purposes, the CNE name should be resolved by means of an IdP service asserting its credentials in a given CCNTX. Such a CNS should include the following main components:

– **A CNS Interface** offering standard APIs to the Cloud Manager layer in order to control the Cloud Name Server.
– **A Cloud Name Server** able to manage the whole cloud name space supporting the mounting, movement, and unmounting of CNE names also in other cloud name spaces, the compatibility with other standard naming system formats, the integration with other CNSs, and the resolution of CNE names by means of the corresponding CNE Descriptors.
– **A CNE Descriptor** describing a generic CNE in several CCNTXs by means of a set of metadata and pointers to one or more CCNTX Resolver Servers.
– **CCNTX Resolver Server** providing data and services representing the CNE in a given CCNTX.
– **A Client Resolver**, the actor which wants to resolve a CNE name. Firstly, it queries the Cloud Name Server obtaining the corresponding CNE descriptor and then it con-

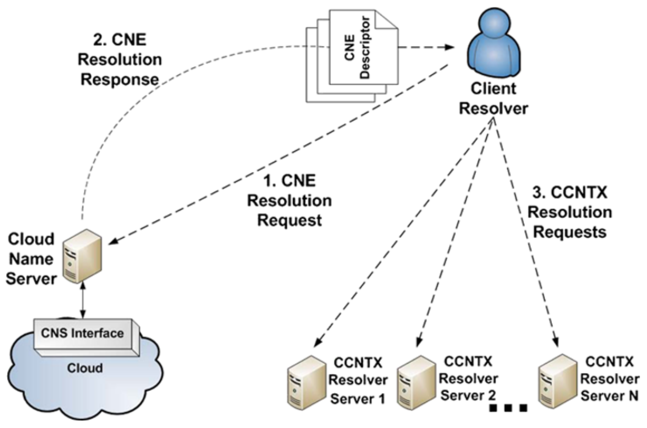

Fig. 4 Example of CNS schema

tacts the CCNTX Resolver Server resolving the CNE in a given CCNTX.

Figure 4 depicts the general schema of such a CNS.

4 The XRI technology for cloud computing

In this Section, after an introduction of the XRI technology, we motivate why it seems to be one of the most viable solutions to address cloud name space management in cloud computing. XRI, defined by the OASIS XRI Technical Committee, is an emerging technology which aims to provide a standard protocol for the identification of generic resources regardless their representations, for both real and totally abstract entities. Nowadays XRI is widely used especially in the security area for the development of new emerging Single Sign-On (SSO) authentication services. Some of the main projects using XRI include OpenID 2.0 [37], Higgins [11], and XDI [12]. As will be argued in the following, we think that such a technology can be adopted even in the cloud name space management field for the development of a seamless CNS.

4.1 Overview on XRI specifications

The design of XRI has started from the solid bases provided by the *Uniform Resource Identifier* (URI) and the *Internationalized Resource Identifier* (IRI), so that URI is extended by IRI and IRI is extended by XRI. As the XRI technology has been developed according to the IRI and URI standards, it inherits lots of the constrains and syntactic rules of them. The URI specification enables to identify resources over heterogeneous and distributed networks, but has the limit to allow only ASCII characters, limit that has been overcome by the IRI specification which considers also *Unicode Character Set* (UCS). XRI solves the problem of the identification of totally abstract resources which was not easy to solve by means of the URI and IRI specifications. The XRI technology is able to solve a wide set of issues:

- Creates a syntax and a protocol for the uniform resolution of real or abstract resources with persistent or reassignable identifiers.
- Provides both generic and trusted protocols to transform an abstract identifier into several concrete identifiers.
- Through a simple standard, it permits the discovery of the URIs associated to a target resource, including the resources requiring additional metadata for their resolutions.
- Provides an uniform syntax for the assignment of identifiers to abstract authority belonging to whatever abstraction level and/or context. An authority in XRI is the identifier of a generic entity.
- Defines a particular entity named “cross-reference” which provides an indexing system to address the resource through different contexts.
- Provides systems which guarantee the privacy of the data associated to an identifier through SAML or HTTPS secure resolutions.
- Permits to extend the system without affects the interoperability.

XRI creates an *Uniform Abstraction Identification Layer* over the *Uniform Concrete Identification Layer* for real resources represented by the IRI, URI, and IRI specifications, so that XRIs and abstract URIs (e.g., URNs) can be resolved to concrete URIs and IRIs. The XRI syntax follows the “Uniform Resource Identifier (URI): Generic Syntax” [13], the “Internationalized Resource Identifier (IRI)” [14], and “Uniform Resource Names (URN)” [15] standards. Basically, an XRI identifier consists of a “xri://” prefix followed by the following syntactic elements:

- The “authority” component indicates a specific name space and it commonly starts with “//” and ends with “/”
- The “/path” component indicates the path to a target authority in a tree.

These components are seen as a superset of the corresponding components of the URI and IRI syntaxes; such a situation allows an easy transformation of lots of URI and IRI identifiers in valid XRI addresses, simply changing the resolution protocol, e.g., from “http://” to “xri://”. As explained in OASIS’s xri syntax version 2.0 [3], the XRI syntax extends the IRI’s syntax on the following aspects:

Persistent and reassignable segments Unlike generic URI syntax, the XRI syntax allows to the internal components of an XRI reference to be explicitly designated as either persistent or reassignable.

Cross-references Cross-references allow XRI references to contain other XRI references or IRIs as syntactically-delimited subsegments. This provides syntactic support for "compound identifiers," i.e., the use of well-known, fully-qualified identifiers within the context of another XRI reference. Typical uses of cross-references include using well-known types of metadata in an XRI reference (such as language or versioning metadata), or the use of globally-defined identifiers to mark parts of an XRI reference as having application- or vocabulary-specific semantics.

Standardized federation Federated identifiers are those delegated across multiple authorities, such as DNS names. Generic URI syntax leaves the syntax for federated identifiers up to individual URI schemes, with the exception of explicit support for IP addresses. XRI syntax standardizes federation of both persistent and reassignable identifiers at any level of the path.

– Global Context Symbol (GCS) Authority
– XRI Authority
– IRI Authority

The GCS authorities have the function to indicate the global logical context of an identifier

```
gcs-authority=  pgcs-authority /
                rgcs-authority
pgcs-authority= "!"
                xri-subseg-pt-nz
                *xri-subseg
rgcs-authority= rgcs-char
                xri-segment
rgcs-char= "=" / "@" / "+" / "$"
```

These authorities are identified by a single character, namely the rgcs-char along with the "!" character (bang character) which is responsible to identify persistent XRI names (i-number). Each rgcs-char indicates a different global context and the most used are "@" for companies and "=" for people. The XRI authorities are other types of authorities which can be formed by means of the GCS authorities themselves or through a cross-reference. The XRI path segment is parsed by an XRI processor in two types of segment: star segment "*" and bang segment "!". The first is for reassignable identifiers, instead the second is for persistent identifiers.

The cross-references is the main tool used by XRI to make flexible and extensible the resources' addressing. Such a mechanism allows the hierarchical addressing or resources and permits to reuse identifiers also in different contexts. From a syntax point of view, a cross-reference is identifiable by means of its delimiters which are "("and")" inside of which it is possible to insert the reference of an XRI or IRI name in absolute form.

```
"(" ( XRI-reference / IRI ) ")" *xri-subseg
```

4.2 Resolution of XRI authorities

Given the multitude of the possible information and addressable resources through the XRI technology, OASIS designers have not defined a unique resolution architecture, but they created a generic format for the description of resources named Extensible Resource Descriptor Sequence (XRDS) and a standard for the request of XRDS document corresponding to a particular XRI name. As well as the DNS, the resolution of XRI identifiers takes place through XRI name server in recursive manner. The resolution can be performed in four different scenarios:

– Local Resolution.
– Local Resolution using recursing authority servers.
– Proxy Resolution.
– Proxy Resolution using recursing authority servers.

Each of the aforementioned scenarios evolves in two phases:

1. *Authority Resolution*. This phase consists of the resolution of the authority or XRI name in a XRDS document which describes a given entity. The resolution depends on the scenario.
2. *Service-End-Point Selection*. This optional phase allows an XRI client resolver to select from the XRDS document the right Service-End-Point (SEP) resolving the XRI name in a given context. The selection of the SEP is performed choosing from the XRDS document a particular eXtensible Resource Descriptor (XRD) including a set of metadata describing the SEP, which actually resolves, the XRI name, and an URI to the server running that SEP.

4.3 Why does XRI suit to cloud computing?

The XRI technology is flexible, interoperable, and extensible and as it meets the requirements of cloud name space management; it can be adopted to develop seamless cloud naming systems. 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 or more name spaces representing its own CNEs by managing one or more XRI schemes. In addition, such a technology can be used for both identify and resolve CNE names by means of the resolution of XRI authorities. For simplicity, from now on, with terms as "CNE name,' "XRI authority," or "XRI authority name," we will refer to the same concept. Moreover, the XRDS document can be used to describe an XRI authority associated to a target CNE, indicating how to resolve the CNE itself in several CCNTXs by means of SEPs acting as CCNTX Resolver Servers.

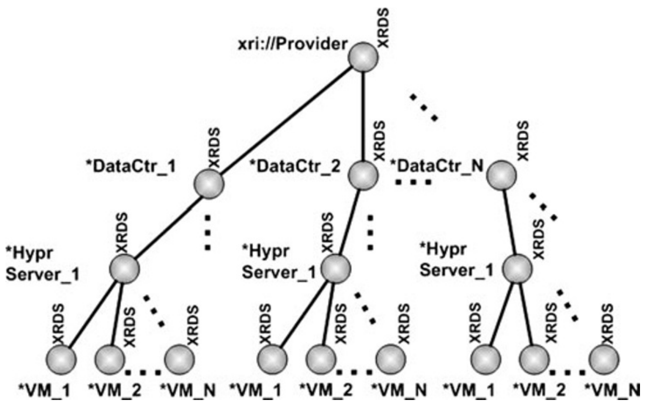

Fig. 5 XRI scheme representing the virtualization infrastructure of a cloud

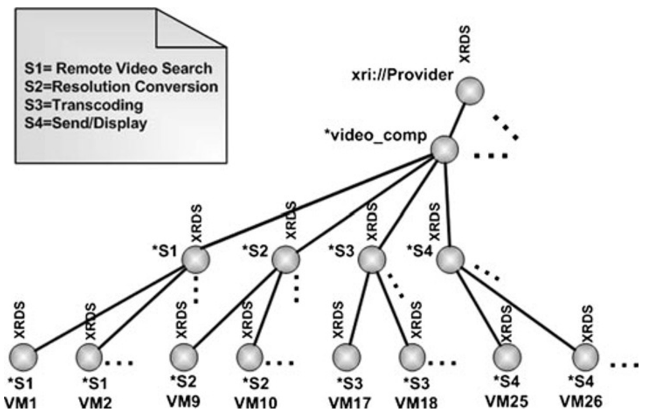

Fig. 6 XRI scheme representing video services, each one running in a different VM

4.3.1 *Representation of cloud-based services through XRI schemes*

As described, XRI is a protocol able to represent any CNE. From now on, in this paper, we will focus on XRI schemes representing cloud-based services. An instance of cloud-based service (e.g., IaaS, PaaS, or SaaS) is an abstract simple/structured object which can be a single VM or a workflow deployed in a number of VMs in the same cloud's virtualization infrastructure or in different federated clouds. Currently, service workflow modeling, composition, and management are widely debated topic in both service oriented computing and cloud computing areas [44]. From the cloud computing point of view, a workflow for the arrangement of SaaS can be seen as a number of cooperating services, each one deployed on a different VM. In this scenario, a cloud provider has to manage the name space including the physical hypervisor servers in which the VMs are running, the VMs themselves, the cloud-based services and their instances. In addition, for each of the aforementioned CNEs, it is needed to retrieve different types of data.

Figure 5 depicts an XRI scheme representing the logical mapping of a virtualization infrastructure including hypervisor servers and VMs. This is a typical internal management use case considering both proprietary providers (e.g., Amazon [16], Rackspace [17]) and open source-based providers (e.g., OpenNebula [39], Eucalyptus [36], and CLEVER [42]) which allow to instantiate on-demand VMs. These providers need to logically map where their VMs are allocated and at the same time to retrieve data about them. Considering proprietary providers, it is not clear how this task can be accomplished. On the contrary, considering open source-based providers, it might be worthwhile to rely on a standard methodology. As shown in Fig. 5, the XRI scheme allows to map in which data-center and in which hypervisor server each VM is hosted.

Figure 6 depicts an XRI scheme representing a provider offering a pool of microservices for the composition of

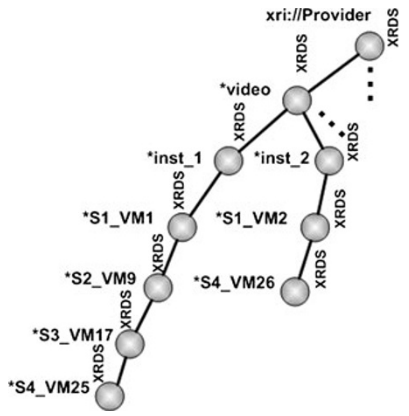

Fig. 7 XRI scheme representing workflow instances of video services

video-service workflow instances. Each microservice has different instances, each one deployed in a different VM. In this schema, as we are considering a single CNS, we assume that the XRI authorities representing the VMs are alias of the VMs of the scheme depicted in Fig. 5, also though XRI cross-references are even possible.

Instead, Fig. 7 depicts the XRI scheme representing the instances of different video-service workflows. For example, instance 1 (*inst_1) represents a cloud-based SaaS performing remote video search, resolution convertion, transcoding, send/display tasks, instead instance 2 (*inst_2) is a cloud-based SaaS simply performing remote video search and send/display tasks.

4.3.2 *Resolution of an authority representing a VM*

The XRI authority representing a VM, for example, can be resolved in three different ways in different CCNTXs. In the fist way the VM has to be resolved by means of general data (e.g., CPU, memory, kernel, operating system, virtualization format version), in the second way the VM has to be resolved by means of instantaneous performance data (e.g. amount of used CPU and memory), in the

```
<XRDS xmlns="xri://$xrds"
ref="xri://@CLOUDA*datacenter-1*host-1*VM3">
        <XRD xmlns="xri://$xrd*($v*2.0)" version="2.0">
                <Query>VM3</Query>
<Status code="100"/>
<ServerStatus code="100"/>
<Expires>2010-10-30T09:30:10Z</Expires>
<ProviderID>xri://@CLOUDA*datacenter-1*host-1</ProviderID>
<LocalID>*VM-Ubuntu-1</LocalID>
<CanonicalID>4161540</CanonicalID>
<Service>
            <ProviderID> ve-info </ProviderID>
            <Path match="null" select="false"/>
            <MediaType match="null" select="false"/>
            <URI append="none">
                        http://cloudA.example.net/res/info/4161540/?
            </URI>
</Service>
<Service>
            <ProviderID> ve-performance </ProviderID>
            <Path match="null" select="false"/>
            <MediaType match="null" select="false"/>
            <URI append="none">
                        http://cloudA.example.net/res/perf/4161540/?
            </URI>
</Service>
<Service>
            <ProviderID>sso-auth</ProviderID>
            <Type>xri://$res*auth*($v*2.0)</Type>
            <MediaType match="null" select="false"/>
            <URI append="none">
                        http://openid.example.net/4161540/?
            </URI>
</Service>
            </XRD>
</XRDS>
```

Fig. 8 Example of XRDS document

third way instead the VM has to be resolved by means of SSO authentication data (e.g., information provided by an Identity Provider (IdP) asserting the trustiness of the VM when it migrates from a place to another in order to avoid identity theft). Such a situation can be addressed by mean of three different XRDs inside the XRDS document corresponding to the VE, each one pointing to a target SEP. Figure 8 shows a simplified example of XRDS document describing a CNE representing a VE. The XRI authority *"VE3"* identifying a CNE "VE" is mounted under the parent XRI authority *xri://@CLOUDA*datacenter-1*host-1*, and has also an unpersistent identifier "VM-Ubuntu-1" (*i*-name) and a persistent "4161540" (*i*-number). As the VE holds three representations in three CCNTXs, each ⟨xrd:service⟩ element indicates how to resolve the VE in a given CCNTX by means of a SEP identified by one or more ⟨xrd:Type⟩ elements. This elements accept a URI, IRI, or XRI to identify the service type. This makes the XRDS format extensible without the need of a central registry. In addition, each SEP can include a set of URIs representing concrete network endpoints where a target service is available. In the example, the VE has three representations in three different CCNTXs. The SEP "ve.info" with URI http://cloudA.example.net/res/info/4161540/? returns general informations regarding the VE. The SEP "ve-performance" with URI, http://cloudA.example.net/res/perf/4161540/? returns performance metadata. The SEP "sso-auth" with URI http://openid.example.net/4161540/? is resolved by an OpenID server [37] asserting the credential of the VE. As the aim of this paper is the evaluation of the performance of an XRI naming system for the management of cloud name spaces, further details regarding XRDS document are omitted.

As far as it is concerned the integration among independent cloud name spaces in a federated environment, the cross-reference mechanism provided by XRI allows to logically mount CNE names belonging to a cloud name space into another one. In order to clarify the ideas, let us consider two cloud platforms: A and B. cloud A rents a VE which is physically placed its own virtualization infrastructure and which is identified by the XRI authority VE3 to cloud B which logically considers such a VE as part of its virtual resource pool along with other VEs physically placed in its own virtualization infrastructure. Using the XRI cross-reference mechanism, the target VE can be at the same time mounted inside the cloud A name space
`xri://@CLOUDA*datacenter-1*host-1*VM3`
and mounted through an XRI cross-reference mounting tasks also in cloud B name space:
`xri://@CLOUDB*resources*`
`(xri://@CLOUDA*datacenter-1*host-1*VM3)`

5 How to manage the cloud name spaces using an XRI architecture

Regarding XRI implementations, the OpenXRI Project, distributed under Apache 2.0 license, is one of the best open source java initiatives which aims to promote the development of XRI-based applications. For this reason, it represents our starting point in the designing of our CNS. Currently, OpenXRI consists of the following packages:

- **SYNTAX**: An XRI parser library;
- **CLIENT**: An XRI resolver library;
- **SERVER-LOGIC**: A library with the business logic of the server;
- **SERVER**: A web application running the XRI Authority Resolution Server (ARS) 1.2.1;
- **ADMIN**: A web application for managing an XRI authority resolution server by means of a graphical administration web interface.

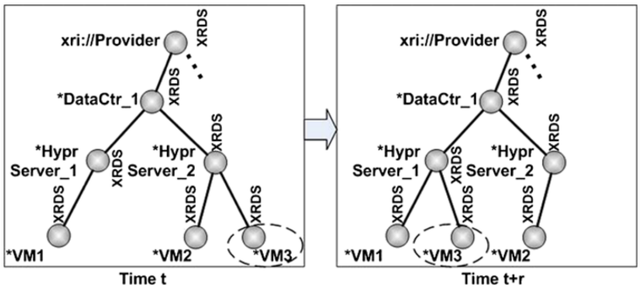

Fig. 9 Example of CNE name movement

5.1 Cloud naming system requirements

The OpenXRI Project, in the "admin" package, provides an administration web interface with limited functionalities allowing an user to interactively manage his/her name space. This administration interface is very penalizing for our scenario, as we assumed that the cloud name space should be also managed automatically by the cloud provider according to its business logic. In fact, there are circumstances where the interaction with an administrator is required and other cases where the cloud has to arrange its assets automatically by itself. For such reasons, we have designed a Cloud Naming System Management (CNSM) front-end offering both a standard SOAP web service interface in order to make the naming system controllable by any cloud platform, and additional specific utilities for cloud name space management. The choice of SOAP is due to the fact that it provides consolidate frameworks for both QoS and security.

As clouds are highly dynamic environments, the logical and physical arrangement of its resources can continuously change. For example, a VM can be physically moved from a hypervisor server to another or can be logically assigned from a hypervisor server to a cloud-based service.

Figure 9 depicts an example CNE movement in an XRI scheme representing the logical organization of a virtualization infrastructure. The CNE movement is due to a migration of a VM named "*VM3" from the hypervisor server "HyperServ_2" to "HyperServ_1" in the same datacenter. This is a very common situation considering a self-optimizing cloud virtualization infrastructure. In fact, migrations can happen for many reasons including service optimization, SLAs constraints, power saving, load balancing, fault tolerance, security, etc. The importance of migration in virtualized environment such as cloud computing, is motivated by the steady rising of many research works which try to bring off better performance [34, 43]. Considering the XRI scheme representing the name space of a cloud, a VM migration implies a CNE movement, i.e., an unmounting task in the source place and a mounting task in the destination place.

XRI also allows to represent federation scenarios where a cooperation takes place between different clouds. A typical

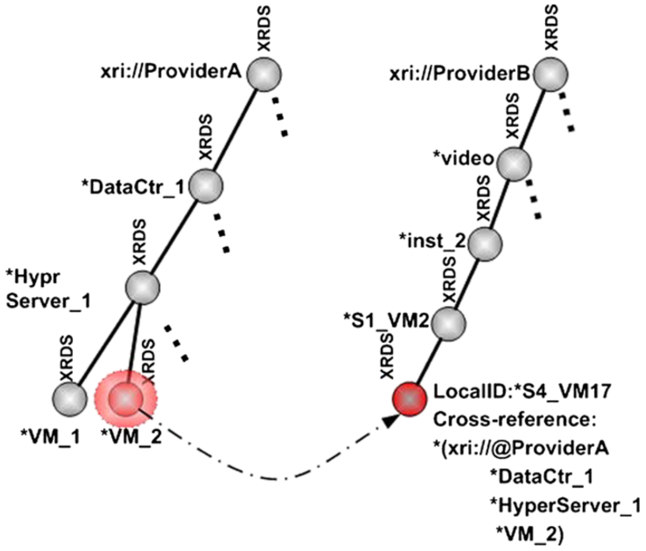

Fig. 10 XRI scheme representing a federation use case

scenario includes, a VM logically assigned from a cloud virtualization infrastructure to a cloud-based service of another federated cloud. This implies a logical mapping and/or integration between the schemes representing the name spaces of the two involved clouds. Thanks to XRI, this is possible by means of cross-reference mechanisms.

Figure 10 depicts an example of resource provisioning between two federated providers, where a cross-mounting take place. More specifically, it consists of the mounting of a cross-reference related to an authority of an XRI scheme in another XRI scheme. In this example, Provider B, in order to instantiate a workflow cloud-based video service, use a VM hosted in Provider A's virtualization infrastructure mounting it in its workflow by means of both XRI aliasing and cross-referencing mechanisms. Thus, the micro-service component will be identified by Provider B as *S4_VM17, but in reality it will be a cross-reference to *VM2 physically placed in Provider A. For these reasons, a CNS should provide both CNE movement and cross-mounting features automatically manageable.

5.2 Overview of our OpenXRI-based CNS practice

Our practice includes the following main components:

– **XRI Cloud Name Space Management (CNSM) front-end**, a SOAP web service interface allowing to a cloud provider managing its name space in XRI ARSs.
– **XRI Authority Resolution Server (ARS) 1.2.1**, a server for the resolution of XRI authorities, i.e., in our scenario, CNE names.
– **XRI Client Resolver (CR)**, a client which resolves a CNE name by means of the ARS, obtaining the corresponding XRDS document and after that performs a "SEP

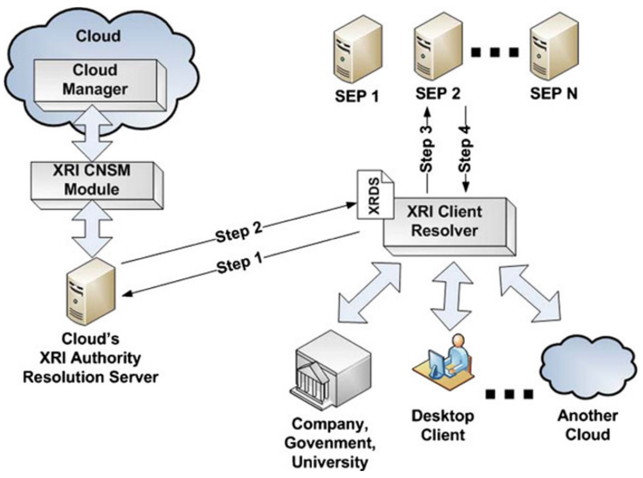

Fig. 11 Example of CNE name resolution

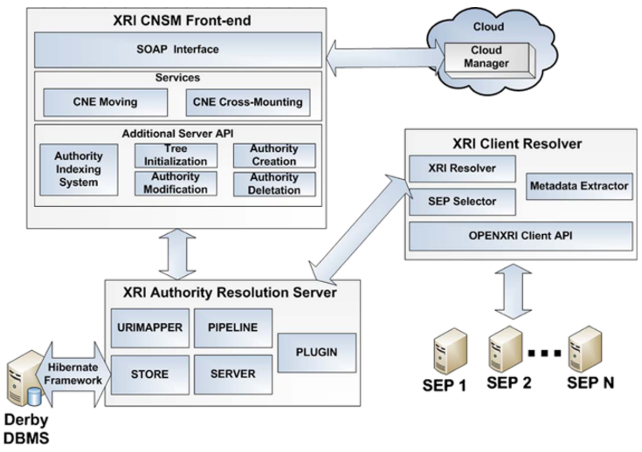

Fig. 12 XRI architecture derived from OpenXRI

Selection Task" choosing the right SEP server for the resolution of the name in a given CCNTX.
– **SEP Server**, a REST web service which resolves a CNE name in a given CCNTX sending data in XML format to the XRI CR. A SEP can be also a server providing a service associated to the CNE name, e.g., a provider of SSO authentication services such as OpenID, an email provider, and so on.

Figure 11 depicts an overall overview of the CNS. On one hand, a cloud provider can manually/automatically administrate its name space by means of both the CNSM front-end and the XRI ARS, performing authority movement and cross-mounting tasks. On the other hand, cloud's clients can perform authority resolution tasks, querying the XRI ARS, hence retrieving data about the CNEs. The process of resolution of CNE names is carried out as follows. In step 1, the XRI CR wants resolver, a CNE name, and contacts the XRI ARS making a resolution request. In step 2, the XRI ARS resolves the name and responds to the XRI CR sending the XRDS document corresponding to the CNE name. In step 3, the XRI CR performs a SEP selection task choosing the server for the resolution of the CNE name in a given CCNTX to which performing a REST web service request. In step 4, SEP server responds with the data resolving the CNE name in the corresponding CCNTX, so that the XRI CR can process the obtained XML data.

5.3 The XRI naming architecture in detail

Figure 12 depicts the XRI Cloud Naming Architecture, highlighting the elements composing each involved component.

5.3.1 The XRI Authority Resolution Server (ARS)

The XRI ARS is an application developed by OpenXRI representing the equivalent for XRI of the DNS infrastructure. The server can host one or more XRI name spaces and its main function is to respond to the name resolution requests sent by the XRI CRs. Moreover, it also supports recursive resolution, synonyms, multiple root namespaces, automatic generation of i-names and i-numbers, and extendibility. The main components of the ARS are:

SERVER, the component responsible to resolve a name making the corresponding XRDS document.
URIMAPPER, the component responsible to parse the received requests picking out the subsegment needed for the resolution of a name.
STORE, the component responsible for the store of the names and of their metadata (the XRD descriptors). In the default architecture, this component interacts by means of the hibernate framework with the Derby DBMS.
PLUGIN, the component responsible to manage the requests that can not be parsed by the URIMAPPER component.
PIPELINE, the component responsible for the creation of XRD descriptors.

5.3.2 The CNSM front-end

As the default ARS provides just a web interface which requires the manual interaction of an administrator, we developed a Cloud Name Space Management (CNSM) front-end, a software allowing to clouds to manage their own name spaces using one or more recursive XRI ARSs. The aim of the CNSM front-end is to provide to clouds several APIs for the automatic creation, modification, and removal of XRI authorities (i.e., CNE names) inside the XRI tree representing the cloud name space. In order to achieve such a goal, at low level, we created several additional APIs using the URIMAPPER, PIPELINE, STORE, SERVER, and PLUGIN components. Compared with the standard ARS, at low level the CNSM front-end provides the following utilities: Authority Indexing System, Tree Initialization, Authority

Creation, Authority Modification, and Authority Deletion. The authority Indexing System represents an utility for the navigation through the XRI tree using a "path" indexing algorithm where the index of each XRI authority corresponds to the XRI address of its father subsegment, which is also the needed path to link the indexing authority with the root namespace of the XRI tree. Tree Initialization allows to initialize the XRI tree using an XML configuration file. This latter utility has been very useful for the arrangement of simulated cloud name spaces for the experiments which will be described in Sect. 6. Authority Creation, Modification, and Deleting represent utilities for the name space management. Using such utilities the front-end provides the following services: "CNE Movement" and "CNE Cross-Mounting."

The CNE Movement service provides a tool addressing cloud scenarios such as the one depicted in Fig. 9. More specifically, it allows to move an authority from a place to another inside the same XRI tree. We can outline the process in the following six steps.

1. Searching of the moving authority.
2. Searching of the destination authority under which the moving authority has to be mounted along with all its children.
3. Searching of the moving authority's children.
4. Creation and cloning of the moving authority.
5. Recursive coping of the moving authority's children in the destination places.
6. Deleting of the authority along with its children in the old place in the XRI tree.

Instead the CNE Cross-Mounting service helps the management of name spaces allowing to logically mount cross-reference referred to an authority of an XRI scheme into another XRI scheme. This feature is very useful allowing to address resource assignment and service-composition also in federated scenarios.

Both low-level utilities and the two aforementioned cloud service are accessible by cloud platforms or other systems through a SOAP web service interface. The SOAP technology has been chosen because it provides an whole framework supporting both Security and QoS, topics which are out of the scope of this paper.

5.3.3 The XRI Client Resolver (CR)

The XRI CR allows to a user to perform the resolution of an XRI name, obtaining the XRDS document describing the authority from which extracting the URI of the SEP server resolving the CNE name in a given CCNTX. It uses at low level the XRI resolver library of the OperXri Project. More specifically. the XRI CR resolves a name obtaining the XRDS document by means of the "Resolver" component. After that, as the XRDS is a XML document, the "SEP Selector" analyzes the document by means of a "DOM" associated to the descriptor selecting the various SEPs entries resolving the XRI authority in several CCNTXs. For each entry, the "Resolver" extracts the metadata and the URI to the physical server which actually resolve the name in a CCNTX. If the "Metadata Extractor" component finds an entry matching he desired resolution, it sends a request to the SEP server indicated by the URI, and if the that SEP server responds correctly, it extract the metadata from the response. Considering a VM such metadata can be related to general information, performance, and other associated services.

5.3.4 Security considerations

Security, is a hot topic in cloud computing. For this reason, even though security is not the focus of this paper is worthwhile to spend a few words about the trustiness of the proposed CNS. In order to make the system trustworthy, it is needed to secure three tasks: CNE name (i.e., XRI authority) resolution, SEP Server information retrieval, and CNS management. Regarding the CNE name resolution performed by an XRI CR, consisting of retrieving the associated XRDS document from the XRI ARS, the XRI specification natively supports secure resolution by means of either Security Assertion Markup Language (SAML) [18] or https. Instead, considering the information retrieval from the Rest-based SEP Server, security can be accomplished using https. However, as SEP Servers can be developed using other technologies, also trustworthy can be accomplished through other solutions. In the end, the interaction between the cloud middleware and the XRI CNSM front-end, for the management of the CNS, can be easily secured using the WS-Security framework [19] of the SOAP protocol.

6 Authority moving and cross-reference performance

In this section, we discuss several experiments we conducted on a real XRI-based CNS testbed. More specifically, we assumed a scenario in which a cloud provider manages its own name space by means of both the CNSM front-end and an XRI ARS, focusing on the administration costs due to both authority movement and cross-mounting tasks. The evaluations have not regarded the mere use of OpenXRI, but the main functionalities developed in the CNSM front-end, i.e., authority movement and cross-mounting. Both have been developed using at low level the APIs of the OpenXRI Project.

We stored XRI schemes representing the cloud's virtualization infrastructure and service workflows, considering both physical and logical changes. For clarity, we remark that a physical change can happen when a VM migrates from a hypervisor server to another, whereas a logical change can happen when a VM placed in a cloud virtualization infrastructure is assigned to a cloud-based service workflow. Considering the XRI CNS, a physical change in the cloud implies a CNE movement, instead a logical change implies a cross-mounting task. In the experiments, we evaluated the time needed for the completion of both the CNE movement and cross-mounting tasks.

In order to evaluate the goodness of the OpenXRI-based CNS, it is necessary to understand the overall behavior of the architecture under particular conditions of workload, hence we decided to stress the operations of both XRI authority movement and cross-mounting considering two possible use cases: “wide tree” (considering 10, 100, and 1000 leaves) and “deep tree” (considering 10, 20, and 30 levels).

Considering the wide tree case, in order to highlight the behavior under different configurations, we identified three possible sub-cases: “subcase 1” consisting of a wide tree with two levels; “subcase 2” consisting of a wide tree with three levels where each node can have no more of one leaf; “subcase 3” consisting of a wide tree with three levels where each node can have one or more leafs. For each of the aforementioned three subcases of the wide tree, we performed three different types XRI authority movement:

– **Movement 1**. It consists in the movement of the last authority on the right under the first authority on the left.
– **Movement 2**. It consists in the movement of the last authority on the right under the authority placed in medium position of the tree.
– **Movement 3**. It consists in the movement of the last authority on the right under an authority placed in a random position of the tree.

Regarding the testing of the cross-mounting task, we performed similar experiments considering as wide tree the subcase 3. Analogous experiments were performed with the deep tree.

As depicted in Fig. 13, our experiments have been performed considering a testbed with a client/server architecture including three tiers: the CNSM front-end, the XRI ARS 1.2.1, and the Derby DBMS which have been deployed inside a computer 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 2.6 GHz 8 GB RAM. The experiments have regarded only the XRI CNS and in order to emulate the cloud provider managing its name space, we used JMeter, an open source automatic client tool, which has been deployed within another computer. The environment was characterized by the following installations: Glassfish v2 for the hosting of both the OpenXRI ARS 1.2.1 and the CNSM front-end; Apache Derby for the hosting of the database used by the OpenXRI ARS 1.2.1; JMeter and Mozilla Firefox as web clients.

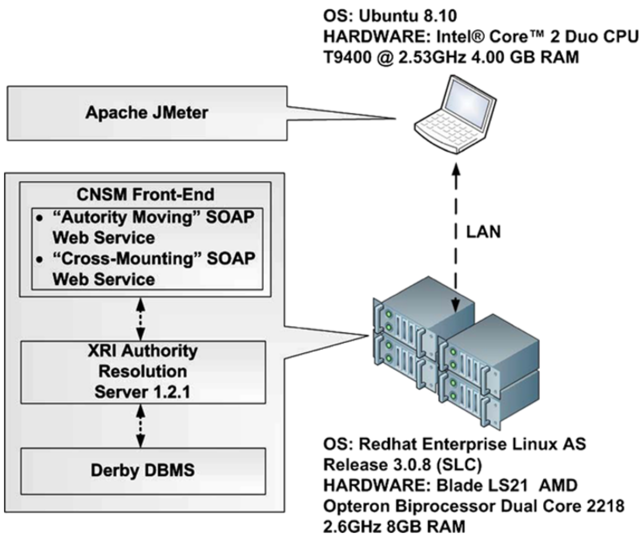
OS: Ubuntu 8.10
HARDWARE: Intel® Core™ 2 Duo CPU T9400 @ 2.53GHz 4.00 GB RAM
Apache JMeter
LAN
CNSM Front-End
• “Autority Moving” SOAP Web Service
• “Cross-Mounting” SOAP Web Service
XRI Authority Resolution Server 1.2.1
Derby DBMS
OS: Redhat Enterprise Linux AS Release 3.0.8 (SLC)
HARDWARE: Blade LS21 AMD Opteron Biprocessor Dual Core 2218 2.6GHz 8GB RAM

Fig. 13 Testbed specifications

All the conducted analyses are very interesting because they show how the complexity of the system does not affect its performance during different actions on the XRI schemes.

6.1 Experimental results

To estimate the workload of OpenXRI ARS 1.2.1 controlled thorough the CNSM front-end, we evaluated the Response Time in *msecs* for both authority moving and cross-mounting tasks. We have measured the time interval between the request phase to the system at T_s and the response time at T_r, taking place in the receiving phase. In our graphs, we reported the total time spent to accomplish each task: $T_t = T_s + T_r$. 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). The series of tests executed (50 runs for each simulation) guarantee a wide coverage of possible results. The confidence interval (at 95%) depicted in all graphs indicates the goodness of our analysis.

6.2 Evaluation of authority moving tasks

Considering the wide tree, Figs. 14, 15, and 16 depict the response time trend of each of the three authority movement types for each of the three tree subcases, respectively, for 10, 100, and 1000 authorities. On the x-axis, we have represented the three types of movement, whereas on the y-axis we have represented the response time expressed in milliseconds. Regarding these experiments, it is important to point out that the carried out measurements did not take into account the time needed for the deletion of a moved XRI authority in its previous place (step 6 of Sect. 5.3.2), because such a time can be neglected.

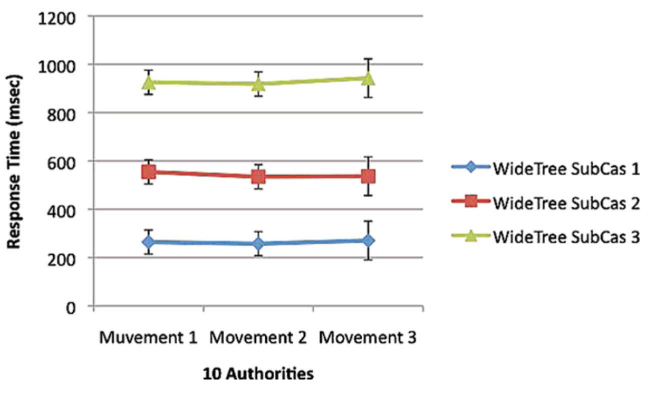

Fig. 14 Authority movement tasks considering different wide tree sub-cases and 10 nodes

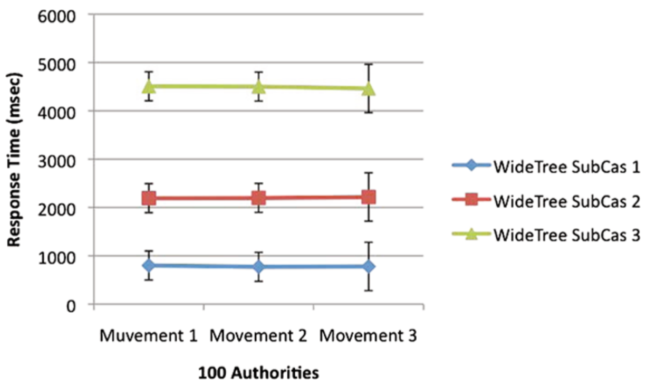

Fig. 15 Authority movement tasks considering different wide tree sub-cases and 100 nodes

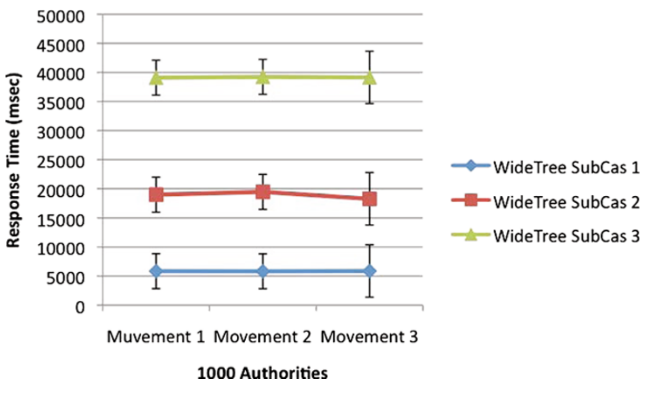

Fig. 16 Authority movement tasks considering different wide tree sub-cases and 1000 nodes

The curves with small circle shapes are related to the wide tree subcase 1), the curves with small rectangle shapes are related to wide tree subcase 2, and finally the curves with small triangle shapes are related to wide tree subcase 3. The graphs highlight that the authority movement task is independent from the starting and ending position within the tree structures, but it is strongly dependent from the type of tree structure (see subcases 1, 2, and 3). Moreover, as shown in the graphs depicted in Figs. 14, 15, and 16, the wide tree subcase 3 represents the worst case because the processing time is higher than the other ones. According to this latter consideration, Fig. 17 summarizes the response time trend regarding the authority "movement 3" tasks using the "subcase 3" tree structure and considering 10, 100, and 1000 nodes. On the x-axis, we have represented the number of considered nodes, whereas on the y-axis we have represented the response time expressed in milliseconds. Observing the graph, we notice that with 1000 nodes we have a response time of 40 seconds, a rather high value, but reasonable considering the presence of 1000 operating VMs. Instead, in the case of 10 and 100 nodes, the subcase 3 needs a time that ranges from about 1 to 5 seconds in order to perform authority movement 3 tasks. This latter results are reasonable in cloud contexts, in particular, if we assume a scenario where a single VM needs to be boot up, from an unrunning state. Usually, the time needed for the VM boot-up is higher than the time spent by OpenXRI ARS for any type of tree reconfiguration.

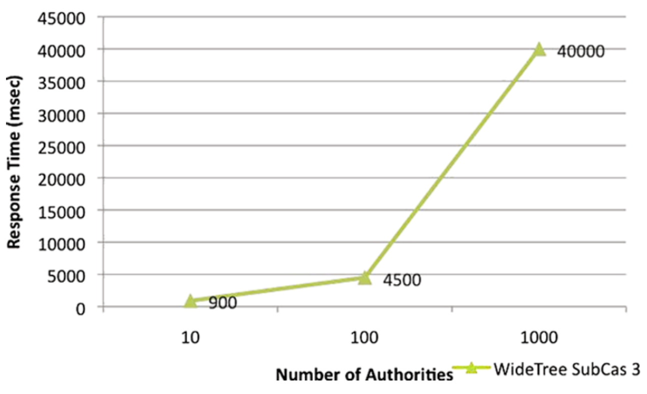

Fig. 17 Authority movement tasks in the wide tree subcase 3 with 10, 100, and 1000 nodes

The graph depicted in Fig. 18 shows the response time trend concerning authority movement tasks from a place to another of the deep tree, considering 10, 20, and 30 tree levels. 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. Observing the graph, the worst case (an authority movement within a tree with 30 levels) implies 12 seconds. In reality, the case under analysis may be considered as an event with a low probability in cloud computing environments. In fact, it represents an hierarchical structure with 30 levels in which we should identify 30 CNEs with hierarchical relationships. However, with a few levels, and with our hardware configuration the response time we can achieved is rather low.

6.3 Evaluation of cross-mounting tasks

The response time of cross-mounting tasks in both wide and deep tree structures is slightly less relevant. Figure 19 depicts the response time of authority cross-mounting tasks

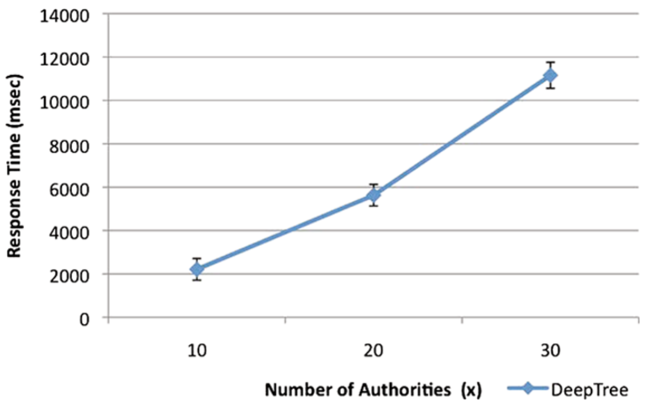

Fig. 18 Authority movement tasks in a deep tree case with 10, 20, and 30 levels

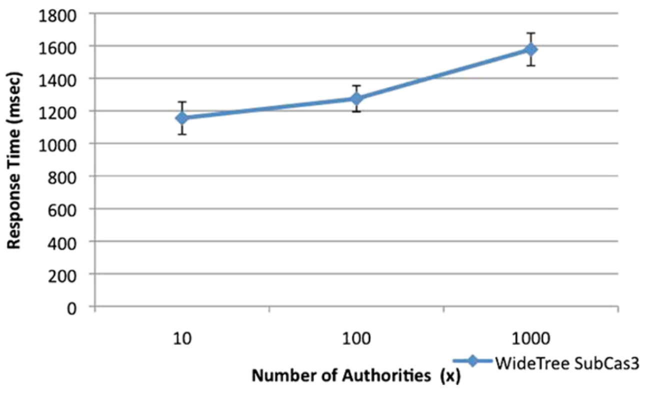

Fig. 19 Cross-mounting task response time: "wide tree"

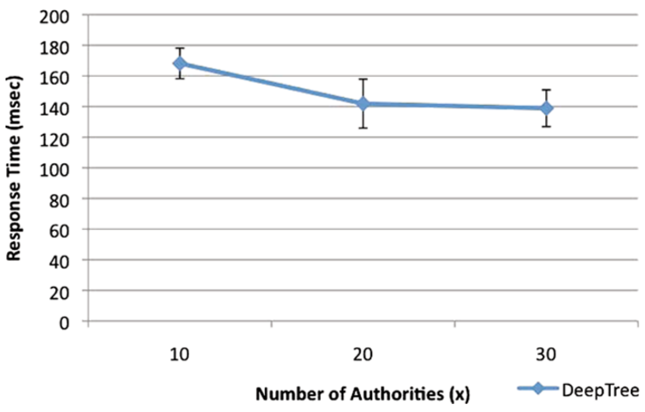

Fig. 20 Cross-mounting task response time: "deep tree"

considering the wide tree sub-case 3 structure. On the x-axis, we have represented the number of authorities, whereas on the y-axis we have represented the response time expressed in milliseconds. For this latter 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.

In the end, Fig. 20 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 levels, we can observe how the response time is indeed negligible: less of 0.2 seconds. In particular, the experiment shows that the result is uncorrelated with the time achieved (decreasing curve with a low slope).

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

7 Conclusions and remarks

In this paper, we presented a solution for cloud name space management.

Our practice of CNS also aims to overcome the issues related to the naming, location, and information retrieval of CNEs. Particularly, we presented the XRI technology as a possible solution to these problems. XRI was not conceived for cloud computing but in our opinion it might be useful indeed for cloud computing purposes. We evaluated an open implementation of XRI, highlighting how several tasks can be accomplished for the management of cloud name spaces. The conducted experiments show the goodness of OpenXRI and how it is particularly suitable to our goals. This assertion is motivated by the fact that if a cloud has thousands of CNEs, this does not mean that a XRI tree must have thousands of nodes and leaves. In fact, in such a situation the cloud might build several different smaller XRI schemes. However, for possible evolving cloud scenarios requiring higher response time for the accomplishment of name space management tasks in huge name spaces, we think that the OpenXRI Authority Resolution Server 1.2.1 solution could not be enough. In such a futuristic situation, in order to continue to benefit of the XRI innovation for the logically mapping and management of cloud name spaces, it would be needed alternative implementations of the XRI architecture presenting enhancements compared to the standard OpenXRI solution.

References

1. Sun Microsystems (2009) Take your business to a higher level—sun cloud computing technology scales your infrastructure to take advantage of new business opportunities, guide. April
2. http://www.novell.com/products/cloud-manager/?redir=vanity-launch. August 2011
3. Extensible Resource Identifier (XRI) Syntax V2.0 (2005) Committee specification (OASIS)
4. Extensible Resource Identifier (XRI) Resolution V2.0 (2008) Committee draft 03 (OASIS)

5. Organization for the Advancement of Structured Information Standards (OASIS). http://www.oasis-open.org
6. OpenXRI project, XRI applications and libraries. http://www.openxri.org/
7. Wikipedia OpenID (2011) http://en.wikipedia.org/wiki/OpenID, July
8. OpenID world wide usage (2007). http://www.ariadne.ac.uk/issue51/powell-recordon/. June
9. The security vulnerability of reassignable identifiers (2011). http://dev.inames.net/wiki/XRI_and_OpenID, July
10. Resources and services virtualization without barriers (reservoir). European project. http://www.reservoir-fp7.eu/
11. Higgins, Open source identity framework. http://www.eclipse.org/higgins/
12. Reed D, Strongin G (2004) XDI (XRI data interchange). A white paper for the OASIS XDI Technical Committee v2 (OASIS)
13. RFC 3986, Uniform Resource Identifier (URI): generic syntax. http://www.ietf.org/rfc/rfc3986.txt
14. RFC 3987, Internationalized Resource Identifiers (IRIs). http://tools.ietf.org/html/rfc3987
15. RFC 2141, Uniform Resource Names (URNs): URN syntax. http://www.ietf.org/rfc/rfc2141.txt
16. Amazon elastic compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/
17. Rackspace, The service leader in cloud computing. http://www.rackspace.com/
18. Security assertion markup language (OASIS). http://www.oasis-open.org/committees/security
19. Web services security: soap message security 1.0 (OASIS). http://www.oasis-open.org/committees/wss
20. Ahn GJ, Ko M, Shehab M (2009) Privacy-enhanced user-centric identity management. In: IEEE international conference on communications (ICC '09), pp 14–18
21. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53:50–58. doi:10.1145/1721654.1721672
22. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M (2009) Above the clouds: a Berkeley view of cloud computing. Tech rep UCB/EECS-2009-28, EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
23. Brynjolfsson E, Hofmann P, Jordan J (2010) Cloud computing and electricity: beyond the utility model. Commun ACM 53:32–34. doi:10.1145/1735223.1735234
24. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616. doi:10.1016/j.future.2008.12.001. http://www.sciencedirect.com/science/article/pii/
25. Celesti A, Tusa F, Villari M, Puliafito A (2010) How to enhance cloud architectures to enable cross-federation. In: 2010 IEEE 3rd international conference on cloud computing. IEEE Press, New York, pp 337–345
26. Celesti A, Villari M, Puliafito A (2010) A naming system applied to a reservoir cloud. In: 2010 sixth international conference on information assurance and security (IAS). IEEE Press, New York, pp 247–252
27. Celesti A, Villari M, Puliafito A (2010) Ecosystem of cloud naming systems: an approach for the management and integration of independent cloud name spaces. In: IEEE international symposium on network computing and applications (IEEE NCA10), pp 68–75
28. Chaisiri S, Uthayopas P (2008) Survey of resource discovery in grid environments. Tech. rep., High Performance Computing and Networking Center, Department of Computer Engineering, Faculty of Engineering, Kasetsart University, 50 Phaholyothin Rd., Chatuchak, Bangkok 10900, Thailand. http://javaboom.files.wordpress.com/2008/04/rs_grid_survey.pdf
29. Doi Y (2005) Dns meets dht: treating massive id resolution using dns over dht. In: Applications and the internet international symposium, pp 9–15
30. Doi Y, Wakayama S, Ishiyama M, Ozaki S, Ishihara T, Uo Y (2006) Ecosystem of naming systems: discussions on a framework to induce smart space naming systems development. In: ARES, p 7
31. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: Grid computing environments workshop (GCE '08), pp 1–10
32. Grossman RL (2009) The case for cloud computing. In: IT professional, vol 11, pp 23–27
33. Hameurlain A, Cokuslu D, Erciyes K (2010) Resource discovery in grid systems&#58; a survey. Int J Metadata Semant Ontol 5:251–263. doi:10.1504/IJMSO.2010.034048
34. Hirofuchi T, Ogawa H, Nakada H, Itoh S, Sekiguchi S (2009) A live storage migration mechanism over wan and its performance evaluation. In: The 3rd international workshop on virtualization technologies in distributed computing, pp 67–74
35. Mei Y, Dong X, Wu W, Guan S, Li J (2007) Sdrd: a novel approach to resource discovery in grid environments. In: Xu M, Zhan Y, Cao J, Liu Y (eds) Advanced parallel processing technologies. Lecture notes in computer science, vol 4847. Springer, Berlin, pp 301–312. doi:10.1007/978-3-540-76837-1_34
36. Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D (2009) The eucalyptus open-source cloud-computing system. In: 9th IEEE/ACM international symposium on cluster computing and the grid (CCGRID '09). , pp 124–131
37. Reed D, Chasen L, Tan W (2008) Openid identity discovery with xri and xrds. In: Proceedings of the 7th symposium on identity and trust on the Internet, pp 19–25
38. Sharma A, Bawa S (2008) Comparative analysis of resource discovery approaches in grid computing. J Comput 3(5):60–64. http://dblp.uni-trier.de/db/journals/jcp/jcp3.html#SharmaB08
39. Sotomayor B, Montero R, Llorente I, Foster I (2009) Resource leasing and the art of suspending virtual machines. In: 11th IEEE international conference on high performance computing and communications (HPCC '09). , pp 59–68
40. Sotomayor B, Montero RS, Llorente IM, Foster I (2009) Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput 13:14–22
41. Sun H, Huai J, Liu Y, Buyya R (2008) RCT: a distributed tree for supporting efficient range and multi-attribute queries in grid computing. Future Gener Comput Syst 24(7):631–643. doi:10.1016/j.future.2007.12.002
42. Tusa F, Paone M, Villari M, Puliafito A (2010) CLEVER: a CLoud-Enabled Virtual EnviRonment. In: 15th IEEE symposium on computers and communications computing and communications (ISCC '10), Riccione
43. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: a performance evaluation. In: CloudCom, pp 254–265
44. Wei Y, Blake MB, Dame N (2010) Service-oriented computing and cloud computing challenges and opportunities. IEEE Internet Comput 14(6):72–75
45. Yang D, Qin Y, Zhang H, Zhou H, Wang B (2006) Urns: a new name service for uniform network resource location. In: Wireless, mobile and multimedia networks, 2006 IET international conference, pp 1–4