Compatibility Aware Cloud Service Composition for non-Xml Based Compositions using Fuzzy Inference System

Naveen Kumar Kalavapalli
Research Scholar, Department of CSE, JNTUA College of Engineering, Ananthapuramu, India.

Abstract – Semantic web services are promoted as a way to integrate web services in and outside the enterprise, while current semantic web service frameworks including OWL-S, SA-WSDL, and WSMO assume a uniform ecosystem of SOAP services and XML serializations, increasing number of real time services are being implemented using XML-RPC, RESTful interfaces and JSON serialization which is a non XML serialization. Semantic service platforms use XML mapping languages to translate an XML serialization of the data into an on-the-wire message format. This process of translation is known as grounding. The XML mapping techniques suffer from two problems: they cannot deal with the rising quantity of non-SOAP, non-XML services that are being deployed on the Web, and requires knowledge representation language used to represent service ontologies and semantic descriptions with the help of modeler which is used to create the descriptions of semantic web services to work with the serializations of the service ontology and syntactic language mapping. The proposed approach draw the service interfaces into the ontology and define ontological objects that represent the entire HTTP message, and then use backward chaining rules to translate between cases of invocation of semantic services and HTTP messages passed from and to service.

Index Terms – Cloud Service Compostion;Translation;Virtual Devices.

1. INTRODUCTION

Cloud Computing has become one of the promising IT industry technologies. Already there exists more than twenty definitions for cloud computing. Amongst them, Ian Foster and his colleague's definition of a cloud highlighted the main aspects of cloud such as dynamic scalability, available in the economy of scale and the ability to scale-up request. According to them cloud definition is : " A paradigm of distributed computing which is driven by economies of scale and a group of abstracted , virtualized, managed computing power, dynamically scalable, storage , platforms and services are provided at the request of external customers on the Internet. Moreover, clusters, supercomputers are invoked on non SOA application, while Cloud focuses on Web 2.0 and SOA technology. Though Clouds implemented widespread communication protocols such as HTTP and SOAP, integration and interoperability of all services and the final service deployment will remain the biggest challenges. Service deployment is the process of making a service ready for use, and often includes the deployment of several interdependent software components in heterogeneous environments. Differe nt approaches and tools discussed in terms of software and hardware by environment description, dependency abstraction, and process automation, to meet user needs. Among them, the virtual appliances have been increasingly adopted by the industry.

Virtual appliances, a meticulous set of virtual machines optimized operating systems, pre-built, pre- configured, ready to run applications and specific components of the integrated system, emerging as a revolutionary technology to solve the complexities of deployment service. Virtual appliances are proven to provide better service deployment mechanism. Therefore, they will be adopted as a major component working in the Cloud layer application. Nevertheless, most related work focused on meeting the needs of the user by using SOA and virtualization, neglecting the suitable cloud computing environment as a service deployment resource provider. It is complex to implement syntax and semantics of virtual machine description and user requirements in a heterogeneous environment such as Cloud. Therefore, the symmetric adjustment based on attributes applied between needs and demand is impossible. To address these problems, we propose a flexible approach to make the discovery of cloud virtual units based on ontologies. Following are the major contributions offered by this work

1) Presenting an approach that gives enough flexibility for end users to find the appropriate appliances from range of suppliers and deploy dynamically on different IaaS infrastructure providers [7, 8].

2) An advertising approach is offered for IaaS providers based on modeling virtual devices in one of the most important initiatives in the services of the Semantic Web, i.e. Web Service Modeling Ontology

3) Using ontological discovery for QoS deployment of virtual appliances on IaaS providers. This allows users to deploy their devices on the most appropriate IaaS providers according to their QoS preferences when both parties (suppliers and users)

are not using the same information to illustrate their services and requirements.

2. RELATED WORK

Konstantinou et al. Proposed an approach to plan, model, and deploy Cloud service compositions. In their approach, the deployment plan and solution model for the composition in Cloud platform are developed by expert users and executed by untrained users. Similarly, in our system, set of compatibility constraints from experts were captured to simplify the process of deployment used for end users by eliminating invalid composition solutions. However, as they also mentioned, their work lacks an approach for appliance selection and their placement on the Cloud which is offered by our work. Likewise Chieu et al. Proposed an automated deployment of integrated solutions using composite appliances. Even in their work, QoS objectives are not considered. In the same way another work has utilized Intuitionistic Fuzzy Set (IFS) for ranking service compositions in the Grid and SOA environments. It does not deal with user's constraints such as compatibility and whenever the problem is NP-hard (like cloud service composition problem) the execution time is unacceptable. Moreover, in comparison with the work which considered evolutionary approach such as NSGA-for service composition our approach improves the composition solution diversity and convergence and decreases the execution time. Unified Cloud Interface (UCI) provides ontology model for modeling Amazon EC2 services. Mosaic project is proposed to develop multi-Cloud oriented applications. In Mosaic, Cloud ontology plays and essential role, and expresses the application's needs for Cloud resources in terms of SLAs and QoS requirements. It is utilized to offer a common access to Cloud services in Cloud federations.

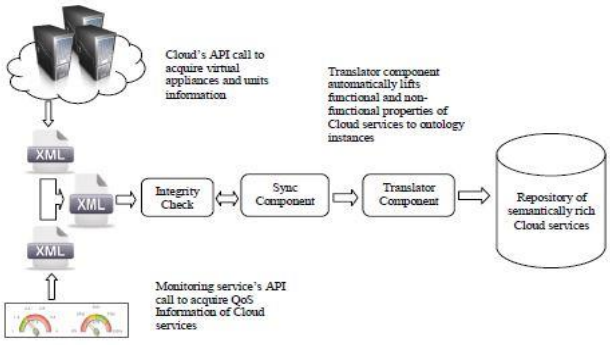


Figure 1: Translation of Cloud Composition

However, none of these ontologies focus on modeling of compatibility of Cloud services. The first step towards describing services and their QoS is to communicate with Clouds and the Cloud monitoring services through their APIs and gather required meta-data for building the repository. The process of metadata translation is demonstrated in fig.1. The components involved in this process are:

2.1 Integrity Checking

This component merges output messages of API calls for getting Cloud services description using Extensible Style sheet Language Transformations (XSLT) and then compares them with the previously merged messages using a hash function. If the outputs of the hash function are not equal, the component calls the Sync component to update the semantic repository.

2.2 Sync Component

The main use of this component is to keep the semantic based repository consistent with the most recent metadata provided by Cloud providers. The synchronization component is computing intensive, hence it is avoided unless the integrity checking component detects any inconsistency. This component receives the output message that is required for synchronization and finds the corresponding semantically rich services and updates them with the output of translator component.

2.3 Translator Component

When communicating a customer at the semantic level and a syntactic level web service, two data transformation directions are necessary. They are: Customer's semantic data must be written in an XML format that can be sent as a service request, and returning response data must be interpreted semantically by the customer. We use our customized Grounding technique on WSDL operations (that are utilized to acquire virtual appliance and unit metadata) output to semantically enrich them with ontology annotations. WSMO offers a package, which utilizes Semantic Annotations for WSDL (SAWSDL) for grounding. It provides 1. Two extensions attribute namely as Lifting Schema Mapping and Lowering Schema Mapping. Lowering Schema Mapping is used to transfer ontology to XML and lifting Schema Mapping does the opposite. In our translator component, the lifting mapping extension is adopted to define the process of how the XML instance data that is obtained from Clouds API calls is transformed to a semantic model.

There are a number of existing approaches that are capable of handling with incompatible services. However, many of them only paid attention on the compatibility of input and output (I/O) services and do not consider the inconsistencies that are caused by regulatory and further factors that are not related to features on duty. In addition, the OPTIMIS optimizes the entire lifecycle of services, from the construction and deployment services, to operating in cloud environments. The criteria of quality of service in OPTIMIS are trust, risk, eco- efficiency and cost. Evaluating cloud provider is achieved through the adoption of analytic hierarchy process approach (AHP) .The proposed approach can deal efficiently with a huge number of cloud services in the repository. The current implementation of the translator component is supporting cloud services based on XML. Specifications like Open Cloud Computing Interface

(OCCI) aims to provide a standard way to describe the cloud resources. Therefore, the research approaches that can enrich semantically this new formalism can be taken as a research direction to further enhancement of translator component.

3. PROPOSED WORK

This approach draws the service's interface into the ontology: it define ontological objects that represent the entire HTTP message, and then use backward chaining rules to translate semantic service invocation instances into the HTTP messages which are passed to and from the service. This novel technique is based on various principles:

1. Aim HTTP, the Web's local protocol: all the three Web service approaches such as SOAP, XML-RPC, and RESTful, use HTTP.

2. HTTP takes not only XML, but others like JSON serialization, text and images, sound and other data.

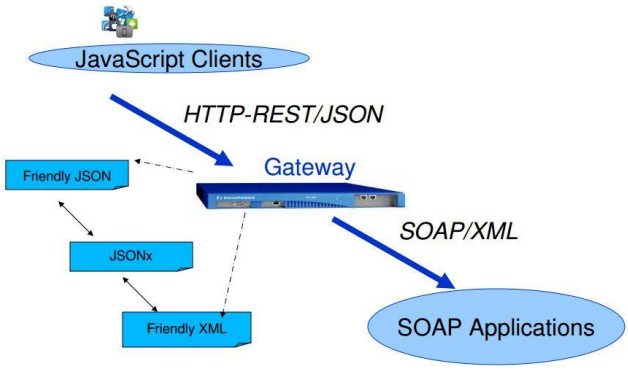
3. Stay in the ontology language: the fewer languages the engineer of the semantic service faces, the better will be the translation. When a semantic broker calls a web service, it results in an ontological abstract representation of the service invocation in a HTTP message that can be sent to the server. The server response is timely, and translates the HTTP message into an ontological representation of the result. These two processes are known as the lowering and lifting respectively. In this system, two generic rules are defined, which are called lower and lift

(lower ? serviceType ? serviceInvocation ? httpRequest)

(lift ? serviceType ? httpResponse ? serviceInvocation)

1. Specification of WSMO deftly dodges the question of grounding any type of Web service, the current SOAP implementations are WSMO centric.

2. Each Web service description can define its own version of lift and lower, which distinguishes the broker to unify? Service Type parameter so as to name the service being invoked. ?service Invocation represents the abstract invocation message and? httpRequest, ?httpResponse are the ontological representations of on-the-wire messages. The successful completion of the lower rule leads to the creation of ?httpRequest, that can be interpreted by the broker to describe the Web service. Once a response is received from the server, the rule of the lift works on the? httpResponse and modifies the original ?service Invocation frame to save the original return values. We applied the method in the Internet Reasoning Service (IRS) and a WSMO based broker using OCML for its Ontology Language. We now demonstrate an approach extends itself JSON serialization. JSON is a simple derivative of JavaScript data format, and is becoming more popular as a lightweight substitute to XML, mainly in the RESTful services. The lift and lower rules could directly operate JSON string representations, but it becomes awkward, prone to error, and fail to model changes significantly. Instead, we introduce a simple, ontologization of JSON.



JSON is built on the basis of objects and arrays. An object is a disordered set of key and value pairs, while an array is a series of values. Values may be numbers, strings, Booleans, objects, or null. In OCML, JSON ontologization enter a value of top-level class, which is then classified by Array and Object. An object consists of a list of elements, where each element of the list is a pair containing a key (a string) and a value. An array consists of a list of items, where each item in the list is worth. Finally, the JSON value - types such as string, number, and Boolean values are correspond to the equivalent OCML built-in types, while JSON null values are considered as nil. The JSON ontologisation in OCML is converted into the string of serialized JSON.

4. ANALYSIS

To illustrate how we can invoke, uniformly, web services using JSON and XML, we use two Flickr services: a) a list of recently changed photos in a user's account (flickr .photos.recentlyUpdated), and b) a list of formats in which those available (flickr.photos.getSizes).

```
(def-class Value () ?value
  :sufficient
  (or (String ?value)
      (Number ?value)
      (Boolean ?value)
      (= ?value nil)))

(def-class Object (Value)
  ((members :type Members)))

(def-class Array (Value)
  ((elements :type Elements)))
```

```
(def-class Elements () ?elements
  :iff-def (and (listp ?elements)
                (every ?elements Value)))

(def-class Members () ?members
  :iff-def (and (listp ?members)
                (every ?members Pair)))

(def-class Pair ()
  ((key :type String)
   (value :type Value)))
```

We are going to land the first service in a RESTful way, consuming JSON, and the second service via XML-RPC, consuming XML output, the integration of services at the ontological level. We start by lowering rule for service flickr.photos.recentlyUpdated

The rule is simple, the first half is concerned with the extraction of the necessary arguments from the? Invocation, and the second half to build body of an argument using those argument values. The rule sign Arguments handle arguments login with

a valid key of the Flickr account, whereas argstoRestRequest will convert the arguments set into URL parameters as Param I = value I & Param II = value II... and then sets the?http-request fields properly. While lifting the response, extract the contents of? Http-response into an ontological model of JSON. This extraction is performed by the two rules extractPhotoFromJson, extractPhotoListFromJson respectively. With the use of these extraction rules a list of photographs is occurred. With this list of photographs represented in OCML, we can invoke the second rule flickr.photos.getSizes with the help of XML -RPC

This time, the conversion of the subject of the invocation argument pairs used by Flickr is done in another rule argsForPhotosGetSizes. The use of a rule that means we could share the logic between the XML-RPC version, and a REST or SOAP version. The whole argument is again signed with signArguments and forwarded to argsToXmlrpcR equest rule. ArgsToXmlrpcRequest The rule fulfills a similar function to argsToRestRequest, but this time, we are creating an XML message - RPC with a structure to hold the pairs, rather than integrating them into a URL.

5. RESULTS

In this paper, we have described Fuzzy Logic, an open source Java library for fuzzy systems which allow us to design FLCs following the standard. It allows us to reduce programming work and extend the range of possible users applying fuzzy systems and FLCs. In this case, we developed an FLC controller for wall following behavior in a robot. The example shows how FLC can be used to easily implement fuzzy logic systems. The Fuzzy Logic software package is continuously being updated and improved. At the moment, we are developing an implementation of a C/C++ compiler for fuzzy inference systems. This will allow easy implementation with embedded control systems using different processors.

In proposed system providing security to cloud service provider to evaluate reliability and trust of providers from user feedbacks and monitoring services together this consists of collecting required raw data from trusted sources and statistically analyzing and aggregating them. When the number of preferences defined by the user using the if-then rule increases, the solution's prominence In proposed system calculate the user trust values because only trusted values to be considered. This will provide the security to cloud. When untrusted values are come it will be deleted Using Integer Linear Programming to optimize the cost and time.

Major Cloud providers have large repository of virtual appliance and unit services. For example, the size of Amazon Web service repository alone is larger than 10.6 megabytes. To enhance the efficiency of the translation approach we only synchronize when the translation service is triggered by checking the integrity of the component. The number of services increased the in the repository by merging repositories from various Cloud providers to investigate the scalability of our approach in terms of execution time needed for the translation. For each case of repository size, the experiment is repeated for 30 times and the values are plotted in Fig3. Regression analysis shows a positive and linear relationship between the repository size and the translation time. The evidence confirms that the regression coefficient is 0.6621, which suggests that if the size of the data to be translated increases by a megabyte, the translation time increases approximately by 0.6 second. Therefore, the synchronization function can be executed online in an adequate time even if a significant percentage of virtual appliance and unit properties are updated.

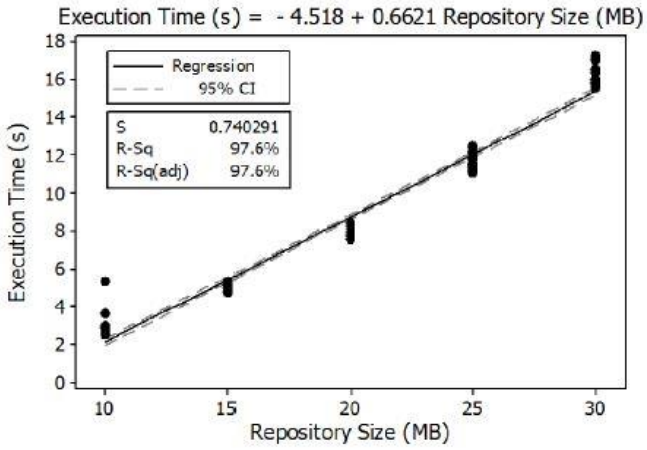


Figure 2: Execution Time Vs Repository Size

6. CONCLUSION

In this work, when communicating a customer at the semantic level and a syntactic level web service, two data transformation directions are necessary. They are: Customer's semantic data must be written in an XML format that can be sent as a service request, and returning response data must be interpreted semantically by the customer. The system uses a customized grounding technique on WSDL operations that are utilized to acquire virtual appliance and unit metadata output to semantically enrich them with ontology annotations. WSMO offers a package, which utilizes semantic annotations for WSDL for grounding. It provided two extensions attributes namely Lifting Schema Mapping and Lowering Schema Mapping. Lowering Schema Mapping is used to transfer ontology to XML and lifting Schema Mapping does the opposite. In the translator component, the lifting mapping extension is adopted to define the process of how the XML instance data that is obtained from Clouds API calls is transformed to a semantic model.

REFERENCES

[1] Amir Vahid Dastjerdi, Member, IEEE and Rajkumar Buyya, Senior Member, IEEE (2014) Compatibility-Aware Cloud Service Composition under Fuzzy Preferences of Users IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 1, JANUARY-MARCH 2014

[2] J. De Bruijn, H. Lausen, A. Polleres, and D. Fensel, "The web service modeling language wsml: An overview," in Proceedings of the 3rd European conference on The Semantic Web: research and applications, 2006.

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in Proceedings of the Grid Computing Environments Workshop (GCE),

[4] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services. W3C member submission, World Wide Web Consortium (W3C) (November 2004)

[5] Lausen, H., Polleres, A., Roman, D.: Web Service Modeling Ontology (WSMO). W3C member submission, World Wide Web Consortium (W3C) (June 2005)

[6] Lambert, D., Domingue, J.: Photorealistic semantic web service groundings: Unifying RESTful and XML-RPC groundings using rules, with an application to Flickr. In: Proceedings of the 4th International Web Rule Symposium (RuleML 2010). (October2010).

[7] Aarti Singh, Manisha Malhotra, "Security Concerns at Various Levels of Cloud Computing Paradigm: A Review", International Journal of Computer Networks and Applications, 2(2), 41-45.

[8] Sumandeep Aujla, Amandeep Ummat. "Task scheduling in Cloud Using Hybrid Cuckoo Algorithm". International Journal of Computer Networks and Applications (IJCNA), 2(3), 144-150.

Authors

Kalavapalli Naveen Kumar is a M.Tech scholar in JNTUA College of Engineering (*Autonomous*), Ananthapuramu. He received his B.Tech degree from JNT University Anantapur, Ananthapuramu. His areas of interests include Cloud Computing, Machine Learning and Data Mining.