Distributed Query Processing in Cloud using Canonicalization Approach

SENTHIL K[1] and CHITRA P[2]

[1]Research Scholar, Department of Computer Science and Engineering,
Thiagarajar College of Engineering,
Thiruparangunram, Madurai.
INDIA
[2]Assistant Professor, Department of Computer Science and Engineering,
Thiagarajar College of Engineering,
Thiruparangunram, Madurai.
INDIA

senthilvikramphd2014@hotmail.com[1], pccse@tce.edu[2]

Abstract: - Nowadays, citations and its formats act as a vital role in scientific publication digital libraries (DLs). Due to various reasons, the citation metadata extraction process make it difficult. The citation used in the extraction process is gathered from web and there is no standard style to the citations. Data gathered from the web is difficult to process due to the erroneous nature of the data. In order to overcome these problems, this paper proposed a distributed query processing based on canonicalization approach. Here, NoSQL database is used for distributed query support and to enhance sustained data throughput. The query is retrieved from the user and then verified with the metadata created. In case of match occurs, the index prediction is performed and the results are evaluated. Semantic query matching is used to match the given query with the database. The Citation Parser (CP) method is used to convert the query string into simple symbols to minimize the query processing time and to enhance the system performance. The experimental results shows that the proposed method results higher accuracy with lesser query processing time, response time and lesser computation cost.

Key-words: - Big Data; Citation Parser; Metadata; NoSQL Database; Semantic Query Matching.

1 Introduction

Big data is also defined as a data analysis methodology enabled by a new generation of architecture and technologies that support high-velocity data storage, capture, and analysis. Cloud computing support an environment for small to medium sized businesses in order to implement big data technology. All small to medium sized businesses use cloud computing for big data technology [1] implementation in order to

- Minimize processing costs
- Minimize hardware costs
- Test the big data value before committing important company resources

Speed is the main issue during the query processing in big data. The process consumes more time as it cannot traverse all the related data in the whole database in a short time. Index is an optimal choice in this case. At present, indices in big data are mainly aiming at simple type of data, while big data is becoming complicated. A lot of companies minimize their IT cost by using online big data application [2]. The Hadoop framework [3] is utilized by major players including Yahoo, Google and IBM, [4] largely for applications such as advertising and search engines. Hadoop provides fast and reliable analysis of both unstructured and structured data. All the modules in Hadoop are designed with a basic assumption that there is a normal occurrence of hardware failures that are handled in software by the framework. It performs the processing of large amount of information by connecting commodity computers together to work in parallel. In this paper, the Canonicalization algorithm is introduced with the semantic query matching to efficiently performing the query retrieving process.

The remaining part of the paper is organized as follows: Section II involves the works related to the query processing techniques in the cloud environment. Section III involves the description of the proposed approach including citation parser method and canonicalization algorithm. Section IV involves performance analysis of the proposed approach. Section V includes conclusion and future enhancement.

2 Related Work

The term NoSQL database [16] is chosen for a loosely specified class of non-relational data stores. These databases mostly do not utilize SQL as their query language. A part of NoSQL databases normally restrict the overhead occurring in fully functional RDBMS. The data are organized into tables on a logical level and are usually accessed through primary key. The operations join and order by are not supported by the NoSQL databases. Simple NoSQL databases known as key-value stores consist of a set of couples (key, value). An integrated platform for big data analysis [5] considers the better parameterization of algorithms, effective usage of system resources, improved scalability of the whole platform, and an enhanced usability during the end-to-end data analysis process. Advanced visualization and data exploration approaches are required to deal with the massive amount of big data. A novel scheme known as VegaGiStore [6] is used to provide effective spatial query processing over big spatial data. The spatial data are organized in terms of geographic proximity using the geography-aware approach. The Map Reduce parallel processing framework and Map Reduce optimization strategies [7] are introduced to deal with the key issues of big data processing, including cloud architecture, cloud computing platform, cloud database and data storage scheme.

Three significant frameworks such as AROM, Haloop, Sailfish [8] are used for processing big data. AROM depends on functional programming and data flow graphs (DFGs). Sailfish is used for large scale data processing. Haloop is a variant of Hadoop. The elastic solutions [9] are proposed for query optimization and execution over big data integration. Hadoop cloud computing framework [10] is used to generate a user application that allows the scientific data processing on the clouds. Hadoop's MapReduce is used to handle scientific data processing problems with inconsistent input formats. Hadoop MapReduce [11] is a popular data processing engine for big data. It consist of two user-defined functions: *map* and *reduce*. One of the main benefits of Hadoop MapReduce is that it allows non-expert users to easily run detailed tasks over big data. The NoSQL databases depending on the key-value model [12] such as HBase and Bigtable are scalable in large scale applications. Providing the ACID (atomicity, consistency, isolation, and durability) for the NoSQL data storage system is a significant problem. An enhanced History record-based Service optimization method (Hiresome-II) [13] is proposed for privacy-aware cross-cloud service composition for big data processing. As only some representative history records are recruited, the approach minimizes the time complexity. The method also protects the privacy in big data.

The parallel and distributed processing capability of Hadoop MapReduce [14] is used to handle heterogeneous query execution on large datasets. Hadoop environment set up includes a great number of parameters that are essential to accomplish best performance. A scalable semantic web framework [15] generates and executes optimal query plans. The approach handles complex queries with optional blocks and queries with basic graph patterns. Hadoop's MapReduce framework is used to process the query plan. An effective and cost-efficient multilevel caching scheme (MERCURY) [16] explore and accomplish data similarity and support effective data placement. A novel MC-LSH (multicore-enabled locality-sensitive hashing) scheme is used to correctly capture the differentiated similarity across the data. Domain ontology based semantic search [17] performs effective information retrieval through automatic query expansion. The query expansion is performed using the domain ontology and meaningful concepts extracted from the user's query. The ranking of retrieved documents is performed according to their relevance to the user's query. The two main performance metrics in MapReduce are [18] job execution time and cluster throughput. They are impacted by straggler machines. Speculative execution is a common method for dealing with the straggler problem.

A new strategy, MCP (Maximum Cost Performance) is developed to enhance the efficiency of speculative execution. Skyline [19] is a significant operation that is used to retrieve set of interesting points from a potentially large data space. A novel skyline algorithm SSPL on big data (skyline with sorted positional index lists) requires low space overhead to minimize I/O cost significantly. A new

Data-gRouping-Aware (DRAW) data placement scheme [20] scrutinizes data access from system log files. The optimal data groupings are extracted and the data layouts are then re-organized to accomplish the maximum parallelism per group. Hadoop, an open-source implementation of MapReduce is widely utilized for short jobs requiring minimum response time.

Nephele [21] is the data processing framework that exploits the dynamic resource allocation provided by IaaS clouds for execution and task scheduling. The specific tasks of a processing job can be assigned to distinct types of virtual machines that are automatically terminated and instantiated during the job execution. The evaluations of MapReduce-inspired processing jobs are performed on an IaaS cloud system and the results are then compared with the popular data processing framework Hadoop. A framework built using Hadoop [22] is used for storing and retrieving large numbers of RDF (Resource Description Framework) triples. An algorithm is proposed to create the best possible query plan in order to answer SPARQL protocol and RDF query language depending on cost model. The architectural features of cloud computing are explored [23] and classified according to the requirements of end-user, cloud providers and enterprises that use the cloud as a platform. The key issues of large scale graph processing on the cloud computing environment [24] are surveyed.

3 Proposed Method

In this section, the distributed query optimization and indexing scheme is introduced across heterogeneous cloud environment. The main objective of this work is to improve the query processing in cloud environment. The proposed scheme uses the NoSQL structure for storing, processing and managing huge amounts of data. The structure of the proposed scheme is depicted in fig.1. The Big data is distributed across the Hadoop platform in order to reduce the query processing time and also to reduce the processing cost. Initially the large amount of data is preprocessed to remove the irrelevant and noisy information present on the database. The metadata is created for managing the information about the data. Metadata is the data which defines a corresponding item of content and where it is positioned.

It captures the essential information about the enterprise environment data, improve integration efficiency, and business logic to accelerate enhancement. Parsing query is the process of analyzing the string of symbols.

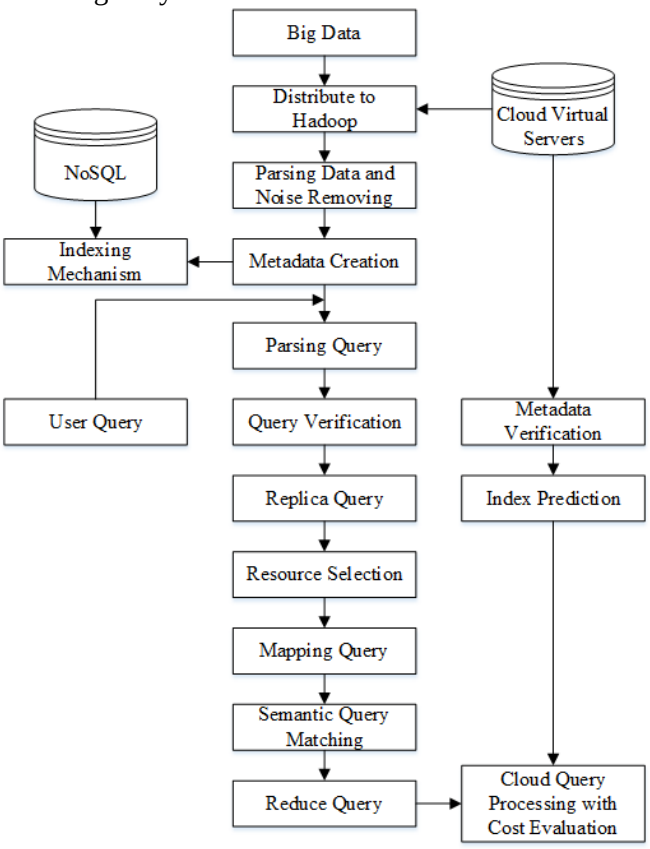

Fig.1 Overall flow of the proposed query processing approach

The given query is verified with the metadata information. Finally, the results are retrieved depending on the semantic similarity between the query symbols and the encoding table. The following section describes the proposed citation parser and canonicalization algorithm with the corresponding step by step procedures.

3.1 Citation Parser (CP)

In the proposed Citation Parser (CP) method, the citation parsing problem is solved based on rephrasing a given citation string into a canonical form. It consists of symbols corresponding to its delimiters and fields. Some punctuation marks used as delimiters may be added inside a certain field. Sometimes it is tough to identify the type of a field by its content and the order of the fields diverges among diverse citation formats. Here, the textual details of the corresponding citation

are removed as much as possible. It stores citation strings with the particular metadata to the template database. A query citation string can be processed to its canonical string and coordinated against canonical string saved in the template database. CP can be categorized into two main elements: 1. Template database construction and 2. Query processing.

In template database, the structural details such as *index form* and *style form* for each citation string is stored. *Index form* is a canonical symbolic representation of a citation string, that is used to align the given query citation string. *Style form* is a symbolic representation of a citation string, from that each field of metadata and each punctuation mark is denoted by a single symbol.

3.1.1 Canonicalization Algorithm

The setup of a citation string is a semi structured. Although it is hard to find the regular rules to analyze the structure accurately, some common features in citation strings can be easily recognized. The canonical string is used to filter out the irrelevant information and noise, also to emphasize the structural features of the citation string. The canonicalization algorithm is introduced to extract the details in a systematic way. It can be classified into the following three components.

1. Encoding table
2. Reserved words
3. Blocking patterns

Category	Symbol	Representation field
Field token	D	Date
	M	Month
	Y	Year
	R	Hour
	N	Minutes
	S	Seconds
	Z	Time zone
	C	HTTP reply code
Delimiters	T	.
	P	-/
	A	[]

Table 1 Encoding Table

Encoding Table

Encoding table assigns every meaningful semantic unit to simplify the difficulty of data format. Encoding table is used to map the tokens in citation strings to symbols in NASA database sequence [25]. The symbols are used to encapsulate the semantic unit including metadata elements and punctuation symbols. Table 1 shows the encoding table with its symbol and the representation field.

Reserved Words

During the encoding process, the knowledge about metadata is unknown. The reserved word is used to distinguish the token about its fields. A term that appears in a particular field is referred as reserved words.

Blocking patterns

The reserved words and blocking patterns are used to arrest the contextual features and inside field structure. After defining the reserved words, the dependency between reserved words and punctuations is determined for each citation field. The local structures can be identified as some precise patterns based on encoding the citation in *base form*. Depending on the patterns, it can be grouped as tokens into a block unit and it is represented by a symbol. It keeps the integrity of the semi structured information, called as blocking pattern.

3.2 Algorithm for Updating Cached Data

Cached data updating algorithm is suggested to calculate the value degree V_i of a cache tuple i:

$$V_i = Q_i \times D_i / (Dc_i - Dl_i) \quad (1)$$

Where Q_i denotes the frequency of tuple i being accessed, D_i is the delay time of fetching tuple i, Dc_i is the current time and Dl_i denotes the last access time. It is observed that V_i increases as the value of Q_i and D_i increases and decreases the interval of fetching information $(Dc_i - Dl_i)$ increases. The updating algorithm is based on the value degree V_i.

Algorithm 1: Updating Cached tuples

Input: Query symbol S
Output: Query Results R
Update Query Cache(S)
For Next [cache] is not empty //Search All queries in Cache
If Key[S] == Key [cache (i)] //Search results match
F [cache (i)] <-- F [cache (i)] + 1 //Access Frequency add 1
P [cache (i)] <-- Processing Time //Update processing time

Return //Search Finish
End

3.3 Semantic query matching

Semantic similarity is the measure of relatedness between any two notions. Hence the relation among the terms can be studied. Each concept is connected to each other based on ontology hierarchy. Semantic query development is performed with the concepts matched the query keywords.

4 Performance Analysis

In this section, the performance of the proposed method is evaluated and compared with the existing methods such as FLUX-CiM (Flexible Unsupervised Extraction of Citation Metadata) [26], Infomap (knowledge-based approach to citation extraction) [27], RQ and RQ-Par [28]. The NASA dataset is taken for performance evaluation [25]. It contains two month worth of all HTTP requests to the NASA Kennedy Space center WWW server in Florida. The logs are an ASCII file with one line per request with the set of following columns:

1. *Host* making the request.
2. *Timestamp* in the format "DAY MON DD:MM:SS YYYY"
3. *Request* given in quotes
4. *HTTP* reply code
5. *Bytes in the reply*

The proposed method with the existing methods are examined based on metadata creation accuracy, accuracy achieved for different number of queries, query processing time, response time and computation cost.

4.1 Metadata creation

Metadata summarizes the basic information about data that make identifying with particular instances of data simpler. The higher accuracy value of the formulated metadata leads to improved query retrieval performance. The accuracy of the metadata retrieval of the proposed canonicalization algorithm is compared with FLUX-CiM and Infomap.

The result is shown in fig. 2. The data retrieval accuracy is represented in terms of percentage. The analysis shows that the proposed canonicalization approach achieves higher accuracy in metadata creation than FLUX-CiM and Infomap. It leads to improve the query processing efficiency of the system.

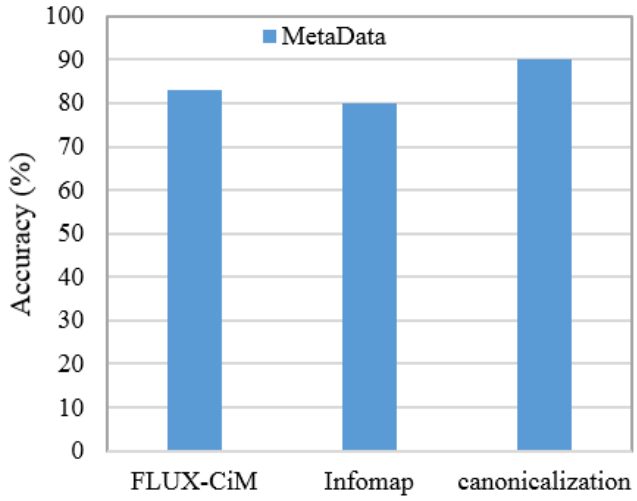

Fig. 2 Comparison of accuracy in metadata creation

4.2 Accuracy

As the proposed method results high accuracy metadata for query retrieval, the retrieval accuracy is also higher. Fig. 3 depicts the accuracy achieved for different number of queries.

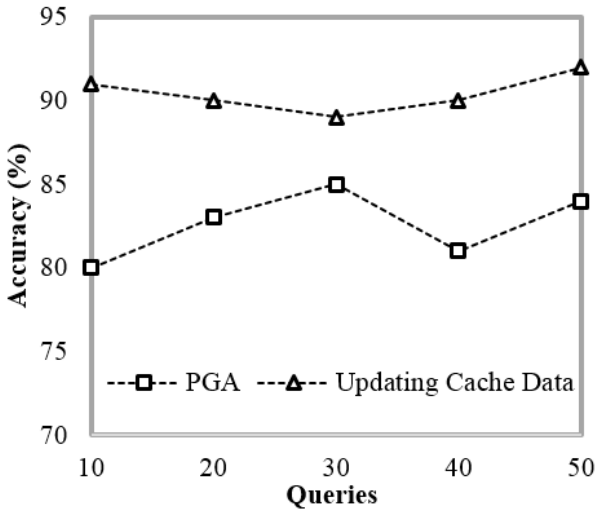

Fig. 3 Accuracy achieved for various number of queries

The proposed method searches the query with the help of encoding table symbols. Also, the semantic meaning is analyzed with the metadata, which helps to retrieve the accurate results. The results show that the proposed approach achieves higher accuracy for various number of queries when compared with PGA

(Polynomial Greedy Algorithm) [29]. The proposed approach accomplishes optimized execution time.

4.3 Time Analysis

In order to show the time efficiency of the proposed method, the query processing time and response time are discussed. The query processing time is the amount of time taken to process the query for searching the citation. The query processing time is updated based on the updating cached tuples algorithm. The lesser query processing time will increases the system performance and it helps to reduce the response time. Table 2 shows the types of query execution plans used for evaluation.

Techniques	Type of execution plan
RQ	Processes all routing operators using P-Grids range query processing
RQ-par	Same as RQ, but query plan is processed using inter-operator parallelism
CP	It follows canonicalization procedure with semantic query matching

Table 2 Types of query execution plans

The proposed method uses the citation parser with the help of canonicalization algorithm. It uses the encoding table to store the symbols for the notations with the corresponding category. Hence, the proposed method easily process the query with the help of symbols with corresponding description. Fig.4 shows the query processing time for CP with the existing methods.

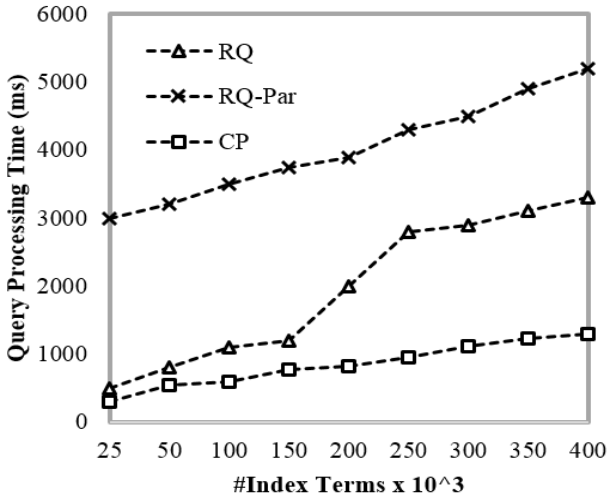

Fig.4 Query Processing Time against the index terms for proposed CP with the existing RQ and RQ-Par

It proves that the proposed method results lesser query processing time than the existing methods. Also, the number of nodes is varied and the query processing time is recorder for the proposed and existing methods. The proposed method results lesser time than the existing RQ and RQ-Par which is shown in fig.5.

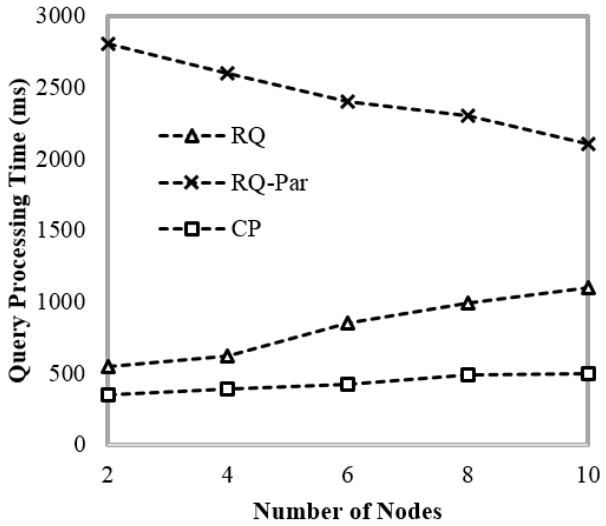

Fig.5 Query Processing Time against number of nodes for proposed CP with the existing RQ and RQ-Par

The response time is the total amount of time the system takes to respond to the given request. The proposed method takes lesser time to process the given query, it leads to reduce the response time.

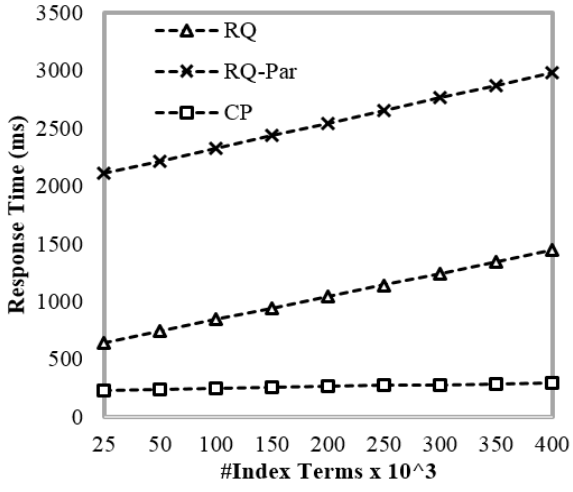

Fig.6 Response Time for the proposed CP with the existing RQ and RQ-Par

Fig.6 shows the response time analysis for the proposed CP with the existing methods. The proposed method takes lesser response time than the existing method.

4.4 Computation cost

Computation cost is analyzed and recorded for Total Cost of Ownership (TCO) and Query Processing Cost (QPC).

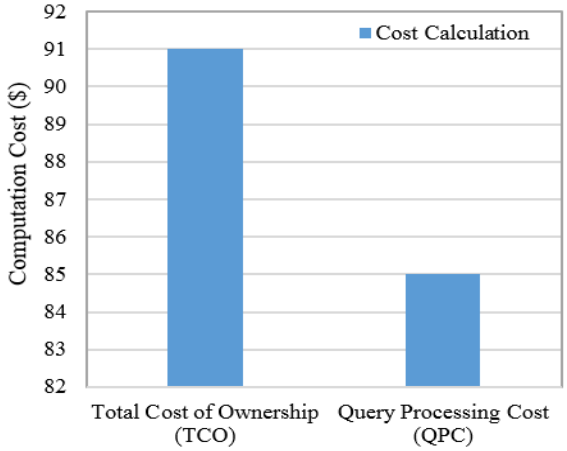

Fig. 7 Comparison of computational cost

Symbols	Description
L	Workload
W	Set of materialized views
t_{iW}	Processing time of query
$c(IE)$	Count of identical instances
nb_{IE}	Constant number of identical instances
$T_{processingL}$	Total processing time

Table 3 Notation used for cost analysis

Table 3 presents the list of notation used for cost analysis. Query processing time is referred based on two parameters namely query workload L and set of materialized views W. Let t_{iW} be the processing time of query L_i when exploiting the set of W. $c(IE)$ is the count of identical instances IE. nb_{IE} is the constant number of identical instances. The total processing time can be defined as:

$$T_{processingL} = \sum_{i=1}^{k} t_{iW} \quad (2)$$

The cost can be computed based on the following equation:

$$Cost_{processingL} = T_{processingL} \times c(IE) \times nb_{IE} \quad (3)$$

Based on the equations, the computation cost are analyzed. The computational cost is reduced for the proposed method. Because, the system uses the encoding table which uses the symbols for the query retrieval. Hence, it takes lesser time to retrieve the result. It automatically reduces the cost to compute the retrieval process. Fig. 7 shows the comparison of computation cost involved in TCO and QPC. The result shows that the computation cost involved in QPC is very much less than when compared with TCO.

5 Conclusion and Future Work

In this paper, an effective query processing approach of big data in a cloud computing environment is proposed. The proposed approach uses Hadoop platform and NoSQL database. Initially, the data is preprocessed to remove noise and then metadata is created. The query is retrieved from the user and then verified with the metadata. Index prediction is then performed and the results are evaluated. Semantic query matching is used to enhance the search accuracy. The proposed method is scalable and results improved query retrieval with the help of metadata indexing and semantic query matching. The experimental results showed that the proposed Citation Parser (CP) method and canonicalization algorithm achieves minimum query processing time, response time and higher accuracy in metadata creation with lesser computation cost than the existing algorithms.

As a future work, the metadata and cache services of the cloud computing can be enhanced. Also, privacy and security greatly affect the big data storage and processing as there is extensive use of third-party services and infrastructures. Unlike conventional security approaches, security in big data is in the form of how to process data mining without exposing sensitive data of users. The secure query processing of the big data deployed in cloud servers can also be improved using the recent and advanced security algorithms.

References

[1] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big data processing in cloud computing environments," in *Proceedings of the International Symposium on*

Parallel Architectures, Algorithms and Networks, I-SPAN, 2012.

[2] T. H. Davenport and J. Dyché, "Big data in big companies," *International Institue for Analytics*, 2013.

[3] Y. Bao, L. Ren, L. Zhang, X. Zhang, and Y. Luo, "Massive sensor data management framework in cloud manufacturing based on Hadoop," in *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, 2012, pp. 397-401.

[4] K. N. Kakade and T. Chavan, "Improving Efficiency of GEO-Distributed Data Sets using Pact," *International Journal of Current Engineering and Technology* vol. 4, pp. 1284-1287, 2014.

[5] M. Bohlouli, F. Schulz, L. Angelis, D. Pahor, I. Brandic, D. Atlan, *et al.*, "Towards an integrated platform for big data analysis," in *Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives*, ed: Springer, 2013, pp. 47-56.

[6] Y. Zhong, J. Han, T. Zhang, Z. Li, J. Fang, and G. Chen, "Towards parallel spatial query processing for big spatial data," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, 2012, pp. 2085-2094.

[7] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big data processing in cloud computing environments," in *Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on*, 2012, pp. 17-23.

[8] D. S. B. Sriramoju, "A Review on Processing Big Data," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, pp. 2672-2685, 2014.

[9] H. Kllapi, D. Bilidas, I. Horrocks, Y. Ioannidis, E. Jimenez-Ruiz, E. Kharlamov, *et al.*, "Distributed query processing on the cloud: the optique point of view (short paper)," in *OWL Experiences and Directions Workshop (OWLED)*, 2013.

[10] C. Zhang, H. De Sterck, A. Aboulnaga, H. Djambazian, and R. Sladek, "Case study of scientific data processing on a cloud using hadoop," in *High performance computing systems and applications*, 2010, pp. 400-415.

[11] J. Dittrich and J.-A. Quiané-Ruiz, "Efficient big data processing in Hadoop MapReduce," *Proceedings of the VLDB Endowment*, vol. 5, pp. 2014-2015, 2012.

[12] V. Padhye and A. Tripathi, "Scalable Transaction Management with Snapshot Isolation for NoSQL Data Storage Systems," 2013.

[13] W. Dou, X. Zhang, J. Liu, and J. Chen, "HireSome-II: Towards Privacy-Aware Cross-Cloud Service Composition for Big Data Applications," 2013.

[14] S. Sathya and M. V. Jose, "Application of Hadoop MapReduce technique to Virtual Database system design," in *2011 International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT)*, , 2011, pp. 892-896.

[15] M. Husain, J. McGlothlin, L. Khan, and B. Thuraisingham, "Scalable complex query processing over large semantic web data using cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 187-194.

[16] Y. Hua, X. Liu, and D. Feng, "Data Similarity-aware Computation Infrastructure for the Cloud," *IEEE Transactions on Computers*, p. 1, 2013.

[17] R. Chauhan, R. Goudar, R. Sharma, and A. Chauhan, "Domain ontology based semantic search for efficient information retrieval through automatic query expansion," in *Intelligent Systems and Signal Processing (ISSP), 2013 International Conference on*, 2013, pp. 397-402.

[18] Q. Chen, C. Liu, and Z. Xiao, "Improving mapreduce performance using smart speculative execution strategy," 2013.

[19] X. Han, J. Li, D. Yang, and J. Wang, "Efficient skyline computation on big data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, pp. 2521-2535, 2013.

[20] P. Shang, Q. Xiao, and J. Wang, "DRAW: A new Data-gRouping-AWare data placement scheme for data intensive applications with interest locality," in *APMRC, 2012 Digest*, 2012, pp. 1-8.

[21] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, pp. 985-997, 2011.

[22] M. Husain, L. Khan, M. Kantarcioglu, and B. Thuraisingham, "Data intensive query processing for large RDF graphs using cloud computing tools," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 1-10.

[23] B. P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural requirements for cloud computing systems: an enterprise cloud approach," *Journal of Grid Computing*, vol. 9, pp. 3-26, 2011.

[24] Y. Zheng, A. A. Hunter III, J. Wu, H. Wang, J. Gao, M. G. Maguire, *et al.*, "Landmark matching based automatic retinal image registration with linear programming and self-similarities," in *Information Processing in Medical Imaging*, 2011, pp. 674-685.

[25] Available: http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[26] E. Cortez, A. S. da Silva, M. A. Gonçalves, F. Mesquita, and E. S. de Moura, "FLUX-CIM: flexible unsupervised extraction of citation metadata," in *Proceedings of the 7th ACM/IEEE-CS*

joint conference on Digital libraries, 2007, pp. 215-224.

[27] M.-Y. Day, T.-H. Tsai, C.-L. Sung, C.-W. Lee, S.-H. Wu, C.-S. Ong, *et al.*, "A knowledge-based approach to citation extraction," in *Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference on.*, 2005, pp. 50-55.

[28] M. Karnstedt, K.-U. Sattler, and M. Hauswirth, "Scalable distributed indexing and query processing over Linked Data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 10, pp. 3-32, 2012.

[29] T. V. Kumar and A. Ghoshal, "Greedy selection of materialized views," *Int. J. of Computer and Communication Technology*, vol. 1, p. 156, 2009.