Hybrid Approach for Cloud Service Discovery System

Samer Hasan
Department of Computer Science & System Engineering,
College of Engineering (A), Andhra University, India.

V. Valli Kumari
Department of Computer Science & System Engineering,
College of Engineering (A), Andhra University, India.

Abstract

Cloud computing delivers services over the internet based on pay as you use financial model. Cloud providers publish service advertisements in different format (text, files, table and images) on the Internet. On the other hand, cloud consumers spend a lot of time and effort to find the appropriate cloud service using public search engines. Nowadays, cloud service discovery process considers as an important challenge for cloud service consumers, especially with massive increase in the number of available cloud services. In this paper, we present a layered architecture for cloud service discovery system. Additionally, we present a hybrid approach for cloud service discovery based on semantic and numerical similarity. Furthermore, we present domain ontology to create a shared understanding for cloud service domain. Finally, we implement proposed approach with protégé and SPARQL. Empirical results show the correctness and completeness of proposed system.

Keywords: Cloud computing; Ontology; Cloud services; Service discovery;
1. INTRODUCTION

Cloud computing [1] enables on-demand access to a shared pool of configurable resources. The top three abstract models are: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). There are four deployment models: Private Cloud, Public Cloud, Community Cloud and Hybrid Cloud. Actually, cloud providers publish cloud service advertisements on the internet on various formats (flat text descriptions, images, tables and files). On the other hand, cloud consumers use public search engine like (Google and Yahoo) to find the best matched service. Unfortunately, public search engines are not designed to support this activity which makes cloud service discovery process a hard and time consuming process. Cloud consumer should visit all providers’ websites to collect cloud service descriptions and compare between them manually. Some service providers don’t remember any word related to cloud in their names like (dropbox). On the other hand, some organizations which are not related to cloud services may use the word “Cloud” in their names like (ParkCloud, CurrencyCloud). Furthermore, cloud services have a lot of special characteristics [2] that need more and more effort from service consumers to find the appropriate one. In this paper, we present a layered architecture for cloud service discovery system and we present a hybrid approach for cloud service matching and ranking based on semantic and numerical similarity. In additional, we present cloud service domain ontology to create a shared understanding for cloud service environment. Finally, we implement an instance of proposed architecture by using protégé and SPRQL for semantic similarity and Percent Distance Similarity algorithm for numerical similarity. The rest of this paper is organized as follows. In Section 2 we survey the related work. In section 3 we explain the layered architecture for cloud service discovery system. In section 4 we present semantic and numerical algorithms. Section 5 shows the results of our experiments and section 6 is conclusion and future works.

2. RELATED WORK

Authors in [3,4,5,6] built a Multi-Agents cloud service discovery system based on search engine. They present three types of cloud service matching: numerical reasoning, compatibility reasoning and similarity reasoning and. In [7] authors present a framework for Cloud Services Comparing and Ranking based on Service Measurement Index (SMI). Three methods of service matching are presented in [8]: Equivalent reasoning, Numerical reasoning and Similarity reasoning. In [9] system used Description Logics (DLs) to query common metadata description of cloud service advertisements. In [10, 11,12] systems stored cloud service descriptions in

3. LAYERED ARCHITECTURE

Proposed layered architecture is divided into four layers as shown in figure 1

- First layer represents User Interface component which receives user queries based on predefined parameters and displays the matched services in ordered list.
- Second layer contains two components Service Matching and Service Ranking. Service Matching component calculates the semantic and numerical similarity between the user request and available services. On the other hand, Service Ranking component returns an ordered list of the matched services.
- Third layer consists of Domain Ontology component and Service Repository component. Domain Ontology is explained in section 3.1. Service Repository stores the available cloud services.
- Fourth layer contain two components Service collector component and Service Identifier component. Service Collector collects cloud service advertisements from cloud service providers and service Identifier classifies these services into different categories.

3.1 Domain Ontology

Cloud service providers publish service advertisements on the internet using various formats and only 1.8% of available cloud services has a semantic description
Additionally, Cloud service providers use different words to refer the same concept. As an example Desktop as a service available under different names like: “Amazon Workspaces” [27], “Desktop as a Service” [28], “Virtual Desktop” [29], “Desktop-Infrastructure-as-a-Service” [30]. Lack of standards consider as mean problem for cloud service discovery process. To overcome this challenge, Cloud service domain ontology describes all the concepts and relationships to create a shared understanding in Cloud service domain. We build cloud service domain ontology based on NIST [31] Cloud Computing Reference Architecture, other standards and information from cloud service providers’ websites. Figure 1 show some classes, attributes and individuals of cloud service domain ontology.

4. PROBLEM DEFINITIONS:

We define cloud service and user request as a set of attributes as follows:

\[ CS = \{s_{a_1}, s_{a_2}, s_{a_3} \ldots s_{a_n}\} \]  
\[ UR = \{u_{a_1}, u_{a_2}, u_{a_3} \ldots u_{a_m}\} \]

We need to find all cloud services with matching score bigger than specified threshold (th). To calculate the matching score we divide cloud service attribute into numerical and non-numerical attributes. Numerical attributes have numerical values like Ram= 8 GB, Hard disk = 500GB and price = 50 $ per month. On the other hand, Non-numerical attributes have non-numerical values like: operating system = Windows 7, location = India and extendable = yes. We define semantic similarity and numerical similarity to calculate the similarity between non-numerical and numerical attributes respectively as follows:

4.1 Semantic Similarity:

Cloud service providers publish service advertisements on the internet in various formats without any standards. Cloud Providers use different words to explain the same concept furthermore; they use the same word to explain contradictory concepts. Semantic similarity between two attributes determines to which degree they are close. We define the semantic similarity [32] between two attribute x, y as follows:

\[ SSim(x,y) = \rho \left| \frac{\alpha(x) \cap \alpha(y)}{\alpha(x)} \right| + (1 - \rho) \left| \frac{\alpha(x) \cap \alpha(y)}{\alpha(y)} \right| \]
$\alpha(x)$ and $\alpha(y)$ are the set of upwards nodes reachable from $x$ and $y$ respectively. $\alpha(x) \cap \alpha(y)$ represents the number of shared reachable nodes between $x$ and $y$. The value of $\rho \in [0,1]$ represents the degree of influence. As shown in figure 3 we calculate the similarity between two operating systems like Windows 8 and Mac as follows: $\alpha($Windows 8$) = \alpha($Mac$) = 4$, $\alpha($Windows 8$) \cap \alpha($Mac$) = 2$ and $\rho = 0.5$ then $SSim($Windows 8, Mac$) = 0.5$.

4.2 Numerical similarity:

Numerical similarity between two attribute values determines to which degree the values are similar. We define Percent Distance Similarity (PDSim) between to attributes values as follows:

$$PDSim(x, y) = \begin{cases} 1 - \frac{|x - y|}{x}, & y < 2x \\ 0, & y \geq 2x \end{cases} \quad (4)$$

$X$ is the attribute value in user request and $y$ is the value of the similar attribute in the candidate service. We use domain ontology and WordNet ontology to map between attributes in user request and available service. For example, we have four cloud services with different virtual CPU number (5, 20, 25, 29) and user request value for the same attribute is 15 then $PDSim(15,5)$ is 0.33, $PDSim(15,20)$ is 0.66, $PDSim(15,25)$ is 0.33 and $PDSim(15,29)$ is 0.06. $PDSim(15,5)$ is equal to $PDSim(15,25)$ because the distance between 15 and 5 is equal to the distance between 15 and 25. Percent Distance Similarity (PDS) algorithm calculates the similarity between two values based on the distance between these values. We define the matching score between user request and candidate cloud service as a sum of numerical similarity as follows:

$$ms = \sum_{i=1}^{k} \frac{Sim(ua_i, sa_i)}{k} : K = \max(|CS|, |UR|) \quad (5)$$

4.3 Matching Score:

We define the matching score between cloud service user request and cloud service candidate as sum of semantic matching and numerical matching of all attributes as follows:

$$ms(ur, cs) = SSim(ur, cs) + PDSim(ur, cs) \quad (6)$$
5. RESULTS AND DISCUSSION:

We collect 178 cloud service advertisements from providers’ websites. In our dataset there are 70 desktop as a service advertisements that are divided into three classes in domain ontology (DaaS, AmazonWorkspace and VirtualDesktop) as shown in figure 2. We built cloud service ontology using protégé and we used SPRQL as a semantic query language. Figure 4 show protégé result for SPRQL query. Figure 5 shows the number of matched services for three different user request based on matching threshold values. Semantic Query (SQ) shows the total number of semantically matched services. First we find the semantic matched service based on semantic query (SQ) then we calculate the numerical similarity for each candidate service with user request based on equation 5. Figure 6 shows cloud service discovery system results for one user request based on threshold value 0.9. The number of matched cloud services decreases while the threshold values increase. The bigger number of matched services requires more effort from user to find the best one, on the other hand, the smaller number of matched services means less chance for user to find the appropriate service.

6. CONCLUSION AND FUTURE WORK

Cloud computing becomes an important part of our life as individuals and organizations. The process of finding the appropriate cloud services is hard and time consuming. In this paper we present a hybrid approach for cloud service discovery system based on semantic similarity and numerical similarity. Semantic similarity determines the likeness between cloud service concepts and numerical similarity determines the likeness between attributes values. Experimental results shows the correctness and completeness of hybrid approach in addressing cloud service discovery problem. As future work we can integrate a semantic search engine with proposed system to automatically update service repository and domain ontology.
Figure 2. Layered architecture for cloud service discovery system

Figure 2. Cloud services domain ontology

Figure 3. DaaS class in cloud service domain ontology
Figure 4. SPRQL query results in protege

Figure 5. Numbers of matched services for different threshold

Figure 6. Cloud service discovery system interface
REFERENCES


Hybrid Approach for Cloud Service Discovery System


[27] https://aws.amazon.com/workspaces/


[29] https://www.clouddesktoponline.com/


