Towards a distributed SaaS management system in a multi-cloud environment

Linda Ouchaou, Hassina Nacer, Chahrazed Labba

A Distributed Service Management System for Community Cloud Federation Environments

Linda Ouchaou[a], Hassina Nacer[a], Chahrazed Labba[b]

[a]*MOVEP Laboratory, Computer Science Department, University of Science and Technology Houari Boumediene, 16000 Algiers, Algeria*
[b]*Lorrain University, LORIA F-54000 Nancy, France*

Abstract

Nowadays, cloud federation is gaining a momentum since it provides its users with reduced costs, greater flexibility and elasticity, high availability and better fault-tolerance. Despite these advantages, these environments present many challenges, including the difficulty of fulfilling diverse user requirements due to the exponential growth of the offered services. Further, the diversity and heterogeneity of cloud providers in the federation lead to interoperability issues that complicate the service management. Both of these issues, if not dealt with effectively, can have adverse impacts on the federation's profit as well as on the user experience. Thus, to meet these challenges, we propose in this paper a threefold contribution (i) a noval cloud federation architecture, (ii) a suitable service management system and (iii) a service publication algorithm in order to manage, store and retrieve efficiently cloud services within the federation. Our solution consists in combining several concepts including trust, semantic Web (ontologies), clustering methods, graph theory and the community deployment model. Our aim is to automate the management process, maximise the profit and ensure a better user experience. Our experiments highlight the effectiveness of the proposed cloud federation architecture as well as the deployed management system in optimizing the storage space and answering effectively and rapidly users' requests.

Keywords: Cloud Computing, Federation of Clouds, Service Management, Virtual Storage, Publication, Discovery, Federation.

1. Introduction

The cloud computing paradigm, as it emerged initially, consists of a single provider offering a centrally managed and administered set of resources as a service. However, being limited by the use of a single cloud service provider is

Email addresses: ouchaoulynda1494@gmail.com (Linda Ouchaou), sino_nacer@yahoo.fr (Hassina Nacer), chahrazed.labba@loria.fr (Chahrazed Labba)

very risky and costly, as it may not meet all of the user's current and future requirements. Today's cloud users and cloud providers tend to avoid this vendor lock-in problem by adopting the use of multi-cloud platforms and, more recently, cloud federations.

In fact, cloud federation is the coalition of multiple Cloud Providers (CP) that provides their users with various benefits. These advantages include surmounting the constraint of finite physical resources of individual CP [1], offering a wider variety of services without any constraints on the number of accepted requests, overcoming the vendor lock-in issue, enjoying lower latency, greater elasticity and scalability, as well as better fault tolerance.

While the use of cloud federation has a long list of benefits, it also comes with a series of challenges. These include (i) trust and interoperability issue introduced by the diversity and the heterogeneity of the CP, (ii) storage management issues due to the exponential growth of the published software services (SaaS) (iii) lack of service description standards and automated service discovery methods that effectively retrieve desired services to respond to users' demands (vi)the inability to deal with the composition and discovery issues of SaaS services due to the fact that cloud federations operate at the IaaS provider level. It therefore focuses only on the IaaS cloud services layer [48 j2].

Several approaches have been proposed in the literature to address these issues. We distinguish on the one hand the solutions that address the service management issue, such as the clustering-based methods [2, 3, 4, 5, 6, 7], registry-based methods [8], graph-based methods [2, 3, 5], ontology-based methods [3, 9, 10, 7, 8], and uni or multi dimensional indexing-based methods [11, 12, 13, 14, 15]). On the other hand, authors proposed several solutions for addressing the cloud federation creation issue [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. However, existing solutions consist either in managing services or in building efficient federation infrastructures, none of them takes into account the possibility of addressing both service management and federation creation at the same time. Furthermore, despite the effectiveness of federated clouds for resource scaling, cloud federations are unable to handle the composition and discovery issues due to the fact that they operate at the provider's (not service) level and focus solely on the IaaS cloud services' layer [20].

Therefor, to cope with these challenges, we propose in this paper (i) a hierarchical cloud federation architecture built by using trust, semantic web (WordNet domain ontology [28]) as well as the community deployment model. The trust concept is used in order to obtain a trustworthy federation that removes the interoperability issues between the involved cloud providers. The ontology and the community deployment model are used to organise and deploy the involved cloud providers in the federation into communities with a domain of interest each, (ii) provide a distributed service management system by using clustering and graph theory methods to efficiently store the published services in the federation and (iii) provide an automated service publication algorithm (while respecting the proposed architecture hierarchy and management system) to handle the service publication requests. The reason we use several concepts (trust, ontologies, the community deployment model, clustering and graph theory) is

due to the fact that we believe that their combination has a greater added value as it allows us to address several issues at once to obtain a more complete solution.

The remainder of this paper is organized as follows. Section 2 presents a review of some existing cloud federation architectures as well as a review of some cloud service management solutions. Section 3 presents the proposed solution for the service management problem in cloud federation environments. Section 4 states the experimental setup and analyzes the experimental results. Finally, section 6 concludes the paper.

2. Related Works

We focus in this section on giving an overview on the move to federated cloud environments as well as presenting the works conducted on cloud service management in either federated of non federated cloud environments.

2.1. From Single Clouds to Community Cloud Federation

Dealing with single-cloud providers became less popular with customers due to several risks such as service availability failure [18], the increasing difficulty in meeting the variety of users' needs [29], the vendor-lock-in issues [30], data reliability and consistency [31], trust [25], and security. To cope with these challenges, the adoption of multi-cloud environments has been advocated by many organizations and researchers, one can cite intercloud [31, 32, 33], cloud-of-clouds [34] and multi-cloud [10]. In [33] the authors assume that the main objective of migrating to multi-cloud environments is to improve the services offered within single-clouds by distributing reliability, trust and security among several CPs. However, the providers involved in a multi-cloud environment create interoperability issues because of them being heterogeneous. This heterogeneity is resulted by the lack of a standard architecture. Consequently, each CP is being responsible for the management of its own services according to their own policies and having no information about the rest of the services of the other CPs. Moreover, this heterogeneity creates difficulties for users to switch between various cloud providers.

To overcome this challenge, the concept of trust is introduced by several research works within multi-cloud environments in order to create a single access point and to run this environment as a single organisation by creating a cloud federation [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]

García [35] assumes that a federation should enable the collaboration and cooperation of different providers, delivering resources to the users when a single resource provider is not able to satisfy the user demands, in a collaborative way. Whereas, Hammoud et al. [26] define cloud federation as an architecture that allows cloud providers using their unallocated virtual machines, by merging their resources to serve a pool of clients whose requests cannot be handled by any of these providers alone. Moreover, Abdo et al. [18] assume that cloud federation is the next step in creating a new cloud computing ecosystem.

The National Institute of Standards and Technology (NIST) presents in [36] the NIST cloud federation reference architecture as a conceptual model in order to identify the fundamental federation functions that may be important to different participating stakeholders in different application domains. It includes the following components: Cloud Service Consumer, Cloud Service Provider, Federation Manager, Federation Operator, Federation Auditor, Federation Carrier, Federation Broker. These components identify how federations can be organized and used, but does not dictate how any of this must be done. That is determined by the requirements of the specific federation instance, as defined by the federation members. They also emphasis that there is a wide spectrum of possible federation deployments that can range from very simple federations where many of the elements of this conceptual model are simply not needed, to very large federations that will require extensive governance machinery to be in place.

For instance, in [26], authors propose an approach based on genetic algorithms and evolutionary game theory to address the problem of forming a cloud federation in order to achieve optimality in profit and stability among federations. Authors in [25] propose to address the problem of building a trustworthy cloud federation through the game theoretic approach by classifying the providers, based on how they cooperate to form a coalition. [23] propose a framework for cloud federation and introduces the basic processes of building service federation among multiple cloud providers in order to address the limitation of provider's capability and the magnanimity of task quantity for big data, as well as a multi-objectives task assigning model in the cloud federation, to help organizations manage and optimize their service resources. In [21], authors propose to find a solution build a cloud Federation by following a game-theoretic approach considering the trust between the participating CSPs. This technique will also allow the CSPs to take decision unilaterally based on their own decisions depending on the profit they would receive upon inclusion into the federation.

2.2. Cloud Service Management

We propose through Fig. 1 a classification of existing cloud service management methods including clustering-based methods, registry-based methods, graph-based methods, ontology-based methods and indexing-based methods (uni or multidimensional). These solutions can be used either in single or multiple cloud environments (federated or not), they only need to be adapted according to the characteristics of each. In this work, we focus on examining the SaaS management solutions for any environment and study their adaptability for a cloud federation. Table **??** gives an overview of this study.

Authors in [2] proposed a solution to manage Mashups services into clusters in a cloud-based Internet of Things environment. To proceed, the authors first built a graph service network to describe Mashups, Web APIs, and their relations. Then, they manage the obtained graph into Clusters using genetic algorithms. The algorithm used determines the number of clusters automatically. The work in [4] propose a clustering-based Approach for SaaS services

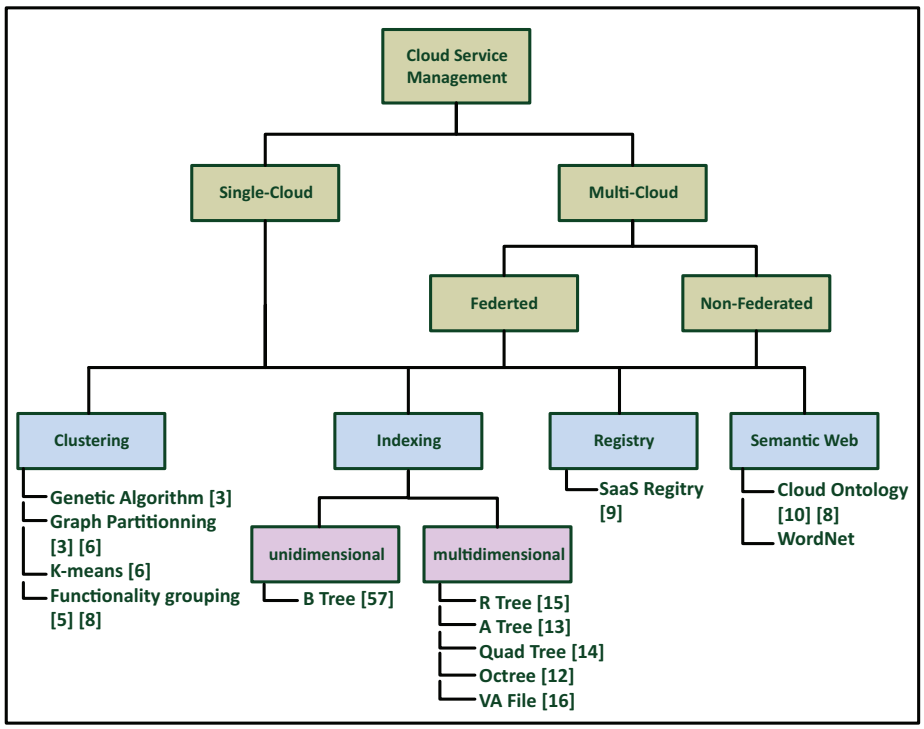

Fig. 1: Cloud Services Management Methods Classification.

discovery in single-cloud environment. They manage and cluster SaaS services into functionally similar clusters. The similarity measure used to build the clusters is based on the number of occurrences (frequency) of each concept in each service, and each cluster is represented by one of its SaaS services (chosen randomly). The approach presents some limitations, it is difficult to guarantee that each service is well described with adequate concepts and adequate number of occurrence. Moreover, the request is modeled using a keyword vector, and it remains difficult for clients to specify queries that perfectly match their requirements due to the lack of knowledge of their expected service functionalities. Authors in [9] worked on developing technology solutions for automated cloud service management using semantic Web and Text Mining techniques in order to represent cloud legal documents in a machine-actionable form. These automated techniques can be developed to understand, analyze and make decisions based on the knowledge extracted from documents such as service level agreements. Therefore, it is crucial to describe the documents in a very adequate way, which can appear to be difficult and time consuming. In [5], authors propose to manage Mashup services into clusters. The authors first build a service graph network based on the relationships among services. Then, a clustering algorithm based on K-means and Agnes methods is used to derive Mashup ser-

vice clusters. In[7] an approach with the capability to automatically identify and cluster cloud services into functionally similar clusters using cloud service ontology and cosine similarity is proposed. The ontology is build using the NIST [37] cloud concepts. This cloud service categorization is helpful in determining whether a given web source is a cloud service or not. We distinguish also management solutions based on uni or multi dimensional index structures [14, 38, 12, 13, 15, 11]. These solutions propose a two-level management model for data (DaaS) in a cloud Computing environment. In a network of several compute nodes, data is managed as following; each compute node manages its data by building a local index according to a multidimensional tree structure (R-Tree, A-Tree, Quad-Tree, etc.). Then, each compute node (or some particular compute nodes) publishes a portion of its local index to a global index. The global index is built on top of the local indexes, and it represents an overview of the offering of the network. However, despite the fact that these solutions solve the problem for large-scale data (DaaS) in the cloud environment, these solutions do not include service types for SaaS service models. Authors in [34] present a virtual cloud storage system called DepSky, which consists of a combination of different clouds to build a multi-cloud environment. The DepSky system addresses the availability and the confidentiality of data in their storage system by using multi-cloud providers, combining Byzantine quorum system protocols, cryptographic secret sharing an d erasure codes.

2.3. Comparative Study

The analysis of this research works revealed some limitations. Despite the effectiveness of federated clouds for resource scaling, cloud federations are unable to handle the composition and discovery issues due to the fact that they operate at the provider's (not service) level and focus solely on the IaaS cloud services' layer [20]. This problem can be countered by using a community deployment in the federation. For example, in [20] a trust framework is proposed to create a federation of cloud communities that supports, in addition to resource scaling, discovery, marketing and composition facilitation. However existing solutions and consist either in managing services or in building efficient federations, none of them takes into account the possibility of addressing both service management and federation creation at the same time.

Table 1 sums up some of the most relevant works related to cloud service management techniques according to the following criteria:

- **Environment**: it defines whether the management solution is design for (single-clouds) or (multi-clouds).
- **Service Type** it determines the type of the service the proposal deals with: Data as a Service (DaaS), Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) or (Mashup services).
- **Service Model:** it specifies whether the proposals use a formal representation (graph or any data structure) to represent the relationship between the services.

- **Management Method:** it indicates which method (Clustering), (Indexing), (Registry), (Ontology) or (Blockchain) is used for the service management.
- **Purpose:** it indicates the issues that the authors work on.
- We note that a cell containing (_) as value means that the information was unavailable in the study.

Table 1: Survey of works related to cloud service management techniques.

Ref	**Year**	**Environment**	**Service**		**Management Method**	**Purpose**
			Type	Model		
[39]	2019	single-cloud	DaaS	–	Blockchain	Management
[25]	2018	multi cloud	SaaS	No	Clustering	Discovery
[2]	2018	single-cloud	Mashup	Graph	Clustering	Discovery
[3]	2018	single-cloud	SaaS	Graph	Clustering, Indexing, Ontology.	Discovery, Publication
[4]	2017	single-cloud	SaaS	–	Clustering	Discovery
[9]	2016	single-cloud	IaaS, PaaS, SaaS.	–	Ontology	Management
[5]	2016	–	Mashup	Graph	Clustering	Discovery
[11]	2016	single-cloud	DaaS	–	multidimensional Indexing	Management
[10]	2016	multi cloud	Micro services	No	Ontology	Management
[6]	2016	single-cloud	DaaS	–	Clustering	Management, Processing.
[7]	2015	single-cloud	IaaS, PaaS, SaaS,	–	Clustering, Ontology	Discovery
[8]	2014	single-cloud	SaaS	–	Ontology, Registry	Recommendation
[15]	2010	single-cloud	DaaS	No	multidimensional Indexing	
[12]	2011	single-cloud	DaaS	No	multidimensional Indexing	Data Processing

Continue on next page

Table 1 – *Continued from previous page*

Ref	Year	Environment	Service		Management Method	Purpose
			Type	Model		
[13]	2011	single-cloud	DaaS	No	multidimensional Indexing	Data Management
[40]	2011	single-cloud	–	No	Ontology	Management
[38]	2010	single-cloud	DaaS	No	unidimensional Indexing	Data Processing
[14]	2010	single-cloud	DaaS	No	multidimensional Indexing	Data Storage

3. The Distributed SaaS Management System for A Cloud Federated Environment

In this section, we provide the details of our work including (i) formal definitions (ii) an overview of our approach, (iii) the proposed cloud federation architecture, (iv) the details of the proposed management system and (iiv) the service publication algorithm.

3.1. Formal Definitions

In this section, we provide the formal definitions of all the important concepts used in the frame of this work.

Definition 1. ***(WordNet domain ontology)*** *The WordNet domain ontology [28] is a formal classification of several concepts (domains and their sub-domains) realizing a hierarchical structure where each concept is characterized by its properties and its relations with other concepts. It is defined as* $\mathscr{O} = (\mathscr{D}, Sub_{\mathscr{D}}, \mathscr{F}, R, I)$ *where:*

- $\mathscr{D} = (\{d_j\}|1 \leq j \leq M)$*: the WordNet domain ontology defines five* $M = 5$ *general categories* $d_j \in \mathscr{D}$ *including* $d_1 =$ *Humanities,* $d_2 =$ *Free Time,* $d_3 =$ *Applied Science,* $d_4 =$ *Pure Science and* $d_5 =$ *Social Science. Each domain represents the root node of a tree structure.*
- $Sub_{\mathscr{D}} = (\{Sub_{d_j}\}|1 \leq j \leq M)$*: each domain* $d_j \in \mathscr{D}$ *provides a set of several sub-domains. For instance, the domain* $d_5 = SocialScience$ *provides some of the following sub domains* $\{Tourism, Finance, Transport, Industry\} \subset Sub_{d_5}$
- $\mathscr{F} = (\{f_i\}|i \geq 1)$*: functionalities are more abstracted concepts provided by the sub-domains of the ontology, each functionality provides several SaaS services. The functionality concepts represent the leaf nodes of the tree structure. For example, the set* $\{MoneyConversion\}$ *represents the functionality concepts of the sub-domain* $Finance \in Sub_{d_5}$*.*

- *R: is the set of relations that describe the interactions between concepts and organise them into sub-super-concept (domain, sub-domain, functionality) tree structures.*
- *I: is the set of individuals or instances. Instances are the "things" represented by a concept. For example, a cloud service S_1 is an instance of the concept Convert.*

Definition 2. ***(Service Description)*** *Each application (SaaS) service is described as a tuple $s = (s_{Id}, s_d, s_f, s_{In}, s_{Out}, s_B)$ where:*

- s_{Id}*: unique identifier of the service s.*
- $s_d \in \mathscr{D} \cup Sub_{\mathscr{D}}$*: domain of the service. The domain is chosen among the existing domains in $\mathscr{O}$.*
- s_f***:*** *functionality concept of s.*
- $s_{In} = (\{e_i\}|i \geq 1)$*: represents a set of service inputs (concepts). An input $e_i \in In$ is what a service s requires to produce expected outputs.*
- $s_{Out} = (\{o_j\}|j \geq 1)$***):*** *represents a set of service outputs (concepts). An output o_j is the result obtained from a set of inputs.*
- s_B*: represents the files constituting the body of the service.*

Definition 3. ***(Cloud Provider)*** *A Cloud provider C is defined by C = (Id, S), where:*

- *Id: is the unique identifier of the Cloud provider;*
- $S = (\{s_i\}|i \geq 1)$*: is the set of cloud SaaS services provided by C.*

For example, several cloud providers are illustrated in Fig. 5 such as $Sl_{1.1}$, $Sl_{j.2}$, $Sl_{5.(K-1)}$, L_1, L_5, CM_F.

Definition 4. ***(Cloud Community)*** *A cloud community represents a set of K cloud providers. It is defined as $com = (d, L, Sl)$ where:*

- $d \in \mathscr{D}$*: represents the domain of interest of the community.*
- *L: represents the cloud leader of the community.*
- $Sl = (\{sl_h\}|1 \leq h \leq K-1)$*: represents a set of (K-1) cloud Slaves of the community.*

An example of cloud community $com_1 = (d_1, L_1, Sl_1)$ is illustrated in Fig.5 where d = d_1 = Humanities, L = L_1, Sl = $Sl_1 = \{sl_{1.1}, sl_{1.2}, ..., sl_{1.(K-1)}\}$.

Definition 5. ***(Cloud Federation)*** *A cloud federation C_F is defined as $C_F = (COM_F, CM_F)$ where:*

- $COM_F = (\{com_j\}|1 \leq j \leq M)$*: represents a set of M communities.*
- CM_F*: is the cloud master and the administrator of the federation Fd.*

An example of a cloud federation $C_F = (COM_F, CM_F)$is depicted in Fig. 5 where $COM_F = \{com_1, com_2, com_3, com_4, com_5\}$ and $CM_F = C_M$

Definition 6. ***Trusted registry*** *A trusted registry is defined as* $TR = (\{tr_j\}|1 \leq j \leq M)$ *with* $tr_j = (d_j, com_j, L_j)$ *where:*

- $d_j \in \mathscr{D}$*: represents the domain of interest of the community* com_j
- com_j*: represents the* j^{th} *community.*
- L_j*: is the cloud leader of the* j^{th} *community.*

Definition 7. ***Community Index*** *It is defined as a tuple* $com_{index} = (d, Sub_d, F) \subset \mathscr{O}$ *where:*

- $d \in \mathscr{D}$*: represents the root node of the index and the domain of interest of the community.*
- $Sub_d \subset Sub_{\mathscr{D}}$*: represents a set of d's sub-domains.*

For example, Fig.8 illustrates a community index Com_{index_j} stored in the cloud leader L_j.

Definition 8. ***Functionality Registry*** *it is defined as* $FR = (\{fr_w\}|w \geq 1)$ *with* $fr_w = (f_w, Cl_w)$ *where:*

- $f_w \in F$*: represents a functionality (concept).*
- Cl_w*: represents the identifier of a service cluster which gathers functionally similar services.*

For example, in the community illustrated by Fig.8, a functionality registries for each of its cloud slaves is represented such as $FR_{j.1}$, $FR_{j.2}$ and $FR_{j.(K-1)}$

Definition 9. ***Service Clusters*** *A service cluster is define as* $Cl = (f, \{S_{Net_z}\}|z \geq 1)$ *where:*

- $f \in F$*: represents a functionality (concept).*
- S_{Net_z}*: represents a set of functionally similar services organized into a semantic network.*

Definition 10. ***Semantic Service networks*** *A service network is represented by a directed network graph denoted by* $S_{Net} = \{S, E\}$ *where:*

- *S: a set of node representing the SaaS services.*
- $E = (\{(s_i, s_j)\}|(s_i, s_j) \in S^2$ *; and ;* $s_i \neq s_j)$*: represents a set of semantic edges built using the input/output similarity.*

3.2. Approach overview

The main objective of our work is to build an efficient cloud federation environments and deploy a distributed management system in order to address the exponential growth of services, effectively use the storage space and furnish automated service publication and discovery methods.

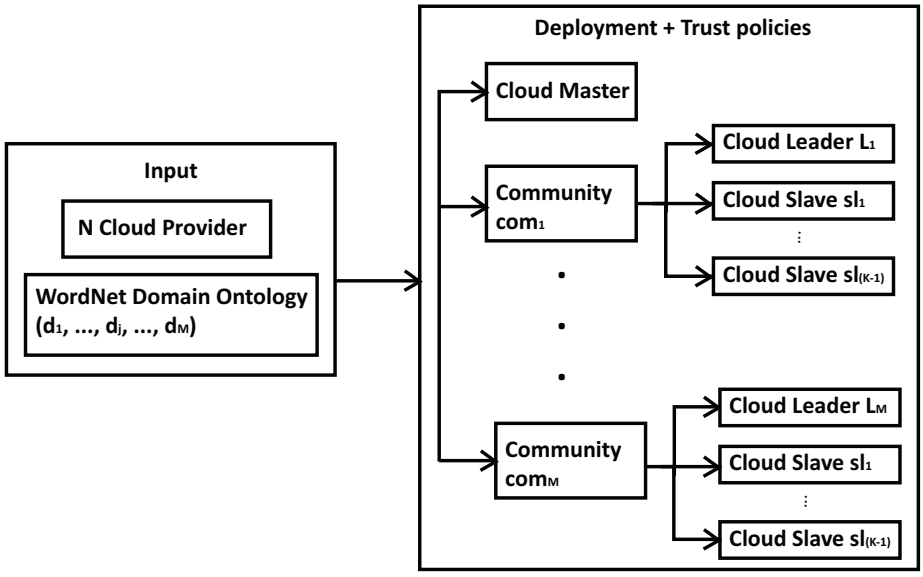

Fig. 2: Cloud Federation Creation Process.

To do so, we start by building the cloud federation as it is illustrated in Fig. 2. A set of N heterogeneous cloud providers will be deployed into M communities ($com_j|1 \leq j \leq M$) by using the domains ($d_j \in \mathscr{D}|1 \leq j \leq M$) offered by the WordNet domain ontology (each main domain d_j offers a set Sub_{d_j} of several sub-domains). Each community will store and host services that belong to the same domain of interest. For example, the community com_3 will offer services that belong to the domain domain d_3 = Applied Science and its sub-domains Sub_{d_3} = { Computer Science, Medicine, Agriculture ,...}. Afterwards, we propose to elect an administrator for each community whom we will appoint as cloud leader. The remaining clouds in the community will be considered as cloud slaves and they will be used for the storage of the cloud services. Thereafter, we propose to elect a cloud master from among the cloud leaders as the primary administrator of the federation. Then, trust policies will be established within and between the cloud communities to overcome interoperability issues between the heterogeneous cloud providers and form a federation with a single access point. As a result, we obtain an architecture organised in a hierarchy comprising a central administrator called the cloud master, several secondary administrators (the cloud leaders) and several cloud slaves who will be in charge

of storing the cloud services. Thus, user requests will first be received by the cloud master and then forwarded to the cloud leaders of the concerned communities, who will then broadcast the request to the relevant cloud slaves. The cloud master being the administrator of the entire federation maintains a special directory we named trusted registry (TR) which contains information about the involved cloud communities, their domain of activity, the services they offer as well as the id of the different cloud leaders. TR can be considered as a global overview of the offerings of the entire federation.

Once the federation is established, built and operates as a single organisation, we will proceed with the management of the existing services in each community. This work concerns the application layer services (SaaS) which are stored physically in the physical infrastructure (servers) of the federation and as virtualisation is one of the key features of cloud computing, these services will be managed virtually (build a virtual view) following the process illustrated in Fig. 3.

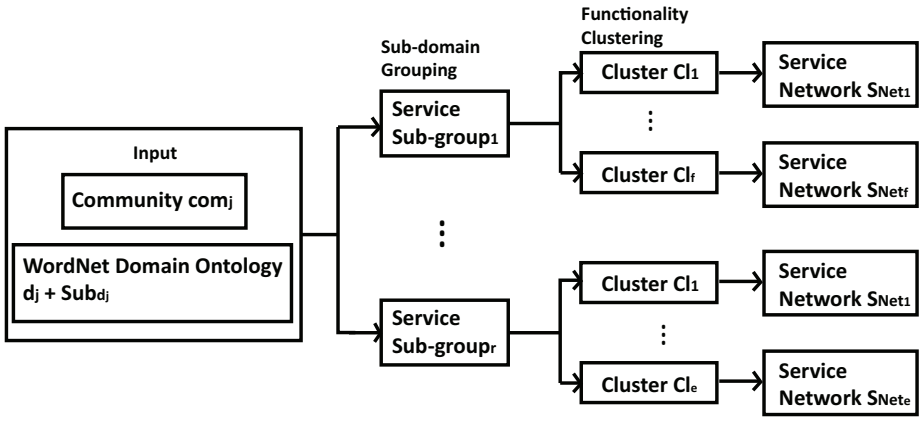

Fig. 3: Service Management Process.

Firstly, the services in each community will be classified into finer groups according to their sub-domain. For example, in the community com_3, services that belong to the *Medicine* sub-domain will be grouped separately from other services that belong to other sub-domains such as the *Computer Science* or *Agriculture*. Afterwards, we propose to further improve and refine the sub-domain grouping of services in each community by creating clusters of functionally similar services using the functionality attribute (concept) in the description of services. This step has several advantages as it allows to avoid grouping services into categories (sub-domains) only not to what they really offer, it also facilitate service discovery by reducing considerably the search space. Next, we made use of the relationships that services may have with each other and constructed a service network S_{Netz} for each cluster using the input/output service attributes. The post-creation of these networks is a considerable time saver in

the execution of the different algorithms. Also, this step enables handling complex user requests that cannot be satisfied with single services by constructing service compositions.

3.3. Cloud Federation Communities Architecture (CFedCom)

Fig. 4 depicts the proposed architecture in which we identify the major actors, layers, their activities and functions in the cloud federation. The main components of our architecture are the federation management layer, the service management layer, the cloud services layer and the physical infrastructure layer. While the main actors of this architecture are the end users, the service providers, the broker slaves, the broker leaders and the broker master. Each actor is an entity that performs tasks and interacts with the different components of the cloud federation. The components and layers are briefly explained in the following:

- *Federation management:* this layer is responsible for providing all the necessary methods for building and maintaining the cloud federation.
- *Service Management:* this layer includes all of the service-related methods and algorithms that are necessary for storing and managing the services of the cloud federation.
- *Physical Infrastructure*: this layer represents the physical resources of the federation (servers and data centers).
- *End User:* this entity represents the cloud client (company or individuals) that requests the use of cloud services.
- *Service Providers:* this entity is responsible for developing and submitting services within the federation.
- *Brokers:* a broker is an entity associated to every cloud in the federation. It is responsible for handling the interactions between the different cloud providers.

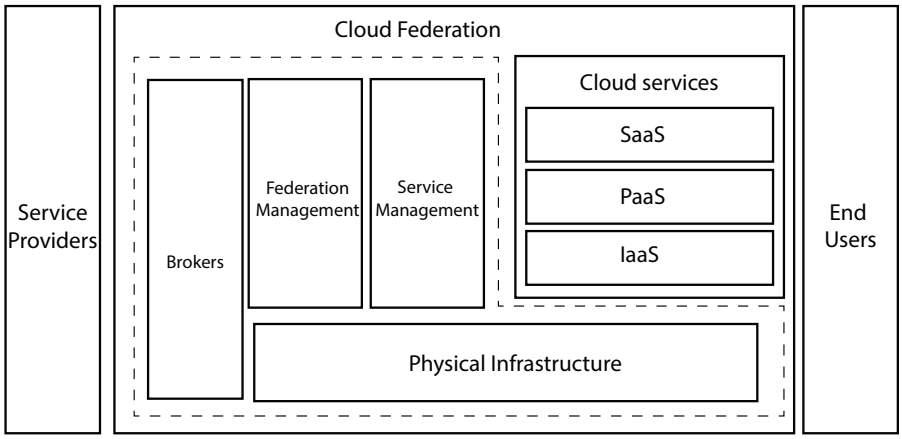

Fig. 4: Proposed Cloud Federation Architecture.

3.3.1. Cloud Federation Creation

The proposed cloud federation is built by using Algorithm 1. It takes as input a set of cloud providers as well as the WordNet domain ontology $\mathscr{O}$. It provides as output the hierarchical architecture illustrated in Fig. 5 composed of a cloud master and a set of cloud communities with a cloud leader and cloud slaves for each. The steps of the algorithm are explained as follows:

- **Step 1: Community Deployment**
  Algorithm 1 starts by grouping a set of (N) clouds (C = { C_1, C_2,..., C_N }) into (M) communities (Com = $\{com_1, com_2, ..., com_M\}$) according to the WordNet domain ontology [28]. The WordNet domain ontology $\mathscr{O}$ enables the definition of a simple vocabulary in terms of concepts and the relationships between them, including classes, sub-classes, etc. It provides a hierarchical semantic classification of existing domains into five general categories (d_1 = Humanities, d_2 = Free Time, d_3 = Applied Science, d_4 = Pure Science, and d_5 = Social Science) where each domain d_j includes several sub-domains; and the advantage of a cloud community lies in the ability to offer optimized cloud solutions to specific category of users, and when cloud communities are used, user requirements can be better satisfied [29]. As a result, we obtain five communities (M = 5) gathering K clouds each and sharing heterogeneous domains of interest (each domain d_j is assigned to a community com_j). The communities provision and the store cloud services that are related to the domain d_j and its sub-domains.

- **Step2: Cloud Leader Election**
  This step consists in electing for each community com_j a cloud leader L_j among its K constituting clouds (Line 3 - Line 7) based on the hardware criteria in order to facilitate its administration and avoid some chal-

Algorithm 1 Cloud federation creation

Inputs:

$\mathbf{C} = \{C_1, C_2, ..., C_N\}$//set of (N) clouds

$\mathscr{O} = \{(d_1, Sub_{d_1}), (d_2, Sub_{d_2}), ..., (d_M, Sub_{d_M}), \}$ //WordNet domain ontology with M principle domains and their sub-domains

Outputs:

CM_F, COM //The cloud master and the set of cloud communities

BEGIN

1: Set_Leaders = $\emptyset$; $COM = \emptyset$;
//Step 1
2: COM = Community_Deployment (C, $\mathscr{O}$);
//Step 2
3: **for each** com_j in COM / j = 1 to M **do**
4: $com_j.L_j$ = Cloud_Leaders_Election (COM_j);
5: Set_Leaders = Set_Leaders + $com_j.L_j$
6: $com_j.sl$ = Slaves_Designation (L_j, com_j);
7: **end for each**
//Step 3
8: CM_F = Cloud_Master_Election(Set_Leaders);
//Step 4
9: Trust_Establishment(CM_F, COM);

END

lenges such as a mesh network communication between the different cloud providers. As a result, we obtain for each community com_j : (i) a central administrator called cloud leader L_j whose hardware capabilities are the highest in the community, (ii) $K-1$ remaining clouds considered as cloud slaves; they are responsible for storing and hosting the cloud services, and (iii) a broker for each cloud in com_j. The broker of a cloud leader is named broker leader, and it interacts with the brokers of the cloud slaves upon receipt of a request for service publication or discovery.

- **Step 3: Cloud Master Election**
  A cloud master CM_F is elected among the M cloud Leaders as the main administrator of the federation based on the hardware criteria (Line 8). As a result, service publication and discovery requests are first received by CM_F then forwarded to the cloud leaders of the concerned communities by interacting through their respective brokers. To enable the proper transfer of requests, the cloud master uses and maintains a directory referred to as the trusted registry TR where all the cloud leader identifiers and their application domain are listed.

- **Step 4: Trust Establishment**
  The concluding stage in the construction of the federation is the trust establishment between the clouds of the federation. In fact, interoperability

issues arises from the use of the heterogeneous N cloud providers in the creation process of the federation. Therefore, the establishment of trust policies inside and across the communities resolves this issue and enables the use of the federation as a single organization.

As a result, we obtain the hierarchical federation architecture illustrated in Fig. 5 composed of a central administrator called cloud master CM_F, M cloud communities, M cloud leaders and several cloud slaves.

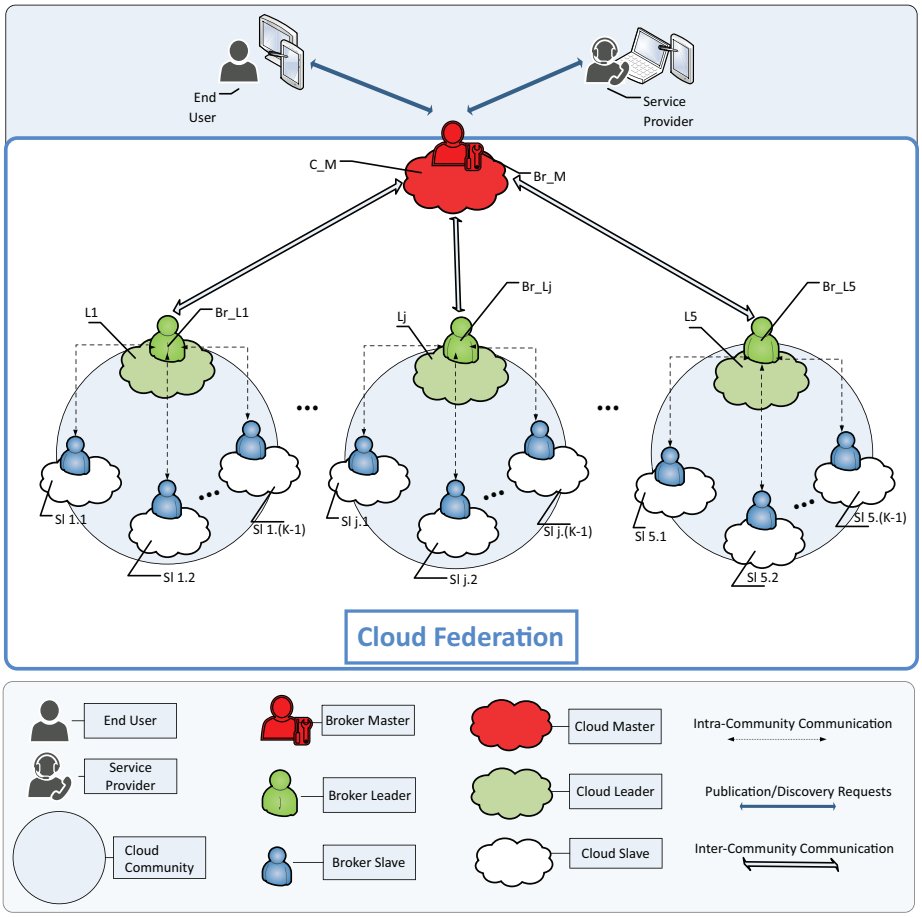

Fig. 5: Proposed Cloud Federation Framework.

3.4. Distributed SaaS Management System

SaaS services are software applications that can provide either one or more functionalities to cloud users. They are described by their functional or non functional attributes such as the domain, functionality, input and output; as well

their body which represents the several coded files which will be published and deployed into the cloud physical infrastructure. In order to effectively manage these software services, two primary aspects regarding the nature of the cloud must be considered managing software services:

- **Standardization**
  One of the main challenges of the cloud environment is the lack of standards notably, regarding the cloud Service Description Languages (SDLs). Existing cloud SDLs differ from one cloud provider to another. As a result, several descriptions are found such as Blueprint [41], Unified Service Description Language (USDL) [42], Web Service Description Language (WSDL), Web Ontology Language (OWL), Semantic Open API Specification (SOAS) [43], Cloud Service Description Model (CSDM) [44]. The fact that cloud SDLs being not standardized makes the service management process more challenging. In this paper, we propose a unified service description (Definition 2) which includes the necessary functional attributes to successfully carry out the management and the publication of services and ensure accurate discovery results.

- **Storage Virtualization**
  With cloud storage, users access a virtual static storage instead of a physical storage space [29] and the actual physical storage location may change as the cloud dynamically manages available storage space. In fact, unlike traditional systems that provide only one physical storage layer, the cloud computing provides a virtual layer created on top of the physical layer by using virtualization technologies as virtualization being one of the key characteristics in cloud computing systems [45].

In consequence, SaaS services are physically stored and virtually managed using their description according to the process illustrated in Fig. 6 which we explain in the following:

- **Step 1: Sub-Domain Grouping and Community Index Building**
  The first step of the SaaS management process consists in improving the domain grouping inside each community. In fact, SaaS services are already grouped into M general categories according to their domain of interest d_j and affected to a community com_j. For example, com_3 involves services that belong to the same domain of interest i.e. $d_3 =$ Applied Science and to different sub-domains such as Medicine, Agriculture or Computer Science. Thus, by using the WordNet domain ontology [28], services in each community are grouped into finer categories according to their sub-domain. Next, we propose to build a community index from the WordNet ontology in order to be able to associate each service group to its corresponding sub-domain. A community index is defined by $com_{index_j} = \{d_j, Sub_{d_j}, F\}$ where d_j represents the root of the index and the domain of interest of the community com_j, Sub_{d_j} represents the set of d_j's sub-domains. The community index represents a local overview of all the services contained in the community com_j.

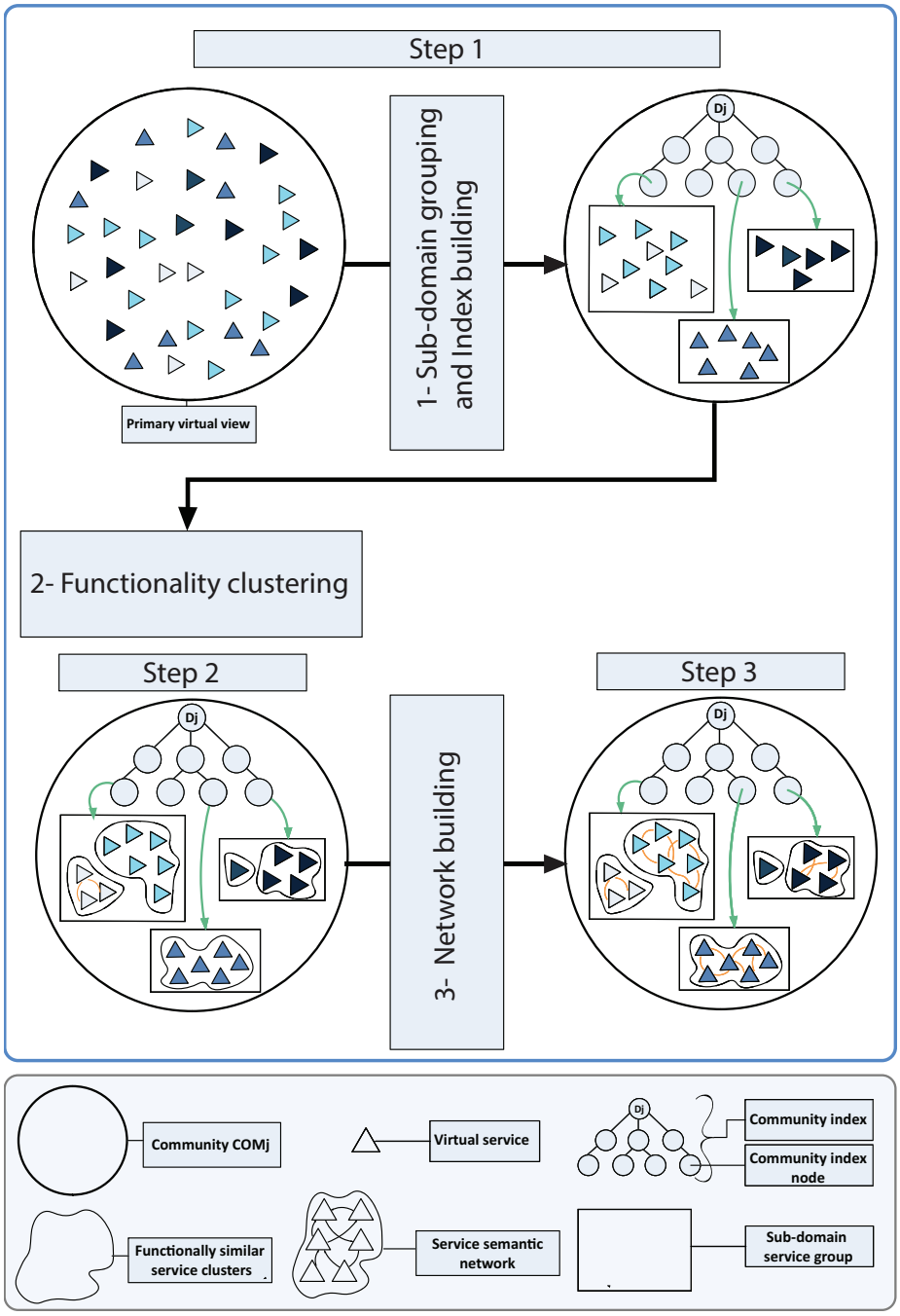

Fig. 6: Virtual Service Management Process.

- **Step 2: Service Clustering**
  The second step in the management process consists in clustering the services in each sub-domain into functionally similar service groups using the functionality attribute in the service description (Fig. 6 step 2). By using clustering methods, similar services can be searched and discovered together to improve both the efficiency and accuracy of service discovery [5] and enhance their visibility towards users [20]. As a result, we obtain for each sub-domain several clusters described by a main functionality f and several functionally similar SaaS services. To facilitate the management of the clusters, we propose to list them in directories called registries of functionalities (see Fig. 7) where each entry links a corresponding cluster.

- **Step 3: Service Networks Construction**
  The third step in the management process consists of exploiting the input/output similarities between services to create service networks (Fig. 6 step 3). Forming a network of services allows to process complex user requests that require multiple functionalities which can't be satisfied by an atomic (single) service. Thus, enabling service composition. Each service network is represented by a directed network graph $S_{Net} = \{S, E\}$ representing the input/output relationships between the functionally similar services, where each edge $(s_i,s_j) \in E$ indicates an input/output similarity between the service (s_i) and the service (s_j).

Therefore, applying the previous three steps on com_3 for example produces the result depicted in the Fig. 7. On top of the hierarchy, the community index com_{index_3} built from the WordNet domain ontology. At the center, the leaf nodes offer several functionalities listed in a directory. At the bottom of the hierarchy, the service networks organized into clusters. This management hierarchy represents the virtual view of the all the SaaS services contained in the community. It is also maintained and updated in case any new submitted services or any additional domains in the WordNet ontology. Thus, we apply a last management step which allows to distribute the virtual view on the different clouds of the community for maintenance efficiency and management facility.

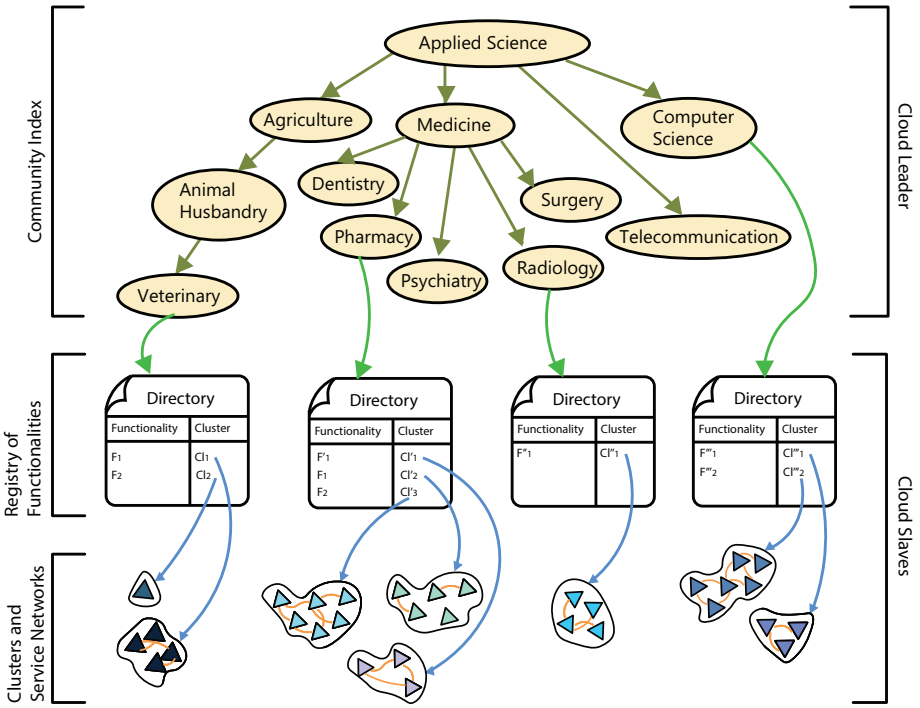

Fig. 7: Service Management Hierarchy.

- **Step 4: Virtual View Distribution**
  In order to optimize the management and facilitate the maintenance, we propose to distribute the virtual view of each community (com_j) between its clouds (leader and slaves) as illustrated in Fig. 8. The leader (L_j) being the administrator of the community (com_j) is charged with the maintenance of the community index. Then, we distribute the directories and the functionally similar clusters between the slaves. The links (index nodes/directories) and (functionality/clusters) are preserved.

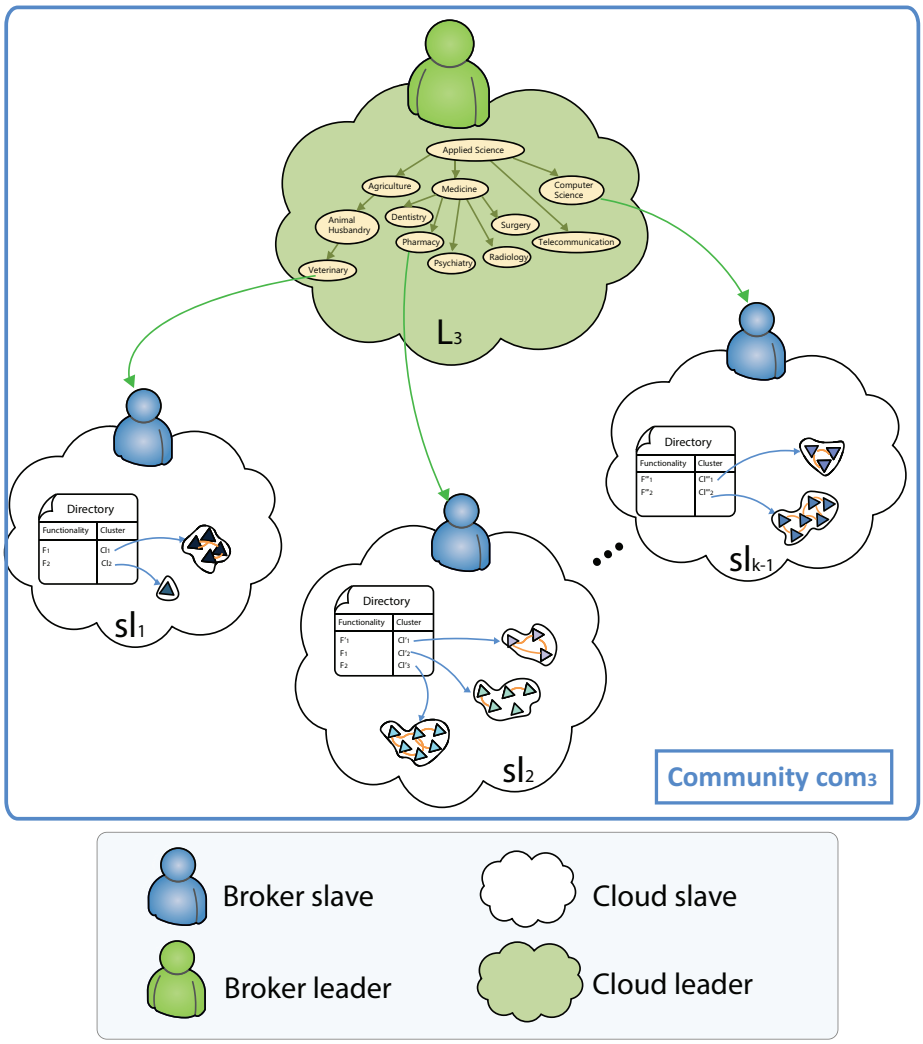

Fig. 8: Distributed Virtual View.

3.5. Service Publication

SaaS services are submitted by service providers then published and stored within the cloud infrastructure. Before proceeding to the publication of any cloud service, it is crucial to determine how services are described and what information should be provided. The challenge that can occur lies in the fact that there is no unified description for cloud services. In fact, the description of cloud services differs from one cloud provider to another due to the lack of standards. As a result, several descriptions are found in the literature such as Blueprint [41], USDL (Unified Service Description Language) [42], WSDL (Web Service Description Language), OWL, SOAS (Semantic Open API Specification) [43], CSDM (Cloud Service Description Model) [44]. These descriptions do not fulfill our need, this is why we propose in this paper a new service description that involves several attributes (domain, functionality, inputs, outputs, body) as explained in definition 2. In the following section we therefore present a detailed publication process while respecting the proposed management solution and the service description.

3.5.1. Service publication Process

The publication process of a service (s_i) within the cloud environment is composed of two essential parts, the virtual storage (step 1, 2, 3) and the physical storage (step 4). The virtual storage consists in adding the new service to the virtual view. Whereas the physical storage is about storing the body of the service in the physical servers of the Cloud. This process is resumed through Algorithm 2 that takes as input the service to submit and provides as output an update in both the virtual and physical views.

- **Step 1:** Service provider (SP_i) → Broker Master (M_Br)
  - The (SP_i) submits its Publication Request PR = { S_i_Domain, S_i_Functionality, S_i_Inputs, S_i_Outputs, S_i_Body } via the Cloud portal (Line 1).
- **Step 2:** Broker Master (M_Br) → Broker Leader (Br_Lj)
  - The Broker Master receives the Publication Request (PR) .
  - Using the domain (S_i_Domain) of the service (S_i), the Broker Master (M_Br) consults its Trust Registry (TR) to determine the Broker Leader (Br_Lj) of the community (COMj) concerned with a domain similar to S_i_*Domain* (Line 5 and 6) .
  - The Broker Master (M_Br) forwards the Publication Request (PR) to (Br_Lj) (Line 7).
- **Step 3:** Broker Leader (Br_Lj) → Broker Slave ($Br\_Sl_{j.h}$)
  - The Broker Leader (Br_Lj) searches its Community Index (COM_Index_j) to find the set of functionalities (Set_F) offered by the domain (D), where (D) is similar to the submitted domain (S_i_*Domain*) (Line 9).

- Using the set of functionalities (Set_F), the Broker Leader (Br_Lj) determines the Cloud Slave (Br_Sl$_{j.h}$) which stores the services that are concerned with the functionality (F), where (F) is similar to (S_i_$Functionality$) (Line 10 and 11).
  - If no similar functionality (F) is found, the new functionality is then added to (Set_F), and the Broker Leader (Br_Lj) determines one of its slaves ($Br\_Sl_{j.h}$) that will store the new service (Line 13).
  - A request is sent to the Cloud Slave ($Sl_{j.h}$) to add the new functionality (S_i_$Functionality$) in its registry (Line 14).
- The Broker Leader (Br_Lj) forwards the publication request to the Broker of the concerned Cloud Dlave (Br_Slj_h) (Line 16).

- **Step 4:** Broker Slave (E.Sl.Br_Sl)
  - Using the functionality (S_i_$Functionality$) of the submitted service, the Broker Slave (Br_Sl$_{j.h}$) searches its registry of functionalities (Reg_F$_{j.h}$) in order to determine the cluster (CL$_j.h.q$) indexed by the functionality (F), where F is similar to (S_i_$Functionality$) (Line 18).
  - (Br_Sl$_{j.h}$) adds the new service (S$_i$) to the semantic network contained in the cluster (CL$_j.h.q$) using the input/output similarity (Line 19).

- **Step 5:** Broker Slave (Br_Sl$_{j.h}$)
  - (Br_Sl$_{j.h}$) Stores the body (S$_i$_Body) of the submitted service in the physical infrastructure (Line 20).

Algorithm 2 SaaS Publication Algorithm

Inputs:
$PR = \{$ S_i_Domain, S_i_Functionality, S_i_Inputs, S_i_Outputs, S_i_Body $\}$
TR $= \{(COM_1, D_1, Br\_L_1), ..., (COM_5, D_5, Br\_L_5)\}$
$\mathscr{O} = \{D_1, D_2, ..., D_5\}$ //the WordNet domain ontology where $D_j = \{Dj\_Sub_1, Dj\_Sub_2, Dj\_Sub_3, ...\}$ j^{th} domain and its sub domains.
Outputs:
E = (COM, Br_L, F, Sl, Cl)

1: **BEGIN**
2: **SPi:** //Step 1
3: Send_Publication_Request(PR) $\longrightarrow$ M_Br;
4: **M_Br:** //Step 2
5: E.COM=Find_Corresponding_Community(TR, PR.S_i_Domain, $\mathscr{O}$)
6: E.Br_L = Find_Corresponding_Leader(E.COM)
7: Forward_PublicationRequest(PR) $\longrightarrow$ E.Br_L
8: **E.Br_L:** //Step 3
9: Set_F = Find_Set_of_Functionalities(S_i_Domain, E.Br_L.CommunityIndex)
10: E.F=Find_Corresponding_Functionality(Set_F, PR.S_i_Functionality);
11: E.Sl=Find_Corresponding_Slave(E.F);
12: **if** (E.Sl == NULL) **then**
13: E.Sl = Determine_CloudSlave(E.COM);
14: Cearte_New_Functionality(E.Sl.Reg_F, PR.S_i_Functionality);
15: **end if**
16: Forward_PublicationRequest(PR) $\longrightarrow$ E.Sl;
17: **E.Sl:** //Step 4
18: E.Cl = Search_Registry_Functionalities_Clusters(E.Sl.Reg_F, PR.S_i_Functionality);
19: Add_NewService_toNetwork(E.Cl.Network, PR);
20: Add_NewService_toPhysical_Databases (S_i_*Body*);//Step 5
21: **END**

4. Experiments and Results

The proposed approach is implemented and a set of experiments are performed in order to: (i) evaluate the performances of the proposed cloud federation architecture (CFedCom) against a non federated environment; (ii) compare our approach in terms of response time to the work proposed in [46]; and (iii) study the amount of the storage space that the proposed management solution requires.

4.1. Experimental setup

The experiments were conducted on an Intel (R) core (TM) i5-8250U CPU 1.80 GHz equiped with 12 Go RAM, running on windows 10. The management and the publication prototype was developed using the C# programming language. Whereas the simulations were undertaken using Cloudsim[1] 3.0.3.

Regarding the services dataset, we have used the OWLSTC[2] v4.0 open source corpus. This dataset is designed for web services, but we chose to use it because, on the one hand, there is no dedicated SaaS services dataset available for the Cloud Computing, and on the other hand, according to H. Nacer et al. [47] a SaaS service is a Web application offered as a service on demand by using Cloud infrastructure. Hence the possibility of its use. We have also used the domains of the WordNet ontology [28].

4.2. Experiment 1: CFedCom Vs Non Federated Environment

We aim through this first experiment to evaluate the performances of the proposed solution i.e. the federation of several cloud communities (CFedCom) compared a single cloud community gathering several cloud providers. Table 2 lists the simulation configuration used in this evaluation.

			VM				
	N	M	Mips	Size	Ram	BandW	CPU
Single community	4	1	1000	10000	512	1000	1
CFedCom	7	3	1000	10000	512	1000	1

Table 2: Virtual Machine Configuration.

We have varied the number of services (cloudlets in cloudsim) between 100 to 50 000. Table 3 lists the results of the obtained response time.

From the results listed in Table 3, the multiple cloud communities offer better response comparing to single cloud communities. Moreover, the advantage of multiple communities federation is in the storage space and the variety of the

[1] https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3
[2] http://projects.semwebcentral.org/projects/owls-tc/

Number of Services	100	500	5000	50 000
single cloud provider (s)	40	200	2000	20000
Single cloud Community (s)	8	40	400	4 000
CFedCom	2. 67	13.34	133.34	1333.34

Table 3: Services Run Time for the Three Scenarios.

services offered by the federation is wider. There is no interoperability problem for this architecture thanks to the trust relationship established between the Clouds and the communities. The advantage of a Cloud community lies in the ability to offer a larger variety of services. We can say that Cloud community implementation offers a better run time than the single-Cloud implementation. It also allows clients to use the best Cloud services from different Cloud providers while minimizing resources. Moreover, a greater number of services can stored using the Cloud community architecture. Yet, this architecture presents some major issues. Firstly, the Clouds constituting the community present a significant concern which is illustrated in the heterogeneity of the Cloud architecture design. This issue creates an interoperability problem between the services offered by the several Clouds. Secondly, the community can only satisfy a specific group of clients because it is dedicated to a unique domain of offerings. Therefore, in order to solve the aforementioned issues, we opt to implement the same management mechanism for all the Clouds of the community and establish trust between the (N) Clouds (intra-community). This will allow the (N) Clouds to be managed as a single organization. Thus, solve the heterogeneity issue. Furthermore, we opt to extend the single community architecture to multiple Cloud communities in order to provide services to as many clients as possible, ensure a better quality of service and further increase storage space.

The use of the federation (CFedCom) reduces the average response time by more than 60% comparing to single-Cloud environments and more than 90% comparing to a single Cloud community federation (see Table 4). As a result, federated Cloud environment provide better performances, wider resources and services.

Improvement : Single Cloud Provider Vs CFedCom	93%
Improvement : Non Federated Environment Vs CFedCom	66%

Table 4: Cloud Federation Improvement Percentage.

4.3. Experiment 2: CFedCom Vs [46]

In this section, we analyze the performance of the proposed cloud federation in terms of response time against the cloud federation proposed in [46]. The response time taken by the two approaches are illustrated in Fig. 9. In this simulation, 500 requests are processed in the both cloud federation architectures. Our approach presented better average response time compared to [46]. This is due to the fact that in our solution, service providers are located directly thanks

to the proposed hierarchical management solution. Whereas in [46], they search each time for the corresponding cloud providers by using ranking algorithms. We note that these experiments are made using the cloudsim simulator which enables the creation of almost a perfect environment, where lots of real world challenges such as server failure or breakdown are not considered. Thus, different performance results may be obtained with a real world evaluation.

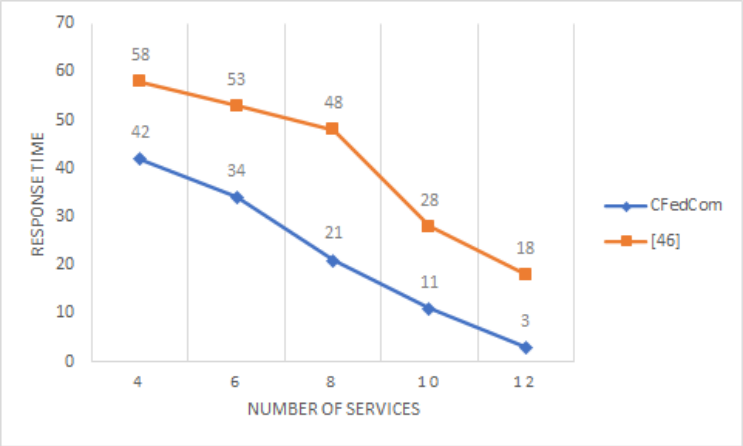

Fig. 9: Average response time of the two approaches.

4.4. Experiment 3: Storage Requirement

In these experiments, we aim to determine which of cloud master, leaders or slaves need the most storage space. Lets consider the scenario described in Table 5, a cloud federation that constituted of a cloud master and three communities (com_1, com_2, com_3) where each community gathers one cloud leader and two cloud slaves (three clouds per community). Each community is associated to a domain (the available domains of WordNet [28]). The community com_1) is associated with the domain (Social Science). The community com_2 is associated with the domain (Applied Science). The community com_3 is associated with the domain (Humanities).

Nb Clouds (N)	Nb Communities (M)	Nb Master	Nb Leaders	Nb Slaves
10	3	1	3	6

Table 5: Cloud Federation.

The cloud master (Br_M) holds and maintains the trusted registry, It enables discovery and publication requests to be forwarded to the corresponding communities based on the domain of interest of the request.

The Cloud Leader (L_j) of a community (com_j) holds and maintains a community index (COM_Index_j) build from the WordNet domain ontology [28].

Each leaf node of the Community Index (COM_{Index_j}) offers several functionalities. Thus, (COM_Index_j) is used by the cloud Leader (L_j) in order to determine in which slave, the services are stored according to the required functionality

The services stored in each Cloud Slave ($Sl_{j.h}$) are grouped into functionally similar clusters. Thus, each Broker Slave ($Br\_Sl_{j.h}$) holds and manages a registry of functionalities ($Reg\_F_{j.h}$) that indicates in which cluster ($Cl_j.h.q$) services belonging to a functionality (F_x) are stored.

4.4.1. Virtual Storage Requirement

First, the cloud master provides two types of resources: computing resources, which are used to process publication/discovery requests, and storage resources, which are allocated to the storage of the registry (TR, see Fig. **??**). According to our scenario, a high level of availability is achieved with only with 90 bytes of storage capacity (30 bytes for each new entry = Community ID/ Domain/ Broker Leader IP Address, see Fig. **??**). High availability is maintained even with a larger number of communities. For example, even with 1000 communities composing the federation, the storage space required is only 31 kB. The storage space (S_1) required by the Trusted Registry at any particular time is given by the Equation 1, where (M) represents the number of communities composing the federation; and (z) represents the required storage space for one entry in (TR).

$$S_1 = M * z \tag{1}$$

Second, each Cloud Leader provides virtual storage space for its Community Index (see Fig. **??**) as well as for the set of functionalities offered by the index leaf nodes. According to the WordNet domain ontology [28], the largest number of nodes an index can have is 35 nodes (this number may increase or decrease for any eventual updates applied to WordNet). Thus, the minimum storage capacity needed for an Community Index is 1,1 kB (30 bytes per node). Now, regarding the set of functionalities indexed by each leaf node, we obtain 30 bytes for each entry (entry = Functionality/ID Cloud Slave, see Fig. **??**). Thus, according to our simulation, we obtain 90 bytes for the set of functionalities indexed by the (Tourism) node. The storage space required for the set of functionalities increases as more functionalities are added but it is still manageable. For example, with 2000 additional functionalities, the required storage space is only 60 kB. Thus, the storage space that a Cloud Leader requires is given by the formula 2, where (Nb_Index_Nodes): represents the number of nodes of the Index Community; (s): denotes the required storage space for one index node; (s_i): represents the required storage space for the i^{th} set of functionalities indexed by the i^{th} leaf index node; and (q): represents the number of leaf nodes in the index.

$$S_2 = Nb\_Index\_Nodes * s + \sum_{i=1}^{q} s_i \tag{2}$$

Finally, each Cloud Slave of the federation provides virtual storage for its registry of functionalities and its service graph networks. The required stor-

age space per Cloud Slave is given by the formula 3, where (F): represents the number of functionalities offered by the registry of functionalities; (r): represents the required storage space for one entry in the registry of functionalities; (Nb_Networks_Nodes) denotes the number of nodes of all the networks saved within the Cloud Slave; (y): represents the required storage space for a single network node.

$$S_3 = F * r + Nb\_Networks\_Nodes * y \tag{3}$$

Fig. 10 illustrates the required storage capacity in the federation using the formulas above. Fixing the number of communities to 3, we notice that it's the Cloud Slaves that require the most virtual storage space.

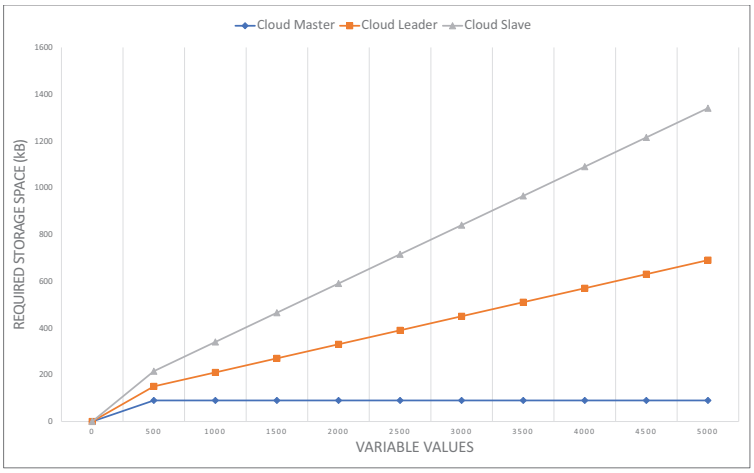

Fig. 10: Storage Space Requirement.

4.4.2. Physical Storage Requirement

The files constituting the submitted services are stored in the Cloud Slaves. We can deduce that slaves are the Clouds that require the highest level of storage resources.

4.5. A Guided Publication Process

Table 6 lists a sample of 10 SaaS services from the (Travel) domain which corresponds to the (Tourism) domain in WordNet [28]. We suppose that these services belong to the same functionality. Consequently, they belong to the same cluster.

Id	Inputs	Outputs	Description
1	Town	Activity	returns the provided activities of a given town.
2	Activity	City	returns names of cities which provide a given activity.
3	City	Accommodation	returns information about accommodations in a given city.
4	City	Hotel	returns information of available hotels of a given city.
5	City + Country	Hotel	returns names of available hotels in a city. The city must be located in the given country.
6	Surfing + Hiking + City	Beach	returns the name of the beaches where both hiking and surfing can be found for a giving city.
7	Beach	Sport	returns the available sports for a given beach
8	Sport	Beach	returns the available beaches for a given sport
9	Hotel	Activities	returns the activities offered by a given hotel
10	Village	Hotel	This service is used to know about list of hotels situated in a given village.

Table 6: OWLS-TC v4.0 Service Dataset

Each service is submitted via a cloud portal. For each new service, a Publication Request (PR) is created then received by the broker master then forwarded by this latter to the concerned community using the domain of interest of PR.

The ten submitted services belong to the same cluster since they have the same functionality. Consequently, the semantic network illustrated in Fig. 11 is built. For example, the output of the service identified by (Id = 2) is equivalent to the input of the service identified by (Id = 3), so an edge connecting the two services is created. Nevertheless, the service identified by (Id = 10) is added

independently to the network because no matching results were found for its output. All submitted SaaS services were added and stored successfully.

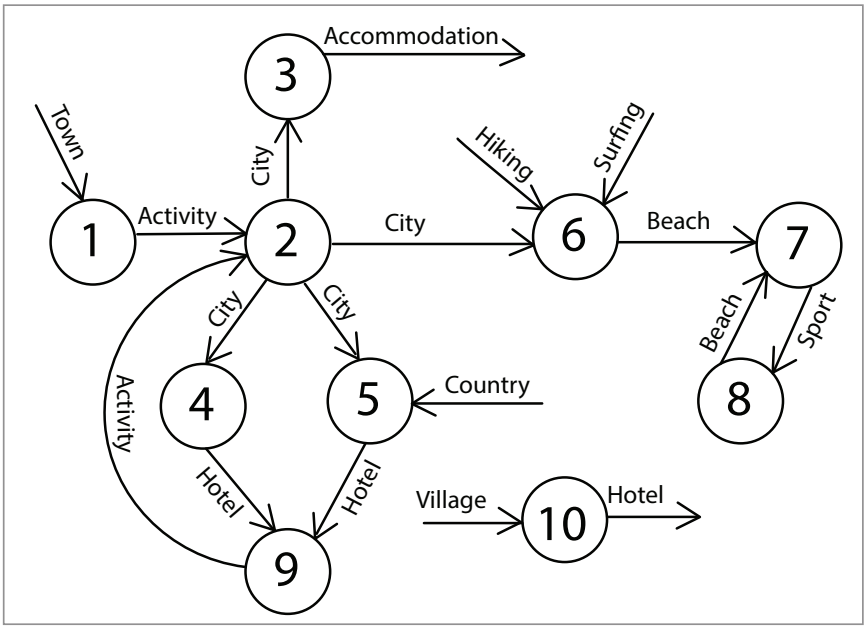

Fig. 11: Semantic Network Graph.

Organizing the services into a connected semantic network (directed graph) facilitates the discovery and the composition of services, specially when a user request is complex. In fact, according to H. Nacer et al. [48] the matching between the service discovery request and the available services gives eight cases, where 3/5 cases are satisfied by composition.

4.5.1. Algorithm Complexity

The Trusted registry, the Community indexes as well as the service networks are built, maintained and stored virtually. This reduces considerably the run time of the algorithm. Otherwise, if we confider for example the semantics networks, if they are built each time a publication request is submitted, the function will run in $O(V+E)+O(V^2)$, such that $O(V^2)$ is the time consumed for the construction of the semantic networks, and $O(V+E)$ is the time complexity of its traversal in order to add the new service. It depends on the data structure used to represent the graph. If it is an adjacency matrix, it will be $O(V^2)$, and if it is an adjacency list, it will be $O(V+E)$. In our case we have used the adjacency list. Thus, we can say that the preliminary construction of the semantic networks offers a real opportunity to reduce time response by a factor of $O(V^2)$.

5. Conclusion

The amount and variety of the services offered by the Cloud Computing are evolving every day, which poses a major challenge for their management. In fact, providing automated management solutions is crucial in order to meet some businesses needs such as providing enough storage space, effectively answer users requests, grantee quality of service. Therefore, we have proposed at first, a detailed Cloud federation architecture which involves several Cloud communities. It is built using the trust concept which overcomes the existing heterogeneity between the Clouds, while also using and semantic Web concepts such as the WordNet domain ontology in order to designate the domain of interest of each community. Second, we have proposed a management system that groups services into functionally similar clusters then build a semantic network for each cluster in order to address complex discovery requests and enable service composition. Finally, we have proposed a detailed publication algorithm according to the proposed architecture and management system. Several simulations and a service publication prototype has been developed as a proof of concept. We noticed that federated environments offer better results than single-Cloud environments or non federated environments. Moreover, the preliminary built of the virtual views reduces the processing time. However, it is essential to ensure that enough resources are available to maintain these virtual views. In our future work, we will focus on developing a detailed discovery and composition algorithm to complete the proposed solution according to the proposed federated environment.

References

[1] S. Dhuria, A. Gupta, R. Singla, Pricing mechanisms for fair bills and profitable revenue share in cloud federation, in: Advances in Communication and Computational Technology, Springer, 2021, pp. 133–148.

[2] W. Pan, C. Chai, Structure-aware mashup service clustering for cloud-based internet of things using genetic algorithm based clustering algorithm, Future Generation Computer Systems 87 (2018) 267–277.

[3] L. Ouchaou, H. Nacer, H. Slimani, S. Boukria, Semantic networks based approach for saas management in cloud computing, in: Proceedings of the International Conference on Smart Communications in Network Technologies (SaCoNeT), IEEE, 2018, pp. 255–260.

[4] K. B. Bey, H. Nacer, M. E. Y. Boudaren, F. Benhammadi, A novel clustering-based approach for saas services discovery in cloud environment., in: Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS), Springer, 2017, pp. 546–553.

[5] B. Cao, X. Liu, B. Li, J. Liu, M. Tang, T. Zhang, M. Shi, Mashup service clustering based on an integration of service content and network via

exploiting a two-level topic model, in: Proceedings of the International Conference on Web Services (ICWS), IEEE, 2016, pp. 212–219.

[6] L. Wang, Y. Ma, J. Yan, V. Chang, A. Y. Zomaya, pipscloud: High performance cloud computing for remote sensing big data management and processing, Future Generation Computer Systems 78 (2018) 353–368.

[7] A. Alfazi, Q. Z. Sheng, Y. Qin, T. H. Noor, Ontology-based automatic cloud service categorization for enhancing cloud service discovery, in: Proceedings of the 19th International Enterprise Distributed Object Computing Conference (EDOC), IEEE, 2015, pp. 151–158.

[8] Y. M. Afify, I. F. Moawad, N. L. Badr, M. F. Tolba, Concept recommendation system for cloud services advertisement, in: Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications (AMLTA), Springer, 2014, pp. 57–66.

[9] A. Gupta, S. Mittal, K. P. Joshi, C. Pearce, A. Joshi, Streamlining management of multiple cloud services, in: Proceedings of the 9th International Conference on Cloud Computing (CLOUD), IEEE, 2016, pp. 481–488.

[10] G. Sousa, W. Rudametkin, L. Duchien, Automated setup of multi-cloud environments for microservices applications, in: Proceedings of the 9th International Conference on Cloud Computing (CLOUD), IEEE, 2016, pp. 327–334.

[11] J. He, Y. Wu, Y. Dong, Y. Zhang, W. Zhou, Dynamic multidimensional index for large-scale cloud data, Journal of Cloud Computing 5 (1) (2016) 10.

[12] A. Papadopoulos, D. Katsaros, A-tree: Distributed indexing of multidimensional data for cloud computing environments, in: Proceedings of the 3rd International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2011, pp. 407–414.

[13] L. Ding, B. Qiao, G. Wang, C. Chen, An efficient quad-tree based index structure for cloud data management, in: Proceedings of the International Conference on Web-Age Information Management (WAIM), Springer, 2011, pp. 238–250.

[14] J. Wang, S. Wu, H. Gao, J. Li, B. C. Ooi, Indexing multi-dimensional data in a cloud system, in: Proceedings of the International Conference on Management of data (MOD), ACM, 2010, pp. 591–602.

[15] C.-L. Cheng, C.-J. Sun, X.-L. Xu, D.-Y. Zhang, A multi-dimensional index structure based on improved va-file and can in the cloud, International Journal of Automation and Computing 11 (1) (2014) 109–117.

[16] R. Buyya, R. Ranjan, R. N. Calheiros, Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services, in: Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Springer, 2010, pp. 13–31.

[17] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, L. Seinturier, A federated multi-cloud paas infrastructure, in: 2012 IEEE Fifth International Conference on Cloud Computing, IEEE, 2012, pp. 392–399.

[18] J. B. Abdo, J. Demerjian, H. Chaouchi, K. Barbar, G. Pujolle, Broker-based cross-cloud federation manager, in: Proceedings of the 8th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2013, pp. 244–251.

[19] G. Andronico, M. Fargetta, S. Monforte, M. Paone, M. Villari, A model for accomplishing and managing dynamic cloud federations, in: Proceedings of the 7th International Conference on Utility and Cloud Computing (UCC), IEEE/ACM, 2014, pp. 744–749.

[20] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game, IEEE Transactions on Services Computing 11 (1) (2016) 184–201.

[21] A. Dhole, M. V. Thomas, K. Chandrasekaran, An efficient trust-based game-theoretic approach for cloud federation formation, in: Proceedings of the 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Vol. 1, IEEE, 2016, pp. 1–6.

[22] A. Margheri, M. S. Ferdous, M. Yang, V. Sassone, A distributed infrastructure for democratic cloud federations, in: Proceedings of the 10th International Conference on Cloud Computing (CLOUD), IEEE, 2017, pp. 688–691.

[23] J. Shu, C. Liang, B. Wang, J. Xu, Building the federation of cloud service for big data, in: Proceedings of the 3rd International Conference on Big Data Analysis (ICBDA), IEEE, 2018, pp. 166–169.

[24] K. Papadakis-Vlachopapadopoulos, R. S. González, I. Dimolitsas, D. Dechouniotis, A. J. Ferrer, S. Papavassiliou, Collaborative sla and reputation-based trust management in cloud federations, Future Generation Computer Systems 100 (2019) 498–512.

[25] S. Udhayakumar, T. Latha, Trustworthy cloud federation through cooperative game using qos assessment, in: Proceedings of the International Conference on Pattern Recognition and Machine Intelligence (PReMI), Springer, 2019, pp. 30–37.

[26] A. Hammoud, A. Mourad, H. Otrok, O. A. Wahab, H. Harmanani, Cloud federation formation using genetic and evolutionary game theoretical models, Future Generation Computer Systems 104 (2020) 92–104.

[27] Y. Jadeja, K. Modi, Cloud computing-concepts, architecture and challenges, in: Proceedings of the International Conference on Computing, Electronics and Electrical Technologies (ICCEET), IEEE, 2012, pp. 877–880.

[28] Wordnet domains, `http://wndomains.fbk.eu/download.html`, accessed: 2020-05-17.

[29] S. Murugesan, I. Bojanova, Encyclopedia of cloud computing, John Wiley & Sons, 2016.

[30] J. Hong, T. Dreibholz, J. A. Schenkel, J. A. Hu, An overview of multi-cloud computing, in: Workshops of the International Conference on Advanced Information Networking and Applications (AINA), Springer, 2019, pp. 1055–1068.

[31] C. Cachin, R. Haas, M. Vukolic, Dependable storage in the intercloud, IBM research 3783 (2010) 1–6.

[32] U. Ahmed, I. Raza, S. A. Hussain, Trust evaluation in cross-cloud federation: Survey and requirement analysis, ACM Computing Surveys (CSUR) 52 (1) (2019) 1–37.

[33] M. Vukolić, The byzantine empire in the intercloud, ACM Sigact News 41 (3) (2010) 105–111.

[34] A. Bessani, M. Correia, B. Quaresma, F. André, P. Sousa, Depsky: dependable and secure storage in a cloud-of-clouds, Acm transactions on storage (tos) 9 (4) (2013) 1–33.

[35] Á. L. García, E. F. del Castillo, P. O. Fernández, Standards for enabling heterogeneous iaas cloud federations, Computer Standards & Interfaces 47 (2016) 19–23.

[36] C. A. Lee, R. B. Bohn, M. Michel, The nist cloud federation reference architecture 5, NIST Special Publication 500 (2020) 332.

[37] P. Mell, T. Grance, et al., The nist definition of cloud computing.

[38] S. Wu, D. Jiang, B. C. Ooi, K.-L. Wu, Efficient b-tree based indexing for cloud data processing, Proceedings of the VLDB Endowment 3 (1-2) (2010) 1207–1218.

[39] L. Zhu, Y. Wu, K. Gai, K.-K. R. Choo, Controllable and trustworthy blockchain-based cloud data management, Future Generation Computer Systems 91 (2019) 527–535.

[40] Y. Deng, M. R. Head, A. Kochut, J. Munson, A. Sailer, H. Shaikh, Introducing semantics to cloud services catalogs, in: Proceedings of the International Conference on Services Computing (SCC), IEEE, 2011, pp. 24–31.

[41] D. K. Nguyen, F. Lelli, Y. Taher, M. Parkin, M. P. Papazoglou, W.-J. van den Heuvel, Blueprint template support for engineering cloud-based services, in: Proceedings of the European Conference on a Service-Based Internet (ServiceWave), Springer, 2011, pp. 26–37.

[42] J. Cardoso, A. Barros, N. May, U. Kylau, Towards a unified service description language for the internet of services: Requirements and first developments, in: Proceedings of the International Conference on Services Computing (SCC), IEEE, 2010, pp. 602–609.

[43] N. Mainas, E. G. Petrakis, S. Sotiriadis, Semantically enriched open api service descriptions in the cloud, in: Proceedings of the 8th International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2017, pp. 66–69.

[44] L. Sun, J. Ma, H. Wang, Y. Zhang, J. Yong, Cloud service description model: an extension of usdl for cloud services, IEEE Transactions on Services Computing 11 (2) (2015) 354–368.

[45] S. Azizi, M. Zandsalimi, D. Li, An energy-efficient algorithm for virtual machine placement optimization in cloud data centers, Cluster Computing.

[46] M. Saravanan, M. Aramudhan, S. S. Pandiyan, T. Avudaiappan, Priority based prediction mechanism for ranking providers in federated cloud architecture, Cluster Computing 22 (4) (2019) 9815–9823.

[47] H. Nacer, K. B. Bey, N. Djebari, Migration from web services to cloud services, in: Proceedings of the International Symposium on Ubiquitous Networking (UNet), Springer, 2017, pp. 179–192.

[48] H. Nacer, D. Aissani, Semantic web services: Standards, applications, challenges and solutions, Journal of Network and Computer Applications 44 (2014) 134–151.