Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 3, March 2016, pg.86 – 92

SELECTIVITY ESTIMATION USING CUSTOMIZED N-TRIPLE TEMPLATE IN RDF

Reena Parmar[1], Hiral Darji[2], Hitul Patel[3]
[1]M.E in C.E, Swaminarayan College of Engineering &Technology (S.C.E.T., Kalol), India
[2]Asst. Prof. in Swaminarayan College of Engineering & Technology (S.C.E.T., Kalol), India
[3]HOD of ME(CE) in Swaminarayan College of Engineering & Technology (S.C.E.T., Kalol), India
reena.parmar32@gmail.com; hiral1516@gmail.com; hitulce@gmail.com

ABSTRACT - Big Data Processing relates to Processing of very large scale data. It includes complex data analysis. RDF is very popular type of semantic Web data that store and retrieve huge amount of useful data as Resource. RDF stands form Resource Description Framework. One useful representation of RDF data is N-Triple, in which Subject-Predicate-Object model is used to represent RDF Data. I am going to develop customized Template for RDF file which can be used for accessing data faster from RDF file. To get data from RDF file we will use SPARQL using Apache Jena. I can also apply Bloom Filter for Text data in RDF file. RDF file and Query file will be input for our algorithm and after processing new RDF file will be generated using customized template that give faster query result then old RDF. The generation of RDF data has accelerated to the point where many data sets need to be partitioned across multiple machines in order to achieve reasonable performance when querying the data. Although tremendous progress has been made in the Semantic Web community for achieving high performance data management on a single node, current solutions that allow the data to be partitioned across multiple machines are highly inefficient. I introduce a scalable RDF data management system that is up to three orders of magnitude more efficient than popular multi-node RDF data management systems.

Keywords— MapReduce, Bloom Filter, Hadoop, SPARQL, Selectivity Estimation, RDF, NTriple

I. INTRODUCTION

This paper focuses on selectivity estimation for SPARQL graph patterns, which is crucial to RDF query optimization. The previous work takes the join uniformity assumption, which would lead to high inaccurate estimation in the cases where properties in SPARQL graph patterns are correlated. We take into account the dependencies among properties in SPARQL graph patterns and propose a more accurate estimation model. We first focus on two common SPARQL graph patterns (star and chain patterns) and propose to use Bayesian network and chain histogram for estimating the selectivity of them. Then, for an arbitrary composite SPARQL graph pattern, we maximally combines the results of the star and chain patterns we have pre-computed. The experiments show that our method outperforms existing approaches in accuracy. Since the use of RDF to represent data has grown dramatically over the last few years, query processing on RDF data becomes an important issue. Selectivity estimation for SPARQL graph patterns is crucial to RDF query processing. As we know, RDF data is a set of triples with the form (subject, property, object). This fine-grained model leads to SPARQL queries on RDF data with a large number of joins. As such, precise estimation of the selectivity of joined triple patterns is very important.

II. Introduction to RDF

RDF Query Language2 (SPARQL). RDF is the standard for storing and representing data and SPARQL is a query language to retrieve data from an RDF store. The power of these Semantic Web technologies can be successfully harnessed in cloud computing environment to provide the user with capability to efficiently store and retrieve data for data intensive applications. Synergy between the semantic web and cloud computing fields offers great benefits, such as standards for data representation across frameworks. One such standard is the Resource Description Framework (RDF). With the explosion of semantic web technologies, large RDF graphs are common place. Current frameworks do not scale for large RDF graphs. I describe a framework that we built using Hadoop, a popular open source framework for Cloud Computing, to store and retrieve large numbers of RDF triples. We describe a scheme to store RDF data in Hadoop Distributed File System. We present an algorithm to generate the best possible query plan to answer a SPARQL Protocol and RDF Query Language (SPARQL) query based on a cost model. I use Hadoop's MapReduce framework to answer the queries. Our results show that we can store large RDF graphs in Hadoop clusters built with cheap commodity class hardware. Furthermore, we show that our framework is scalable and efficient and can easily handle billions of RDF triples, unlike traditional approaches.

III. Introduction to MapReduce

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmer's the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day. Over the past years, the authors and many others at Google have implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of web documents, summaries of the number of pages crawled per host, the set of most frequent queries in a given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues. As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a *map* operation to each logical .record. in our input in order to compute a set of intermediate key/value pairs, and then applying a *reduce* operation to all the values that shared the same key, in order to combine the derived data appropriately. Our use of a functional model with user specified map and reduce operations allows us to parallelize large computations easily and to use re-execution as the primary mechanism for fault tolerance. The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.

IV. Introduction to Hadoop

Hadoop is an open-source software for reliable scalable, distributed computing. Many large-scale data analysis tasks have been done on this platform. Using a cluster with lots of computers, Hadoop provides a powerful computing ability to handle the scalability well. Hadoop consists of two layers, the data storage layer or the Hadoop Distributed File System (HDFS) and the data processing layer or the MapReduce Framework. HDFS is a block-structured system. Files are cut into blocks and stored in the cluster nodes which are called datanode. And there is a namenode which contains metadata about the size and the location of the blocks. MapReduce is a framework for the computing in Hadoop, which follows master-slave architecture. Each job is broken into map jabs and reduce jobs, which are executed in the slave nodes. HadoopRDF combines both of the systems. Hadoop is used as the base architecture. A cluster is arranged by hadoop which is considered as the communication and controlling tool. All the jobs are ordered and arranged by the internal rules in hadoop cluster. RDF triple store is placed in the slave nodes in the cluster which supplies the basic storage and retrieves needs. To make the methods e_ective, several special parts should be considered during the design of the system. As data on the public web has been expanded rapidly and more data is represented in the form of RDF as for its advantages, the scalability problem while dealing with it becomes important. To store and retrieve the RDF data, triple stores such as same, 4store, have been researched. However, these tools are designed on a single computer and the ability to handle the scalability is limited. Hadoop is an open source platform for distributed computing and has been widely used for largescale data analysis.

V. Method

Methods of Selectivity Estimation in SPARQL
1) Histogram Method
Basically Histogram methods are most popular method to get the results from database and also from RDF. Histogram methods are also used for SPARQL and RDF. In Histogram methods dataset is divided in 'Buckets'. And when query is executed it will fired against the bucket in place of entire dataset. Data are distributed in 'Buckets' in some common fashion. And it is based on partitioning rule.

Every histogram has some common characteristics like.

1) Partition class: The restricted class of histograms considered by the partitioning rule.
2) Partition constraint: The mathematical constraint that uniquely identifies the histogram within its partition class.
3) Sort parameter and source parameter.
4) Approximation of values within a bucket.
5) Approximation of frequencies within a bucket.

2) Index Method
Like its name suggest Indexing/Ordering of data is considered in this method.
This is most common method used in RDBMS also. Aggregated indexes can be also used for ordering in SPARQL Joins. That can be like
An index I over a data graph G is a pair I = (P, O), where P represents a pattern and O represents the set of all occurrences of P in G.

VI. Related Work

1. In 2010 Hai Huang (Faculty of ICT – Swinburne University of Technology) Chengfei Liu (Faculty of ICT – Swinburne University of Technology) This paper focuses on selectivity estimation for SPARQL graph patterns. They first focus on two common SPARQL graph patterns (star and chain patterns) and propose to use Bayesian network and chain histogram for estimating the selectivity of them. Then, for an arbitrary composite SPARQL graph pattern, we maximally combines the results of the star and chain patterns we have precomputed.

For Star Graph Pattern they have used equation like

$$\begin{aligned} sel(Q) &= \Pr(prop_1 = o_1, prop_2 = o_2, \ldots, prop_n = o_n) \cdot \|R| \\ &\approx Pr_\beta(prop_1 = o_1, prop_2 = o_2, \ldots, prop_n = o_n) \cdot |R| \\ &= \prod_{i=1}^{n} \Pr(prop_i = o_i \mid Parents(prop_i) = \vec{o_k}) \cdot |R| \end{aligned}$$

Fig 1

Where Parents denotes the set of immediate predecessors of propi in the Bayesian network; ~ ok denotes the set of values of Parents.
For example

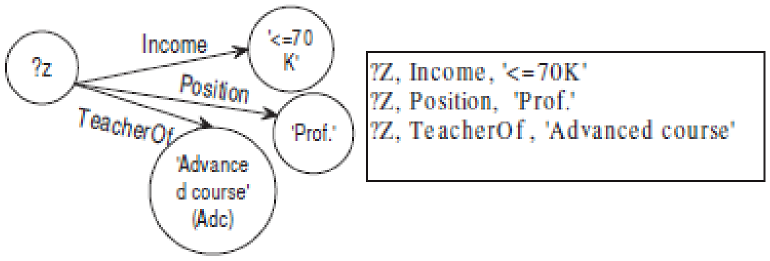
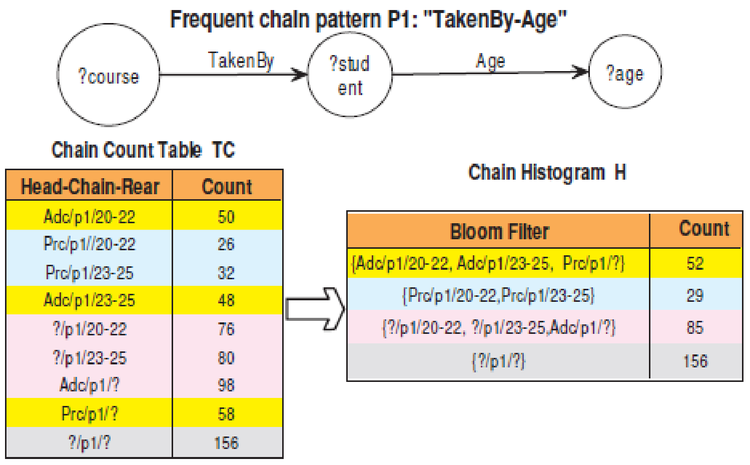


Fig 2

For estimating the selectivity of frequent star patterns, we construct the cluster-property table R for each one.

For Chain Pattern
They construct the chain count table TC for frequent chain patterns, which has two attributes: Head-Chain-Rear and Count.
For example



Chain Count Table TC

Head-Chain-Rear	Count
Adc/p1/20-22	50
Prc/p1//20-22	26
Prc/p1/23-25	32
Adc/p1/23-25	48
?/p1/20-22	76
?/p1/23-25	80
Adc/p1/?	98
Prc/p1/?	58
?/p1/?	156

Chain Histogram H

Bloom Filter	Count
{Adc/p1/20-22, Adc/p1/23-25, Prc/p1/?}	52
{Prc/p1/20-22,Prc/p1/23-25}	29
{?/p1/20-22, ?/p1/23-25,Adc/p1/?}	85
{?/p1/?}	156

Fig 3

Thus, we group the chain patterns in TC into several buckets according to their frequencies. And for each bucket we only need to save the average frequency and its chain pattern members. So given a chain pattern and which bucket it belongs to, we can easily get the frequency of this chain pattern. For efficient processing the membership queries, we use bloom filter, a space-efficient probabilistic data structure often used to test whether an element is a member of a set. Here, we use bloom filter to test whether a chain pattern is a member of a bucket.

Result

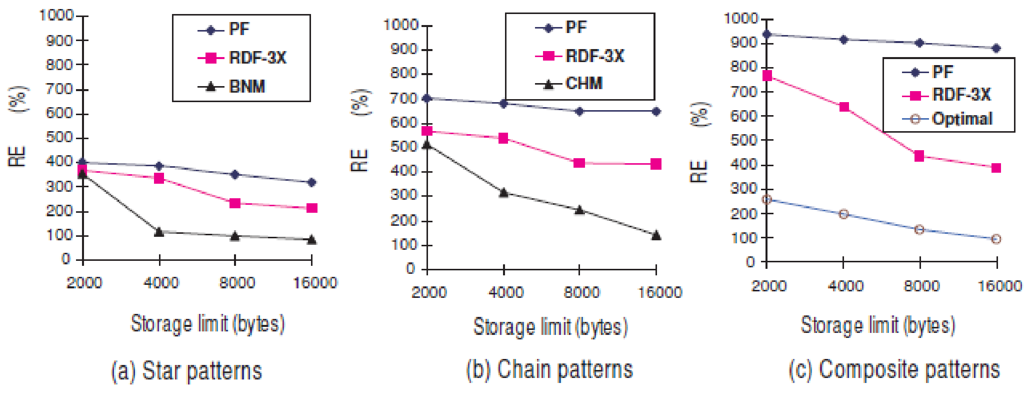


Fig 4

2. In 2008 Markus Stocker (HP Laboratories – Bristol UK)Andy Seaborne (HP Laboratories – Bristol UK) Dave Raynolds (HP Laboratories – Bristol UK) Abraham Bernstein (Department of Informatics University of Zurich) Christoph Keifer (Department of Informatics University of Zurich) approaches this paper, they formalize the problem of Basic Graph Pattern (BGP) optimization for SPARQL queries and main memory graph implementations of RDF data. They define and analyze the characteristics of heuristics for selectivity based static BGP optimization. The heuristics range from simple triple pattern variable counting to more sophisticated selectivity estimation techniques.
Method
They used following formula for calculating the Selectivity Estimation

$$sel(o) = \begin{cases} sel(p, o_c), & \text{if p is bound;} \\ \sum_{p_i \in \mathbb{P}} sel(p_i, o_c), & \text{otherwise.} \end{cases}$$

Fig 5

In order to work on this equation they also calculated the summary statistics like join triple statistics and Simple Triple Pattern statistics of various RDF details.

3. In 2010 Mohmmad Farhan Hussain – University of Texas ,Latifar Khan - University of Texas at Dallas, Murat Kantarcioglu – University of Texas at Dallas ,Bhavani Thuraisingham - University of Texas at Dallas They describe a framework that they built using Hadoop, a popular open source framework for Cloud Computing, to store and retrieve large numbers of RDF triples. We describe a scheme to store RDF data in Hadoop Distributed File System. We present an algorithm to generate the best possible query plan to answer a SPARQL Protocol and RDF Query Language (SPARQL) query based on a cost model. They have presented a framework capable of handling enormous amount of RDF data. Since our framework is based on Hadoop, which is a distributed and highly fault tolerant system, it inherits those two properties automatically. The framework is highly scalable. To increase capacity of our system all that needs to be done is to add new nodes to the Hadoop cluster. They have proposed a schema to store RDF data in plain text files, an algorithm to determine the best processing plan to answer a SPARQL query and a cost model to be used by the algorithm. Our experiments demonstrate that our system is highly scalable. If they add data, the delay introduced to answer a query does not increase as much as the increment in the data size.

4. In 2011 Daisy Zhe Wang - University of California, Berkeley ,Long Wei - University of California, Berkeley ,Yunyao Li - University of California, Berkeley, Frederick Reiss - University of California, Berkeley, Shivakumar Vaithyanathan- University of California, Berkeley. They also develop techniques to decompose a complicated regular expression into subparts to achieve more effective and accurate estimation. They conduct experiments over the Enron email corpus using both real-world and synthetic workloads to compare the accuracy of the selectivity estimation over different classes and variations of synopses. The results show that,

the top-k stratified bloom filter synopsis and the roll-up synopsis is the most accurate in dictionary and regular expression selectivity estimation respectively. They proposed a document synopsis-based approach for selectivity estimation. They developed three classes of document synopsis: n-gram synopsis, bloom filter synopsis and roll-up synopsis. Our experimental results show that these synopsis are compact and enable accurate selectivity estimations. As future work, we intend to look at cost estimation of extraction operators and extend a database query optimizer to use these statistics.

5. In 2012 Jens Dittrich - Saarland University, Jorge Arnulfo QuianeRuiz - Saarland University they teach such techniques. First, they will briefly familiarize the audience with Hadoop MapReduce and motivate its use for big data processing. Then, they will focus on different data management techniques, going from job optimization to physical data organization like data layouts and indexes. Throughout this tutorial, they will highlight the similarities and differences between Hadoop MapReduce and Parallel DBMS. Furthermore, they will point out unresolved research problems and open issues. They conclude by providing a holistic view on how to leverage state-of-the-art approaches (presented in the first three parts) to significantly improve the performance of Hadoop MapReduce jobs. In particular, they will identify open challenges. In addition, they will sketch a vision on how future Hadoop MapReduce platforms may look like. Finally, they will open the floor for questions on dedicated topics presented in this tutorial. Target Audience. The first part of this tutorial covers the basics of Hadoop MapReduce. Therefore this part is interesting for all VLDB attendees who want to learn how Hadoop MapReduce can be used for big data analytics. The second and third part of this tutorial are designed for attendees — researchers as well as practitioners with an interest in performance optimization of Hadoop MapReduce jobs.

CONCLUSION

In my report, the goal of Hadoop MapReduce method using to retrieve the large scale of RDF dataset processed. RDF data process using the MapReduce to extract the data against SPARQL queries. We have focused on Start Pattern Query and Chain Pattern Query for our research work. We are trying to speed up the execution of complex queries agains RDF. Use the String Synopsis which can remove repeated Object literals in Maps. This algorithm removes unnecessary Map Reduce Cycle to generate optimized RDF. Bloom Filter can check the data available in set. Algorithm can be reduced the queries execution time as compare to original RDF.

ACKNOWLEDGEMENT

The authors would like to thank the reviewers for their precious comments and suggestions that contributed to the expansion of this work.

REFERENCES

[1] Selectivity Estimation for SPARQL Graph Pattern Hai Huang Faculty of ICT Swinburne University of Technology,IEEE 2010

[2] SPARQL Query Optimization Using Selectivity Estimation Abraham Bernstein, Markus Stocker, and Christoph Kiefer Department of Informatics, University of Zurich, Switzerland.

[3] 2010 IEEE 3rd International Conference on Cloud Computing Data Intensive Query Processing for Large RDF Graphs Using Cloud Computing Tools Mohammad Farhan Husain University of Texas at Dallas.

[4] Selectivity Estimationz for Extraction Operators over Text Data Daisy Zhe Wang, Long Wei, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan University of California, Berkeley and IBM Almaden Research Center issue 2000.

[5] Efficient Big Data Processing in Hadoop MapReduce Jens Dittrich JorgeArnulfo Quian´eRuiz Information Systems Group Saarland University http://infosys.cs.unisaarland.de

[6] Selectivity Estimationz for Extraction Operators over Text Data Daisy Zhe Wang, Long Wei, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan University of California, Berkeley and IBM Almaden Research Center issue 2000.

[7] Martin Przyjaciel-Zablocki, Alexander Schatzle, Eduard Skaley, “Map-Side Merge Joins for Scalable,” IEEE, 2013.

[8] Shih-Ying Chen, Po-Chun Chen “An Efficient Join Query Processing based on MJR Framework ,” Department of Computer Science and Information Engineering National Taichung University of Science and Technology Taichung, Taiwan 2012 IEEE.

[9] Jiewen Huang, Daniel J. Abadi , Kun Ren ,”Scalable SPARQL Querying of Large RDF,” Yale University, VLDB Endowment, Vol. 4, No. 11 , September 3rd 2011 Seattle Washington.

[10] Prasad Kulkarni ,"Distributed SPARQL query engine using MapReduce " , Master of Science Computer Science School of Informatics University of Edinburgh 2010.

[11] Martin Przyjaciel-Zablocki, Alexander Sch¨atzle ,"RDFPath: Path Query Processing on Large RDF Graphs with MapReduce" ,Springer-Verlag Berlin Heidelberg 2011.

[12] Malini Siva, A. Poobalan, "Semantic Web Standard in Cloud Computing," ISSN: 2231-2307, Volume-1, Issue-ETIC2011,January 2012.

[13] Jose M. Gimenez-Garcıa , Javier D. Fernandez," MapReduce-based Solutions for Scalable SPARQL Querying , " Open Journal of Semantic Web (OJSW)Volume 1,Issue 1 , 2014.

[14] Konstantina Palla , "A Comparative Analysis of Join Algorithms Using the Hadoop Map/Reduce Framework" , Master of Science School of Informatics University of Edinburgh, 2009

[15] 2010 IEEE 3rd International Conference on Cloud Computing Data Intensive Query Processing for Large RDF Graphs Using Cloud Computing Tools Mohammad Farhan Husain University of Texas at Dallas.

[16] Holistic and Compact Selectivity Estimation for Hybrid Queries over RDF Graphs? Andreas Wagnery, Veli Bicerz, Thanh Tranx, and Rudi Studery Karlsruhe Institute of Technology, z IBM Research Centre Dublin issue 2001

[17] Dynamic and Fast Processing of Queries on Large Scale RDF Data Pingpeng Yuan1, Changfeng Xie1, Hai Jin1, Ling Liu2, Guang Yang1 and Xuanhua Shi1 1Services Computing Technology and System Lab., School of Computer Science and Technology issue 2003.

[18] OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation Abraham Bernstein, Christoph Kiefer, Markus Stocker Department of Informatics University of Zurich {bernstein,kiefer,stocker}@ifi.unizh.ch Technical Report No. ifi-2007.03 March 2, 2007

[19] Dynamic and fast processing of queries on large-scale RDF data, Pingpeng Yuan Changfeng Xie · Hai Jin ·Ling Liu · Guang Yang · Xuanhua Shi Received: 6 May 2013 / Revised: 16 August 2013 / Accepted: 27 December 2013 © Springer-Verlag London 2014.

[20] Holistic and Compact Selectivity Estimation for Hybrid Queries over RDF Graphs? Andreas Wagnery, Veli Bicerz, Thanh Tranx, and Rudi Studery, Karlsruhe Institute of Technology, z IBM Research Centre Dublin San Jose State University.