Bridging the Gap for Retrieving DBpedia Data

AHMED SALAMA ISMAIL¹, HAYTHAM AL-FEEL², HODA M. O.MOKHTAR³
Information Systems Department, Faculty of Computers and Information
¹, ²Fayoum University
³Cairo University
¹, ²Fayoum, Egypt
³Cairo, Egypt
¹asi01@fayoum.edu.eg
²htf00@fayoum.edu.eg
³h.mokhtar@fci-cu.edu.eg

Abstract: - DBpedia is nowadays considered one of the main projects in the World Wide Web that extracts and enriches Wikipedia data in a structured form. Also, it is considered the central hub for the Linked Open Data. Querying DBpedia using big data approaches such as Hive-QL is regarded as one of the new techniques to solve the shortcomings of SPARQL; the main query language of DBpedia and the Semantic Web. Nevertheless, despite the speed of Hive-QL compared to SPARQL, it has a stability problem. Our paper presents a new architecture and implementation for querying DBpedia using Shark query language in addition to Hive-QL. As a result of this work, An obvious decrease in execution time, as well as, an increase in the degree of stability have been attained.

Key-Words: - Semantic Web; Spark; Shark; Big Data; Hadoop; Hive; DBpedia; MapReduce

1. Introduction

Nowadays, various types of structured and unstructured data volume are exploded, so it is increasingly important for users to analyze huge amounts of data to get the most advantages of those large data. However, the volume, variety and velocity of the large data may cause some problems in its performance when analyzing or managing it using traditional data processing methodologies. Hence, companies, and research communities changed their vision for big data analytics [1] and methodologies. New approaches were developed, such as Apache Hadoop [2] which is an open source project that provides new solutions to the problems of the large data processing using some techniques like Pig [3], Mahout[4] and Hive[5]. On the other hand, Semantic web [6] is considered the web of data that is used to improve the meaning of web pages content and increases the co-operation between machines and users. DBpedia is regarded as a clear example of both Semantic Web and big data due to its enormous volume of data in the triple form [7] that uses SPARQL [8] as its query language. Hive-QL could be the suitable alternative for the SPARQL for querying DBpedia if it does not have a stability problem. From this point of view; our work tries to refine and optimize the performance of the DBpedia queries as a step to increase its scalability using Shark query in the Apache Spark [9] as a new technique for querying and managing DBpedia data. The rest of this paper is organized as follows: Section 2 discusses the related work. Section 3 highlights the SPARQL query language that is currently used for querying DBpedia project. Section 4 overviews the big data techniques. Section 5 presents an architecture for retrieving DBpedia content using HIVE-QL and Shark. Section 6 describes the implementation process of the proposed architecture. Test cases using HIVE-QL and Shark query language are presented in section 7, which also highlights the implementation results. Finally, Section 8 concludes our paper and discusses possible directions for future work.

2. Related Work

Querying huge volume of data as in DBpedia using big data techniques, such as Hive-QL in the Hive
environment, was discussed in [10]. The main objective was to decrease the retrieval time needed for replaying a query. This work adds a new vision for retrieving data on DBpedia and shows that Hive-QL is better than SPARQL in query retrieval time, but the proposed architecture has a stability problem. On the other hand, the work represented in [11] defined a comprehensive performance evaluation methods to assess different query techniques, such as Hive, Impala, and Shark. The work presented in [14] introduced the building of Shark on distributed environment such as Graphlab [12], Haloop [13] and Spark[5]. Demonstrating partial DAG execution using Shark query language that is used to increase the join performance through run-time re-optimization of query processing using query optimization techniques was proposed in [15]. Another in-depth performance evaluation was presented in [16] that used more metrics and evaluated more performance aspects including scalability.

3. Querying DBpedia using SPARQL

SPARQL [7] is the Semantic Web query language that is used to perform queries on RDF graphs. The RDF graph [17] is a set of triples that consists of a subject, predicate, and object. One of the most significant achievements of Semantic Web projects is the DBpedia project that aims to extract and enrich Wikipedia content semantically [18]. Wikipedia is considered the source of data that has infoboxes been extracted by different extractors, such as label, redirect, and abstract extractors. These extracted data were embedded into DBpedia in the form of N-Triples dump stored in Virtuoso triplestore that enables users to execute sophisticated queries through SPARQL endpoint as shown in Fig.1. We can perform any semantic query using SPARQL on DBpedia as any Semantic Web content, as shown in the DBpedia architecture [19]. The first dataset of DBpedia was published in 2007 under the free licenses, allowing others to reuse the dataset [20][21]. There are different multilingual chapters of DBpedia available nowadays and predicted to be increased in the coming few days which lead to the increasing of datasets [22] which will be reflected on the usage of big data techniques to retrieve DBpedia data.

![Virtuoso SPARQL Query Editor](image)

Fig. 1. Virtuoso the DBpedia SPARQL endpoint

4. Big data techniques

Hive is considered the open-source MapReduce paradigm that was built on the top of the Hadoop platform to be used as an alternative methodology to manage and process an enormous amount of datasets [23]. It is most suited for data warehouse applications in which the fast response time is not required. The Hive language [24] performs MapReduce jobs in its back end and enables scaling up from a single machine to a number of machines, each of which has its local computation and storage. On the other hand, the Hadoop Distributed File System (HDFS) [25] facilitates the work of the MapReduce function and manage the relations between the master and slaves machines. Hive-QL is one of the key success of the Hive environment due to it is capability to retrieve data via communicating with MapReduce function[26]. Hive-QL is powered by Apache Hive[27] and doesn’t fully conform to any particular revision of the ANSI SQL standard as all SQL languages in widespread use [27]. Also, it is perhaps similar to the MySQL’s language, but with significant differences, Hive offers no support for row-level inserts, updates, and deletes. While Apache Spark is considered an emerging distributed processing framework that supports big data analytics and large-scale data processing. Spark SQL is regarded as a Spark Core top layer that defines a new data abstraction called Schema RDD that provides support for structured and semi-structured data. On the other hand, Shark [28] is considered the usage of Apache Hive in the environment of Spark without using the MapReduce scripts. It can answer Hive-QL queries faster than Hive without modification to the neither existing
data nor queries. While Shark supports Hive's query language, meta store, serialization formats, and user-defined functions.

5. Proposed Architecture

Dealing with DBpedia queries using big data techniques is considered one of the new trends of research that aims to retrieve data efficiently in a suitable period. Our proposed work focuses on using of Hive-QL and Shark to query data on DBpedia instead of SPARQL. The question that may come to the mind is: why do we use both query languages? Moreover, do we use them in parallel? The answer to those questions is that we use both languages to add a stability as an advantage to our architecture which is not found to the best of our knowledge in other work done in the same area [10][14][15][16]. On the other hand, we do not use both languages in parallel; but in case of failure of one query language the other takes its place. Our architecture consists of five layers as [32] shown in Fig.2. Layer 0 contains DBpedia dataset in Comma Separated Values (CSV) which can be downloaded directly from DBpedia or via the conversion of RDF datasets to CSV. The datasets will be loaded to the HDFS as the Distributed File System on the Hadoop platform that is placed on Layer 1. While Layer 2 contains Hive, which includes Hive-QL as a query language and Spark that is considered a distributed processing framework different from Hive. Hive-QL depends on MapReduce functions for distributing files, and it works directly on Hive on the Hadoop platform. While Spark framework uses Shark, which is placed in Layer 4 in our architecture. On the other hand, the question that might be raised is: "Which one will be the primary query language Hive-QL or Shark?". We cannot answer this issue till testing both languages in our architecture. Finally, the upper layer in our architecture is the Application Programming Interface (API) that is considered the gate used by different applications to deal with the architecture.

6. Implementation

We used DBpedia 3.9 as our dataset that is considered the last version of DBpedia at the time of writing. It consists of 583 million triples that added new classes and properties to the DBpedia ontology via the mappings of Wikipedia. It consists of 735 classes, 1098 object properties, 1583 datatype properties, 132 specialized datatype properties, 408 equivalent classes and 200 equivalent properties according to the last statistics on the DBpedia website [29]. Our proposed work is considered a desktop application that is implemented using Scala functional programming language [30] on Intel Core i5 with 4 GB memory and 300 GBs hard disks run on Ubuntu 12.04 Linux Operating System on VMware [31]. The process in our architecture begins with the loading of CSV file into the HDFS of the Hadoop platform and then we can start the spark service on top of Hadoop if we use Shark as the query language. While CSV files will be loaded into the Hive table to be ready for querying using Hive-QL query language. Both Shark and Hive-QL use the same meta store that consider an advantage. These meta store services will be launched using Mysql server. Through our work with Shark and Hive, we discover a compatibility problem between the last version of Shark and the final version of Hive, which is upgraded to allow users to perform sub-queries to carry out non-trivial queries. Our implementation methodology proposed a solution to solve this problem by using version 0.9 of Hive and creating user defined functions that allow users to perform those sub-queries that are compatible with the last version of Shark.
7. Results

This section introduces the results of the test cases performed on our architecture that proposes Shark query language in addition to Hive-QL instead of SPARQL as query languages for the DBpedia datasets. Also, this section answers the question of which language will be the primary and will be used first depending on different factors such as the stability and the execution time. The execution time can be defined as the spent time of executing system’s task, including the time spent in run-time [33]. Our approach is tested using three different non-trivial queries as presented below. Our general schema depends on applying join operations and subselect queries to perform those non-trivial queries because DBpedia dataset treats with each subject, object as an individual entity. The three test queries that we used as a benchmark are described as follow:

The First Query (Q1):

The first query asks to retrieve the birthdate of the artist who drew the popular portrait that is called Mona-Liza. This query is performed using both of two query techniques Hive-QL and Shark. The execution time of the query using Hive-QL takes 111 secs as shown in Fig.3, while the execution time for the same query using Shark query language takes 65 secs as shown in Fig.4.

The Second Query (Q2):

The second query asks to retrieve the current football club for the top scorer player of the previous FIFA World Cup 2014 which was held in Brazil. This query is also considered a sophisticated query, and it has a subquery used to retrieve the top scorer name and then apply the result to another query to retrieve his current club. This query was performed using both of query techniques Hive-QL and Shark. The execution time of the query using Hive-QL takes 138 secs as shown in Fig.5, on the other hand, the execution time for the same query using Shark query language takes 61 secs as shown in Fig.6.

The Third Query (Q3):

The third query concerns retrieving the headquarter of the organization where the Egyptian scientist Farouk Al-baz work. Firstly we performed subquery to retrieve the organization name where Farouk Al-baz works and then we use this value to retrieve the location of this scientific organization. This query was performed using both of the two query techniques Hive-QL and Shark. The execution time of the query using Hive-QL takes 129 secs as shown in Fig.7, while the execution time of the same query using Shark query language takes 47 secs as shown in Fig.8.
The results of those three tested queries in both Hive-QL and Shark ensure that Shark query language is faster than Hive-QL in the execution time and performs better, while both are efficient in querying semantic datasets and retrieving the same data as shown in Table I and described clearly in Fig.9.

Table I. The Execution Time for HiveQL vs. Shark

<table>
<thead>
<tr>
<th>Query language</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hive-QL</td>
<td>111 secs</td>
<td>138 secs</td>
<td>129 secs</td>
</tr>
<tr>
<td>Shark</td>
<td>65 secs</td>
<td>61 secs</td>
<td>47 secs</td>
</tr>
</tbody>
</table>

7.1 Architecture Evaluation

Our evaluation of the proposed architecture will focus on the four criteria that Hive-QL met difficulties to satisfy and presented on [10] as shown in Table II, in addition to the execution time. We find that Shark is more stable than Hive-QL, in addition to that has a better execution time as described before, and both of them provide the same usability and availability. So, according to the high stability of Shark and better execution time we will use it as the primary query language for our architecture while HiveQL will be the secondary language in case of any failure that may occur while using Shark.

Table II. Architecture Evaluation

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Availability</th>
<th>Usability</th>
<th>Stability</th>
<th>Efficiency</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shark</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>HiveQL</td>
<td>+</td>
<td>+</td>
<td>+/</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

8. Conclusions and Future Work

This work proposes a new architecture, which presents an effective technique to query large amount of semantic content as in DBpedia datasets on distributed processing using Apache Spark environment based on Shark query language, in addition to the usage of Hive-QL. Also, our paper highlights our implementation that shows that Apache Spark and its query language Shark provide a better execution time and more stability. For that reason, we select it as the primary query language. Using both languages together in our architecture shows us an improvement in the stability. Therefore, our future work will concern building desktop applications based on DBpedia datasets using big data techniques.

References


[22] H. Alfeel, 'The Roadmap for the Arabic chapter of DBpedia', Wseas; 14th International Conference on Telecommunications and Informatics (TELE-INFO '15),2015


