Asian Journal of Information Technology 15 (16): 2908-2926, 2016
ISSN: 1682-3915

Web Mining for Potentially Value Added Services

[1]M. Akila Rani and [2]D. Shanthi
[1]Department of Computer Science and Engineering, NPR College of Engineering and Technology,
[2]Department of Computer Science and Engineering,
PSNA College of Engineering and Technology, Natham, Dindigul, Tamil Nadu, India

Abstract: The study focuses on Hadoop-based approach for web service management. Due to small amount of web service repository, performance of traditional web service discovery approaches does not offers relevant services based on user given query. To carry all those complications and also focuses on discovering potentially value added web services define architecture as WSDH (Web Service Discovery using Hadoop). Hadoop is a framework which is designed to support processing of large data sets in distributed computing environment. We propose a service discovery approach which retrieves relevant web services based on the user query. In which, we first perform scheduling. In scheduling, schedule the user given query based on time and then logical query execution tree is constructed. After that, preprocessing the relevant data which are maintained in hadoop frame work. To offer a deeper understanding of time series data we use deep learning technique. In deep learning the relevant data are maintained in Hbase-OpenTSDB. So efficiently merge time series data into open TSDB. We assess TF-IDF to calculate matching using Map/Reducer frame and similarity is calculated using vector space model. Comparing with the existing system proves that the proposed system provide an optimal solution for user query with minimal search time and maximum accuracy.

Key words: Web service discovery, Hadoop-HDFS, HBase (Open TSDB), map reduce, India

INTRODUCTION

Recent times have seen an explosive growth in the availability of various kinds of data. So it require massive amount of computation. When interacting with the web, finding relevant information from the large dataset is difficult. A person either browses or uses the search service when they want to find specific information on the Web. When a user uses search service inputs a simple keyword query and the query response is the list of pages ranked based on their similarity to the query (Alonso *et al*., 2003). However, today's a search tool have the following problems. The first problem is low precision which is due to the irrelevance of many of the search results. This result is difficult to find relevant information. The second problem is low recall which is due to the inability to index all the information available on the Web. This results in a difficulty finding the un indexed information that is relevant. Web mining techniques could be used to solve the information overload problems above directly or indirectly. However, we do not claim that web mining techniques are the only tool to solve those problems (Gunasri and Kanagaraj, 2014). Other techniques and works from different research areas such as Database (DB), Information Retrieval (IR), Natural Language Processing (NLP) and Scheduling Tang *et al*. (2015) and Amotz *et al*. (2014) could also be used. Semantic web has attached machine-interpretable information to the web content for achieving high accuracy. The enormous amounts of information are handling in web without any confusion by using the semantic web. The web services present in web are decoupled in nature. The semantic web services are employed to sense acute web services such as robotic discovery, composition, invocation and interoperation. Descriptions of web services are manipulated to discover various kinds of web services (Dogan and Ozguner, 2002). So, there is a need to build an automated system to manage web service by selecting an optimal web service by filtering the unwanted web service and also automated web service composition. To improve the performance of web service management we propose a scheduling with tree based approach for efficient management for scale multiple users. Hadoop is one of the big data problem resolver tool which is used to efficiently manage the web service and map reduce algorithm is used to automatically choose the optimal web service (Wang *et al*., 2013; Shetty *et al*., 2014). To improve the performance of Web service management, we propose a

Corresponding Author: M. Akila Rani, Department of Computer Science and Engineering,
NPR College of Engineering and Technology, Natham, Dindigul, Tamil Nadu, India

Hadoop based approach for efficient management of large scale Web services, where Hadoop can overcome the drawback which occurs in the traditional Web service management infrastructure (Shashank *et al*., 2014). There are three components are integrated into our approach, HDFS, HBase and Map Reduce where an HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing (Xie *et al*., 2010). In HBase table and a non-functional property index mechanism is designed to strengthen the retrieving performance of the functional and nonfunctional properties of Web services. HBase makes an excellent choice for applications such as Open-TSDB because it provides scalable data storage with support for low-latency queries. It's a general-purpose Open-TSDB using deep learning. It is a distributed; Time Series Database (TSDB) written on top of HBase (Agrawal *et al*., 2014). Deep learning algorithms have received significant attention in the last few years. Their popularity is due to their ability to achieve higher accuracy than conventional machine learning in many research areas such as speech recognition, image processing and natural language processing. However deep learning algorithms require a large amount of computational power and significant amount of time to trained. We aim in this work fetch time series data in to Open-TSDB using Deep learning.

Moreover, a TF-IDF algorithm based on Map Reduce is proposed to achieve the optimal service to satisfy user's requirement. To overcome above discussed issues, the Proposed architecture is defined as WSDH (Web Service Discovery using Hadoop). Generally, in map reduce framework use random shuffle procedure for discovering services (Lu *et al*., 2014). But in this study, we propose efficient solution for scheduling technique instead of the random shuffle manner and clustering. Performance analysis measurement is conducted between existing web service discovery (Crasso *et al*., 2008) and proposed WSDH. From that result, we prove our proposed defined architecture achieves more efficient result when compared with the previous algorithms, methods and techniques. Our proposed work is focused towards the management of multiple user requests and retrieving relevant results from Hadoop. Overall contribution of our proposed research is mentioned below:

- Creative way of scheduling the user queries based on construction of tree
- Deep Learning idea is involved for the purpose of storing data in time series over the Hadoop HBase-Open TSDB
- Matching and similarity is performed in Map Reduce by TF-IDF and Cosine Similarity function by which we rank the queries with their scoring values

Literature review: Web service discovery is an important issue for Web service management. Researchers discusses (Amotz *et al*., 2014) the overall periodic scheduling and tree. In this study, space and length of cycle is very expensive for n users and tree is a cyclic graph. By motivated the goal of minimizing waiting time, much research has focused on scheduling which is not perfectly periodic using average measure as target function. In this study, researchers considered several aspects of a tree scheduling problem. First, we presented a constant amortized time algorithm for choosing the next client to scheduling, given a schedule tree, using only polynomial space for most practical trees. Second, researchers considered a problem of constructing good schedule trees and describe an exponential time algorithm to find the optimal schedule tree. Speculative execution in the case of map reduce is "Improve response time" of a job by relaunching delay processing tasks. Crucially, speculative execution improves job response time in large and heterogeneous cluster. There are two important issues involved in speculative execution. First one is deciding which of the currently executing tasks are progressing slowly relative to other tasks. Secondly deciding a node on which a slow task must be relaunched. Problem with this speculation execution user searching time gets increased due to relaunching the task. Periodic scheduling is introduced for n number of user's. Perfectly periodic scheduling is defined as "how to predict the good schedules". Let we take the set of user requests {q1 ,q2, ..., qn} where each qi represents the fraction of the bandwidth requested by user i, i.e., $\sum_{i=1}^{n} q_i = 1$. Given this point, an algorithm computes a perfectly periodic schedule that matches the users as "closely" as possible. The scheduling is implies a period B_i and a $b_i = 1/B_i$. Measuring the goodness of a schedule is done based on the ratio of the requested shares to the granted shares qi/bi. Depending on the target application, the weighted average and maximum ratios are concerned. Formally, for each i, let $p_i = q_i/b_i$ where pi's define the performance measures:

$$\text{Maximum: } MAX = \max \{p_i | 1 \le i \le n\},$$

$$\text{Weighted average: } AVE = \sum_{i=1}^{n} q_i p_i$$

Next, based on the optimal algorithms, we develop a few efficient (good polynomial time) heuristics algorithms. These heuristics produced by the uniform distribution and zipf's distri bution under both the MAX and AVE measures. But perfect periodic scheduling is not always possible (Amotz *et al*., 2004). The study (Tang *et al*., 2015) proposed SARS (Self Adaptive Reduce Scheduling) for reduce tasks start times in Hadoop system. An optimal reduce scheduling is reduce task by dynamically allocating the jobs. Hadoop introduces the greedy strategy to schedule the reduce tasks and to perform very fast manner. SARS algorithm schedules the reduce tasks in optimal time point. Using this optimum point we can reduce the system delay and to increase the resource utilization. In this study, performance of the algorithm is improved by the reduce tasks start time and job completion time. To improve performance further more map reduce technique is used in hadoop frame work (Mohan and Remya, 2014). In this study discussed about "Factors affecting by the job performance". To increase the job performance and effective resource management is a fundamental need in Map Reduce technique. The following factors are affecting job performance: job scheduling, speculative execution, data locality issue, task assignment, admission control. To overcome these problems MTSD (Map Reduce task scheduling algorithm for deadline constraints) algorithm is used. The new mechanism (Xie *et al*., 2010) distributes fragments of an input file to heterogeneous nodes based on their computing capacities. The approach improves performance of Hadoop heterogeneous clusters. In this system, performance is not improved due to clustering algorithm in hadoop. In view of this study, performed speculative execution to balance the performance tradeoff between a single job and a batch of jobs (n number of jobs). Delay scheduling is improve the data locality but at the cost of fairness. So, researchers proposed a technique called slot rescheduling that can improve the data locality but with no impact on fairness. Finally, by combining these techniques together, a step-by-step slot allocation system is introduced. Dynamic map reduce technique can improve the performance of map reduce workloads substantially. Hadoop scheduling model is a Master/Slave (Master/Worker) cluster structure. The master node (Job tracker) coordinates the worker machines (Task tracker). Job Tracker is a process which manages jobs and Task Tracker is a process which manages tasks on the corresponding nodes. The scheduler resides in the Job tracker and allocates Task Tracker resources to running tasks: Map and Reduce tasks are granted independent slots on each machine. While performing the clustering in hadoop, the relevant datasets are unable to retrieve because of grouping the datasets. Drawback in this system it's only applicable for clustering environment.So this system is not used for when multiple users access the web services (Pakize, 2014). In hadoop multiple schedulers are introduced but each scheduler has some limitations Researchers Shashank and colleagues introduce a Hadoop ecosystem based on Web Service Management System (HWSMS) which is mainly focuses on selecting the optimal web service. There are two algorithms gets implemented in this paper. First one is Optimal Web Service Selection (OWSS) and another one is Advanced Stop Words Based Query Search (ASWQS). In (Shetty *et al*., 2014), the map reduce algorithm is used for web service composition and discovery. For automatically controlling interaction between the web services is achieved by using the DAML-S. By using the DAML-S languages the autonomous web services are discovered which was discussed in (Gholamzadeh and Taghiyareh, 2010) Ouzzani introduced efficiently querying with web service. Fuzzy clustering is proposed and probability vector is used for retrieving the semantic web services (Alonso *et al*., 2003) Proposes in clustered environment to overcome the task assignment problem using task selection algorithm. In this research, discussed the HBASE. It is database that stores the weights of individual web services on the top of the HDFS. In some situations two web services have the same weight that time we cannot to store the information in database. This frame work to analyze Open-TSDB for storing information based on the time series (Agrawal *et al*., 2014).

Hadoop allows user to compose the job, submit it, control its execution and query problem the state. Most of the job contains individual tasks and all the tasks need to have a machine slot to run in Hadoop all scheduling and allocation purpose are made on a task and node level for both the map and reduce phases. Lu and Liu (2013) designed the mapping mechanism for k nearest neighbor join for large dataset using divide and conquer approach. Kamal and Anyanwu (2010). In this study for dealing with deadline requirements in Hadoop-based data processing. Presenting the design of constraint-Based Hadoop Scheduler that takes user deadlines as part of its input and determines the schedulability of a job based on the proposed job execution cost model. Studies by Anuraj and Remya and Hesami and colleagues discussed semantic web service composition by using the clustering and ant colony optimization algorithms. The main goal of this paper is to provide a unique web service has huge joining capability to meet customer requirement. Connecting various web services and provides high level abstraction. Numbers of issues are arising in traditional

composition methods such as reduction of accuracy, increase of response time and unselect optimal composition. The OWL-S languages are used to describe the web services. After describing these services perform cluster formation. The relevant services are grouped into form cluster. The clusters are maintains in repository. The services are retrieved based upon the user given query. For identify the optimal or best set of services use Ant Colony Optimization (ACO) algorithm which is having excessive integrate ability can derived at optimum solution for each of the data used, i.e., number of mappers per stage, the number of ants per mapper and the number of stages in the chained job. More studies can be done in order to find a better way to implement the pheromone updates. Because it was unable to share pheromone updates between mappers in Hadoop, in the current implementation all the pheromone updates are done by the single reducer in each stage. So, the reducer has too much work to do as the number of ant's increases. This behavior can be improved by studying the possibility of having multiple reducers for pheromone updates and reducing the duplication of research. In web service discovery UDDI gives service registry, SOAP (Simple Object Access Protocol) is a foundation framework for communication of web service. The WSDL is utilized to define the web service and DAML's is an ontology web service, (BPEL4WS) Business Process Execution Language for web service to develop various processes, these are not understandable by the machine; by using the semantic web solve that issue. Semantic web has attached machine-interpretable information to the web content for achieving high accuracy. The enormous amounts of information are handling in web without any confusion by using the semantic web. The web services present in web are decoupled in nature. The semantic web services are employed to sense acute web services such as robotic discovery, composition, invocation and interoperation (Ruben *et al*., 2003). Only users who can enter the system will get the accurate recommendations. This fails to give accurate recommendations when the multiple users enters the system and requires personalized recommendation list for Ranking based on the query execution time (Padmapriya and Appandairaj, 2015). Dimitrios Skoutas, Dimitris Sacharidis, Alkis Simitsis and Timos Sellis, discuss the "Ranking Technique" for web services (Skoutas *et al*., 2010). As the web is increasingly used not only to find answers to specific information needs but also to carry out various tasks, enhancing the capabilities of current web search engines with effective and efficient techniques for web service retrieval and selection becomes an important issue. Ranking methodology provides the relevant services for a given user request. NLP and clustering algorithms were used in service discovery. Every Semantic Web Service Definition (SWSD) framework needs semantic languages to describe the web services. In this study, they consider the WSMO and OWL languages. Each web service has semantic description. The semantic description are first extracted the words that is constitute as context of words. The context contains the information of noun, name of operations. After extracting words perform disambiguation and form cluster. User query also extracted, disambiguated after that form cluster. Clustered terms are maintained in some order called ranking. Matching will be applied to discover relevant services from the repository based on user given query. Web services are discovered based upon which one has highest priority. Natural Language Processing (NLP) employed keyword based matching for matching algorithm with web service context. It gives accurate matching because for word extraction process use the Wordnet tool, that has high word senses (Gunasri and Kanagaraj, 2014). The proposed NLP with clustering gives efficient result when compared with existing general NLP method. Discuss the Score based Web Page Ranking Algorithm in following studies (Sanjay and Kumar, 2015; Aditi, 2014; Xu *et al*., 2008). In page ranking algorithm consider the number of users visits the particular web page. So, in this algorithm uses ranking the web pages based on user's visits.

Problem definitions: Clustering in hadoop is a challenging task to manipulate larger datasets. A cluster contains a large pool of datasets ranging from a few hundreds to thousands since the final output is generated from the clusters which leads accuracy problem. Whenever a new query is added in cluster identifying relevant query will be a tedious research (Pakize, 2014). In hadoop multiple schedulers are introduced but each scheduler has some limitations and disadvantages. FIFO scheduler is designed only for single type of job based on the job priorities in first-in first out manner. When running multiple types of jobs it gives low performance and poor response time for short jobs compared to large jobs. Fair scheduler can cover some limitations of FIFO scheduler such as work in both large and small clusters. Fair scheduling algorithm does not consider the job weight of web services in HBase which is an important disadvantage of it. Capacity Scheduler is most complex among three schedulers. Another one major issue is "Dead line constraint problem". Dead line of job (job completion time) retrieved after completing the preprocessing stage in hadoop (Kc, 2010)

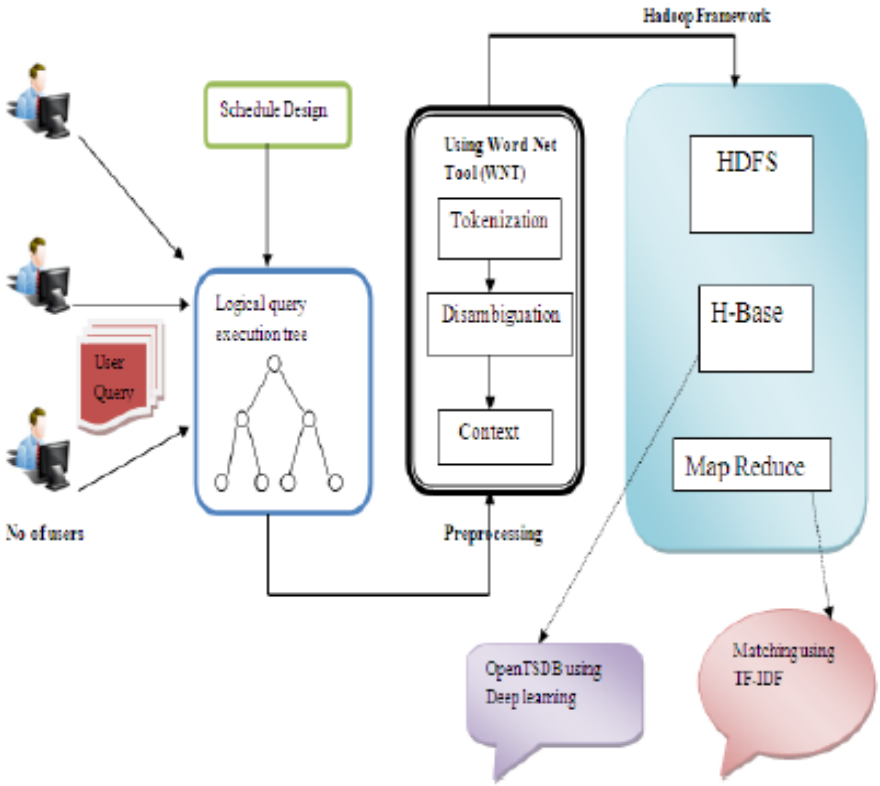

Fig. 1: Architecture of WSDH

2010). Most of the researchers use Ant Colony Optimization (ACO) or Hierarchical clustering algorithm. Hence, this algorithm increases response time due to large number of input parameters used in map reduce process (Mohan and Remya, 2014). Hadoop-HBASE is a scalable and distributed database that stores billions of web services. Hence, an efficient indexing strategy is important for processing large amounts of web services. In this index mechanism with HBase contains functional and non-functional property of web services. Index mechanism is more complex due to non-functional properties such as reliability, availability and reputation (Wang *et al*., 2013). In previous studies, web service selection usually uses any one of the method such as matching (or) similarity calculation. When using any one of the method results are not accurate and scalability issues are arise in matching in such situations necessary to perform pair wise matching (Wang *et al*., 2013). In the proposed research, we perform both matching and similarity in order to achieve high accuracy and reduce search space for user input without any error.

Proposed work: The proposed frame work mainly concentrates on scheduling based on tree instead of using clustering. Scheduling based tree is a very important task when schedule the n number of users hence clustering is a grouping mechanism that consists of matched and similar documents. Hadoop process is carried out by clustering and so various algorithms and techniques were introduced for clustering (Mohan and Remya, 2014). We prefer tree formation due to the increased amount of dataset. In web service, clustering was based on ant colony optimization but with the growing number of user queries clustering become more challenging in retrieving relevant results Scheduling based tree is defined an architecture WSDH (Web Service Discovery using Hadoop) Semantic web service is client server architecture. In client side just give a query and in server side the overall web service selection, composition and preprocessing steps are performed (Fig. 1). We clearly show that, after giving the user query, query is move on to schedule design. Schedule is prepared for multiple users. When a query is submitted by a user, particular query is placed in a pending list, periodically the scheduler will attempt to assign highest priority query in top of the tree. There are two approaches involved in this technique: query selection and query scheduling. Query scheduling to use of ordering the execution on independent queries while ignoring the commonality among queries. We introduce query scheduler to improve

the query scheduling. Query scheduler tool allows users to schedule the queries or statements based on a time interval of multiple users. Based on this estimation, the fastest query is selected. Next construct tree for user given queries. Tree is in a form of binary tree. Tree is traverse from Left-root-Right. Next step is preprocessing user queries. First step is tokenization. Tokenization is the process of splitting sequence of sentence into individual words, phrases and symbols. Its output is an input of another process. After extracting the words, we cannot identify the context of web services such as name of operations and non functional operations of concepts and definitions which are stated in noun and adverbs. Next step is disambiguation which means each word has >1 meaning for the single word, by using wordnet tool remove disambiguation present in the word. Wordnet is just look like a dictionary, in which the words are arranged in semantically instead of alphabetical order. Synonyms words are combined into form group called synsets or synonym set, so each synset has unique concept. Finally we got context for each web services and applying matching and similarity functions for discovering the optimal services. Context contains the concepts of web services. Hadoop ecosystem is a combination of HDFS and Map Reduce. Hadoop database (HBASE) is a database that stores data in top of the HDFS. It contains of Open TSDB. HBASE-Open TSDB maintains all time series data. In the proposed research, we achieve high accuracy while perform deep learning. In this technique small amount of relevant data is retrieved from a large amount of data sets. So, deep learning make it possible to maintain database and improve efficiency of overall system. From that relevant data matching and similarity operation is done. Map reduce technique is used for calculating matching document and Vector space model is used for calculating similarity in a document. TF-IDF is a Term frequency and Inverse Document frequency which is used for calculating the term weights. After calculates similarity value, scoring query using scoring function. Final step is ranking the query based on score list. The following mechanisms are involved in this study:

- Tree based scheduling
- Preprocessing steps
- Hadoop frame work
    - HDFS
    - HBASE-Deep learning in Open TSDB
    - Map Reduce: Matching using TF-IDF calculation
    - Similarity using VSM
    - Ranking

Scheduling: Our goal is expose a control between the time and space complexity. Memory requirements within the context of just specified computation load. Toward this end we must specify enough structure of a scheduling to identify its control complexity and memory requirements. While performing scheduling and tree we must consider the following two objectives: Make span; first objective is make span. Make span is a total execution time of tree which corresponds to the times-span between the beginning of the execution of left side of the tree then Root and end with processing of right side of the tree. Memory: Second objective is the amount of memory is needed for the computation of scheduling. The peak memory is the maximum usage of the memory over the whole schedule. The goal is minimizing the memory usage. In the scheduling, more than one user giving queries at a time how the scheduling process will be happened. There are two different roles are present in our scheduling algorithms. There are query Selection and query Scheduling. When a query is submitted by a user, the query is placed in a pending list, periodically the scheduler will attempt to assign the query in the top of the tree. Query selection: every query is placed in a pending list and the entire list is sorted in order of priority.Query Scheduling is use of ordering the execution on independent queries while ignoring the commonality among queries. We introduce scheduler to improve Query Scheduling. The Scheduler tool allows users to schedule user queries or statements based on a time interval between one users to another user. Based on this work, the fastest query is selected. Query scheduling is a tree based scheduling algorithm. We apply concept in order to reduce time. This process is continues until the end of process. So, the response time of the service will be reduced (Fig. 2-4).

Tree based scheduling: In the proposed work we construct the logical query execution tree. Tree is in a form of binary tree. A good scheduling algorithm should leads to better resource utilization and better performance in insertion, minimum execution time and minimizes the searching cost. Tree scheduling is a methodology for construct perfect periodic scheduling based on ordered trees. Root node and each leaf nodes correspond to a distinct user and the period of time is scheduling. Let CT_n be the completion time when the last node 'n' finishes processing. The main objective is minimizing the completion time. Consider queries an increasing order of finish time. Important operations on a tree are:

- Searching query
- Inserting a node
- Deleting a node
- Finding the minimum or maximum of values stored in the tree (Fig. 5-7)

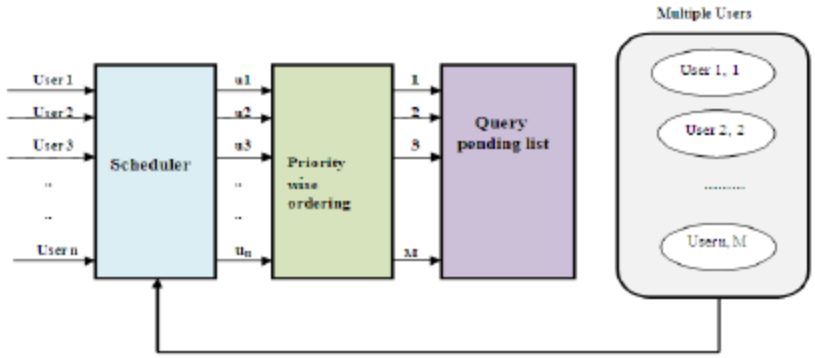

Fig. 2: Schedule design architecture

Fig. 3: Multiple queries executed in web service

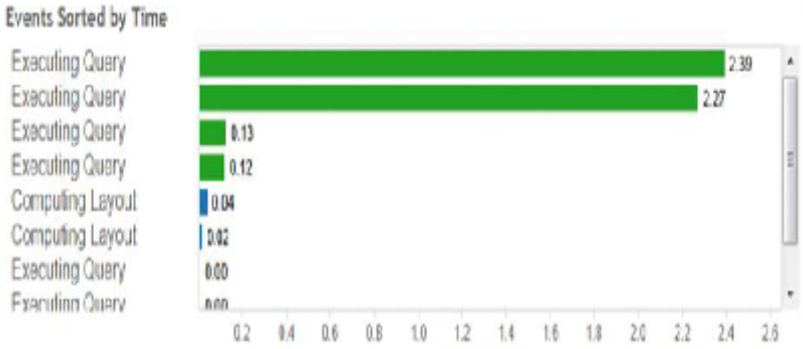

Fig. 4: Query execution vs time

The above example is explained the node which consists of query execution time.Based on the timing tree is constructed in bottom up approach.

Algorithm: scheduling multiple users:

Input:set of users u1,u2 ,….,u_n
Initialize the set of users U and construct Schedule S
Compute the priority order for all users u1,u2 ,…$u_n \in U$
Selection: Determine umin and add user umin to S and it remove from U do for all U
Shedding: Remove from U for all those users,if it were to be added to S
Termination: If U is empty

Algorithm: tree based scheduling:

Input: A sequence of ni = {n_1, n_2,….., n_i} is number of nodes
$n_i = n_{i-1} \geq$, ..., $\geq n_1$ and U = {u1, u2, u3, ..., u } be the list of users and the number of users ordered based on the job execution time.
for ni=1 to ni do
for each user following the finish time ft starting from the leaf node do select minimum finish time in qpp
if the node ((u1<u2) &&(u1. u 2)) then
$u1_1 \neg n1$
else if((u2<u3) and ($u_2 \neq u_3$)) then
$u_2 \neg n2$ do until all the nodes are predicted
end if
end if

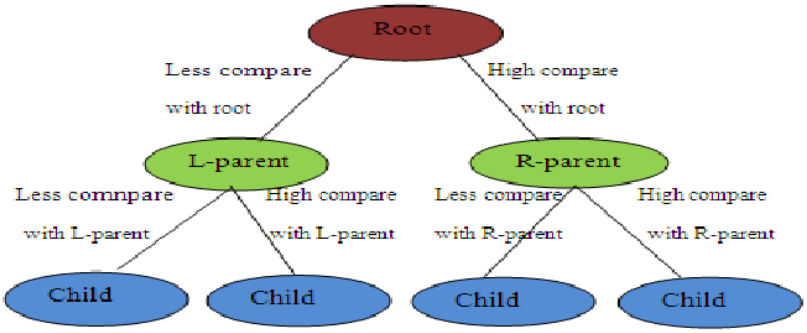

Fig. 5: Logical query execution tree formation

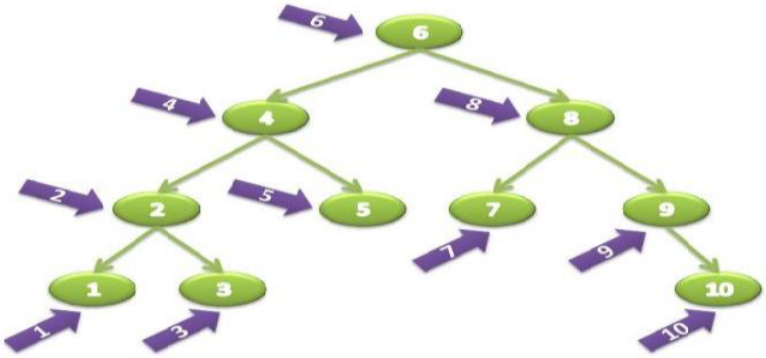

Fig. 6: Tree traversal

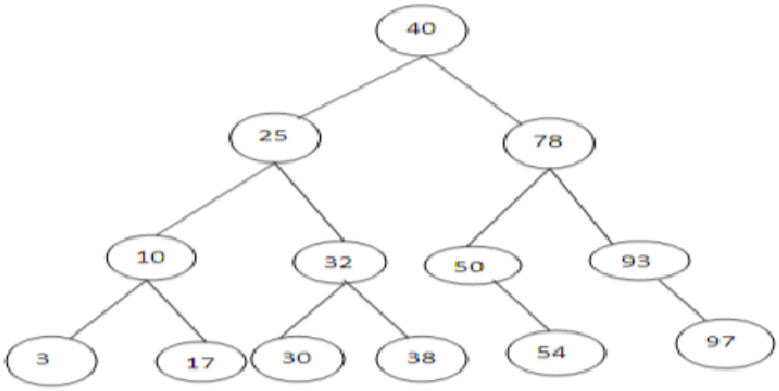

Fig. 7: Example for scheduling based tree for multiple users

end for
end for
Output: A tree with minimum execution time

Starting at the root node, the search algorithm compares the search key (query) with the data stored in the current node:

- If the search key is equal to the data of the current node, the value has been found and the search is terminated
- If the search key is greater than the data of the current node, the search proceeds with the right child as the new current node
- If the search key is less than the data of the current node, the search proceeds with the left child as the new current node
- If the current node is null, the search is terminated as unsuccessful

Algorithm: searching user queries

Search LQET(Val root<pointer>,Val argument<key>)
//Search a logical Query Execution Tree (LQET) for a
given value
Pre: Rt is the root to a LQET//Rt is the root of logical Query Execution Tree.
Return the Na if the value is found or null if the node is not in the tree//Na be the value address
if (R_t is null)
Return null
end if
if (argument <R_t->key)
return search LQET(R_t->left, argument)
else if (argument > R_t->key)
Return search LQET (R_t->right, argument)
else
Return R_t
end if
end search LQET

Since, the above search algorithm visits at most one node in each level of the logical query execution tree, the algorithm runs in O(h) time where h is the height of the tree since O(n) is the time complexity of tree based scheduling algorithm.

Preprocessing: This study describes the pre processing. After constructing tree preprocess the user requests. In pre processing stage User given query may be a single word, single sentence or paragraph. First step is tokenization. Tokenization is a process of splitting sequence of sentence into individual words, phrases and symbols which is called tokens. Main use of tokenization is identification of meaning for full keywords. Its output is an input of another process. After extracting the words, we cannot identify the context of web services such as name of operations and non functional operations of concepts and definitions which are stated in noun and adverbs. Next step is disambiguation which means each single word has more than one meaning, by using Wordnet tool remove disambiguation present in the word.

MATERIALS AND METHODS

Overview of hadoop mechanisms

Hadoop: Hadoop was created by Doug Cutting, the creator of apache Lucene. Hadoop frame work supports processing of large datasets in a distributed environment. It can provide robustness and scalability for a distributed system. Hadoop enables applications to work with thousands of nodes and terabyte of data, without need of user distributing the large data and allocating the memory for massive data storage. Hadoop using three components: HDFS, HBase and Map Reduce. HDFS is a data storing and it is a distributed file system which holds a large amount of data and provide access to these data for many users distributed across a network. Hbase is a database and using Map Reduce to process the large data.

Table 1: Time series data in HBase

	Column Family:t					
Row keys	+0	--	+10	+25	--	--
	0.69	-	0.51	0.42	-	-

HDFS (Hadoop Distributed File System): HDFS is a distributed file system and it is used to store data in a distributed manner in order to solve big data problem an effective manner. By distributing data storage and computation across many servers, the combined storage resource can grow with demand while remaining economical at every size (Table 1).

HBase: HBase is a distributed column-oriented and NoSQL database built on top of HDFS. HBase provides real-time read/write random access to very large datasets. In HBase, a table is physically divided into many regions which are in turn served by different Regional Servers. One of the biggest utility in HBase is to combine real-time. HBase queries with batch Map Reduce jobs, using HDFS as a shared storage platform. HBase can be efficient to use in millions or billions of rows. All rows in Hbase are sorted lexicographically by their row key. HBase introduces time series database technology. Open TSDB stores huge amount of (time series) data and queries mostly based on time or time ranges. Creating HBase table instance is time consuming and ability to reuse the HBase table. Different types of schema available to store time series data in HBase.

Pseudo code for HBase Table

```
public class User Insert {
static string table Name = "users";
static String family Name = "info";
public static void main(String[] args) throws Exception {
Configuration config = HBaseConfiguration.create();
// change the following to connect to remote clusters
// config.set("HBase.zookeeper.quorum", "localhost");
long t1a = System.currentTimeMillis();
HTable htable = new HTable(config, tableName);
long t1b = System.currentTimeMillis();
System.out.println ("Connected to HTable in : " + (t1b-t1a)+" ms");
int total = 100;
long t2a = System.currentTimeMillis();
for (int i = 0; i < total; i++) {
int userid = i;
byte[] key = Bytes.toBytes(userid);
Put put = new Put(key);
put.add(Bytes.toBytes(familyName),
htable.put(put);
}
```

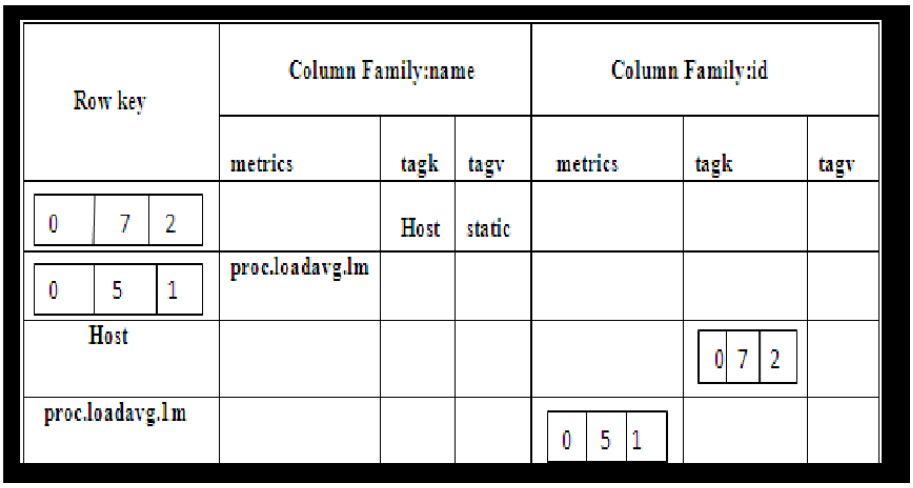

Fig. 8: Open-TSDB structure

Table 2: TSDB-uid Open-TSDB in HBase

Row key	Family: Column	Version
Metric-ID	Properties	Current time stamp

Table 3: TSDB inside HBase

	Column Family:t						
Row keys	+0	+15	+20	-	+1890	--	+3600
	0.69	-	0.51		0.42	-	-
	0.99	0.72					

```
long t2b = System.currentTimeMillis();
System.out.println("inserted "+total+" users in "+(t2b
-t2a)+" ms")
htable.close()
}
}
```

Output: Connected to Htable in: 1190 m sec
Inserted 100 users in 350 m sec

Open TSDB: Open TSDB is an open source distributed and scalable time-series database, developed by Stumble upon. It supports real time collection of data points from various sources. It is designed to handle terabytes of data with better performance for different monitoring needs. It stores, indexes and serves metrics at a large scale. All the data is stored in HBase in two different tables: the TSDB table provides storage and query support overtime-series data and the TSDB-uid table maintains an index of globally unique values for all metrics and tags. Open TSDB can store metrics per second. Performance measure of Open TSDB:

- Read queries are generally capable of retrieving, munging and plotting over 500k data points/sec
- Over 1 Byte new data points per day is 12 k sec^{-1}
- Hundreds of millions and billions of data points stored
- The <2 TB of disk space consumed

Open TSDB design: In Open TSDB the TSDB table stores data using composite row keys, comprising of metric id (3 byte), time stamp (hourly in 4 byte), tag key-ids (3 byte) and tag values (3 byte). Each column is grouped under one column family consisting of column qualifiers that representing time as shown in Table 1. The TSDB table is the heart of the time series database which stores time series data points. TSDB-uid is a look up table for metrics and tags (Table 2 and 3 and Fig. 8).

Deep learning in Open TSDB: Normally H-Base storing the large amount of data sets in a database. But in this system storing only relevant datasets which is retrieved from the tree. So we introduce the new methodology for "Deep Learning" in HBase-OpenTSDB. Deep learning achieves high accuracy and improves performance of precision and recall factor. Deep learning is a deep structured learning which is used for retrieving small amount of relevant data from a large amount of data sets. This deep learning methodology is used in Open TSDB. It is a distributed time series database that is stored in top of the HBase that enables to store, index the query and plot the query time (Fig. 9).

Calculating data size in Hbase: HBase stores the data in key value format. It is explain the key value, data type and byte required for each field (Table 4).

Calculate the data size: Key values consist of two parts: fixed part and variable part.

Fixed part:

HBASE data size = Key length +Value length+
Row length+CF length+Key value+Time stamp
= 4+4+2+1+1+8 = 20 bytes

Table 4: TSDB inside HBASE

Key length	Value length	Row length	Row	Column family length	Column family	Column qualifier	Time stamp	Key type	Value
Integer	Integer	Short	Byte	Byte	Byte	Byte	Long	Byte	Byte
4	4	1	1	1	1		8	1	

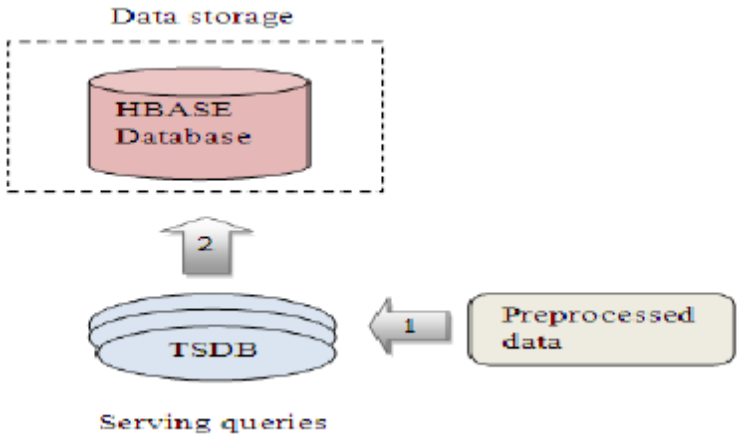

Fig. 9: Deep learning in open TSDB architecture

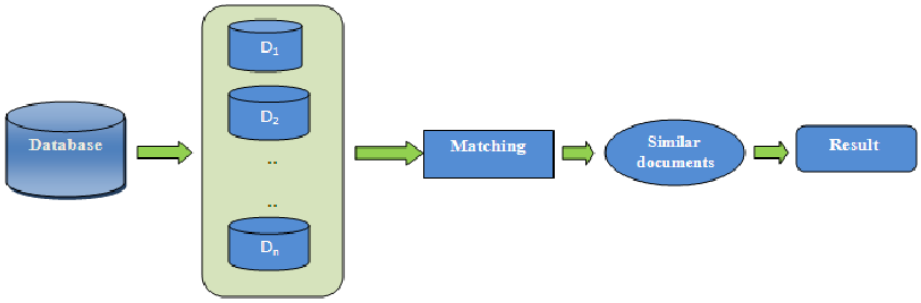

Fig. 10: Matching and similarity

Variable part:

HBASE data size =
Row+Column family+Column qualifier+value
= 13+2+2+8 = 25 Bytes
Total size = Fixed part+Variable part
Column 1 = 20+25 = 45 bytes

Open TSDB consist of many column, if it send data in every 15 second then, we will have 240 columns in each row for 60 min boundary as mention by Open TSDB Each row size is calculated as:

$$\text{Row } 1 = 240 \times 45 = 10800 \text{ Bytes}$$

To store 1 million records, the space required is 10 GB.

$$= 10800 \times 1000 = 10 \text{ GB}$$

Matching and similarity: Extracting accurate information for user input is the main drawback for information retrieval system in web. For accurate results we bring both similarity and matching procedures. If one process is used there may occur error and so the retrieved results would not be effective. Due to verify the keywords without any error we perform both the process. Our proposed work implements matching using map reduce technique and similarity using vector space model. Open-TSDB contains time series data using this Open-TSDB data matching and similarity is performed. Figure 10 shows, the data retrieved from the database then matching, it provides similar documents. In this study, researchers present a concept of matching method based on TF-IDF using Map Reduce technique. We conducted a comparative evaluation of the Hadoop based TF-IDF and without Hadoop. The results empirically prove the strength of our approach.

Matching between two words is usually calculated using in the space of TF-IDF. Terms are simple words,

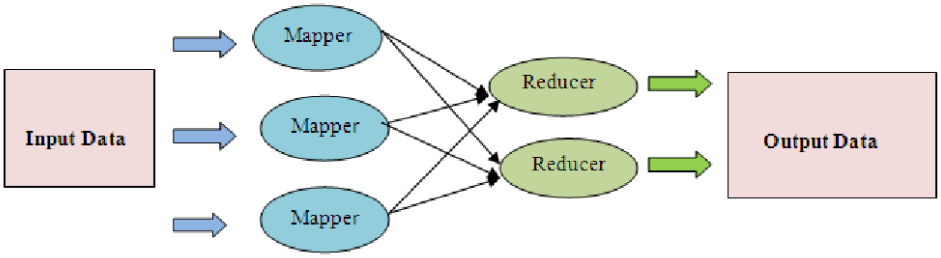

Fig. 11: Map reduce framework

phrases and symbols and special characters. One of the most known techniques for calculating these measures or term weights is Term Frequency and Inverse Document Frequency (TF-IDF). Many formulas and methods used for calculating matching documents. TF-IDF weighting scheme are often used by search engines for scoring and ranking a document's relevance based on user given query.

TF-IDF without using Hadoop: TF-IDF weight (Term Frequency-Inverse Document Frequency) is a weight scheme often used in information retrieval and text mining. This weight is a statistical measurement used to evaluate how much a word present in a document from the collection of a document sets. The importance increases proportionally to the number of times, a word appears in the document. But i t's offset by frequency of the word in the collection. So, how quickly and efficiently calculate the TF-IDF is very important. TF-IDF is composed by two terms one is term frequency and another one is inverse document frequency.

TF calculation: Term count in the given document is simply the number of times a given term appears in that document. For the term t within the particular document d, so its term frequency is defined as follows:

$$Tf_{a,b} = \frac{F(a,b)}{\sum_c F(c,b)} \quad (1)$$

In the formula, F (a,b) is the number of occurrences of the considered term (ta) in document db and the denominator is the sum of number of occurrences of all terms in document

IDF calculation: Inverse document frequency is a measure of the general importance of the term. The formula is defined as follows:

$$Idf_a = \frac{\log|D|}{|\{b : t\alpha \in db\}|} \quad (2)$$

In the above mentioned formula, |D| is the total number of documents in the Collection; {b; ta. db} is the number of documents where the term ta appears and $F_{a,b} \neq 0$.

TF-IDF calculation: The TF-IDF weight of term is the product of tf and idf. Equation is defined as follows:

$$(TF\text{-}IDF)_{a,b} = Tf_{a,b} \times Idf_a \quad (3)$$

If term t is appears in a documents calculating the following functions:

- Number of times term t appears in a given document
- Number of terms in each document
- Number of documents t appears in total number of documents

TF-IDF with Hadoop using Map Reduce technique: Map Reduce is a programming paradigm for associated implementation for processing large datasets and generating similar data sets. Map reduce model contains two phases namely mapper and reducer phases. Users specify a mapper that processes a pair to generate a set of intermediate pairs and a reducer that merges all intermediate values associated with the same intermediate key. We give a TF-IDF algorithm with hadoop based on the map reduces scheme. It provides high throughput and accuracy to access the application data and it's suitable for applications that have large data sets. An experiment shows that in the case of massive data computing and mining, the new method applying hadoop framework which is more efficient than the traditional methods (Fig. 11).

Calculate frequent word in the document: In mapper, we use regular expressions to match words and write <<word in document Name>, 1> pairs to intermediate values which will be processed by the reducer. Then we calculate the frequent word in the document directly to the reducer. Output of the reducer need to be written to intermediate

files which will be processed in next Map Reduce process. The output is using as key and as value 'F' as a value and "word document Name" is a key. Function is designed as follows:

Mapper ( ) :Input: <Document Line No, contents>
Output: < <word in document name>, 1>
Reducer ( ): Input: <<word in document name>, 1>
Output: <<word in document name>, F>

Calculate the total number of words in each document: In this stage, we rearranged the pairs in mapper (key and value). We calculate the total number of words in each document in reducer. Reducer output is needs to be written to the intermediate files or temporary files which will be help full to process in next Map Reduce process. The output is using as the key as the value 'F' is the number of frequent term, 'word' in the document 'document Name' and 'N' is the total number of words of 'document name'. Map/Reduce Function is designed as follows:

Mapper () :Input: <<<word in document name>, F >
Output: <document Name, < word=F> >
Reducer ( ) :Input: <document Name, < word=F> >
Output: <<word in document name>, <F/N>>

Calculate TF-IDF: We reorganized the pairs in mapper (using key and value). Then we calculate the number „d. which is the number of documents containing this word (or) term and the number 'D' which is the total number of whole collection of documents. We can calculate the TF-IDF according equation

$$\text{TF-IDF} = \frac{F}{N \times \log\left(\frac{D}{d}\right)} \quad (4)$$

Function is designed as follows:

Mapper ( ): Input: <<word in document name>,< F/N> >
Output: <word, <document name F/N>>
Reducer ( ): Input: <word, <document name F/N >>
Output: << word in document name>, F / N * log (D / d) >

Similarity calculation using vector space model: When two user preferences have similar, matching only is not sufficient to retrieve the accurate services. Matching is only matching query to the particular documents. So it will give n number of services. But our main scope is retrieve accurate web services based on the user preferences. Similarity measure is a function that computes degree of similarity using similarity function. The vector space model represents, documents and queries as vectors in dimensional space (multi dimensional), the terms are used as dimensions to build an index to represent the documents. Each term represents documents. If a term occurs in document, its value is calculated using cosine similarity function. It is used in information retrieval, indexing and ranking and can be successfully used in evaluation of search engines. The vector space model can be divided in to two stages. The first stage is similarity calculated using cosine similarity function. A common similarity measure known as cosine measure determines angle between the query vector and document vector. A vector distance measure between the query and documents is used to rank retrieved documents. The second stage, rank the documents with respect to the query according to a similarity values.

$$X=\{x_1, x_2,\ldots,x_n\}; Y=\{y_1, y_2,\ldots,y_n\}$$

Where:
x = A set of queries
y = A set of documents

The cosine of the angle. The θ between x and y can be as follows:

$$\text{Cos}(x,y) = \frac{\sum_{i=1}^{n} xi \times yi}{\sqrt{\sum_{i=1}^{n} xi2}\sqrt{\sum_{i=1}^{n} yi2}} \quad (5)$$

Where, xi be the TF-IDF weight of term I in the query and Yi be the TF-IDF term weight of term I in the document (Fig. 12):

$$= \frac{xy}{\sqrt{xx}\sqrt{yy}} = \frac{x}{\|x\|}\frac{y}{\|y\|}$$

Scoring documents: Scoring each document in the datasets is very essential technique. In case of large

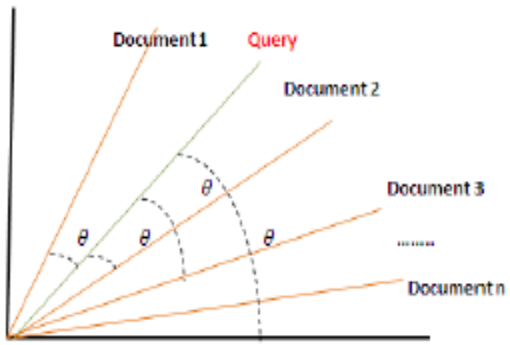

Fig. 12: Vector space model

Table 5: Calculating relevance score and rank for the word retrieve from document

Querries	Score values	Ranks
Q_1	0.00045	2
Q_2	0.00123	1
Q_3	0.000432	3
Q_4	0.00473	4

document collections, the resulting number of matching documents can far exceed the number of a human user. Accordingly, it is important for a search engine to rank (or) order the queries. To do this, the search engine computes a rank for each matching document using score list.

Scoring function: Score calculation for each document is done using Vector space of query and document**s.** If the query term does not occur in the document score should be 0.The most frequent query term in the document, the score should be higher.

$$\text{Score}(q,d)\ \frac{\vec{v}(q)\,\vec{v}(d)}{|\vec{v}(q)||\vec{v}(q)|} \qquad (6)$$

Where, q is a query and d is a document.

Example for ranking technique

Ranking technique: Documents are rank based on the score list. To rank similar documents based on score values. We do it by assigning numerical rank to each document based on a scoring value function which incorporates features of the query (q), document (d) and the overall document collection (D). Ranking technique is indexed by the score list, to be able to provide the order of relevance documents, for the web services, with respect to the user query (Table 5).

Algorithm for Ranking

Input: set of queries
Word tokenizes the query to a set of words.
Set ranks to an empty list ranks= { } output will be produced here.
Compute for each document di in the n documents.
Initializes score of d=0.
Compute for each term t in document D.
//Matching-Map Reduce Technique
compute term frequency: n=tf (t,d)
compute idf for across documents: m=idf(t,d)
n×m = tf*idf
//Similarity-Vector Space Model Technique
Calculate score value:

$$\text{Score}(q,d)\ \frac{\vec{v}(q)\,\vec{v}(d)}{|\vec{v}(q)||\vec{v}(q)|}$$

Construct (index of d, score for d)
sort the list
Return ranks
Output: Rank list Rl(Top ranked document)

Performance evaluations

Performance metrics: In the experiments we compare our proposed system with existing mechanisms. From that we are going to prove the proposed method is achieving efficient result when compared with remaining algorithms. The main objective is to increase the accuracy rate, efficient precision, recall and response time. In existing system only focused scheduling using different Hadoop scheduler. Using these schedulers retrieve similar web services from large size datasets. So, the proposed research, we considered the following parameters in the experiments:

- Accuracy
- Precision and recall factor
- Response time vs. number of web services
- Differences between open TSDB and TSDB using deep learning
- Hadoop scheduling vs. tree based scheduling

Experimental setup: We conduct the experiment on Hadoop framework installed on single computer windows environment. The configuration of this stand alone computer is 6 GB Ram and 3 GHZ CPU, Pentium or dual core processor and the Hadoop version is 2.60. Map Reduce algorithm is implemented using jdk 1.8 and Net beans 8.0. We use tour dataset from our tour-pedia.org. This dataset has information of various places and reviews. But, reseachers considered the place of dataset only for the experiments. The reviews contain the information of review of described places present in place dataset.

Place dataset collects information from the various social medias and which contains the following fields such as id, name, address, category, location, latitude, longitude, services, subc ategory, website.icon, description, external_url's, statistics and polarity. Each record nearly contains the 5500 records.

RESULTS AND DISCUSSION

In this study, we discuss about performance results of all parameters and it's discussed in section 5.1. For conducting the experiments we use 5000 records. Table 6 shows the performance result of our proposed concept SDH.

To analyze the performance metrics in Hadoop, we use measurement indicators like accuracy, precision, recall and response time. Result is shown in Table 7. A perfect precision score of 1.0 means that every correspondence computed by the algorithm was correct (correctness), whereas a perfect recall scores of 1.0 means that all correct correspondences were found (completeness). To compute

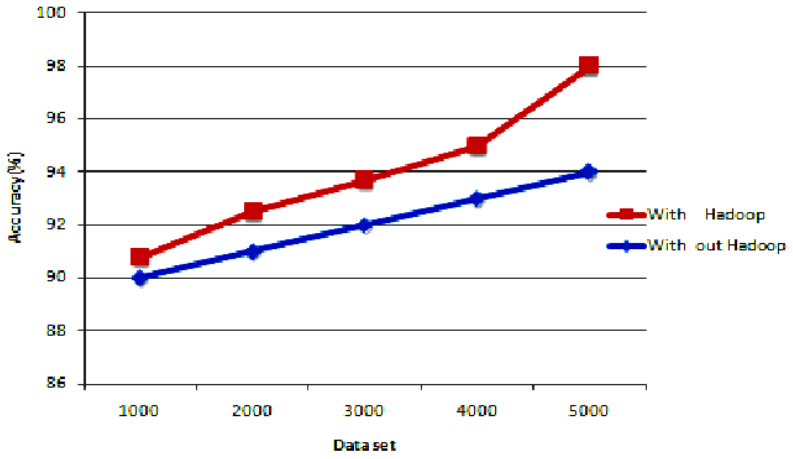

Fig. 13: Performance analysis of accuracy

Table 6: Calculating relevance score and rank for the word retrieve from document

Querries/Results	Response time (sec)	Accuracy (%)	Precision	Recall
1000	1.0	91.0	0.735	0.865
2000	2.0	92.3	0.741	0.875
3000	3.4	94.0	0.743	0.880
4000	5.2	95.0	0.756	0.885
5000	9.1	98.0	0.778	0.890

precision and recall, the alignment X returned by the algorithm is compared with a reference alignment Y. Precision is given by equation:

$$P(X,Y) = \frac{|X \cap Y|}{|Y|} \quad (7)$$

Whereas recall is defined as:

$$R(X,Y) = \frac{|X \cap Y|}{|X|} \quad (8)$$

Accuracy: Accuracy is reduced, when increases the amount of data. In the proposed system shows when increase the amount of records, accuracy is also increased. Because, it is one of major factors in performance metrics. So main scope of our web service discovery is to achieve high accuracy. In existing paper using Apache Hadoop and Mahout can recommend large amount of data efficiently (Thangavel *et al*., 2013). But when it comes to real time, random access is not possible by using Apache Hadoop. So, the proposed work we use Hadoop framework alone to increase the accuracy, in that system matching and similarity is done using map reduce technique. In this technique similar web services are sorted based on their scores. Based upon user given query the accuracy value will be differing. An experimental result shows the proposed SDH give more coherent result when considering Hadoop framework (Fig. 13):

$$\text{Accuracy} = \frac{\text{Number of words succesfully corrected}}{\text{Number of input words}} \quad (9)$$

Precision and recall: Precision and recall factor is used to measure the efficiency of similarity searching and matching. Figure 14 shows the result of precision and recall factors.

Response time: Figure 15 gives the results of response time of proposed hadoop system with existing without hadoop system. Response time of each service will be varied when increasing the number of web services (Shetty *et al*., 2014).

From that result we prove that our proposed approach achieves efficient result. The amount of response time will be gradually increased with respect to increase the web services. Response time is represent as, difference between the starting time of query processing and finishing time of process. During this result web service will be discovered. In previous papers they calculate the response time between Hadoop and without Hadoop framework. Response time of existing system has high response time when increasing records but in our proposed system only slight increase of response time results.

Speed up: In this experiment, we are comparing speed up of the system when increasing number of nodes, accuracy will also increased. Speed up is given by the ratio of execution time with increasing the number of nodes. Given by shown in Fig. 16:

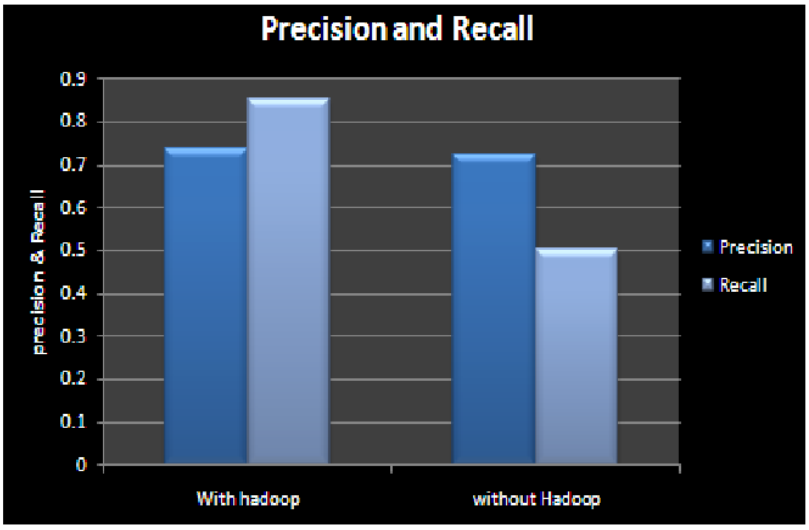

Fig. 14: Comparative analysis of precision and recall queries

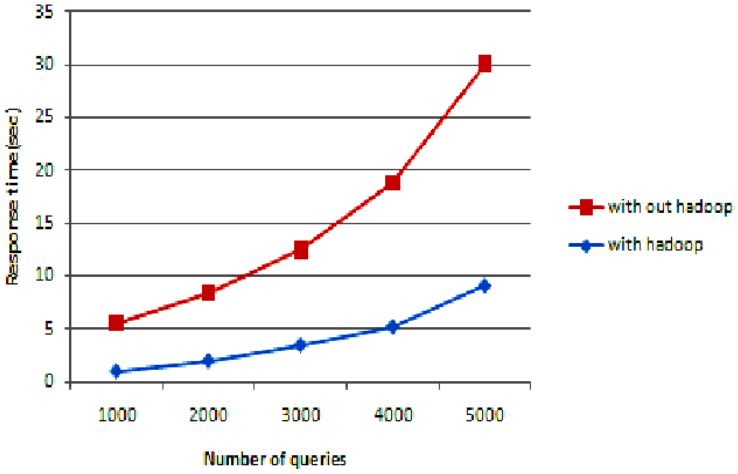

Fig. 15: Comparative analysis of response time

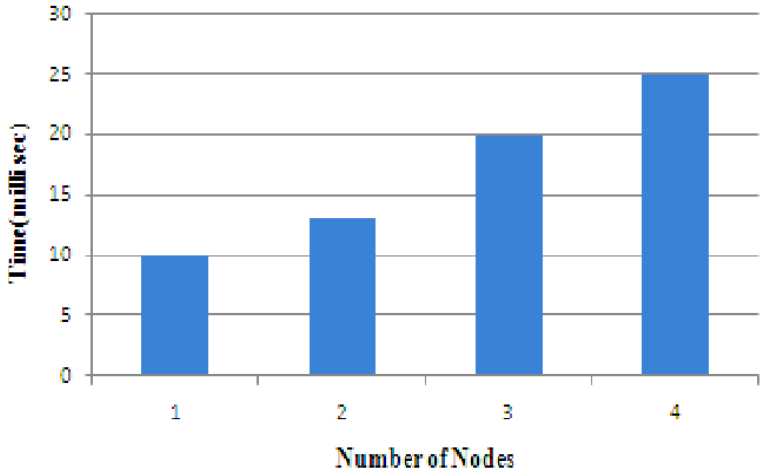

Fig. 16: Speed up with respect to number of nodes

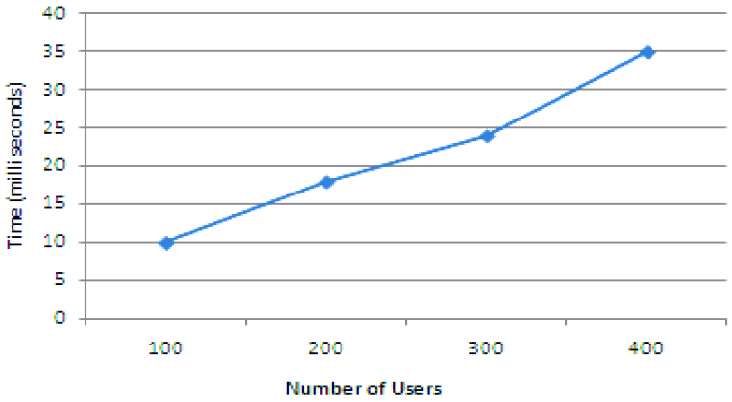

Fig. 17: Speed up with respect to number of users

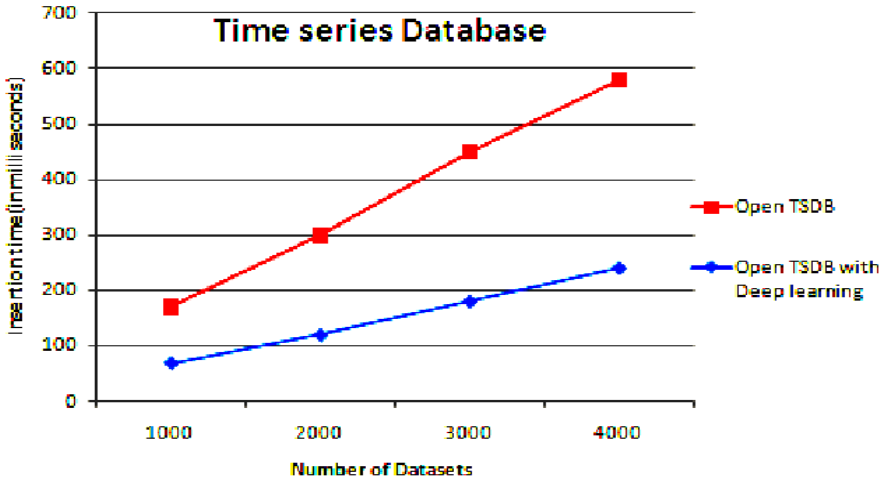

Fig. 18: Difference between Open TSDB and Open TSDB With deep learning

$$\text{Speed up} = \frac{T(1)}{T(N)}$$

Where:
N = Number of nodes
T = Execution time

The T (1) represents times taken by single node and T (N) represents time taken by n number of nodes. Figure 17 shows that the number of users increased ,the speed has also increased. This is achieved by using "best performance scheduling".

Differences between Open TSDB and TSDB using deep learning: Open TSDB is a time series database that stores data on top of the HBase. In existing system it will stores the information in database for huge amount of data. But our proposed work we use scheduling with tree, in that we retrieve relevant results from scheduling technique. This information is stored in open TSDB using deep learning. In deep learning technique small amount of data retrieve from huge set of datasets. When we compare with existing Open TSDB (Agrwal, 2014) and our proposed Open TSDB system. Figure 18 shows the effective results.

Hadoop vs proposed scheduling: Hadoop system introduced many schedulers to minimize the job completion time. It is critical to select a scheduling algorithm by considering the Hadoop factors like scalability, accuracy, precision and recall factor and desired performance level. Scheduling algorithm which performs well in one Hadoop system, may not work well for a system that differs in these factors. These entire algorithms proposed to address one (or) more problems and none of them is suitable for our objectives (Pakize, 2014). Due to these important issues, we described Tree based scheduling. In this technique scheduling is performed after given user query. Our proposed scheduler attempts to schedule every user by the user query with completion time is determined by the logical query execution tree (Fig. 19).

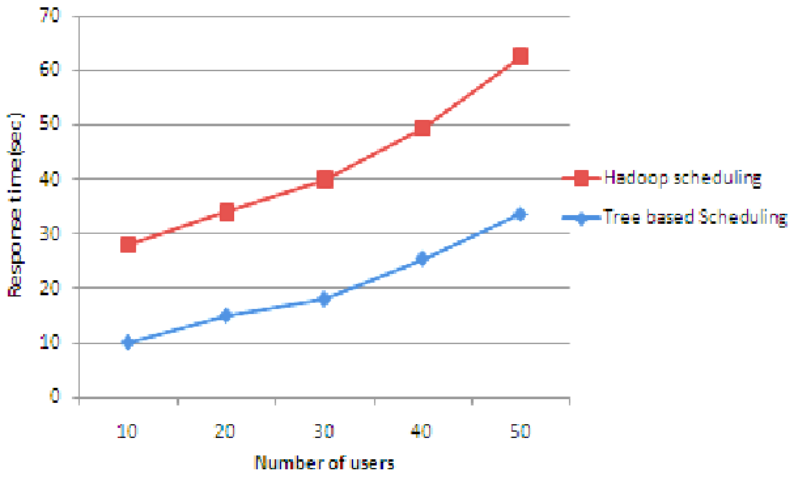

Fig. 19: Comparative analysis of scheduling vs Hadoop scheduling

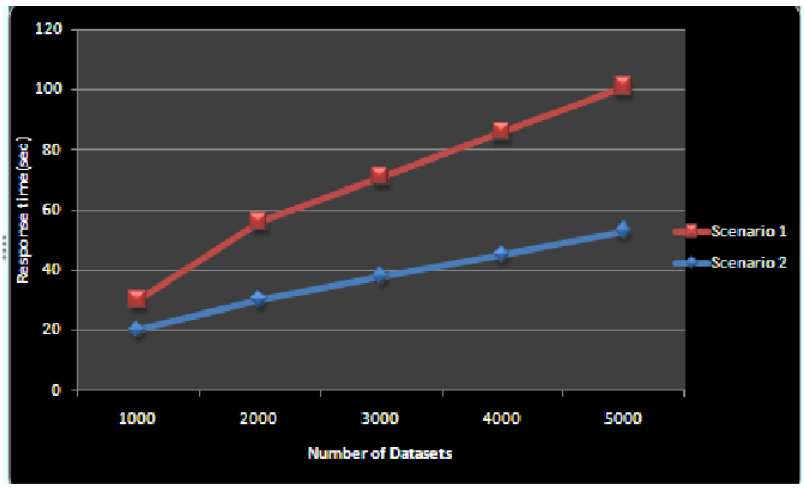

Fig. 20: Comparative analysis for overall scenario

Comparative analysis of overall scenario: Figure 20 shows the performance analysis by two measurement first one is Entire Scenario with following methodologies: scheduling with tree, preprocessing and deep learning then calculate matching and similarity using Hadoop framework and second one scenario is scheduling with tree, preprocessing, Hadoop-Deep learning in HBase Open TSDB, then calculating matching and similarity. Deep learning is a deep structured learning used for high level abstractions in data. When compare with deep learning and open TSDB based Deep Learning. Open TSDB based deep learning technique gives the better performance and high accuracy that shows in Fig. 18.

CONCLUSION

Owing to massive vogue of service oriented architecture, there has been notable extension in service repository when the repository size will be increased response time of service discovery is reduced. A traditional web service does not offer ideal result for user given query due to storage issue. To avoid this problems Hadoop framework is introduced. In hadoop framework we invite WSDH (Web Service Discovery using Hadoop). In this architecture, we perform Tree based scheduling, preprocessing, Deep learning and matching and similarity using Map Reduce and Vector Space Model in order to achieve high accuracy, reduce response time and increase precision and recall factor values. We conduct experimental analysis for WSDH in Hadoop environment and consider the parameter as response time, accuracy, precision and recall. From the experimental results we prove that our approach attain effective results. The values are stored in tabular format. The approach achieves 98% accuracy when amount of records increasing in web services.

REFERENCES

Aditi, H. and V. Solanki, 2014. A new context driven page ranking algorithm. Int. J. Latest Trends Eng. Techonol., 4: 6-11.

Agrawal, B., A. Chakravorty, C. Rong and T.W. Wlodarczyk, 2014. R2 Time: A framework to analyse open TSDB Time-Series Data in HBase. Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom.), December 15-18, 2014, IEEE, New York, USA., ISBN: 978-1-4799-4093-6, pp: 970-975.

Alonso, G., F. Casati, H. Kuno and V. Machiraju, 2003. Web Services: Concepts, Architecture and Applications. Springer, New York, USA., ISBN:3540440089, Pages: 359.

Amotz, B.N., V. Dreizin and P.S. Boaz, 2014. Efficient periodic scheduling by Trees. IEEE. Trans., 2: 1-10.

Crasso, M., A. Zunino and M. Campo, 2008. Easy web service discovery: A query-by-example approach. Sci. Comput. Programm., 71: 144-164.

Dogan, A. and F. Ozguner, 2002. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. IEEE Trans. Parallel Distrib. Syst., 13: 308-323.

Gholamzadeh, N. and F. Taghiyareh, 2010. Ontology-based fuzzy web services clustering. Proceedings of the 2010 5th International Symposium on Telecommunications (IST.), December 4-6, 2010, IEEE, New York, USA., ISBN:978-1-4244-8183-5, pp: 721-725.

Gunasri, R. and R. Kanagaraj, 2014. Natural language processing and clustering based service discovery. Intl. J. Technol. Enhancements Emerging Eng. Res., 3: 28-31.

Kc, K. and K. Anyanwu, 2010. Scheduling hadoop jobs to meet deadlines. Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom.), November 30-December 3, 2010, IEEE, New York, USA., ISBN:978-1-4244-9405-7, pp: 388-392.

Lin, W. and J. Liu, 2013. Performance analysis of Map Reduce program in heterogeneous cloud computing. J. Networks, 8: 1734-1741.

Lu, W., Y. Shen, S. Chen and B.C. Ooi, 2012. Efficient processing of k nearest neighbor joins using mapreduce. Proc. VLDB. Endowment, 5: 1016-1027.

Mohan, A. and G. Remya, 2014. A parallel implementation of ant colony optimization for TSP based on map reduce framework. Int. J. Comput. Appl., 88: 9-12.

Padmapriya, V. and A. Appandairaj, 2015. Rank boosting approach on hadoop for consumer preference recommender system. Int. J. Recent Res. Sci. Eng. Technol., 1: 15-22.

Pakize, S.R., 2014. A comprehensive view of Hadoop MapReduce scheduling algorithms. Int. J. Comput. Networks Commun. Secur., 2: 308-317.

Sanjay and D. Kumar, 2015. A review paper on page ranking algorithms. Int. J. Adv. Res. Comput. Eng. Techonol., 4: 2806-2811.

Shashank, S., P.R. Shalini and A.K. Sinha, 2014. A novel web service composition and web service discovery based on map reduce algorithm. Proceedings of the IJCA International Conference on Information and Communication Technologies ICICT, December 3-5, 2014, Elsevier, Amsterdam, Netherlands, pp: 41-45.

Shetty, S., P.R. Shalini and A.K. Sinha, 2014. An efficient web service selection using hadoop ecosystem based web service management system (HWSMS). Int. J. Eng. Dev. Res., 2: 1876-1883.

Skoutas, D., D. Sacharidis, A. Simitsis and T. Sellis, 2010. Ranking and clustering web services using multicriteria dominance relationships. IEEE Trans. Serv. Comput., 3: 163-177.

Tang, Z., L. Jiang, J. Zhou, K. Li and K. Li, 2015. A self-adaptive scheduling algorithm for reduce start time. Future Generation Comput. Syst., 43: 51-60.

Thangavel, S.K., N.S. Thampi and C.I. Johnpaul, 2013. Performance analysis of various recommendation algorithm using apache hadoop and mahout. Int. J. Scient. Eng. Res., 4: 279-287.

Wang, S., W. Su, X. Zhu and H. Zhang, 2013. A Hadoop-based approach for efficient web service management. Int. J. Web Grid Serv., 9: 18-34.

Xie, J., S. Yin, X. Ruan, Z. Ding and Y. Tian *et al*., 2010. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, April 19-23, 2010, Atlanta, GA., USA., pp: 1-9.

Xu, S., Y. Zhu, H. Jiang and F.C. Lau, 2008. A user-oriented webpage ranking algorithm based on user attention time. Proceedings of the 23rd AAAI Conference on Artificial Intelligence, July 13-17, 2008, AAAI Press, Chicago, Illinois, pp: 1255-1260.