An Efficient Web Service Selection using Hadoop ecosystem based Web Service Management System (HWSMS)

[1]Shashank, [2]Shalini P.R, [3]Aditya Kumar Sinha
[1]PG Scholar, [2]Assistant Professor, [3]Principal Technical Officer
[1]Computer Science and Engineering,
[1]NMAM Institute of Technology, Nitte, Karnataka, India
[1]Centre for Development of Advanced Computing (CDAC),, Pune, India
[1]Shashankshetty06@gmail.com, [2]Shalini.pr.2007@gmail.com, [3]Saditya@cdac.in

***Abstract*— A Hadoop ecosystem based Web Service Management System (HWSMS) mainly focuses on selecting the optimal web service using the two algorithms Optimal Web Service Selection (OWSS) and Advanced Stop Words Based Query Search (ASWQS). The distributed management framework is the key research issue that still needs to be addressed in service management research field. As the part of this effort, this paper focuses on the study on web services and discusses on the Hadoop Web Service Management System (HWSMS) which manages web services using the big data problem resolver tool, Hadoop Ecosystem. Due to the growing popularity of Service Oriented Architecture (SOA) and the web technology, there has been an huge impact on the web service repository. The data on the network has been increasing in the daily basis, hence making it huge and complex. This causes a gradual decrease in the performance and does not fulfill the user requirement and also QoS metric. Hence, In the proposed method, web service management using Hadoop ecosystem has been discussed. The user query is processed using the Map Reduce Optimal Web Service Search (OWSS) algorithm and the optimal web service out of several web services is delivered to the service requestor by filtering the unnecessary web services. Hence, an efficient web service selection process can be carried out and better QoS to the customer. The experimental setup of HWSMS is implemented and the results are analyzed with the traditional System**

***Index Terms*— SOA ; Web service; Hadoop; MapReduce; Big data; Web service management; Web service Composition**

I. INTRODUCTION

Today, the amount of information that is currently available in internet is XML format which is growing is in a fast pace. Hence we can consider the web as the biggest knowledge base, which is made available to the public. Due to the convergence of many new technologies such as Service Oriented Architecture (SOA), high speed internet and web service [15] has given rise to a new kind of software developers: Web service providers These web service providers develop their application in such a way that it can function as technology independent and also they are reusable. These web services are accessible through service oriented architecture via internet.

The success of these web service providers mostly depend on the Quality of services (QoS) that they offer their customers and also meeting all the customer requirements. Hence, there is requirement of adequate service management, which is essential for their business. It also helps in fulfilling the current and future requirements of the customers and also to improve the QoS. Some of the issues that occur during the management of the web service are network failure, web service security and web service availability.

The paper mainly focuses on the two main issues and we treat these issues as a big data problem. One is the storage efficiency, that is when there is huge amount of requests from the customer, and then the web service repository must be efficiently searched. Hence this huge amount of data comes under big data problem. That is dataset which continuously grows and cannot be processed using the traditional relational database [5]. Henceforth, this web services cannot be managed using a traditional databases efficiently. That is traditional UDDI [2] service repository model cannot be used when the web services is large. Secondly, when there is a complex web services, then efficient selection of web services is a difficult task. Hadoop [6][9] is one of the tool used to tackle against big data problem [8]. Hence novel approach of managing web services using Hadoop is developed [7].

This approach overcomes the complexities which arise in traditional web service management methods. This paper discuses about the novel Hadoop based approach of managing a web service. Some of the Hadoop ecosystems like Hadoop Distributed file system (HDFS) [6], MapReduce [6] and Hadoop database (HBASE)[6] is combined together to manage the web services and also selecting a best optimal web service from the several web services. HDFS provides a reliable distributed storage of the web service, Map-Reduce retrieves optimum services by filtering the unwanted services and HBASE is used to store the functional and non functional property.

II. RECENT WORKS

In simple terms, the web service is the piece of the software application whose features are defined by the XML based language [42]. 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) [1] [2], Simple Object Access Protocol (SOAP) [3] and Web Service Description Language (WSDL) [4]. 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 [37] 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) [46] 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. Both DAML-S and BPEL4WS focuses on the representation of web service composition, where process and also binding of it is known priori.

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 [2] 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 [18] [19] [20] 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. [40] 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 [21] 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. [17] 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. [22] 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. 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.

Q. Yu et al. [39] discusses various research issues and solutions for deploying and managing web services. U. Srivastava et al. [14] in his work reviews about the web service management system [13] which allows multiple querying simultaneously in an integrated manner. M. Ouzzani et al. [16] introduced efficiently querying with web service. Proposed query model consist of three levels, query level which acts as a user interface for giving queries, virtual level for web service operations, and finally concrete level consist of all the related web services. Zheng et al. [41] provides a experimental evaluation on the real world dataset of web services when accessed across the country.

When the query request becomes huge then the response time will gradually decrease. So, In the proposed method we integrate web service management 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.

III. COMPARATIVE ANALYSIS OF WEB SERVICE MANAGEMENT METHODS

The comparative analysis between the traditional web service management methods with the proposed methods has been discussed in the Table 1. The comparative analysis briefly explains the advantages of the proposed methods upon traditional management methods.

Table 1 Difference between the traditional and the proposed method

Web Service Management Methods	Description	Advantages	Disadvantages
Traditional UDDI [2] and SOAP [3] based architecture	Most of the traditional web services management methods [18] [19] [20] [23] [40] relies on the UDDI and SOAP based architecture.	1) Efficient management of web service and increase in response time when it comes to small task or fetching web service from smaller data	1) These traditional distributed architecture provides a loose coupling between various components and are sensitive. 2) As the rate in the business

	The SOAP which is a framework that gives the foundation for messaging and upon which various web services are built.	repository.	change increase and large numbers of user delegate their task into the system, the problem arises. Those are web sites unavailable or unresponsive,
Proposed Hadoop based web service management architecture	The proposed Hadoop based architecture is integration of the big data problem resolver tool Hadoop ecosystems with the web service. The proposed MapReduce algorithm is used to provide the optimal web services by searching the huge amount of web services from the HDFS.	1) Since the Hadoop works on large amount of data, the large repository can be efficiently searched to provide the optimal web services by filtering the unwanted web services. 2) Higher QoS when compared to the traditional architecture when processing terabytes of data	1) When the web services are small, the traditional architectures can be used instead of Hadoop. But nowadays we don't find such small repositories.

IV. Problem Statement and Implementation

The core framework of the Hadoop based web service management is similar to the reference [7]. Due to the overwhelming popularity of a web service, there has been a huge amount of web request and can be treated it has a big data. Hence the web service architecture is integrated with the Hadoop ecosystems like MapReduce, HDFS and HBASE to manage it more efficiently to obtain the necessary QoS metrics.

The managing web service using Hadoop ecosystem framework has four major components (fig 1):

A. Infrastructural Setup of Hadoop

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

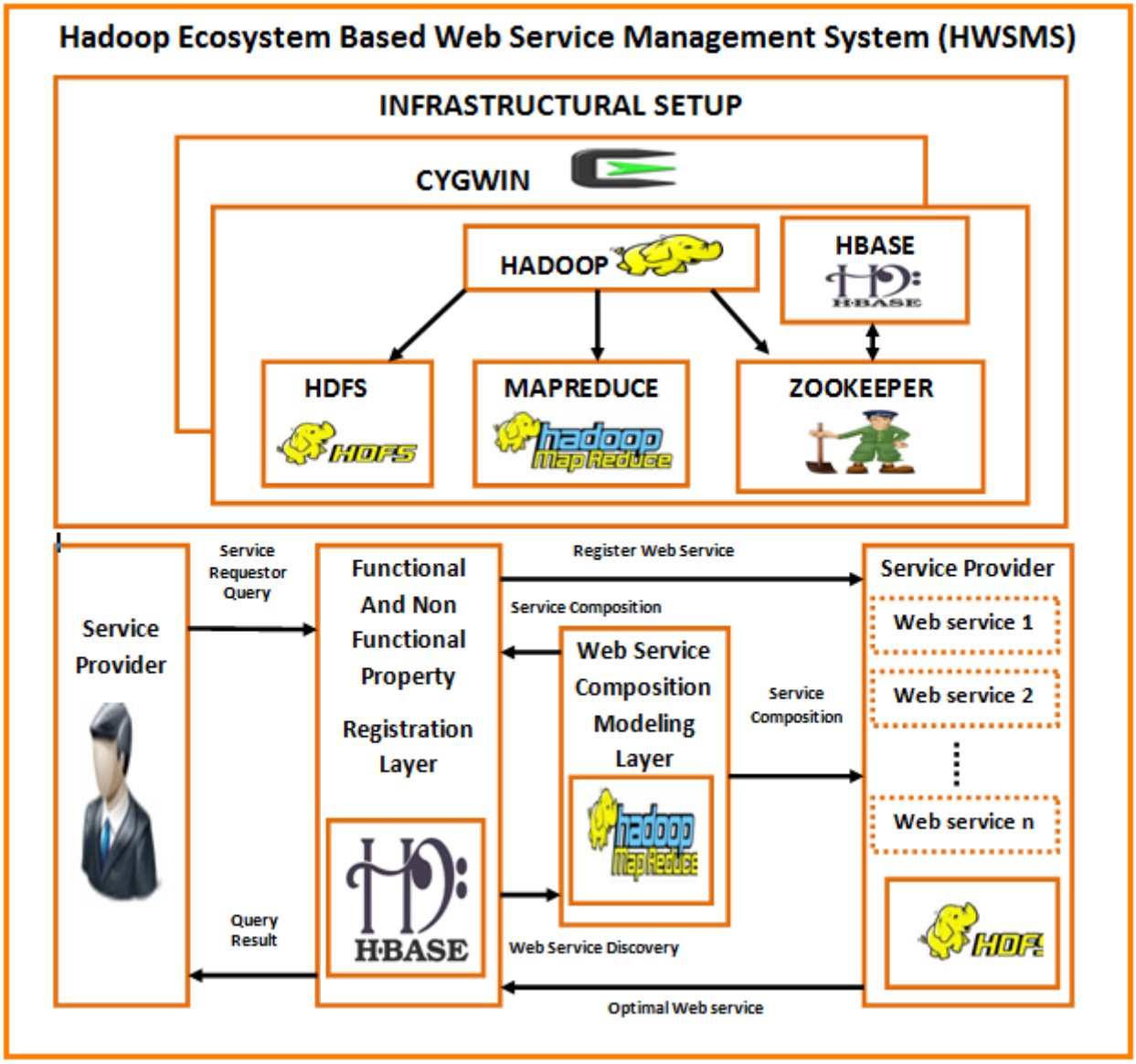

Fig 1: Hadoop ecosystem to manage and select web services

1. Downloading and installing Java 1.6 from http:// www.oracle.com [48], Eclipse Europa 3.3.2 from http:// www.eclipse.org [49] and Cygwin from http://www.cygwin.com [50] by configuring and starting SSH Daemon.
2. Download Hadoop from http://archive.apache.org/ [51], copy it in Cygwin home folder, and unpack Hadoop
3. Configure Hadoop by editing the configuration file hdfs-site.xml, mapred-site.xml and core-site.xml . Also install Hadoop eclipse plug-in [52] and change the eclipse java perspective to MapReduce environment.
4. Download and install Zookeeper [53] and HBASE [54] from http://archive.apache.org/, Setup Hadoop location in eclipse and start all the cluster. Now, Hadoop ecosystem is setup for any task.

B. Service Provider

The Service provider is integrated with HDFS to provide a service to a Requestor entity. HDFS [6] [35] is used for distributed storage of data to solve the problem of storing big data. HDFS mainly represented by Name node and by Datanode which are master and slave node respectively. HDFS consist of one Namenode, one secondary node and many Datanodes. The Namenode is used to store metadata of a file system, providing user access permission to the user. Secondary Namenode acts has a backup to the master Namenode. All the data are stored in the Datanode, and replication of data is also done to three different nodes to achieve copy storage policy. Periodically, data node has to be in contact with the master node. when user wants to read from HDFS then request is sent to the Namenode. Namenode gets the node, file block information from the metadata table by querying it. These information is sent back to user and user can directly have an access to the Datanodes. When user wants to write or store in a HDFS, firstly request is sent to the master node then the master node divides a file into blocks and allocates it to the Datanode. Later, writes file name to the namespace and then sends all the metadata information to the user, so that data can be written to the Datanode. So this HDFS is applied to store large amount web service. This core concept of Namenode and Datanode is applied to the web service to store it and manage it efficiently with QoS service satisfaction. 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.

C. Functional and non functional property registration layer

HBASE [33] [36] is an distributed, NoSql and an column oriented database, which s built on top of Hadoop distributed file system. HBASE allows random read/ write access to its database and is organized in tables, which are labeled. HBASE Consist of six main components, i.e., table, row, column family, column qualifier, cell, version. A cell in a HBASE always consist of column name and column family. Hence the main advantage of this is that a program can always understand, what kind of data it contains in the cell. Characteristics of HBASE can be applied to loose structure data because HBASE has varying column and row have sortable key [7]. HBASE clusters are efficiently managed by the one of the Apache subproject Zookeeper [37]. 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.

Algorithm: Creating Table and inserting values in HBASE Table

Input: Table Name and the values to be Inserted

Output: HBASE Table with Web services

```
1.  Function CreateTable(TableName)
2.          Create a Hbase configuration object 'hc'
3.          HBaseConfiguration hc = new HBaseConfiguration()
4.          Create a Hbase table descriptor object 'ht' with the table name
5.          HTableDescriptor ht = new HTableDescriptor("TableName");
6.          Add two column family 'DomainName' and 'PathName'
7.          ht.addFamily( new HColumnDescriptor("DomainName"));
8.          ht.addFamily( new HColumnDescriptor("PathName"));
9.          Create a Hbase Admin object 'hba' to get the access permission for the hbase configuration object 'hc'
10.         HBaseAdmin hba = new HBaseAdmin( hc );
11.         Create a Table  'ht' using Hbase Admin object
12.         hba.createTable( ht );
13.         WebServices(TableName);
14. End CreateTable
15. Function WebServices(TableName)
16.         Create a Hbase object 'h' to access the created table
17.         Hbase h=new Hbase();
18.         Insert n number of query requirements and web file path to be displayed
19.         String QueryRequirements="gmail"
20.         String Webfile="www.gmail.com/index.html";
21.         h.Insert("unique row key", QueryRequirements, Webfile);
22. End WebServices
23. Function Insert(Row, Req, Webfile)
```

```
24.          Convert a unique row key into byte format to protect it from the outside users
25.          Create a table object 'table' and insert the values into the table.
26.          table.put(Row,Req,Webfile);
27.  End Insert
```

Algorithm mentioned above produces the ontology table of domain name and the path name in the HBASE. This table acts as a reference for a Search algorithm to search from the HDFS. The function CreateTable() creates a table with the domain name and pathname with unique row id for the table. The function WebServices() with function Insert() allows to insert 'n' number of domain name and path name of the web services.

D. Web service Composition modelling layer

The MapReduce [6][30][35] also performs master slave operation that is by using a single job tracker and many task tracker. Job tracker is a master which looks after the scheduling policy of job to the task tracker. The main advantage of MapReduce is it transfers computing not the actual data. Hence there is the reduce in the network bandwidth and data transfer is made economical. The jobs are usually submitted to the job tracker and this job is in turn given to the task tracker. The task tracker as to periodically send the report back to the job tracker. If the job tracker does not receive any report with some stipulated time period then the job tracker will assume that the task tracker is failed, hence that job is given to the other task tracker. The main functionality of the MapReduce is it maps the input with some key values (i.e., < key, value> in our proposed system it will be < web links, web data >) using mapper function. Later, the mapped 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 OWSS()
2.       Get the user requirement using the scanner object
3.       Display("Enter  Domain/file u need: ");
4.       name=Scanner.nextLine();
5.       Using the hbase object get the required web service
6.       file=Hbase.GetOneRecord(TableName, UniqueRowKey, name);
7.       Read the selected web service from the HDFS
8.       Path(HDFS path)
9.       Output(File directory with HDFS files)
10. End OWSS
11. Function GetOneRecord(TableName, UniqueRowKey, name);
12.       Configure the table object 'table'
13.       HTable table = new HTable(tableName);
14.       Get all the web data from the table using the unique row key
15.       Result rs = table.get(UniqueRowKey.getBytes());
16.       Scan whole HBase using the HBase KeyValue 'kv'
17.       For(KeyValue: rs.raw())
18.            Get each column qualifier from the raw HBase and then store in 'q'
19.            String ColumnQualifier=new String(KeyValue.getQualifier());
20.            Compare the each column qualifier name with the user requirement web service 'name'
21.             If(ColumnQualifier==name)
22.             Display(KeyValue.getValue()));
23.             End If
24.             Display("No such web service, try with different query);
25.       End For
26.  End GetOneRecord
```

The above Optimal Web Service Search (OWSS) Algorithm provides optimal web service search for the simple web service request. The function OWSS() takes the request from the service requestor and sends the request of getting the optimal web service to the function GetOneRecord(). The GetOneRecord() function using the UniqueRowKey of the table gets the all the value from the table created in the HBASE. Then compares each domain name of the extracted values from the table with the query requested and displays the optimally matched web service.

Algorithm: Advanced Stop Words based Query Search (ASWQS)

Input: User requirement with Stop Words

Output: Optimal best matched web service

```
1.  Function ASWQS()
2.      Get the query with the stop words as the input using the Scanner object
3.      Display("Enter  Domain/file u need: ");
4.      UserRequest=Scanner.nextLine();
5.      Create a request query object and call the function 'StopWordsRemoval'
6.      Result=ReqQuery.StopWordsRemoval(UserRequest);
7.      Split the result further and store it in a array 'WithoutStopWords[]'
8.      WithoutStopWords[N]=Result.split();
9.      For(I=0 to N-1)
10.         Get the matching Web Service
11.         file=Hbase.GetOneRecord(TableName, UniqueRowKey, WithoutStopWords[I]);
12.         Read the data from the input buffer
13.         Path(file);
14.         Output(File directory with HDFS files);
15.     End For
16. End ASWQS
17. Function StopWordsRemoval(UserRequest);
18.     StopWords[]={All possible stop words like 'I', 'need', 'a', 'an', ... , etc};
19.     Split the given UserRequest and store in the array 'Words[]'
20.      Words[N]=UserRequest.split();
21.      For(j=0 to N-1/Words.length)
22.          For(i=0 to StopWords.length)
23.              Compare the split words with the Stop words
24.              if(Words[j]≠StopWords[i])
25.                  The Functional words which are filtered are stored in the array 'Request[]'
26.                   Request[]=Words[j];
27.              End If
28.          End For
29.      End For
30. Return Request[];
31. End StopWordsRemoval
```

The advanced stop words based search algorithm is used when the requested query has some non functional words or stop words. For example: I need a gmail. Where "I", "need", "a" are stop words and "gmail" is the functional word. These stop words cause the problem during the matching of web service and produces more number of search result instead of optimal web search. The ASWQS() function takes the user query with the stop words and splits the query into words. Later, removes the stop words by passing to the StopWordsRemoval() function and searches the optimal web service.

V. EXPERIMENTAL SETUP

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 web services scaling from 1000 to 6000 and compared the discovery of web service in Hadoop with the web service discovery without using Hadoop. The web services up to 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. (Fig. 2).

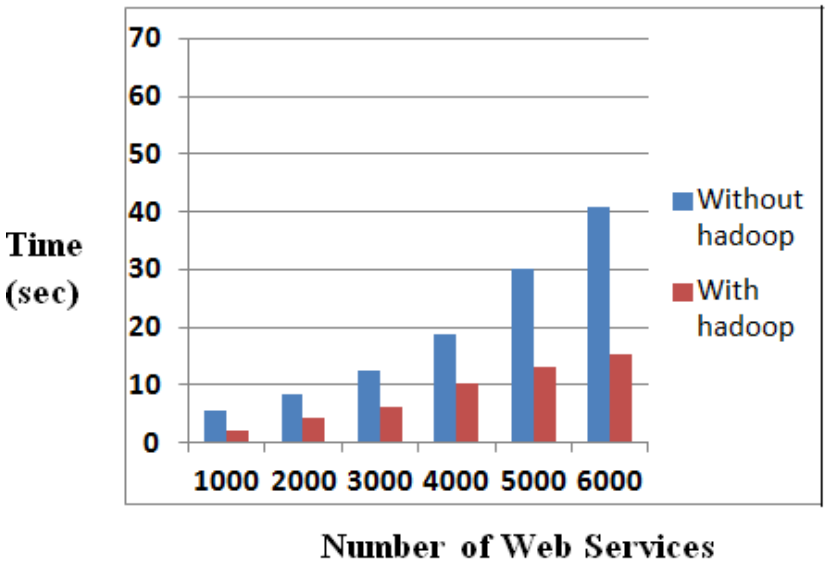

Fig 2: Response time for a User.

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. 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.

Table 2. Comparative Analysis 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

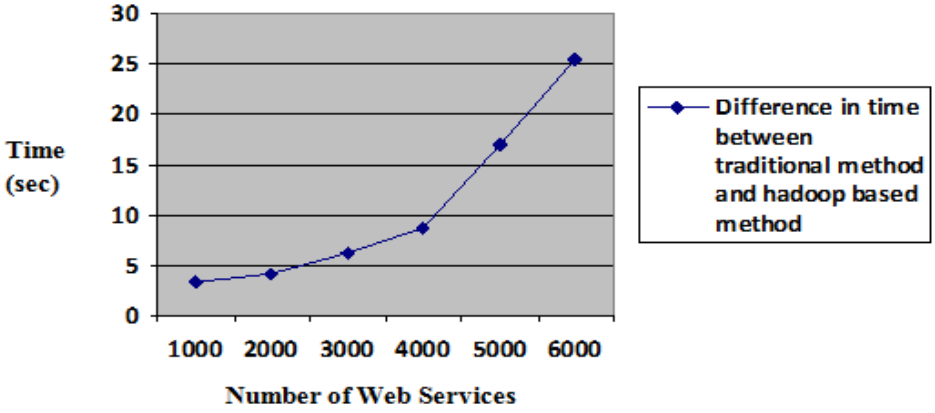

Fig 3: Graphical representation of difference in response time between the traditional system and Hadoop based approach

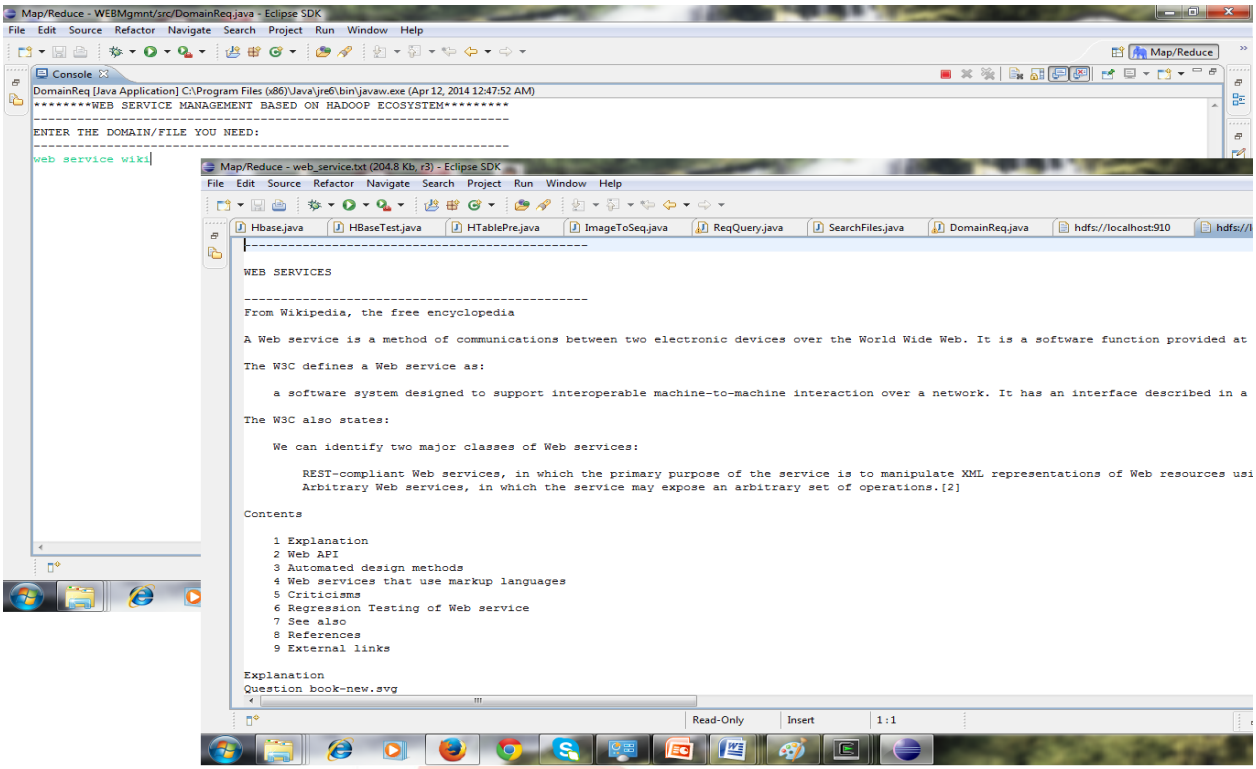

Fig 4. Snapshot of web service request by the user for OWSS algorithm

The fig 4. shows the snapshot of the web service request "web service wiki" for the OWSS algorithm. The HWSMS system retrieves the optimal web service stored in the HDFS by comparing with the ontology table stored in the HBASE.

Table 3 Comparative Result of OWSS and ASWQS Algorithm

Number of queries	Optimal Web Service Search (OWSS) Algorithm for queries without stop words		Advanced Stop Words based Query Search (ASWQS) for queries with stop words	
	Positive Result	Negative result	Positive Result	Negative result
100	82	18	87	13

The OWSS and ASWQS algorithm was tested by giving 100 queries without stop word and with stop word respectively. The positive result obtained for 100 queries is 82 and negative result of 18 for OWSS algorithm and 87 positive result and 13 negative result for ASWQS algorithm. The Positive result can be increased by adding more and more number of web services to the HDFS and improving the Ontology relationship in the HBASE.

VI. Conclusion

The Due to the popularity of Service oriented architecture, there has been significant growth in data in the web repository. Hence to manage this web services and efficiently search a optimal web services is a major issue. In a Proposed method, the Hadoop Web Service Management System (HWSMS) provides a method to manage web services using the Hadoop ecosystem and by using OWSS and ASWQS algorithm proposed the web service selection is carried out. The result shows that the web service in traditional Myself server mechanism takes longer time due to the overloading of web service request to the server. The response time of both with Hadoop and without Hadoop system increases with the increase in the number of web services in the repositories. But, response time of HWSMS is less compared to the traditional system. The OWSS and ASWQS algorithm is evaluated with query requests and both the algorithm have more positive result. Hence, the Web service management based on Hadoop ecosystem proves to be more efficient than the traditional methods..

References

[1] 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.
[2] UDDI, UDDI V1 Technical White Paper, Sep. 2000, http://www.uddi.org.
[3] Box, et al., Simple Object Access Protocol (SOAP) 1.1, W3C, May 2000, http://www.w3.org/TR/SOAP/.
[4] Christensen, et al., WSDL 1.1. Mar. 2001, http://www.w3.org/TR/wsdl.

[5] S Singh., N Singh, "Big Data Analytics", International Conference on Communication, Information & Computing Technology (ICCICT), 2012, pp.208-214

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

[7] 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.

[8] A B. Patel, M Birla, U Nair, "Addressing Big Data Problem Using Hadoop and MapReduce", Nirma University International Conference on Engineering, NUICONE, 2012, pp.1-5

[9] Apache Hadoop, 2013, http://Hadoop.Apache.org

[10] L. An, J J. Jeng, "Web service management using system dynamics", Proceedings of the IEEE International Conference on Web Services (ICWS'05), 2005, pp.347-354

[11] Ludwig, A. Keller, A. Dan, R P. King, R. Franck, "Web Service Level Agreement (WSLA) Language Specification", Journal of network and systems management, 2003, pp 57-81

[12] Web Service Management System (WSMS), 2005, http://infolab.stanford.edu/wsms/

[13] U. Srivastava, K Munagala, J. Widom, R. Motwani, "Query Optimization over Web Services", VLDB '06 Proceedings of the 32nd international conference large databases, 2006, pp. 355-366

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

[15] M. Ouzzani, A. Bouguettaya, "Efficient access to web services", IEEE Internet Achievement of web service managing and selection Computing, 2004, pp. 34-44.

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

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

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

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

[20] 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.

[21] 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

[22] Sreenath R.M., Singh M.P, "Agent-based service selection", In Web Semantics: Science, Service and Agents on the World Wide Web, 2004, pp. 261-279.

[23] D B. Claro, P. Albers, J K. Hao, "Web Services composition", Semantic Web services, Processes and Applications, 2006, pp. 195-225.

[24] Cardoso J., Sheth A., Miller J., Arnold J., Kochut K, "Quality of Service for Workflows andWeb Service Processes", In Web Semantics: Sciences, Services and Agents on the World Wide Web, 2004, pp. 281-308.

[25] D. Booth et al, Web Service Architecture, 2004, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[26] HIVE, http://Hadoop.Apache.org/HIVE/

[27] Hadoop Distributed File System (HDFS) Design, http://Hadoop.Apache.org/core/docs/current/hdfs_design.html

[28] Mike Barlow, Real time data analytics, Emerging Architecture, February 2013, First edition.

[29] 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.

[30] PIG, http://Hadoop.Apache.org/pig/

[31] SQOOP, http://Hadoop.Apache.org/sqoop/

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

[33] MAPREDUCE, http://Hadoop.Apache.org/mapreduce/

[34] 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

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

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

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

[38] Q. Yu, X. Liu, A. Bouguettaya, B. Medjahed, "Deploying and managing Web services: issues, solutions, and directions", The VLDB Journal, May 2008, pp. 537-572.

[39] 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.

[40] Z. Zheng, Y. Zhang, Michael R. L, " Distributed QoS Evaluation for Real-World Web Services", IEEE International Conference on Web Services, 2010, pp. 83-90

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

[42] P. Mier, M. Mucientes, M.Lama, "Automatic web service composition with a heuristic-based search algorithm", International Conference on Web Services, 2011, pp.81-88

[43] S. Kona, A. Bansal, M. Blake, G. Gupta, "Generalized semantics-based service composition", IEEE International Conference on Web Services, 2008, pp. 219–227.

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

[45] JAVA 1.6, http:// www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html.
[46] Eclipse Europa 3.3.2, http:// www.eclipse.org/downloads/packages/eclipse-ide-java-developers/europawinter.
[47] Cygwin, http://www.cygwin.com.
[48] Hadoop 0.20.1, http://archive.apache.org/dist/hadoop/core/hadoop-0.20.1/.
[49] Hadoop Eclipse Plugin, http://code.google.com/p/hadoop-eclipse-plugin/downloads/detail?name=hadoop-0.20.1-eclipse-plugin.jar&can=2&q=.
[50] Zookeeper, http://archive.apache.org/dist/hadoop/zookeeper/zookeeper-3.3.1/.
[51] HBASE, http://archive.apache.org/dist/hbase/hbase-0.20.6/.