Report e procedure per la migrazione ed interoperabilità 3.13.2

Versione 0.1
Data: 03/06/2014

Progetto iCaro
La piattaforma cloud per l'accelerazione del business delle PMI toscane
[CUP 6408.30122011.026000074]

Informazioni sul documento

ID Deliverable	3.13.2
Titolo Deliverable	Report e procedure per la migrazione e interoperabilità
ID Attività	3.2
N. Versione / Revisione	0.3
Natura: Bozza / Definitivo	Bozza
Partner responsabile	DISIT
Distribuzione: Riservato / Pubblico	Pubblico
Riferimenti Autore	Daniele Cenni, Marco Serena, Paolo Nesi
Data redazione	03/06/2014
Riferimenti revisore	Paolo Nesi
Data revisione	
Riferimenti soggetto che approva	Paolo Nesi
Data approvazione e consegna	

Controllo delle revisioni

Oggetto	Numero	Data
Prima stesura e revisione	0.1	03/06/2014
Seconda stesura e revisione	0.2	04/06/2014
Terza stesura e revisione	0.3	30/06/2014

Nota di riservatezza

Indice

Legenda Acronimi e sigle

Acronimo / Sigla	Dettaglio
OVF	Open Virtualization Format
CDMI	Cloud Data Management Interface
OCCI	Open Cloud Computing Interface

1. Introduction

Cloud computing has recently emerged as a new computing paradigm for organizing a shared pool of servers in data centers into a cloud infrastructure, which can provide on-demand computing resources (e.g., CPU, storage, network, database, applications and services) to users anywhere anytime, in a pay-as-you-go manner. Cloud computing facilitates scalable and cost-effective computing solutions, by rapidly provisioning and releasing resources with minimal user management effort and cloud provider interaction. According to the NIST definition of cloud computing, the cloud computing model is composed of five essential characteristics, *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity* and *measured service*, and three service models, *SaaS* (Software as a Service), *PaaS* (Platform as a Service) and *IaaS* (Software as a Service). A cloud has been very useful in supporting various applications, including computation-intensive applications storage-intensive applications, and bandwidth demanding Web applications. Vendors in the cloud market has rapidly proliferated in the past 6-7 years, including Amazon, Google, Microsoft and SalesForce. Each of them promotes its own cloud infrastructure, and incompatible standards and formats to access the cloud, preventing them from agreeing upon a widely accepted, standardized way to support cloud applications. Nonetheless, the need for multiple clouds to be able to work seamlessly together, *i.e., cloud interoperability*, is rising.
From the users' point of view, cloud customers are typically reluctant to be bound to the cloud offered by a single provider, since it might not be able to satisfy all the needs of the user, such as on geographic distribution of the cloud data centers, SLAs (Service Level Agreements), etc., and might bring the potential risk of ridiculously high charges later on, due to the development lock-in. The cloud users are most likely looking for interoperable clouds, where they can have full control on where to deploy their applications and migrate their services easily when need arises, without extra development investments. From the providers' perspective, this incompatibility among cloud providers can protect the interest of each provider temporarily, but not in the long run when the cloud market becomes more mature. Efforts have emerged to establish standards for federating clouds owned by different providers, especially advocated by relatively small cloud providers (*e.g.*, Rackspace, GoGrid and late-comers (*e.g.*, Red Hat, Dell, Oracle, etc.). Interoperable clouds promisingly represent the trend of cloud technologies, since they better fulfill the ultimate goal of the cloud computing paradigm, in providing global-scale, "unlimited" computing utilities with unified access interfaces. The importance of cloud interoperability has been highlighted by both the industry and the academia. The industry tries to address the cloud interoperability issues via *standardization* and many cloud interoperability standards have been proposed and even put into use in recent years. The European Commission has also aimed to "identify by 2013 a detailed map of the necessary standards (inter alia for security, interoperability, data portability and reversibility)". However, it may take years for the standards to be fully agreed upon and adopted, if ever. As practical, short-term solutions, researchers in both the industry and the academia have been developing technologies to enable interoperation among clouds, from both the cloud provider's and user's perspectives.
In this article, we conduct a comprehensive survey on the state-of-the-art efforts to enable cloud interoperability, with a focus on interoperability among different IaaS (infrastructure as a service) cloud platforms. We investigate the existing taxonomies, standards and implementation of cloud interoperability, as well as practical cloud technologies (*e.g.*, nested virtualization) to enable interoperation. We pose issues and challenges to advance the topic area, and hope to pave a way for the forthcoming research.

2. Cloud interoperability: Motivations

Existing Cloud computing solutions have not been built with interoperability in mind [8]. They usually lock customers into a single Cloud infrastructure, platform or service preventing the portability of

data or software created by them [9]. Moreover, the battle for dominance between the big vendors, like Amazon, Google and SalesForce, makes them reluctant to agree on widely accepted standards promoting their own, incompatible formats . This dominance increases the lock-in effect and affects the competition since Small and Medium Enterprises (SMEs) deter from entering the Cloud market. The European Network and Information Security Agency (ENISA) and European Commission have recognized the vendor lock-in problem as a high risk that Cloud infrastructures entail.
Interoperability is the missing element that will remedy this situation and benefit both Cloud customers and Cloud providers. In particular, in an interoperable Cloud environment customers will be able to compare and choose among Cloud offerings with different characteristics while they will switch between Cloud providers whenever needed without setting data and applications at risk. Moreover, an interoperable Cloud market will open up the IT industry to SMEs and strengthen their market position. They will interoperate and cooperate creating new business models according to demand without conflicts due to interoperability problems.
However, a danger lurks. Different (semantic) interoperability standards and frameworks can possibly lead to different interoperability solutions which are not interoperable between each other. Therefore, standardization bodies and researchers need to sit together and agree on a set of common principles that all interoperability solutions will adhere to.

3. Cloud interoperability: An Overview

A first challenge in cloud interoperability consists in its definition. As general term, interoperability is a property referring to the ability of diverse systems and organizations to work together (inter-operate). In computer world, this property has the concrete meaning of exchanging information and use of information that has been exchanged between two or more systems or components. It is a property of a product or system, whose interfaces are completely understood, allowing it to work with other products or systems.
The most simple way to describe the cloud interoperability is by its most used mottos like "avoid vendor lock-in", "develop your application one, deploy anywhere", "enable hybrid clouds" , or even "one API to rule them all". Browsing the literature, one can found various definition referring it as the ability to:

- abstract the programmatic differences from one cloud to another;
- translate between the abstractions supported by different clouds;
- flexible run applications locally or in the cloud, or in a combination;
- move applications from one environment to another or run in multiple clouds;
- move services, processes, workloads, and data between clouds;
- use same management tools, server images, software in multiple clouds;
- communicate between providers, port application and data between them;
- federate multiple clouds to support a single application.

3.1 Portability vs Interoperability

Portability and interoperability relate to the ability to build systems from re-usable components that will work together “out of the box”.
A particular concern for cloud computing is *cloud on-boarding* – the deployment or migration of systems to a cloud service or set of cloud services. A common scenario is that some components cannot be moved to the cloud; for example, because of requirements for the enterprise to have

complete control over personal data. On-boarding requires portability of those components that can be moved to the cloud, and interoperability with them of components that remain on in-house systems.
A system that involves cloud computing typically includes data, application, platform, and infrastructure components, where:

- Data is the machine-processable representation of information, held in computer storage.
- Applications are software programs that perform functions related to business problems.
- Platforms are programs that support the applications and perform generic functions that are not business-related.
- Infrastructure is a collection of physical computation, storage, and communication resources.

The application, platform, and infrastructure components can be as in traditional enterprise computing, or they can be cloud resources that are (respectively) software application programs (SaaS), software application platforms (PaaS), and virtual processors and data stores (IaaS).
Non-cloud systems include mainframes, minicomputers, personal computers, and mobile devices owned and used by enterprises and individuals.
Data components interoperate via application components rather than directly. There are no "data interoperability" interfaces.
Portability and interoperability of infrastructure components are achieved by hardware and virtualization architectures.
The interfaces exposed by these components are physical communications interfaces: these are important, but are the same as for traditional computing. For these reasons, infrastructure portability and interoperability are not discussed further in this Guide.
The main kinds of cloud computing portability to consider are data portability, application portability, and platform portability. These are the portability respectively of data, application, and platform components.
Application interoperability between SaaS services and applications, and platform interoperability between PaaS services and platforms are important kinds of cloud computing interoperability to consider.
Applications can include programs concerned with the deployment, configuration, provisioning, and operation of cloud resources. Interoperability between these programs and the cloud resource environments is important. This is management interoperability.
Applications can also include programs such as app stores (for applications), data markets (for, e.g., openly available data) and cloud catalogues (e.g., reserved capacity exchanges, cloud service catalogs) from which users can acquire software products, data and cloud services, and to which developers can publish applications, data, and cloud services. In this Guide, all such programs are referred to as marketplaces. Publication and acquisition of products is performed by platforms, including PaaS services, that interface to the marketplaces. This is the final important cloud interoperability interface.
The cloud computing portability and interoperability categories to consider are thus:

- Data Portability
- Application Portability
- Platform Portability
- Application Interoperability
- Platform Interoperability
- Management Interoperability
- Publication and Acquisition Interoperability

3.1.1 Data Portability

Data portability enables re-use of data components across different applications.

Suppose that an enterprise uses a SaaS product for Customer Relations Management (CRM), for example, and the commercial terms for use of that product become unattractive compared with other SaaS products or with use of an in-house CRM solution. The customer data held by the SaaS product may be crucial to the enterprise's operation. How easy will it be to move that data to another CRM solution?
In many cases, it will be very difficult. The structure of the data is often designed to fit a particular form of application processing, and a significant transformation is needed to produce data that can be handled by a different product.
This is no difference from the difficulty of moving data between different products in a traditional environment. But, in a traditional environment, the customer is more often able to do nothing; to stay with an old version of a product, for example, rather than upgrading to a newer, more expensive one. With SaaS, the vendor can more easily force the customer to pay more or lose the service altogether.
Cloud introduces no new technical problems, but its different commercial arrangements can make the old technical problems much more serious.

3.1.2 Application Portability

Application portability enables the re-use of application components across cloud PaaS services and traditional computing platforms.
Suppose that an enterprise has an application built on a particular cloud PaaS service and, for cost, performance, or other reasons, wishes to move it to another PaaS service or to in-house systems. How easy will this be?
If the application uses features that are specific to the platform, or if the platform interface is non-standard, then it will not be easy.
Application portability requires a standard interface exposed by the supporting platform.
This must enable the application to use the service discovery and information communication protocols implemented by the platform, as well as providing access to the platform capabilities that support the application directly. On a cloud PaaS platform, or a platform running on a cloud IaaS service, it may also enable applications to manage the underlying resources.
A particular application portability issue that arises with cloud computing is portability between development and operational environments. Cloud PaaS is particularly attractive for development environments from a financial perspective, because it avoids the need for investment in expensive systems that will be unused once the development is complete. But, where a different environment is to be used at run time – either on in-house systems or on different cloud services – it is essential that the applications can be moved unchanged between the two environments. Cloud computing is bringing development and operations closer together, and indeed increasingly leading to the two being integrated as devOps. This can only work if the same environment is used for development and operation, or if there is application portability between development and operation environments.

3.1.3 Platform Portability

There are two kinds of platform portability:

- Re-use of platform components across cloud IaaS services and non-cloud infrastructure – platform source portability,
- Re-use of bundles containing applications and data with their supporting platforms – machine image portability.

The UNIX operating system provides an example of platform source portability. It is mostly written in the C programming language, and can be implemented on different hardware by re-compiling it and re-writing a few small hardware-dependent sections that are not coded in C. Some other operating systems can be ported in a similar way. This is the traditional approach to platform portability. It enables applications portability because applications that use the standard operating system interface can similarly be re-compiled and run on systems that have different hardware.

3.1.4 Application Interoperability

Application interoperability is interoperability between application components deployed as SaaS, as applications using PaaS, as applications on platforms using IaaS, in a traditional enterprise IT environment, or on client devices. An application component may be a complete monolithic application, or a part of a distributed application.
Interoperability is required, not just between different components, but between identical components running in different clouds. For example, in a hybrid cloud solution, an application component may be deployed in a private cloud, with provision for a copy to be run in a public cloud to handle traffic peaks. The two components must work together.
Data synchronization is a particular issue when components in different clouds or internal resources work together, whether or not they are identical. Such components often keep copies of the same data, and these copies must be maintained in a consistent state. Communication between clouds typically has a high latency, which makes synchronization difficult. Also, the two clouds may have different access control regimes, complicating the task of moving data between them. The design approach must address:
Management of "system of record" sources
Management of data at rest and data in transit across domains that may be under control of a cloud service consumer or provider
Data visibility and transparency
Full interoperability includes dynamic discovery and composition: the ability to discover instances of application components, and combine them with other application component instances, at run time.
Cloud SaaS gives enterprises the possibility of acquiring new application capabilities quickly and easily, but much of the benefit of this is lost if costly integration work is needed to make the SaaS service interoperate with other applications and services that the enterprise uses.
Application components typically intercommunicate by invoking their respective platforms, which implement the necessary communications protocols. Protocol standards enable platform interoperability directly and are discussed under that heading. They are indirect enablers of application interoperability.
Application interoperability requires more than communications protocols. It requires that the interoperating applications share common process and data models. These are not appropriate subjects for generic standards, although there are specific standards for some particular applications and business areas.
There are, however, some design principles that improve application interoperability. Integration of applications designed following these principles still requires some effort, but is much less difficult and expensive than integration of applications that do not follow them.

3.1.5 Platform Interoperability

Platform interoperability is interoperability between platform components, which may be deployed as PaaS, as platforms on IaaS, in a traditional enterprise IT environment, or on client devices.
Platform interoperability requires standard protocols for service discovery and information exchange. As discussed above, these indirectly enable interoperability of the applications that use the platforms. Application interoperability cannot be achieved without platform interoperability.
Service discovery is currently used by a minority of applications, but is essential to achieve the highest levels of service integration maturity . Standard service discovery protocols should be supported by platforms used by service registries and other applications.
Protocols for information exchange between platforms should support the establishment of sessions and transfer of session information, as well as information transport. (In the case of IaaS, the

platform in question is not part of the infrastructure service but implemented on top of it.) Session information might, for example, include the user's identity, the authorization level established by the user for access control purposes, the user's time-zone, the user's language, and the user's preferred cultural environment.

3.2 Standardization

Standardization bodies, not-for-profit groups and member operated organizations are working on advancing Cloud computing interoperability standards, with the collaboration of academia and researchers, governments and vendors. The Distributed Management Task Force (DMTF) has introduced the Open Cloud Standards Incubator, recently replaced by the Cloud Management Working Group (CMWG), which aims at standardizing the interactions among Cloud environments. Moreover, the Open Virtualization Format (OVF), released by DMTF, describes a portable and efficient format for packaging and distribution of software to be run across multiple virtual machines. IEEE has introduced two working groups, named P2301 and P2302. The first is responsible for developing a standard that will enable portability, whereas the second concentrates on allowing a system in one Cloud to work with a system in another. The Cloud Computing Interoperability Forum (CCIF) is planning to come up with a global Cloud computing ecosystem, where two or more Cloud platforms will be able to work together seamlessly. Key factors are the standardization of Cloud interfaces and the unified description of semantic Cloud data models. Meanwhile, the Organization for the Advancement of Structured Information Standards (OASIS) sees Cloud computing as an extension of Service-Oriented Architecture (SOA) and plans to develop Cloud models, profiles and extensions on existing standards to support Cloud security and interoperability.

The **Open Group Cloud Work Group** aims to create a common understanding among buyers and suppliers, eliminating the vendor lock-in problem. The **Open Cloud Consortium (OCC)** will support the development of reference implementations, benchmarks and frameworks for interoperation between different Cloud providers. Moreover, a consortium of businesses launched by Intel, called the **Open Data Center Alliance**, scopes to specify the future hardware and software requirements that lead to more open and interoperable Cloud and datacenter solutions. The **Cloud Industry Forum (CIF)** scopes to advance and advocate the adoption and use of Cloud-based services by businesses and individuals creating a marketplace while the growth of a marketplace is also the primary objective of M Forum's Cloud Services Initiative. The **Open Cloud Computing Interface (OCCI)** from OGF is an example of a standard IaaS resource management interface interfacing IaaS Cloud computing facilities and allowing users interoperate using the same context. Similarly, Storage **Networking Industry Association (SNIA)** has produced the Cloud Data Management Interface (CDMI), an interface standard that enables interoperation with storage Clouds and provides a standardized way to access all such services.

3.2.1 OVF

The Open Virtualization Format (OVF) [10] is a packaging standard initiated by DMTF [11] for deployment of 16 virtual appliances. By abstracting a virtual appliance as a single atomic unit, it enables simplified and error-free deployment and migration of virtual appliances across multiple virtualization platforms, including IBM, Microsoft, Jump-Box, VirtualBox, XenServer, AbiCloud, OpenNode Cloud, SUSE Studio, Morfeo Claudia, and OpenStack [12]. An OVF package contains multiple files in a single directory. The directory always contains an XML file called OVF descriptor with the VA name, hardware specifications and references to other files in the package. In addition, the OVF package typically contains a network description, a list of virtual hardware, virtual disks,

certificate files, information about the operating system. DMTF advertises this format as vendor-neutral as it contains no reference to any vendor-specific information.
The OVF describes an open, portable, efficient and flexible format for packaging and distributing virtual appliances, with the following key features: (1) Enabling optimized distribution of virtual appliances; (2) Providing a simple, automated user experience, as it offers a robust and user friendly approach to streamline the VA installation process, (3) Supporting both single and multi virtual machine configurations. (4) Enabling portable VM packaging, as OVF does not rely on any specific host platform, virtualization platform, or guest operating system.
OVF addresses the following issues of *Virtual Appliance* in the cloud interoperability taxonomy.
• *Life Cycle*: With a standard format of VA, the five stages of VA life cycle can now be handled freely by the developers, due to the flexibility of the XML descriptor employed in OVF, which contains all the metadata of the VA. Cloud providers are also capable to build pre-configured VM images based on OVF, therefore allowing cloud users to better automate their service deployment.
• *Virtualization Platform*: The extensibility of OVF allows different virtualization platforms to be defined within a VA, whether it be OS-level virtualization, paravirtualization, or hardware-assisted virtualization, thus allowing cloud vendors with different virtualization platforms (*e.g.*, AWS, Microsoft Azure) to interoperate together.
• *Virtualization Manager*: OVF also provides a simple and unified interface for the virtualization managers to perform migration tasks within or across data centers of different cloud providers (*e.g., libvirt*).

3.2.2 CDMI

The Cloud Data Management Interface (CDMI) [13], proposed by the Storage Networking Industry Association (SNIA), defines the functional interface that applications can use to create, retrieve, update and delete data elements from a cloud. It is the storage backbone in cloud interoperability. CDMI features functions that (1) allow clients to discover the capabilities available in the cloud storage offering, (2) manage containers and the data that are placed in them, and (3) allow metadata to be associated with containers and the objects they contain.
By abstracting the storage as containers which contain data objects, CDMI addresses the following issues on *storage* of the cloud interoperability taxonomy.
• *Backup*: Backup is the most common operation for cloud storage, which is enabled by CDMI as a simple interface. Cloud users can schedule the backup or perform backup operations manually via the interface.
• *Replication*: CDMI containers and objects are mapped to a mounted file system's directories and files. This mapping allows their flexible replication in the low level data storage cloud, even across different data centers.
• *Snapshots*: CDMI defines a snapshot as a point-in-time copy (image) of a container and all of its content. A snapshot operation is requested using the container update operation. A snapshot may be accessed in the same way that any other CDMI object is accessed. An important use of a snapshot is to allow the contents of a source container to be restored to their values at a previous point in time, even across different cloud providers, using a CDMI copy operation.

3.2.3 OCCI

Open Cloud Computing Interface (OCCI) [14] is a boundary API that acts as a service front-end to an IaaS provider's internal infrastructure management framework. The OCCI community is an open community under the umbrella of the Open Grid Forum (OGF), with contributing members from both the industry and the academia.
OCCI was originally initiated to create a remote management API for IaaS model-based services, allowing for the development of interoperable tools for common tasks including deployment,

autonomic scaling and monitoring. It has since evolved into a flexible API with a strong focus on interoperability while still offering a high degree of extensibility. It is compatible with the existing standards such as the OVF and the CDMI, and serves as an integration point for standardization efforts among DMTF, IETF [15] and SNIA [16]. The current OCCI specifications define a set of common standard interfaces of management for IaaS cloud interoperability. The documents are divided into three categories consisting of the OCCI Core, the OCCI Renderings and the OCCI Extensions.

• The *OCCI Core* defines a representation of instance types which can be manipulated through an OCCI rendering implementation. It is an abstraction of real world resources, including the means to identify, classify, associate and extend those resources. It interacts with the renderings (including their associated behaviors) and can be expanded through extensions.

• The *OCCI Renderings* each describe a particular rendering of the OCCI Core model, *i.e.*, how each OCCI Core model is rendered over the HTTP protocol that leads to a RESTful API implementation. Multiple renderings can interact with the same instance of the OCCI Core model and will automatically support any additions to the model which follow the extension rules defined in the OCCI Core.

• The *OCCI Extension* each describe a particular addition to the OCCI Core model.

OCCI addresses the following issues on *Access Mechanism*, *Network*, *Security* and *SLA* of the cloud interoperability taxonomy.

• *Access Mechanism*: Access mechanisms are exclusively described in the OCCI Rendering specifications, where RESTful HTTP APIs are defined. Each resource instance within an OCCI system, *e.g.*, Compute, Network, Storage, etc., has a unique identifier, such that they can be accessed just like how we visit the websites on the Internet.

• *Network*: The Network within an OCCI system is a layer-2 networking entity (e.g. a virtual switch), in which the network addressing issue among clouds could be addressed. It can be extended using the mixin mechanism to support layer-3 and layer-4 capabilities such as TCP/IP etc. In regard to application-level communication, OCCI facilitates a RESTful API over HTTP.

• *Security*: Elements described by the OCCI Core specification need to implement a proper authorization scheme, which must be a part of a concrete OCCI rendering specification, part of an OCCI specification profile, or part of the specific OCCI implementation. Concrete security mechanisms and protection against attacks should be specified by an OCCI rendering specification, which in any case must address transport-level security and authentication on the protocol level.

• *SLA*: Standardization on SLAs is addressed in the *Billing and Negotiation/Agreement* part of the OCCI specification, which is currently still in progress.

3.3 Interoperability in IAAS Clouds: Taxonomy

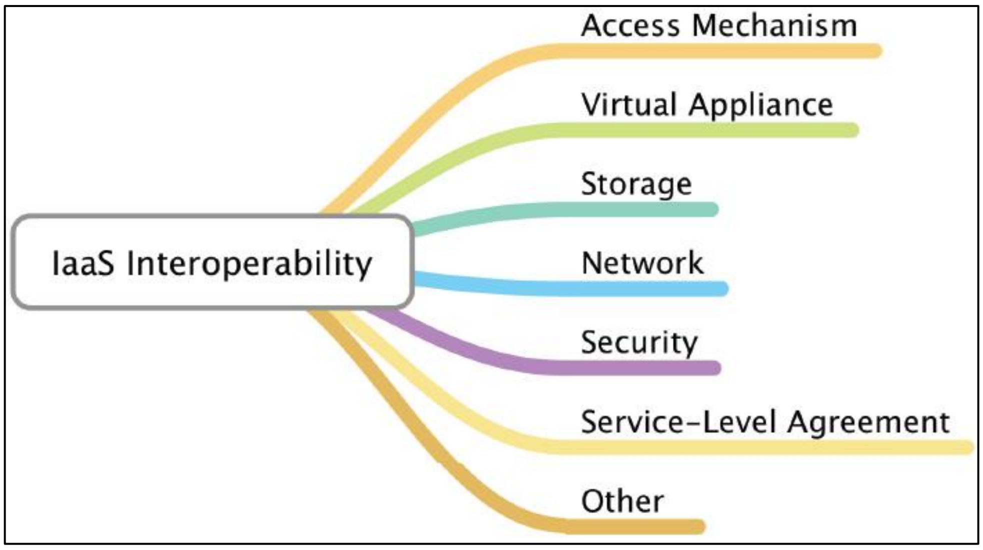


Figure 1: Interoperability of IaaS Taxonomy

Teckelmann and Reich [17, 18] have presented a taxonomy of interoperability for IaaS clouds (Fig. 1). The taxonomy discusses all important issues, to demonstrate the needs and trends in the state-of-the-art development aiming for IaaS interoperability.

(1) *Access Mechanisms*. Access mechanisms define how a cloud service can be accessed by developers or end users. The industry has considered three common standard types of access mechanisms: *Application Programming Interface (API)*, *Graphical User Interface (GUI)*, and *Command-Line Interface (CLI)*. GUIs and CLIs are mainly implemented on top of the existing APIs, which may use proprietary data formats based on XML, JSON or plain HTTP, and lack interoperability.

(2) *Virtual Appliance (VA)*. The idea of a virtual appliance is to deliver a service as a complete software stack installed on the VMs, as proposed by DMTF. Three issues of virtual appliances must be addressed, towards its goal of enabling cloud interoperability:

- *Life Cycle*. As defined by DMTF, the five stages of the life cycle of a VA are: *develop*, *package and distribute*, *deploy*, *manage*, and *retire*. Due to software upgrades or security fixes, update management is needed in all five stages of the life cycle, which will pose a significant problem to cloud providers. Starting from pre-configured VM images, updates should be applied across the whole life cycle of a VA carefully so as not to incur SLA violations that disrupt VMs' uptime and network performance.
- *Virtualization Platform*. Many different virtualization technologies exist, i.e., *OS-level virtualization*, *full virtualization*, and *paravirtualization*, each of which has its own advantages and disadvantages. Furthermore, when hardware-assisted virtualization is taken into consideration, cloud providers may also need to choose between Intel's VT and AMD's AMD-V. Solutions are needed to enable interoperability among clouds employing different virtualization technologies. In addition, *host architecture*, *host operating system* and *license* are also important to enable interoperability of the upper-level layer (*i.e.*, Hypervisor), which, to some extent, depends on the host architecture and operating system.
- *Virtualization Manager*. A virtualization manager is responsible for managing VAs on physical hosts. For interoperability, an important task is to move the VAs to different hosts and/or cloud providers, *i.e.*, migration is an essential task for the virtualization managers.

(3) *Storage*. Both the management and organization of storage are important issues to address for compatibility among different clouds. There are three topics on storage management in a cloud:

backup, *replication*, and *snapshots*. The backup functionality is important to guarantee longterm preservation of the data on a non-volatile storage media. Replication means that data are distributed to several sites. Snapshots are full copies of data objects. On the organization of storage, the schemes of organization and the image format are the major concerns. To facilitate compatibility, the introduction of an intelligent object storage layer can be a solution, where VM images and related data are decoupled and stored on the most appropriate storage, and logical connections can be kept using metadata information.

(4) *Network*. There are two important topics in terms of network interoperability: *addressing* and *application-level communication*.

- *Addressing*. A major issue is to have a reliable remote access to applications and their underlying VMs, even when they are being moved between subnets within a cloud or across the wide area networks.
- *Application-level communication*. The APIs should be RESTful, in order to minimize the coupling between client and server components in a distributed application. In the context of interoperable IaaS clouds, RESTful APIs are the appropriate choice due to the requirement that the client should be extremely resilient to changes on the server. There are two major communication protocols in concern, which implement RESTful APIs: *HTTP* (application-level transport protocol) and *XMPP* (XML over TCP).

In terms of IaaS interoperability, IP mobility is essential during live migration of virtual machines; otherwise it would directly cause service downtime and therefore violate the SLA. Many solutions exist to extend the capability of IPv4, including IPv6 and software-defined networking, which can better facilitate IP mobility.

(5) *Security*. Important security topics for cloud interoperability include *authentication*, *authorization*, *accounting* and *encryption*.

- *Authentication*. The confirmation of a stated identity is an essential security mechanism in clouds. To achieve cloud interoperability, authentication mechanisms have to be agreed upon among clouds to facilitate accessing resources from each cloud. Two kinds of authentication mechanisms are commonly presented: *HTTP Authentication* and *Public Key Infrastructure*.
- *Authorization*. Cloud providers need to allocate the resources or rights according to a user's credentials after the user has proven to be what he/she stated. In a federation of clouds, users with different backgrounds and requirements should be granted accesses to different resources of each cloud. Therefore, a proper authorization mechanism is necessary for the cloud federation to cooperate together and provide the best user experience possible for the cloud users.
- *Accounting*. In conjunction to security, accounting is necessary for the record of events and operations, and the saving of log information about them, for system and fault analysis. Interoperability among clouds can be seriously affected if no such information is available. The current issue is that a well-defined best practice guideline (not necessarily standard) on accounting should be agreed upon and adopted, such that interoperability can be achieved.
- *Encryption*. Challenges arise when ensuring the security between endpoints through communication encryption. Encryption mechanisms should be agreed upon in order for cloud users from different endpoints to access the resources of a cloud federation. Encryption mechanisms (SSL, TSL, VPN) are some of the widely adopted protocols.

(6) *Service Level Agreement*. SLAs are ubiquitous in modern IT systems. In terms of cloud interoperability, four important topics are involved on cloud SLA: *architecture*, *template format*, *monitoring* and *SLA objectives*.

- *Architecture*. Different cloud providers within a federation of clouds might have different SLAs. Thus, it becomes essentially important to have a well-adopted SLA architecture to measure and manage SLA requirements for different clouds. Otherwise, cloud users might get confused of the different SLA specifications. For example, Web Service Agreement

Specification (WSA) is the standard for the SLA management architecture in Web service environments.

- *Template Format*. Template formats are electronic representations of SLAs for the purpose of automated management. In order to achieve interoperability, it is necessary to agree upon a single template format. While most of the previous generation of template formats are *not* machine-readable, which brings trouble for automated management, the machine-readable S-A seems to be a good choice, but cloud providers have not yet started to offer machine-readable SLAs.
- *Monitoring*. Monitoring of SLAs is important for both cloud providers and users. A provider wants to ensure that the provision of resources is according to the SLA and no liability arises. On the other hand, a cloud user wants to ensure the adherence of cloud services as mentioned in the contract.
- *SLA Objectives*. Service level objectives (SLOs) are the core components of a SLA, which are quality-ofservice measurements for the performance of the service provider. In a scenario where a customer wants to change its cloud provider, the SLOs should be made consistent in order to compare the old SLA with a new one.

The aforementioned aspects of SLA interoperability share a common characteristic: they tend to provide the necessary functionality in the form of APIs to the cloud providers and users. APIs are crucial to interoperability for exporting SLA management functionalities. By separating SLA managing entities into modular components, their interoperability can be strengthened through exchangeability and extensibility.

3.4 Federation CloudThe federation manager enables access to remote cloud infrastructures, which can be either partner infrastructures governed by a similar cloud OS entity or public cloud providers. The federation manager should provide basic mechanisms for deployment, runtime management, and termination of virtual resources in remote clouds; remote resource monitoring; user authentication in remote cloud instances; access control management and remote resource permission; and tools for image building on different clouds with different image formats. The support for other advanced features such as creation of cross-site networks and virtual storage systems or cross-site VM migration will depend on the federation capabilities that remote clouds offer, as well as the level of coupling and interoperability supported by the different clouds involved. The federation manager's design could differ depending on the supported types of federation, for example cloud aggregation, bursting, or brokering, and levels of coupling and interoperability. The cloud OS must implement the federation manager as an internal component to support federation architectures at the infrastructure level. However, user-level federation scenarios could be implemented with a third-party stand-alone service, such as Aeolus (http://aeolusproject.org), which offers brokering services to access different cloud providers.

3.4.1 The Reservoir Example

The primary goal of Reservoir [19] (an acronym derived from resources and services virtualization without boundaries), a European research initiative, is to develop the technologies needed to deal with the scalability problem inherent in the single-provider cloud computing model. Reservoir explores the notion of a federated cloud in which computing infrastructure providers lease excess capacity to others in need of temporary additional resources.

In the Reservoir model, two or more independent cloud computing providers can join together to create a federated cloud. Federation participants who have excess capacity can share their resources, for an agreed-upon price, with participants needing additional resources. This sharing and paying model helps individual providers avoid overprovisioning of resources to deal with spikes in capacity demand.

In a multicloud environment, a system that makes automated decisions must address federated placement that is, the process of determining which cloud to use for a particular workload, given that not all clouds are equal in terms of warranties and prices. For example, a particular cloud might be inexpensive, but might not provide availability warranties, making it inappropriate for mission-critical workloads.

3.5 Cloud Management Systems supporting interoperability

Cloud standards could not evolve by themselves without the cooperation of their open source implementation.
The representative, state-of-the-art implementation of IaaS cloud management systems includes *OpenStack*, *OpenNebula* and *Eucalyptus* and *Nimbus*. Based on the current IaaS practice, almost all representative implementation of such cloud management platforms attend to interoperability with the dominating IaaS cloud platforms such as Amazon EC2.

3.5.1 OpenStack

As an open-source project managed by the OpenStack Foundation [20], OpenStack is an IaaS cloud management platform consisting of a trio of “core” service:
• The compute controller is named as *Nova*, which provides virtual servers upon demand. Nova’s architecture is designed to scale horizontally on standard hardware with no proprietary hardware or software requirements, and with the ability to integrate with legacy systems as well as other IaaS systems. In terms of supported hypervisors, Nova supports Xen [21] , KVM [22], LXC [23] and even Microsoft Hyper-V [24]. It also supports different CPU architectures (*e.g.*, x86, amd64 and ARM). These make Nova interoperable with the *Compute* service of other IaaS Clouds, such as Amazon EC2.
• The storage systems, *Swift* and *Cinder*, provide object storage and block storage, respectively. They are designed to be interoperable with Amazon’s object storage (S3) and block storage (EBS). In particular, Swift implements a subset of S3 APIs with the WSGI middleware.
• The image service, *Glance*, provides a catalog for virtual disk images, which includes many of the same features as Amazon’s AMI (Amazon Machine Image) catalog [25]. These disk images are commonly used in the OpenStack Compute module. Although the AMIs from Amazon can not be used directly in Glance (yet), their common APIs enable third-party IaaS management tools such as RightScale to automate management of images in OpenStack (using its own image format, compatible with AMI). The OpenStack networking system, *Quantum*, manages networks and IP addresses in the cloud, and intends to provide “networking as a service” between interface devices, *e.g.*, virtual network interface cards (vNICs), managed by other OpenStack services (*e.g.*, Nova). Quantum facilitates software-defined network capability in OpenStack, which is quite an effective solution for the network addressing issue over WAN. It also makes possible native live migration of VMs over WAN, given that the underlying hypervisor supports it. Equipped with the software-defined network functionality, OpenStack has overcome many of the obstacles (network-wise) towards interoperable clouds.

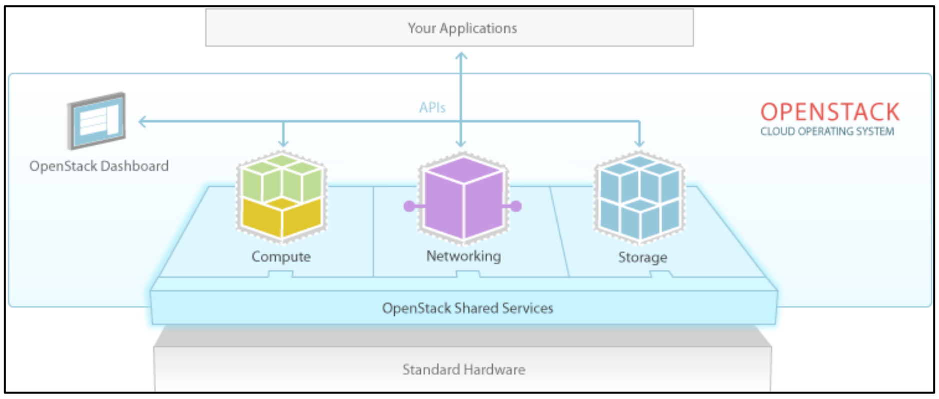


Figure 2: OpenStack

3.5.2 OpenNebula

OpenNebula [26], promoted by the OpenNebula Community, is an open-source data center virtualization technology, offering feature-rich, flexible solutions for the comprehensive management of virtualized data centers to enable on-premise infrastructure as a service clouds. It provides many different interfaces that can be used to interact with the functionality offered to manage physical and virtual resources. There are four main different perspectives to interact with OpenNebula (Fig. 3):

• Cloud interfaces for *Cloud Consumers*, including the OCCI, EC2 Query and EBS interfaces, and a simple self-service portal;

• Administration interfaces for *Cloud Administrators*, like a Unix-like command-line interface and the powerful Sunstone GUI;

• Extensible low-level APIs for *Cloud Integrators* in Ruby, Java and XMLRPC API [27];

• A marketplace for *Appliance Builders* with a catalog of virtual appliances ready to run in OpenNebula environments.

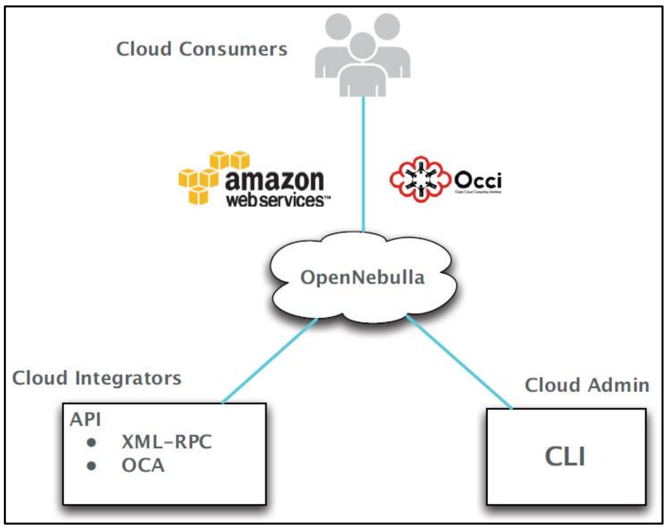


Figure 3: OpenNebula

In terms of cloud interoperability, OpenNebula implements a full EC2 Query interface, which enables seamless integration with Amazon EC2 clouds. Along with EC2 Query interface, OpenNebula natively includes a second remote interface: the *OCCI* interface which was discussed in details in Section. 3.2.3. By enabling the EC2 Query interface and OCCI interface, large-scale cloud infrastructures can be potentially built, opened to a wider audience. Manageability of a large infrastructure poses a significant challenge on interoperability among resources in the infrastructure. For this reason, OpenNebula implements *oZones* and *VDCs (Virtual Data Centers)*:

• oZones allow the centralized management of multiple running OpenNebula instances that are called zones (whether it be OpenNebula or AWS instances). This allows the oZone administrators to monitor each zone, and to grant access to different zones to particular users, from a centralized CLI and web GUI.

• Inside each zone, it is possible to define multiple VDCsthat contain a set of virtual resources (images, VM templates, virtual networks and virtual machines) and users that use and control those virtual resources.

3.5.3 Eucalyptus

Eucalyptus [28], created by Eucalyptus Systems, Inc., is a software architecture that creates scalable private and hybrid clouds within one's existing IT infrastructure. It features high-fidelity Amazon AWS

(Amazon Web Services) API implementation, in order to be easily interoperable with AWS clouds. Eucalyptus consists of the following key components (Fig. 4):
• Cloud Controller, which provides the Amazon EC2- compatible functionality;
• Walrus Storage, which provides the Amazon S3-compatible functionality;
• Cluster Controller, which manages a cluster in the cloud;
• Storage Controller, which provides the Amazon EBScompatible functionality;
• Node Controller, which controls the virtual machine instances.

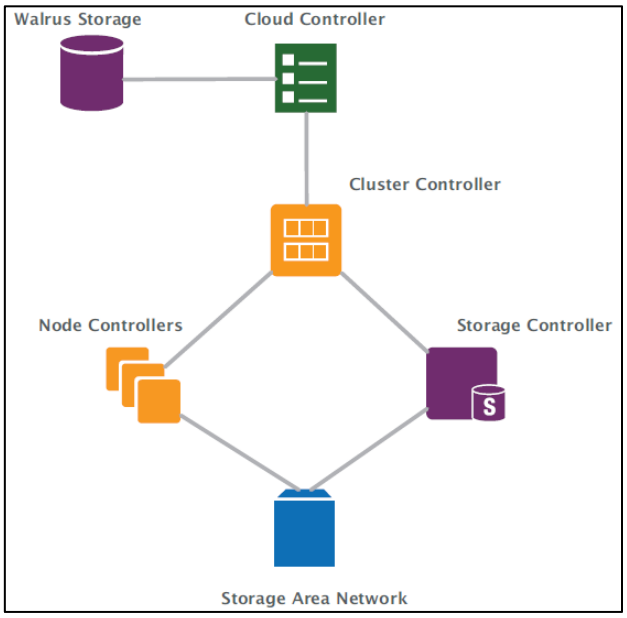


Figure 4: Eucalyptus

Eucalyptus has announced an agreement that enables customers to more efficiently migrate workloads between their existing data centers and Amazon clouds while using the same management tools and skills across both environments.
• *Flexibility in VM Formats*: Cloud users can run multiple versions of Windows and Linux virtual machine images on Eucalyptus clouds, and can build a library of Eucalyptus or Amazon Machine Images (AMIs) with application metadata that are decoupled from infrastructure details to allow seamless operation on Eucalyptus or AWS clouds. In addition, VMware images and vApps (multi-VM OVF packages) can be easily converted to run on Eucalyptus or AWS clouds.
• *Heterogeneous Hypervisor Management*: Cloud users can build and manage mixed hypervisor cluster environments in Eucalyptus clouds and manage their existing vSphere, ESX, KVM and Xen virtual environments as AWS-compatible Eucalyptus hybrid clouds. A self-service management portal can manage Xen, KVM, and VMware virtual environments from a single panel, leading to much interoperable and efficient clouds.
• *Interoperable Identity Management*: As for *Access Mechanisms*, Eucalyptus user identity management can be integrated with the existing Microsoft Active Directory or LDAP systems. In addition, Eucalyptus identity management interfaces are compatible with the AWS Identity and Access Management (IAM) APIs, which enables integrated management of hybrid clouds of Eucalyptus and AWS.

In summary, Eucalyptus provides a robust, on-premise cloud platform for cloud users to take advantages of AWS by implementing AWS EC2, EBS, S3 and IAM APIs natively, leading to interoperable hybrid clouds of Eucalyptus and AWS.

3.5.4 Nimbus

Nimbus [29] allows a client to lease remote resources by deploying virtual machines (VMs) on those resources and configuring them to represent an environment desired by the user. This is known casually as cloud computing (one definition of it), a little more specifically as an "infrastructure-as-a-service" (IaaS) solution.
Also included in a Nimbus installation is a storage cloud implementation called Cumulus that has been tightly integrated with the other central services (but it can also be used standalone). Cumulus is compatible with the Amazon Web Services S3 REST API, but extends it as well (to include quota management, for example). A set of software needs to be installed to one service node in a non-root account and a separate piece of software needs to be installed on any number of virtual machine monitor (VMM) nodes (some root permissions are required). There is also a separate download for a client called the cloud client that makes life very easy for users.

4. Cloud Migration

4.1 What is Cloud Migration

Cloud migration is the process of partially or completely deploying an organization's digital assets, services, IT resources or applications to the cloud. The migrated assets are accessible behind the cloud's firewall. Cloud migration is also known as business process outsourcing (BPO), which may entail migrating a total organizational infrastructure, where computing, storage, software and platform services are transferred to the cloud for access.
Cloud computing is attractive to many organizations due to its scalability, ease of management and low costs. Cloud migration facilitates the adoption of flexible cloud computing.
An organization's cloud migration process often involves merging an on-site IT infrastructure with a hybrid cloud solution, which may be accessed over the Internet for a fee. Hybrid cloud solutions transition between one or more cloud providers and usually provide on-demand and provisioned server space, applications and services.
Cloud migration is critical for achieving real-time and updated performance and efficiency. Thus, cloud migration requires careful analysis, planning and execution to ensure the cloud solution's compatibility with organizational requirements.

4.2 Cloud Migration Best Practices

Many different aspects have to be considered when selecting, evaluating, and planning the migration of a data center application to a cloud environment. The process begins with analysis of the factors for the application and a comparison of these factors to different types of cloud computing environments. It is important to analyze and measure application details that help in building a migration plan, and a plan for testing each migration phase. The above process can be iterative, because data might be uncovered, leading to a re-evaluation of the results in prior phases. Although no single approach will work for all applications, best practices will help in determining application suitability and performing a smooth migration.

A typical workflow migration process, as defined in [3], consists of the following phases:

The Migration Process

- *Establish an owner/team responsible for the cloud migration process*. The process of migrating to the cloud for the medium and large enterprise can be a significant effort. It is important to start the initiative by identifying an executive owner, business champion and cross-discipline transition team.
- *Establish the scope and business case*. Cloud migrations can be large efforts requiring an initial investment before the payback. It is important to establish the intended scope of systems that are being targeted for migration. To support the program, a business case should be published, describing the efficiency, agility or cost savings goals. The initial pass on scope doesn't have to be perfect but it should indicate an initial sizing. In further revisions, the scope will vary as some systems will be identified as good candidates to migrate while others will be removed from the list.
- *Establish the target clouds*. In many cases, companies have already established their target clouds, either private or public offerings. Those who haven't will go through the exercise of identifying the cloud service providers (CSP) for their workloads. This selection process reviews the CSP's offerings, SLA's, prices, geographies, customer satisfaction records, and more. For private clouds, the effort focuses on reviewing commercial and open source solutions and the appropriate architecture. The initial clouds are usually non-production environments. It is important to stabilize the operations of the cloud prior to moving business critical workloads.
- *Establish the transition framework*. The transition framework is the process that will be used to ensure a predictable outcome in the migration. Phases include:

Discovery

- *Discovery of current assets and usage*. In a cloud migration, it is necessary to identify the inventory of current assets and the extent to which they are used. Some of these may move to the cloud, some may be sunset and others may stay in their current location.
- *Discovery of topologies and dependencies*. Many applications span multiple machines with complex multi-tiered topologies. Even simple applications often have dependencies on storage devices, or network settings. Understanding the web of dependencies is critical to a successful migration.
- *Discovery of platforms and* licenses. It is common to find that some applications have dependencies on operating systems or platforms that are not supported by the cloud providers. Some commercial applications have software licenses that are based on CPU's, cores, memory, etc. that may affect the pricing of the software.
- *Discovery of SLA's, security and compliance*. Applications will have different levels of importance to the business. Some will require very high up-time, response time, security or may have to adhere to government regulations. Applications that require HA, low-latency IOPS, data encryption or similar, will need to be identified so that equivalent cloud solutions are made available.

Analysis

- *Cloud feature/fit analysis*. By using industry best practices and the information gathered in the Discovery phase, the portfolio of applications can be quantitatively analyzed to identify which systems are candidates for cloud migration.
- *Provider analysis*. For companies that have multiple cloud providers, the analysis will indicate which cloud provider has best-fit for a given workload.
- *Contract model analysis*. For workloads being migrated to an external cloud provider, the contractual model will vary. Models include: on-demand pricing, reserved instances, dedicated instances, bid/ask, etc.
- *Sizing analysis*. As systems are migrated to the cloud, it is necessary to allocate the right amount resources for them. Compute instances, storage devices and network bandwidth must all be appropriately aligned to the system needs based on data obtained in the Discovery phase.

Migration and Validation

- *Establish the migration toolkit*. Identify the appropriate toolkit to enable both one-time and repeat installations, configurations, validations and exception analysis
- *Migrate infrastructure*. VM, OS, Load balancing requirements, Network security: NAT, firewalls, disk
- *Migrate applications, platforms and data*. Migrate multi-tiered custom applications, packaged applications, middleware platforms, database, turnkey systems (e.g., Exchange, LDAP, Active Directory)

- *Migrate operations services*. Reinstate VPN’s, monitors and alarms, disk backup and restore
- *Validate migration*. Ensure that the systems have been migrated without loss of fidelity in functional and non-functional requirements

Lift-and-shift, Lift-and-refit, Cloud Modernization

As defined in [3], cloud migrations typically fall into one of three categories:

- *Lift-and-shift*. Fast, economical, focus is on pricing advantage
- *Lift-and-refit*. Efficient, moderately priced, take advantage of some cloud services without major investment
- *Cloud Modernization*. More costly up front, but takes full advantage of cloud agility, elasticity and pricing

Lift-and-Shift Migration

In a “Lift-and-Shift”, the emphasis is on migration speed and cost. Systems are moved from their current location (e.g., a hosting provider or internal data center) to a cloud provider with little or no change to the architecture or configuration. Because energy isn’t spent rethinking the architecture, the process required to make a move is typically very quick. The downside of this approach is that the application may not be leveraging the elastic properties of the cloud like auto-scaling. In Lift-and-Shift the cloud is treated as an on-demand, pay-per-use hosting environment.

Lift-and-Refit Migration

A popular approach to cloud migration is the “Lift-and-Refit”. Here, the applications are redeployed onto a cloud with a moderate number of changes that allow them to take advantage of many of the popular cloud capabilities such as:

- Auto-healing and recovery
- Data backup and restore
- Auto-scaling (up/down) based on current or predicted usage
- Orchestrated provisioning
- Monitoring and alerting

In a Lift-and-Refit, there are typically no code changes made to the system. The refitting occurs outside of the code by repackaging the software and giving the cloud environment enough information about the software to automate common operational activities, leading to reduced maintenance.

Cloud Modernization

In a Cloud Modernization effort, the application is redesigned from the ground-up with the assumption that it will be deployed in a cloud environment. Reengineering applications for a native cloud environment provide the greatest degree of flexibility; however they are also the most costly in the short term. Over time, the new application may provide enough benefits related to reduced administrative costs, reduced computing charges or other cost recovery vehicles to warrant an over-haul. Common changes made in a re-engineering effort include:

- Moving from a stateful to stateless service design
- Reducing storage costs by using scale-out storage strategies
- Sun-setting fixed networking approaches in favor of software defined networking (SDN)
- Removing expensive commercial software licenses in favor of open cloud friendly licenses like open source
- Switching from disk intensive designs to large memory footprints

Migration Key Points

According to [4], the following key points are worth considering, when performing an analysis on cloud migration:

- *Look for an established vendor with a track record*. Cloud vendors that are well established have a wider breadth of knowledge and deeper insights into potential pitfalls than a smaller less-established vendor. They are also more likely to have higher security standards, a better range of services, more resources available to meet peak demand, and a better quality of support and training available for their users. The support and training provisions alone are likely to make the biggest difference to a customer that is new to the cloud. The vendor must be able to answer questions in great detail and within an appropriate timescale.
- *Does the project really need to be migrated?* Not every project is suited to migration to the cloud. If management and customers are happy with the current hosting arrangements, and particularly so if they are cheaper than the cloud option, then there is no reason to move. Cloud migrations are usually only necessary when considering large-scale hardware purchases in order to sustain or scale existing in-house projects or enable new projects to

take place. Such migrations are only value-for-money if the perceived benefits of the migration outweigh the costs of performing it.

- *Consider data security*. Putting applications and data onto a system outside an in-house data center, customers will want to be sure that only the right people can access it and that its contents remain secure. It is important to have a detailed look at the applications that will be migrated, and to consider getting an ethical hacker to attempt to break into them so that developers can close any loopholes before the move takes place. Firewalls must be used to ensure no accidental backdoors are opened through routes other than the application itself. Communications should be encrypted with the external application and locked behind a proven authentication system, that will guarantee access only to authorized people.
- *Data transfer*. IaaS clouds are Internet-based, therefore there is usually only one method of getting data into them: uploading files across the internet. Many internet connections used by smaller businesses offer far slower upload speeds than download speeds and it can take an eternity to upload even a gigabyte of data. Even the fastest corporate networks struggle to upload a few tens of gigabytes within a reasonable timeframe (e.g. a dataset from a DNA sequencing machine). Some IaaS providers offer the option of shipping hard drives of data to them to avoid these upload bottlenecks, though shipping delays and workload at the vendor’s data center may mean that the process is far from instant. The best approach is to take a very close look at the data transfer requirements and see if they can be minimized (e.g. by pre-processing large datasets locally to produce a smaller summary dataset for upload). The cloud can be a good way of improving download speeds for customers using an application, deployed by cloud vendors on whichever of their physical servers are closest to a customer’s location. This can make a big difference to the response times customers get from the application.
- *Data storage and location*. How much data does the application really need? Cloud data storage can be expensive, particularly for very large quantities, so consideration should be given to data retention policies. Should old data be archived off-cloud to a tape library or other external resource? Is raw data needed at all or are summaries sufficient? Whilst not hugely expensive, movement of data within the cloud does cost money in terms of internal bandwidth charging, depending on the vendor, so applications should avoid moving it around unnecessarily. Shared data resources such as shared drives or central relational databases can be more effective than directly copying data between virtual machines, particularly for data sources that are usually accessed randomly or partially rather than sequentially or in their entirety.
- *Scaling*. The scalability of cloud applications is not something that magically happens upon deployment (at least, not in IaaS, although PaaS deployments of single applications inherit a certain amount of scalability from the host environment). Applications have to be placed behind load-balancers/auto-scalers within the cloud in order to be scaled up on demand. Some cloud vendors offer these as part of the service, others require the installation of third-party tools; however most scaling and balancing solutions incur some additional expense. Once the application is behind a load-balancer or auto-scaler, the application itself needs to be aware that it can be scaled. If migrating an in-house application that already sits behind an in-house load-balancer, then probably minor changes have to be made to support scaling in the cloud. For applications that have not yet been load-balanced in house, developers will need to assess the code to ensure that it can cope with a changing environment. How will

user sessions be persisted? How will they co-ordinate access to any central data resources in order to avoid conflict?

- *Service level guarantees*. The first question to ask any cloud vendor is what their availability guarantee is, and what could happen if they fail to live up to their claims. A cloud vendor failing to provide the agreed service is the worst possible situation for any cloud application to be in. Particular attention should be paid to the processes in place in case of vendor collapse or takeover. Once confident of the vendor's service guarantees, the next check is to look at vendor backup plans. Do they take on- or off-site backups? What is their disaster recovery plan in case of loss of a data center? Do they guarantee to recover the contents of the virtual servers, i.e. the applications and data, or will they only recover the base operating systems? Independent off-site backup plans should be built with these answers in mind.
- *Upgrade and maintenance schedules*. Cloud vendors will need to update their systems from time-to-time to install the latest security patches and new features. Applications built on top of the cloud will need to be aware that these patches take place and have plans in place to ensure that they won't be adversely affected after the upgrade. Vendors often give an option to decide when the upgrade will take place, subject to a fixed final deadline when it will happen anyway, so application developers should carry out testing well in advance to ensure service continuity. Likewise, if a vendor schedules planned downtime of cloud services to perform maintenance, application maintenance windows must be scheduled to coincide with this, in order to prevent an excessive numbers of outages for customers using the application.
- *Software architecture*. Traditional applications are designed for traditional hardware configurations. Cloud applications are designed for cloud infrastructure and features. The two are not necessarily equivalent. Whilst it is entirely feasible to take a traditional application and simply copy it to a cloud-based replica of its original environment, this is not always the most effective use of cloud functionality. Questions need to be asked regarding the choice of infrastructure (e.g., does it need a grid/cluster equivalent or can cloud alternatives such as Hadoop or self-instantiating instances provide a better service? Does it need an integrated load-balancer or can the cloud's default load-balancer suffice? Does it need database replication to distribute requests or can a single larger virtual database server handle all the traffic on its own?).
- *Check with the lawyers*. The final hurdle when migrating to the cloud is almost certainly going to be a legal one. Data protection or other acts of law may prevent the placement of data in certain locations (e.g. French law prevents clinical trial data from being transferred to locations in other countries, even within the EU). The contract with the cloud provider must also provide suitable protection for data transmitted to it. Checks must also be made to establish which jurisdiction's laws will apply in case of a dispute (e.g., the application owner's, or the vendor's head office, or the vendor's data center locations where the application and data is being kept?). Some lawyers express concerns regarding intellectual property (IP) of any data that is outsourced to an external location, cloud or otherwise. The opinions and rules vary widely depending on local custom and precedent so seek legal advice before putting anything on the cloud that could construe potential IP. Using licensed software on the cloud may in some cases contravene the terms of the license, or may invoke special clauses that would not apply elsewhere. License text must be checked carefully and if

necessary the software vendor must be consulted to get appropriate permissions or renegotiate the license.

4.2.1 Application Migration

Modern enterprise data centers have a number of applications deployed on existing infrastructures with known scalability or resiliency issues. Although migration of an application to a cloud will not solve all intrinsic application scalability or resiliency issues, enterprises might be able to derive tangible financial and operational benefits in moving an application from legacy servers to a cloud. Application migration is the process of redeploying an application, typically on newer platforms and infrastructure. The process involves the staging of the new environment before the actual cutover and requires coordination of IT teams at the time of cutover. If the migration is on a compatible platform, the application does not need to be recompiled. In the case of the cloud, the application can be migrated from an existing data center to the target cloud. The target infrastructure can be a public, private, or hybrid cloud, that is an environment transparently combining multiple clouds, both private and public. The migration can involve a physical-to-virtual (P2V) migration if the existing application is not running on a virtualized platform. To identify applications for migration to a cloud, it is necessary to first identify and understand the business and technical factors for the migration. Reducing costs and business agility are typical business factors for application migration to clouds. Cloud computing can provide significant cost savings because of the increased utilization resulting from the pooling of resources and the standardization and automation required for cloud services. Cloud computing enables rapid delivery of IT services, which increases business efficiency. This increased IT efficiency translates to overall business efficiency and has the potential to unleash new innovations and opportunities. On the operational side, manageability, performance, and scalability are the typical reasons why businesses consider cloud computing. By delegating the management of infrastructure and software platforms to a cloud service provider, customers can offload operational responsibilities to service providers. In an enterprise private cloud model, a common cloud IT management team can also help provide a similar service. A cloud computing environment might offer increased resources, which can lead to performance improvements for certain applications. Applications that are designed to spread their workload across multiple servers will be able to benefit from automated scaling of resources to match the current demand. This is especially appealing for applications with unpredictable or cyclical usage patterns, because a cloud orchestrator can monitor usage and can dynamically scale resources up or down. This behavior, combined with the pay-by-usage characteristic of a cloud, can lead to significant financial savings.

4.2.2 Migration Options

After an application has been identified as a candidate for cloud migration, based on business and technical factors, it is necessary to consider for what type of cloud environment—SaaS, PaaS, or IaaS—the application is best suited.

4.2.2.1 Software as a Service

Based on the type of application, and if SaaS-based alternatives exist, it is worth considering if the SaaS alternatives can meet both business and technical needs. Such a change is no longer an application migration but more of a replacement of the existing application with a SaaS option. There might still be a need to migrate existing data to the new application. SaaS removes the need to manage both the application and the infrastructure on which the application is deployed. This approach can be attractive, but certain criteria, such as service-level agreements (SLAs), data portability, and long-term costs, must be carefully evaluated when considering SaaS deployments.

• *SLAs*: The SaaS vendor should provide a SLA for application overall availability, scalability, and performance, as well as provide clear policies and guidelines for application maintenance and upgrade windows.

• *Data portability*: The SaaS vendor should provide a way to allow customers to own and control their application data. SaaS customers should have the ability to export all application data that belongs to them, in a format that can be easily parsed and migrated to other internal or external applications.

• *Long-term costs*: The SaaS model can be financially attractive for a small business or enterprise that has not made any significant investment in its own data center, because of the low initial costs. As the number of users and the period of ownership increase, however, it is necessary to carefully compare longer-term recurring costs over time against amortized cost of ownership. As with other leasing-based financial models, the overall costs of ownership might be greater for a lease as time and usage increase.

• *User management*: Enterprise users are typically managed using directory services such as Microsoft's Active Directory or other Lightweight Directory Access Protocol (LDAP)–based services. As user accounts are added, deleted, or modified, most SaaS applications will not be automatically updated with these changes, thus creating additional work and potential security risks for IT administrators.

• *Security*: The SaaS application can contain sensitive corporate data when stored on the SaaS provider's infrastructure. You should require transparency in the service provider's security policies to be able to determine whether adequate security is provided, based on the nature of application data.

4.2.2.2 Platform as a Service

Platform as a service might be an option for migrating business applications that are based on standard application server software such as Java EE 5 or Microsoft's .NET platform. In this model, the service provider manages the application platform software and might provide access to common application services such as SQL databases. The application platform might be shared by multiple applications belonging to different customers. How the application platform is mapped to the physical infrastructure is typically controlled by the cloud service provider. The decision factors in such a migration will depend on the type and version of the application server used. Some PaaS environments might not support all the features of the application server and might require application changes.

All of the same criteria used for considering a SaaS deployment, including SLAs, data portability, long-term costs, user management, and security, should be considered for a PaaS migration.

• *SLAs*: A PaaS vendor should provide SLAs for application platform availability and performance. The cloud service provider should provide clear policies and guidelines for maintenance and version management of the platform, and policies for APIs version compatibility between the platform and the application.

• *Data portability*: In a PaaS model, the application data is typically stored in a database provided by the cloud service provider. The customer must be able to export data in a format that can be migrated to other databases.

• *Long-term costs*: The financial model for a PaaS should be compared against those of an internal deployment of the infrastructure and the application server/platform using IaaS and deploying the application server using the cloud-based servers.

• *User management*: A PaaS application will require administrative and application user accounts. For both account types, customers should understand how the user management aligns with their existing directory services and user management processes.

• *Security*: In a PaaS deployment, the same application server might host applications from different customers. In such an environment, additional security is necessary to make sure that rogue applications are not able to exploit vulnerabilities in the platform software to affect other applications.

When evaluating PaaS, enterprises should also consider platform management and scalability.

• *Platform management*: Application servers provide management consoles and tools for monitoring and management of applications that are deployed on them. A PaaS deployment should allow customers to use similar tools to manage and tune their applications.

• *Platform scalability*: A PaaS environment might offer dynamic scaling (up or down) as an optional feature, based on the capabilities of the underlying application server. Dynamic scaling works well with applications where the user load is nondeterministic, such as for consumer Internet applications. If this feature is used, the PaaS provider should clearly indicate how the application will be scaled up or scaled down and how contention for resources will be handled.

4.2.2.3 Infrastructure as a Service

Migration of an application to IaaS involves deploying the application on the cloud service provider's servers. The initial step when migrating to IaaS is to determine whether the cloud-based server hardware and operating system (OS) are compatible with the current server's hardware and OS. For example, if an application is running on an x86 server, the cloud servers must be able to implement x86 instructions. If the hardware is not compatible, the application might need to be recompiled or redeployed for the new platform. If the OS is compatible, few changes will be required when the application is migrated. Most of the same criteria that are used for considering SaaS or PaaS application migrations, including SLAs, data portability, long-term costs, user management, and security, should be considered for the migration to IaaS.

• *SLAs*: For IaaS deployments, it is required a SLA for the availability and performance of the server, network, and storage infrastructure. Informations about the maintenance and management procedures for the infrastructure and how any potential downtime is handled must be at disposal.

• *Data portability*: In IaaS deployments, the application can use a database server that is also deployed in the cloud. In these case, the cloud service provider must provide a way to replicate or migrate the block or file storage that is used by the database server.

• *Long-term costs*: The cost of an IaaS application should be compared against the cost of deploying that application on enterprise servers. In some cases, a public cloud IaaS deployment might have benefits because of dynamic scaling and usage-based pricing. For always-on applications, based on the computing resources required, the longer term cost of ownership in a public cloud might be greater than the cost of ownership in a private cloud.

• *User management*: In IaaS models, up to three different user roles are possible:

- Server administrator
- Application administrator
- Application user

The user management procedures and tools for each of these roles should be evaluated.

• *Security*: In IaaS deployments, virtual machines belonging to different customers can be implemented on shared physical infrastructures. When considering an application migration, the cloud service provider's security policies for virtual and physical isolation as well as compliance should be examined. The cloud service provider should allow auditing of security and compliance policies, using emerging technologies and specifications such as those of the CloudAudit forum (http://cloudaudit.org).

• *Scalability*: Applications that are designed to scale out will benefit from dynamic scaling features in a cloud. Typically, these applications are multitiered and have request load-balancing features, such that a pool of stateless application servers can be dynamically scaled up or down. When using this feature, the cloud service provider should provide clear policies on how this type of scaling will function, and how resource contention across customers and applications is handled.

4.2.2.4 Public or Private Clouds

Cloud service providers can offer advanced enterprise-grade capabilities, for security and performance, combined with network services such as load balancing and WAN optimization services. After evaluating these aspects of a cloud service provider's offerings, it is necessary to consider whether it is always a good strategy to migrate applications to a public cloud. Not every application is suitable for migration to a public cloud. When deciding whether to choose a public or private cloud for an application migration, the WAN traffic architecture must be examined, together with data security and management, legacy application integration, and security and compliance.

• *WAN traffic*: If an application workload is traffic intensive and communicates with other data center resources or applications, migration to a public cloud will not be optimal because of WAN bandwidth costs and potential performance effects.

• *Data security and management*: Applications might depend on a shared data center storage or other resources such as directory services for user, profile, and data management. When evaluating a migration, such dependencies need to be identified and addressed.

• *Legacy application integration*: Legacy business applications running on platforms such as a mainframe and AS400 that might be tightly integrated with other business applications might be a risk for migration to public cloud offers. Applications that depend on legacy applications are definitely a good candidate for private cloud migration.

• *Security and compliance needs*: In a private cloud, an enterprise will have more direct influence over the infrastructure architecture and operational policies. This control can lead to efficiencies because of customization and optimization, which need to be evaluated against the additional costs.

4.2.3 Evaluation Criteria

After a type of cloud environment and the cloud service delivery model are chosen, the candidate application needs to be further evaluated to make sure that it is feasible to migrate and also to prepare for the migration process itself.

4.2.3.1 Application Architectures

The application architecture will affect how an application can be migrated to cloud environments and sometimes whether an application is suitable for migration.

4.2.3.1.1 Multitiered Applications

The majority of enterprise applications are built using multiple tiers to decouple the major functions and modules in the system. One such approach is to organize the application using three tiers as follows:

• *A data management tier*, which consists of relational or other database components

• *A business logic tier*, which uses application platform or containers, such as Java EE or Microsoft's .NET

• *A presentation tier*, which is responsible for interfacing with the user interfaces or other external systems, including managing state and data for presentation to these external systems

Applications that use a layered architecture approach have well-defined interfaces between these layers. Based on application usage patterns, it might be possible to migrate application tiers, or modules within a tier, separately. For example, in a web application, static content can be migrated to a content delivery network provider to allow parts of a website to load faster. In other cases, WAN bandwidth restrictions might prevent tiers from being separated, and all layers of the application will be migrated to a cloud. In either case, each layer and its major distributed components modules should be evaluated separately for how it should be sized and migrated to a cloud. Application tiers might also have varying security and zoning requirements. For example, some application data might have to be secured behind a firewall.

4.2.3.1.2 Scale-Up and Scale-Out Architectures

A "scale-up" architecture is defined when the application can benefit from more resources, such as CPU and memory added to a single server or node. In contrast, a "scale-out" architecture is one where an application scales horizontally by additional nodes being made available for the workload. Scale-out applications can take advantage of the pay-by-usage cost model of the cloud. When there are increased requests for an application, more nodes can be deployed to handle the increased load. When the requests slow down, the additional nodes can be powered off to reduce costs. Today, it is not possible to dynamically scale up an application running on a single machine instance. This might change in the future, because virtualization systems are starting to support hot-plug features, where more memory and CPU can be added dynamically.

4.2.3.2 Geographic Access

Some applications are used from a single geographic location, while others might be used from multiple locations or even worldwide. For example, consumer-facing Internet applications often are used in multiple regions of the world in an unpredictable manner. Since enterprises also have multiple corporate locations, they might benefit from allowing applications to be distributed closer to points of access. When migrating an application to a cloud, it is necessary to consider the locations from which the application will be accessed. Most global cloud providers will provide a mechanism to configure the expected access for an application by purchasing capacity in different regions or geographic zones. In other cases, application access might need to be blocked or disallowed in a region, often for regulatory or security purposes. When selecting a cloud, it is important to consider these factors and make sure that geographic access can be monitored, controlled, and optimized. The data collection can be done in multiple phases: identify all of the application’s immediate dependencies and what other applications are dependent on the applications’ dependencies. For example, if both application A and application B are using the same database server, this needs to be identified so that the migration plan can include a combined move or can include steps to split the dependencies.

4.2.3.3 Application Profiling

Application profiling is used to measure and collect real usage data of an application before it is migrated. This data can help size the application deployment in a cloud. Ideally, application data should be collected for at least 10 to 15 days to allow capture of variances in daily and weekly usage patterns. For each node on which the application runs, the following data should be collected:

- CPU usage
- Memory usage
- Storage data such as throughput, latency, and input/output operations per second (IOPS)
- Network data such as throughput, connections per second, and dropped connections

The node-level data can be used to estimate how many machines, and what type of machines, will be necessary when the application is migrated. In addition to node-level statistics, it is also important to profile user activity, such as the total number of connected users, request and transaction rates, and request latencies. The usage data can also be used to build automated tests for the application to make sure of the same or an improved level of service after the application is migrated. The node data, along with application usage data, can also provide an initial estimate of the costs of the cloud resources.

4.2.4 Planning and Testing

The final step in preparation for the migration of an application to the cloud is to develop a work plan and tests for the actual migration. Based on the type of application and its business continuity requirements, the amount of planning required can vary. If downtime is not acceptable or needs to be minimized, the application should be migrated in phases, with both the existing and migrated applications available for some period of time. After initial tests, users can also be migrated in batches, and cloud capacity can be increased over time. The application data collected in previous steps can be used for creating test suites and simulating different user types and loads. Having a test plan and automated tests will help make sure of a successful migration.

4.2.4.1 Solution Design

Evaluating applications, designing migration plans, and migrating applications to a targeted cloud computing model is a demanding task. It requires detailed application migration process, design experience and deep understanding of cloud computing models, as well as detailed knowledge of how the applications interact with both each other, the external cloud environment, and the underlying infrastructure. It also requires experience integrating IT systems and cloud management, and a structured approach to program management.

4.2.5 Migration Tools

Some cloud migration solutions include:

- *Racemi Cloud Path* [6]
  - Migration of an existing physical or virtual server to a cloud provider
  - Migration of a cloud server instance to another cloud provider
  - Supported platforms: Amazon Web Services, GoGrid, IBM, Rackspace, Softlayer, Terremark
- *HP Cloud Workload Migration Services* [5]
  - Migration of the application and its data, not the infrastructure
  - recipe-based approaches that don’t require code changes to the application when deploying to the cloud
  - Chef recipes, Puppet, or custom shell scripts for deployment and configuration
  - Compatible with Tomcat, Microsoft IIS, Ruby, LAMP, and other open source servers
  - Compatible with Websphere, Oracle Fusion/WebLogic, JBoss, .Net
  - Databases – Oracle, Microsoft SQL, MySQL
  - Big data – Cassandra, Hadoop, MongoDB, CouchBase
  - Content management – Manage using Drupal and other platforms
  - Desktop apps – Utilize Office (virtual desktop)
  - CRM – SAP, Oracle, SugarCRM
  - Deployment automation via UI console – Automate instance provisioning, network configuration, and storage management
  - Support operations & management – Monitoring, auto-scaling, high availability
  - No VM packaging or hypervisor dependencies – Deploy apps natively on HP Cloud VMs with no need for VM packaging by the enterprise before deploying
  - Hybrid cloud deployments and cloud-to-cloud migration – Supports migrating workloads from private cloud and other public clouds without having to worry about VM or hypervisor compatibilities
- *RiverMeadow Cloud Migration SaaS* [7]
  - Assign access and usage rights to individuals. The RiverMeadow SaaS allows to identify which target cloud environments a user can migrate into while allotting migrations on a per user and target cloud basis.
  - Multi-tier business structures provide a means to create hierarchical
    organization schemas that replicate channel business models within the RiverMeadow SaaS billing and audit system

- A RESTFul API allows for programmatic control of all functionality of the RiverMeadow SaaS – anything that can be done using the standard GUI can be accomplished via the API. Embed RiverMeadow into your own system, skin it with your look-and-feel or develop functionality on top of the RiverMeadow platform.
- User roles allow you to assign RiverMeadow SaaS usage right based on corporate job functions; Administrator, End User and Finance.
- The RiverMeadow SaaS is managed and kept up to date at the RiverMeadow POP. There is nothing to install on the source server or within the datacenter. Virtualized, regional POPs provide optimized performance and fault tolerance, ensuring your migrations run smoothly and without interruption.
- Auto-virtualizes physical servers
- Windows & Linux OS source server support
- Automated target cloud & user set-up
- Web GUI with integrated RESTFul API
- Agentless collection & deployment
- Hypervisor agnostic (VMware ESX, KVM, Xen)
- Minimize server (business) downtime
- Consolidate hypervisors
- Combat cloud sprawl
- Migrate server workloads, live & as-is
- Accelerates time to revenue
- Higher average revenue per user (ARPU)
- Increases addressable market
- Ability to acquire customers from other clouds

5. Proposta di soluzione per la clonazione di una VM da e per la piattaforma ICARO

In questa sezione verranno analizzati gli scenari per la clonazione di macchine virtuali partendo da un Data Center sorgente verso la piattaforma ICARO attraverso un processo di import e export, e viceversa dalla piattaforma ICARO a un altro data center. In questo ultimo caso verrà fatta la distinzione nel caso in cui il data center sia un vCenter di VMWare o altro Data Center. Vedremo più avanti cosa significa e cosa cambia nel processo di clonazione.

Vediamo innanzitutto il percorso che ha portato alla definizione delle soluzioni per lo scenario applicativo in cui ci troviamo, analizzando lo stato dell'arte e verificando come le soluzioni proposte e i tool presenti in commercio siano insufficienti, o meglio inadatti al nostro scopo.

5.1 Scenario applicativo

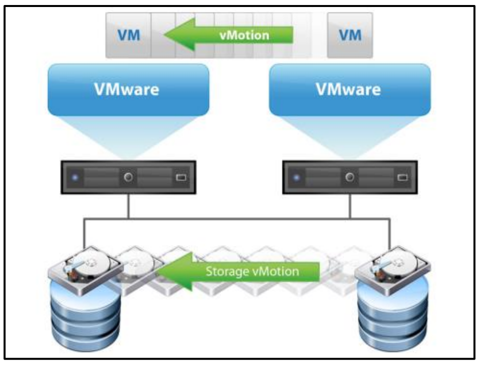
La situazione in cui ci troviamo è quella di due Data Center locati geograficamente distanti.

L'obiettivo è quello di poter rendere possibile la clonazione di Macchine Virtuali da un data center all'altro. In seguito, partendo da una clonazione, dovrà essere sviluppato un sistema per la migrazione vera e propria di macchine virtuali e servizi tenendo conto del fatto che siamo in presenza di due data center diversi, geograficamente separati. Ciò comporta la riconfigurazione delle macchine per le sottoreti a cui appartengono, del DNS, dei servizi offerti a terzi nel caso sia una macchina preposta a ciò, etc.

5.2 Stato dell'arte

Rimanendo in ambito VMWare è stato analizzato il funzionamento di vMotion e della cold migration per verificare se si potessero utilizzare per il nostro scopo.

5.2.1 vMotion



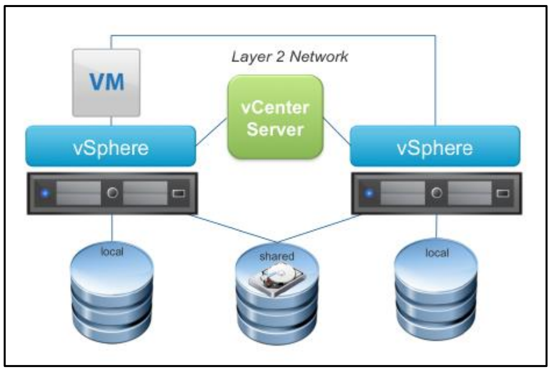


Figura 5: vMotion con storage condiviso o meno

Quando si esegue la migrazione di una macchina virtuale con vSphere vMotion, l’intero stato della VM viene traslato da un host ad un altro. Per stato di una VM si intende il contenuto della sua memoria e tutte le informazioni relative all’hardware, quali BIOS, elenco dispositivi, CPU, interfacce di rete e relativi MAC-address. Prima di iniziare la migrazione della VM, il vCenter Server effettuerà una pre-verifica dei requisiti per il vMotion: gli avvisi appariranno in giallo, e consentiranno di proseguire, mentre gli errori appariranno in rosso, e non consentiranno la migrazione.

5.2.1.1 Funzionamento di vSphere vMotion

Una migrazione vMotion prevede i seguenti passaggi:

- La memoria della macchina virtuale viene copiata dall’host sorgente a quello di destinazione, attraverso la rete, con gli utenti che possono continuare ad accedere alla VM. Di conseguenza è possibile una modifica di pagine di memoria anche durante il vMotion; il meccanismo prevede che di queste modifiche sia tenuta traccia nell’host sorgente, su una memory bitmap.
- Prima che sia completamente trasferita sull’host di destinazione, la VM è portata in uno stato di quiescenza, dove non sono più possibili modifiche. Nella fase di quiescenza sono trasferiti sull’host di destinazione lo stato dei dispositivi e la memory bitmap.
- Non appena la VM entra in quiescenza, viene avviata nell’host di destinazione; la rete è informata del cambio di switch tramite protocollo RARP (Reverse ARP) e, terminata la fase di sincronizzazione del punto precedente, gli utenti potranno accedere alla macchina ospitata sul nuovo host.

5.2.1.2 Procedura di migrazione tramite vSphere Client

Fare clic con il tasto destro sulla macchina accesa e selezionare la voce Migrate.

In caso di storage condiviso, nella procedura guidata selezionare la voce **Change host**. In tal caso, dopo la migrazione, i file della VM saranno sempre nello stesso storage. Se invece la VM risiede su un datastore locale, sarà necessario procedere con un cambio simultaneo di host e datastore, selezionando la voce **Change both host and datastore**.

È possibile impostare la priorità del vMotion utilizzando le opzioni High Priority e Standard Priority.

- **High Priority** – negli host con ESX/ESXi versioni 4.1 e successive, vCenter Server cerca di riservare le risorse ad entrambi gli host di origine e di destinazione, affinché siano disponibili a tutte le migrazioni simultanee. Il vCenter Server garantisce una quota maggiore di risorse CPU alle migrazioni ad alta priorità rispetto alle migrazioni a priorità standard. In ogni caso le migrazioni saranno portate a termine indipendentemente dalle risorse riservate.
- **Standard Priority** – negli host con ESX/ESXi versioni 4.1 e successive, vCenter Server riserva le risorse ad entrambi gli host di origine e di destinazione, affinché siano rese disponibili a tutte le migrazioni simultanee. Il vCenter Server concede una quota minore di risorse CPU alle migrazioni a priorità standard rispetto alle migrazioni ad alta priorità. In ogni caso le migrazioni saranno portate a termine indipendentemente dalle risorse riservate.

5.2.1.3 Requisiti per le migrazioni con vSphere vMotion

La migrazione di una VM richiede una corretta configurazione della rete sia per l'host sorgente, sia per quello di destinazione. In particolare si consiglia quanto segue:

- su ogni host, configurare un'interfaccia VMkernel per il vMotion;
- utilizzare interfacce fisiche Gigabit per il vMotion; è preferibile che gli host abilitati al vMotion siano attestati su una rete Gigabit Ethernet;

Se nell'host sono disponibili solo due interfacce di rete:

- Dedicare l'interfaccia gigabit al vMotion, e nell'altra interfaccia utilizzare le VLAN per dividere il traffico delle VM dal traffico di management;
- In alternativa, per una miglior disponibilità, combinare entrambe le interfacce in teaming, e utilizzare le VLAN per la separazione del traffico: una o più VLAN per il traffico delle VM e una per il vMotion;
- Assicurarsi che la rete su cui è attestata una VM sia presente anche nell'host di destinazione; durante la migrazione il vCenter Server assegna le macchine virtuali ad un determinato switch sulla base del nome assegnato al port group, nome che deve avere corrispondenza tra host sorgente e host di destinazione;
- Affinché la migrazione con vMotion possa essere portata avanti, la VM non deve trovarsi su switch privi di uplink (virtual intranet);

Per quanto riguarda gli host, i requisiti per una migrazione vMotion sono i seguenti:

- se si esegue il vMotion classico, con cambio del solo host, è richiesta la visibilità dello stesso datastore da entrambi gli host (sorgente e destinazione);

- compatibilità tra host a livello di CPU; ad esempio, se la CPU dell'host sorgente supporta le estensioni SSE4.1 e questo supporto non esiste nella CPU di destinazione, la migrazione vMotion fallisce;
- se la VM ha dischi RDM, questi devono essere visibili e accessibili anche dall'host di destinazione; inoltre i file di mapping dei dischi RDM devono trovarsi nello stesso datastore;
- la VM da migrare non può avere immagini CDROM e floppy montate.

Come si evince dai requisiti per la procedura di migrazione utilizzando vMotion, possiamo dire che questi sono incompatibili con la nostra situazione.

5.2.2 vCloudConnector

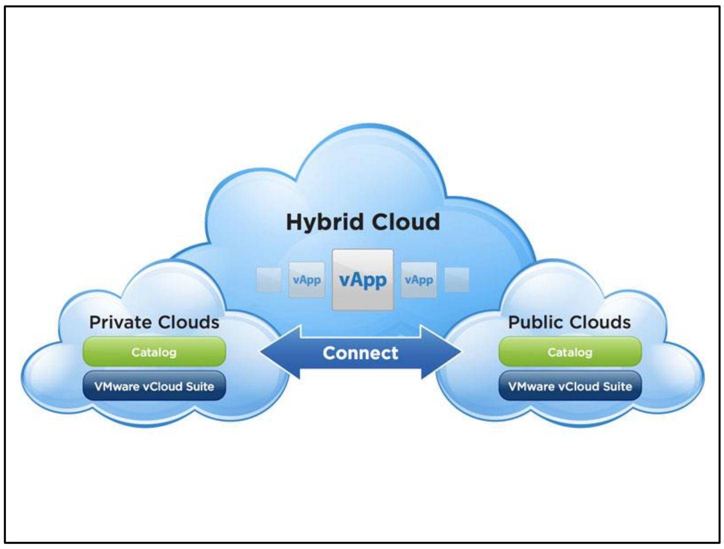


Figura 6: Cloud Connector

vCloud Connector makes hybrid cloud computing easy. Bring your existing applications, workloads and templates to a public cloud such as the vCloud Hybrid Service while retaining the freedom to move these applications back into your own data center when needed. Built-in compression, an improved and “path-optimized” multi-part transfer and checkpoint-restart makes transferring workloads between your connected clouds faster and more reliable than ever before.

By extending the logical boundaries of the data center, you can accelerate the deployment time of your workloads while maintaining consistency across clouds. Data Center Extension (DCE) allows for the transfer of workloads across clouds without needing to reconfigure the network settings in the destination. This increases operational agility of the cloud, realizing “one network” to deliver your applications.

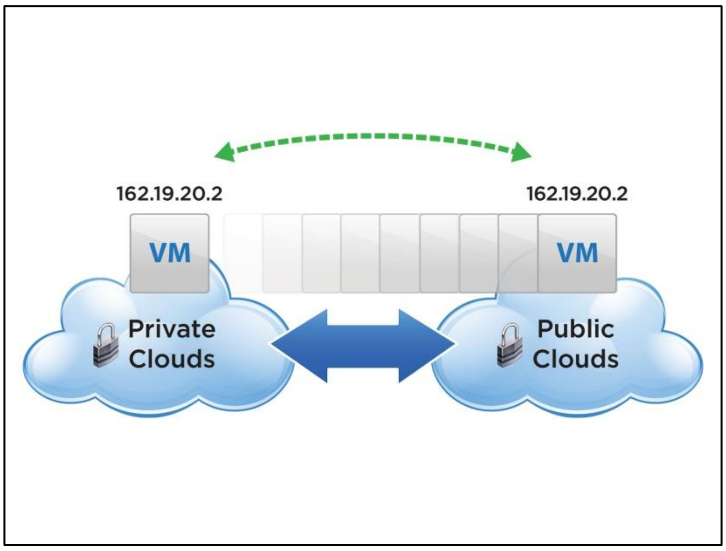


Figura 7: data center extension with vCloud Connector

5.2.3 vShpere Replication

vSphere Replication, il motore di replica proprietario di VMware, copia soltanto i blocchi modificati nel sito di ripristino. Tale approccio riduce l'utilizzo della larghezza di banda e consente obiettivi del punto di ripristino (RPO, Recovery Point Objective) più mirati rispetto alla replica manuale dell'intero sistema delle macchine virtuali. Replication consente di:

- Utilizzare una "copia seed" dei dati delle macchine virtuali durante la sincronizzazione iniziale.
- Garantire un utilizzo efficiente della rete attraverso il monitoraggio delle aree del disco che hanno subito modifiche e la replica solo dei delta.

Durante la configurazione di una macchina virtuale per la replica, è possibile garantire la coerenza dei dati di applicazioni e macchine virtuali con un solo clic.

- Integrazione automatica con Microsoft Volume Shadow Copy Service (VSS) per garantire la coerenza delle copie di ripristino.
- Obiettivi del punto di ripristino flessibili su un arco temporale di 15 minuti/24 ore.
- Controllo della replica delle macchine virtuali tramite VMware vCenter Server.
- Possibilità di scalare fino a centinaia di macchine virtuali per cluster.
- Utilizzo di più snapshot point-in-time per ripristinare stati noti precedenti.

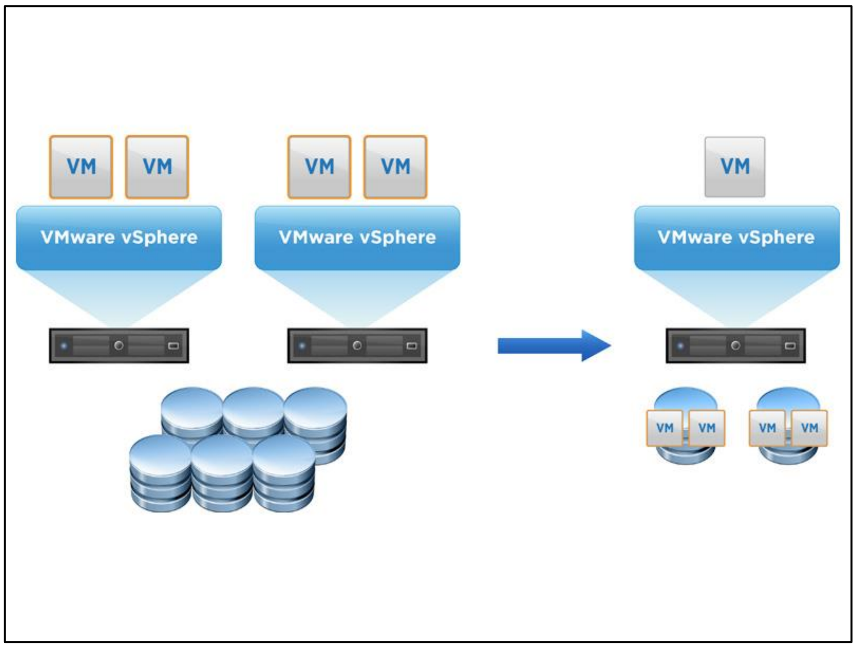


Figura 8: VMware data Replication

5.2.3.1 Dettagli tecnici

Replica delle macchine virtuali a livello di hypervisor

Replication è un componente profondamente integrato di VMware vSphere, l'unico motore di replica di macchine virtuali realmente di "livello hypervisor" del settore. I blocchi modificati nei dischi di una macchina virtuale in esecuzione presso un sito primario vengono inviati a un sito secondario, dove vengono applicati ai dischi della macchina virtuale per la copia (di protezione) offline della macchina virtuale stessa.

Gestione della replica basata su agente

vSphere Replication include un agente nel pacchetto d'installazione di base di vSphere su ciascun host, oltre a un insieme di appliance virtuali che vengono distribuite dall'interfaccia di gestione. L'agente invia i dati modificati da una macchina virtuale in esecuzione all'appliance presso un sito remoto. L'appliance quindi aggiunge la replica ai file dei dischi offline di tale macchina virtuale. L'appliance gestisce inoltre il processo di replica, offrendo agli amministratori visibilità sullo stato di protezione della macchina virtuale nonché la possibilità di ripristinare tali macchine con pochi semplici clic.

Obiettivi del punto di ripristino regolabili

Configurare la replica delle macchine virtuali (fino a un massimo di 500) è semplice tramite l'interfaccia di gestione standard vCenter Server: è sufficiente fare clic con il pulsante destro del mouse su una macchina virtuale e selezionare la destinazione della replica. Il processo include l'impostazione di un obiettivo del punto di ripristino per definire il periodo di conservazione massimo di ogni copia della macchina virtuale. vSphere Replication replicherà quindi i dati in modo da rispettare sempre l'obiettivo del punto di ripristino definito, garantendo così che il contenuto della macchina virtuale non superi mai il termine stabilito nella policy di replica.

Sincronizzazione e copie seed di una macchina virtuale

vSphere Replication eseguirà una sincronizzazione iniziale della macchina virtuale di origine e della relativa copia di replica. Se lo si desidera, è possibile inserire una copia seed dei dati nella destinazione per ridurre al minimo il tempo e la larghezza di banda necessari per la prima replica. Una copia seed di una macchina virtuale consiste in un file dei dischi della macchina virtuale che può essere inserito nella posizione di destinazione con pressoché qualsiasi metodo. Il seed può essere creato manualmente oppure copiato nella posizione desiderata mediante il metodo scelto dall'amministratore, ad esempio copia offline, FTP, "sneakernet" oppure un clone della macchina virtuale o ISO.

Trasferimenti intelligenti

Una volta completata la sincronizzazione di base, vSphere Replication trasferirà soltanto i blocchi di dati che hanno subito modifiche. Il kernel di vSphere tiene traccia delle operazioni univoche di scrittura sulle macchine virtuali protette, identificando e replicando solo i blocchi che hanno subito operazioni esclusive di scrittura durante l'obiettivo del punto di ripristino preconfigurato. In questo modo il traffico di rete viene ridotto al minimo e sono possibili obiettivi del punto di ripristino mirati. È sufficiente inviare i dati univoci una sola volta. Solo le modifiche vengono replicate e inviate all'appliance vSphere Replication della posizione di destinazione.

Replica non intrusiva

Il processo di replica delle macchine virtuali non è intrusivo e non dipende dal sistema operativo. È trasparente per le macchine virtuali protette e non richiede alcuna modifica della configurazione o della gestione continua.

5.2.4 vSphere Data Protection

vSphere Data Protection is a new backup and recovery solution designed for VMware vSphere. It provides advanced deduplication and is included with vSphere 5.1.

vSphere Data Protection provides easy setup and management, deduplication, and high-performance virtual machine image-level backup and recovery. It is ideally suited to protect smaller environments of up to 2 TB or 100 virtual machines. Larger environments can be protected by scaling out multiple vSphere Data Protection appliances or by deploying vSphere Data Protection Advanced.

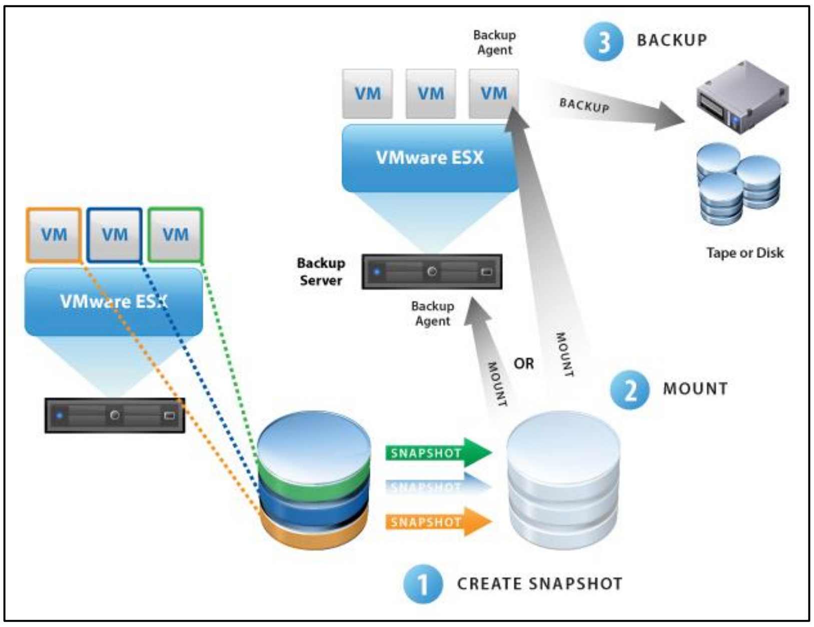


Figura 9: vSpehere data protection

5.2.5 VM-Converter

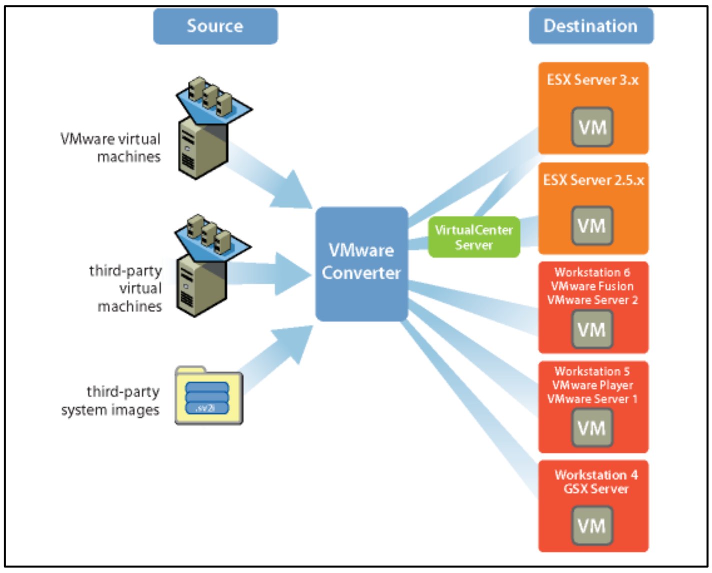


Figura 10: VM Converter

La console di gestione centralizzata di VMware Converter permette ai clienti di gestire facilmente migliaia di conversioni e trarre vantaggio da:

- Conversione rapida e affidabile di macchine fisiche in ambiente Windows in macchine virtuali, senza interruzione delle attività o tempi di inattività.
- Importazione di macchine virtuali VMWare in un vecchio formato all'interno di macchine virtuali VMWare con un nuovo formato.
- Consente la gestione centralizzata e simultanea delle conversioni remote di molti server fisici o macchine virtuali.

- Popolamento rapido dei nuovi ambienti di macchine virtuali da grandi directory di appliance.
- Creazione di cloni di un computer fisico a scopo di disaster recovery.
- Ripristino delle immagini di macchine virtuali VMWare Consolidated Backup (VCB) in macchine virtuali funzionanti.

5.2.2.1 Funzionamento di VMConverter

La gestione di VMware vCenter Converter avviene mediante una semplice interfaccia utente basata sulle attività, che consente la conversione di macchine fisiche, di immagini disco in formati di terze parti, o di immagini VMware Consolidated Backup in macchine virtuali VMware. L'operazione si svolge in tre facili passaggi:

1. Specificare il server fisico, la macchina virtuale o il formato di origine da convertire.
2. Specificare il formato di destinazione, il nome della macchina virtuale e la posizione in cui creare la nuova macchina virtuale.
3. Creare o convertire sulla destinazione una macchina e configurarla.

VMware vCenter Converter velocizza il processo di conversione utilizzando la copia basata su settori anziché la copia a livello di file, come nel caso di altri prodotti. VMware vCenter Converter esegue una snapshot del sistema di origine prima di migrare i dati e ciò consente di ridurre le conversioni non andate a buon fine e le interruzioni sul server di origine. VMware vCenter Converter comunica direttamente con il sistema operativo guest della macchina fisica di origine, eseguendo così l'hot cloning di questi sistemi senza interruzioni e quindi senza alcuna dipendenza diretta a livello di hardware.

5.3 Soluzione

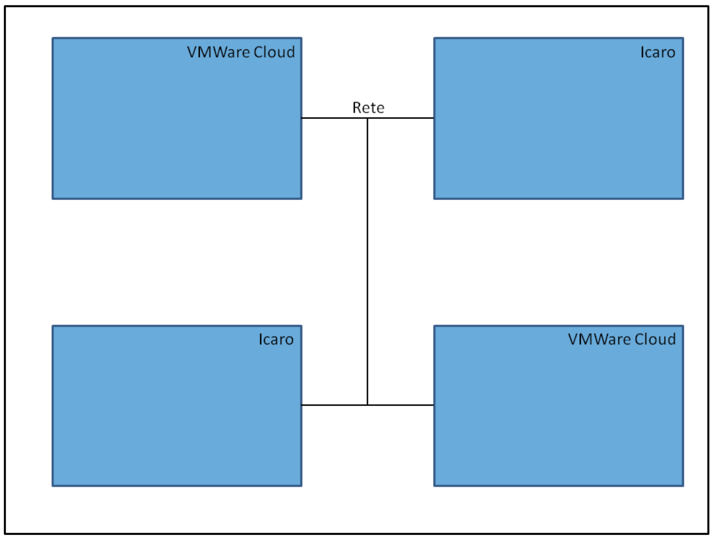


Figura 11: Scenario dei quattro casi

La soluzione viene proposte per ognuno dei quattro scenari della tabella 1.

Caso	Sorgente	Destinazione
1.a	Icaro	VMWare
1.b	Icaro	Icaro
2.a	VMWare	Icaro
2.b	VMWare	VMWare

Tabella 1: Scenari della soluzione proposta

5.4 Scenari 1a e 1b : Export di una macchina virtuale

Si parla di Export quando da Icaro si deve poter clonare una macchina virtuale in un'altra struttura. Questa struttura può essere un Cloud con architettura stile ICARO, oppure un data center VMware. In entrambi i casi, avremo a che fare con un data center vCenter per cui servirà VMConverter per poter virtualizzare le macchina e crearne l'immagine da spostare con FTP.
Nel caso in cui ci troviamo con VMWare ci deve essere un programma PHP/Java che interroga le API VMWare per ottenere le informazioni per la configurazione della VM da esportare, in maniera speculare rispetto al processo di import.
I passi necessari per poter esportare una VM da Icaro in una struttura VMWare sono i seguenti:

1. Viene fatta richiesta di export della VM.
2. 3. L'Export Manager si occupa di interrogare il data center per prendere le informazioni relative alla macchina richiesta, formattandole nell'XML conforme al modello Icaro.
4. Viene chiamato il VM Converter per richiedere la conversione della macchina per poterla esportare verso un disco di appoggio.
5. 6. Il VMConverter si occupa di recuperare la macchina nel data center e metterla nel disco dal quale poi sarà esportata.
7. Anche il file XML con la descrizione viene messo nel disco per l'esportazione.
8. Inizia il trasferimento via FTP.

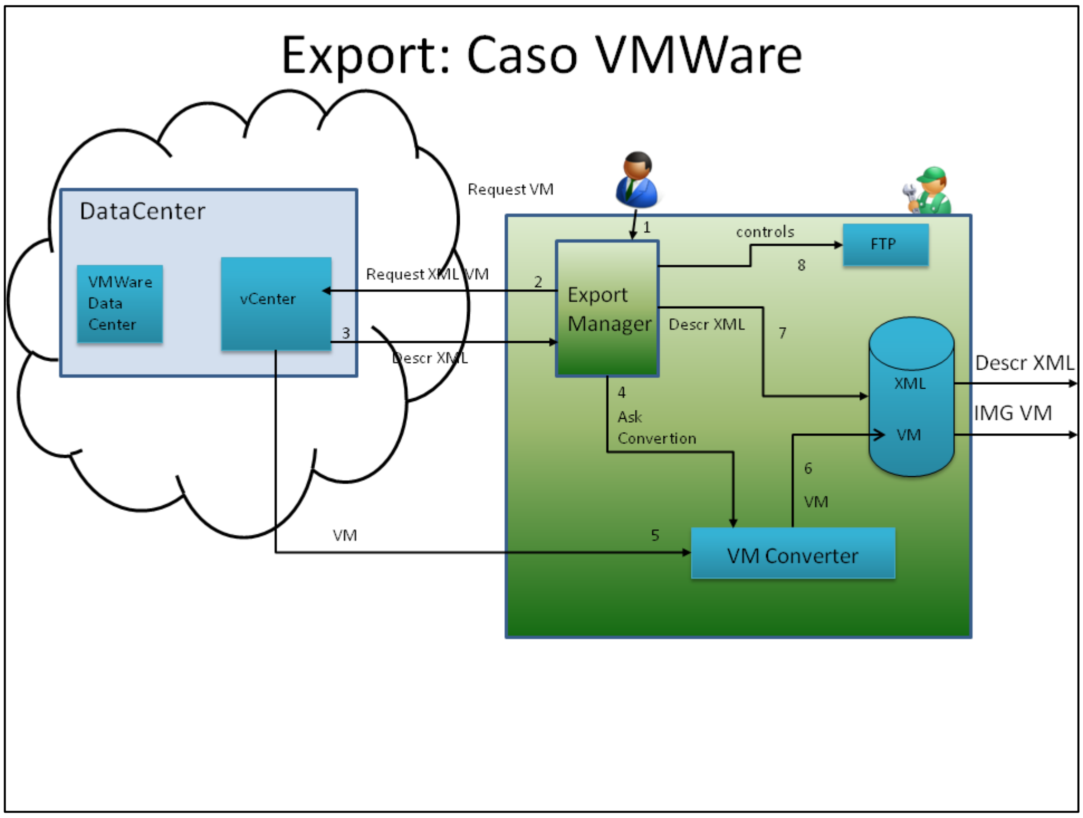


Figura 12 - Caso 1.a: Architettura interessata durante la fase di export di una VM nel caso si esporti su un data center con VMWare

Nel caso in cui ci troviamo con un'altra struttura Cloud come Icaro, la VM dovrà contenere al suo interno un tool che esegua query SPARQL sulla KB per ottenere la configurazione e un descrittore della VM da esportare.

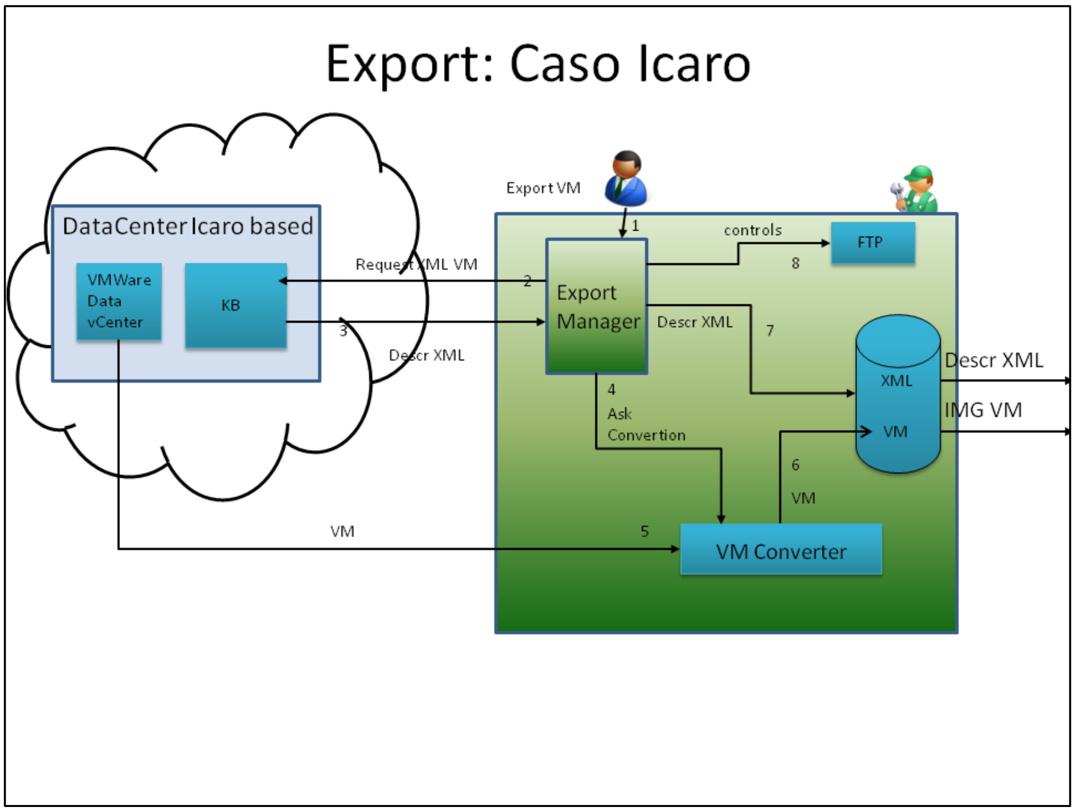


Figura 13 - Caso 1.b: Architettura coinvolta nella esportazione di una macchina virtuale su una struttura come il Cloud Icaro

Anche nel caso della fase di Export sono necessarie attività di monitoraggio.

5.5 Scenario 2.a: Importare una VM in Icaro

In questo scenario verranno prese in considerazione le problematiche che si devono affrontare nel clonare una macchina virtuale da un data center verso una struttura come Icaro, e quale è la soluzione proposta a riguardo tenendo in considerazione l'architettura Icaro e le funzionalità dei vari componenti.

Affinché una macchina virtuale possa esser clonata da un data center verso Icaro, è necessario avere una descrizione della configurazione della macchina virtuale e un sistema per trasferire l'immagine della macchina stessa.

Le informazioni che devono essere contenute nella descrizione devono riguardare principalmente la CPU, la RAM, la configurazione della scheda di rete, ed eventualmente la SLA e altri dati.

Si utilizza un file XML descrittivo nel quale sono specificate queste informazioni conforme con il modello di Icaro.

Il punto principale è quello di clonare la macchina virtuale.

Visto che al momento lo stato dell'arte non ci viene in aiuto per poter eseguire direttamente la migrazione da vCenter a vCenter su VMWare Web Client neanche sfruttando vMotion o la Cold Migration, occorre utilizzare FTP per poter clonare la macchina virtuale da un data center all'altro.

Vediamo adesso quali sono le accortezze e le necessità da avere nella parte Icaro. Partiamo dal descrivere come funziona la creazione di una nuova macchina virtuale e la sua configurazione.

Bisogna fare alcune premesse prima di descrivere la sequenza descrittiva della soluzione:

a) la VM arriva tramite FTP

b) il servizio di gestione Import Manager prende la VM e la inserisce in una Sand Box cloud usando il VMConverter.

b1) Se non lo fa automaticamente, un operatore può farlo manualmente.

Descriviamo adesso i passi principali per la fase di Import di una VM in un Data Center Icaro:

1. Viene richiesto tramite il Configuration Manager la macchina virtuale. (1.1) Il Configuration Manager prende il file descrittivo XML della VM dal disco dove è stata copiata tramite FTP, dove troverà informazioni aggiuntive riguardo allo spostamento della VM nella Sand Box (XML*).
2. 3. Nel caso in cui la configurazione debba essere chiesta alla KB, è necessario un ulteriore passaggio che porta il CM a comunicare con SC per poter prendere la configurazione della macchina virtuale da KB e ritrasmetterla al BP che si occupa di riconfigurare la macchina e metterla nel Data Center.
4. Il Configuration Manager passa al BP le informazioni per la macchina virtuale.
5. BP riconfigura la macchina virtuale nella Sand Box.
6. La clona nel Data Center utilizzando VMConverter. Ci può essere (6.1) una riconfigurazione una volta spostata dalla Sand Box al Data Center.
7. CM registra la configurazione, SLA etc, su SC, nella KB.
8. KB comunica con SM per il monitoraggio della macchina.
9. 10. 11. SM si interfaccia con il data center per monitorare la VM in questione ritrasmettendo i dati alla KB.
12. I dati sono disponibili anche per il Subscription Portal
13. Alarms provenienti da SM dovuti a malfunzionamenti si propagano fino a CM.

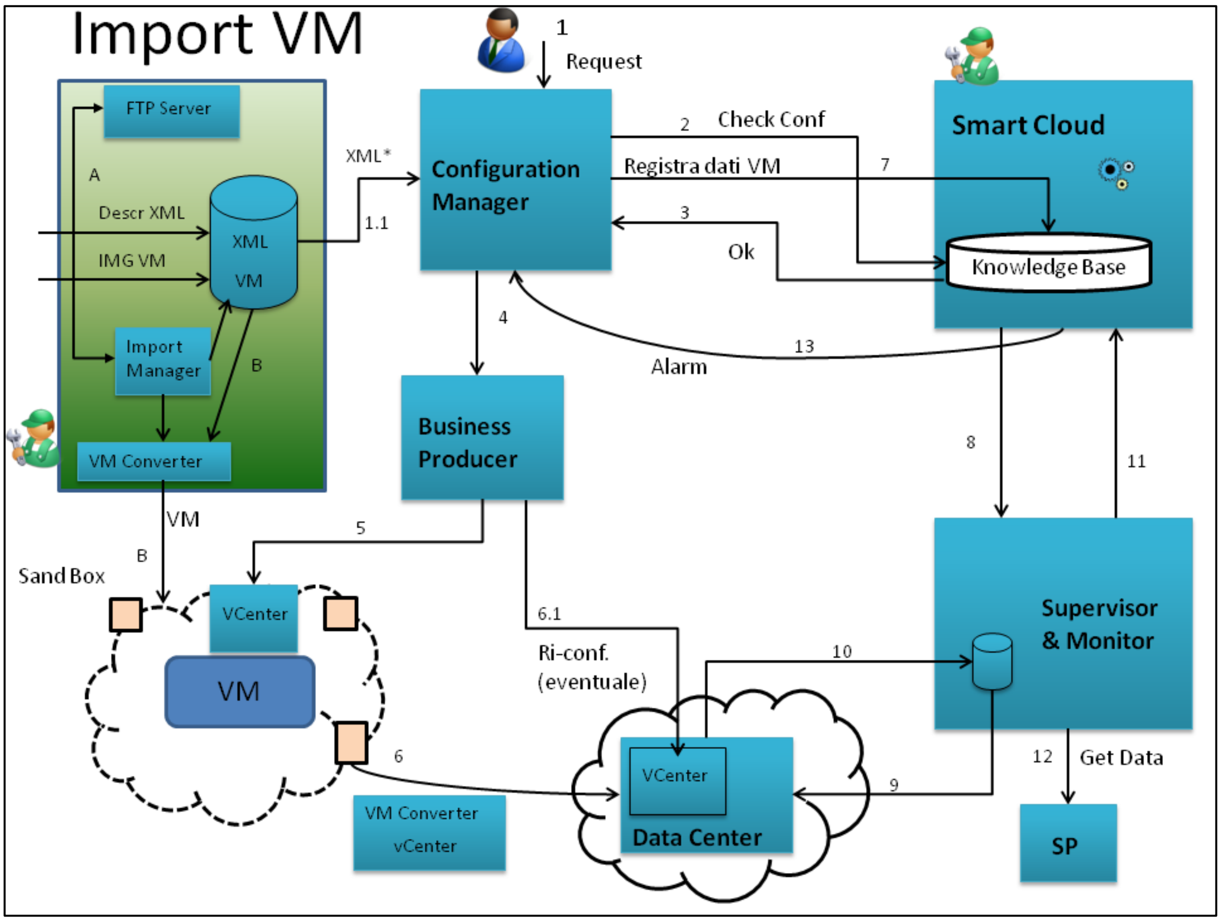


Figura 14: Architettura coinvolta nella fase di import

Rispetto alla situazione "normale", il passaggio da eseguire è quello della ri-configurazione della macchina una volta che è situata nella SandBox per spostarla definitivamente nel data center. La SandBox è una porzione di infrastruttura che replica un piccolo cloud per il test e la validazione delle applicazioni e dei servizi in preparazione.

Allo stato attuale, si opera direttamente su CM per far partire questa serie di azioni. Nello scenario che ci porta alla soluzione della clonazione si pensa di dover inserire nell'architettura un componente ad hoc che esegua questa operazione, nella fase di import e come vedremo nella parte successiva del documento, nella fase di export. Si può dire che possiamo assimilare la funzione di questa VM ad una frontiera nella quale arrivano e partono le VM migrate da e sul cloud.

Tale componente dovrà avere al suo interno un server FTP, un disco che possa ricevere l'immagine della VM, un lettore di XML per poter trasmettere la configurazione da e per la KB, un VMConverter che serve per creare la VM che inizialmente verrà posta nella SandBox su richiesta del BP. Inoltre, è necessario sviluppare un tool che possa comandare un FTP server anche per la fase di export della VM.

Focalizziamo l'attenzione su un aspetto importante di questo processo di migrazione, ovvero la fase di monitoraggio.

Se spostiamo una VM da un data center verso Icaro attraverso FileZilla, sarà necessario, visto il tempo che ci vuole per eseguire l'operazione, un continuo monitoraggio che mi dica istante per istante lo stato di avanzamento dell'operazione in modo tale che se qualche errore occorre, si possa ripristinare lo stato precedente.

5.5.1 I passi, analizzati più in dettaglio

5.3.1.1 Le funzioni del BP

Vediamo in dettaglio le diverse funzioni del BP Engine.

- Verifica della presenza di risorse per procedere alla produzione descritta nel flusso di deploy (attraverso il file XML caricato).
- Verifica che le risorse necessarie siano presenti e disponibili.
- chiama VMConverter per la creazione della VM che verrà spostata ne Data Center dalla Sand Box.
- Instanzia la macchina, gli applicativi, i servizi etc. Prima verifica, poi instanzia, in seguito registra su SM e aggiorna la KB. Se vi sono problemi, deve poter tornare indietro rimovendo le tracce dei vari passi.
- Per instanziare a livello IaaS ovviamente utilizza il gestore della parte della parte IaaS, vCenter.
- Inviare la configurazione alla KB in modo da registrare il vero deploy, sia come tipologia che come istanza specifica, con i relativi id di SLA, SLA stessa, i parametri, gli id del Cliente, etc.
- Registrare sulla KB quale sia la SLA di riferimento della configurazione specifica.
- Attivazione del BP Engine. Il BP Engine è lo strumento che mette in esecuzione un Workflow di deploy, lo interpreta. Può essere attivato dallo SCE o da SP.

5.5.1.2 Le funzioni di SM

SM si compone di alcune sottoparti:

- SM Configuration Manager è l'interfaccia tramite la quale SM riceve richieste per aggiungere il monitoraggio di VM, servizi, etc. Queste richieste arrivano dal BP durante la produzione di un deploy e/o di un cambio configurazione.
- SM Core, basato su Nagios, permette la configurazione del monitoraggio delle risorse a livello IaaS. L'arrivo di un nuovo host e di nuove schede di rete, la presenza di un nuovo storage vengono registrati tramite l'interfaccia di SM-Core. Qui dovrà essere registrata la nuova VM che verrà importata. SM-Core interagisce anche con il SM Configuration Manager per richiedere e collezionare dati di monitoraggio.

5.5.1.3 Le funzioni di SC

Lo Smart Cloud è composto da diversi componenti:

- Smart Cloud Engine (SCE) formalizza le funzioni di controllo sui dati di monitoraggio dei livelli IaaS, SaaS, PaaS del CMW che tramite SM finiscono nella KB.
- Knowledge Base (KB) si fonda sull'ontologia descrittiva del cloud Icaro, contiene e gestisce in un database semantico. La KB viene interrogata da un Reasoner.

KB è il componente che mantiene al suo interno tutte le configurazioni relative alla piattaforma sia riguardo ad aspetti statici di configurazione, che di aspetti dinamici sull'uso dei servizi/risorse, attraverso il Reasoner che è strettamente legato alla KB. Nuova conoscenza viene inferita tramite regole di reasoning implicite legate alla definizione dell'ontologia o esplicite legate allo specifico dominio applicativo.

5.5.1.4 Le funzioni di CM

CMW è una interfaccia Java con un'interfaccia REST che consente di gestire la richiesta di nuove istanze dei servizi applicativi richieste dagli utenti della piattaforma Icaro attraverso BP e rendere disponibili gli entry point per l'utilizzo delle nuove istanze.

Gestire e rendere disponibile per SM le metriche applicative relative all'utilizzo delle singole istanze dei servizi applicativi.

- CM/SLA - Editor: strumento per la definizione e formalizzazione della SLA in un formato formale XML.
- CM, tool per la valutazione delle possibili configurazioni del sistema, effettiva verifica della consistenza e della completezza tramite modelli formali. CM effettua la generazione della logica/ flusso di produzione/ deploy della configurazione scelta (eseguita da BP).

Verifica un confronto fra la SLA associata ad una configurazione e la descrizione della configurazione stessa potrebbe essere svolta dal Reasoner (SC), in modo da mettere in evidenza eventuali dimenticanze e/o inconsistenze. Questa verifica la può fare il Reasoner quando il BP invia la configurazione e la SLA di una nuova attivazione.

CM-Backend fornisce la configurazione di una nuova soluzione attraverso un'interfaccia fornita da KB (SC).

5.6 Scenario 2.b: Importare una VM in VMWare

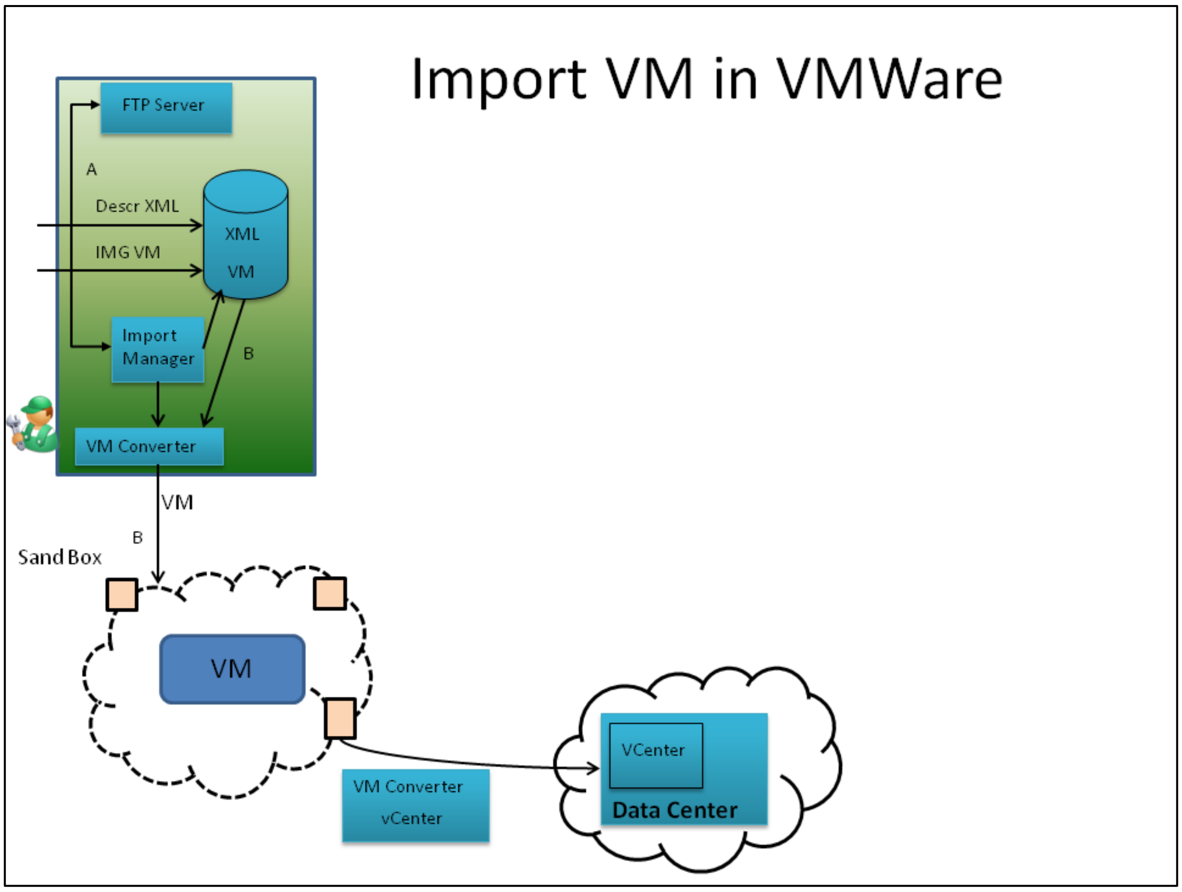


Figura 15: import di una VM in un data center VMWare

In questo ultimo caso la macchina dal momento in cui è inserita nella SandBox, verrà spostata direttamente nel data center attraverso una conversione di VMConverter, e impostata secondo le configurazioni del data center VMWare in cui viene importata.

5.7 Conclusioni

L'idea è quella di creare una VM che esegua entrambe le operazioni di Import/Export.

In pratica, nella fase di Import dovrà sostituire, o meglio integrare, quello che nella situazione attuale viene svolto da CM per creare una nuova VM. Ciò avviene nei passi descritti precedentemente. Nel caso Export, invece deve leggere tramite query SPARQL le impostazioni della KB del data center del Cloud sul quale migrare la macchina, oppure, nel caso vada esportata su VMWare, una interfaccia PHP per poter prendere le informazioni e impostare la configurazione.

Bibliography

[1] http://cloudaudit.org

[2] Planning the Migration of Enterprise Applications to the Cloud, CISCO Whitepaper, http://www.cisco.com/en/US/services/ps2961/ps10364/ps10370/ps11104/Migration_of_Enterprise_Apps_to_Cloud_White_Paper.pdf

[3] http://cloudmigration.com

[4] http://www.eaglegenomics.com

[5] http://www.hpcloud.com/solutions/migration

[6] http://www.racemi.com/cloud-migration

[7] http://www.rivermeadow.com

[8] A. Sheth and A. Ranabahu, "Semantic Modeling for Cloud Computing, Part I & II," IEEE Internet Computing Magazine, vol. 14, pp. 81-83, 2010.

[9] J. McKendrick, "Does Platform as a Service have interoperability issues?," in DOI: http://www.zdnet.com/blog/service-oriented/doesplatform-as-a-service-have-interoperability-issues/4890, 2010.

[10] Open Virtualization Format, http://www.dmtf.org/standards/ovf

[11] DMTF, http://www.dmtf.org

[12] OpenStack, http://www.openstack.org/ .

[13] Cloud Data Management Interface by SNIA, http://snia.org/cdmi .

[14] Open Cloud Computing Interface, http://occi-wg.org/ .

[15] Storage Networking Industry Association, http://www.snia.org

[16] The Internet Engineering Task Force, http://ietf.org

[17] R. Teckelmann, C. Reich, and A. Sulistio, "Mapping of Cloud Standards to the Taxonomy of Interoperability in IaaS," Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp. 522–526, 2011.

[18] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," Grid Computing, 2009 10th IEEE/ACM International Conference on, pp. 17–25, 2009.

[19] B. Rochwerger et al., "Reservoir—When One Cloud Is Not Enough," Computer, Mar. 2011, pp. 44-51.

[20] OpenStack, http://www.openstack.org/

[21] The Xen Hypervisor, http://www.xen.org/

[22] KVM - Kernel Based Virtual Machine, http://www.linux-kvm.org/

[23] Lxc Linux Containers, http://lxc.sourceforge.net/

[24] Microsoft Hyper-V, http://www.microsoft.com/enus/server-cloud/hyper-v-server/

[25]Amazon Machine Images, https://aws.amazon.com/amis

[26] OpenNebula, http://opennebula.org/ .

[27] XML-RPC, http://en.wikipedia.org/wiki/XML-RPC

[28] Eucalyptus, http://www.eucalyptus.com

[29] Nymbus, http://www.nimbusproject.org/

[30] vMotion: https://www.vmware.com/it/products/vsphere/features-vmotion

[31] VMConverter: http://www.vmware.com/it/products/converter

[32] VMWare vSphere documentation: http://pubs.vmware.com/vsphere-51/index.jsp#Welcome/welcome.html