Web Service Discovery using Relational Database and Apache Lucene

Bhavin Solanki, Jasmin Jha
L. J. Institute of Engineering & Technology, Department of Computer Science and Engineering, Gujarat Technological University, Ahmedabad, Gujarat, India

Abstract- **The number of available Internet services increases every day at rapid speed. This demands distributed models and architectures to support scalability to enable efficient publication and retrieval of services. Almost everything is going to be distributed. We have proposed an approach which using relational database and Apache lucene for web service discovery. Relational Database used for storing wsdls' information. Web service search is done by querying that relational database by sql engine and using Apache lucene. Apache lucene is the modern enterprise text search engine. The proposed approach is simpler and modern compared to using traditional standard UDDI for web service discovery process. The approach will greatly help the developers of web services for searching desired web services from very large collection for composition of web services.**

Index Terms- **Web Service Discovery, Service Oriented Architecture, Apache Lucene, WSDL Parsing**

I. INTRODUCTION

Service Oriented Architecture (SOA) enables developers and end-users to dynamically discover and assemble loosely coupled software modules (represented as services) to get their work done. The SOA paradigm is therefore a viable model for the modular composition and reuse of third-party software components on a large scale which is now evolved as cloud computing on addition of distributed computing systems. The SOA has been widely used as a promising approach for providing a high-level interoperability of distributed resources by eliminating the technical inconsistency of different software, platforms and infrastructures within an enterprise or across geographically distributed enterprises.

Service discovery problems in SOA are described as the task of efficiently and accurately finding a relevant set of services that satisfies a given service request from a user. The problem of service discovery becomes computationally difficult due to these four main reasons: 1) the number of services are large and increasing (e.g., Web Services); 2) the system is highly dynamic because of asynchronous service communication, addition, modification and deletion of services; 3) the system is distributed because of independent ownership and hosting of services geographically at different places; and 4) accurate and quality based service matchmaking for service discovery is computationally very expensive.

Many approaches have been proposed to solve the service discovery problem. The common tasks in all of these approaches are two. The first task is to organize available service descriptions. The second task is to run a discovery algorithm to efficiently navigate the search space of services and find which are relevant to the user request. There are two traditional ways of organizing services: 1) the thematic way, where services are organized based on their domains [1] according to the thematic categorization such as NAICS [2] and UNSPCS [3]; and 2) a functional feature based classification [4] and clustering [5].

Service matchmaking is also an important aspect in service discovery. Service matchmaking is pair-wise matching analysis of service descriptions to get the functional similarity between services. The

two common service matchmaking paradigms are syntactic and semantic matchmaking. Syntactic matchmaking is based on keyword-based similarity using statistical similarity measures (e.g., cosine similarity, KL divergence) or semantic lexicon (e.g., WordNet) based statistical similarity measures. Semantic matchmaking techniques is based on using ontologies and match services using description logic (DL) based subsumption reasoning or taxonomic structural reasoning. Semantic matchmaking considered as more accurate results as compared to the syntactic approaches.

Service discovery can be designed to be centralized or distributed. Centralized service discovery suffers from some major drawbacks that are common to all centralized systems like not scalable, single point of failure and mainly leading to high network traffic.

In our approach, we have parse the WSDL files and stored them in the relational database. Additionally, we have used Apache lucene. The extracted information like web service name, operation, input, output and web service URL stored in relational database as well as in the Apache lucene. Apache lucene is Java full-text search engine. We will query the relational database and Apache lucene with user requirements like category, operation name, input and output. The user input format and user output format is described in the approach overview section.

II. RELATED WORK

There are many models have been available in the literature but we have concentrated on the most recent models have been seen by us in the literature. Five most recently published models in the literature are described here.

i)Self-Organized P2P Approach to Manufacturing Service Discovery for Cross-Enterprise Collaboration[7]

In this paper, they have present a self-organized semi-structured P2P framework that supports scalable and efficient MS discovery for cross-enterprise collaboration by forming and maintaining autonomous enterprise peer groups (PG).

Super Peer (SP)-based P2P overlay network is self-organized which realizes both advantages of centralized search and decentralized search. The presented framework addresses the drawbacks of both unstructured and structured P2P architectures, by minimizing the overhead either incurred by the random peer communication or maintenance of peer overlay structure. Manufacturing Service (MS) request (SR) is first routed to the suitable SP and further to its leaf peer in a systematic way.

ii) SMARTSPACE: Multiagent Based Distributed Platform for Semantic Service Discovery [8]

In this Paper, they have presented the SMARTSPACE platform that is based on the multiagent based distributed semantic service discovery. SMARTSPACE is multiagent based distributed SOA middleware. Services and users' queries are modelled as agents. The middleware is distributed into a system of federated registries, each of which is managed by a pair of middleware agents.

Presented the SmartDiscover algorithm offers an agent based hybrid-P2P approach, which is similar to the super-peer approach, which combined the positives of P2P based approaches.SmartDiscover is the composition of three component distributed algorithms: 1) SmartCluster, 2) SmartDirect and 3) SmartMap. SmartCluster is clustering service descriptions into semantic taxonomic cluster (STC) space. SmartDirect and SmartMap algorithms satisfying a user query by crawling the structural and semantic properties of the STC space to discover the relevant service agents.

iii) A Framework for Goal-Oriented Discovery of Resources in the RESTful Architecture [9]

This paper presented framework for the goal-oriented discovery of services and content in the web, described an algorithm for the induction of rules for discovery and its application to data and resource levels in the REST architectural style of the web. An agent architecture compatible with the discovery framework also defined for implementing combined services or contents in cases where discovery is required. The use case of an agent architecture to discover related news items illustrated how the agent performs discovery tasks whenever appropriate and makes plans.

iv)A Quality-based Semantic Service Broker Using Reachability Indexes [10]

This paper presented a quality-based semantic service broker using reachability indexes. In the presented service broker, the service information, including inputs, outputs and descriptions are translated to semantic concepts by comparing semantic terms. The

services are modelled as vertices in a conceptual aggregation graph and connected to other vertices based on the translated semantic concepts. The reachability indexes are calculated and assigned to each vertex. Searching for the composition between two services done by finding a path between two vertices in the conceptual aggregation graph. The reachability between two vertices checked immediately and the path searching been accelerated. When all the paths between two vertices have been found, all the possible composition solutions for the two services have been found also. The qualities of these solutions have been evaluated and ranked. Even if there is no solution that fits the specified quality requirements, a suggested solution from the similarity distance relaxation is provided.

v) A DHT-based semantic overlay network for service discovery [6]

This Paper introduced Efficient Routing Grounded on Taxonomy (ERGOT) decentralized system built upon three main elements:

1) A DHT (Distributed Hash Table) protocol, used to advertise semantic service descriptions annotated using ontology concepts. With assumption that semantic annotations of service descriptions are expressed in W3C's standard SAWSDL.

2) A SON (Semantic Overlay Network) which enables the clustering of peers that have semantically similar service descriptions. The SON is constructed incrementally as by-product of service advertising via a DHT.

3) A measure of semantic similarity between service descriptions which overcomes the exact-term search limitations of DHTs. This measure combines the feature-based and information content-based similarity models.

III. APPROACH OVERVIEW

Figure 1 show the workflow of our approach for web service search using relational database and apache lucene. We have almost 6500 WSDL files' collection.

We have three datasets named as: 1) WSDream[13] 2) SAWSDL-TC3 [14] and 3) JGD_WSDLS_FULL [15]

We have parse the all WSDL datasets using Java and stored them into the relational database of MySQL. We have some WSDL files already in the category-wise and most in mixed way. We have created the category-wise separate tables like Communication, Economy, Education, Food, Geography, Medical, Simulation, Travel and Weapon. If we have the category of the WSDL then we have stored it in the specific category otherwise in the "others" category. The third dataset JGD is specifically related to geography category.

After parsing WSDL files, we have stored that extracted data into the relational database of MySQL. We have extracted elements like id, wsdlname, webService, Operation, input, output and web_service_url from the WSDL files.

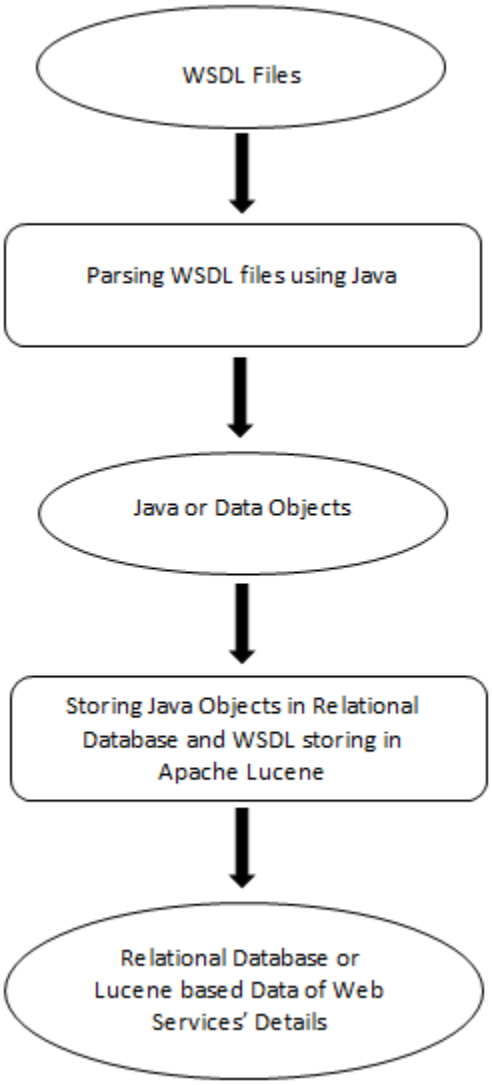


Figure 1 Proposed Solution's Workflow for Web Service Discovery

After parsing WSDL files, we have stored that extracted data into the relational database of MySQL.

We have extracted elements like id, wsdlname, webService, Operation, input, output and web_service_url from the WSDL files.

We are using the popular enterprise text search engine Apache Lucene for the web service discovery. Apache Lucene supports many file formats like HTML, PDF, Microsoft Office Formats, Plain text, XML, JSON, CSV (Comma/Column Separated Values).

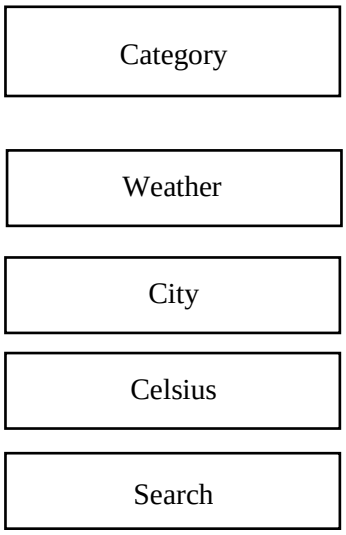
After storing that extracted WSDL information in relational database and in Apache Lucene we will query that information using SQL engine and lucene engine.

We know that SQL engine is really fast in querying database for medium size of database, but we also used the Apache Lucene which is popular enterprise text search engine in the market today and used by many giant enterprises.

The general user input format from the user and query response format is as follows:

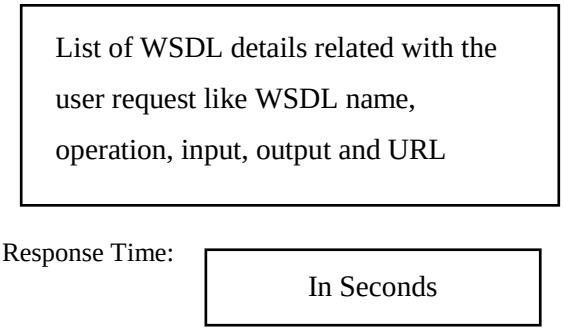
User Input Format

Category (Dropdown Menu):



Query Response Format

Result Set:



If the user selects certain category then the search will be done on that specific category's table in relational database. If the user do not know about the category or did not get the desired results then the user will select the others category. The selection of "others" category is due to we have only partial WSDLs in category-wise.

IV. IMPLEMENTATION SETUP AND RESULTS

We have used java language for parsing WSDL files. The extracted information is stored in MySQL database. The WSDL Parsing is shown in figure 2. We have used Apache Lucene version 4.9.

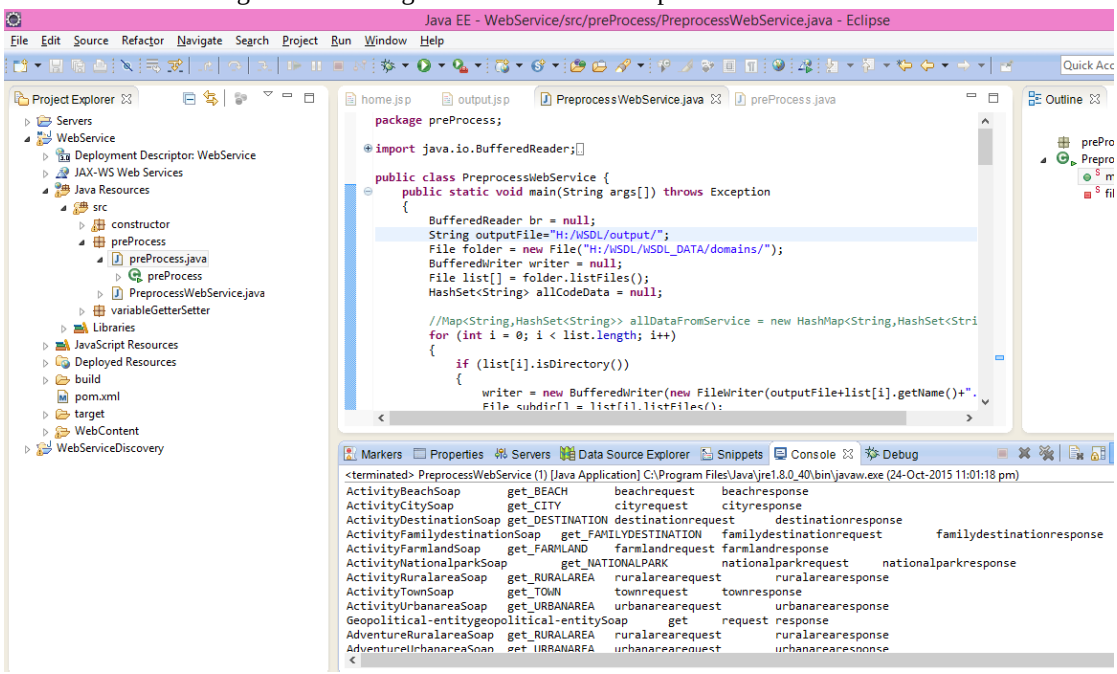


Figure 2 WSDL Parsing

Figure 3 User Input Format in Relational Database

Web Service Discovery Using Relational Database

WebServiceName	Operation	Input	Output	URL
Academic-degreegovernmentFundingSoap	get_FUNDING	fundingrequest	fundingresponse	http://127.0.0.1/services/sawsdl_wsdl11/Academic-degreegovernmentFunding/
AwardgovernmentFundingSoap	get_FUNDING	fundingrequest	fundingresponse	http://127.0.0.1/services/sawsdl_wsdl11/AwardgovernmentFunding/
DegreegovernmentFundingSoap	get_FUNDING	fundingrequest	fundingresponse	http://127.0.0.1/services/sawsdl_wsdl11/DegreegovernmentFunding/
GovernmentFundingSoap	get_FUNDING	fundingrequest	fundingresponse	http://127.0.0.1/services/sawsdl_wsdl11/GovernmentFunding/
ProjectilegovernmentFundingSoap	get_FUNDING	fundingrequest	fundingresponse	http://127.0.0.1/services/sawsdl_wsdl11/ProjectilegovernmentFunding/

Time Taken (In Milli Second): 281.0 ms

Back

Figure 4 Results using Relational Database

Web Service Discovery Using Apache Lucence

Category : Education

Web Service : governmentfunding

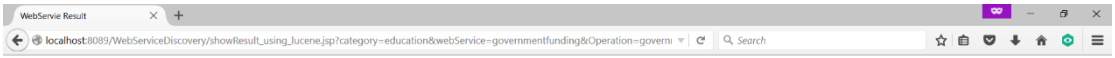
Operation : governmentfunding

Input : funding

output : funding

Search

Figure 5 User Input Format in Apache Lucene



Web Service Discovery Using Apache Lucence

WebServiceName: GovernmentFundingSoap
Operation: get_FUNDING
Input: get_FUNDINGRequest
Output: get_FUNDINGResponse
WebServiceUrl: http://127.0.0.1/services/sawsdl_wsdl11/GovernmentFunding/

Time Taken (In Milli Second): 428.0 ms

Figure 6 Results using Apache Lucene

As we can see in the figure 4 and 6, the results using relational database and apache lucene are given in milliseconds. We are discovering a web service related to Government Funding. The results using Relational Database gives 5 related web services in the category of education within 281 milliseconds. On the other hand, the same web service discovery is done in Apache Lucene also. Apache Lucene gives the single web service according to our requirement within 428 millisecond. The results are taken on restart of fresh eclipse and apache tomcat. If we are querying same web service then the response time will be very low. This time the response time will be within 20 to 60 milliseconds. In Apache Lucene, we have to input some exact related words and have to try more than one search by tweaking the search inputs. This is because our Lucene is working on nearly exact match and relational database system using wildcard entries using "Like" based queries.

No one system can be perfect and best always. Some systems may give you better results and some systems my give you better speed. Speed and Quality are inversely proportional to each other. If we want speed then we have to compromise with quality. If we want quality then we have to compromise with speed.

The main idea behind our system is we are using both relational database approach and apache lucene approach in our system. This means that the users will use both approach as their convenience. Most probably, our suggestion is to use both approach and have final decision by their own regarding which web service to use.

The scope of work is limited to finding the appropriate WSDL file means the web service which may be able to fulfil our web service requirement.

V. CONCLUSION

Our approach is fall into the category of private enterprise software for enterprise specific web service discovery process. The main aim of the approach is to make the web service discovery process simpler especially from development perspective of web service composition. Our approach will help the developer to find the required web services which will fulfil the requirement of final web service after their composition.

We have parsed the WSDLs files of the web service as much as we have. We have used the MySQL relational database for the storage of the extracted data from WSDL parsing. We have also used the Apache Lucene for indexing of the WSDL files and their elements. So, we have two option to query the WSDL information extracted from WSDL files. One is query via SQL query using SQL engine and second is using Apache Lucene. Apache Lucene is used by tech giant website like Amazon for search products.

According to our best known findings, there is no any system which uses these kind of workflow and our kind of system for the web service discovery. UDDI is complex while our system is simpler and modern.

VI. FUTURE WORK

We have around 6500 WSDL files in the implementation. In future we will work on very large collection of the WSDL files. We wish to develop some interesting features like Mashape web-based registry has implemented. We wish to include the quality of service feature in our system like ratings given by the user of the web service. Solr and SolrCloud would be the next enhancement in the present work because solr is based on apache lucene gives better results and flexibility in user interface. SolrCloud is used for the distributed search in the distributed environment.

REFERENCES

[1] UDDI. (200). The UDDI technical white paper [Online]. Available: http://www.uddi.org/

[2] North American industry classification system [Online]. Available: http://www.census.gov/eos/www/naics/

[3] United Nations. United Nations standard products and service code [Online]. Available: http://www.unspsc.org

[4] A. HeB and N. Kushmerick, "Learning to attach semantic metadata to web services," in Proc. 2nd ISWC, 2003, pp. 258–273.

[5] M. A. Corella and P. A Castells, "Heuristic approach to semantic web services classification," in Proc. 10th Int. Conf. KES, 2006, pp. 598–600.

[6] Giuseppe Pirro, Domenico Talia, Paolo Trunfio, "A DHT-based semantic overlay network for service discovery", ELSEVIER, Future Generation

Computer Systems, VOL NO. 28 (2012), pp. 689-707, ISSN: 0167-739X
[7] Wenyu Zhang, Shuai Zhang, Feng Qi, and Ming Cai, "Self-Organized P2P Approach to Manufacturing Service Discovery for Cross-Enterprise Collaboration", IEEE Transactions On Systems, Man, And Cybernetics: Systems, Vol. 44, No. 3, Page 263-276, March 2014
[8] Sourish Dasgupta, Anoop Aroor, Feichen Shen and Yugyung Lee, Member, IEEE "SMARTSPACE: Multiagent Based Distributed Platform for Semantic Service Discovery", IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 7, pp. 805-821, July 2014
[9] Jos´e Ignacio Fern´andez-Villamor, Carlos A. Iglesias, and Mercedes Garijo, "A Framework for Goal-Oriented Discovery of Resources in the RESTful Architecture", IEEE Transactions On Systems, Man, And Cybernetics: Systems, vol. 44, no. 6, pp.796-803 June 2014
[10] Yen-Ting Lee, Ching-She Wu, "A Quality-based Semantic Service Broker Using Reachability Indexes" IEEE World Forum on Internet of Things (WF-IoT), pp. 277-282, Date Of Conference 6-8 March 2014, Place Seoul
[11] Sven Kaffile, Karsten Loesing, "Open Chord version 1.0.4 User's Manual", Distributed and Mobile Systems Group, Otto-Fredrich-Universitat Bamberg, Germany, 2007
[12] Yilei Zhang, Zibin Zheng, and Michael R. Lyu, "WSExpress: A QoSaware Search Engine for Web Services," in Proceedings of the 8th International Conference on Web Services (ICWS2010), Miami, Florida, USA, July 510, 2010, pp.8390.
[13] WSDream WSDL Dataset, http://www.wsdream.net/dataset.html
[14] SAWSDL-TC3 WSDL Dataset, http://projects.semwebcentral.org/frs/?group_id=156&release_id=381
[15] JGD_WSDLS_FULL WSDL Dataset, http://fusion.cs.uni-jena.de/fusion/activity/jena-geography-dataset/