Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 9 (1): 195-202, 2015
ISSN 1819-4311 / DOI: 10.3923/jse.2015.195.202
© 2015 Academic Journals Inc.

Review of Large-Scale RDF Data Processing in MapReduce

[1,2]Ke Hou, [1]Jing Zhang and [2]Xing Fang
[1]School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, 710048, China
[2]School of Economic and Management, Xi'an Shiyou University, Xi'an 710065, China

Corresponding Author: Ke Hou, School of Economic and Management, Xi'an Shiyou University, No. 18 East 2th Dianzi Road, Xi'an, Shaanxi, 710065, China Tel: +86 29 8838 2653

ABSTRACT
Resource Description Framework (RDF) is an important data presenting standard of semantic web and how to process, the increasing RDF data is a key problem for development of semantic web. MapReduce is a widely-used parallel programming model which can provide a solution to large-scale RDF data processing. This study reviews the recent literatures on RDF data processing in MapReduce framework in aspects of the forward-chaining reasoning, the simple querying and the storage mode determined by the related querying method. Finally, it is proposed that the future research direction of RDF data processing should aim at the scalable, increasing and complex RDF data query.

Key words: RDF, RDF reasoning, RDF query, RDF storage

INTRODUCTION
Semantic data description is the standard method to describe basic data and manage knowledge base in many fields. The Resource Description Framework (RDF) and the RDF Schema (RDFS) are main identity information of semantic data. With the expanding of semantic web, the RDF data is increasing and its amounts reach the large-scale level, so how to process massive RDF data becomes the key problem of developing semantic web. The distributed computing and parallel computing technologies provide better solutions to process large-scale data, in which the MapReduce is an outstanding programming model. MapReduce has good scalability, data-locality and reliability and it becomes the most widely-used data-intensive computing model. In this study, a synthesis analysis of recent research results on RDF data reasoning, query and storage in MapReduce is shown.

RDF DATA REASONING
Recently, there are mainly the forward-chaining reasoning about RDF data reasoning researches Forward-chaining reasoning is data driving, the total reasoning process begins with data fact and executes repeated iteration under related rules to get all of the solution spaces with the given data. According to logic process, the incremental reasoning algorithm can be divided into monotonic reasoning and nonmonotonic reasoning.

Urbani *et al*. (2009) proposed the RDFS forward-chaining reasoning algorithm on MapReduce. It is a direct RDFS data processing based on MapReduce implementation which focuses on RDFS closure infinite and ignores some RDFS rules and derives. This algorithm also is optimized in respect of the derived copies, the triple connecting pattern and the fixed-point iterations. It is implemented on Hadoop and 865M triple data with 64 machines are handled within two hours.

Husain *et al*. (2009) used MapReduce and Hadoop to store and query RDF graphs. They divided a big RDF graph into small files and stored them in Hadoop and then generated one or more MapReduce tasks and used SPARQL querying to get the result. But this method cannot support the interaction during the query and some useful information is hardly found. Moreover, this method cannot support the SPARQL keywords definition, such as UNION, DISTINCT and FILTER. In addition, their study does not give the details of map and reduce execution.

Schlicht and Stuckenschmidt (2011) thought that the big disadvantage of using MapReduce to execute semantic reasoning is repeat reasoning. Although performing separately MapReduce task in limited RDFS calculation can avoid this problem, in the fixed-point iterations of more ontology the repeat reasoning problem will appear. They proposed a Mapresolve method based on MapReduce optimizing to process more expression logic. This method can cause task node idle because the tasks are into streaming structure, the read and write process of each iteration will cause extra overhead. But it is the cost to ensure fault tolerance and usability.

The scalable parallel processing method WebPIE (Urbani *et al*., 2012) can use RDF closure in MapReduce to process large-scale RDF dataset. Although WebPIE is verified and is thought as an incremental reasoning, its performance still relies on the input data. In other words, this method does not consider the relationship between previous data and increasing data.

The nonmonotonic reasoning method of semantic scale data in MapReduce (Tachmazidis *et al*., 2012) use the defeasible stratified logics. The result shows that this method has good scalability. It is twice as fast as monotonic reasoning to process one billion triple basic data.

Due to more and more monotonic increasing transitive closures Jang and Ha (2013) proposed the distributed processing method of large-scale RDF dataset with MapReduce.

To resolve the non-monotonicity of knowledge base expanding with increasing knowledge, the nonmonotonic reasoning of social science data increment with MapReduce is implemented (Antoniou *et al*., 2014). To support decision-making, this method can clean interference data and extract high-level information from low-level input data.

To accelerate process of data update and meet online search demand of user, Liu *et al*. (2014) put forward an incremental and distributed Inference Method (IDIM) with MapReduce and Hadoop. This method can balance old and new data and shorten update time, thereby reduce the reasoning time of large-scale RDF data.

RDF reasoning with MapReduce is to solve the massive data processing problems under the background of different reasoning purposes and the RDF rules constraints. By using the MapReduce programming model, scholars implemented the iteration, data update and data repetition during RDF reasoning. They use the defeasible logic, non-monotonicity and incremental reasoning methods to reduce the reasoning time of massive RDF data. Although every method has its own innovation and improvement, most of them are stopped at the laboratory level and need a further apply in industries.

RDF DATA QUERY

The distributed query of RDF data is to do the RDF data search on distributed system. According to definition the RDF data query can be divided into the query on SPARQL and the query on other frameworks. Because the data is distributed on different nodes, query sentences need to be decomposed according to rules of dividing data. Therefore, the RDF data query in distributed environment includes decomposition of the join query and the distributed query. Besides, those two queries can be divided from data distribution and query language constitution.

Mika and Tummarello (2008) used MapReduce to calculate closures in large RDF graph and response SPARQL query request but did not provide any details.

SPIDER querying framework (Choi *et al.*, 2009) includes loading and querying of RDF graph. The former can enter SPIDER store and the latter queries RDF sub-graph defined by SPRAQL querying format. When SPIDER is realized in MapReduce, the map task firstly search parts of sub-graph matching, then transmit the matching result to the reduce task which handle the query request separately and give results. Because every node only store RDF data partly, SPIDER needs merge sub-query results of each nodes. This process will weaken performance of the system. Although RDF data is sorted in SPIDER, there is still the before-mentioned performance loses. The experiment data come from 100 million triples in Berlin SPARQL Benchmark database.

The data flow language RAPID (Sridhar *et al.*, 2009) is an extension of Pig for RDF data querying. The RAPID language defines the type, property and path data. The query is divided into Generating Fact Diagram (GFD), Generating Basic Diagram (GBD) and Multi-Dimensional Join (MDJ) and their MapReduce implementation are given. The RAPID+ model (Ravindra *et al.*, 2010) reconstructs the RDF graph by group, increases data processing parallel and reduces I/O costs. The forward querying method based on RAPID+ model can merge calculation steps and simplify process. Although RAPID+ model are tested on 120 million triples, those are only basic data query of RDF graph and do not involve complex query.

Myung *et al.* (2010) proposed a MapReduce algorithm for SPARQL basic graph mode. Running multiple tasks on MapReduce will bring much extra overhead, for example, at the beginning of every query the calculation resources are needed to be reset, map function and reduce function will sort data. Hence, this algorithm uses the greedy strategy to realize selection of join key in SPARQL basic graph mode query sentences and applies multi-way join in MapReduce query task which can reduce the useless task iteration. In the simulative cloud environment, experimental results show that the calculation performance and scalability of the method is better than traditional SPARQL query tools while executing SPARQL query on large-scale RDF data.

Kotoulas and Urbani (2010) gave the preliminary definition of the common analysis query, store the possible query name into the separate table in which indexes are loaded. So, SPARQL query can find the corresponding index by the query name. This method can reduce connection request but there will be vast index storage and the data update will bring enormous overhead for rebuilding index.

PigSPARQL (Schatzle *et al.*, 2011) is a directly SPARQL mapping method realized on Pig. Although it did not do any optimization, it is proved that the SPARQL query can directly implement with MapReduce and does not need any other programming model.

According to data distribution, the query is decomposed into parallelizable without communication (PWOC) sub-query (Huang *et al.*, 2011), every node only implements one sub-query. The nearby points of RDF graph are divided to the same machine node. It can execute the SPARQL query effectively and do not need the communication within nodes. While doing this in MapReduce framework, the communication between nodes is still necessary.

Kim *et al.* (2011) divided the query into many star sub-query. While processing the query, every map-reduce distribution and recycling iteration only needs handle one star sub-query. The final result will be gotten after the sub-query results are jointed.

Husain *et al.* (2011) proposed a heuristics processing framework of large-scale RDF data query based on MapReduce. They designed the SPARQL basic graph to divide RDF data into different predicate files and then divide further the predicate file according to the types of objects. The query is classified by object types and the related data files are loaded to query. While doing the RDF query, a greedy MapReduce task generation algorithm uses multiple MapReduce tasks to process the SPARQL BGP join operation iteratively and each task processes preferentially the triple pattern sub-sentence in which the shared variable appear most times. This type of query response strategy is too simple to guarantee the efficiency of query.

Ravindra *et al.* (2011) thought that the current MapReduce system realizes the related join operation by implanting the multi-join query into map-reduce iteration which cause heavy I/O load and communication cost. In view of SPARQL graph match mainly demands join operation, they proposed the combination operation for optimizing graph match which is achieved through reinterpreted the join tree. The middle data are treated as "triple set" which can reduce the map-reduce iteration during the query. Comparing with Pig and RAPID+, this method has 60% performance increase.

Pig language is used to execute SPARQL query (Kotoulas *et al.*, 2012) and this method is achieved in MapReduce model which can be used in RDF forward-chaining reasoning. It can process complex SPARQL 1.1 query and optimize join operation of runtime and give the solution to the skew problem in MapReduce.

To satisfy the demand of social network data analysis described by RDF framework, two primitives (Liu *et al.*, 2013) are proposed to achieve higher efficiency while using MapReduce to process the SPARQL query.

The request speed and efficiency are two important aspects to evaluating the MapReduce tasks of RDF querying. The common process of querying are as follows: Input data, generating the index table, processing query and merge the result. To optimize those processes, the above literatures use some methods, for example, choosing different RDF primitives, dividing index table in different ways and revising the execute ways of MapReduce query. Similar with RDF reasoning, the RDF query process also has updating massive data, renewing index table problem and so on which are waiting for future researches.

RDF DATA STORAGE

According to partition results, RDF data is stored with key/value pairs. The partition strategies include the partition on graph structure and storage with key/value pairs directly.

The RDF molecule method (Newman *et al.*, 2008) is put forward on integration and processing of RDF protein interaction data. The RDF molecule definition includes RDF triple (subject, predicate, object) and molecule ID (spom, posm, ospm). The RDF molecule is a medium granularity between RDF graph and triple. This method can divide RDF graphs into small units which are decomposed and merged on MapReduce. This RDF molecule base provides expanding method for storing and processing large-scale biomedicine data distributed. But the RDF molecule query method in detail is not given and the amount of data is only 4015778 triples.

SPIDER (Choi *et al.*, 2009) uses HBase database to store RDF data. Because RDF uses triples to represent data, while HBase is oriented to column-storage, RDF data need to be transferred by column-oriented way. For RDF data is sparse itself, it is suited to store in HBase. But strategy of storing RDF triple in HBase and experimental results are not introduced minutely.

Sun and Jin (2010) provided the RDF storage solution in HBase and a SPARQL basic graph query method with MapReduce. According to composition of subject-verb-object triples, they designed six kinds of HBase tables to match the possible SPARQL triple pattern and proposed the method of selecting join key based on greedy strategy. But this method will copy RDF data six times and need more storage space for ontology data. While modifying and deleting the RDF data, six tables will be accessed simultaneously. Because HBase database does not support transaction processing, this method also has data synchronous problem.

The triple storage SHARD (Rohloff and Schantz, 2010) based on Hadoop, supports large-scale, high-powered and robust data storage. The method of establishing structure information system with MapReduce is also given. In 2011, the MapReduce model based on SHARD for optimize graph data processing is proposed, it uses sub-sentences and iteration method to respond SPARQL query request of RDF data.

Husain *et al.* (2011) proposed a solution to store and process large-scale RDF data with a HDFS and MapReduce. RDF data are stored in HDFS as N-triples file to guarantee the integrity of RDF data and convenience of implementing MapReduce task. Nevertheless there are some shortages. Firstly, the RDF data file must be reprocess in order to store in HDFS. Moreover, storing in HDFS directly is lack of efficient index structure. Secondly, for all of the solutions in this study are based on HDFS file system, it is hard to modify large-scale RDF data randomly. Hence, using HBase and other distributed database system to store ontology data is easy to access and modify data randomly.

Through the distributed computing ability of cloud platform, TripleCloud (Gueret *et al.*, 2011) stores triples into Hbase in the form of key/value pair which can reduce the cost of inputing large-scale RDF dataset and achieve management of RDF data.

Franke *et al.* (2011) proposed a distributed semantic web data management framework based on HBase. They designed a RDF data strategy using two HBase data table and realized a MapReduce algorithm for SPARQL basic graph mode query. Compared with MySQL cluster database experiment, it is certificated that the scalability and data query efficiency of semantic web data management under cloud environment is better than traditional relational database. But their solution is still limited in the calculation of SPARQL basic graph.

HadoopRDF (Du *et al.*, 2012) is a RDF data storage and analysis system based on Hadoop and Sesame which is a RDF store system. It deploys a Sesame Server instance on every node of Hadoop cluster, then uses the Sesame interface to store and query RDF data. HadoopRDF uses Hadoop to ensure high reliability and fault tolerance recovery ability. Similarly, MapReduce is useful for parallel implementation of RDF data query.

H2RDF (Papailiou *et al.*, 2012) is a cloud RDF data query system based on HBase and MapReduce. It indexes RDF data three times as SPO, POS and OSP and stores them into HBase as key/value pairs. There are two types of query method in H2RDF which are MapReduce query and Centralized query. For the SPARQL query, the H2RDF first uses Jena to analysis the query and then uses join algorithm to decide query method. The purpose of this operation is to get higher efficiency which can use centralized query to perform simple query and use MapReduce to perform complex query. In 2014, H2RDF+ which extends H2RDF is proposed to get better performance.

As the basic level in the system, RDF storing assists an effective execution of reasoning and query, the distributed store of cloud platform provides more possible for massive RDF data storage. Current researches are aiming at improving the storage system based on cloud platform (e.g., HBase) and doing some modifications according to RDF data characteristics.

CONCLUSION

Using MapReduce to process RDF data is the leading edge of semantic web research and it is useful to develop a scalable fundamental data processing platform. The current researches of RDF data processing focus on the forward-chaining RDF reasoning, the simple RDF query and the RDF storage strategy which are related to the query method. In future, researches of RDF data processing should aim at the large-scale, dynamic growing and complex RDF data query.

ACKNOWLEDGMENT

This study is supported by the 2007 Nationality 863 Project "Software Product Line based on Workflow" (No. 2007AA010305), the 2012 Shaanxi Provincial Department of Education project "Research on application strategy of mass data processing technology for oil and gas resources enterprise in Shaanxi Province" (No. 12JK0065), the State Key Laboratory for Manufacturing Systems Engineering Open Project (No. sklms2011011). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers which have improved the presentation.

REFERENCES

Antoniou, G., J.Z. Pan and I. Tachmazidis, 2014. Large-scale complex reasoning with semantics: Approaches and challenges. Proceedings of the International Workshops on BigWebData, MBC, PCS, STeH, QUAT, SCEH and STSC, October 13-15, 2013, Nanjing, China, pp: 1-10.

Choi, H., J. Son, Y.H. Cho, M.K. Sung and Y.D. Chung, 2009. Spider: A system for scalable, parallel/distributed evaluation of large-scale rdf data. Proceedings of the 18th ACM Conference on Information and Knowledge Management, November 2-6, 2009, Hong Kong, China, pp: 2087-2088.

Du, J.H., H.F. Wang, Y. Ni and Y. Yu, 2012. Hadooprdf: A scalable semantic data analytical engine. Proceedings of the 8th International Conference on Intelligent Computing Theories and Applications, Volume 7390, July 25-29, 2012, Huangshan, China, pp: 633-641.

Franke, C., S. Morin, A. Chebotko, J. Abraham and P. Brazier, 2011. Distributed semantic web data management in hbase and mysql cluster. Proceedings of the IEEE International Conference on Cloud Computing, July 4-9, 2011, Washington, DC., pp: 105-112.

Gueret, C., S. Kotoulas and P. Groth, 2011. Triplecloud: An infrastructure for exploratory querying over web-scale rdf data. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, August 22-27, 2011, Lyon, pp: 245-248.

Huang, J., D.J. Abadi and K. Ren, 2011. Scalable sparql querying of large rdf graphs. Proc. VLDB Endowment, 4: 1123-1134.

Husain, M., J. McGlothlin, M.M. Masud, L. Khan and B. Thuraisingham, 2011. Heuristics-based query processing for large rdf graphs using cloud computing. IEEE Trans. Knowledge Data Eng., 23: 1312-1327.

Husain, M.F., P. Doshi, L. Khan and B. Thuraisingham, 2009. Storage and retrieval of large rdf graph using hadoop and mapreduce. Proceedings of the 1st International Conference on Cloud Computing, December 1-4, 2009, Beijing, China, pp: 680-686.

Jang, B. and Y.G. Ha, 2013. Transitivity reasoning for rdf ontology with iterative mapreduce. Proceedings of the IEEE 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, July 3-5, 2013, Taichung, pp: 232-237.

Kim, H.S., P. Ravindra and K. Anyanwu, 2011. From sparql to mapreduce: The journey using a nested triplegroup algebra. Proc. VLDB Endow, 4: 1426-1429.

Kotoulas, S. and J. Urbani, 2010. Sparql query answering on a shared-nothing architecture. Proceedings of the SemData Workshop and VLDB, September 13-17, 2010, Singapore, pp: 90.

Kotoulas, S., J. Urbani, P. Boncz and P. Mika, 2012. Robust runtime optimization and skew-resistant execution of analytical sparql queries on pig. Proceedings of the 11th International Semantic Web Conference, Volume 7649, November 11-15, 2012, Boston, MA., USA., pp: 247-262.

Liu, B., K. Huang, J. Li and M. Zhou, 2014. An incremental and distributed inference method for large-scale ontologies based on mapreduce paradigm. IEEE Trans. Cybernetics, Vol. PP. 10.1109/TCYB.2014.2318898

Liu, L., J. Yin and L. Gao, 2013. Efficient social network data query processing on mapreduce. Proceedings of the 5th ACM workshop on HotPlanet, August 12-16, 2013, Hong Kong, China, pp: 27-32.

Mika, P. and G. Tummarello, 2008. Web semantics in the clouds. IEEE Intelli Syst., 23: 82-87.

Myung, J., J. Yeon and S.G. Lee, 2010. Sparql basic graph pattern processing with iterative mapreduce. Proceedings of the Workshop on Massive Data Analytics on the Cloud, April 26, 2010, Raleigh, North Carolina, USA., pp: 6.

Newman, A., Y.F. Li and J. Hunter, 2008. Scalable semantics: The silver lining of cloud computing. Proceedings of the IEEE 4th International Conference on eScience, December 7-12, 2008, Indianapolis, IN., pp: 111-118.

Papailiou, N., I. Konstantinou, D. Tsoumakos and N. Koziris, 2012. H2rdf: Adaptive query processing on rdf data in the cloud. Proceedings of the 21st International Conference Companion on World Wide Web, April 16-20, 2012, Lyon, France, pp: 397-400.

Ravindra, P., V.V. Deshpande and K. Anyanwu, 2010. Towards scalable rdf graph analytics on mapreduce. Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud, April 26, 2010, Raleigh, NC., pp: 5.

Ravindra, P., H. Kim and K. Anyanwu, 2011. An intermediate algebra for optimizing rdf graph pattern matching on mapreduce. Proceedings of the 8th Extended Semantic Web Conference on Heraklion, Crete, May 29-June 2, 2011, Greece, pp: 46-61.

Rohloff, K. and R.E. Schantz, 2010. High-performance, massively scalable distributed systems using the mapreduce software framework: The shard triple-store. Proceedings of the Programming Support Innovations for Emerging Distributed Applications, October 17-21, 2010, Reno, NV., USA., pp: 4.

Schatzle, A., M. Przyjaciel-Zablocki and G. Lausen, 2011. Pigsparql: Mapping sparql to pig latin. Proceedings of the International Workshop on Semantic Web Information Management, June 12-16, 2011, Athens, Greece, pp: 4.

Schlicht, A. and H. Stuckenschmidt, 2011. Mapresolve. Proceedings of the 5th International Conference On Web Reasoning and Rule Systems, August 29-30, 2011, Galway, Ireland, pp: 294-299.

Sridhar, R., P. Ravindra and K. Anyanwu, 2009. Rapid: Enabling scalable ad-hoc analytics on the semantic web. Proceedings of the 8th International Semantic Web Conference, October 25-29, 2009, Washington, DC., pp: 715-730.

Sun, J. and Q. Jin, 2010. Scalable rdf store based on hbase and mapreduce. Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering, Volume 1, August 20-22, 2010, Chengdu, pp: V1-633-V1-636.

Tachmazidis, I., G. Antoniou, G. Flouris and S. Kotoulas, 2012. Scalable nonmonotonic reasoning over RDF data using mapreduce. Proceedings of the Joint Workshop on Scalable and High-Performance Semantic Web Systems, November 11, 2012, Boston, USA., pp: 75-90.

Urbani, J., S. Kotoulas, E. Oren and F. Van Harmelen, 2009. Scalable distributed reasoning using mapreduce. Proceedings of the 8th International Semantic Web Conference, October 25-29, 2009, Chantilly, VA., USA., pp: 634-649.

Urbani, J., S. Kotoulas, J. Maassen, F. Van Harmelen and H. Bal, 2012. Webpie: A web-scale parallel inference engine using mapreduce. Web Semantics: Sci. Services Agents World Wide Web, 10: 59-75.