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. 5, May 2016, pg.241 – 248

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 data processing very large scale. It includes analysis of complex data. RDF is very popular Semantic Web data type that store and retrieve large amount of useful data and resources. RDF Resource Description Framework highlights form. A useful representation of RDF data is N-triple in which the subject-predicate-object model is used to represent RDF data. I will develop customized template for RDF file that can be used to access faster RDF data. For RDF data using SPARQL file will use Apache Jena. I can also request Bloom filter for text data in the RDF file. RDF file and the query will be input to the algorithm and after processing new file will be generated using RDF custom template that give faster query results below age RDF. RDF data generation has accelerated to the point that many data sets have to be divided into several teams in order to achieve a reasonable performance when querying data. Although it has made enormous progress in the Semantic Web community for achieving results of high data management on a single node, current solutions that allow data were divided into several teams are highly inefficient. Introducing a scalable data management RDF is up to three orders of magnitude more efficient.

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

I. INTRODUCTION

This paper focuses on estimating the selectivity of SPARQL graph patterns, which is crucial for the optimization of RDF queries. Previous work takes the assumption join uniformity, leading to high inaccurate estimates in cases where the properties are correlated in SPARQL graph patterns. We take into account the dependencies between the properties of SPARQL graph patterns and propose a model more accurate estimate. We focus on the first two patterns of common graphics (SPARQL star chain patterns) and proposes the use of the Bayesian network and histogram chain to estimate the selectivity of them. Then, by a graphic pattern it made arbitrary

SPARQL, we maximum combines the results of the stars and chain patterns that have previously calculated. Experiments show that our method outperforms existing approaches in accuracy. Since the use of RDF to represent data has grown dramatically in recent years, query processing in RDF data becomes an important issue. selective estimation of SPARQL graph patterns is crucial for processing RDF queries. As we know, the RDF data is a set of triples with the form (subject, property, object). This model leads to fine grain SPARQL RDF queries on data with a large number of combinations. As such, accurate estimation of the selectivity of triple patterns together is very important.

II. **Introduction to RDF**

LANGUAGE2 RDF query (SPARQL). RDF is the standard for storing and displaying data and SPARQL is a query language to retrieve data from an RDF store. The power of these Semantic Web technologies can be exploited successfully in the environment of cloud computing to provide the user the ability to store and retrieve data for data-intensive applications efficiently. The synergy between the web and cloud computing semantic fields offers great benefits, such as standards for data representation through the frames. One of these rules is the Resource Description Framework (RDF). With the explosion of technology Semantic Web, RDF large graphs are commonplace. current frameworks do not scale for large RDF graphs. Describe a framework that we have built using Hadoop, a framework for popular open source cloud computing to store and retrieve large amounts of RDF triples. It describes a scheme for storing RDF data in Hadoop Distributed File System. We present an algorithm to generate the best possible query plan answer a query SPARQL Protocol and RDF Query Language (SPARQL) based on a cost model. I use Hadoop MapReduce framework to respond to queries. Our results show that we can store large RDF graphs Hadoop cluster built with inexpensive hardware commodity class. 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 processing and generating large data sets application. Users specify a map function that processes a key / value pair to generate a set of key / intermediate value, and reduce the role that combines 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 parallelized and executed on a large group of machines raw materials automatically. The system runtime takes care of the details of the partition of the input data, scheduling program execution across a set of machines, handling machine failures, and managing communication between the machine required. This allows programmers do not have any experience with parallel and distributed systems facilitate the use of the resources of a large distributed system. Our implementation of MapReduce runs on a large group of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. The programmer easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs work groups running on Google every day. In recent years, authors and many others at Google have implemented hundreds of special purpose calculations that process large amounts of raw data such as crawled documents, records of web applications, etc., to calculate various types of data derived such as inverted indexes, various representations of the graphical web, summaries of the number of pages crawled per host, the set of more frequent consultations on a given day documents, etc. most of these calculations are conceptually simple. However, the input data is usually large and calculations have to be distributed across hundreds or thousands of machines to within a reasonable period of time. The questions of how to parallelize the calculation, distribute the data, and handle errors conspire to obscure the simple original calculation with large amounts of complex code to deal with these problems. Reacting to this complexity, a new abstraction that allows us to express the simple calculations we were trying to do, but hides the dirty details of parallelization, fault tolerance, data distribution and load balancing in a library was designed. Our abstraction is inspired by the map and reduce primitives present in Lisp and many other functional languages. We realized that most of our calculations necessary for the implementation of an operation map for each logical .record. in our entrance in order to calculate a set of key / intermediate value and then applying an operation to reduce all values that share the same key, in order to combine the data properly derivatives. Our use of a functional model with the user specifies the map and reduce operations allows us to parallelize large calculations with ease and use new execution as the main mechanism for fault tolerance. The main 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 in large clusters of PCs raw materials.

IV. **Introduction to Hadoop**

Hadoop is an open source software for reliable, scalable computing, distributed code. Many of the tasks of data analysis on a large scale have been conducted on this platform. Using a cluster with a lot of computers, Hadoop provides a powerful computing capacity to handle well scalability. Hadoop consists of two layers, the data storage layer or the Hadoop Distributed File System (HDFS) and layer data processing or MapReduce framework. HDFS is a block structured system. The files are cut into blocks and stored in the cluster nodes are called DataNode. And there is a NameNode that contains metadata about the size and location of the blocks. MapReduce is a framework for computing Hadoop, which follows the master-slave architecture. Each work is divided into map bumps and reduce jobs, running on the slave nodes. HadoopRDF combines both systems. Hadoop is used as the base architecture. A cluster is arranged by hadoop is considered as the tool of communication and control. All papers are sorted and prepared by the internal rules Hadoop cluster. RDF triple store is placed in the slave nodes in the cluster that provides the basic storage and retrieves needs. To make e_ective methods, various special parts need to be considered when designing the system. As data on public web has expanded rapidly and more data are represented as RDF as its advantages, the problem scalability while dealing with it becomes important. To store and retrieve RDF data stores triples as equals, 4store, have been investigated. 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 code has been widely used for the analysis of large-scale data.

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.

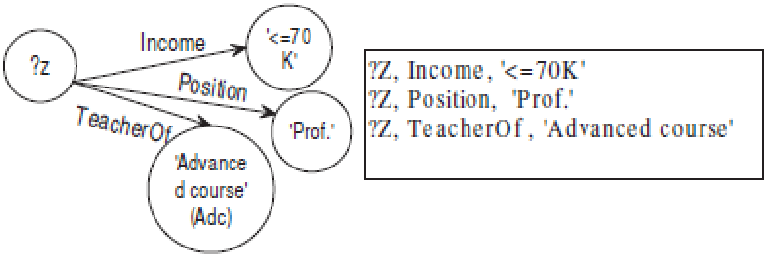
VI. **Related Work**

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

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

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

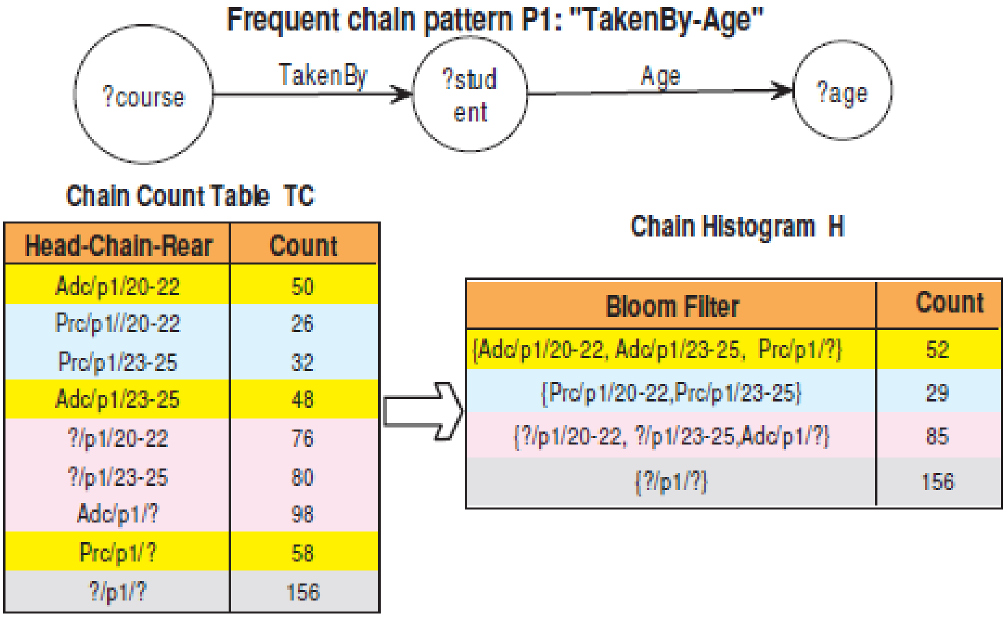


?Z, Income, '<=70K'
?Z, Position, 'Prof.'
?Z, TeacherOf, 'Advanced course'

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

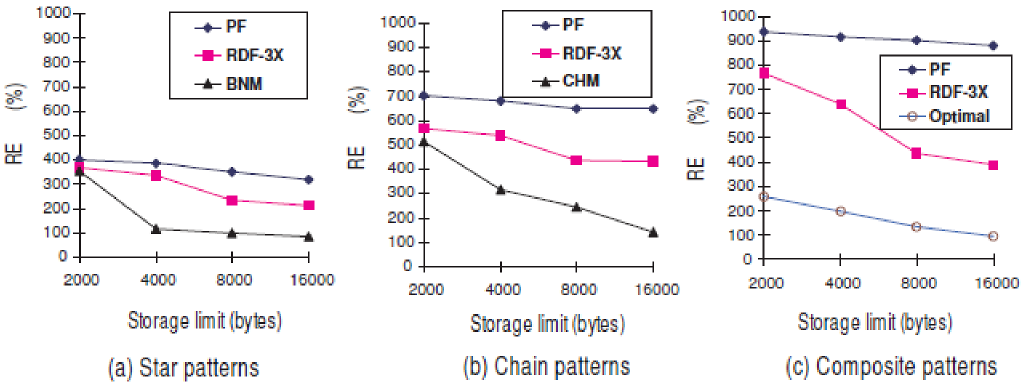
Chain Histogram H

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



(a) Star patterns (b) Chain patterns (c) Composite patterns

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

VII**. Proposed Algorithm**

Input: Query File RDF File
Output: Reduced RDF File and result based on query

1. Read The SPARQL Queries ,Stored in A File
2. Analyse the Query using Jena API
3. Parse the RDF File
4. Convert it to N-Triple (Subjects and Objects and Relation)
5. Apply String Synopsis :Normalize large text data
6. Generate the Map from N-Triple.
7. Index the Map data on Subject or Object Appearance
8. In Map Reducing Remove UN Used and Redundant Subject & Object from Map.

   (Those are not appeared in Join Query Group)
9. New set of N-Triple will be generated
10. Create RDF based on New N-Triple
11. Fire the Queries and get the Estimation

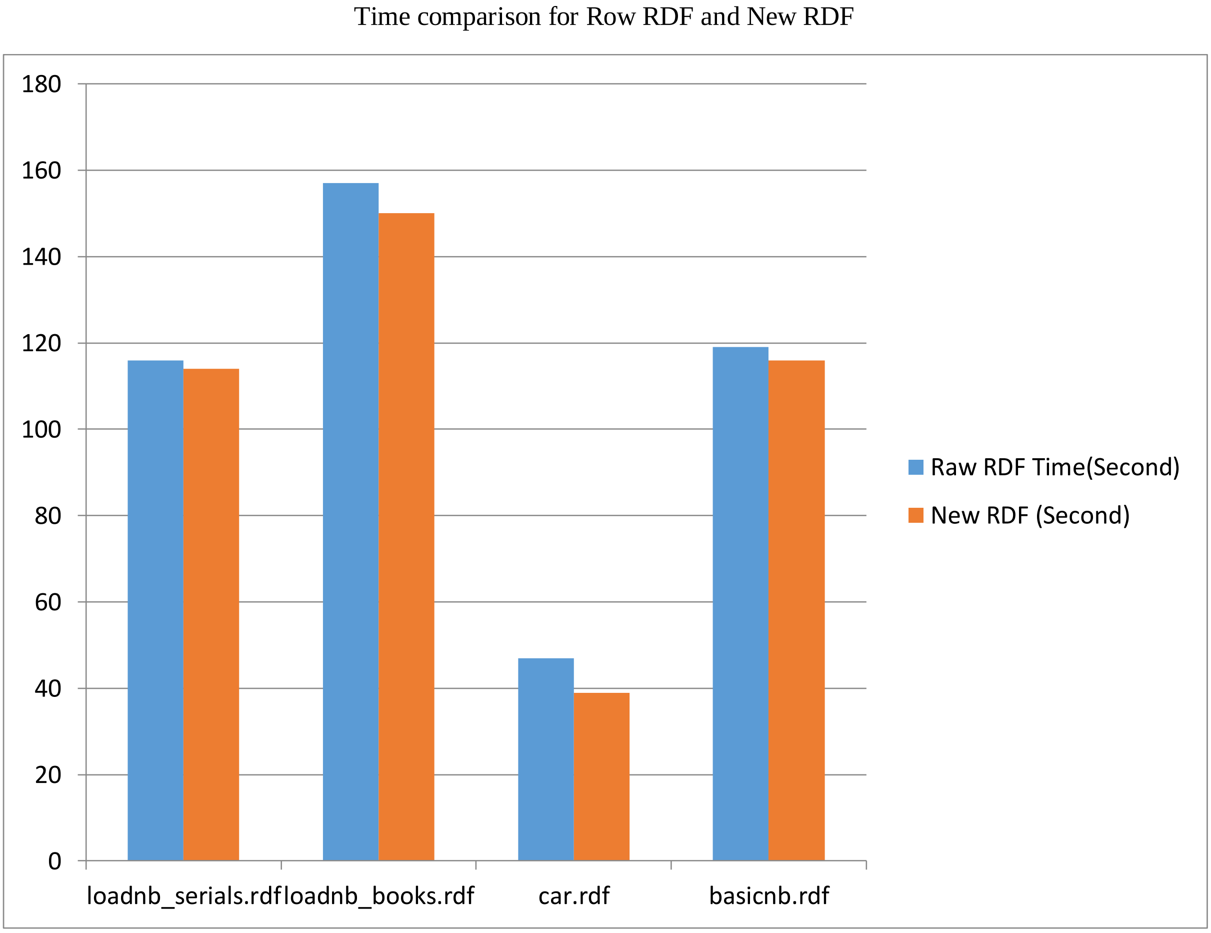


Fig. Comparison graph

Time with various dataset

DataSet	Size	Query size	Raw RDF(Second)	New RDF(second)
loadnb_serials.rdf	19612	50	116	114
loadnb_books.rdf	24520	50	157	150
car.rdf	21	20	47	39
basicnb.rdf	50715	50	119	116

CONCLUSION

In my paper, 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.

FUTURE ENHANCEMENT

As in my current work I have tried to improve Query Execution Time for SPARQL over BigData using Hadoop Map Reducing. We have customized N-Triple for Query Execution. My approach is highlighted by dynamic plan generation and pipelining execution. I process an RDF query in two phases: plan generation phase and execution phase. Two phases are executed iteratively. Plan generation phase identifies blocks of queries and orders them according to the cost estimation. In the second phase, each block is executed using dynamic pipelining, which dynamically and adaptively select the next operator to execute in order to minimize the size of intermediate results. In future we can reduce complexity for the Mapping and Reducing Process.

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.