A Novel Web Service Composition and Web Service Discovery based on Map Reduce Algorithm

Shashank Shetty
PG Scholar, Dept. of C.S.E
NMAMIT, Nitte
CDAC, Pune

Shalini P.R
Dept. Of C.S.E
NMAMIT, Nitte
Karnataka, India

Aditya Kumar Sinha
Principal Technical Officer
CDAC
Pune, India

ABSTRACT
The paper focuses on the web service composition and web service discovery based MapReduce algorithm, which is the one of the component of the big data problem resolver tool Hadoop. Many of the IT companies are currently in the journey to Service Oriented Architecture (SOA) with web service as the standard protocol for implementation. The overwhelming popularity of web service has made a huge impact on web service repository, because of which managing it using traditional UDDI platforms has become difficult. Integrating Hadoop ecosystem with web services can provide higher QoS to the user request for web services. Web service composition and Web service discovery plays a crucial role in the management of web service. Web service discovery involves locating or finding the exact individual web services from the service registry and retrieving previously published description for new web application. Web service composition is technique to combine simple web service to satisfy the user requirement. An efficient MapReduce algorithm can help in providing better management of web services

General Terms
Web services and Technologies, Cloud Computing, Semantic Web and its applications.

Keywords
Web service, Web service discovery, Hadoop, Map reduce, Web service management, Web service Composition, Big data

1. INTRODUCTION
Web services are the self describing and modular applications that can be published, discovered and accessed from various locations through web. Web services has gained an increasing amount of popularity in many organizations. Due to the overwhelming popularity, there has been need to maintain these web services by efficiently selecting the optimal web services out of several unwanted web services by filtering it. In particular, if a single web services cannot satisfy the user requirement then there is a necessity to combine the existing web services in order to fulfill the user requirements. Hence the notion web service composition came into picture.

In recent years, there has been substantial attraction towards managing a web service by a industry and many researchers. This popularity is due to the interoperability attribute of the web service [6]. Since web service is interoperable, this made it to grow web services across the organization and made it difficult to manage. In simple terms, the web service is the piece of the software application whose features are defined by the XML based language [7]. Some of the examples of the web service are online ticket purchase, online hotel reservation and auction. As a building block of web service many protocols and technologies were developed. Some of the few technologies are as follows: Universal Description, Discovery, and Integration (UDDI) [2] [3], Simple Object Access Protocol (SOAP) [4] and Web Service Description Language (WSDL) [5]. UDDI provides a service registry for web service discovery and advertisement. SOAP acts as an foundation framework for communication of the web services. WSDL provides the service provider an platform to describe their applications. DAML-S [9] is DAML based web service ontology which defines a standard for web service discovery and message passing. It provides the standard set of mark-up language for the web service providers to describe the properties of their web services in computer interoperable form. Some of the other initiatives towards web service are Business Process Execution language for web service (BPEL4WS) [8] which is considered as the standard for web service composition. This BPEL4WS allows to create different complex processes and wire them together, for example invoking web services, data manipulation, throw errors or end the process. So, these activities are grouped together and structure these activities by showing the sequence of execution, such as sequential, parallel or depending on certain condition. Both DAML-S and BPEL4WS focuses on the representation of web service composition, where process and also binding of it is known priori.

Despite of all these efforts, this web service discovery and composition as been a complex task and its difficult to manually deal through it. Problem occurs due to the following reasons. First, due to customers delegating huge amount of web service task and creating a huge amount of web service repositories. Secondly, web services can be created easily within ample amount of time. Hence, web service composition and discovery must be done in runtime. Also, the selection process must be carried out based on the up-to- date information. Hence there is a need to build an automated system to manage web service by selecting a optimal web service by filtering the unwanted web service and also automated web service composition. The Hadoop, one of the big data problem resolver tool is used to efficiently manage the web service and MapReduce algorithm is used to automatically choose the optimal web service.

The paper is organized as follows: in section 2 we discuss various related works on Web service discovery and web service composition. Section 3 reviews the methodologies involved in the proposed method and also how it is achieved. In Section 4, we present the methods to achieve the proposed system. Finally, we conclude by giving a brief overview of the proposed method and also give a glimpse on future works.

2. RECENT WORKS

Web service has been a promising and an emerging technology, there has been considerable amount of recent works on the issues related to web service discovery and composition. Web service discovery involves locating or finding the exact individual web services from the service registry and retrieving previously published description for new web application. For example UDDI registry [3] where it contains white pages for all contact information, yellow pages for industry taxonomies and green pages for technical information about web services. UDDI enquiry is an API provided by the UDDI to interact with the system, that is it locates and finds UDDI registry entry. Simple search engines like [11] [12] [13] are used to find the required web services. As of now these search engines provide a simple keyword based search on web service description and the most of the UDDI search engines are limited to syntax based search. The client can just search the UDDI registry based on the string in the service description. Bellur et al. [16] discusses about the improved matchmaking algorithm for dynamic discovery of the web service. The method involves selecting a web services by constructing the bipartite graph and defining the optimal web services from it. The large number of possible paths in a larger data paths need to be searched in a given time period. Hence this high space complexity makes above traditional methodology weaker when searching the large web service repositories. Dong et al [14] discusses a new method of search engine using similarity search mechanism. The query is transformed into a common representation and then for a particular query related web service is found. This method may have a problem while searching a large collection within a certain time limit. N. Gholamzadeh et al. [10] proposes data mining based web service discovery technique. Here fuzzy clustering based algorithm for discovering similar web service using a single query is developed. Y. Zhang et al. [15] defines a web search engine approach for finding the desired services using functional and non functional QoS characteristics. Sreenath et al. [23] discusses that web service discovery is an exhaustive process, because lot of services are found. Selecting best among them is a complex task. Hence the agent based approach is formulated for web selection.

Web service composition also plays a vital role in web service research. Web service composition is technique to combine simple web service to satisfy the user requirement. For example, if user wants to have a tour from India to New York then there is a need of booking an air ticket, booking a hotel room, booking a taxi to airport and also taking care of entertainment and so on. So to have all these there is a need of performing these task one by one. This makes it time consuming and takes lot of efforts to carry out his task. Since user requires all in one service, the web service composition notion as been developed. Kona et al [45] discusses about the semantic web service composition where acyclic graph from the input request is generated iteratively. Hence, all possible services that are invoked are added to the graph. Therefore, causing difficulty in eliminating the unwanted web service. Mier et al. [43] discusses about the automatic web service composition using A* algorithm. Firstly, web service dependency graph is computed using the method discussed in Kona et al's [45] work. Later, the unwanted web services are eliminated and finally A* search algorithm is applied to find the optimal web service. Shiaa et al [44] discusses the automatic web service matching using the semantic matching. Based on the user request, the set of similar matching web services are extracted and graph is created dynamically by matching semantic web services. Once the acyclic graph is created, search from goal to start node is performed. The main drawback of the above mentioned methods are while searching the huge repositories, the response time for the user request decreases drastically. So, In the proposed method web service is integrated with Hadoop ecosystems to gain more accurate results. So key difference between the Hadoop model and all the above model is that we intend to build the web service management system with QoS metrics like reliability, higher throughput, higher response time and availability. With the QoS metrics, the proposed model works on larger web request and in response produces optimal web services by filtering unwanted web services. Hence in the proposed methods, the MapReduce algorithms are used for web service composition and HBASE for the web service discovery are used.

3. PROPOSED METHODOLOGIES AND ITS IMPLEMENTATIONS

The core framework of the Hadoop based web service management is similar to the reference [17]. The proposed system is divided into four modules and are explained briefly as follows (see Figure 1):

3.1 Infrastructural Setup of Hadoop Ecosystem

The Hadoop is framework which is used to process and store a huge amount of data. This framework is integrated with the web service to manage it efficiently to gain higher QoS.

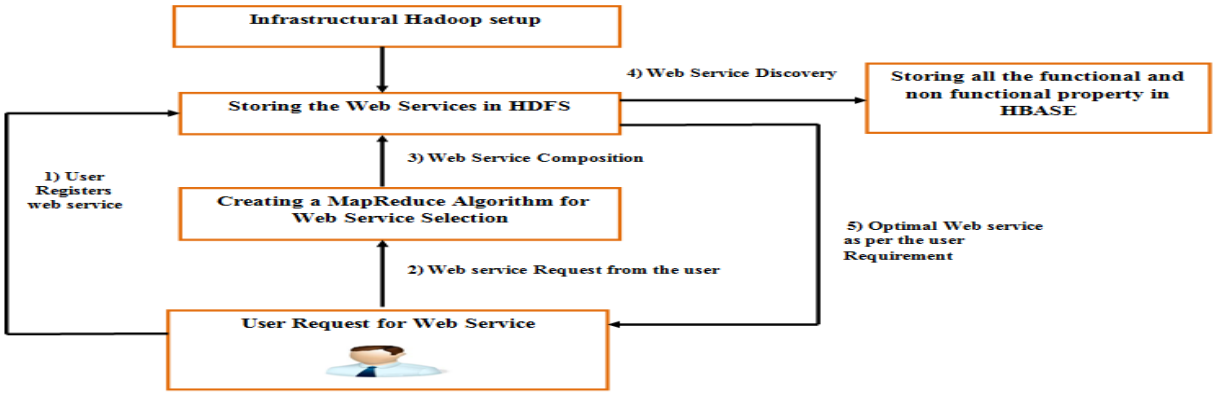

Fig 1: A novel web service composition and web service discovery using MapReduce algorithm.

Hence the infrastructural setup of hadoop ecosystem is built using the following steps:

Step 1: Downloading and installing Java 1.6 from http:// www.oracle.com [18], Eclipse Europa 3.3.2 from http:// www.eclipse.org [19] and Cygwin from http://www.cygwin.com [20] by configuring and starting SSH Daemon.

Step 2: Download Hadoop from http://archive.apache.org/ [21], copy it in Cygwin home folder, and unpack hadoop

Step 3: Configure Hadoop by editing the configuration file hdfs-site.xml, mapred-site.xml and core-site.xml . Also install hadoop eclipse plug-in [22] and change the eclipse java perspective to MapReduce environment.

Step 4: Download and install Zookeeper [23] and HBASE [24] from http://archive.apache.org/, Setup hadoop location in eclipse and start all the cluster. Now, Hadoop ecosystem is setup for any task.

3.2 Storing the web service in Hadoop distributed File System (HDFS)

HDFS [25] [26] is used for distributed storage of data to solve the problem of storing big data. HDFS follows the master slave architecture. The HDFS consist of one Namenode, one secondary Namenode and many Datanodes. Namenode is the master node where all the metadata of the file and also the file access permission for the user. Secondary Namenode is the backup node to the Namenode and Datanode is the slave node where all the actual data will be stored. HDFS also provides replication of data into three nodes for error recovery. The Datanode periodically sends reports to the Namenode. If Namenode does not receive any report then it treats the node as failure. HDFS is a reliable distributed file system and also used to store large amount of data. Hence, HDFS is used to store the huge amount of web services using MapReduce algorithm.

Algorithm: Storing Web Services in HDFS

Input: Web services to be stored

Output: Directory with files of web services

```
1.  Function Mapper(Key,Value)
2.      Path(Path of the web services .XML files to be stored)
3.      FileSystem.get(Configure file system)
4.      Output(OutputCollector, Reporter)
5.  End Mapper
6.  Function Reducer(OutputCollector, Reporter)
7.      Path(HDFS path)
8.      Output(File directory with HDFS files)
9.  End Reducer
```

According to the above algorithm , the mapper function configures file system of HDFS and the path of .xml files is given as the output. The reducer takes this output as the input and stores this .xml files in blocks in HDFS for further processing.

3.3 Storing the Web Service property in HBASE

HBASE [29] [30] stores the data in a table, which is a distributed, NoSql, column oriented database built on top of Hadoop Distributed File System (HDFS) and is different from the conventional relational databases. The data rows in a HBASE table are stored and sorted based on its row keys. The HBASE as a unique row key and varying arbitrary columns. The column in two different rows need not be similar. The column name is divided into column family and a column qualifier. The column families are added during the creation of the table and it's not changeable. The qualifiers are added or deleted dynamically as needed. The cell access of the HBASE is by its primary key and MapReduce jobs scans HBASE for data retrieval. The parallel execution of MapReduce jobs cause higher response time and also throughput. Hence we use, HBASE to store the functional and non functional property of the web service by using ontology tree, QoS table and also an Interface table. These properties are stored to specify the optimal service, hence the optimal search of web services will be efficient. HBASE clusters are efficiently managed by the one of the Apache subproject Zookeeper [32].

The QoS requirement by the user can be solved for simple web services by creating the QoS tree. In web browsing, the main aim is to select appropriate ontologies for given browsing in run time. Eventually, web pages will be linked or composed to other web pages whose contents may differ from the actual web page. So, in order to provide a better user experience, QoS ontology tree is created in a HBASE. Here, Strong Dominance and weak dominance between the web services property. If one web page as better QoS than the linked Web page, then that web page is considered as the strongly dominant or else weakly dominant. Based on this QoS tree is created where all the strongly dominated web services is stored in the left side of the tree and weak web services in the right with the index. Hence, forming the relationship between the parent node and the child nodes. During the MapReduce based web service search is performed, the properties of the web services is got from the HBASE and due to this, optimal web service is presented to the user, by filtering the unwanted web services.

3.4 MapReduce Algorithm for Web Service Selection and Composition

MapReduce [26][27][28] also uses master-slave architecture to process huge amount of data. MapReduce consist of one job tracker and multiple task trackers to process the huge amount of data. Job tracker is a master, which schedules the task tracker jobs. In MapReduce, instead of sending the actual data, just an computing is sent to reduce the network bandwidth. The task tracker periodically sends the report to the Job tracker. If the job tracker does not receive any report within certain time period then job tracker will treat that node as failure and will assign the job to different task tracker. MapReduce takes the input in the form of <key, value> pair. Later, this input is sorted and reduced by the reducer function. This core concept is applied to manage the web service and using map and reduce operation the optimal web service can be filtered out of several web services, Hence used for web service composition. Where MapReduce will structure all the web services by indexing it and later removing all the duplication and establishing an index. These structured web services are stored in HDFS for further processing.

Algorithm: Optimal Web Service Search (OWSS)

Input: User requirement

Output: Optimal best matched web service

```
1.  Function Mapper (Key,Value)
2.      Output(WebServiceName, AllTheMatchedWebService)
3.  End Mapper
4.  Function Reducer(WebServiceName, AllTheMatchedWebService)
5.      For(i: AllTheMatchedWebService)
6.          For(j: AllTheMatchedWebService)
7.              OptimalBestMatchedWebService();
8.      Output(BestMatchedWebService)
9.  End Reducer
```

The above MapReduce Algorithm provides optimal web service search for the simple web service request. The mapper function takes user requirement as the input and maps all the matched web services to the web service name asked by the user. This mapper output is fed to the reducer function, which runs the OptimalBestMatchedWebService() function by parallely matching two web services.

4. EXPERIMENTAL RESULT

The above mentioned algorithms have been evaluated based on the Hadoop platform in windows environment. The implementation is performed in the standalone computer of 6GB RAM, 2GHZ CPU. All the MapReduce algorithm have been implemented using Java 1.6. We evaluated our algorithm with the ECML PKDD Discovery Challenge 2008 [33] and compared the discovery of web service in Hadoop with the web service discovery without using hadoop. The web services upto 6000 is evaluated using the above algorithm. Since, hadoop has been used, which is a big data resolver tool, the response time of the proposed system is higher compared to the traditional web service. (See Figure 2).

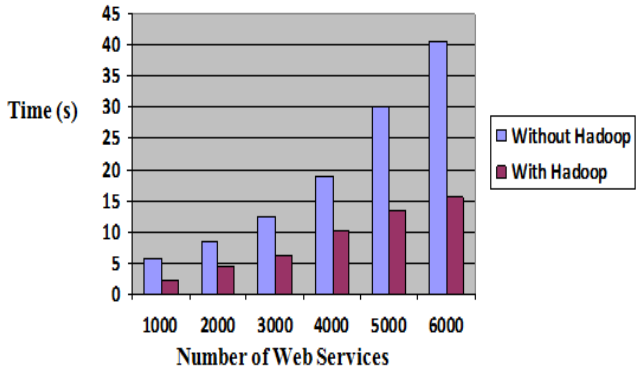

Fig 2: Response time for a User.

The response time for system without hadoop will increase gradually with the increase in the number of web services. But, with Hadoop there won't be much increase in the response time when the number of web services increase.

Table 1. Comparative result of web service with and without Hadoop

Number of Web Services	Without Hadoop (Sec)	With Hadoop (Sec)	Difference in time between two methods (sec)
1000	**5.5**	**2.2**	**3.3**
2000	**8.4**	**4.2**	**4.2**
3000	**12.4**	**6.2**	**6.2**
4000	**18.8**	**10.2**	**8.6**
5000	**30.1**	**13.2**	**16.9**
6000	**40.8**	**15.4**	**25.4**

Tabular comparative analysis of the web service with and without Hadoop is shown in Table 1 and graphical representation showing the peak point between the two systems is shown in the figure 3. Even though, the number of web services increases the time taken by the Hadoop based approach is less to a greater extent.

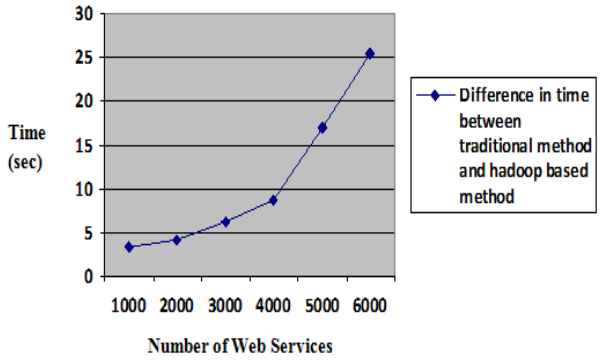

Fig 3: Graphical representation of difference in response time between the traditional system and Hadoop based approach

5. CONCLUSION

The response time decrease gradually with the increase in amount of data in a web service repository. Hence, due to the overwhelming popularity of web services cause the repositories to grow. Many traditional web service search engines failed to discover the optimal web services, when the repository was huge. Henceforth, to improve the performance of the web service discovery and the web service composition, a Hadoop based framework is considered to manage web service. The MapReduce based OWSS algorithm provides higher response time when compared to the traditional systems for simple web services. Even though, the number of web services increases the time taken by the Hadoop based approach is less to a greater extent. In Future, the complex web service are considered and separate map reduce algorithm will be implemented to provide the optimal web services.

6. REFERENCES

[1] Shashank Shetty, Shalini P.R, Aditya Kumar Sinha, "A Survey: The study and implementation of web services and managing it using Hadoop ecosystem, extending to web service selection", In Proc. National Conference on Multimedia and Information Security (NCMIS), 2014, pp. 73-78.

[2] Srinivasan, N., Paolucci, M., Sycara, K.P, "An Efficient Algorithm for OWL-S Based Semantic Search in UDDI", In: First International Workshop on. Semantic Web Services and Web Process Composition., 2004, pp. 96-110.

[3] UDDI, UDDI V1 Technical White Paper, Sep. 2000, http://www.uddi.org.

[4] D. Box, et al., Simple Object Access Protocol (SOAP) 1.1, W3C, May 2000, http://www.w3.org/TR/SOAP/.

[5] E. Christensen, et al., WSDL 1.1. Mar. 2001, http://www.w3.org/TR/wsdl

[6] Web Services, 2002. http://www.w3c.org/2002/ws.

[7] Alonso, G. Casati, F. Kuno, H. Machiraju,V, "Web Services: Concepts, Architecture, and Applications", Springer, New York (ISBN: 3540440089), 2003.

[8] S. Weerawarana, F.Curbera, "Business Process with BPEL4WS: Understanding BPEL4WS, Part 1", IBM corporation, 2002, pp.1-8.

[9] DAML-S, http://www.daml.org/services/daml-s/0.7/, 2002.

[10] N. Gholamzadeh, F. Taghiyareh, "Ontology-based Fuzzy Web Services Clustering", 5th International Symposium on Telecommunications, 2010, pp. 721-725.

[11] Binding point, http://www.bindingpoint.com/.

[12] Grandcentral, http://www.grandcentral.com/directory/.

[13] Web service list, http://www.webservicelist.com/.

[14] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. VLDB Endowment, 2004, pp. 372-383.

[15] Y. Zhang, Z. Zheng, M R. Lyu, "WSExpress: A QoS-aware search engine for Web services", in Proceedings of 2010 IEEE 8th International Conference on Web Services, ICWS 2010, July 5, 2010 - July 10, 2010, pp. 91-98

[16] U. Bellur, R. Kulkarni, " Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching", IEEE International Conference on Web Services, 2007, pp. 86-93.

[17] Zhu, X. L., Wang, B, "Web service management based on Hadoop", Proceedings of 8th International Conference on Service Systems and Service Management, ICSSSM'11, June 25, 2011 - June 27, 2011, pp.1-6.

[18] JAVA 1.6, http:// www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html.

[19] Eclipse Europa 3.3.2, http:// www.eclipse.org/downloads/packages/eclipse-ide-java-developers/europawinter.

[20] Cygwin, http://www.cygwin.com.

[21] Hadoop 0.20.1, http://archive.apache.org/dist/hadoop/core/hadoop-0.20.1/.

[22] Hadoop Eclipse Plugin, http://code.google.com/p/hadoop-eclipse-plugin/downloads/detail?name=hadoop-0.20.1-eclipse-plugin.jar&can=2&q=.

[23] Zookeeper, http://archive.apache.org/dist/hadoop/zookeeper/zookeeper-3.3.1/.

[24] HBASE, http://archive.apache.org/dist/hbase/hbase-0.20.6/.

[25] H. Wang, Haihong E, Xiulan Kong, "The Intelligent Hadoop-based Book Management System", 6th international conference on Pervasive Computing and applications (ICPCA), 2011, pp. 202-207.

[26] Tom white., "Hadoop: The Definitive guide".

[27] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Magazine communication of the ACM- 50th anniversary issue:1958, January 2008, pp. 107-113.

[28] H. Wang, Haihong E, Xiulan Kong, "The Intelligent Hadoop-based Book Management System", 6th international conference on Pervasive Computing and applications (ICPCA), 2011, pp. 202-207.

[29] HBASE, http://Hadoop.Apache.org/HBASE/

[30] Mehul Nalin Vora, "Hadoop-HBASE for Large-Scale Data", 2011 International Conference on Computer Science and Network Technology (ICCSNIT), 2011, pp.601-605.

[31] C. Franke, S. Morin, A. Chebotko , J.Abraham, P. Brazier "Distributed Semantic Web Data Management in HBase and MySQL Cluster", IEEE International Conference on Cloud Computing (CLOUD), 2011, pp. 105-112

[32] Apache ZooKeeper, http://zookeeper.Apache.org/

[33] Web Service data set, http://www.kde.cs.uni-kassel.de/ws/rsdc08/dataset.html