


MEDITERRANEAN UNIVERSITY OF REGGIO CALABRIA
UNIVERSITY OF MESSINA
DEPARTMENT OF ENGINEERING
DOCTORAL PROGRAMME IN
"INGEGNERIA CIVILE, AMBIENTALE E DELLA SICUREZZA"

---

Curriculum :
SCIENZE E TECNOLOGIE, MATERIALI, ENERGIA E SISTEMI COMPLESSI
PER IL CALCOLO DISTRIBUITO E LE RETI

# STUDY, REALIZATION AND EVALUATION OF SYSTEMS FOR THE AUTOMATIC DEPLOYMENT AND SCALABLE DELIVERY OF SERVICES AND RESOURCES IN A FEDERATED CLOUD ENVIRONMENT

Doctoral Dissertation of:
**Eng. Alfonso Panarello**

Tutor:
**Prof. Antonio Puliafito**
[redacted]

The Chair of the Doctoral Program:
**Prof. Felice Arena**

2015-2016 – XXIX Cycle

# Introduction

CLOUD computing is a new paradigm able to provide on-demand services in a transparent way to users according to given pay-per-use constrains. These services are arranged by providers using distributed and virtualized computing resources by means of the virtualization technology that allows the decoupling applications from the physical machine on which they run on Virtual Machines (VMs). Furthermore, VM migration gives the opportunity to guarantee a high degree of Quality of Service (QoS). Thus, aggregating and mapping VMs, a Cloud provider is able to supply different levels of service: Infrastructure as a Service (IaaS), Platform as a service (PaaS) and Software as a Service (SaaS). However, considering the cloud computing ecosystem, besides large cloud providers (Google, Amazon etc..), smaller ones which are also becoming popular even though their own virtualization infrastructures (i.e., deployed in their datacenters) cannot directly compete with the bigger market leaders. The result is that often small/medium cloud providers have to exploit the services of mega-providers (vertically service exploitation) in order to develop services. Thus, a possible future alternative scenario is represented by the promotion of a cooperation among those small/medium providers, enabling the sharing of computational and storage resources. Doing a parallelism with the Internet, which

is considered a network of networks, the creation of a cooperative cloud environment (Multicloud environment) can not be simply considered as cloud of clouds. The clouds composing the "Multicloud" need to be regulated by policies defining their relationships in the same way the networks composing the Internet are governed by the policies which define access control rules. However the definition of such relationships is not simple because the "Multicloud" can theoretically consider infinite possible scenarios depending on the business model which the involved clouds want to accomplish. Behind these considerations there is the concept of *federation*, which is an indispensable requirement for the establishment of relationships between clouds. The Multicloud vision can be achieved with strong federation technologies providing gateways between different clouds and their internal data centers. Thus, a "Cloud Federation" can be defined as a relationship between two or more independent homogeneous/heterogeneous cloud platforms that establish a trusted federation agreement in order to obtain of a particular form of business advantage. This latter definition is quite generic and does not specify which type of relationship may be established between two clouds. In fact, guidelines regarding the design and implementation of the functionalities enabling Federated Cloud Environment are not precisely defined and still today several challenges about several issues are open and being faced. For example, some of these are: *(i)* how to plan strategies allowing the easy discovery of other cloud providers in order to establish federated partnerships; *(ii)* to find out solutions to solve the intrinsic technological diversity among the involved parts. In fact considering that the cloud providers can be heterogeneous, technology compliance is a sensitive topic and it involves the communication interfaces between clouds, the protocols used in cloud-based services, virtualization technologies (e.g., KVM, Xen, VmWare, Virtual Box, Virtul PC, etc); *(iii)* solutions for the trusted and secure access to the federated resources as cloud providers can adopt different security models and technologies (e.g., Username/-Password, Digital Certificate, PKI schemes, SAML, OpenID, Shibboleth,

etc), and so on. The proof that these topics are of sizeable importance for the IT community is given by the number of Cloud Projects that the European Union has been financing recently. Just to cite some of them we have RESERVOIR Project [1] and Contrail Project [2]. A detailed presentation of all the European projects focusing on the subject of this thesis will be provided in Chapter 6. Moreover, *Horizon 2020*, which is


**Figure 1:** *Horizon2020 Programme in ICT Sector*

the largest program ever made by the European Union (EU) for research and innovation, is investing billions of Euro into the research of futuristic and emerging technologies. In particular the Figure 1 shows the main challenges and the assigned budget for the IT research. It is possible to see that the challenges are perfectly overlapped with those of this thesis work. Particularly, one of the main ICT founded Projects concerning the Cloud Federation subject is BEACON [3]. This project has a two-fold goal: *(i)* the research and develop of techniques to federated cloud network

resources and *(ii)* to derive the integrated management cloud layer for enabling an efficient and secure deployment of federated cloud applications. BEACON Project counts several European partners including the University of Messina. In this thesis, Federated Cloud environments have been created to be applied in several types of cloud contexts in order *(i)* to improve the cloud service performances i. e., reduction of the application execution time; *(ii)* to overcome the technological limits of standards i. e., allows features to the existing technologies *(iii)* to figure out how to adapt the cloud provider offers to the user application requirements i. e., selecting the best solution (from the cost or performance point of view) for a given user service request.

For the exposed reasons, during the carried research activities it has been faced the analysis of ***federated*** cloud computing, dedicating a preliminary stage to the wide-ranging study of the *cloud computing* and to the countless research opportunities that it inherently brings. However, still today a universal definition of *cloud computing* does not exist. For this reason it has been carried out a thorough study of major scientific importance works on *cloud computing*, trying to acquire a general knowledge about the subject and the involved issues. This allowed the identification of the common features present in most of the *cloud*s and to acquire the needed knowledge to address an analysis on issues related to the distributed environments especially those composed of different and cooperating *federated cloud providers*. Therefore, in the second phase of the research activity it has been paid attention on a specific aspect that today is one of the topics that most fascinates the researchers in the IT field, i. e., the federation concept of cloud computing environments. Even this second phase of the study has two distinct but related activities. The first one concerns the study of the concept of *Federation*, which is a very general term and in turn leads several research opportunities. This study led to the publication of the scientific paper, "REQUIREMENTS ANALYSIS FOR IAAS CLOUD FEDERATION" [4], in the ***4th International Conference on Cloud Computing and Services Science***. The work will be deeply

argued in Chapter 3. In parallel to this research phase, it has been undertaken the study of the "*Apache Hadoop*" framework. This is a distributed computing system capable of processing and managing the so-called "Big Data". This calculation model is suitable for the parallel processing. A flowing step of the research focused vertically in the concept of Federation, by analysing a specific use-case namely the achievement of a federated environment among multiple Hadoop-based clusters. This second work led to the publication of a second scientific paper "CLOUD FEDERATION TO INCREASE ELASTICALLY MAPREDUCE PROCESSING RESOURCES" [5] in the "***Euro-Par 2014: Parallel Processing Workshops***". This work, therefore, is the cornerstone of all subsequent activities performed during the research period. In this regard it has been undertaken the study of the BigData Analitycs and Parallel Processing concepts and carefully studied the Apache Hadoop MapReduce Framework. This technology has an important role in ICT and opens new research horizons in several fields, such as medicine, security and prevention, finance, physics and economics. After a careful analysis of the above mentioned technologies the attention focused on the development of use cases involving in a synergic manner the concepts of Cloud Federation and BigData Processing in the context of Distributed Video Transcoding Services. In the wake of the scientific success the just mentioned scientific paper underwent important changes. Several system features, not yet implemented, have been added. These works led to the publication of the scientific paper entitled "COSTS OF A FEDERATED AND HYBRID CLOUD ENVIRONMENT AIMED AT MAPREDUCE VIDEO TRANSCODING" [6] in the "***ISSC'15 Workshops (MOCS) - (The Twentieth IEEE Symposium on Computers and Communications***."). The added new features to the system can be resumed in the following bulleted list:

- Features for the automation of the entire processing workflow.
- The provider selection functions within the federation through a phase of Discovery.

- automated retrieving of the video file to be transcoded from Amazon S3 Storage Provider by means the Amazon Java APIs.

Staying in the context of PaaS for video transcoding it is possible to say that the advent of social networks has brutally swept away the traditional media and has greatly accelerated the ageing of content flowing on the web and in particular through the Social Network platforms. The foregoing acquires even more fullness if we consider the creation of applications that promise to broadcast live fragments of the life of every user. The best known examples are the new applications such as *Periscope* and *Meerkat*. These applications, essentially, open a virtual window through which any user has the opportunity to observe the activities of other users life. The latest generation of smartphones offers hardware resources that allow watching and capture videos in High Definition. But not all users have got the latest model of smartphone able to support high transmission rates or viewing HD content. Moreover, users do not always have sufficient available bandwidth. So there is the need to allow all users to view videos shared in the best format possible, considering both their mobile hardware power than their available network bandwidth. Today many innovative protocols are available. They are able to on fly adapt the quality of the video stream to the quality of the network bandwidth and to the user smartphone hardware specifications. They are the so called *Adaptive-bitrate-Streaming* protocols [1] [2] [3] [4]. Based on this reasoning, it has been modelled another use-case that brings together all of the just exposed concepts. The work in question is titled FEDERATED SYSTEM FOR MAPREDUCE-BASED VIDEO TRANSCODING TO FACE THE FUTURE MASSIVE VIDEO-SELFIE SHARING TREND [7] and it has been published in the ***Fourth European Conference on Service-Oriented and Cloud Computing (ESOCC)***. The work will be discussed in the following

[1] http://www.images.adobe.com/content/dam/Adobe/en/products/hdsdynamic-streaming/pdfs/hds_datasheet.pdf

[2] https://msdn.microsoft.com/enus/library/ff469518.aspx

[3] https://www.iso.org/obp/ui/#iso:std:iso-iec:23009:-1:ed-2:v1:en

[4] https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/StreamingMediaGuide.pdf

chapter 4. After the development phase, in order to quantify and show the benefits of the federation in terms of temporal performance of the whole transcoding process, real testbeds have been performed. The subsequent studies and research activities also touched other sectors of the cloud panorama. The research, in fact, during the last year of studies shifted the focus on the issues of multi-tier applications deployment in cloud computing environments. In collaboration with the researchers of "**Institute of Architecture of Application Systems**" of the **University of Stuttgart**, it has been taken up a new line of research that brings together, both "Cloud Federation" and "Multi-tier Application Deployment" aspects. After a phase of study of the state of the art regarding the "Algorithm for the Best Selection of Cloud Providers in Federated Environment" subject (Chapter 6), it was modelled a system that allows an intelligent deployment of a multi-component web application over several Cloud Providers belonging to the same federation. This work, entitled "AUTOMATING THE DEPLOYMENT OF MULTI-CLOUD APPLICATIONS IN FEDERATED CLOUD ENVIRONMENTS", has been accepted at the "***10th EAI International Conference on Performance Evaluation Methodologies and Tools***" and will be deeply argued in Chapter 7.

All the details and descriptions of the investigated architectures, scenarios and case studies therefore will be provided, while also reporting and highlighting design choices.

# Contents

# List of Figures

# List of Tables

CHAPTER 1

# State of the Art

## 1.1 The Cloud Computing

THE last twenty years the IT world has been considerably shaken by the advent of the Internet. This model, which has brought new business opportunities and encouraged the emergence of new computing paradigms, has dramatically changed the face of telecommunications and business infrastructures and has modified deeply the culture of those peoples who have got in touch with it. In the last few years, researchers and developers of the IT sector focused their attention and energies on developing a new computing paradigm, the so-called *Cloud Computing*. The rapid technological progress and the consequent evolution of IT services allowed to offer to the end-user more and more efficient large scale services leading inevitably to an increase in management costs for the providers of such services. In fact, today a large part of the providers investments are used in the maintenance of such sys-

tems. Therefore it has been necessary to adopt more effective strategies to meet different needs. Indeed, there is a continuously increasing request of QoS (Quality of Service), and the need of lower costs to manage a growing user basin. This is the context where the Cloud Computing is placed. It is basically a new way for users and companies to approach the use of computing resources, storage and software applications. These resources, in fact, do not reside locally into a user PC or office cluster anymore, but they are allocated within the *cloud*, dynamically virtualized and mapped to physical hosts distributed throughout the network in a totally transparent manner to the end-user. The term *cloud*, in fact, means precisely the absence of effective information about the services implementation. Therefore the expression "cloud computing" refers to a collection of technologies that allows storing and / or processing data, using hardware or software resources distributed and virtualized on the network and offered to the customers typically in the form of on-demand services according to the client-server model. A cloud computing service offers interfaces that allow users to request a specified number of permitted operations, by totally hiding the underlying hardware and software architecture or features. In this way, users of a cloud computing system can think about their provider as an abstract entity that guarantees their services, without keeping them aware of how it does that and where their data is physically located. The user awareness loss regarding the information and the management of their own resources is the basis of the strong debate about the privacy and security offered by cloud computing systems. Against of this major problem, however, there are many benefits that cloud allows. Firstly it provides an abstraction of the hardware and software technologies that the user wishes to exploit, guaranteeing a certain level of reliability and system availability, enabling enterprises (especially small and medium) to concentrate on business without having to spend capital in the purchase and maintenance of hardware and software resources necessary for the implementation of their applications. Secondly it provides a reduction of costs to the provider (and thus to the

users), because the resource management is carried out in a distributed manner by algorithms allowing to optimally manage the computational capabilities of the machines, and then to minimize the costs of energy consumption, maintenance and so on. From the point of view of the large companies, however, the implementation of a private cloud computing system, internally to its own administrative domain, can be beneficial for all the reasons of reliability, availability, energy saving and maintenance of the above mentioned machines, without facing the risk to fall into the data security and privacy issues. The key concept on which the cloud computing is based is the *virtualization*. It consists of technologies that enable the creation of a virtual instance of a hardware platform, thus to provide to the low-level software layers, typically the operating system and its various components, an environment that faithfully replicates a physical device, composed of CPU, memory ram, storage drives and network resources. A virtual machine (VM), thus, can be understood as a logical representation of a physical machine (PM) consisting of hardware and firmware. Moreover, other advantages of virtualization applied to the distributed systems, are the ability to make backups and status updates, migrations of virtual machines from a host or server to another that, by saving the VM state, will be able to continue its running normally, as if nothing happened.

### 1.1.1 Key Features

THE NIST article [8] associated to the *cloud computing* has five essential characteristics that are resumed in the list below:

- *On-demand self-service*. No need of human interaction between the user and service provider. The user can in fact request and use autonomously the offered services;
- *Broad network access*. Network services are provided according formats adhering to the existing standards, thus they can be used on different platforms;

- *Resource pooling*. The physical and virtual resources are dynamically allocated to the users of the service, who have no vision of how those are physically distributed, but they know only a logical abstraction of the system;

- *Rapid elasticity*. Virtual and physical resources must be provided in a fast and dynamic allowing for easy system scalability; in order to allow an easy system scalability the virtual and physical resources MUST be provided as fast and dynamically as possible.

- *Measured service*. The use of resources is controlled and optimized by means of measurements carried out on the use of different offered service types in order to commensurate the cost of the service to its actual use.

### 1.1.2 Typology of Cloud Services

CLOUD computing is currently one of the major research topics in the IT world and, therefore, it is still now under development. Therefore a standard representing its architecture does not exist but it is possible to classify the main offered services as *IaaS*, *SaaS* and *PaaS*. The Figure 1.1 shows the general vision of the cloud computing from the user perspective.

Observing the Figure 1.1 from the top to the bottom, the first found architecture layer is the *IaaS* (Infrastructure as a Service). It is the lowest abstraction level and makes available to the users a virtual computing infrastructure for the execution of entire platforms and specific application, adjusting the charge according to the employed resources, that can dynamically grow or decrease, as a function of the actual load and of the requests to be served. Moreover the pay-per-use infrastructure includes all of the hardware needed on the network, for example servers, firewalls, switches with certain characteristics. The user will have to install his own applications on the infrastructure, configure and use them remotely, while the Cloud Provider must ensure the service provided by means the

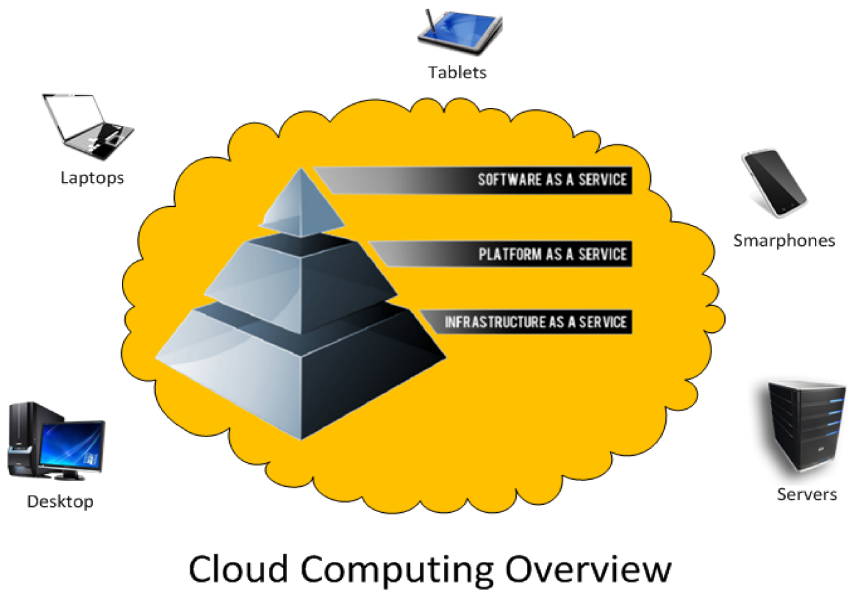



**Figure 1.1:** *Cloud Computing Overview*

infrastructure configuration, in order to allow its use to the customer. The Cloud Provider, moreover, had to guarantee the system maintenance and its replacement in case of damage to the machines. One of the most peculiar characteristic of the *IaaS* layer is the scalability property of the offered services. In fact, according to customer requirements it must be necessary to add computing power or storage resources to the infrastructure without the necessity for the user to reconfigure everything. This allows the user to start its business without having to estimate the infrastructure cost and then to jump in the market with incremental costs, without the risk of the loss of big capital to purchase large machines that may be unused and thus unnecessary and avoids the user to replace the machines in the future. With the Platform as a Service (PaaS) the companies offer to the customers hardware and software infrastructures for running their applications without the need to configure them but without all of the *SaaS* paradigm limitations. It is a kind of intermediate service between *IaaS* and *SaaS*. In other words it provides an integrated environment for developing, testing and maintaining of applications. The compromise for developers is to accept some restrictions on the available tools, APIs and platforms in exchange of a greater scalability and the lack of the

onus related to infrastructure management. The higher layer of the cloud stack is the *SaaS* (Software as a Service). This type of service can be implemented over the *PaaS* layer or directly on the *IaaS* one. The purpose is to allow remote access, typically via Web, to the services and functions offered by software, the use of which, also in this case, is subjected to the pay-per-use paradigm, although often many *SaaS* are offered for free by unloading the costs exclusively on advertisers (i.e. Gmail). Who uses these services will not need any installation or particular hardware resources on his local machine, to exploit the most of the *SaaS* potential. The user has only to accept the requirement of a relatively recent browser compliant to the currently used standards: *IETF* [9], *W3C* [10].

### 1.1.3 Cloud Typologies

REGARDLESS of the desired level of abstraction, a Cloud can be classified taking into account the considered operating context:

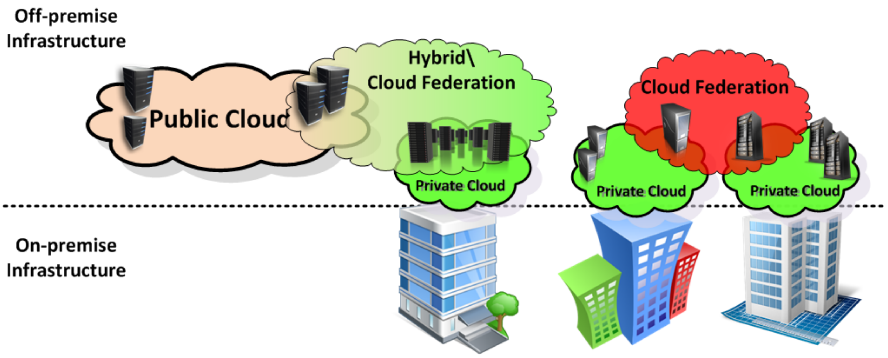



**Figure 1.2:** *Cloud Computing Typologies*

- *Public Cloud*: It is a computing infrastructure publicly accessible through the Internet network. The customer pays only for the actual usage of the service, amortizing the costs, because they are usually much lower than those they would have using the same "in-promise" service. An example of public cloud is Amazon EC2.
- *Private Cloud*: It is a computing infrastructure, whose suppliers and consumers belong to the same organization. It makes sense when

the advantages, in terms of reliability and availability, introduced by cloud computing paradigm are required, without the privacy issues introduced by the *public cloud*, because the use of the infrastructure is limited to a small number of users.

- *Hybrid Cloud*: It is a compromise between the Public and the Private Cloud. It is publicly accessible but in a limited way and tries to take advantage of both typology, with an increased complexity of the system and a greater difficulty of its realization and management. For example, a company might decide to use their private infrastructure to store and manage critical data, and instead to exploit the public cloud services for storing and managing no-critical data and information. It is possible to think of the *Hybrid Cloud* as a special case of another innovative concept in the cloud ecosystem, namely the concept of *Federation*.

- *Cloud Federation*: It is the newest paradigm of cloud and it descends directly from the *Hybrid Cloud* and according to which multiple providers are combined together in order to share resources with each other. *Cloud Federation* is that to which the IT world is heading, but this target opposes various difficulties of technical and legal nature.

### 1.1.4 The Infrastructure

THE logical organization of a cloud infrastructure counts four levels, each one referred to different types of entities (*Cloud Consumers*, *Cloud Management*, *Virtual Infrastructure Management* (VIM) and *VM Managers* (VMM)). Figure 1.3 shows the concepts described above. The first of the above cited layers brings together all the users of the IaaS service types accessed through the *Cloud Management* software (second layer). They ask for resources for different reasons: for example an individual user can ask for a virtual raw infrastructure, other clouds can need to outsource excess of workload or a PaaS Cloud service can have the necessity of resources in order to apply its services for providing

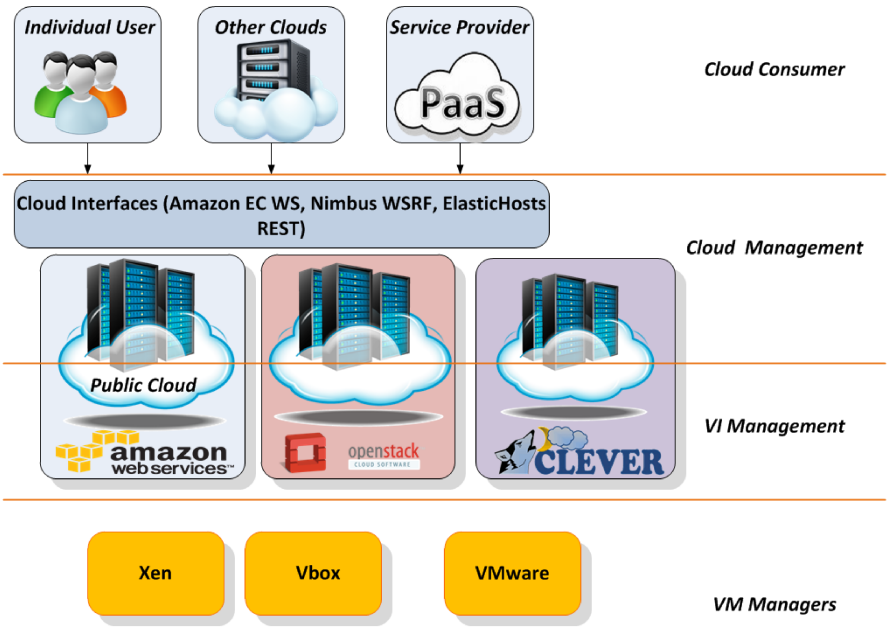



**Figure 1.3:** *Logical Cloud Computing Infrastructure*

them to its end-users. The second layer includes the "software", also known as cloud *toolkits*, to provide a remote and safe interface for the creation, control and monitoring of virtualized resources. Within the *Virtual Infrastructure Management* layer there are all the solutions that take charge of the low-level cloud computing infrastructure administration, providing primitives for the management and allocation of the users virtual machines over several physical hosts. The control of the virtual machines takes place within a single host by means of virtualization technologies that reside within the *VM Managers* layer. They guarantee the execution of simple operations on a virtual environment such as launching, interruption,resumption and destruction of a VM.

## 1.2 Existing Middleware Projects

THERE are different architectures or middleware solutions, both open source than not, dealing with the management of cloud computing systems. This paragraph provides an overview of the major existing open source pieces of middleware for

the management of a *IaaS* environment, evaluating their main features. *Nimbus* [11] is an open source toolkit that allows turning a set of computing resources into an IaaS cloud. Nimbus comes with a component called workspace control, installed on each node, used to start, stop, and suspend VMs. It implements disk image reconstruction and management, securely connects the VMs to the network and delivers contextualization. Nimbus workspace control tools work with *Xen* and *KVM* but only the *Xen* version is distributed. Nimbus provides interfaces to allow turning management functions based on the *WSRF* set of protocols. *Eucalyptus* [12] is an open-source framework that uses the computational and storage infrastructures commonly available at academic research groups to provide a platform that is modular and open to experimental instrumentation and study. Eucalyptus addresses several crucial cloud computing questions, including VM instance scheduling, administrative interfaces, construction of virtual networks, definition and execution of service level agreements (SLA) (cloud-to-user and cloud-to-cloud), and cloud computing user interfaces. *OpenQRM* [13] is an open-source platform for enabling flexible management of computing infrastructures. Thanks to its pluggable architecture, OpenQRM is able to implement a cloud with several features that allows the automatic deployment of services. It supports different virtualization technologies managing *Xen*, *KVM* and *Linux-VServer*. It also supports *P2V* (physical to virtual), *V2P* (virtual to physical) and *V2V* (virtual to virtual) migration. This means Virtual Environments (VEs) (appliances in the OpenQRM terminology) cannot only easily move from physical to virtual (and back), but that they can also be migrated from different virtualization technologies. *OpenNebula* [14] is an open and flexible tool that fits into existing data center environments to build a cloud computing environment. OpenNebula can be primarily used as a virtualization tool to manage virtual infrastructures in the data center or cluster, that is usually referred as private cloud. OpenNebula also supports public Clouds by providing cloud interfaces to expose its functionalities for VM, storage, and network management. *OpenStack* [15] is an IaaS cloud computing

project that is a free open source software released under the terms of the *Apache License*. The project is managed by the OpenStack Foundation, a no-profit corporate organization. The technology consists of a series of interrelated projects that controls large pools of processing, storage and networking resources throughout a data-center, all managed through a dashboard. *CLEVER* (CLoud-Enabled Virtual EnviRonment) [16] is a modular and pluggable middleware that specifically aims at the administration of private cloud infrastructures. *CLEVER* is able to manage cluster of nodes each containing a host level management module (Host Manager). A single node may also include a cluster level management module (Cluster Manager). All these entities interact exchanging information by means of the Communication System based on the *Extensible Messaging and Presence Protocol* (XMPP) [17]. The set of data required to enable the middleware functionalities is stored within a specific database deployed in a distributed fashion. CLEVER offers security, fault-tolerance and federated features.

CHAPTER 2

# Operating Context

In this Chapter will be presented all the technologies adopted during the research activities also providing their peculiarities and main features. Moreover, in order to better contextualize the thesis work, the concepts of BigData and Cloud Federation will be introduced. The latter will be examine in depth in Chapter 3.

## 2.1 CLEVER (CLoud-Enabled Virtual Environment)

CLEVER (Cloud-Enabled Virtual Environment) [16] is a project developed at the University of Messina. It was born as part of the *VI Management* layer and it provides all the tools to control and manage cloud infrastructure with a high level of scalability, modularity, fault tolerance and flexibility thanks to the use of a pluggable structure. This means that it is possible to add other

new supplementary features by simply programming and integrating new components or modules within the middleware architecture, without upsetting its organization. The CLEVER strengths are:

- a persistent communication between the middleware entities;
- transparency to users requests;
- fault tolerance for physical hosts and software modules;
- highly modular design based on the plug-in programming paradigm;
- high scalability of the system;
- automatic workload balancing through dynamic allocation mechanisms and migration of virtual environments.

### 2.1.1 CLEVER Architecture Structure

THE reference scenario, from which the architecture of CLEVER has been developed, consists of a set of physical resources interconnected each other by means of a network, that is a cluster of machines (hosts), where the virtual environments are dynamically created and allocated into the hosts depending on the load that each host has to manage. How it is possible to observe in the Fig-

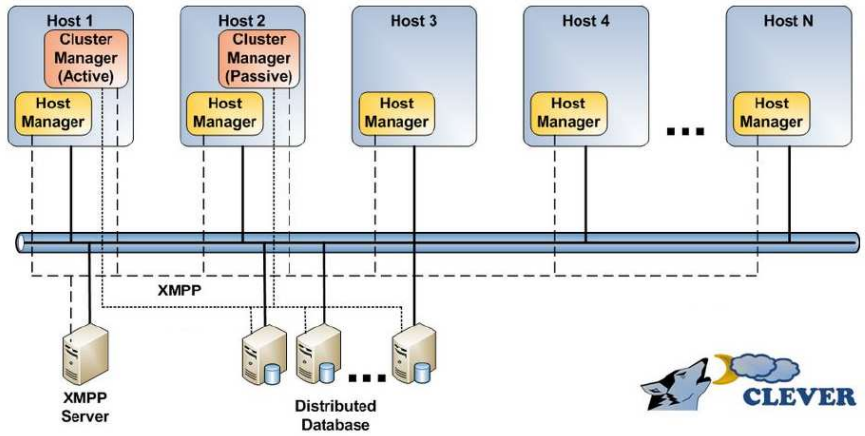



**Figure 2.1:** *An high level CLEVER architecture*

ure 2.1, the main components of the system are *(i)* a cluster of physical nodes (hosts), *(ii)* a XMPP server for the components communication and *(iii)* a distributed Database (XML-Based document oriented DB). All of the nodes have an "Host-level" management module, called *Host Manager* (HM), while only one node has an "Cluster-level" engine called *Cluster Manager* (CM). The two entities exchange messages via a *XMPP* protocol-based communication system; this justifies the presence of the *XMPP* server. All information necessary to operate the middleware is stored in a distributed XML-Document-Based database. The CM receives commands from clients, gives instructions to HMs, elaborates information and finally sends back results to clients. It also performs tasks for the management of cloud resources and the monitoring of the working state of the cluster. Communications among distributed components into a CLEVER Cloud environment are based on XMPP, due to its flexibility and high level of re-activeness. A *Jabber/XMPP server* provides basic messaging, presence, and XML routing features within the cloud. All the cluster hosts in the cloud are connected via a *Multi User Chat* (MUC) and cooperate according to the CM orchestration directives. In CLEVER, CM and HMs implements software Agents communicating through XMPP. Hence, it is easy to include new modules and functionalities to the CLEVER environment by adding new Agents and updating the CM and HMs configurations for the correct delivering of messages.

## 2.2 Apache Hadoop Framework

Hadoop is Apache project, but there are also some alternative distributions. The Apache foundation develops Hadoop in 3 different sub-projects: *hadoop-common*, hadoop-HDFS and *MapReduce hadoop*. The framework consists of two main parts: the first deals with the scheduling and parallel processing, called *MapReduce*, and second one instead deals with the management of the distributed file system, called *Hadoop Distributed File System* (HDFS).

More specifically, Hadoop *MapReduce* is a software framework to write and run applications to parallel process huge amounts of data (e.g. terabyte of datasets) on large clusters in a reliable, fault tolerant manner. A *MapReduce* job usually splits the input dataset into independent chunks, which are processed by the **map tasks** in a completely parallel manner. The framework sorts the outputs of the maps, which are then sent input to the reduce tasks. Both the input and the output of the job are stored in a distributed file system, that is the *Hadoop Distributed File System* (HDFS).

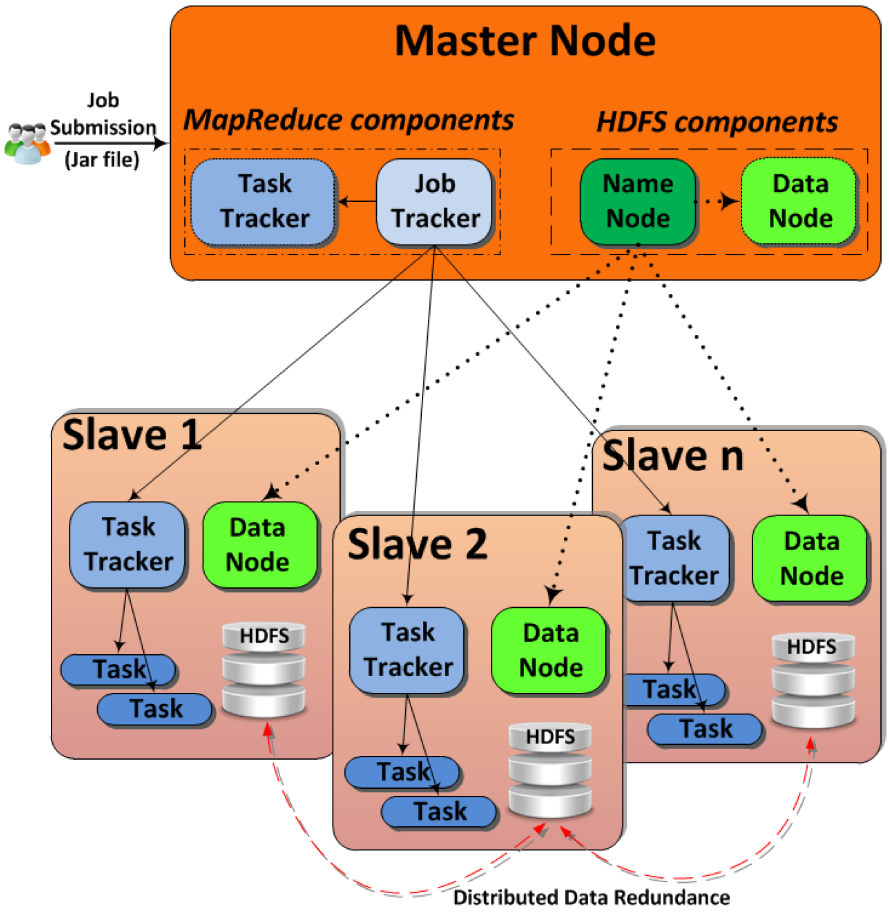



**Figure 2.2:** *Apache Hadoop architecture*

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the HDFS are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high

aggregate bandwidth across the cluster. The Hadoop framework has a Master/Slave architecture (Figure 2.2. *MapReduce* components consist of a single master JOBTRACKER and more slaves TASKTRACKER (one per cluster-node). The master is responsible for scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. The master node of the HDFS is called NAMENODE. It manages the namespace file system by maintaining a file metadata image that includes file name, location and replication state. It also instructs file system operations like opening, closing, and renaming files and directories. HDFS exposes a file system namespace and allows users to save their files. Within HDFS, a file is split into one or more chunks and these blocks are stored in a set of slave DataNodes. DataNodes manage storage resources into the host they run on and allow read/write accesses. A typical Block size is $64MB$. Thus, a HDFS file is chopped up into $64MB$ chunks, and, if possible, chunks are located at different DataNodes.

### 2.2.1 Configuration of Hadoop

THIS subsection is going to provide an explanation of the distributed configuration of Hadoop used in the project. For a more detailed explanation regarding all the properties it is possible to do reference to the official Hadoop documentation. The Hadoop version used in the project is the "release 1.0.4". This after unpacking and once specified the Java home directory is ready to run in pseudo-distributed mode. However, there is the necessity to configure the framework in a fully distributed mode. To this end it has been necessary to modify the configuration files of the framework.

#### 2.2.1.1 The file *hadoop-env.sh*

THE first modification was to specify the home directory of Java. This information is located in the configuration file "*conf/hadoop-env.sh*. The

performed operation in this file are listed below:

to remove the comment from the row

**# export JAVA_HOME**;

to add the Java location path: ;

**export JAVA_HOME=/usr/lib/jvm/java-7-oracle**

to remove the comment from the row

**# export HADOOP_CLASSPATH**;

to add the library folder path of Hadoop, namely **HADOOP_HOME/lib**:

**export HADOOP_CLASSPATH=/home/<USER>/hadoop-1.0.4/lib**;

Figure 2.3 shows the real hadoop-env.sh configuration used in the project.

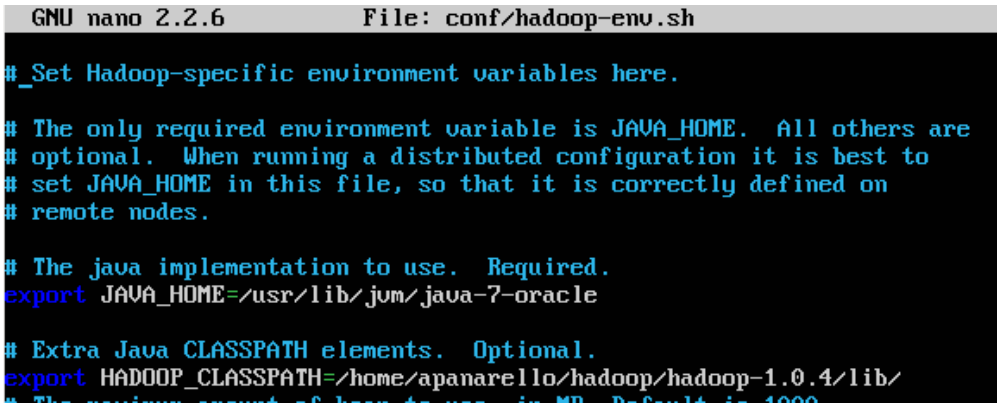



**Figure 2.3:** *hadoop-env.sh configuration file.*

#### 2.2.1.2 The file *hdsf-site.xml*

THIS file contains the information regarding the management of the HDFS. Hadoop, by default, saves the data in a temporary directory that will be cleaned at every reboot of the system. However, in order to avoid the loss of the data at system restart, it would be better to set-up a persistent path folder. The configuration file *hdsf-site.xml* contains several properties allowing the user to personalize the management of the distributed filesystem.

- The property *dfs.name.dir* refers to the directory where the *namenode* stores the meta-data useful to manage the distributed data.
- The property *dfs.data.dir*, instead, contains the folder where the *datanode* stores the blocks of the files.

For a correct use of the distributed data, both the previous folders must to have the permission set as follow *rwxr-xr-x* (755). All these properties must to be inserted within the tag "*configuration*" following the syntax showed in Figure 2.4.

```
 GNU nano 2.2.6          File: conf/hdfs-site.xml

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
   <property>
        <name>dfs.replication</name>
        <value>2</value></property>
        <property>
        <name>dfs.name.dir</name>
        <value>/home/apanarello/hadoop/hadoop-1.0.4/FILESYSTEM/name</value>
   </property>
   <property>
        <name>dfs.data.dir</name>
        <value>/home/apanarello/hadoop/hadoop-1.0.4/FILESYSTEM/data</value>
   </property>
</configuration>
```

**Figure 2.4:** *hdfs-site.xml configuration file.*

#### 2.2.1.3 The file *core-site.xml*

THE configuration file *core-site.xml* specifies the most important property of the system; namely the ip address of the systems **namenode**. It is useful to correct connect the slave nodes to the master nodo of the Hadoop cluster. This property is *fs.default.name* and it is formatted as follow:

**hdfs://namenode_hostname:listening_port**

The structure of the file is presented in Figure 2.5.

```
 GNU nano 2.2.6          File: conf/core-site.xml

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
   <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/apanarello/hadoop/tmp</value>
   </property>
   <property>
        <name>fs.default.name</name>
        <value>hdfs://nodo1A:8020</value>
   </property>
</configuration>
```

**Figure 2.5:** *core-site.xml configuration file.*

#### 2.2.1.4 The files *masters and slaves*

THE configuration files "*masters*" and "*slaves*" are used during the booting of the Hadoop system. The commands that call these files are: *bin/start-dfs.sh* and *bin/start-mapred.sh* The first command launches the "*namenode* on the current node. After the file "*masters*" is read and a *secondary-namenode* is launched on the ip-address present in such file by means an SSH connection to that ip. After that, the file "*slaves*" is read. It counts several ip-addresses (or hostnames). For each of them an SSH login is performed and the *datanodes* are launched on each slave. A the end of this phase the cluster is ready to be operative. The values

```
 GNU nano 2.2.6          File: conf/masters

nodo1A
```

**(a)** *"masters" configuration file.*

```
 GNU nano 2.2.6          File: conf/slaves

nodo1A
nodo2A
nodo3A
nodo4A
nodo5A
nodo6A
nodo7A
nodo8A
```

**(b)** *"slaves" configuration file.*

**Figure 2.6:** *"masters" and "slaves" configuration files.*

"nodo1A", "nodo2A" ... "nodo*nX*" (see Figure 2.6a and 2.6b) are the "hostnames" that are in turn resolved with the file "*/etc/hosts*" (see Figure 2.7).

```
  GNU nano 2.2.6            File: /etc/hosts

127.0.0.1       localhost
#127.0.1.1      nodo1A

# The following lines are desirable for IPv6 capable hosts

#CLEVER

10.10.110.101   nodo1A
10.10.110.103   nodo2A
10.10.110.105   nodo3A
10.10.110.107   nodo4A
10.10.110.109   nodo5A
10.10.110.111   nodo6A
10.10.110.113   nodo7A
10.10.110.115   nodo8A
```

**Figure 2.7:** *hosts file.*

### 2.2.2 BigData

THE concept of *Big Data* is very recent in the world the Information Technology (IT). The word *Big Data* can be associated with multiple definitions, depending on the considered context. It can be said that *Big Data* is the natural consequence of the change that the world is going through for around ten years. The unstoppable technological evolution is leading us to have today new connectivity devices, hardware and software resources that enable much of the world population to be constantly connected to the network, hence the advent of social networks, the exchange of files and information of many different types. All of this has generated in a very short time a impressive quantity of information, and therefore data, to the point that the conventional data management systems began to not be efficient anymore. Currently it does not exist a universally accepted definition of the *Big Data* concept. It may have a different definition depending on the operating context: for example someone talks about large amounts of business data, others refer to the huge amount of information generated by *social networks*. The

description that better explains the concept of *Big Data* is the work of Gartner analyst Douglas Laney, that describe *Big Data* based on three fundamental concepts: volume, variety and velocity, the so-called "*three-V paradigm*" [18]. But in the last years other two V have been considering, namely *Variability* and *Value*/*Veracity*

#### 2.2.2.1 5-V Paradigm Overview

PROBABLY the more intuitive concept regarding the *Big Data* is its size, its *Volume* (V). The amount of data produced in recent years is extremely large, especially if it is compared with the amount of data and information produced around *ten-fifteen* years ago. Until year 2000 they have been stored about $800,000$ *Petabyte* of information, and since that year until now the amount of data has increased to reach $30$ *Zettabyte*. For example, *Facebook* on its own produces about $10$ *Terabyte* of contents every day. If we consider how many people surf the web every day, making actions on it such as purchases, researches, downloads of files, sharing information and so on, then we can easy understand that the amount of daily generated data does not have any precedent in the past. In this scenario, the organizations must face a new problem never tackled in the past namely the incapability to manage and process the immense amount of data. Those organizations that do not yet have adequate tools to do it are literally overwhelmed by the data. The consequence is that today the percentage of the actually available data greatly exceeds that one the company are able to process. All this can be translate in a loss of information. The Figure 2.8 [19] show the trend of the available data, that the companies would be able to process, and the trend of the data that is really processed, highlighting the quantity of loss information due to the inadequate technologies for the Big Data management.

Another challenge that the modern data processing systems face is related to the "*second V*", the Variety of data. Simply it indicates all the format types in which the data can be represented. Every day, in fact, a


**Figure 2.8:** *The trend of available data against the percentage of processed data*

variety of sensors, social networks, mobile devices, etc. produce data with different formats making hard its processing by means the conventional storage systems. The third "V" of the paradigm is the "V" of *Velocity*. Often there is the need to rapidly precess the growing and heterogeneous amount of data. This because the data losses quickly the value of its information becoming obsolete in a few time. Another problem linked to the Velocity is simply the satisfaction of a client that wants to find in almost real time the most appropriate results to his request among the billions of data across the Internet. The fourth "V" is considered to be *Variability*: the meaning or interpretation of the same data may vary depending on the context in which the same data is collected and analysed. For example, let's consider a simple blogger statement such as "please, read the book", in case it is expressed on a blog of literature enthusiast the phrase may mean that the blogger enjoyed the book in question, if instead the exact same phrase is posted on a movie lovers blog, its meaning can change completely and indicate, for example, that the blogger did not like the movie of the book to which he is referring. The data value, therefore, resides not only in the data itself but is closely linked to the context in which we get the data. Finally the last "V" *Value*/*Veracity*: All these collected data represent a *value* for a company. From these data it is possible to seize opportunities and gain support for decision-making processes so that a company can have a big impact on its business. It is easy to understand that more data you have to be analysed access more information and value you can extract from it. However, the only volume

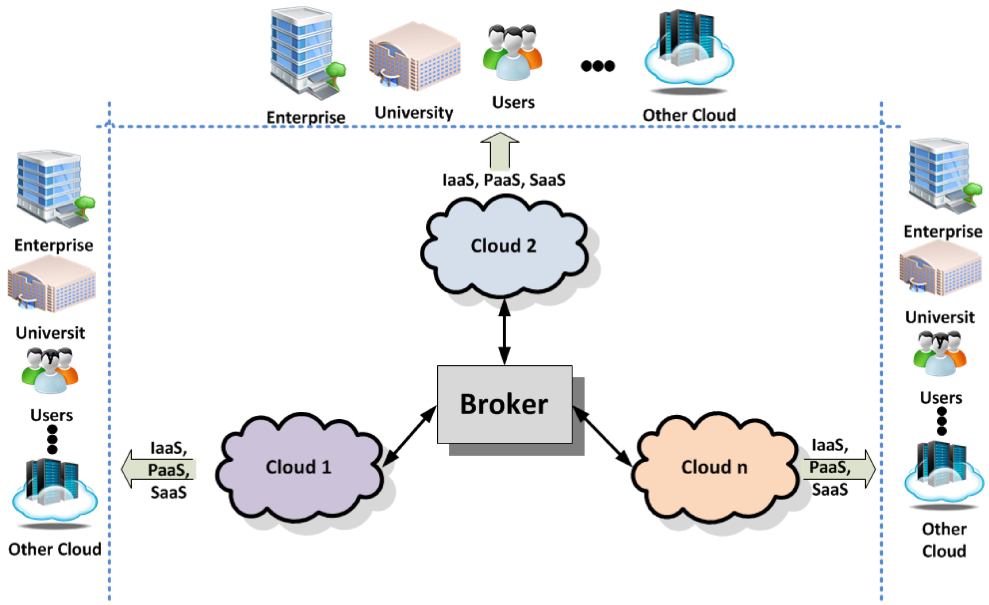
of data does not sufficiently guarantee the "quality" of the data: are we sure that these data are reliable and thus we can use them to support decision making in a profitable way? The accuracy and data quality becomes an important requirement for the data so that they can actually "feed" new insights and ideas and create a real value. Therefore the actual situation is that the companies are facing problems such as the capture, research, processing, sharing and viewing of this huge amount of data. The *Big Data* led, therefore, the need to find out intelligent strategies for the use of hardware and software technologies to process this data in order to renew the tools traditionally used for an efficient data management. It is exactly the context where most of this thesis is placed, namely *Cloud Federation for an efficient Big Data Management*.

## 2.3 Cloud Federation and Its Benefits

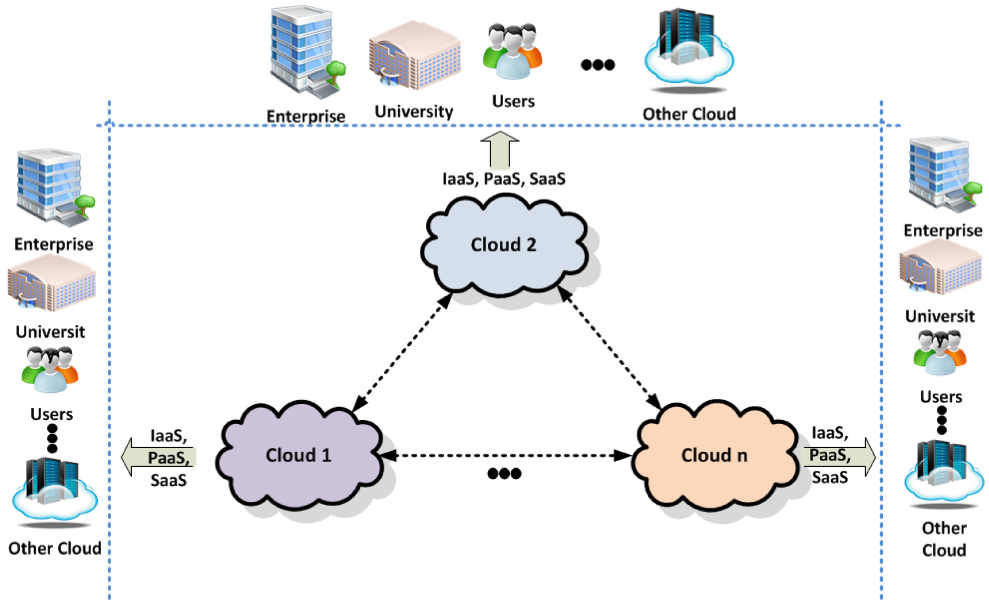
THE Federation concept finds its own roots in the political field and refers to a political organization where several "self-governing" entities are coordinated by a "Central Government" but, at the same time, keeping their "independence" from it. Adopting this organization model in the IT field we imagine a federated scenario where different clouds, belonging to different and independent administrative domains, interact with each other, sharing their available resources and playing themselves both the user and Cloud Resource Provider (CRP) roles at the same time. An in-depth definition of cloud federation together with its related benefits has been provided by the European project *RESERVOIR* [20] [1]. *RESERVOIR* is a European research initiative, which that focused the goal of developing technologies needed to address the scalability problem inherent to the single cloud service provider. In particular in the Rochwerger et al. [20] the authors present the *Reservoir* architecture giving an explanation of its main components. A *Reservoir* site is a Resource Service Provider that shares its resources with the other federated sites. A cloud service provider is an entity which

owns computational resources and makes them available for the service providers that in turn offer services to address the user needs. *Reservoir* architecture creates a dedicated virtual environment (*VEE*) where virtual resources (vm) can be executed. These *VEE*s can be placed in the *VEE hosts* within a single *Reservoir* site or over different sites. One of the main *Reservoir* elements is the *Service Manifest* (SM), which contains important information such as: master image references; information and rules to automatically create *VEE* instances without they negatively affect each other; resource requirements of a single instance (V-CPUs, Memory size etc.) and more. The SM will extend the *OVF* standard. Another basic component of the system is the the *"Service Manager"* (one per RESERVOIR site) that receives and analyse the manifest sent by the service provider asking for additional virtual appliances and together with the *VEE Manager* makes decisions about the allocation of the VEEs and their virtual resource. Therefore the *Reservoir* sharing and payment model (agreed between the federated entities) helps individual cloud providers to avoid both the "over-provisioning" (reducing resource costs) than "under-provisioning" (improving QoS) problems. The Reservoir project is considered the ancestor of the Federation topic.

In such a scenario the CRP makes its own unused resources available for the asking CRP, which at a given time needs external assets in order to be able to fulfill its users requests. Cloud federation offers two main benefits to CRPs. First, it allows providers to earn revenue from computing resources that otherwise would be underutilized or worse still unused. Moreover, Cloud Federation enables CRPs to virtually expand their assets making them able to face sudden spikes in demand without having to invest money in new hardware resources. There are several situations which push the CRP to join to a federated ecosystem: (i) a CRP temporarily runs out its own storage and computing capabilities (violating so the SLA); (ii) cloud needs particular types of services or resources that it does not hold; (iii) cloud wants to perform software consolidation in order to save energy cost; (iv) a CRP wants to move part of processing




**(a)** *Centralized Federation Organization*




**(b)** *Decentralized Federation Organization*

**Figure 2.9:** *Centralized Vs Decentralized Federation scheme*

into other providers (e.g., for security, performance, or for the deployment of particular location-dependent services). Obviously the interaction and cooperation among the entities of this scenario are too far to be easy and they still today need to be deeply investigated. Till today there is no universally accepted architecture for a federated environment. In literature

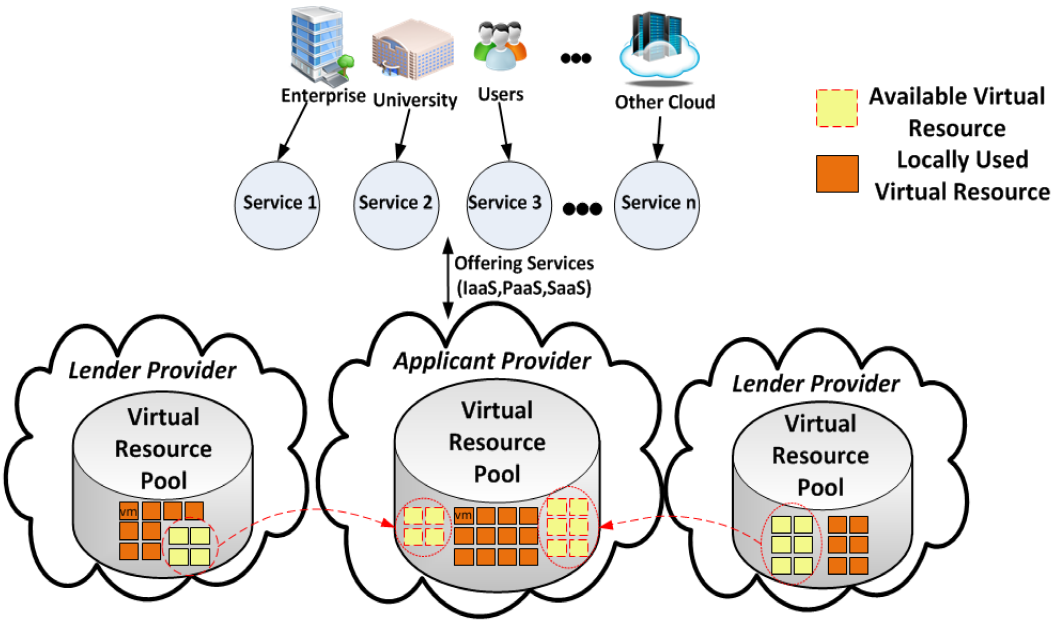



**Figure 2.10:** *Horizontal Federated Scenario*

there are two possible main federated scenarios, depending on whether it is centralized or not. In the first case, the federation among clouds is established, maintained and operated by one or more "third party entities" called "brokers" that allows a cloud to look for other clouds according to its own requirements. In the case of decentralized scheme instead all the maintenance, creation and management functions of the federation are held by each cloud belonging to that federation. The Figures 2.9a and 2.9b show the two possible schemes. The starting point of all the reasoning present in this work is showed in Figure 2.10 which is drawn from the work of Celesti et al. [21]. This work falls into the decentralized federation scheme (Figure 2.9b). The Figure 2.10 shows a scenario where a Cloud Provider (*Applicant Cloud*) that saturated it own virtual capabilities, in order to continue to provide services to its end-users, asks for external additional resources to the other Cloud Providers (*Lender Cloud*) into the Federation. In this manner, the *Applicant Cloud* will be able to instantiate its VMs into *Lender Cloud* physical data-center enlarging the amount of its available virtual infrastructure. So the rent resources which are physically placed into the Lender Cloud data-center will be logically considered hosted into the Applicant Cloud infrastructure. This brigs

a twofold benefit, first the *Applicant Cloud* can continue to provide its services with observance of the *QoS* (Quality of Service), and the *Lender Cloud*s can obtain an economic profit by lending their unused resources.

However the achievement of such a scenario is far to be easy. Therefore, in order to realise the previously discussed benchmark scenario a deep study of the requirements that have to be met has been carried out. It will be presented in the Chapter 3 Moreover, a more thorough investigation of the European Union effort about the cloud federation topic will be presented throughout the Chapter 6.1.

CHAPTER 3

# Study of the IaaS Cloud Federation Requirements and Proposal of High Level Architectural Solution

## 3.1 Federation Requirement Analysis

DESPITE of the obvious previously discussed advantages offered by Cloud Federation in Section 2.3, its implementation is not at all trivial. The main reason is that Cloud providers are more complicated than traditional systems and the existing federation models are not easy to be applied. In fact, while Cloud providers are typically heterogeneous and dynamic, the existing federation models are designed for static environments where it is needed an a priori agreement among the parties. With regard to *IaaS* Cloud providers, we identified several requirements that a federated environment has to meet:

1. *Identification of the actors in a federated system*: it is necessary to define which are the entities that cooperate each other in a federated

environment. In particular, it is necessary to exactly know who plays the role of the cloud system provider (Cloud Lender) and what is the role of cloud system consumer (Cloud Applicant).

2. *Automatism and scalability*: In a federated environment, a Cloud provider that accomplishes automated decisions has to determine which external provider has to be used for a particular workload, because not all Clouds are equal in terms of warranty, cost, reliability, QoS and resource availability. For example, a particular Cloud provider may be cheaper, but it could not provide guarantees of availability, making it unsuitable for mission-critical workloads. Another Cloud provider, however, could provide "five nines" availability but be more expensive. For this reason, a Cloud provider requiring additional resources, should be able to pick out the right Clouds provider which satisfies its requirements reacting also to sudden environmental changes;

3. *Versatility and Security*: *(i)* In a heterogeneous Cloud federation scenario, interoperability is a key concept. A Cloud provider requiring additional resources must be able to work with multiple Cloud providers based on different pieces of middleware. The ability of a Cloud applicant to be able to integrate different external Cloud providers (Cloud Lender) in a transparent manner allows to integrate computing, storage, and network services across multiple operators simultaneously. *(ii)* In order to achieve federation, it is fundamental the integration of different security technologies, for example, permitting a Cloud provider to be able to join the federation without changing its security policies. The Cloud systems require, generally, a simple, secure access to resources that they make available. The access to Cloud services is often achieved through web interfaces. Regardless of the model that will be implemented, an authentication system that allows a Cloud provider to access the resources offered by other operators maintaining its own identity is a necessary el-

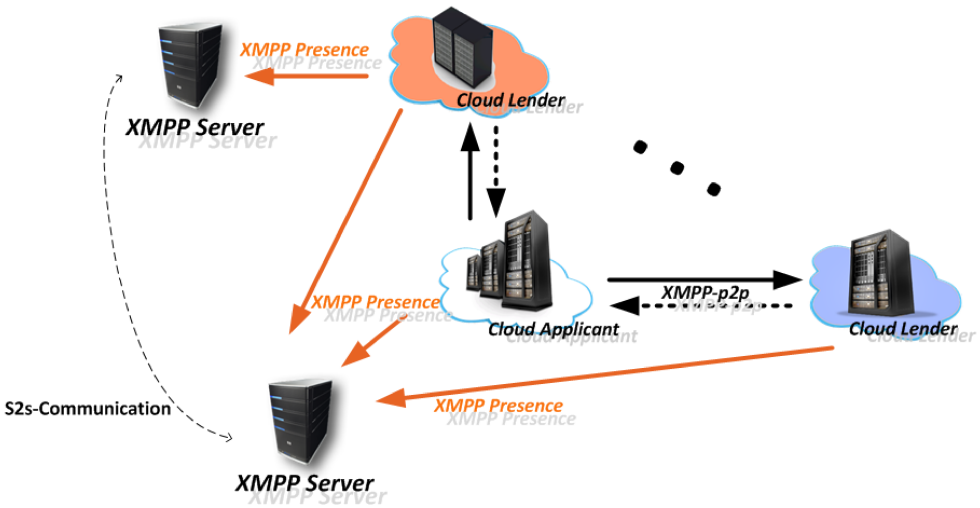
ement. Hence, a **Single Sign-On** (SSO) authentication system is required.

4. *Authorization*: In a federated environment it is necessary that the internal management policies of the Cloud operators coexist without interfering with each other. Nowadays, there is the need to apply access policies to user profiles and resources, because in a distributed system the access to a particular resource *must be controlled. Typically, these policies establish which are the entities that may access a specific resource in a distributed system according to a matching criterion that allows to check whether the requirements of the subject correspond to the requirements of the system.*

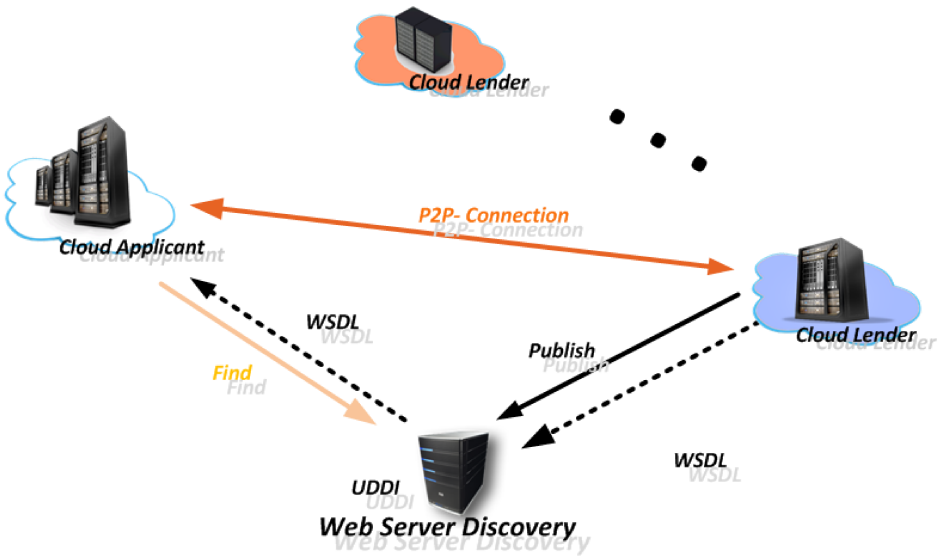
### 3.1.1 Cases of Study For the Federation Accomplishment

CURRENTLY, there are not standards that determine how the process of federation should be exactly accomplished. In this Section, in order to describe how the Cloud Federation, for the IaaS environment, can take place the Celesti et al. three-phase model [22] and several available technologies for its implementation will be considered.

The three-phase model implies three subsequent phases for federation establishment: *discovery*, *match-making*, and *authentication*. During the discovery phase, all the available Cloud providers have to be discovered. As this environment cannot be a priori known, but it is pretty flexible and dynamic, the discovery process should be implemented in a totally distributed fashion with a logical peer-to-peer (p2p) approach. After that, the matchmaking phase has to choose the more convenient Cloud providers with which to establish the federation. All the available (discovered) providers have to be associated to several policies, describing the offered resources, according to particular conditions. In addition, these policies have to match the policies of the provider requiring external resources. During the authentication phase, after that the Cloud providers have been selected for federation, a mechanism for creating a security




**(a)** *XMPP-Based Scenario*




**(b)** *WSDL-Based Scenario*

**Figure 3.1:** *Discovery Phase Solutions*

context among the involved Cloud providers should be accomplished by means of Single Sign-On mechanisms.

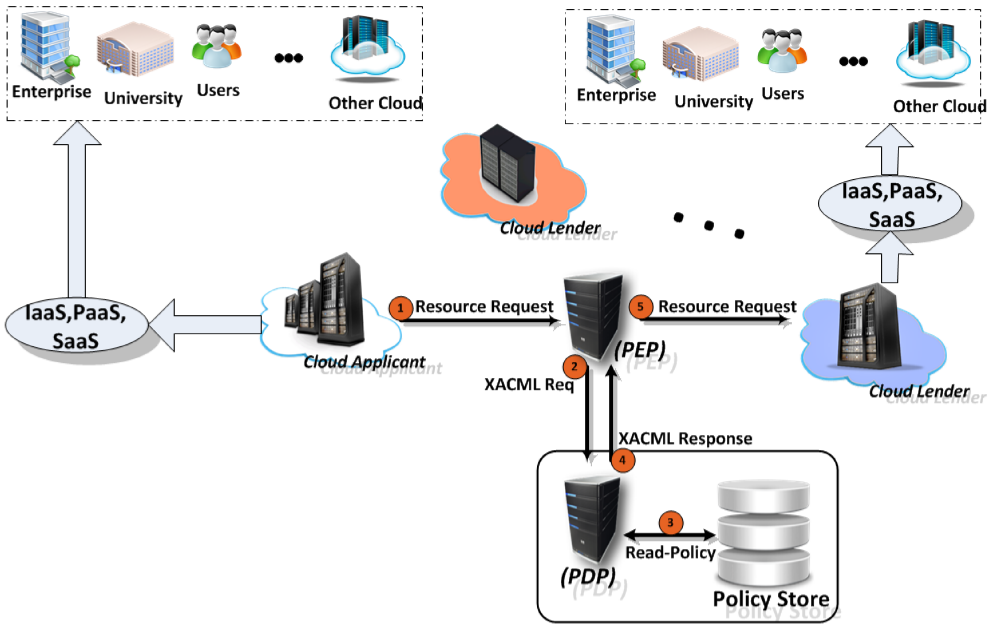
#### 3.1.1.1 Discovery Phase

THE Figures 3.1 depicts two possible solutions to the discovery problem consisting in using XMPP or Web Services. The architecture of the

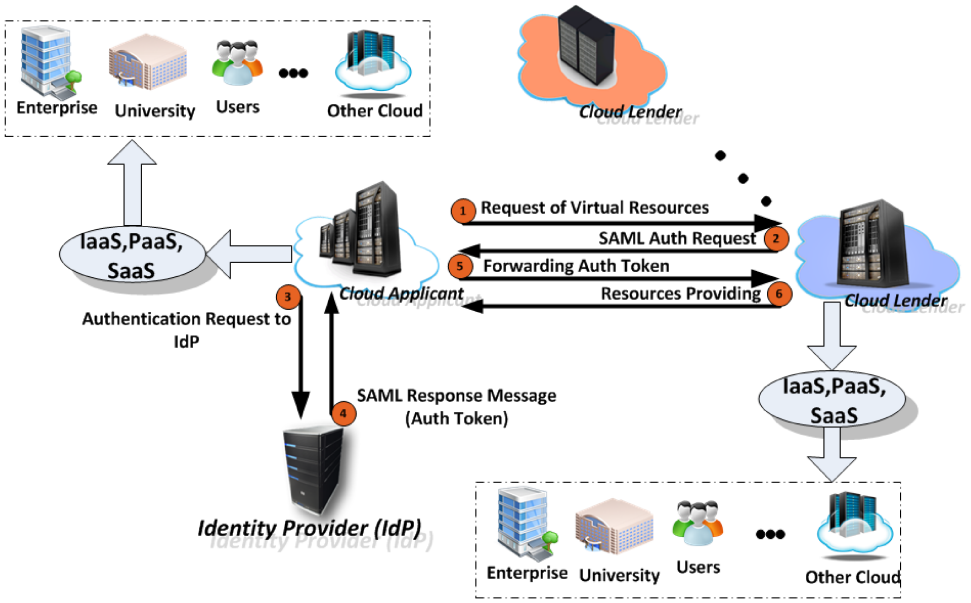
XMPP [17] network is quite similar to the approach used by the e-mail service. Every user on the network is associated to a Jabber ID (JID) and communicates with other users by means a chat-room. Scenario 3.1a shows a Cloud applicant that in order to gain knowledge about all the available Cloud Lenders, retrieves other providers availability information merely querying the "shared location" on which it is published. More specifically, in a XMPP-based scenario, the "location" is identified with a specific chat room, where the operators aiming to take part the federation hold a subscription. Scenario 3.1b instead shows another possible alternative solution based on Web Services. This scenario includes a *UDDI* web server that is a xml-based registry (a sorted, indexed database) which allows Cloud Lenders to publish their *WSDL* documents. A *WSDL* [23] document gives a formal description of the public interface of a web service. Therefore, a Cloud applicant can "read" the *WSDL* document related to a Web Service to determinate how the offered service can be used.

#### 3.1.1.2 Match Making Phase

DURING the Match-Making phase, it is needed to pick out one or more of the discovered operators. More specifically, the policies of the involved Cloud providers must coexist without affecting each other. Nowadays, in large-scale distributed systems there is the need to enforce policies in many different enforcement points. Typically such policies govern which subjects can access to a target resource of a distributed system according to a policy matching task performed in order to check if the requirements of the subject match the requirements of the system. To this end, a Cloud Provider (Cloud Lender) should be able to choose whether to accept or not a given Cloud Applicant and it must also be able to restrict such access as appropriate. One of the best solutions which is able to address the aforementioned scenarios is the *eXtensible Access Control Markup Language* (*XACML*) technology [24]. *XACML* allows




**(a)** *XACML Authorization Scenario*




**(b)** *SAML Authentication Scenario*

**Figure 3.2:** *Match Making and Authentication Phase*

to express policies by means of four major components: *attributes, rules, policies, and policy set.*

- Attributes are characteristics of subjects, resources, actions or environments that can be used to define a restriction.

- A Rule is the basic element of a policy. It identifies a complete and atomic authorization constraint which exists in isolation respecting to policy in which it has been created. A rule is composed by a *Target*, to identify the set of requests the rule is intended to restrict, an *Effect*, which is either " PERMITTED" or "DENIED".

- A Policy is a combination of one or more rules. A policy contains a *Target* (specified using the same components as the rule Target), a set of rules, and a rule combination algorithm.

Due to its nature, the *XACML* technology can be successfully used to achieve the match-making phase. Figure 3.2a shows how *XACML* allows to accomplish the match-making phase considering two Cloud operators. The actors of the above scenario are: a Cloud Applicant, a Cloud Lender, a **PEP** server (Policy Enforcement Point), a **PDP** server (Policy Decision Point) and a **Policy Store**. In the first step of the authorization process, the Cloud Applicant sends a resources request to PEP. The second step shows that the **PEP** translates the resource of the Cloud Applicant in a **XACML** request. **PEP** forwards the **XACML** request to **PDP** and asks whether the Cloud Applicant is authorized to access the requested resources. **PEP** (step 3) evaluates the policies in the **Policy Store** and takes the decision according to the defined policies. At the step 4 **PDP** returns to **PEP** a *XACML* response message which contains the taken authorization decision. At the last step (step 5), the **PEP**, depending on the *XACML* message received, grants or denies access to resources.

#### 3.1.1.3 Authentication Phase

THROUGHOUT the Authentication phase it is needed to establish a trust context among different operators by means of *SSO* authentication mechanisms. A single sign-on is the property of an access control system that allows a user to carry out only one valid authentication for multiple software systems and IT resources where he/it is enabled to access. In order to achieve such a goal, one of the major technologies, using the

Identity Provider/Service Provider (IdP/SP) model, is the *Security Assertion Markup Language* (*SAML*) [25]. *SAML* is an open standard based on XML for the exchange of authentication assertions among different parties. Figure 3.2b shows how the IdP/SP model can be used to establish a secure context between two operators. At the step 1 the Cloud Applicant tries to access a Cloud Lender. The Cloud Lender requires a *SAML* assertion type (step 2). Therefore, the Cloud Lender redirects the user to the IdP (step 3). The IdP authenticates the Cloud Applicant that responses with authentication information provided by the IdP (step 4). The Cloud Applicant forwards to Cloud Lender a resources request attaching to it the authentication token obtained by IdP (step 5). Cloud Applicant unpacks the request obtaining the token and verifies its correctness. Finally, the Cloud Lender contacts the Cloud Applicant notifying where and how to access the requested resources (step 6). The Cloud Applicant, subsequently, can request access the resources of other Cloud Lenders relying on IdP without further authentication tasks. Other valuable technologies used to accomplish the SSO authentication are OpenID [26] and Shibboleth [27].

CHAPTER 4

# MapReduce in a Federated Video Management Context

## 4.1 Motivations: a better management of Big Data

BIG Data needs to be managed in efficient way, these new solutions have to be conceived for speeding-up new services for Internet end-users. One of the main problems is providing meaningful techniques able to process even more data involving even more computational resources in distributed and scalable way. Cloud Federation may provide advanced features and capabilities useful for dealing with the massive data computation. To this end one of the main opensource MapReduce platforms in the context of Big Data, which is Apache Hadoop, has been selected to operate in a federated fashion. In particular the main contribution of this Chapter is the way on how to coordinate multiple Cloud Providers (CPs), having their Map-reduce platform, into a Federation in order to offer a highly scalable service for MapReduce

processing of Big Data [28]. In a federated Cloud environment, a CP can benefit of the storage and computational resources of other CPs acting on other administrative domains. To satisfy customers requests, each CP in the Federation asks for available resources to the other federated CPs offering their unused resources at that time. Of course, the amount of resources offered for each request can be regulated by specific Federation agreements, but such an issue is out of the scope of this work. A CP can require to establish a partnership with other CPs for multiple reasons: it has saturated its own resources and it needs external assets, it wants to perform software consolidation in order to save energy cost, it wants to move part of processing into other providers for improving security or performance in order to respect particular Service Level Agreements (SLAs). In particular, the attention focused on a federated Cloud scenario offering MapReduce processing service. Computational processes can occur on data stored either in a file system (unstructured) or in a database (structured). MapReduce can take the advantages of data locality, processing them on closer storage assets in order to reduce data transmission delay. The Hadoop usage is agnostic respect to the presented solution in Section 4.5, any other platform can use this approach.

The proposed solution exploits the Extensible Messaging and Presence Protocol (XMPP) technology to implement the communication among different federated CPs and adopts an external Cloud storage provider, i.e., Amazon S3, in order to minimize the overhead in terms of data transmission. As far as the communication system, the proposed solution integrates the **Hadoop** functionalities into a Cloud middleware for federated environments, called **CLEVER**. The choice of **CLEVER** is because, although it arises as middleware for the management of IaaS, it has been designed looking to the future and keeping an eye to the Federation issues. In fact, all communications, both inter-domain that intra-domain, use the XMPP technology, which in our opinion is a powerful solution to manage and to support the Cloud Federation. Thus, the aim of this Chapter is to provide a Platform as a Service (PaaS) for a MapReduce

processing of big data in a federated Cloud scenario. In particular, the proposed solution integrates the Hadoop functionalities into the the above mentioned CLEVER. Moreover, the video management use case (Section 4.5) allowed to drive our assessment.

## 4.2 CLEVER-BASED Federation Management

To deal with federated environments, the overall framework presented hereby is compounded from more complex parts. The core is represented by the **CLEVER** cloud. It is a challenging middleware because it fulfil many IEEE directives in the context of Cloud federation (see [29] and [30]). CLEVER accomplishes many features presented in these references. Other Cloud platforms like, OpenNebula or OpenStack have a weak approach in satisfying the federation. The XMPP protocol, on which CLEVER is based, natively supports federation capabilities. With CLEVER, each Cloud involved in the federation is identified by a Jabber ID (JID). As shown in Figure 4.1, in order to set up a federation, CMs belonging to different administrative domains exchange messages through the MUC (2.1.1) with the unique room ID *Federation* and only CMs of federated Clouds access it. The XMPP server is responsible to manage the Federation Room and it can be entrusted by a third part entity. Since the MUC can be only accessed by components into federated domains, it is necessary to check for their credentials. Credentials can be managed by the XMPP server itself or by external third party entities.

## 4.3 Integration of Hadoop in CLEVER

In order to make the Hadoop functionalities Cloud-like, as above remarked, it has been necessary the use of the virtual infrastructure provided by CLEVER. VMs run on HMs

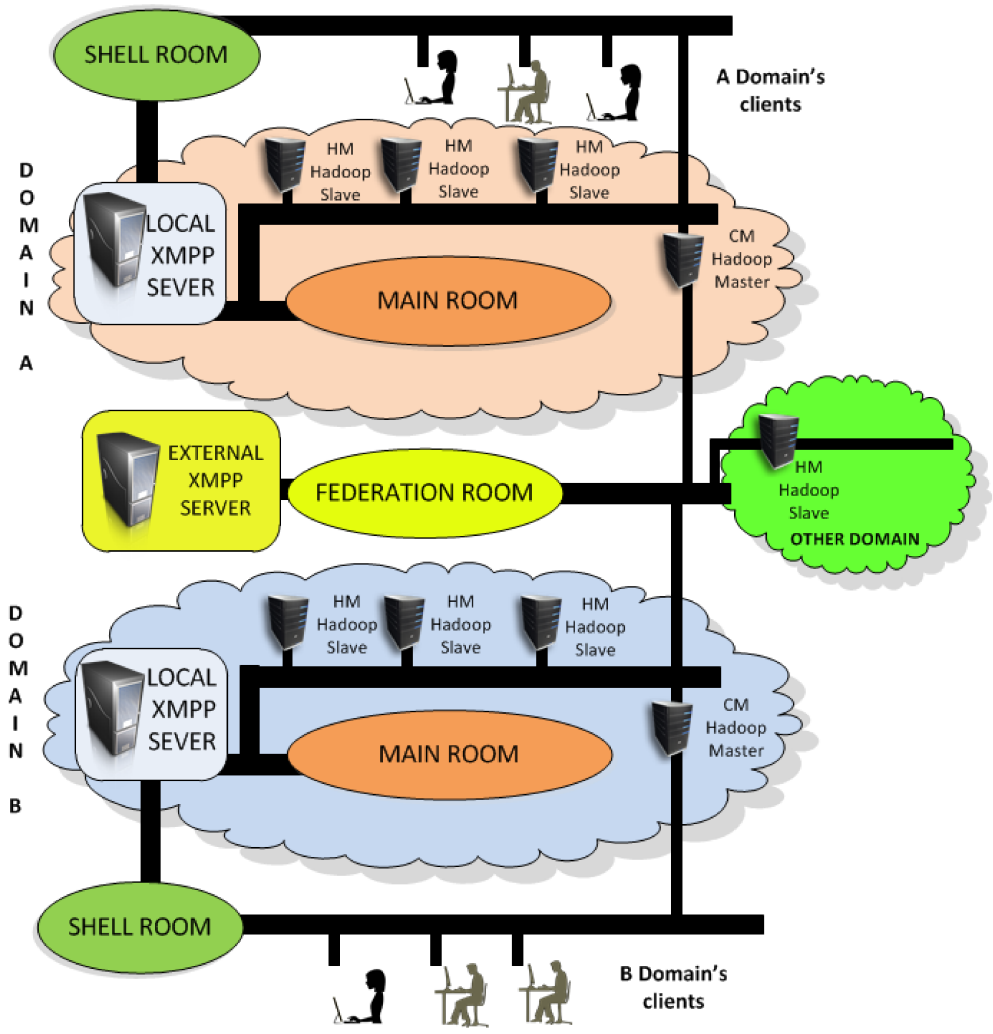



**Figure 4.1:** *CLEVER Federation Management.*

and work as slaves of the Hadoop cluster. Virtual Hadoop slaves are coordinated by the Hadoop Master arranged at the CLEVER CM. The first advantage of the integration of Hadoop in CLEVER is that, typically, Hadoop uses the TCP/IP layer for communication and it is a problem during the inter-domain communication due to heavy usage of firewalls by each domain that takes part to federation. In fact usually firewalls block inter-domain communications. So, integrating Hadoop in CLEVER, federation messages can be sent on port 80 thanks to XMPP technology. The second one advantage is that the system can automatically scale according to real time requirements. In CLEVER, the *Cluster Coordinator* (CC), inside the CM, is responsible for the cluster management and

service provisioning. To this aim, it interacts with both the HMs into the cluster, by means of an HMs interface, and the Cloud clients which request a specific service by means of the Client interface. All these interactions are based on XML message exchange into XMPP MUCs. Through the HMs interface, the CC communicates with all the HMs in the cluster, exchanging information on available resources, running tasks, work specifications and offered services. The CC makes use of the Client interface to interact with Cloud clients, in order to receive client requests, and to give back inquired services. The HM agent specifically designed to support the Hadoop activities in the Cloud is the HMN Agent. It provides the configuration settings to all the virtual nodes in the Hadoop cluster. The CLEVER HMN works as master for Hadoop cluster. Specifically, it implements the Hadoop functionalities to manage the Hadoop system. The Network Manager (NM) Agent allows to implement the virtual communications among Hadoop nodes through Notifications that arrives to the CM via the Dispatcher Agent. These notifications inform the HMN Agent about the presence or the absence of a host within the cluster.

## 4.4 The Public Cloud Amazon S3

THE Hybrid Cloud is accomplished using the Amazon S3 as Public Cloud storage service. It is designed to make web scale computing easier for developers. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It give every developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers. In the work Amazon S3 represents the common storage shared among the Federated Cloud Providers.

## 4.5 Use Case: Distributed Video Transcoding Processing Service in Cloud Federation

THE Social Networks users (SN) have begun to abandon the photo selfies sharing, turning their interest toward a new way to share their life. The new trend is the movie selfies sharing, live or not, which allows to show to the friends virtual small fragments of their own daily lives. Mobile users want to share their produced video in the SN such as Twitter, Google +, Facebook, etc. On one hand there are hundreds of millions of users that will put their selfie videos into the web and on the other hand there are other millions of users that will wish to become followers of one or more specific shared videos. For this reason an efficient cloud system providing a service able to manage all these video has to be discovered and designed.

The proposed scenario focuses on the *on-line video sharing*: the Social Network Cloud (SNC), received the users requests for a video sharing, interacts with the public Cloud Storage Providers (CSPs) for storing these users acquisitions. After that the SNC selects and contacts a specific CP and assigns it the Video Transcoding Job. This CP is defined *Cloud Broker* and it handles the entire work-flow. The choice to rely on external Public CSP as Amazon was made to minimize the overhead associated to the data transmission between the federated domains hence to be able to evaluate only the cost due to the Federation management.The idea behind such a service is shown in Figure 4.2. When a user requests to share a video-selfie, he/her contacts his/her SNC clud (i.e. Twitter or Facebook, see Figure 4.2) and uploads his/her video.The SN, in the future, could not be able to fulfil all the user video upload requests, supposing that it will vertically exploit the storage services of others CSPs (Amazon S3 in Figure 4.2 ) to face, in a scalable way, this growing demand for these kind of upload activities, in a logic where a composition of cloud services is accomplished. It is assumed that the input data to be processed are memorized in a CSP that supports *multi-part download*. The SNC,

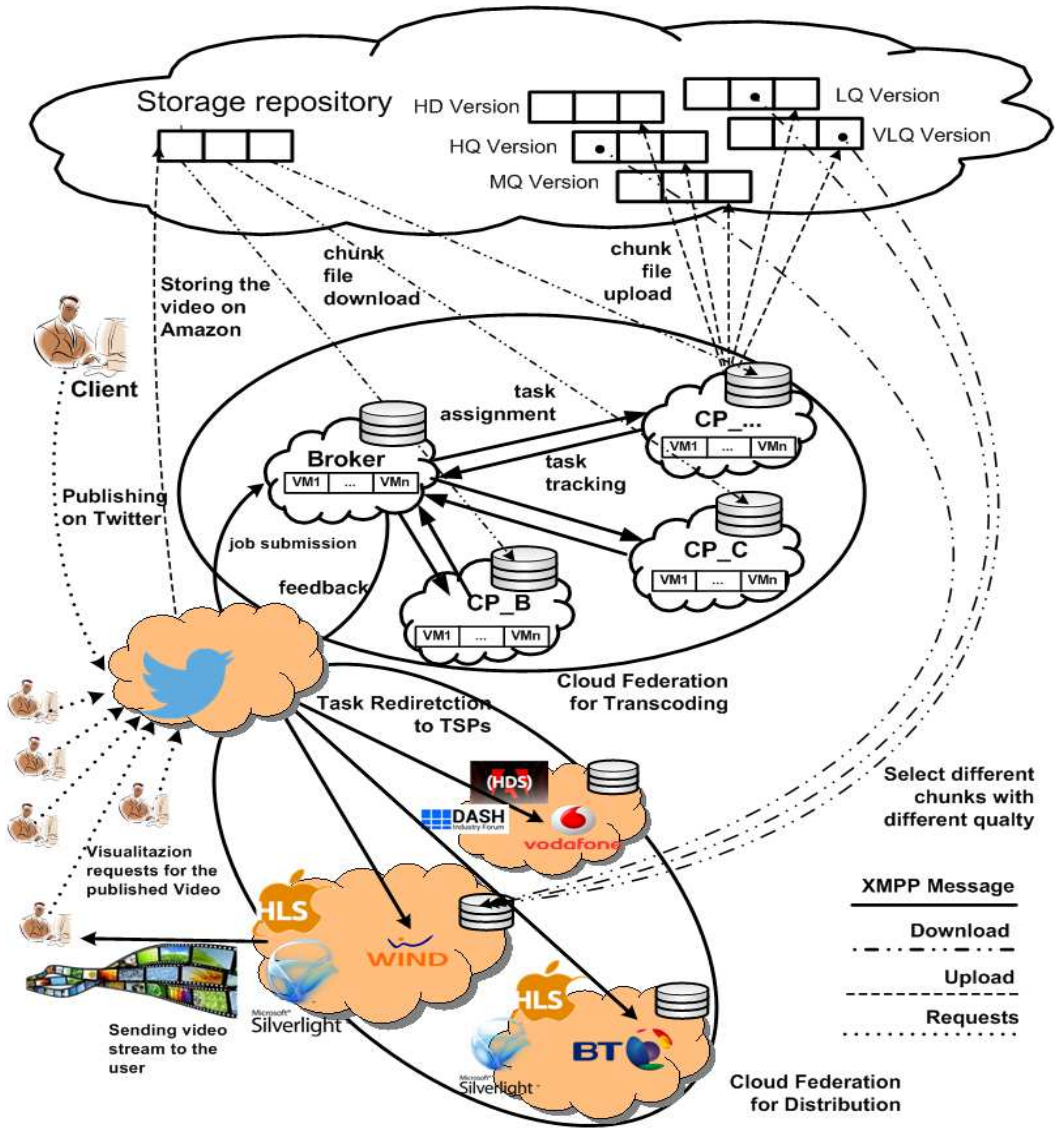



**Figure 4.2:** *Processing and Distribution service management.*

after the storing, in order to be able to allow all its users to watch the shared video in the best format possible, considering not only the users mobile hardware power but also the network available bandwidth at a given time and place where the users are located, has to transcode these shared videos in multiple formats and it has to be able to manipulate the greatest possible number of videos in the shortest possible time. To this end a processing service in Cloud Federation has been identified and exploited. Several CPs belonging to a Cloud Federation can offer

this Processing Service. This federation is called *"Cloud Federation for Transcoding"* as it is possible to see in Figure 4.2. This specification of the Federation name is necessary because the hypothesized scenario counts two different Federations: the first one, called *"Cloud Federation for Transcoding"*, having the purpose to execute the transcoding task in the shortest possible time; the second one, called *"Cloud Federation for Distribution"*, instead aiming to handle millions of requests for video visualization, shifting its burden both hardware and software to mobile operators who manage the users who make that request. In order to exploit this Federation service, after the storing of the video on Amazon, the SNC contacts the Broker of the Federation to submit the transcoding task. For simplicity, in this scenario, the broker (Broker in Figure 4.2) plays only the role of communication mediator, but it could have their own resources to be used for the transcoding tasks. Moreover it has been assumed that each CP in the *"Cloud Federation for Transcoding"* has one or more VM images including the piece of middleware for processing the task. It is important to say that the system can scale both horizontally and vertically. It scales horizontally when the Broker dynamically forwards the task requests to the federated domains (Foreign CP) and it scales vertically when a foreign CP decides in turn to forward the requests to others available foreign CPs or when it plays the role of Broker for another Cloud Federation. However, the Broker retrieves information about the availability of the federated domains and their available resources and then sends them the instructions (e.g. URL, size of the assigned block) to fulfil a task. As soon as CP_n receives the file localization information (i.e., the URL on Amazon S3), it starts the download of the file chunks and puts them (uploads) in its HDFS cluster for local processing. At the end of the processing step, CP_n stores the result of its processing in the CSP and sends to the Broker an end task notification. Once the Broker has received all the end task notifications from all the involved CPs, it communicates to the SNC the location of the multiple video streams and the necessary information for the client player to reach the desired

video streams. The steps accomplished to obtain the transcoded video are shown in Figure 4.3.

**Algorithm 1** Discovery Algorithm

```
1: procedure Discovery()
2:     List<String>DomainsInChat ← getDomainsChat();
3:     List<String>domainsInDB ← getDomainsInDB();
4:     int sizeDB ← domainsInDB.size();
5:     List<Object>domainsToDelete ← new List<Object>();
6:     for i = 0 to sizeDB − 1 do
7:         found ← false;
8:         for j = 0 to DomainsInChat.size() − 1 do
9:             if
10:                domainsInDB.getName(i)==( DomainsInChat.get(j)) then
11:                found ← true;
12:            end if
13:        end for
14:        if (!found) then
15:            domainsToDelete.clear();
16:            domainsToDelete.add(domainsInDB.getName(i));
17:            deleteDomain(domainsToDelete);
18:            dominiInDB.remove(i);
19:            i ← i − 1;
20:            sizeDB ← sizeDB − 1;
21:        end if
22:    end for
23:    if !(domainsInDB.isEmpty()) then
24:        HashMap <String,String> response;
25:        response ← new HashMap<String,String>();
26:        String Resources[][] ← new String[domainsInDB.size()][2];
27:        for i = 0 to domainsInDB.size() − 1 do
28:            Resources[i][0] ← domainsInDB.getName(i);
29:            new Thread(new DiscoveryThread(response)).start();
30:        end for
31:        while
32:            !domainsInDB.isEmpty() or (response.size() < domainsInDB.size()) do
33:            Thread.sleep(250);
34:        end while
35:        for i = 0 to domainsInDB.size − 1 do
36:            Resources[i][1] ← response.get(domainsInDB.getName(i));
37:        end for return Resource
38:    else
39:        String Resources[][] ← new String[1][2];
40:        Resources[i][1] ← "";
41:        Resources[i][2] ← ""; return Resources;
42:    end if
43: end procedure
```

As it is possible to see in Figure 4.2, and as previously mentioned, the scenario envisages the implementation of two different kinds of Federation. The "Federation for distribution" aims to lighten the system for what that concerns the requests video displaying management. The current trend will lead to millions requests of video sharing, and an even greater number of view requests. Thus, a solution to make the system scalable to grow of the huge visualization requests has to be implemented. To this end, a second kind of Federation has been implemented. When a

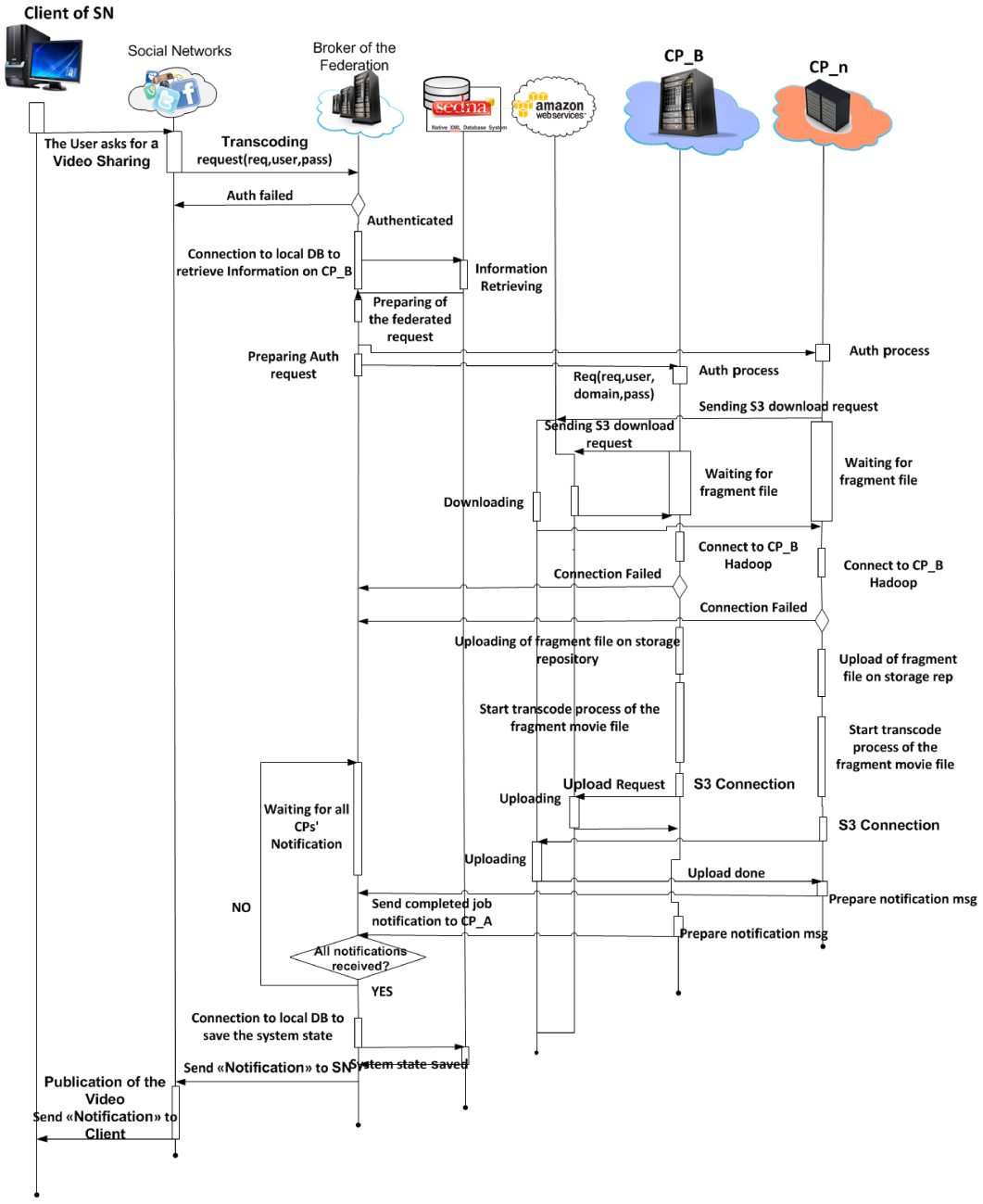



**Figure 4.3:** *Service Work Flow.*

user requests to view a specific shared video on "Twitter", he will first make a search for a specific #hashtag and after to have found that he will attempt to display it. To avoid that, in the distribution process, Twitter will become the bottle neck of the system, it will not handle the transmission of the video streams in first person, but it will redirect the view burden to the Mobile Phone Services Providers (MPSP) that has in managing users requiring that service. Each provider, in order to optimize the quality of the video visualization, according to the user actual hardware resources and network, will manage the visualization by means of an *Adaptive Stream Protocol* (for example HLS or Microsoft Smooth Streaming). Thus, in this way, the provider will act as a cache for the system, going to download and store locally only the required video stream fragments, thereby significantly reducing also the number of the accesses to the CSP (i.e., Amazon).

In order to accomplish such a federated service, the Broker has to pick out one or more CPs into the federation to assign the distributed job. The used discovery algorithm is shown in Listing 1. A short description of the algorithm is provided below. The code lines from 2 to 5 show the needed data structures to fulfil the discovery phase. The code, starting from line 6 to line 22, aims to make consistent the system state. Specifically, for each domain stored in *Sedna DB* it is checked if that domain is also logged in the Federation XMPP chat-room. If a domain in DB is not logged in the Federation XMPP chatRoom, it is removed from SednaDB. How said, it is necessary because the database of the system has always to be in a consistent state and moreover there is the need to be sure that all domains stored in the database are logged in the federation room too. The lines 24 and 26 define a shared hash map structure which is used to collect the responses of each federated domain (Key= DomainName, Value= n of the available nodes). In the actual state of the algorithm, the response simply contains the number of the available computing virtual nodes of each domain. The code lines from 27 to 30 show a simple for cycle. The Broker sends a XMPP message to each Domain, which is present contemporary

in the federation ROOM and in sednaDB, to ask for its available resources. Each thread call, by means the XMPP APIs, sends a XMPP message to a target Domain. All of the launched threads write into the hash map the value of the obtained response. In the lines from 32 to 34 the current thread is waiting for all the federated domains responses. If there are not Domains in the Federation, the discovery-agent returns an empty bi-dimensional array which will be managed in the other subsequent part of the code, else it returns a bi-dimensional array containing, in each row, the values pair: $DomainName - RespectiveAvailableResources$ (lines 35 to 42).

## 4.6 Cost estimation of the Federation

WITH reference to the previous section, where it has been made a high-level description of the process of managing the service in question, it is possible to identify eight steps of the multiple transcoding process. It starts at time $t_{t0}$ when a user sends a video sharing request to his SN where he has a valid identity registration (account). At time $t_{t1}$ the SN, exploiting a software CLEVER agent, which makes Twitter able to speak the XMPP language, contacts the Broker of the Federation to communicate the need to transcode a video and all the necessary information to perform correctly that task. At the same time it put the video to be shared and transcoded into the CSP (Amazon S3). At time $t_{t2}$, the Broker asks to each domain, which takes part to the Federation, how many VMs it can provide. At $t_{t3}$, the Broker performs a task assignment involving the whole federated environment. At $t_{t4}$, each involved federated CLEVER Cloud will exactly download only a specific part (Block) of the movie file precisely subdividing the download of that part in a number of chunks equal to the number of VMs that it has previously communicated to the Broker. This is possible using the *RESTful multi-part download mechanism provided by Amazon APIs*. At the time $t_{t5}$, each domain starts to transcode the downloaded chunks.

The $t_{t6}$ indicates the starting time when each CLEVER Cloud begins to upload the transcoded video chunks, in multiple format, on Amazon repository. Finally at the $t_{t7}$ time, the broker, after receiving all responses from the foreign clouds, starts to notify the end of the transcoding process. In order to obtain information regarding the actual costs of the Federation process it has been necessary to identify what are the steps that actually can have a greater negative impact, from the point of view of the delay and the overhead, on the whole process. In other words, it has been performed an analysis of the above mentioned steps, to figure out which of them represent the bottleneck of the entire executive process. The study led to the identification of the main tasks for which the sum of the times of each of them, in terms of delay and overhead, represents the total cost of the federation. With reference to the distribution process, instead, it is possible to identify three steps: at $t_{d0}$, the user, after made a #hashtag search, clicks on the video preview to watch the movie. The SNC, at $t_{d1}$, by analysing the users IP, redirects the visualization task to the appropriate (Mobile Phone Service Provider) MPSP. The MPSP at the $t_{d2}$ begins the download, from Amazon S3, of the required stream, adapting it to the users bandwidth by means the HLS or others Adaptive Stream protocol. Each MPSP will store in its own data-center only the required video stream chunks at the request time. The basic steps of the two processes are listed below:

- **Federation set-up**: This step was not pursued in the previous section as it takes one-off at the time of taking part in the Federation and therefore it does not affect the process described above.

- **Service Discovery**: Even this step, albeit critical to the above described scenario, has a negligible impact in terms of time if compared to the process lifetime. Furthermore it is a time that regards only the *"Cloud Federation for Transcoding"*. In fact in the *"Cloud Federation for Distribution"* there is not the need of a discovery phase: the participants are statically known. In fact while the *"Cloud*

*Federation for Transcoding"* is a dynamic environment, where the CPs can take part and leave the Federation whenever they want, in the *"Cloud Federation for Distribution"* it is not true anymore because it is an a priori federated environment where all of the participants are statically known.

- **Communication Cost**: this is the time that a XMPP message takes to reach its destination. It is independent from the number of cloud providers participating to the federation and therefore it is a negligible time too.

- **Download + Upload from and to Amazon costs**: these two phases of the process represent two of the most important impacts for the whole process. The download and upload steps have been grouped here together as both have similar features and despite some small difference in terms of time the two quantities are comparable. Considering a download speed of about $3MB/s$ and file size equal to $512MB$ there is a download times of about $200s$. While considering a speed of $2MB/s$ they have been obtained upload times that are in the neighbourhood of $270s$. Obviously increasing the number of the federated domains, these values of time decrease according to a pattern which can be approximated to the following function $f = a * x^{-c}$ (Figure 5.3. Thus, if the value of $x$ (domains) grows beyond measure, the download and the upload times tend to be respectively stable at the values of approximately $5$ and $9$ seconds, considering the hypothetical case of parallelization of $100$ domains and a file size equal to $512MB$.

- **Hadoop Cost**: This cost is related to the time needed to Hadoop to write data on HDFS and that necessary to read from HDSF and write on the physical file system. It is well known that Hadoop works better with a few large files rather than with many small files. In other words, from the point of view of the performance is better to write a $1GB$ file size instead to write 10 $100MB$ files

size [31], [32], [33], [34]. Furthermore, the time of such an operation does not decrease linearly with decreasing the file size taken into account. The trend of the time, depending on the file size, is similar to that one seen for the download times from Amazon. From the obtained measurements it has been observed that with files smaller than $20MB$ and with block size equal to $64MB$ (default Hadoop), the upload times into HDFS get highly comparable each other ($T_{UpLoad}20MB \cong T_{UpLoad}10MB \cong T_{UpLoad}5MB$) (Figure 5.5b,5.6b and therefore there are not benefit anymore from horizontal parallelization in terms of Hadoop cost.

**Computational costs**: This step is exactly the cost due to the transcoding time. In these tests, the video transcoding times measurements have not been performed. However, the OpenCV transcoding timing, reported in [35], has been considered. It focuses on measuring the total transcoding time varying (i) the size of the data, (ii) the number of cluster nodes and (iii) Hadoop configuration files values (block size and replication). The involved magnitudes are those that have the main impact on the timing of the total transcoding federated process. The values of the examined times are between a minimum of hundreds seconds for files of $512MB$ and a maximum of thousands of seconds to file $10GB$.

In the next section the times of the three phases of the process, which have the main impact on the entire transcoding federated flow, will be analysed.

CHAPTER 5

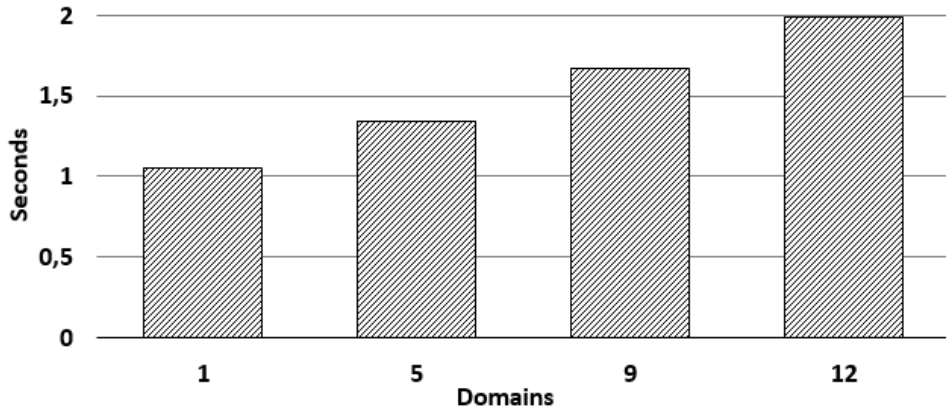
# Experimental Results

THIS Section analyses several real experimental test-beds to validate the proposed scenario (Chapter 4. The tests have been conducted considering thirteen different CLEVER/Hadoop administrative domains. Twelve of them act as federated Cloud providers and only one of them acts as Broker. Each domain, in the scenario, could be able to behave like both Broker and Cloud Provider. This Section shows that by adding domains to the Federation the total service lifetime will tend to the effective computational cost discussed in Chapter 4.6. Thus, the video transcoding is obtained in a time that is much smaller than it would have obtained without federation system. In fact, the Federation allows to horizontally spread the workloads, significantly reducing the overhead and delays that the transcoding process as a whole introduces. Specifically, these studies show that only using the parallel processing provided by Hadoop it is possible to achieve a reduction of the computation time, instead by means more federated Hadoop-based

environments, adding a horizontal cooperation, it is possible not only to reduce the computational time related to the video-transcoding but also those related to the delays and overheads introduced from the other phases of the process. These tests give us information about behaviour time of the whole system. In particular, the arranged test-bed take in consideration the following cluster configuration:

- 13 physical servers (one per CP);
- each node of each cluster is a VM with the following virtual hardware and software: 1 CPU, 768 MB RAM, Ubuntu OS, CLEVER middleware including the Hadoop plug-in.

Experiments have been conducted with the following physical hardware configuration: CPU: AMD Opteron 2218 HE Santa Rosa with two Dual-Core 2.6GHz processors; 8GB RAM, running Linux Ubuntu 12.04 x86_64 OS and VirtualBox. The transcoding tool used in Hadoop is the OpenCV framework converted in MapReduce shape. Each experiment was repeated 30 times in order to consider mean values with 95% confidence intervals. In the following, the main phases involved during the experiments have been considered.




**(a)** *Average time required to retrieve domain information on Clouds*

**Figure 5.1:** *Retrieving information and forward request times*

Figure 5.1a shows the average time required for the accomplishment of the Discovery Phase ($t_{t3} - t_{t2}$). (For the intervals of time to do reference

to the Section 4.6). It is possible to observe that increasing the number of domains, the time required to complete the discovery phase increases too. This behaviour is because the algorithm, which manages the discovery, is implemented in a multi-thread fashion. In this manner it has been possible to make parallel the information retrieving of the available resources of each domain. So this growth of time is related to the time needed to create a new thread. However, this behaviour is negligible but it could begin to have a little negative impact if the number of federated domains becomes extremely large.

At phase $t_{t5} - t_{t4}$ each domain performs two tasks; at the first it downloads the assigned file block and then writes it into HDFS. Observing the Figure 5.3a it is possible to notice that, if there is only one domain taking part to the federation, it has to download the whole video file. When there are more than one domain into the federation, each of them has to retrieve only a part of the original file. To better understand the Figure 5.3a it is necessary to consider two different parallelization levels. The first one is that provided by the horizontal federation. In fact the download is parallelized among more domains.

The Broker, after the Discovery Phase, known the number of the federated domains, will assign to each of them a different block of the file that has to be downloaded. After that, it must be considered the second parallelization level, namely that is provided by the Hadoop cluster. Each domain, depending on its own available VMs (Task Trackers), splits again the download of that block in a number of chunks equal to the number of its available VMs. Figure 5.2a explains a possible way of the splitting process of the file. In this case the whole file will be partitioned in a number of chunks equal to the total sum of the available VMs into the federation. Thus, to each domain will be assigned exactly a number of chunks equal to its own available VMs. All the chunks have the same size. It is clear that in this manner there is a balancing from the processing point of view. Figure **??** instead shows another solution to perform the job assignment. The whole file is split in a number of blocks equal to the

number of the federated domains. Each block is after spit in a number of chunks related to the VMs of the specific domain. This solution balances the download process allowing the domains to download the same block size. These reasoning are needed because the MapReduce job assigns to each available processing node a different chunk. In particular Figure 5.3a shows the download time from Amazon S3 considering both parallelization levels. In other words, the figure shows the download times varying both the file size and the number of the chunks. Observing the graphs, the download time for the whole $512MB$ file takes roughly $200seconds$ while the time needed to download an eighth of file ($64MB$ - eight domains) takes roughly $30seconds$. Each download takes place in parallel; this brings a double benefit, the first one due to the smaller blocks size that have to be downloaded, the second one due to the parallelization of the download of these blocks. How stated in the previous section, if the number of domains grows beyond measure, the download time tends to remain stable around to a value of approximately $5seconds$ considering the hypothetical case of 100 domains that simultaneously download different parts (about $5MB$) of the same file with original size equal to $512MB$. Figure 5.3b, instead, shows the time needed to upload the transcoded files into Amazon S3 repository ($t_{t7} - t_{t6}$). The trend of such a time is similar to that one shown in Figure 5.3a. The upload time is not dependent on the number of chunks of the file block.

Observing Figure 5.3a, it is possible to see that increasing the number of the chunks the download time increases too. In other words, the time to download a monolithic file is smaller than that one needed to retrieve the same file using the multi-part download Amazon S3 features. This behaviour has been attributed to the overhead time needed to find the correct starting byte of the chunk to be downloaded. This overhead depends by two factors: the number of chunks (more chunks mean more research operations into the file) and the file size; because bigger is the file longer is the time it takes to move to the correct byte of the selected chunk. Considering the first case, keeping stable the file size and increasing the

number of the chunks, a growth of the overhead of a constant factor has been observed. In particular it consists of about $3,5$ additional seconds per chunk considering $512MB$ file size, about $2,5$ additional seconds per chunk considering $128MB$ file size and about $0,3$ additional seconds per chunk considering $8MB$ file size.

However, this overhead decreases when the file size decreases too. In fact, considering a small file and changing the number of the chunks the download time remains approximately the same. The overhead follows the logarithm trend and it is shown in Figure 5.4.

Figure 5.5a shows the average upload time of blocks of files into the HDFS of each domain. This time changes according to the video block sizes. Observing Figure 5.5a, when the block size decreases, obviously, the upload time decreases too. Instead, Figure 5.5b shows that increasing the number of chunks for the same file size, the upload time increases. This trend is because Hadoop works better with a few of big files than a big number of small files. This is true both for writing and reading operation into HDFS [31], [32], [33], [34]. The same kind of problem is encountered in the data transfer phase from Hadoop to the local file system. Figure 5.6b shows that, keeping constant the file size at $512MB$, as the number of chunks to read increases the download time from HDFS grows considerably.

Figure 5.6a highlights that when the file size of the considered block decreases, the time to read from HDFS and to write it into the local FS decreases too. However the obtained gain is smallest and almost negligible when the considered blocks are too small. Thus, in order to not lose the advantages given by the horizontal federation,it is needed to avoid to use many chunks. Considering $512MB$ size it has been taken as upper bound the value of $5$ chunks; because going beyond this value a degradation of the upload/download Hadoop times is triggered.

At the phase $t_{t6} - t_{t5}$, each federated domain performs the video transcoding. The work of Kim et al. [35] demonstrated that from the computational point of view, to have one domain with $4$ available pro-

cessing nodes is the same that to have 4 domains with only one available processing node. What has been said can be seen by looking at the table 5.1 which has been extrapolated from a more complete and complex table of the above cited paper [35].

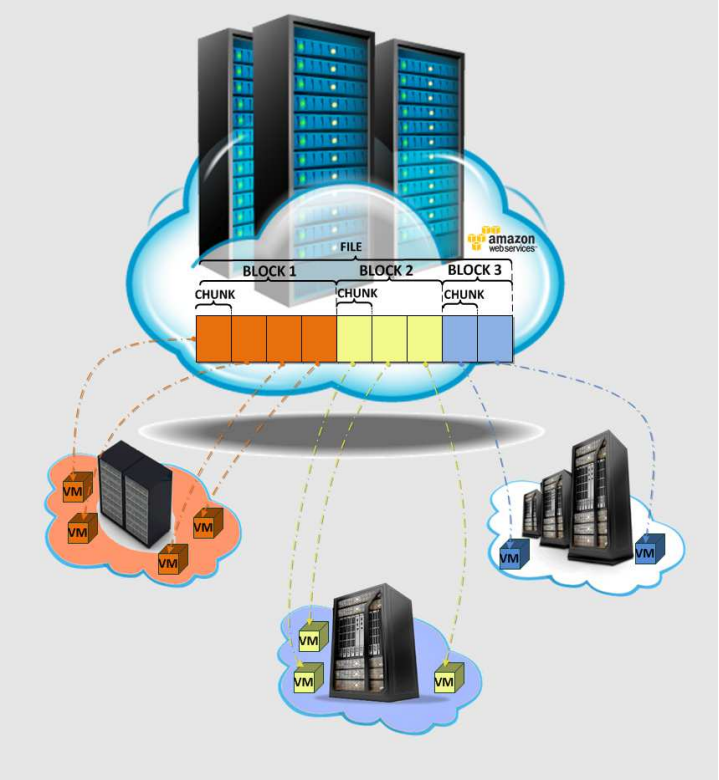
| Nodes | 0,5GB | 1GB | 2GB | 4GB | 8GB | 16GB |
|---|---|---|---|---|---|---|
| **1** | 486 | 916 | 1792 | 3465 | 6934 | 13816 |
| **2** | 270 | 516 | 930 | 1810 | 3480 | 6980 |
| **4** | 167 | 278 | 516 | 950 | 1810 | 3595 |
| **8** | 127 | 183 | 276 | 512 | 967 | 1858 |
| **16** | 140 | 166 | 188 | 282 | 517 | 980 |

**Table 5.1:** *Transcoding Time (Seconds).*

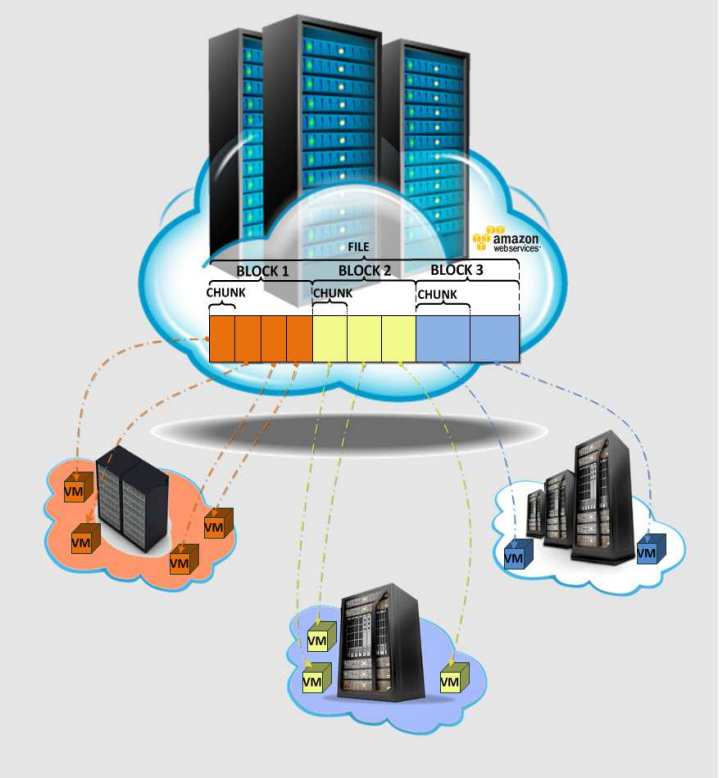
Observing Table 5.1, how just said, from the computational time point of view, to transcode, e.g., a $8GB$ file size by means of one domain with eight processing nodes (8 VMs) is the same that to transcode a $8GB$ file size by means 8 domains with only one available processing node (1GB per domain). However, it is very important to emphasize that as just said is true only from the computational point of view, in fact considering the whole federated transcoding process this is not true anymore. In fact, in both cases the computational time is the same but in the first one the download from S3, the writing into HDFS and the upload to S3 times considerably increase. The above is summarized in Figure 5.7a. It is clear that while the discovery time and that one needed to forward the requests to the federated domains are small and negligible, the download, upload, Hadoop and transcoding time by means OpenCV tools represent the most important impact for the examined scenario. This work demonstrated that, by means the horizontal Federation among several domains, it is possible to significantly reduce the above mentioned times. In other words, the federated parallelization has improved the transcoding process adding an additional parallelism level without which it could be only possible to parallelize the transcoding phase. Observing Figure 5.7b it is possible to see that, considering a $512MB$ file size, increasing the number of the

domains, from one to eight, a strong reduction of the total time is obtained. Specifically, the first column shows the whole process time considering a single domain with eight available processing nodes. The other columns show respectively the process time when the Federation is made by two domains with four available computing nodes, four domains with two available computing nodes and eight domains with only one available computing node. It is possible to see that the discovery time and that one necessary to forward the request to the other foreign domains are negligible, in fact they are not visible in the picture due their very little dimension. Moreover, while the transcoding time remains the same (in the Figure the area with horizontal lines), the time of the other phases visibly decreases. It is important to say that this is only the worst case. In fact the number of the processing nodes decreases in an inversely proportional manner to the growth of the number of the domains. Thus, it is clear that the horizontal parallelization is better than the vertical one provided by a single Hadoop-based domain. Moreover, the first choice should be to parallelize horizontally as much as possible and than to proceed whit vertical parallelization, choosing a number of nodes per domain equal to that one of the domain that has the lowest number of available nodes. This would be useful to maximize the inter-domain parallelization and to equally distribute the overhead, which is introduced by the different process phases, among the involved federated domains.

Finally, at the phase $Ends - t_{t7}$ the broker sends a notification to the SN CLEVER agent, that will communicate to the SN that the video is available for visualization. Regarding the three steps of the distribution phase, a part from the step $t_{d0} - t_{d1}$ that is outside of the Federation scope, the steps $t_{d1} - t_{d2}$ and $t_{d2} - t_{d3}$ can be discussed similarly to the $t_{t4} - t_{t3}$ and $t_{t5} - t_{t4}$ of the transcoding federated process.

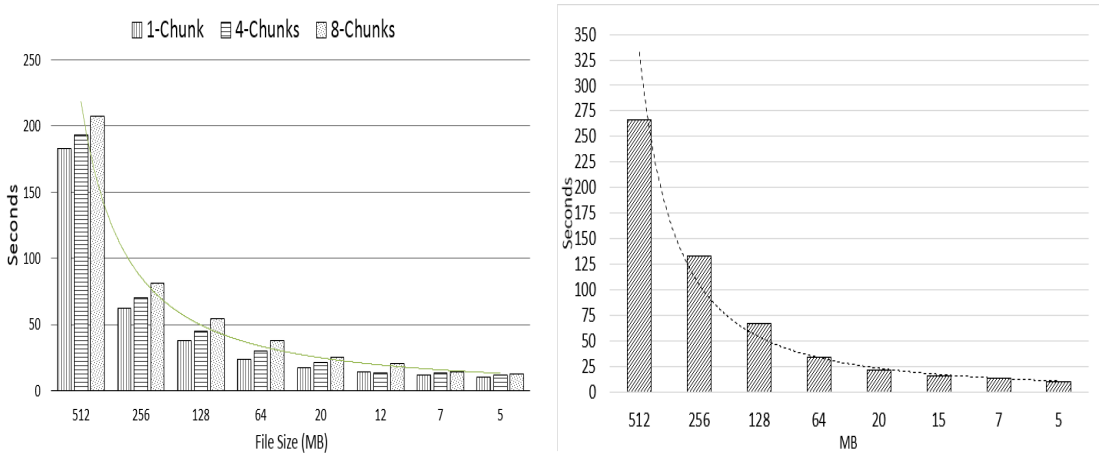



**(a)** *Processing-Balancing-based Splitting*




**(b)** *Download-Balancing-based Splitting*

**Figure 5.2:** *Distribution of the file chunks among the federated Clouds*




**(a)** *Average download time of file blocks from Amazon S3.*

**(b)** *Average Upload time of file blocks into Amazon S3.*

**Figure 5.3:** *Average S3 download and upload Time.*

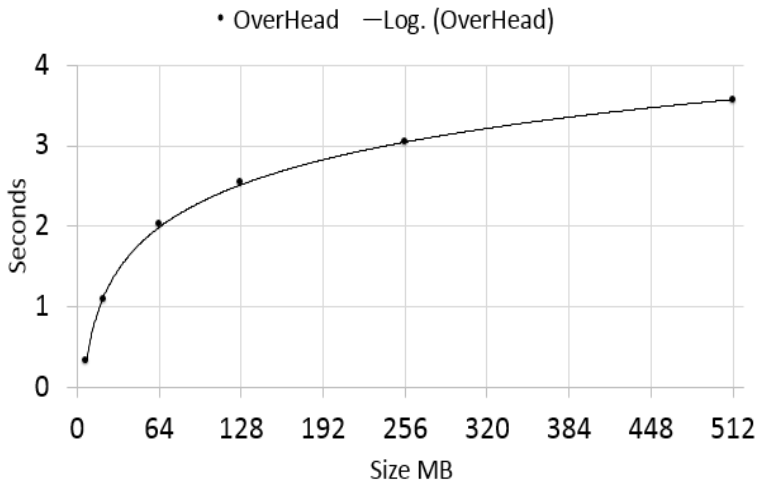
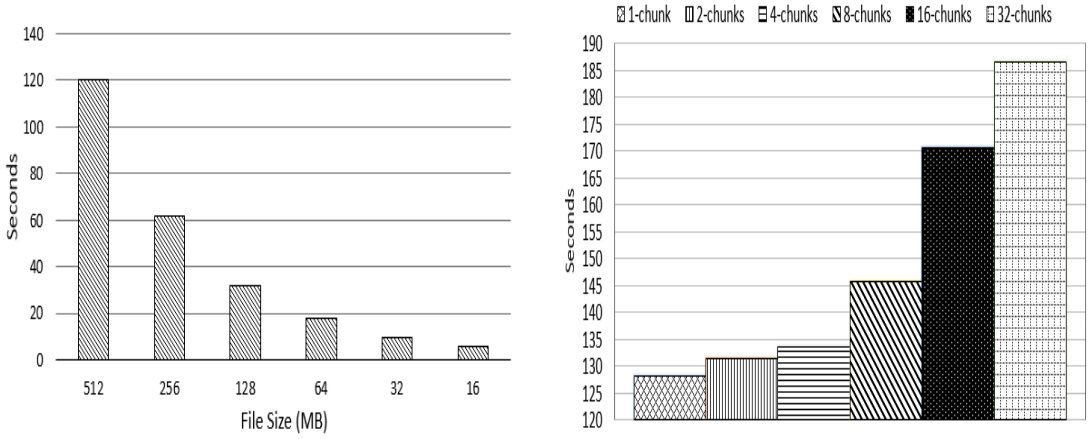
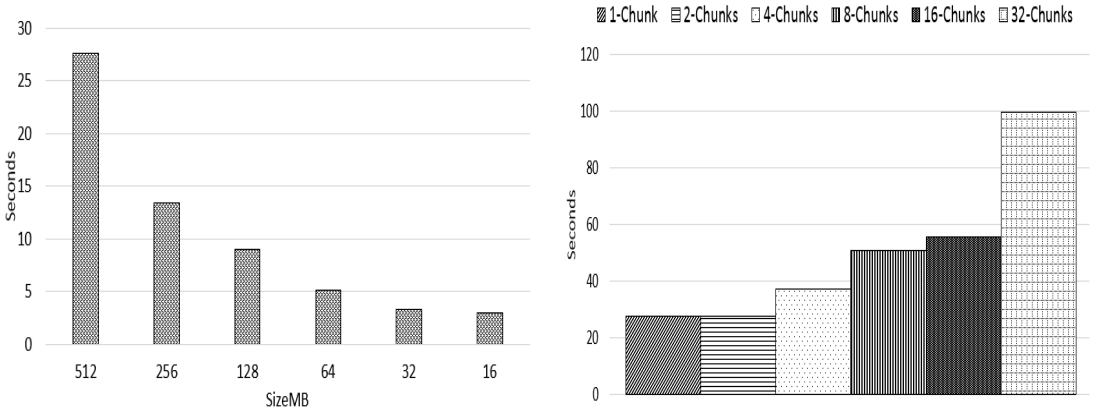



**Figure 5.4:** *Download overhead trend varying the file size*



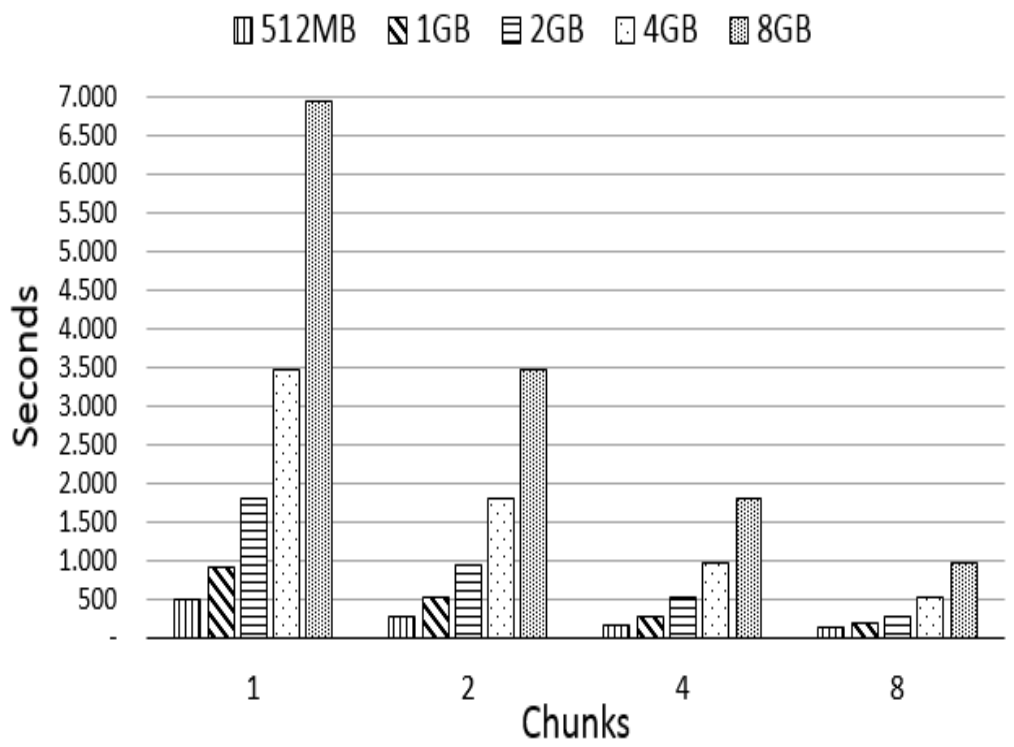

**(a)** *Average Upload time of file blocks in Hadoop.* **(b)** *Average Upload time of 512 file size changing the number of chunks in Hadoop.*

**Figure 5.5:** *Upload time of file blocks to HDFS.*

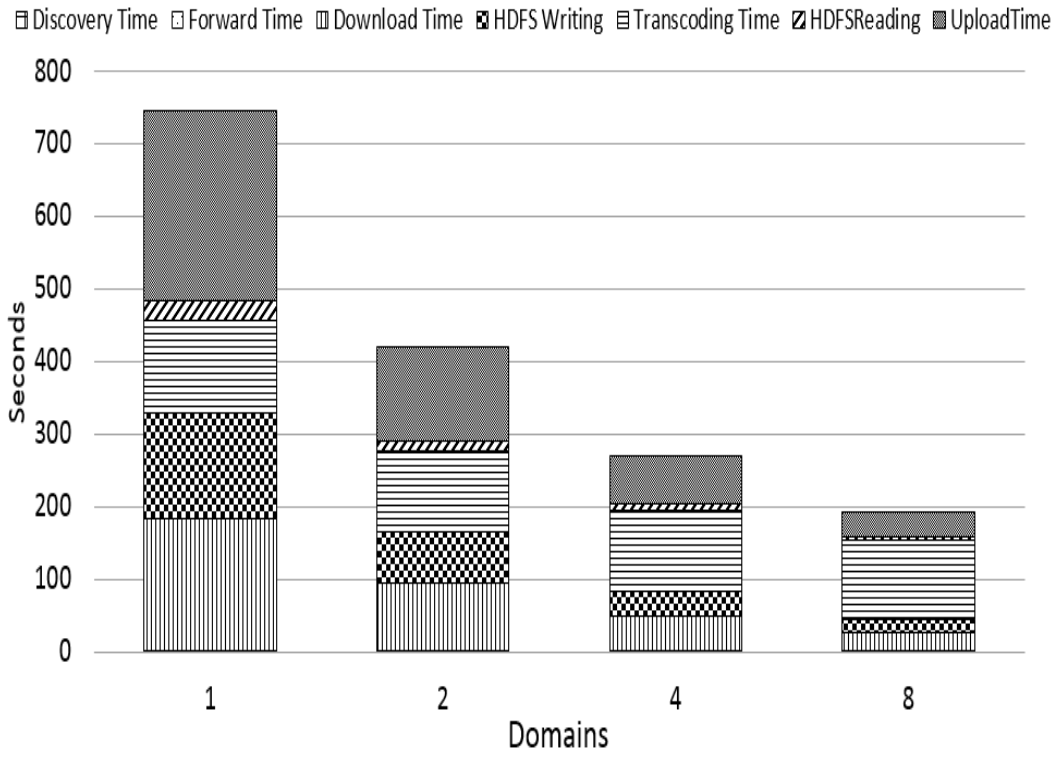



**(a)** *Average Download time of file blocks from Hadoop.* **(b)** *Average Download time of 512 file size changing the number of chunks from Hadoop.*

**Figure 5.6:** *Download time of file blocks from HDFS.*




**(a)** *Transcoding times varying the file size and the number of chunks.*




**(b)** *Total process time varying the domains and chunks number.*

**Figure 5.7:** *Transcoding time and Total process time.*

CHAPTER 6

# Analysis of Algorithms for Optimizing Selection of Cloud Providers in a Cloud Federation

STILL today the challenges to build a federated cloud environment are countless. Academics of the IT sector are constantly looking for new solutions, technologies and protocols to achieve this goal.

The "Federation" topic is considered crucial in the context of cloud computing, in fact the European Union has been funding a considerable amount of projects concerning the Federation or Multi-Cloud subjects. The following Section 6.1, therefore, highlights the main ended/ongoing European Union (EU) projects and other published works about these topics.

## 6.1 EU Projects Overview

RESERVOIR: *Resources and Services Virtualization without Barriers* is a European Commission Seventh Framework Programme (FP7/2006-2013) funded Integrated Project. (November 2007 - January 2011) [1]. (See also Section 2.3). To get a complete view of this project is important to say that, RESERVOIR initially aimed to extend, combine and integrate three different technologies: virtualization, grid computing and business service management (BSM). This research strategy has been very successful in high performance scientific computing. *RESERVOIR* added "virtualization-aware" to the grid computing by moving the focus from job scheduling (typical in grid-computing) towards the creation and placement of generic and virtual computing resources. RESERVOIR primarily focuses its attention to the construction of an infrastructure in which virtual machines are dynamically relocated at any node, regardless of location, network configurations and administrative domains. *RESERVOIR* has got the challenge ability to place virtual machines in an effective manner and cost-efficient finding the best mapping or placement of virtual machines to physical machines in the Federation.

***VISION Cloud***: *Virtualized Storage Services Foundation for the Future Internet* [36] is a European Commission Seventh Framework Programme (FP7/2006-2013) funded Integrated Project. (October 2010 - December 2013). The goal of VISION Cloud is to design a novel scalable and flexible storage cloud architecture able to face and satisfy, thanks to a high level of scalability and flexibility, a large number of simultaneous users by enabling the delivery of different types of storage services. The idea of VISION Cloud was to build a cloud storage environment on the top of an infrastructure made up of multiple potentially worldwide distributed data center. Each data center has multiple storage clusters containing physical resources that have computational, storage and network capabilities. Several works exist in literature. They are linked to

VISION CLOUD project. For example Bruneo et al. in [37] provided an analytic model for the availability investigation of a storage cluster in the context of the storage cloud environment proposed by the VISION Cloud project, and Kolodner et al. in [38] presented two real application scenarios from the healthcare and media domains to demonstrate the validity of their presented architecture which addresses the challenge of providing data-intensive storage cloud services.

***CONTRAIL*** [2] provided federation of all types of Clouds. Specifically it introduced a new Platform-as-a-Service layer allowing an easy management and deployment of applications and data storage [39]. The CONTRAIL project presents many relationships with RESERVOIR both dealing with Cloud Federation. In particular, while CONTRAIL mainly deals with identity management, RESERVOIR focuses on the technological issues related to migration among federated Clouds.

***mOSAIC*** [40] was a project devoted to the Cloud-based application developers, maintainers and users allowing them the specifying of the service requirements in terms of Cloud ontology. This is useful to find best-fitting Cloud services to users actual needs and efficiently outsource computation. ***mOASIC*** presented a multi-agent brokering mechanism-based framework able to search for services matching the application requests, and possibly to make a service composition if no direct hit is found.

***BonFIRE*** [41] was a project aimed to design, implement and manage a multi-cloud environment supporting applications, services and systems research in the field of the Future Internet (FI). ***BonFIRE*** targets the Internet of Services community and offers a test infrastructure that is ideal for performing experiments relating to distributed applications and services [42]. ***BonFIRE*** adopted a federated multiple platform approach providing interconnection and interoperation between novel services and networking testbeds. Bonfire currently counts 7 geographically distributed testbeds across Europe, offering heterogeneous cloud resources, such as compute, storage and networking resources. The ***BonFIRE*** is a broker-

based Cloud federation model where a broker component interacts among the user requests, the experimenters, and the different infrastructures. In order to expose all the Clouds and network features as resources to the user, it provides a common OCCI-based interface [43]. The ***BonFIRE*** and ***mOSAIC*** project have some features in common, for example the service discovery capabilities.

***StratusLab*** [44] was a project aimed to provide to the system administrators and to the resource providers the functionalities enabling the efficient exploitation of computing resources. It is also able to use external resources, following a hybrid architecture model, hence transforming the local Cloud in a Hybrid infrastructure.

***CloudWave***: Agile Service Engineering for the Future Internet [45] is a European Commission Seventh Framework Programme (FP7/2006-2013) funded Integrated Project started at the beginning of November 2013 and still in progress. The overall aim of the *CloudWave* project is to provide a powerful foundation for the development, deployment and management of a new generation of Cloud-aware services. Cloud services are hosted and deployed by Cloud providers in Cloud operation centers. In order to improve service quality and to optimize resource utilisation, the *CloudWave* approach dynamically adapts cloud services to their environment. Nowadays there are situations where it is not possible to improve application performance by simply adding extra hardware, so there is the need to change the program and its logic to adapt the applications to the new situation. *CloudWave* is able to automatically select the best adaptation method such as adding more resources to the cloud application or migrating application components [46] .

***OPTIMIS*** project [47] has been aimed to allow organizations the automatic outsourcing services and applications to trustworthy Cloud providers in a hybrid-based model. ***OPTIMIS*** supported and facilitated an ecosystem of providers and consumers that take advantage from the optimal operation of services and infrastructures thanks to the optimization of the the whole service life cycle, including construction, deployment,

and operation services [48].

***4CaaSt*** [49] was a project aimed in creating advanced *PaaS* cloud platforms supporting the optimized and elastic hosting of Internet-scale, multi-tier applications and incorporating all the necessary features in order to ease program rich applications and to enable the creation of a real business ecosystem where applications from different vendors can be customized, combined and exchanged together.

The France ***CompatibleOne*** project [50] consists in an open source cloud services broker able to supply, deploy and manage any type of cloud services provided by heterogeneous providers. ***CompatibleOne*** is therefore a key element for the federation and interoperability of cloud systems. ***CompatibleOne***, indeed, has got the capabilities of federating heterogeneous resources and integrating different cloud services of different cloud services providers. The ***CompatibleOne*** provides an own object-based description model of Cloud resources called CORDS (CompatibleOne Resource Description System). The *CORDS* is OCCI-based model. This brokering platform offers services to provide Cloud resources from different IaaS and PaaS providers selecting them according to Service Level Agreement (*SLA*). The Broker processes the provided *SLA* and the user requirements to create an instance of service. The broker also performs a selection process of the most appropriate providers for the provisioning of the described resources.

EU-FP7-ICT-610531 ***SeaClouds*** project [51] aims to provide an adaptive multi-cloud management of complex service-based applications by developing Cloud Service Orchestrators and a set of tools able to manage them. *SeaClouds* wants to provide and support skills such as service orchestration, adaptation and verification in a multi-cloud context, providing a unified Cloud-independent procedure for managing services distributed across several Cloud Providers [52]. To achieve these challenges *SeaClouds* is aligned with the most important interoperability standards such as OASIS CAMP and TOSCA.

This Chapter will provide a review of the current literature on the

challenges and approaches to optimally detect and select the cloud service providers in a Federated or Multi-Cloud environment, with a special focus on how these aspects have been handled in the last years.

### 6.1.1 Other related works

THIS section provides other works facing different type of analysis on the selection topic. With the work Sun et al. [53] the authors deeply analysed the state of the art regarding the cloud service selection topic considering well seven points of view in building the survey namely *context*, *goal*, *data representation models*, *selection techniques*, *selection parameters*, *methods for quantifying qualitative parameters* and *criteria for weighting methods*. Each of the examined selection methods have been put in a specific group taking into consideration the way (decision-making-based, optimization-based, and logic-based approaches) to accomplish that goal. In [54] the authors presented a way to create a federation among several cloud providers by means of the mediation operations of a Broker. In order to explain their federation concept, they used a **WRF** use case application (*Weather Research and Forecasting service*). They illustrated a layered cloud service model (SaaS,PaaS, IaaS) making a comparison between the delegation and federation concepts. Moreover the authors argue the main goals of the Broker for each layer (SaaS, PaaS or IaaS). However the considered work does not focus on the provider selection topic but on the importance in the decoupling of the cloud layers (*aaS) in order to distribute the application execution over different providers. Buyya et al. in [55] present "*InterCloud*" which is a federated environment to support the scaling of applications across multiple cloud providers. They identified three main kinds of actors for the presented *InterCloud* : *Cloud Coordinators*, *Brokers* and an *Exchange*. The first two manage a specific Cloud enterprise and its membership to the federation. The *Broker* acts on behalf of users and it looks for available Cloud services by means of the "*Exchange*" and negotiates with the Coordinators for the resources allocation. The "*Exchange*" plays the

role of information registry storing all the necessary federated Cloud information, and providing a match-making service to map the users requirements to suitable service providers. In the [56] and [57] the authors made a wide analysis of the current state of the art regarding the needs to build a Multi-Clouds environment. In [57] they focused on the Interoperability and Portability challenges and their requirements, classifying them in several solution groups. In [56] the author divided the Multi-Cloud requirements in three different groups namely: "Development, Deployment and Execution". One of these requirements is the ability in the selection of the Clouds services and resources. As consequence, methodologies to compare cloud services in a multi-cloud scenario are needed. But these works are only surveys and they do not give us a concrete way to accomplish goals such as a best selection of cloud service providers in a federated or multi-clouds scenario. In [4], an analysis of the requirements for the establishment of an IaaS Cloud Federation is proposed. The subject of this work has been just discussed in Chapter 3. Subramanian et al. in [58] proposed a wide study concerning the resource provisioning in a federated context by analysing several deployment architectures present in literature. The authors proposed their own vision of the different "Federated architectures" that have been categorized in "*Cloud Bursting Architecture*", "*Cloud Broker Architecture*", "*Aggregated Cloud Architecture*" and "*Multi-site Cloud Architecture*" analysing their specific features.

## 6.2 An Algorithm Classification for the Cloud Provider Selection

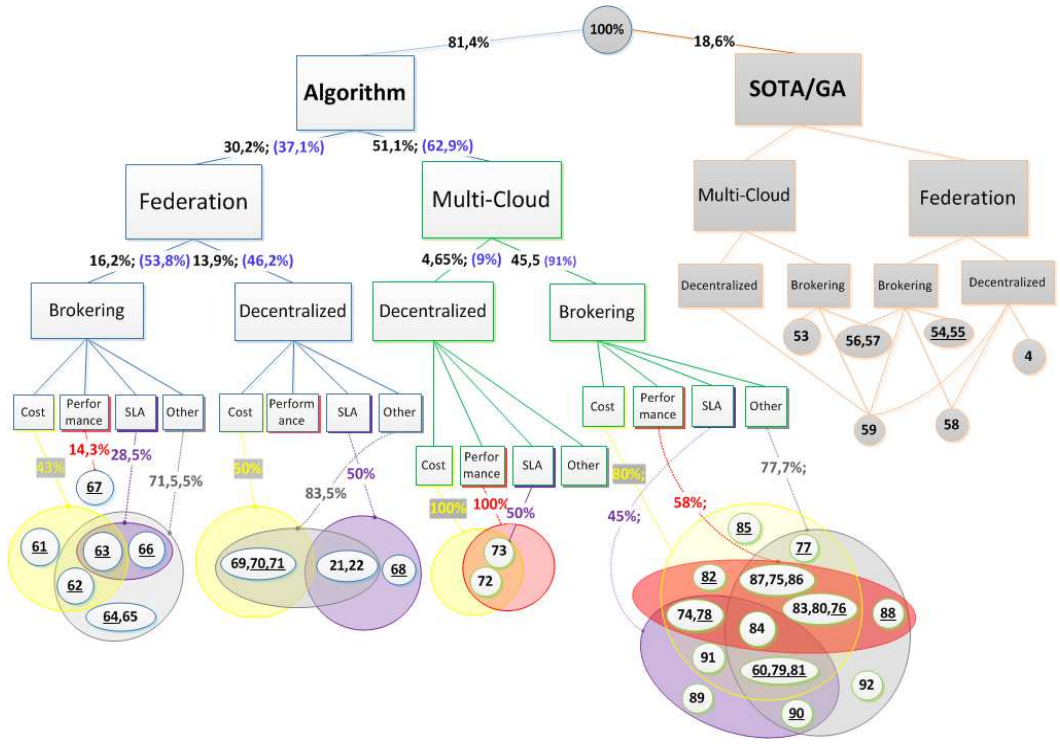



**Figure 6.1:** *Taxonomy Tree*

THE Figure 6.1 shows a visual organisation of the analysed works. They are organized in a taxonomy tree structure. The nodes of the taxonomy tree are the key words that have been selected to characterize the found papers. The leaves of the tree are exactly the reviewed papers. The tree root represents obviously the research topic "*Best Selection of Service Cloud Providers in a Federation or Multi-Cloud environment*". The considered tree has six levels (root level 1, leaves level 6). The first two nodes are "**Algorithm**" and "**SOTA/GA**". They make a distinction between the papers presenting a Selection Algorithm and those presenting only a verbal analysis of such a topic (level 2). Under the sub-tree with root the node "**SOTA/GA**" there are only three levels as they are sufficient to categorise the papers

presenting a SOTA or a Generic analysis. The works presented under this node have been just discussed in section 6.1.1. The level three of the main tree counts two nodes namely "*Federation*" and "*Multi-Cloud*". In fact, it has been decided to separate the papers talking about the best selection topic in a "Federated environment" from those talking about the "Multi-Cloud one". This is an important distinction because even if Federation and the Multi-Cloud seem to be the same thing, they are driven by different needs [59]. These two nodes can be seen as the roots of two specular sub-trees. From now on the focus falls on the left sub-tree with root "*Algorithm*" and only one of its sub-trees will be presented, considering the other one having the same structure. At the level four there are the nodes "*Brokering*" and "*Decentralized*". They grouped the works proposing "*Brokering*-based" or "*Decentralized*" solutions in best provider selection. Finally the algorithms have been grouped in $4$ sets depending on the parameters taken into account during the execution. A detailed explanation of these sub-set parameters is given in Table 6.1. The nodes of the level five are "**Cost**": containing the algorithms considering non-functional parameters like cost or pricing; "**Performance**": algorithm aiming to a performance optimization such as execution time or communication time; "**SLA**": (Availability, Reliability, Usability, etc.); and "**Other**": parameters not included in the previous non-functional parameters (for example placement constraints or reallocation constraints or number of available services, Collision or Collusion [60], or functional parameters like virtual CPU number, amount of Memory and so on) (see Table 6.1 for additional details). The last level is composed by the leaves, each of them is an analysed paper. The analysis in "Best-selection" topic is based on $43$ reviewed papers organized following the above presented structure. The discovered papers, presenting various kinds of analysis (SOTA or Requirments) of such a topic, represent approximately $18,6\%$ (the right sub-tree) of all the papers. The remaining $81,4\%$ (the left sub-tree) of the papers provides instead different algorithmic solutions at the problem. The attention exactly falls on this left branch. The Figure

6.1 also shows the percentage at each level of the tree. In particular it is possible to observe that the $30,2\%$ of the papers presents algorithms operating in a *Federation* context and about $51,1\%$ operating in a Multi-Cloud environment. In order to have a better readability, the percentage values in blue-brackets referring to the papers under a node which represents the root of a new sub-tree have been added. These percentage values tell us that in literature the "Best Selection of Service Cloud Providers" is slightly more debated in Multi-Cloud environments than in Federation ones $(37,1\% vs 62,9\%)$. While under the "Federation" node the papers are almost equally distributed between "*Brokering*" and "*Decentralized*" solutions $(53,8\% vs 46,2\%)$ , the Multi-Cloud sub-tree is strongly unbalanced in favour of the "Brokering" $(9\% vs 91\%)$. An overall vision of the paper distribution is given in Figure 6.2. Specifically, in order to better understand the percentage results of the research, four main key sets have been extracted from the "Taxonomy Tree" and the papers have been placed inside them according with the percentage belonging values showed in Figure 6.1. However, these results are logical as typically in a *Multi-Cloud* environment a "Cloud Provider" does not have awareness of the others "Cloud Providers", so there is the need of a central aware entity able to work with different administration domains. After these first considerations our analysis will proceed presenting each of the papers following a "post order" like visiting.

### 6.2.1 Brokering Solutions in a Federated Context

THE papers grouped under this sub-section are those providing algorithmic optimization solutions in the selection task in a *Federated* environment.

#### 6.2.1.1 Cost

THE first three following papers have the same characteristic to provide solution trying yo optimize the service provider selection from the profit/cost point of view. The first work focuses on the selection during the phase of establishment of the federation instead the other two papers of the same sub-set proposed solutions for the provider selection in a just established federated environment. Specifically Mashayekhy et al. [61]

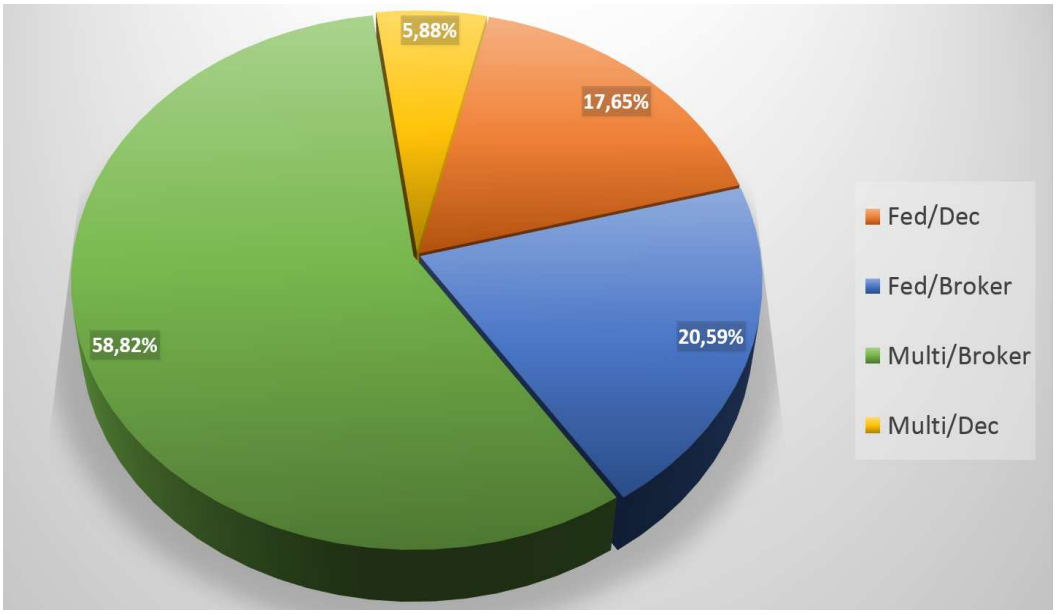



**Figure 6.2:** *Pie Chart Concerning the Paper Organization*

proposed a mathematical algorithm to improve the automatic scaling capabilities of a cloud provider in fulfilling the users demands. They proposed a "*Cloud Federation Formation Games*" that considers the profit a cloud obtains taking part to the federation. The algorithm is based on the idea

that a cloud will want to participate to the federation, if the profit that it will have is higher than that one it would have staying alone. The process is run by the *Broker* having all the information about the Cloud Providers such as their available resource, cost etc. The proposed algorithm focuses on the selection of Cloud Providers to create a Federation able to meet the users requests maximizing the federation profit. So this work together with Gahlawatet al. [62] and Abdo et al. [63] are grouped in a sub-set of works that aim to optimized costs and profits. In fact Gahlawatet al. [62] proposed a framework able to improve the federated VMs selection process by means of a Divided KD tree data structure in a market oriented cloud computing. The selection process is based on a matching mechanism between the offered VMs and the VM requests. The research looks like to a binary search tree. When an offer arrives, the VM is added to the tree. When a request arrives, a matching mechanism stars to find the similar VM into the tree. If any of the virtual machines matches the criteria, that VM is removed from the tree. The authors do not provide any implementation detail for the selection algorithm. The proposed approach seems to be a selection of VMs and so it can be considered a selection at the *IaaS* layer. This work besides takes into consideration functional constrains like number of CPUs, amount of memory and so on. For this reason it is present in another set (**Other**) together to Badidi et al. [64], Abdo et al. [63], and Caballer et al. [65]. Abdo et al. [63] proposed a *rearrangement* of the ***CCFM*** process presented by Celesti et al. In the authors opinion the fully distributed approach followed by Celesti et al. ( [21] [22]) in performing the discovery and matchmaking processes is valid but it presents several lacks in terms of communication delay and overhead and reputation mechanism. The authors solution rearranges the ***CCFM*** in a *Broker-based CCFM*. They argue how the *Brokering* in ***CCFM*** can improve the *Cross Cloud Federation Manager*. In this novel scenario, as the *Broker* periodically updates a centralized file containing the Clouds status, a home cloud, only with a message pair, can obtain all the Clouds offers information. Moreover the algorithms of the matchmak-

**Table 6.1:** *Key-Parameters of the considered grouping sets*

| SLA | COST | PERFORMANCES | OTHER |
|---|---|---|---|
| QoS, Availability,Reliability, Usability,Resiliency/fault tolerance, Interoperability, Recoverability, Security/Confidentiality, Agility, Durability, SLO (service level objective) | Financial charges, Communication costs, VM Price, Storage Cost, Down/Upload, Electric Power, Computing Power | Execution Time, Response Time, Throughput, Network Transfer, Service Response Time, Latency | Functional (CPUs, Memory), Number Of Available Service, Service Type, Collision/-Collusion, Number of Available VMs, Computational Capacity, Trust/Reputation, Lock-in, Co2 Reduction |

ing (selection of the best Cloud Providers) should be run by the *Broker*. In other words the authors move the intelligence from the "Cloud" to the "*Broker*" so reducing the messages between the "Home Cloud and the Foreign Cloud". The authors present the high level algorithms concerning the discovery and matchmaking processes. The algorithm takes into consideration Functional (amount of resources) and Non-Functional parameter (Cost, SLA and QoS, Trust Relationship) so seeing the Figure 6.1 it is present under three nodes : **Cost**, **SLA** and **Other**.

#### 6.2.1.2 SLA

In a federated environment the cloud providers need to have guarantees about the sharing of their own resources by the stipulation of a priori agreements among the interested parts. The papers under this tree node aim to perform a cloud provider selection considering the parameters inside the SLAs throughout the execution of the algorithm. Nine et al. [66] shares the "**SLA**" subset with the previous cited work Abdo et al. [63]. It proposes a *Broker*-based approach to optimize the outsourcing the user

requests. Especially the *Broker*, taking into consideration the available resources of the Cloud Providers, has to decide whether to forward the request toward another cloud provider of the federation or to assign that request to the provider receiving that service request first. The proposed approach considers both functional and non-functional requirements and tries to find the best way to fulfil the users requests without violating the *SLA*s constraints.

#### 6.2.1.3 Other

BADIDI [64] and Caballer et al. [65] presented two different brokering frameworks. The first is a brokering solution driven by a mathematical optimization function, whereas the second is a resource description language based matching solution between the users requirements and the resource constraints. In detail [64] describes a framework for (Quality of Context) QoC-driven selection of context services. They specifically proposed a "Federation" of cloud-based context *Brokers*. The main actors are: *Context Consumers* (CCs) which consume the context information, *Context Brokers* (CBs) allow several clouds to cooperate with each other and *Context Providers* (CPs) deploy their context services offering several types of context information. Each *Context Provider* monitors, collects and elaborates sensors data to determine its **QoC** value. The authors describe the mathematical selection algorithm based on a probability of correctness function $U_P(p)$ and the freshness function $U_F(f)$ which are the only two considered quality attributes for the tested algorithm. These two functions put in relation the offer of the *Context Service* ($p$ and $f$) with the required value of the *Context Consumer* ($\alpha_P and \beta_f$). Each of the *Broker*s executes the selection algorithm to find out the best *Context Service* which maximizes the global utility function $U = w_1U_1+w_2U_2+...+w_nU_n$. In Caballer et al. [65] instead, a framework for the easy and automatic deployment, selection, configuration and monitoring of VMIs is presented. The authors show the main components of their system. The architecture

of the framework counts different components: (*i*) RADL "Resource and Application Description Language" which is a language describing the requirements of the virtual infrastructure. (*ii*) A repository "**VMRC** - Virtual Machine Image Repository and Catalog" used to find the available VMIs taking in consideration the users requirements. (*iii*) The IM - "Infrastructure Manager" is the central part of the system and has three main components the "*Cloud Selector*" , the "*Cloud Connector*" and the "*Configuration Manager*". The selection process is performed by the "*Cloud Selector*". It selects the best combination of VMIs and Cloud Providers by querying the **VMRC** to pick out the VMIs that better meet the users requirements. After that it retrieves the list of the available Cloud Providers and selects those compatible with the selected VMIs. The last two previous papers talk about selection approaches that consider "functional and non-functional parameters. They are placed under the "Other" node. Whereas Badidi et al. [64] present a mathematical algorithm Caballer et al. [65] focus on the *RADL* language for the requirement descriptions without providing details of the selection process.

#### 6.2.1.4 Performance

FINALLY Liu et al. [67] is the only one that takes into consideration performance requirements. It illustrates a mathematical mechanism to allocate virtual resources in a federated cloud environment in an efficient way. The *Federation Portal* through a monitoring and profiling system collects information about both the status of the federated resources and the application behaviour. The algorithm aims to allocate the resources in the federation reducing the *communication time* and the system *throughput* due the VMs repacking to obtain better performances. The authors approach is based on the estimated communication time ***ECommT***, estimated computation time ***EcompT***, the $CCR = \frac{ECommT}{ECompT}$ and on the Estimated execution time $EET = ECommT + ECompT$ of each cloud. When a job arrives the *Federation Portal* calculates the average value of

$CCR$ of the first $J$ jobs in the waiting queue of each cloud and considers this value as $CCRthreshold$. After, for each job the system checks its $CCR$ value and depending on whether it exceeds or not the threshold, it decides if to allocate resources in one of the federated cloud minimizing the $EET$. The algorithm can aggregate resources from some participated clouds by selecting a sub-set of clouds whose resources sum is equal to the job requested ones.

### 6.2.2 Decentralized approaches in a Federated Context

THE following works tackle the best provider selection in a decentralized way. These works constitute the $46,2\%$ of the papers focusing on the federation context and are organized under 3 sub-sets: *Cost, SLA and Other*. All the papers in this sub-section consider "Other" requirements together with "Cost" or "SLA" constraints except Kertesz et al. [68] considering only the "SLA" one. Thus we can simplify the reading by separating the papers tackling the cost optimization from those ones aiming to optimize the SLA parameters.

#### 6.2.2.1 Cost

GIACOBBE et al. [69] surely falls in the first set. The authors in fact present a selection algorithm that makes able the federated Cloud Providers to determine and so to choose the best destination, from the reduction carbon dioxide point of view, where to migrate their VMs. The algorithm counts two steps, the first one creates a "destination/granularity matrix" for "energy cost-evaluation" in terms of carbon dioxide emissions-per-kWh, the second one instead finds the "optimum migration path for carbon dioxide emission reduction". During the first step of the algorithm a monitoring of each site for the forecast period is performed. It evaluates energy consumption at the $i-th$ site for a specific workload that has to be migrated. If the monitored $i-th$ host site has sufficient capabilities to manage the considered workload then it is added to the destination

matrix. The second algorithm step has in input the destination matrix and extracts from it that destination with the minimum CO2 emission for the considered forecast period. Moreover it gives as output a matrix with the best migration path. Even Yeh et al. [70] is placed in the same set and presented a dynamic way to allocate resources across multiple clouds in a federated environment. The basic considerations made in this work are similar to those discussed in Celesti et al. [22] (placed in the set). However the Federation here is created following the concept of *Cross-IdP*. We can see this concept as a ***chain of trust***. It works like the syllogism concept: $if\ A \xrightarrow{trusts} B\ \ \&\ \ B \xrightarrow{Trusts} C\ Then\ A \xrightarrow{Trusts} C$. So when a cloud provider needs external resources, it asks for them to the *IdP* where it has a trusted relationship. Firstly a local cloud asks for resources to those external clouds having a trusted relationship with the same *IdP* (first relationship layer), after, if the provided resources in this first layer are not sufficient to fulfil the user request, it virtually considers the external clouds in the first relationship layer as local clouds and then forwards the request to the clouds having a direct trusted relationship with them and so on. The algorithm finishes when the external rent resources meet the requested ones. The algorithm takes into account three main parameters. (i)*The IdP reliability value* that is incremented of 0,05 whenever that IdPs is used; (ii) *Resource information* of each cloud provider (idle resources and requested resources); (iii) *The network transmission cost* between two clouds. The first step is to choose the *IdPs* with the major reliability value, the second step, in order to select the external cloud, the ration $\frac{IdleResource}{NetworkCost}$ with highest value is chosen. After that if the request is fulfilled the algorithm finishes otherwise the research continues firstly in the same layer and, if the layer is not able to fulfil the request, in the other layers. Rebai et al. [71] proposed a mathematical algorithm that helps the Cloud Providers in a federation to automate and optimise the resources allocation among multiple clouds. The users requests are modelled as *Graph*, where the nodes are the requested VMs and the edges are the traffic flow between neighbouring nodes. The algorithm is executed in

each cloud provider and aims to optimally distribute the requests across the federation in order to maximizing revenues and to minimize the costs for each provider.

#### 6.2.2.2 SLA

THE Celesti et al. [21] , how just said, fall in two sets "**SLA**" and "**Other**". It proposed an approach for the federation establishment considering generic cloud architectures according to a *three-phase model*, representing an architectural solution for federation by means of a Cross-Cloud Federation Manager (CCFM), that is a software component in charge of executing the three main functionalities required for a federation. In particular, the component explicitly manages: (*i*) the discovery phase in which information about other clouds are received and sent, (*ii*) the match-making phase performing the best choice of the provider according to some utility measure and (*iii*) the authentication phase creating a trusted context for the federated clouds. In particular the authors give a deep mathematical explanation of the matchmaking algorithm to select the cloud provider that better meets the cloud consumer requirements. A high level description of the *XACML*-based algorithm is provided in Celesti et al. [22]. The match making selection is based on two different evaluation tasks. The first one takes as input the set of the discovered Cloud Providers $S_D$=$A, B, C, D$.... and gives as output a new subset of clouds $S_R$=$A, B, D$ which are those that better satisfy the home cloud requests from the resources availability point of view (CPU,RAM...). The second evaluation step takes as input the set of the discovered Cloud Providers $S_D$=$A, B, C, D$.... and gives as output a new subset of clouds $S_{IdP}$=$A, D$ which are those clouds providers having a trusted relationship with the same IdP where the cloud consumer has a valid identity. After that an intersection of the two output subsets is made to produce a matching subset $S_M$= $S_R \cap S_{IdP}$. The algorithm continues considering several metrics: requested resources $R_{req}$ and offered resources $R_{off}(F_i)$ by the

Cloud Providers $F_i$. The matchmaking agent sorts the $S_M$ subset in a offered resources descending order $F_{ord(S_M)}$ and will consider the first **k** clouds of the $F_{ord(S_M)}$ that satisfy the condition $R_{req} \leq \Sigma_{i=1}^{k} R_{off}(F_i)$ , $1 \leq k \leq n$. Finally Kertesz et al. [68] proposed a solution to manage a federated environment by using a way looking like a distributed brokering system. Each federated cloud has an own broker that manages the internal resources. The proposed architecture is an entry point for the cloud federation. The most important component is the "meta-brokering" (interface) that is able to interconnect several different cloud *Brokers* available in the system. It is able to decide among various *Brokers* taking in consideration the metrics gathered from a service monitoring subsystem (**SALMon**). An important key actor of the system is the *Generic Service Registry* where the services information are stored ( WSDLs, VMIs VAs). The meta-brokering layer receives the users service calls, checks if the service exists in the **GSR**, and selects a suitable *Cloud Broker*. It runs a matchmaking algorithm that combines the **GSR** information, the status information on Broker and **SALMon**. The GMBS (Generic Meta-Broker Service) builds a *Cloud Federation* interconnecting different clouds by means of *Broker* interconnections. Consequently the incoming calls are queued in a specific IaaS system and afterwards they are scheduled toward the available VMs in a queue managed by the *Broker* of that IaaS system. The calls are associated to the Virtual Machines present in that queue. More simply the *Broker*'s task is to manage the arriving service calls and the queue of the VMs queue, associating a call to a specific VM able to meet the call requirements.

### 6.2.3 Decentralized Approaches in a Multi Cloud Context

How just said at the beginning of this section (6.2) the "*Multi-Cloud*" sub-tree is strongly unbalanced. The $91\%$ of the *Multi-Cloud* tree leaves falls under the "Brokering" branch. It is obvious due to the lack of a holistic awareness, on the cloud service providers part, of the surrounding cloud panorama. Usually the Cloud Providers in a

Multi-Cloud environment make available tools, APIs, libraries to make accessible their own resources to other cloud entities. Consequently, often there is the need of a Brokering entity able to manage these multitudes of APIs, and tools provided by these heterogeneous Cloud Providers. Also for the description of the section a "post-order"-like visiting is adopted. With reference to the Figure 6.1 the first two papers and the only ones falling under the "Decentralized" node are : Negru et al. [72] and Kajiura et al. [73]. Both of them share the **Cost** and **Performance** subsets, whereas Kajiura et al. [73] presented a SLA-based approach too. By looking into the details of these two works it is possible to state that Negru et al. [72] presents a mathematical algorithm to optimally select a sub-set of Cloud Storage Service Providers (**CSSP**) to store large amount of data coming from different sources geographically distributed. They hypothesized to have several distributed sources acquiring environmental data. The data of each source is stored in a different **CSSP**. Finally the whole amount of data of all **CSSP** will be sent to a central Cloud Provider to be processed. The authors algorithm is matrix-based and performs an overall best provider selection considering cost and latency constrains. However it is a general algorithm and it is not referred to a specific cloud scenario. Kajiaura et al. [73] instead proposed a mechanism for the dynamical choosing of the optimal cloud service provider combination in a heterogeneous Multi-Cloud environment. The authors assume an environment composed by several Cloud Storage Service Providers **CSSP**. Each of their services has a XML-based SLA file containing the conditions that the service needs to meet. The authors considered four indicators for the users requirements:

- **costs**;
- **confidentiality**;
- **availability**;
- **transfer time**;

and four *SLA* items for the services:

- **costs**;
- **leakage probability**;
- **operating rate**;
- **communication speed**;

The authors found mathematical formulas to create a correspondence between the users requirements and the Clouds services SLA items. The paper presents their selection technique as a user-centric solution but in our opinion the presented algorithm can be generally useful and applicable in several selection contexts.

### 6.2.4 Brokering Solutions in a Multi-Cloud Context

THE same visiting approach used in the previous section 6.2 has been used here too. The leaves of this *Multi-Cloud* sub-tree are about the $91\%$ of the totality of the works analysed for such a topic. The remaining $9\%$ has been explored in the previous sub-section 6.2.3. By analysing the Figure 6.1 it is possible to assert that the totality of the reviewed papers take into consideration "*Cost Optimization*" or "**Other**" various optimization goals. These two big sub-sets include another smaller one namely the "**Performance**". Only the $58\%$ of the papers have as goal a performance optimization. Finally there is the $45\%$ of the solutions that are "*SLA-based*". Some papers presented below, besides the "*Cost-optimization*", have other similar characteristics so that they will be present in other sub-sets. By seeing the Figure 6.1 and having a look at percentage values of the sets placed in the bottom of the pictures, it is possible to observe that the sum of these values exceeds the value of $100\%$ due to the fact that several works share more characteristics like "Cost, Performance ...etc."). Therefore the percentage values at the last level of the tree tell us only how many works fall inside a specific sub-set. Table 6.2 gives a summarizing vision of just said.

#### 6.2.4.1 Cost

THIS sub-set contains the $80\%$ of the papers under the node "**Multi-Cloud**". The first three works present matching-based selection strategies. All of them proposed a Broker-based framework adopting different ways and technologies to perform the selection process. In particular in Sundareswaran et al. [60] an interesting architecture, where a *Cloud Broker* manages an automatic service selection in a multi cloud environment, is presented. Their approach is based on "*Unique Indexing Technique*" (CSP-index). The *Cloud Broker* stores the Cloud Providers information in a *B-tree* data structure. It is a structure similar to the binary tree but in the *B-tree* only the leaf nodes contain information. The other nodes instead work like directories. Moreover each non-leaf node can have more than two child nodes. After the encoding in a B-tree structure of the *CP*s information ($Key_sp, SID, p_1, p_2, .., p_n$), when a user sends his service request in the form $Q = (D_1, D_2...D_n)$ , where $D_i$ are the expected properties of the Cloud Providers, the *Broker* encodes that request in the same way of the *CP*s properties information so obtaining a $Key_q$ (index key of the request). After, the *Broker* performs a searching process browsing the *CSP*-index to find the $k$ candidate service providers whose encoded properties are the $k$ nearest neighbours for that request. The search starts from the root and follows the path with the smallest **hamming distance** to $Key_q$ ( index key of the request). In Jrad et al. [74] the presented *Broker*-based framework aims to deploy application workflows in a multi-Cloud environment. The framework is able to perform two main tasks: an automatic selection of the *Cloud Service Providers* and a workflow data management respecting the user **SLA**s requirements. The brain of the system is a *Cloud Service Broker* that performs several tasks like: Identity management, *Match Making*, SLA, Discovery management and Scheduling. It is the mediator between the client and the *Cloud Service Providers*. The authors show the application workflow deployment flow. Regarding the *Match Making*, the authors used a simple matching policy called

"*Sieving*". It selects only the Cloud that meets all the user requirement and the algorithm will select randomly only one if more Clouds match the users requirements. Jrad et al. [75] gave the details of (i) simple static matching scheme, called the "*Sieving*" algorithm and (ii) the utility-based matching algorithm which takes in consideration both the functional and non-functional parameters (response time, throughput, price etc.). The *Sieving* algorithm performs a one by one comparison between the user requirement parameters and the "*SLA*" metrics ($availability = 100\%$) of the *Cloud Provides* which are the two inputs of the algorithm. The output is a set of selected clouds matching all of the requirement attributes ($price < 0.02.\$/h$). Ngan et al. [76] instead proposed an OWL-S Based Semantic Broker system which provides service discovery and selection capabilities. The system allows semantic discovering and picking out those cloud provider services that meet the cloud consumers requirements. The *Cloud Providers* offer their services to the brokerage system. All of these services are stored in a Semantic Service Repository (SSR) . The user contacts the Broker asking for a service and the *Broker* executes a matching algorithm to pick out the best service composition. The authors provided a wide and deep analysis of the problem and verbal explanation of the algorithm. Calheiros et al. [77] focus on the architecture of the *Cloud Coordinator* component of the Broker-based architecture presented in Buyya et al. [55]. The *Broker* acts on behalf of users in looking for available cloud services by means of an interaction between *Broker* and the *Cloud Coordinator* (*CC*) that has to be present in all of the cloud service providers. In this work the user does not need of the *Broker* support, but he can directly interact with the "*CC*" that will discover and negotiate for the resources. The authors thought the "*CC*" as a data center without available resources. The main functionalities of the *CC* are: to communicate with the other cloud coordinators; to mediate the interaction between *CC*s, to intermediate the access to the local infrastructure; to determine the resources price. All the needed messages are SOAP-based message. Anyway this papers lacks of details regarding the algorithm

used by the *CC* to select the best Cloud Service Provider (*CSP*) offering its *aaS. Another interesting framework is the "*SMICloud*" proposed in Garg et al. [78]. This ***Broker***-based framework lets the comparison between several cloud providers offerings, helping the users in selecting the services according with his requirements. The framework performs first a measurement of the services **SMI** attributes, that dynamically vary over the time and subsequently performs a ranking of these services according to the measured **SMI** attributes. In this way the users can exploit the *SMICLoud*' features during the selection of cloud services. *SMICloud* takes into consideration parameters such as: Performance, Security and Privacy, Usability etc., and it is composed by three main components: a *Broker* which receives the user requests for deployment of an application, and performs a discovering and a ranking of the available services exploiting other its internal components namely a SMICalculator, a Ranking Systems and a SLA Management; the *Monitoring* component performs a discovery task and a monitoring of the service performances; and finally a *Service Catalogue* that collects the cloud providers offering and their features. The authors proposed for each considered parameter its mathematical model. Cloud services are ranked according to the Analytic Hierarchy Process technique(AHP). Anyway the paper focuses more on the "ranking" process than on the "selection" process of the ranked available services and furthermore it does not present any algorithm regarding such a purpose. The following four works focus on the Storage Service Provider Selection and aim to find a trade-off between the storage service cost and the QoS. The first and the fourth ones are strongly mathematical-based solutions. The second is a middle way between a mathematical and match-making approach and finally the third paper to follow instead is closer to a matching solution. In particular, from an overall point of view, the ideas presented in the first, third and fourth papers are similar to each other. Entering into the detail, Mansouri et al. [79] developed three different complex mathematical algorithms for the best placement of object replica chunks among different independent data centers (DCs).

Each object is split in more chunks and they have a number of replicas $r$ usually $r = 2 or r = 3$. These algorithms are run by the *Broker* to help users to find a suitable placement of objects as long as the required *QoS* is fulfilled. The three algorithm are: (i)The *Minimizing cost with given expected availability* that finds a subset of *DC*s that minimize the cost while fulfilling the required *QoS*; (ii) *Maximum expected availability with given Budget* to maximize the availability of the replica chunks without overstep a usable budget; (iii) *Optimal Chunks Placement* that operates taking in consideration the number of data center $n$ and the number of the replicas $r$ to find the optimal chunks placement. The authors reported in this works the pseudo-code of the above algorithms. As they are mathematical-based algorithm, in our opinion, they can find applicability in several different scenarios too. The second work, Esposito et al. [80] conceived a selection strategy which makes use of the subjective preferences of the customers. The authors provided a fuzzy-inference-based mathematical matching algorithm to accomplish the storage service provider selection. The algorithm has to match the *QOS*-based users requests to the crispy *QoS* provided by the cloud storage providers (*SSP*). The algorithm can be run in a centralized or non-centralized manner. In the first case *Broker* will collect the *QoS* levels of the Cloud Providers and it will run the genetic algorithm: Each *SSP* sends periodically this information to the *Broker*. In the non-centralized version, Dempster-Shafer theory-based, the customer requests can be served by passing them directly to the *SSP*s involved in the selection. It is a general algorithm which could be applied on a single cloud, federation or multi cloud scenario. Papaioannou et al. [81] introduced a broker-based architecture solution named *Scalia* able to continuously adapt the placement of the stored data among several Cloud Providers, taking in consideration the users *SLA* requirements and the access pattern to the data. The data is split in $n$ chunks and they are stored among $m$ Cloud Providers with $n = m$. A number of $x - subset$, with $n > x$, of clouds is sufficient to reconstruct a complete copy of the data. *Scalia* has a three layer architecture (i) Engine Layer, (ii) Caching Layer

(iii) Database Layer. Our attention falls on the first layer which is responsible of the best data placement solution. It is composed by more engine components managing independently the selection algorithm. It considers the object access history which is a list of statistics of that data object (used storage, the incoming bandwidth, the outgoing bandwidth as well as the number of operations). The algorithm uses the user *SLA* requirements (e.g. durability, availability etc..) to pick out from all the possible combinations of the available Cloud Providers that one has a price less than the desired price by the user. In other words the authors want to minimize price, while satisfying the minimum availability, durability, and lock-in constraints. Finally Yao et al. [82] showed a Multi-Cloud architecture that by means of a mathematical algorithm is able, starting from a set "$A$" of Cloud Providers, to optimally select a subset "$B$" of the available ones for data placement. The idea is to split the data in more chunks and store each of them in a different cloud provider (***IDA*** = *Information Dispersal Algorithm*). The mathematical algorithm considers different factors for the best selection: (i) *storage cost* (upload and storing cost along the time), (ii) the service availability of a specific service provider; (iii) Network performance; (iv) Vendor lock-in; (v) Algorithm time execution cost. They define an objective function that takes into consideration all of the above factors. The best selection means to minimize the objective function by minimizing (i)(iv)(v) and maximizing (ii)(iii). The authors present the main components of the architecture and the pseudo-code of the mathematical algorithm.

The remaining works of this sub-set tackle the selection topic following mathematical approaches and the working area of their presented frameworks is the IaaS layer of the cloud service stack.

The first Broker-based approach we are going to analysing is Tordsson et. at [83] where a Broker-based approach to optimize the VMs placement across multiple clouds is introduced. The *Broker* has two main tasks: the optimal deployment and the interfacing management task. To accomplish the first task a scheduling mechanisms based on pseudo mathematical

algorithm is provided. When the user asks for virtual resources he uses a *service description template* to specify the requested resources and optimization criteria. The Cloud Providers offer several VM configurations. The *Broker* has two main components: the *cloud scheduler* and the *VIM* (Virtual Infrastructure manager). The first one performs the optimization algorithm, the latter one provides an abstraction layer on top of the heterogeneous Cloud Providers by means of a uniform and generic interface to communicate with the other different Cloud Providers. The VIM is an open-nebula-based system. The algorithm has to maximize a mathematical optimization function: "*TIC*" (total infrastructure capacity). For this purpose the authors selected a scheduler based on the so called *AMPL* modelling language to solve the optimization function. Another framework working at the IaaS layer has been presented in Subramanian et al. [84]. The authors proposed a cloud *Broker*-based architecture that aims to provide an optimal virtual resource placement in a multi-clouds environment. The architecture counts 3 main actors: *Consumer*, *Broker* and *Cloud Provider*. All the Cloud Providers publish the offered resources in a central *Service Catalogue*. The placement process is divided in three steps. In the first one the *Broker* receives the user service request description and the weights of the *SMI* (Service Measurement Index). The *SMI* value is provided by a component that calculates the *SMI* values of the published offerings in the service catalogue. The user request description contains all the needed information: type of application, VMs config., location, minimum *SMI* score etc. In the second step the *Broker* calculates a possible set of cloud resources and services that can satisfy the request. In the last step an *SMI*-based evaluation of the Cloud Providers is performed and a cost optimized placement is developed. Also in Chaisiri et al. [85] a *Cloud Broker* and a mathematical algorithm to optimize the VMs placement across multiple clouds are presented. But here the algorithm optimizes the placement taking into consideration three different provisioning plans: (i) *reservation*, (ii) *utilization* and (iii) *on-demand*. It considers the resources price in each of the considered plans. The al-

gorithm aims to minimize the placement cost optimizing a mathematical objective function. The system counts four main components: *Cloud Broker*, *VM repository*, *Users* and *Independent Cloud Providers*. The *Cloud Broker* in order to solve the placement problem has to execute the algorithm. The authors consider two use cases: (i) the amount of requested resources is a priori known ; (ii) the requested resources are not known precisely. In the first case the problem can be solved by means of a deterministic integer program, whereas in the second case a stochastic integer program is needed. Anyway this paper focuses only on the mathematical algorithm without giving details about the technologies and standards used by the *Broker*. In Lucas-Simarro et al. [86] the authors depicted a *Broker*-base architecture able to work with several automatic scheduling strategies for a Multi-Cloud service deployment. The idea is to allow the users to distribute their services among multiple available clouds in a transparent manner taking into consideration different factors like cost optimization or performance optimization. The service is composed by two or more components and each of them can be deployed in a different service cloud providers. The *Broker*-based middleware counts three main components: *Cloud Manager*, *Scheduler*, *VM Manger* beside a central *DB*. There are two actors: a *User* and an *Administrator*. The *Administrator* sets the configuration of the *Broker*, the *User* instead receives information from the *Broker* and sends a service deployment request describing the service by means of a service description file. The *CM* periodically collects in the central *DB* all the information regarding the available resources and their price. The *VMm* finally deploys the virtual resources among multiple Cloud Providers managing and monitoring them. Moreover the authors give a wide explanation of the mathematical algorithms for scheduling processes which are used to perform an optimal deployment of the user service components. The last work of this sub-set is the framework-based solution of Kurdi et al. [87] that presented a combinatorial optimization framework (COM2) to develop a service composition across multiple-Clouds. The Framework has four

**Table 6.2:** *Percentage paper population values of each considered set of parameters*

| | ***FederationFederation*** | | ***Multi-CloudMulti-Cloud*** | |
|---|---|---|---|---|
| | **Brokering** | **Decentralized** | **Brokering** | **Decentralized** |
| *Cost* | 43% | 50% | 80% | 100% |
| *Performance* | 14,3% | - | 58% | 100% |
| *SLA* | 28,5% | 50% | 45% | 50% |
| *Other* | 71,5% | 83,5% | 77,7% | - |

main components: (i) a multi-Cloud environment composed by several Clouds. Each of them has a set of services file "*F*" and each file contains several services "*S*", (ii) a user interface for the user requests and to show the service composition sequence, (iii) the *cloud combiner* which selects the suitable set of clouds and composes a cloud combination list based on that set, (iv) the *service composer* which, taking as input the combination list from the *cloud combiner*, determines in each of the selected cloud the services that best fulfil the user request. At the end it produces a service composition sequence. The user gives as input a set of service files. The cloud combiner, starting from the cloud with the major number of service files, checks if there is an intersection between the set of the required files and the set of the provided *F*. If the answer yes, the cloud in question is added to the cloud combination list and the *F* files in the composer list. The algorithm finishes when the *composer list* is equal to the set of the user requested files. The selection algorithm aims to select a combination of services minimizing the overhead, and therefore maximizing the service performances beside a financial charge reduction.

#### 6.2.4.2 Performance

THIS sub-set, how just said before, includes about the 58% of the papers of the "**Brokering**" sub-tree. But eight of them (Jrad et al. [74], Yao et al. [82], Jrad et al. [75], Subramanian et al. [84], Tordsson et al. [83], Lucas-Simarro et al. [86], Garg et al. [78], Esposito et al. [80], Ngan et al. [76]) are shared with the sub-set "**Cost**" 6.2.4.1. However there is

another paper that proposes selection algorithms that do not consider cost factors in their execution but a multi-factor performance optimization. It is the work of Rehman et al. [88] where a multi-criteria cloud service selection methodology is presented. The authors did a mathematical formalization of the problem. The service selection process is based on the comparison between the users requirements and the descriptor vectors of the service. The algorithm will select the service which has the descriptor vector that best matches with the user requirement vector. This algorithm is general, consequently it is possible to consider any kind of functional and non-functional requirements.

#### 6.2.4.3 SLA

THIS sub-set contains about the 45% of the papers of the "**Brokering**" sub-tree. The papers Jrad et al. [75], Papaioannou et al. [81], Garg et al. [78], Subramanian et al. [84], Mansouri et al. [79] and Sundares et al. [60] are just analysed in the 6.2.4.1 while the approach presented in Redl et al. [89] falls out of that sub-set although it has a similar *SLA*-based approach. In this paper the authors present an automatic method to find a semantic equality among different *SLA* elements and a method which allows an automatic selection of optimal services. The system is based on a cloud market platform, performing a semantic matching between different *SLA* template documents. The algorithm is based on a probability of equivalence of the *SLA* templates. This value is used to calculate the *SLA*s elements equivalence. In order to provide an automatic selection of the Cloud Providers, a semantic equivalence between the user private *SLA* template and more public *SLA* templates is assumed. At the end of the process the public *SLA* template with the highest equivalence probability is chosen as the optimal offering. The algorithm anyway is able to select only one of the available clouds. In D'Andria et al. [90] a semantic match-making solutions to find the best overlapping between the user requirements and the Provider offering

is exploited. More in detail D'Andria et al. present the Cloud4SOA project which introduces a brokering architecture aiming to face the semantic interoperability between PaaS service providers. In order to reduce the vendor lock-in problems in the PaaS cloud layer, a uniform and global language in PaaS offering definition is needed. The main goals of Cloud4SOA are: (i) to propose architecture that helps the application developers in deploying their application on the PaaS provider that better meets their necessity, and (ii) to allow the applications to be deployed and seamlessly migrated between PaaS Cloud Providers that use the same technologies but a different language to define and model them. CLOUD4SOA uses a semantic-based matchmaking process to equalize the user application requirements terms to the PaaS offerings ones. The powerful of Cloud4SOA semantic matchmaking process is to harmonize the differences between different terms standing for the same concept by providing a set of relations between several different terms used by PaaS providers. In order to deploy an application on a specific PaaS Provider, the "deployment" module of Cloud4SOA, taking into consideration the application requirements, creates a specific application descriptor that is compatible with the format used by the PaaS Provider that the user has chosen to place that application. For the migration instead the "migration" module has to semantically translate the application requirements in a new application descriptor compatible with the new PaaS Provider. Finally Massonet et al. [91] faces the application deployment optimization in a multi-cloud environment taking into account not only the cost factors but also the security constrains. The case of study of the work is an application deployed over different clouds (public and private) belonging to different administration domains. The idea is to place the web servers and application servers close the customers (UK and Germany) and the database server in a private cloud located in Spain. The authors said that, during the selection process of the best CSP where to deploy an application component, security considerations have to be considered. To this goal the application components have to be modelled by making a

description of the arifacts, artifacts security and scalability requirements etc. . This model construction represents the "configuration" phase of the deployment workflow. The second phase is the "deployment" where a software component builds a deployment plan considering the constraints present in the model. Specifically the "Reasoner" component will perform a matching process between the application security and the CSP features. The third and last phase is the "execution" where another software component will execute the deployment plan. The selection process uses security service level objectives *SLO* in the *SLA* to create a common language in describing the security assurances. The "reasoner" through the analysis of the deployment model creates a"utility function" that optimises the deployment goal. The paper shows the pseudo-mathematical equations aiming to optimise the deployment cost and to select a CSP per component. The selection process for each application component will select from those CSPs that meet the security constraints the one having the lower price. We decided to put this work in this section because the first phase of the selection performs a matching between the security constraints of the component and the CSP features, and only after the cost considerations.

#### 6.2.4.4 Other

THE $78,5\%$ of the papers in this set are in common with the previously analysed "*Cost*" set in 6.2.4.1, the $57\%$ is shared with the "*Perfomance*" one discussed in 6.2.4.2, the $35,7\%$ is shared with the "*SLA*" while the remaining paper does not take into consideration neither cost factors nor performance or *SLA* requirements. This latter paper Fan et al. [92] considered a mathematical approach based on the user recommendations according with his satisfaction in service usage. Specifically in Fan et al. [92] the authors propose a mechanism that performs a service selection taking in consideration multi-dimensional users trust feedback ratings to create a reputation for a specific cloud service provider. It is based on

two different kinds of databases: (i) "trust value db" and (ii) reputation value database. When a user uses a target cloud service, he will assign, according to his satisfaction, the local trust value to the used service, storing on the service that value. The reputation base instead stores the reputation value of each service which is obtained by aggregating all the users feedback. The authors give a deep explanation of the mathematical algorithm.

CHAPTER 7

# Automating the Deployment of Multi-Cloud Applications in Federated Cloud Environments

THIS last work focuses on another operating context against those discussed in the previous Chapters. Although the subject remains to be the Cloud Federation, in order to demonstrate its validity and applicability also in other types of context, it has been applied the acquired experience in the filed of the "*Application Deployment in Cloud Computing*. Specifically this Chapter proposes an automated approach for deploying federated cloud applications across multiple providers based on standards. An architecture for a Coordinated Application Deployment System has been introduced and the way on how the deployment of an application, consisting of multiple components, can be coordinated across multiple providers belonging to the same federation by automatically adapting the deployment model has been shown.

## 7.1 Motivation & Fundamentals

In this section, we introduce a motivating scenario that is used to explain the presented approach. Afterwards, we introduce the TOSCA standard to provide the background required for understanding the concepts of this paper.

### 7.1.1 Motivating Scenario

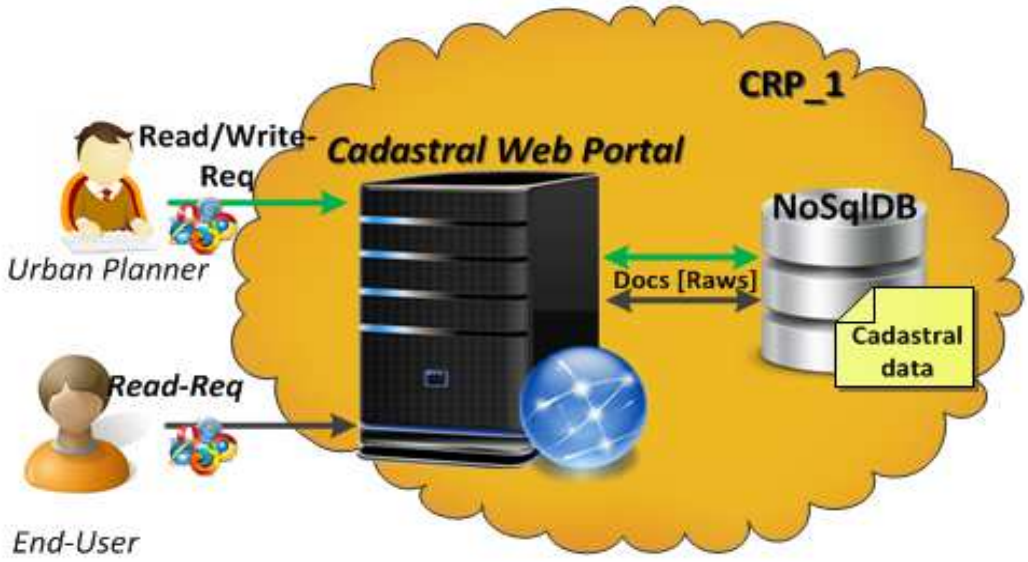



**Figure 7.1:** *Single cloud Cadastral Web Application*

In this subsection, we first describe a motivating scenario that is used to explain the presented concepts. Figure 7.1 depicts a web application, made available by the public administration, providing "Cadastral Servicies" to citizens. This application enables users to access the cadastral archives containing documents like maps, legal documents, data of real estates owners, economic rents to calculate the taxes, and so on. The typical users of the system are basically (i) the *end user* and (ii) the *urban planner*. The former assesses the system only in read-mode, while the urban planner has got both read and write permissions. The application consists of two main application blocks: a web portal

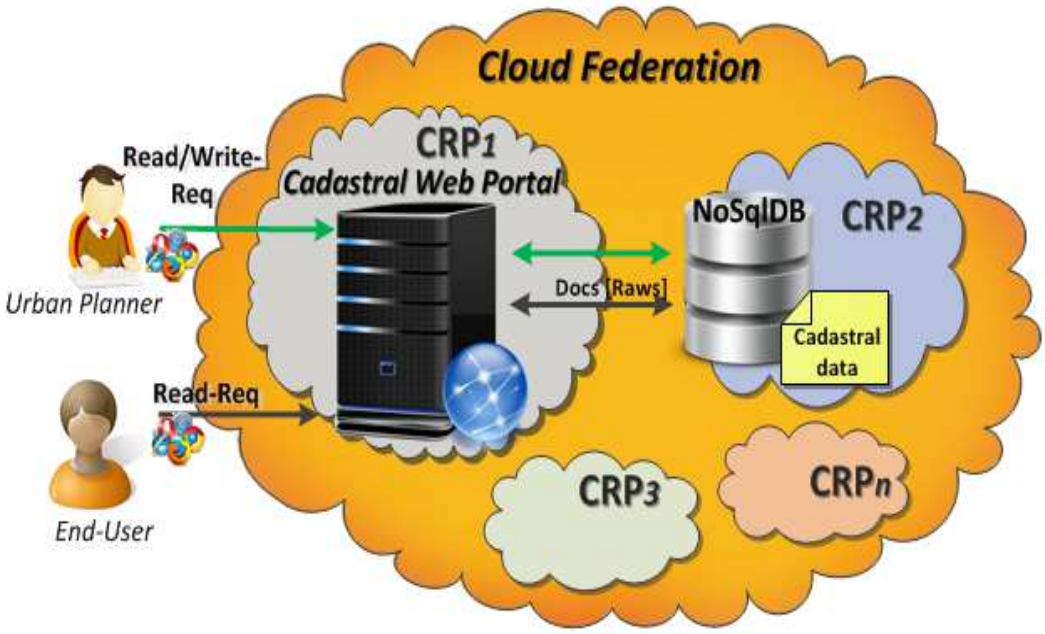



**Figure 7.2:** *Federated Multi-Cloud Cadastral Web Application*

acting as user interface between the client and the application logic, and a NoSQL-based DB for collecting all the cadastral and user data. The choice to use a NoSQL database is motivated by the kind of data has to be collected: maps, documents, geographic coordinates, and other kind of related meta-data. Our goal is to realize the automated deployment of the application components across several cloud providers participating in the same federation at runtime, i. e., when a *Public Administration (PA)* asks one CRP to deploy and to set up this an application. The resulting scenario is shown in Fig. 7.2, where the two mentioned blocks are deployed on two different Cloud Resource Providers, namely $CRP_1$ and $CRP_2$ .

In order to be able to automate this kind of deployment, in this paper, we will take advantages of the application modelling and orchestration features provided by the OASIS standard TOSCA. To provide all required background information about TOSCA, we explain the standard in the next Subsection 7.1.2. However, since TOSCA does not provide cloud federation management capabilities natively, we will exploit two other technologies to manage the federated communications between the in-

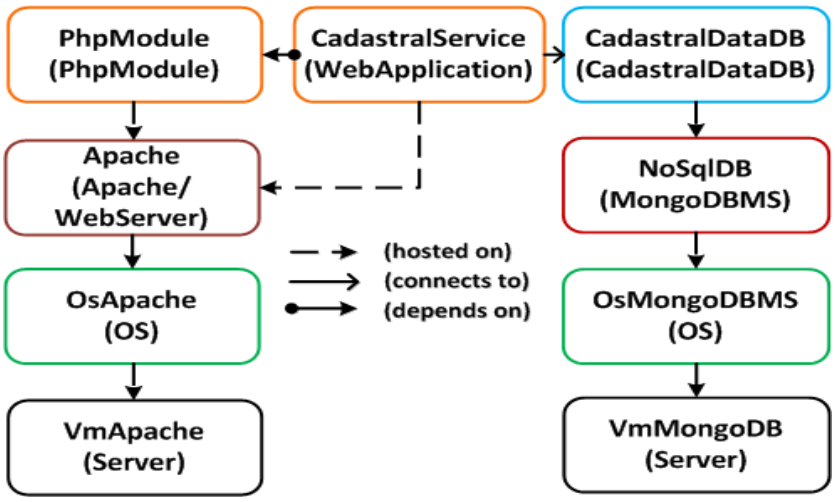



**Figure 7.3:** *Cadastral Web Application Topology Template*

volved CRPs: (i) the *XMPP* protocol for message exchanging and (ii) an additional abstraction layer made on top of a Message Oriented Middleware (MOM), which is able to maintain the federation and to drive the underlying piece of middleware of each CRP. These technologies will be deeply discussed in Section 7.3.

### 7.1.2 The TOSCA Standard

THE Topology and Orchestration Specification for Cloud Applications (TOSCA) [93–95] is an official OASIS standard. The main goals of the standard are to describe cloud- applications in an interoperable and portable manner in order to enable automating their deployment and management. TOSCA enables cloud users to model the structure of an application to be deployed using a *topology model*. Applications can be deployed based on these models by a *TOSCA Runtime Environment*, e. g., OpenTOSCA [96]. To automate management functionalities, for example, to scale out an application component, TOSCA employs the concept of *management plans*, which are executable process models, e. g., workflows, implementing a certain management functionality. Based on these two concepts, TOSCA enables the automated provisioning and management of cloud-applications. TOSCA topology models are modelled in the form

of a *topology template* (see Figure 7.3), which is a directed graph consisting of *node templates* representing the components (e. g., an Apache HTTP Server) and infrastructure resources (e.g. a virtual machine). These nodes are connected through directed edges, called *relationship templates*, which represent the dependencies between the nodes. For example, a relationship template may model that a PHP Web Application is *hosted on* an Apache HTTP Server and needs to be *connected to* a MongoDB database. TOSCA also enables modelling *node types* and *relationship types*, which define the semantics of the node and relationship templates. These type definitions are reusable building blocks specifying properties (e.g. the IP address) and management operations (e.g. "install the component"). These management operations are implemented using so called *implementation artifacts*, which are executable programs such as scripts. The business logic of the nodes themselves are implemented using *deployment artifacts*, which are implementations of the component—for example, a deployment artifact could be an archive containing the PHP files of the application to be deployed.

The standard also specifies the portable and self-contained *Cloud Service ARchive* (CSAR), which enables packaging all the aforementioned management plans, topology models, type definitions, and artifacts. As a result, CSARs can be automatically processed by any standard-compliant TOSCA Runtime Environment to deploy the described application.

## 7.2 A Conceptual Architecture for Federated Cloud Deployment

In the perspective of deploying multiple application components in a federated cloud environment, this section presents the first contribution of this paper in the form of an approach for coordinating and automating the deployment of components across multiple cloud providers participating in a cloud federation based on a single deployment model. We (i) first describe the conceptual architecture

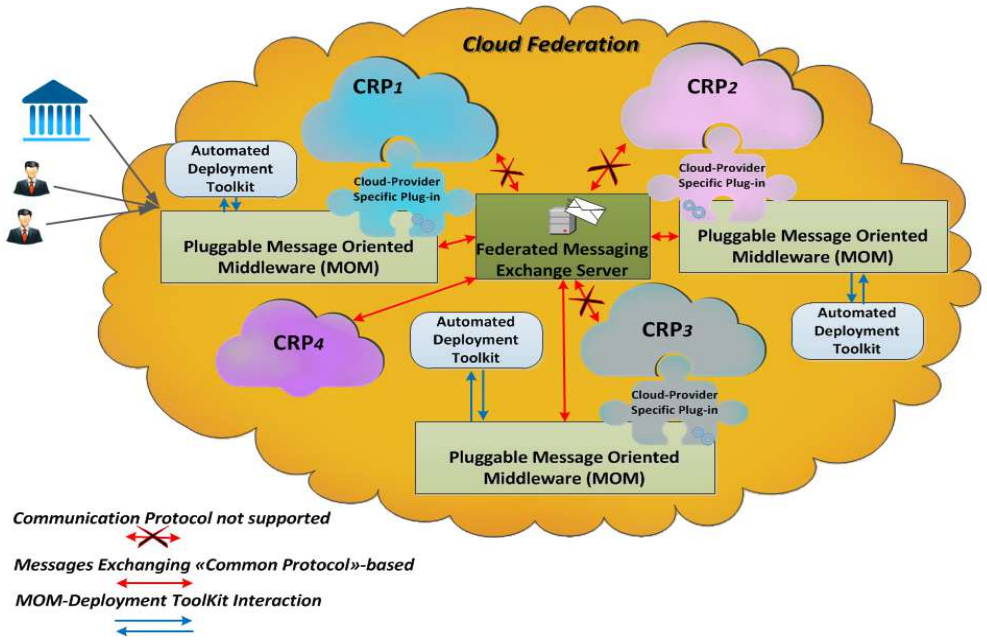



**Figure 7.4:** *Conceptual Architecture of CADS*

of a *Coordinated Application Deployment System (CADS)* in Section 7.2.1, followed by concepts for (ii) decision making, and (iii) deployment model completion in Section 7.2.2 and Section 7.2.3.

### 7.2.1 Overall CADS Architecture

FIGURE 7.4 shows a conceptual, high-level architecture of the proposed system. The depicted system aims for managing the multi-component application deployment over multiple federated clouds by splitting the application in different components and placing them on (possibly) different CRPs.

The main actors of the system are explained in the following. (i) The *end-user* sends an application deployment request to one CRP participating in the federation where the user has a valid account. This request (a) either refers to an application offered by the provider using an unique application ID or (b) contains a complete deployment model describing the application that shall be deployed, for example, a TOSCA CSAR (see Section 7.3). In this paper, we focus on the latter scenario. Moreover,

(ii) several *CRPs* taking part in the same federation and offering their available resources to end-users and other CRPs represent the second class of actors. (iii) Another actor is introduced by *Message Oriented Middlewares (MOM)* enabling the federation establishment, maintenance, and the communication among the CRPs, for example, to ask for available resources. Each CRP may be connected directly or through such a MOM to a (iv) *Federated Messaging Exchange Server*, which is the central core of the communication mechanism, through which all the messages exchanged by CRPs to coordinate the federated application deployment flow. Each CRP hosts an (vii) *application deployment toolkit*, which is responsible for receiving the deployment requests from the end-user. The toolkit then first decides which components of the received deployment model can be hosted directly by the respective CRP itself and which components shall be hosted on other CRPs of the federation by using the communication mechanism described above. Therefore, the toolkit sends out broadcast requests via the MOM and the exchange server to the federated CRPs that respond with their capabilities, i. e., the resources they are able to offer. After receiving these offers, the toolkit selects the CRPs that fit best for hosting the components and executes the deployment on them.

As shown in Figure 7.4, we can count four CRPs in the depicted scenario namely $CRP_1$, $CRP_2$, $CRP_3$ and $CRP_4$. The first three providers do not directly have support for the central communication mechanism, so in order to take part in the federation, they *must* set up and configure the MOM, whereas $CRP_4$ is natively compliant with that communication mechanism, so it can join the federation without an own MOM. Anyway, in order to handle the runtime application deployment, $CRP_4$ has to implement its own business intelligence controlling local component deployments requested by a federated provider. In the figure, missing support of the communication mechanism is highlighted by the red-crossed-arrows. The deployment toolkit is the entity in charge to perform the application component deployment.

In a dynamic environment like a cloud federation where CRPs can

join and leave whenever they want, each CRP needs to know how many and which CRPs are present within the federation at a given time. For this reason, the communication mechanism is based on the presence concept. In other words, in order to be easily identified and to identify, each available CRP has to provide a software agent that can access to a specific shared location. By exploiting this kind of communication mechanism, CRPs can retrieve (in addition to the presence information) the available resources of the other CRPs participating in the federation.

### 7.2.2 Discovery Phase and Decision-Making

According to the previous sections, our goal is to set up an environment which is able to accomplish the automated multi-component application deployment over multiple federated cloud providers. To achieve this goal, we need a software entity that takes as input the desired application deployment model (e. g., a topology model) as well as information about the participating CRPs resource offerings in order to give as output the deployment choices, namely *where* (on which CRP) the application components have to be deployed on. However, we are not discussing selection algorithms because this is out of the scope of this paper. Anyway, there are many works in the literature facing this topic in different ways. For example, [21] and [70] propose two similar match-making-based approaches, whereas [67], [71], and [82] propose different mathematical solutions. In this paper, we instead focus on the information retrieval concerning the availability of the CRPs resources. Because our system adopts a decentralized solution, a CRP, in order to find the other CRPs resources, needs to broadcast to each of them discovery messages. We call this message broadcasting process and the related response messages *discovery phase*. These information exchanges are performed by the MOMs and the central exchange server that, by means of the presence-based communication mechanism, will query all the other present CRPs retrieving their detailed resource information including the amount of available resources. For the sake of simplicity, we

focus on virtual machine resources (VMs), VM types (amount of CPUs, memory, storage), and their price in this paper. However, the presented approach and the following details can be easily applied to other *as a Service* offerings, too, e. g., PaaS offerings. These information about the CRPs will be used as input for the decision-making process. Then, after the system decided on which CRPs the application components shall be deployed, these results are sent to the automated deployment toolkit for the final *deployment phase*. In this phase, the original deployment request or the deployment model sent by the end-user (cf. Section 7.2.1) gets adapted for the chosen deployment decisions. For example, in case an end-user requested the deployment of an application consisting of several virtual machines described by a topology model, based on the selected CRPs, this model gets automatically adapted by the toolkit in terms of inserting the selected CRP nodes below the virtual machines. In particular, if the topology model depicted in Figure 7.2 was initially requested by the end-user, the system selects appropriate CRPs for hosting the two servers as virtual machines and inserts the respective nodes that describe these CRPs. Then, the resulting model gets automatically deployed on these CRPs. Based on this adaptation, the application components get distributed over different cloud providers participating in the federation. This step consists of exploiting the selected CRPs information to complete the final deployment model and gets explained in the next subsection.

### 7.2.3 Deployment Phase and Model Completion

After the CRPs to be employed are selected and which component stack of the application should be deployed to which CRP, the original deployment model needs to be adapted accordingly. There are works, for example the approaches presented by Hirmer et al. [97], for automatically completing incomplete deployment models. However, the available approaches lack support for choosing a specific CRP of a cloud federation based on requirements such as available computing resources or memory — moreover, many available approaches are not based on stan-

dards. Therefore, we introduce a software component called *Deployment Model Completion Manager*, which is able to adapt the stacks modelled in a deployment model to specific CRPs. As input, this component gets (i) the deployment model to be adapted as well as (ii) the decisions made by the decision making component described in Section 7.2.2. Afterwards, for every stack to be deployed, the Deployment Model Completion Manager adds the corresponding infrastructure node defined for the specific selected CRP. This requires a *CRP Registry* containing a list of CRPs and the required information to deploy software on this CRP, for example, information about the employed infrastructure for instantiating new virtual machines, e. g., OpenStack, and the required credentials. Therefore, each CRP maintains such a registry that is based on contracts between the CRPs. Thus, if a new CRP enters the cloud federation, its specific properties need to be added to the registries of participating CRPs, which is typically a manual process based on contracts. Using this information, properties of the infrastructure nodes, for example, credentials and desired ports, which are CRP-specific, are added automatically by the Deployment Model Completion Manager. After this step, the resulting model gets automatically deployed on the modelled CRPs using an appropriate runtime, which accesses the APIs of the involved CRPs to deploy the respective components remotely. In the following section, we explain these steps in detail based on standards.

## 7.3 Validation of the CADS Approach

In this section, we refine the conceptually described CADS approach presented in the previous section based on concrete standards and technologies. In Section 7.3.1, we first refine the presented CADS architecture and describe in Section 7.3.2 how the TOSCA standard can be employed to automate deploying application components on different federated providers.

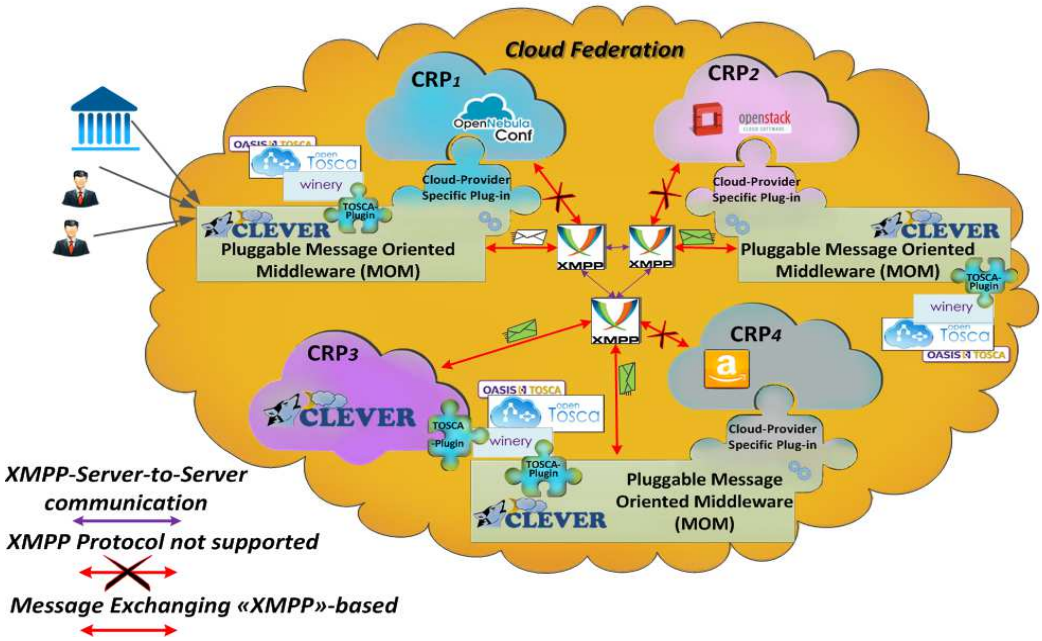



**Figure 7.5:** *Technical Realization of the CADS Architecture*

### 7.3.1 Standards-based CADS Architecture

SECTION 7.2.1 showed an overview of the conceptual CADS architecture. Based on that abstract system presentation, this section presents a validation based on standards and existing technologies. Figure 7.5 shows a detailed version of the architecture previously seen in Figure 7.4. Substantially, we replaced the conceptual entities with concrete technologies we identified as best solutions to achieve our goal.

The first refined entity is the end-user asking for the deployment of an application. In this refinement, we focus on the scenario in which the end-user sends a deployment model to be provisioned to a CRP. As we have several CRPs participating in the federation, each of them has its own hardware asset and provides virtualization middleware such as OpenStack. As mentioned earlier, we focus on VMs but the architecture can be easily adapted for other aaS-offerings, too. Another important system entity is an open source communication protocol exploited by the federated CRPs to exchange all the communication needed to manage the federation. Such

a protocol should be based on the presence concept (cf. Section 7.2.1). In this domain, the *Instant Messaging and Presence Service (IMPS)*[1], the *Session Initiation Protocol (SIP)* and its derivative *SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)*[2], as well as the *Extensible Messaging and Presence Protocol (XMPP)* [17] are the most important protocols. We chose XMPP due to its extensible paradigm, its decentralized architecture, its flexibility, and high-level of re-activeness. To exploit the XMPP protocol, there is the need to set up a Jabber/XMPP server (or more) providing basic messaging and XML routing features within the federation. It is possible to have more than one XMPP server, as it can be seen in Figure 7.5.

However, typically not all CRPs are natively XMPP-compliant. Therefore, in order to join the federation, each of them needs to exploit the features of an additional abstraction layer namely a Message Oriented Middleware (MOM) that supports XMPP. Our choice is the *CLoud-Enabled-Virtual-EnviRonment (CLEVER)* [98], which is a secure Message-Oriented Middleware for cloud federation. Our choice is validated by three main reasons: (i) the CLEVER inherent XMPP-based communication, (ii) its pluggable and (iii) federation management features. These features, therefore, make the CLEVER middleware suitable to manage both the federated communication and the underlying cloud platform. Moreover, in our federated scenario, we can directly have a CLEVER-based CRP ($CRP_3$ in Figure 7.5). Once the CLEVER-MOM of a specific CRP receives a deployment application request (including a deployment model), it first analyses the request to figure out how many resources that application needs, and afterwards triggers the "discovery phase". In particular, it sends discovery messages to the other CRPs to retrieve the federated resources availability and the related information (via response messages). This will take place merely querying the *shared location* on which it is published. More specifically, in an XMPP-based environment, the shared

[1] http://www.ietf.org/rfc/rfc2778.txt/
[2] https://datatracker.ietf.org/wg/simple/charter/

location consists of a specific *XMPP chat-room* where the cloud providers taking part in the federation have to hold a software agent subscription. The above mentioned messages are XML-based messages. Figure 7.6 shows a possible structure of these query messages:

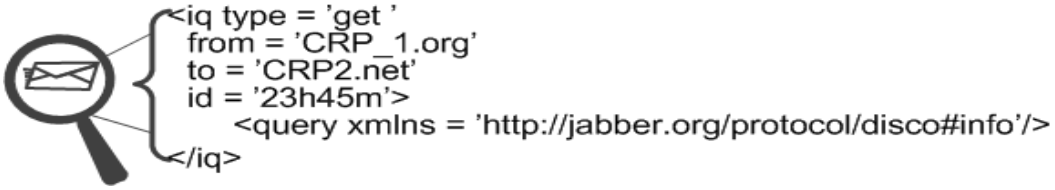



**Figure 7.6:** *Simplified XMPP Query Message*

Figure 7.7 shows a possible response of a CRP.


**Figure 7.7:** *Simplified XMPP Result Message*

In order to retrieve information about the CRPs resource state, the requesting cloud provider sends and XMPP *IQ* stanza of type `get` containing an empty <query/> element qualified by the <http://jabber.org/protocol/disco#info> name-space to all other providers present in the federation room. Each CRP receiving the query message will respond sending an XMPP *IQ* stanza of type `result`, which contains one or more <identity/> elements including static information and a potentially

unbounded number of <feature/> elements describing resource availability information (e. g., number of available VMs at a given moment for a specific type, VMs properties, VM price, etc.). This information will be elaborated by the decision-making software component to find out the best deployment solution.

To realize the automated deployment of federated cloud applications, the TOSCA standard is employed. Therefore, a Java-based plugin component is integrated into CLEVER providing a *TOSCA Topology Template Parser* and a *TOSCA Topology Completion Manager (TTCM)*, which implements the concept of the Deployment Model Completion Manager for TOSCA (cf. Section 7.2.3). The parser is responsible for analysing TOSCA Topology Models requested for deployment by the end-user. Afterwards, CLEVER uses the discovered information to utilize the TTCM for inserting corresponding node and relationship templates for injecting the selected CRPs into the topology template. For automating the actual deployment of the resulting model, a TOSCA Runtime Environment is employed. In our realization, we employ the *OpenTOSCA Runtime Environment* [96] [99] as a deployment toolkit, which is a standard-compliant, open-source implementation of a TOSCA Runtime. These two entities work together to accomplish the final topology completion and application deployment. The topology completion process will be discussed in detail in the next subsection.

### 7.3.2 TOSCA Topology Completion

In this section, we describe how the Deployment Model Completion Manager can be realized using TOSCA. Figure 7.8 shows CLEVER internal components as well as the necessary external tools to achieve the federated application deployment based on TOSCA. The Java-based TOSCA plugin is composed by (i) the TOSCA Topology Parser, (ii) the TOSCA Topology Completion Manager and (iii) the *Decision-Making component*. The process starts (step 1) with the user deployment request submission. It consists in sending a CSAR containing the incomplete

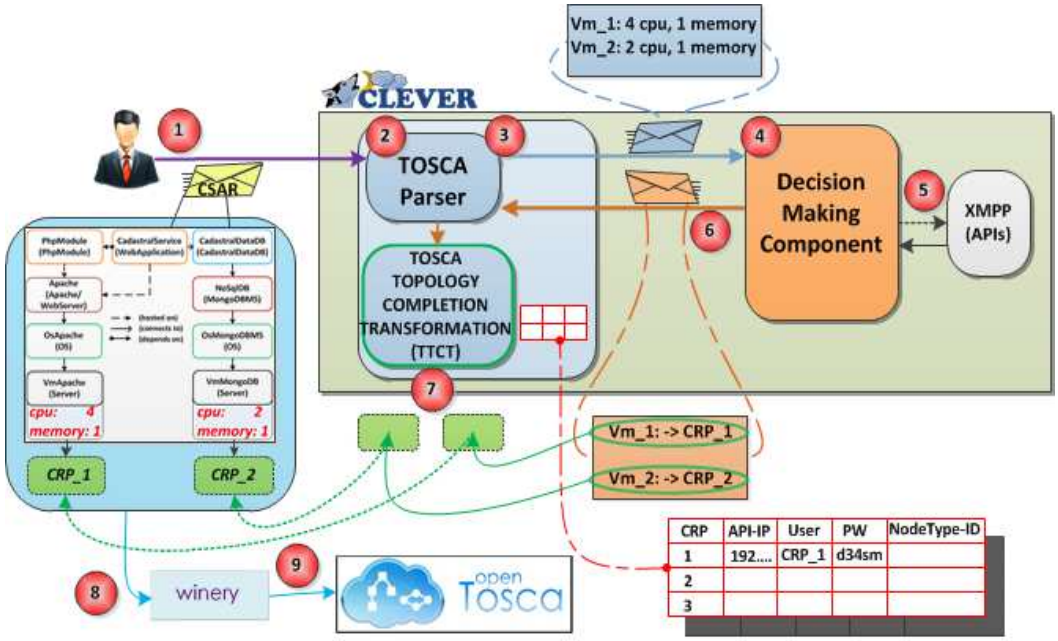


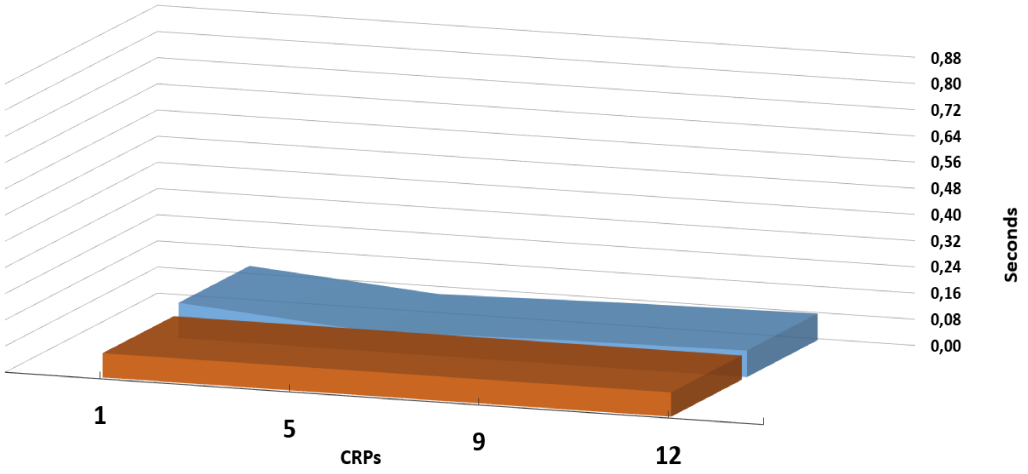

**Figure 7.8:** *Federated Topology Completion Process*

topology template (VMs are modelled but without CRPs). The submitted topology model is the same as the one shown in Figure 7.3, but now decorated with additional resource requirements. At step 2, the parser gets the CSAR and analyses the topology to figure out how many resources (VMs, CPUs, memory) the application to be deployed needs. At step 3, the parser composes an XML-based message with the above information and then sends it to the Decision-Making component (step 4). In order to retrieve the federated CRPs resource offerings, it invokes the XMPP APIs (step 5), thus, triggering the Discovery-Phase. After receiving the federated offerings, the Decision-Making component selects the CRPs which better meet the application components requirements. Then at step 6, it sends a response XML-message to the parser containing information about the selected CRPs to host the requested VMs. This information is forwarded to the TOSCA Topology Completion Manager, that (step 7) adds the node templates corresponding to the made decision as well as required properties to the incomplete topology model. Therefore, the manager also contains the registry described in Section 7.2.3, which con-

tains all CRPs involved in the federation and the associated properties as well as the id of the corresponding node template of each CRP. The completed topology template is then sent (step 8) to Winery [100]. Primarily, Winery is a modelling tool supporting the modelling of topologies based on the TOSCA standard. But besides that, Winery also has a repository containing TOSCA related artifacts like implementation artifacts. Therefore, Winery is able to fetch further required artifacts such as the implementation artifacts belonging to the previously added node templates and packages them into a final CSAR. This CSAR now contains a complete topology model and all artifacts required for executing the deployment. Therefore, it can be deployed fully automatically (step 9) using an appropriate standard-compliant TOSCA Runtime Environment.

## 7.4 Evaluation

THIS Section focuses on several experiments that we carried out on a real test-bed taking in consideration thirteen different CLEVER-based CRPs. With this section, we want to demonstrate that the addition of domains to the federation does not negatively affect the whole application deployment process. We arranged the test-bed considering the same federation configuration considered in Chapter 5. Each trial was repeated for 30 times and finally the mean value and confidence intervals were evaluated. Our interest falls on the XMPP-Communication times, specifically we analysed the time to obtain the resource availability from the federated CRPs and the effective time needed to a XMPP-message to go from a CRP to another one. In particular, the delay experienced by a CRP when asking for external resources to obtain availability information from all the federated CRPs can be considered the same of that obtained in Figure 5.1a in Chapter 5. We can see how the delay (due to the discovery phase) increases almost linearly when the number of CRPs grows although it is of only 1 second per each twelve additional CRPs. In fact, Figure 7.9 shows that the point-to-




| Average Time (s) | 0,074 | | | |
|---|---|---|---|---|
| **CRPs** | 1 | 5 | 9 | 12 |
| Forward Time (s) | 0,10745 | 0,060951299 | 0,073818345 | 0,081233154 |
| Conf Interval (98%) (s) | 0,00282097 | 0,00215871 | 0,001922828 | 0,002362203 |

**Figure 7.9:** *CRP-to-CRP message delivery time*

point message delivery time remains constant, very small, and *negligible* (roughly $74ms$), regardless of the number of CRPs present within the federation. The differences among the obtained values are very small and have an order of magnitude of $ms$, therefore, they are comparable with network delays. For this reason, it is hard to figure out and argue the trend of the $ForwardTime$. Surely it does not depend on the number of CRPs within the federation.

The experimental results just exposed demonstrate the XMPP communication delay does not negatively affect the overall application deployment process. In future work, we are going to test the performances of such a cross-cloud federation deployments to evaluate not only the XMPP communication time but all its components behaviour. In this scope, we will consider, for example, realistic and more complex multi-component applications and a federated environments characterized by hundreds of clouds that can dynamically take part and leave the federation in order to test the system scalability.

CHAPTER 8

# Conclusions

WITHOUT any doubt, the federation of cloud computing environments is till today one of the most challenging subject that IT specialists are facing. In fact, the federation is a very broad concept that encompasses many aspects of research such as the interoperability between different systems, the users data security, creation of trusted and high-quality environments in terms of both the availability and reliability of the offered services. After the provision of a first requirement analysis of such a federated environment, this work presented the application of the MapReduce paradigm in a federated Cloud environment, focusing on the video management context. It has been demonstrated how the federation can optimize the MapReduce video transcoding Job execution. The proposed solution integrated the Hadoop framework into CLEVER and used Amazon S3 as external CSP. The experimental results underlined, by means of the federation, it is possible to streamline the whole job process by exploiting

a further parallelization level, exactly that one provided by the federated environment. The proposed processing service has been deeply discussed focusing on job submission and analysing all of the involved delay and overhead in the process. In order to demonstrate that the Federation in Cloud Computing environment is the key word for the future and its advantages are not tangible only in the video management use case, it has been applied to another interesting use case. In particular, the challenge of a multi-component application deployment over multiple federated cloud providers has been tackled and the deployment system architecture has been introduced. The architectural solution was described and designed also highlighting the peculiarities of the selected technologies to achieve the desired goal. Moreover, in order to demonstrate that the XMPP communication does not negatively affect the overall application deployment process, the XMPP-based communication delay among the federated cloud domains have been thoroughly analysed. In future works, we are going to test the performances of such cross-cloud federation deployments to evaluate not only the XMPP communication time, but also all its components behaviour. For this purpose, realistic and more complex multi-component applications and federated environments, characterized by hundreds of clouds, will be considered. Other interesting topics to be analysed and integrated into the system are those of security and authorization. These two issues are a crucial step in Cloud Computing environments, especially in the Federation ones. In this context it is important to create an environment of trust among the involved Federated Clouds. Moreover in order to regulate the access to the resources that every Cloud provides, an access control system is needed. For these reasons, an aspect that certainly can be improved in the approach followed in this thesis is to integrate authentication systems (SSO) not yet implemented and to develop match-making policy-based solutions (XACML) during the selection process of the available clouds present in the federation. For example, selection algorithms based on policies could be developed. These selection algorithms aim to optimize: (i) the performance of the

system; (ii) the costs by choosing the cheapest resources which guarantee at the same time the SLA requirements requested by the user; (iii) the communication latency among users and cloud providers by placing the virtual resources near the users location. This could be possible implementing selection policies that take into consideration the geo-referencing parameters provided by the user. Moreover it would be interesting to investigate the new emerging Blockchain technology [101] to figure out how it can be integrated to support Federated security aspects. For example it could be exploited to create a fully decentralized control of the offered and purchased resources by registering transactions in a ledger, shared among the involved Federated clouds.

# Bibliography


[1] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Mu, and G. Tofetti, "Reservoir - when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, Mar. 2011.

[2] "Contrail - cloud federation computing project," 2014. [Online]. Available: http://contrail-project.eu/

[3] "Beacon - enabling federated cloud networking," 2014-2017. [Online]. Available: http://www.beacon-project.eu/

[4] A. Panarello, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "A requirements analysis for IaaS cloud federation," in *Proceedings of the 4$^{th}$ International Conference on Cloud Computing and Services Science*. Scitepress, 2014.

[5] A. Panarello, M. Fazio, A. Celesti, A. Puliafito, and M. Villari, "Cloud federation to elastically increase MapReduce processing resources," in *Lecture Notes in Computer Science*. Springer Science + Business Media, 2014, pp. 97–108.

[6] A. Panarello, A. Celesti, M. Fazio, A. Puliafito, and M. Villari, "Costs of a federated and hybrid cloud environment aimed at MapReduce video transcoding," in *2015 IEEE Symposium on Computers and Communication (ISCC)*. Institute of Electrical & Electronics Engineers (IEEE), Jul. 2015.

[7] ——, "A federated system for MapReduce-based video transcoding to face the future massive video-selfie sharing trend," in *Communications in Computer and Information Science*. Springer Science + Business Media, 2016, pp. 48–62.

[8] P. M. Mell and T. Grance, "The NIST definition of cloud computing," Tech. Rep., 2011.

[9] IETF, *Hypertext Transfer Protocol – HTTP/1.1*, https://www.ietf.org/rfc/rfc2616.txt, Std., June 1999.

[10] W3C, *ALL STANDARDS AND DRAFTS*, https://www.w3.org/TR/tr-technology-stds, Std., 2016.

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

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

[13] OpenQRM, "the next generation, open-source Data-center management platform", http://www.openqrm.com/.

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

[15] OpenStack, "Open source software for building private and public clouds", http://www.openstack.org.

[16] F. Tusa, M. Paone, and M. Villari, "CLEVER," in *Theory and Practice*. IGI Global, 2012, pp. 219–241.

[17] X. S. F. (XSF), "Extensible Messaging and Presence Protocol (XMPP): Core," Internet RFC 3920, October 2004. [Online]. Available: http://xmpp.org/extensions/index.html

[18] D. Laney, "3d data management: Controlling data volume, velocity and variety," February 2001. [Online]. Available: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

[19] IBM, "What is bigdata." [Online]. Available: https://www-01.ibm.com/software/vn/data/bigdata/

[20] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4:1–4:11, Jul. 2009.

[21] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *2010 IEEE 3rd International Conference on Cloud Computing*. Institute of Electrical & Electronics Engineers (IEEE), Jul. 2010.

[22] ——, "Three-phase cross-cloud federation model: The cloud SSO authentication," in *2010 Second International Conference on Advances in Future Internet*. Institute of Electrical & Electronics Engineers (IEEE), Jul. 2010.

[23] W3C, "Web services description language (wsdl)." [Online]. Available: https://www.w3.org/TR/wsdl

[24] OASIS, "extensible access control markup language." [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

[25] ——, "Security assertion markup language (saml)." [Online]. Available: http://saml.xml.org/saml-specifications

[26] "Openid." [Online]. Available: http://openid.net/developers/specs/

[27] S. Consortium, "Shibboleth." [Online]. Available: https://wiki.shibboleth.net/confluence/display/SHIB

[28] D. Laney, "The importance of big data: A definition."

[29] D. Bernstein, "Ieee p2302/d0.2 draft standard for intercloud interoperability and federation (siif)," IEEE, January 2012, https://www.oasis-open.org/committees/download.php/46205/p2302-12-0002-00-DRFT-intercloud-p2302-draft-0-2.pdf.

[30] D. Bernstein and Y. Demchenko, "The ieee intercloud testbed - creating the global cloud of clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 2, Dec 2013, pp. 45–50.

[31] B. Dong, Q. Zheng, F. Tian, K.-M. Chao, R. Ma, and R. Anane, "An optimized approach for storing and accessing small files on cloud storage," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1847–1862, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804512001610

[32] B. Dong, J. Qiu, Q. Zheng, X. Zhong, J. Li, and Y. Li, "A novel approach to improving the efficiency of storing and accessing small files on hadoop: A case study by powerpoint files," in *2010 IEEE International Conference on Services Computing (SCC)*, Jul. 2010, pp. 65–72.

[33] L. Jiang, B. Li, and M. Song, "The optimization of hdfs based on small files," in *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, Oct 2010, pp. 912–915.

[34] K. Aishwarya, A. A. Ram, M. Sreevatson, C. Babu, and B. Prabavathy, "Efficient prefetching technique for storage of heterogeneous small files in hadoop distributed file system federation," in *2013 Fifth International Conference on Advanced Computing (ICoAC)*, Dec 2013, pp. 523–530.

[35] M. Kim, Y. Cui, S. Han, and H. Lee, "Towards efficient design and implementation of a hadoop-based distributed video transcoding system in cloud computing environment," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 2, pp. 213–224, March 2013.

[36] "Vision cloud project, funded by the european commission seventh framework programme (fp7/2006-2013) under grant agreement n. 257019." 2014. [Online]. Available: http://www.visioncloud.eu/

[37] D. Bruneo, F. Longo, D. Hadas, and E. K. Kolodner, "Analytical investigation of availability in a vision cloud storage cluster," *SCPE*, vol. 14, no. 4, Jan. 2014.

[38] E. K. Kolodner, S. Tal, D. Kyriazis, D. Naor, M. Allalouf, L. Bonelli, P. Brand, A. Eckert, E. Elmroth, S. V. Gogouvitis, D. Harnik, F. Hernandez, M. C. Jaeger, E. B. Lakew, J. M. Lopez, M. Lorenz, A. Messina, A. Shulman-Peleg, R. Talyansky, A. Voulodimos, and Y. Wolfsthal, "A cloud environment for data-intensive storage services," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. Institute of Electrical & Electronics Engineers (IEEE), Nov. 2011.

[39] R. G. Cascella, L. Blasi, Y. Jegou, M. Coppola, and C. Morin, "Contrail: Distributed application deployment under SLA in federated heterogeneous clouds," in *The Future Internet*. Springer Science + Business Media, 2013, pp. 91–103.

[40] "Mosaic - multi-modal situation assessment and analytics platform." 2014. [Online]. Available: http://www.mosaic-fp7.eu/

[41] "Bonfire project," http://www.bonfire-project.eu/, 2014. [Online]. Available: http://www.bonfire-project.eu/

[42] A. C. Hume, Y. Al-Hazmi, B. Belter, K. Campowsky, L. M. Carril, G. Carrozzo, V. Engen, D. García-Pérez, J. J. Ponsatí, R. Kűbert, Y. Liang, C. Rohr, and G. V. Seghbroeck, "BonFIRE: A multi-cloud test facility for internet of services experimentation," in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Science + Business Media, 2012, pp. 81–96.

[43] D. García-Pérez, J. Á. L. del Castillo, Y. Al-Hazmi, J. Martrat, K. Kavoussanakis, A. C. Hume, C. V. López, G. Landi, T. Wauters, M. Gienger, and D. Margery, "Cloud and network facilities federation in BonFIRE," in *Euro-Par 2013: Parallel Processing Workshops*. Springer Science + Business Media, 2014, pp. 126–135.

[44] "Stratuslab, darn simple cloud." 2014. [Online]. Available: http://www.stratuslab.eu/

[45] "Cloudwave project, funded by the european commission seventh framework programme (fp7/2006-2013) under grant agreement n. 610802." 2013. [Online]. Available: http://cloudwave-fp7.eu/

[46] A. Nus and D. Raz, "Migration plans with minimum overall migration time," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. Institute of Electrical & Electronics Engineers (IEEE), May 2014.

[47] "Optimis - optimized infrastructure service," 2014. [Online]. Available: http://www.optimis-project.eu/

[48] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan, "OPTIMIS: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, Jan. 2012.

[49] "The 4caast project," 2014. [Online]. Available: http://www.4caast.eu/

[50] S. Yangui, I.-J. Marshall, J.-P. Laisne, and S. Tata, "CompatibleOne: The open source cloud broker," *J Grid Computing*, vol. 12, no. 1, pp. 93–109, Nov. 2013.

[51] "Seaclouds projects," February 2016. [Online]. Available: http://www.seaclouds-project.eu/

[52] A. Brogi, M. Fazzolari, A. Ibrahim, J. Soldani, J. Carrasco, J. Cubo, F. Durán, E. Pimentel, E. Di Nitto, and F. D Andria, "Adaptive management of applications across multiple clouds: The seaclouds approach," *CLEI Electronic Journal*, vol. 18, no. 1, pp. 2–2, 2015. [Online]. Available: http://www.scielo.edu.uy/pdf/cleiej/v18n1/v18n1a02.pdf

[53] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, vol. 45, pp. 134–150, Oct. 2014.

[54] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. M. Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, Sep. 2012.

[55] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and Architectures for Parallel Processing*. Springer Science + Business Media, 2010, pp. 13–31.

[56] D. Petcu, "Multi-cloud," in *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds - MultiCloud*. Association for Computing Machinery (ACM), 2013.

[57] ——, "Portability and interoperability between clouds: Challenges and case study," in *Towards a Service-Based Internet*. Springer Science + Business Media, 2011, pp. 62–74.

[58] T. Subramanian and N. Savarimuthu, "A study on optimized resource provisioning in federated cloud," Mar. 2015.

[59] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments," *CSUR*, vol. 47, no. 1, pp. 1–47, May 2014.

[60] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection," in *2012 IEEE Fifth International Conference on Cloud Computing*. Institute of Electrical & Electronics Engineers (IEEE), Jun. 2012.

[61] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 14–27, Jan. 2015.

[62] M. Gahlawat and P. Sharma, "VM selection framework for market based federated cloud environment," in *International Conference on Computing, Communication & Automation*. Institute of Electrical & Electronics Engineers (IEEE), May 2015.

[63] J. B. Abdo, J. Demerjian, H. Chaouchi, K. Barbar, and G. Pujolle, "Broker-based cross-cloud federation manager," in *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*. Institute of Electrical & Electronics Engineers (IEEE), Dec. 2013.

[64] E. Badidi, "A context broker federation for QoC-driven selection of cloud-based context services," in *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*. Institute of Electrical & Electronics Engineers (IEEE), Dec. 2014.

[65] M. Caballer, I. Blanquer, G. Moltó, and C. de Alfonso, "Dynamic management of virtual infrastructures," *J Grid Computing*, vol. 13, no. 1, pp. 53–70, Apr. 2014.

[66] M. S. Q. Z. Nine, M. A. K. Azad, S. Abdullah, and N. Ahmed, "Dynamic load sharing to maximize resource utilization within cloud federation," in *Cloud Computing and Big Data*. Springer Science + Business Media, 2015, pp. 125–137.

[67] C.-Y. Liu, K.-C. Huang, Y.-H. Lee, and K.-C. Lai, "Efficient resource allocation mechanism for federated clouds," *International Journal of Grid and High Performance Computing*, vol. 7, no. 4, pp. 74–87, Oct. 2015.

[68] A. Kertesz, G. Kecskemeti, M. Oriol, P. Kotcauer, S. Acs, M. Rodríguez, O. Mercè, A. C. Marosi, J. Marco, and X. Franch, "Enhancing federated cloud management with an integrated service monitoring approach," *J Grid Computing*, vol. 11, no. 4, pp. 699–720, Jun. 2013.

[69] M. Giacobbe, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "An approach to reduce carbon dioxide emissions through virtual machine migrations in a sustainable cloud federation," in *2015 Sustainable Internet and ICT for Sustainability (SustainIT)*. Institute of Electrical & Electronics Engineers (IEEE), Apr. 2015.

[70] K.-H. Yeh, "An efficient resource allocation framework for cloud federations," *Information Technology And Control*, vol. 44, no. 1, Mar. 2015.

[71] S. Rebai, M. Hadji, and D. Zeghlache, "Improving profit through cloud federation," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. Institute of Electrical & Electronics Engineers (IEEE), Jan. 2015.

[72] C. Negru, F. Pop, O. C. Marcu, M. Mocanu, and V. Cristea, "Budget constrained selection of cloud storage services for advanced processing in datacenters," in *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*. Institute of Electrical & Electronics Engineers (IEEE), Sep. 2015.

[73] Y. Kajiura, S. Ueno, A. Kanai, S. Tanimoto, and H. Sato, "An approach to selecting cloud services for data storage in heterogneous-multicloud environment with high availability and confidentiality," in *2015 IEEE 12th International Symposium on Autonomous Decentralized Systems*. Institute of Electrical & Electronics Engineers (IEEE), Mar. 2015.

[74] F. Jrad, J. Tao, and A. Streit, "A broker-based framework for multi-cloud workflows," in *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds - MultiCloud 13*. Association for Computing Machinery (ACM), 2013.

[75] F. Jrad, J. Tao, A. Streit, R. Knapper, and C. Flath, "A utility-based approach for customised cloud service selection," *IJCSE*, vol. 10, no. 1/2, p. 32, 2015.

[76] L. D. Ngan and R. Kanagasabai, "OWL-s based semantic cloud service broker," in *2012 IEEE 19th International Conference on Web Services*. Institute of Electrical & Electronics Engineers (IEEE), Jun. 2012.

[77] R. N. Calheiros, A. N. Toosi, C. Vecchiola, and R. Buyya, "A coordinator for scaling elastic applications across multiple clouds," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1350–1362, Oct. 2012.

[78] S. K. Garg, S. Versteeg, and R. Buyya, "SMICloud: A framework for comparing and ranking cloud services," in *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. Institute of Electrical & Electronics Engineers (IEEE), Dec. 2011.

[79] Y. Mansouri, A. N. Toosi, and R. Buyya, "Brokering algorithms for optimizing the availability and cost of cloud storage services," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. Institute of Electrical & Electronics Engineers (IEEE), Dec. 2013.

[80] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," *IEEE Transactions on Computers*, pp. 1–1, 2015.

[81] T. G. Papaioannou, N. Bonvin, and K. Aberer, "Scalia: An adaptive scheme for efficient multi-cloud storage," in *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*. Institute of Electrical & Electronics Engineers (IEEE), Nov. 2012.

[82] W. Yao and L. Lu, "A selection algorithm of service providers for optimized data placement in multi-cloud storage environment," in *Communications in Computer and Information Science*. Springer Science + Business Media, 2015, pp. 81–92.

[83] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358–367, Feb. 2012.

[84] T. Subramanian and N. Savarimuthu, "Application based brokering algorithm for optimal resource provisioning in multiple heterogeneous clouds," *Vietnam J Comput Sci*, vol. 3, no. 1, pp. 57–70, Dec. 2015.

[85] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. Institute of Electrical & Electronics Engineers (IEEE), Dec. 2009.

[86] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1431–1441, Aug. 2013.

[87] H. Kurdi, A. Al-Anazi, C. Campbell, and A. A. Faries, "A combinatorial optimization algorithm for multiple cloud service composition," *Computers & Electrical Engineering*, vol. 42, pp. 107–113, Feb. 2015.

[88] Z. ur Rehman, F. K. Hussain, and O. K. Hussain, "Towards multi-criteria cloud service selection," in *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Institute of Electrical & Electronics Engineers (IEEE), Jun. 2011.

[89] C. Redl, I. Breskovic, I. Brandic, and S. Dustdar, "Automatic SLA matching and provider selection in grid and cloud computing markets," in *2012 ACM/IEEE 13$^{th}$ International Conference on Grid Computing*. Institute of Electrical & Electronics Engineers (IEEE), Sep. 2012.

[90] F. DAndria, S. Bocconi, J. G. Cruz, J. Ahtes, and D. Zeginis, "Cloud4Soa: Multi-cloud application management across PaaS offerings," in *2012 14$^{th}$ International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. Institute of Electrical & Electronics Engineers (IEEE), Sep. 2012.

[91] P. Massonet, J. Luna, A. Pannetrat, and R. Trapero, "Idea: Optimising multi-cloud deployments with security controls as constraints," in *Lecture Notes in Computer Science*. Springer Science + Business Media, 2015, pp. 102–110.

[92] W.-J. Fan, S.-L. Yang, H. Perros, and J. Pei, "A multi-dimensional trust-aware cloud service selection mechanism based on evidential reasoning approach," *International Journal of Automation and Computing*, vol. 12, no. 2, pp. 208–219, Dec. 2014.

[93] OASIS. (2013) Topology and Orchestration Specification for Cloud Applications Version 1.0.

[94] OASIS, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0*, 2013.

[95] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, *TOSCA: Portable Automated Deployment and Management of Cloud Applications*, ser. Advanced Web Services. Springer, Jan. 2014, pp. 527–549.

[96] T. Binz *et al.*, "OpenTOSCA - A Runtime for TOSCA-based Cloud Applications," in *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013)*. Springer, Dec. 2013.

[97] P. Hirmer, U. Breitenbücher, T. Binz, and F. Leymann, "Automatic Topology Completion of TOSCA-based Cloud Applications," in *Proceedings des CloudCycle14 Workshops auf der 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*. GI, 2014, pp. 247–258.

[98] A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "SE CLEVER: A Secure Message Oriented Middleware for Cloud Federation," in *IEEE Symposium on Computers and Communications (ISCC 2013)*, Split, Croatia, July 7 2013.

[99] University of Stuttgart – Institute of Architecture of Application Systems. (2016) OpenTOSCA – Open Source TOSCA Ecosystem. http://www.iaas.uni-stuttgart.de/OpenTOSCA/indexE.php.

[100] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann, "Winery – A Modeling Tool for TOSCA-based Cloud Applications," in *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013)*. Springer, Dec. 2013, pp. 700–704.

[101] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," *ITA-SEC. CEUR-WS. org, To Appear*, 2017.