ISSN: 0975-766X
CODEN: IJPTFI
Available Online through *Research Article*
www.ijptonline.com

A DEVELOPMENT OF RDF DATA TRANSFER AND QUERY ON HADOOP FRAMEWORK

Swathi*, Uma Priyadarsini P.S**
UG Scholar*, Assistant professor**
Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha University, Chennai.
Email:swathialis@gmail.com

Received on 10-08-2016 *Accepted on 06-09-2016*

Abstract

A RDF chart is regularly put away in XML document or social database. Nonetheless, when it turns into a huge RDF diagram, an option approach to handle the putting away and inquiry RDF chart or connected information is to utilize MapReduce calculation and Hadoop structure. In this paper, we propose a supporting apparatus to perform information exchange and inquiry on enormous RDF diagram. We plan to decrease the entrance time and question reaction time by utilizing Hadoop Framework. The RDF/XML or connected information is changed over into a tremendous arrangement of N-triples and they are transferred onto Hadoop putting away in information hubs of Hadoop Distributed File System (HDFS). The inquiry of RDF diagram as far as SPARQL is broke down and changed over into a particular N-triple organization as to pursuit the answer utilizing Jena Algebra. The MapReduce calculation is created to pertinently control the RDF diagram

Keywords: Big data, Apache Hadoop, MapReduce, SPARQL, RDF, N-Triple , Linked data, HDFS

1. Introduction

Semantic web is concerning how to publish a machine readable data so that the machine can understand what is the meaning of the contents and derive the next conclusion out of the existing ones. To manage these readable data for semantic web, one typical way is using linked data. The linked data contains nodes and connections. The uniform resource identifier (URIs) technology is basically used to identify each node which is stored as web data anywhere in the world. However, when the linked data becomes huge and complex, the data manipulation and searching are more time-consuming and inefficient. Several technologies are common to solve these problems and become the semantic web standard [1]. These are RDF (Resource Description Framework), SPARQL (SPARQL Protocol and RDF Query

Language), and OWL (Web Ontology Language). Moreover, Hadoop Technology is one the best choices to archive and searching the RDF data.

2. Background and Related Work.

It is difficult to find a solution for the big data storage and its access issue. Moreover, the data becomes unstructured and likely to explode. The researches of Husain, M. F., [2] and Chooan R. [3] focused on data storage and its access by using Hadoop for RDF data, the access is more efficient after applying the MapReduce principle. However, it takes a longer time for data query if there are too many data nodes and the data is not sufficient large. Actually, data access in each level needs an appropriate querying. Joldzic O. V. [4] has gotten the result of a comparison between data querying from the data in RDBMS and the other one from the data in HBase. He told changing query and sorting a data access table yield an improvement of performance on any storage. Olaf Hartig [5] and Park, S. [6], Researched about performance on display data processing between use data server and add visualization via MapReduce. The data access influences a processing and the final visualization in a system. After that, they utilize Hadoop framework with the MapReduce concept by making it participating in processing. This technology is found to increase the performance. However, HTTP protocol causes data access and visualization take longer time from protocol delay. It is also probably to cause a failure to happens. So, the best selection of data storage methods depends on a volume, a pattern of data access and an amount of data node in a system.

Algorithm and Implementation

This paper proposes a support tool to convert linked data into RDF/XML. We populate our big test linked data using The Lehigh University Benchmark (LUBM) [7]. Then the RDF/XML is translated into N-triple form to be ready for uploaded into Hadoop. The uploaded RDF N-triples are distributed stored into Hadoop's cluster. When it comes to search and query, the query written by SPARQL is given and we develop Jena algebra to translate these SPARQL into the Basic Graph Pattern (BGP). These BGP is concerned when we develop map and reduce functions. A map function specifies parameter key and a reduce function help post processing to combine and get the final answers in term of XML format.

We have two phases processing. The first phase concerns the preparation of the RDF N-triple from the given linked data. We use LUBM tool to generate the test data suite of linked data in the form of RDF/XML. The sample of the generated RDF/XML .

The RDF/XML linked data is translated into N-triple form correctly. We ensure the N-triple holding in the well formed Subject – Predicate - Object relation before proceeding to the next step. The well formed N-triples of the LUBM test RDF data are now uploaded into HDFS using the relevant commands at command line. The HDFS performs its distributed storing into Hadoop's clusters.

The well formed Subject – Predicate - Object relation of the RDF N-triples.

For the second phase, we provide how to access the stored RDF N-triple in the Hadoop with SPARQL in Fig. 1. The query written in SPARQL is translated a basic graph pattern (BGP). We provide a template code to translate SPARQL and it is easy to just fill in the parameters from SPARQL. A tool is developed to automatically generate BGP from SPARQL using Jena algebra.

The sample of the query is shown in Fig. 4. The BGP query is exploited in the MapReduce algorithm. The fundamental concept of the map and reduce functions used including domainMap (k1, v1) Í list(K2, v2) and domainReduce (K2, list(v2)) Í list(v3).

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
3 SELECT ?X
4 WHERE
5 {?X rdf:type ub:GraduateStudent .
6  ?X ub:takesCourse "http://www.Department0.University0.edu/GraduateCourse0"
7 }
```

Fig. 1. The sample of the query.

A. The map algorithm for BGP query

We design a map algorithm to get key(k1) and value(v1) in the Hadoop's cluster. Thus, the return list of new key consists of the predicate of data in line along with the value, which is subject concat object data for feed list of the reduce algorithm. The sample of the map algorithm is shown in Fig. 2.

```
1  function map
2      Map(Key, Value, Context)
3          while(Value is not null)
4             write(value.predicate,
5                   value.subject + value.object)
6          End while
7      End Map
8  End function
```

Fig. 2. Sample of the map algorithm for BGP query.

B. The reduce algorithm for BGP query

We also design reduce algorithm to perform the post processing of the result from the map algorithm. In addition, the inputs of the reduce algorithm are the list of new key value and SPARQL from files. The SPARQL query will be converted into the BGP patterns. All queries show the well formed structure as Subject – Predicate – Object. The BGP query could be categorized into three patterns as follows.

Pattern 1) ?subject predicate Data object Data

Pattern 2) ?subject predicateData objectData Pattern 3) subjectData predicateData ?object

According to the query formats mentioned above, we develop the reduce algorithm as shown in Fig.3 .

```
function reduce(Key, Values, Context){
    do  algebra convert to BGP
    do checkFormatQuery(String sqlInput)

    processMethod(sqlType)
        for i = 0 to i < sqlType size
            if match pattern 1
                while value is not null
                    map predicate with value
                    map object with value
                End while
            else if match  pattern 2
                while value is not null
                    map predicate with value
                    map subject with value
                    map object with value
                End while
            else if match  pattern 3
                while value is not null
                    map predicate with value
                    map subject with value
                End while
            End if
        End for

    do addToListResult
    do convertToXML
    End processMethod
End function

Function checkFormatQuery(String sql){
    for i = 0 to i < sql line
        if match pattern 1
            return sqlType[subject]
        else if match  pattern 2
            return sqlType[subObj]
        else if match  pattern 3
            return sqlType[object]
        End if
    End for
}
```

Fig. 3. Sample of our reduce algorithm for BGP query.

Experiments and Result

In this section, we describe how to setup the configuration for Hadoop, data benchmark, sql and result of experiment. We install and run the software with the system configuration specified in Table I. The RDF/XML test data from LUBM is a collection of 3,284 XML files with size 1.74 gigabytes. It took 37.40 minutes to convert into RDF N-triple. Then, the N-triple RDF file is now 3.76 gigabytes. When this N-triple RDF file is splitted into 600 files with 537 blocks of directory.

Table-I. Configuration setup and parameter.

Environment setup	Parameter
Platform	CentOS release 6.7
Ram	8 GB
Hard disk	50 GB
Hadoop	2.7.1
Data node	1
Java	1.7.0_95

A. Data RDF/XML

Benmarch in the experiment use The Lehigh University Benchmark (LUBM). It is separated to part of file 3,284 files. Size of dataset 1.74 GB. Time to use convert from RDF/XML to N-Triple is 37.40 minutes.

B. Data N-Triple

From convert LUBM data to N-Triple have size of data file about 3.76 GB.

C. Import file to Hadoop's cluster

Importing file N-Triple after convert data to Hadoop's cluster by command line. It will split 600 files and directories, 527 blocks = 1,127 total filesystem objects.

D. SPARQL query and result

Query1. This query bears large input and high selectivity. It queries about just one class and one property and does not assume any hierarchy information or inference.

In Fig. 4. We choose Query1. for inqueries from Hadoop's cluster and in Fig. 5. BGP query after convert from SPARQL.

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
3 SELECT ?X
4 WHERE
5 {?X rdf:type ub:GraduateStudent .
6  ?X ub:takesCourse "http://www.Department0.University0.edu/GraduateCourse0"
7 }
```

Fig. 4. SPARQL query: Query1.

```
(project (?X)
 (BGP
  (triple ?X
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#GraduateStudent>)
  (triple ?X
  <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#takesCourse>
  "http://www.Department0.University0.edu/GraduateCourse0")
 ))
```

Fig 5. BGP query format: Query 1.

We show result on graph for time to access performance separated by number of triples shown in Fig. 6. The result with using Query1. We can analyze performance access data for this query in Hadoop's cluster trend of time to use is higher follow increase number of triples. It takes time to higher than more on one point when size of triple and number of data node not balance

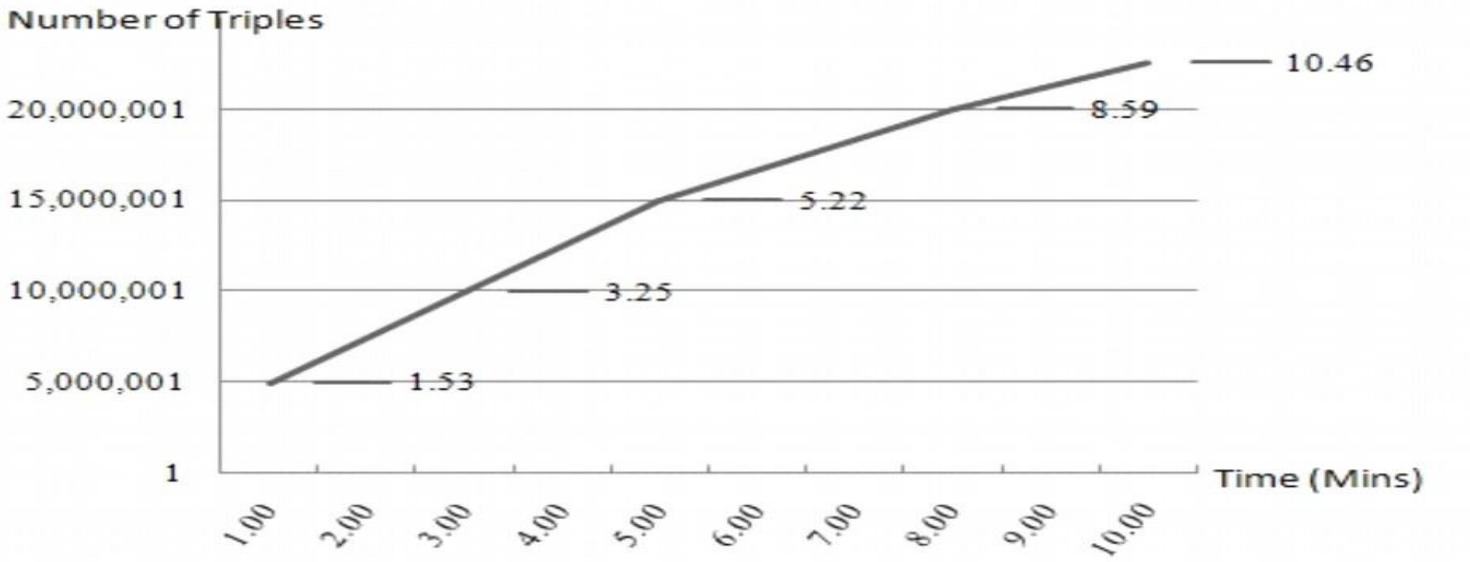

Fig 6. Graph time to use access by Query1.

Conclusion

In this paper, we propose a supporting tool to perform RDF data transfer onto Hadoop and perform SPARQL query to get the answers. We provide the map and reduce algorithms in the generic form. Our SPARQL is translated into BGP query using Jena algebra automatically. The BGP query is to be used in the map and reduce algorithms. The BGP query is analyzed to extract the parameters which are relevant to the map and reduce algorithms. In order to test our algorithms,

we populate the linked data from LUMB and have them prepared and uploaded into the Hadoop's cluster. The BGP query is finally generated and the map and reduce functions are developed. The performance test is conducted.

References

1. Husain, M. F., Doshi, P., & Khan, L. (2009). Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce, 680–686.
2. (Online Resource) Tutorial 2: Introducing RDF/XML. http://www.linkeddatatools.com/introducing-rdf-part-2
3. Choopan Rattanapoka (2012). The Design and Implementation of Computer Traffic Log Searcher System using Hadoop Map/Reduce Framework., The Journal of Industrial Technology, Vol. 8, No. 3September – December 2012.
4. Joldzic, O. V., &Vukovic, D. R. (2013). The impact of cluster characteristics on HiveQL query optimization. 2013 21st Telecommunications Forum Telfor (TELFOR), 837–840. doi:10.1109/TELFOR.2013.6716360
5. Olaf Hartig , Christian Bizer, Johann-Christoph Freytag. Executing SPARQL Queries over the Web of Linked Data, ISWC 09, 2009.
6. Park, S. (2013). Visualization of Resource Description Framework
7. Ontology Using Hadoop, 228–231. doi:10.1109/IMIS.2013.46
8. (Online Resource) The Lehigh University Benchmark (LUBM). http://swat.cse.lehigh.edu/projects/lubm/