COPQ: Continuous Optimization of the Parallel Queries of the Distributed Triple Stores

[1]K. SHAILAJA, Department of CSE, Methodist College of Engineering and Technology, Hyderabad, Telangana State, India.

[1]Email: shailajamtech2006@yahoo.co.in.

[2]DR. P.V. KUMAR, UCE, OU, HYDERABAD, Telangana State.

[2]Email: pvkumar58@gmail.com.

[3]DR S. DURGA BHAVANI, School of IT, JNTUH Hyderabad, Telangana State, India.

[3]Email: sdurga.bhavani@gmail.com.

Abstract— This manuscript tends to portray a novel incremental approach to define the optimal query formats to perform search operations on Distributed RDF. Most of the benchmark models targeting to optimize the query formats those intended to execute in parallel that are received from multiple users at an event of time. However, the queries intended to perform on distributed RDF are continuous, hence it can be defined as query stream. In regard to this argument, the proposal defines the history log that buffers the statistics observed in query formats processed in past events. The model portrayed in this manuscript is using the statistics related to distributed triple stores. The statistics of the triple stores are defined as history log, which termed as Query Index Map (QIM) that updates continuously from the outcomes of the queries that already executed. The performance analysis from the experimental results obtained for the proposal and other benchmark methods concluding the performance advantage of the proposed model that compared to the other benchmark models.

Keywords— Query Index Map, Lusail, RDF, Hi-BISCuS, parallel execution, query optimization.

I. INTRODUCTION

The work [1] presents that data method is designed which is called “RDF (Resource Description Framework)” to details the web resources metadata, forming the reports regarding the resources in form of (s (subject), p (predicate), o (object)) automatically states that o & s are associated by the ‘p’. Gathering of RDF reports depicts the resources to be tagged, as directed multigraph. The work [2] presents that SPARQL is W3C candidate commendation query language for the RDF which is utilized broadly in “semantic-web-communities” acting as a significant role in both the field of application and research. More efforts are taken place in constructing the query system of RDF, targeting to develop the efficiency of time for the SPARQL queries over huge scale of RDF information. The work [3] presents that ‘Jena’ is the familiar “single-machine RDF” influences and query the engine assisted through “Apache Foundation”. When the volume is getting maximum and maximum then the people improved various disseminated applications of the RDF data hoard and the query engine of SPARQL like TriAD [4]. The work [5] presents that S2X is other kind of query engine of RDF constructed on uppermost of the GraphX [6] that os novel element in the spark for the “graphs & graph-parallel computation”, elongating the spark-RDF through introducing the “graph-abstraction-layer”.

The extensive utilization of semantic web RDF concerned several works of research on optimizing the RDF query. On the hand there might be several outcomes regarding entailment rules of SPARQL to make the query easier or attain greater result set on the basis of RDF data. The work [7] presents that “Apache Jena” offers “Ontology API” to cause algebraic possessions of ‘p’ assisted by the OWL2. The work [8] presents that SPARQL-S suggested other entailment-engine on the basis of OWL method. The work [9] presents that the information of RDFS T and A-boxes in the SPARQL are utilized for reasoning the queries before assessment.

The effectiveness of the query assessment is the concentration of the researchers in the community of RDF. The work [10] presents that the views of triple-patterns are utilized as “linked data fragments” to have replying the query at the low cost for the “SPARQL conjunctive queries. The work [11] presents that some of the applications are concentrating on in what way we can optimize indexing of the RDF data on the disseminated storage platform for minimizing the interaction costs among the nodes in cluster of large-scale RDF queries assessment. And also, some of the optimization techniques in the traditional association database have introduced the domain of SPARQL [12], [13].

Corresponding Author: K.SHAILAJA, ***Email Id:*** *shailajamtech2006@yahoo.co.in*
Article History: Received: September 04, 2018, Revised: November 25, 2018, Accepted: December 28, 2018

In this article we reviewed the classical issue of optimizing the various queries given in parallel in domain of SPARQL/RDF by diminishing the search space and assessment time and available cost of various queries given in parallel, which are possessing common triple patterns. The scope of the research paper is establishing optimal query chains from the multiple queries buffered in a stipulated time, which are targeting to search distributed RDF stores. In this regard the proposal portrayed a query Index Map that updates continuously with statistics obtained from queries executed earlier.

II. RELATED WORK

The primary objective of the distributed or decentralized RDF systems [14-19] denote the data projection across the multiple data sources fall in same cluster, which is either in the form of replication or of the partitioned data tuples [20], [21]. However, each of these clusters referred as single end point in regard to RDF querying process. These systems are intended to minimize the processing time taken by the query execution on single end point. The other dimension of the RDF systems that referred as federated RDF systems allows accessing the data through SPARQL queries having independent or remote endpoints, which is since the federated RDF systems are not having control over the data.

Federated SPARQL methods such as Hi-BISCuS [22], ANAPSID [23], SPLENDID [24] are on basis of gathered statistics and information regarding the data held at every endpoint. Cost of addition of novel endpoint is comparative to the data size. The other methods like Lusail [25], [26], and FedX [27] uses the SPARQL-ASK queries in finding the related endpoint, and the cache outcomes of queries are used for future. Hence the cost of startup and addition of novel endpoint is minor. The Federated SPARQL methods generally separate query into the exclusive-clusters of the triple –patterns; where every cluster has a result at only single endpoint. The works [28-31] have many efforts to concentrate on the "web-based data integration" on the "heterogeneous information sources" [32].

Generally, wrapper is allowed at every source of data to convert among the data models and assisted languages. Furthermore, methods like [33], [34], [35] are the peer-peer methods which connect the heterogeneous network of data-sources. Unlike standard methods, Lusail decays on basis of query for checking the instances of data thus shifting many of the computation intermediate outcomes towards endpoints. Furthermore, "sub-query ordering" procedure in the lusail is conducted by connecting the outcomes of sub-queries by join-operations; and the cost of communication for executing every sub-query to find the finest ordering that manages among the cost of communication and degree of the parallelism. The work [26] however not planned to search space optimization and available cost optimization. In concern to this dispute, the suggestion of this paper is modifying the features of Lusail, further the suggestion optimizes queries constructed by several object and subject pairs connected with similar predicate, and reconstruct the referred triplets in the specified query so that triplet provides less number of outcomes.

III. METHODS AND MATERIALS

This section describes the formulation of the proposal to achieve the optimization of the queries those intended to execute in parallel on distributed triple stores. The queries buffered from a stream for a given stipulated time will be processed to achieve optimal access cost and minimal search space.

A. Query Index Map

In order to optimize the multiple queries buffered to perform parallel execution, the proposal of this manuscript is defining a data structure called Query Index Map (QIM). This data structure logs the elements count that explored under a query executed for a given triplet, in this regard, the QIM uses each RDF reference rid as top level key of the map, which addresses a map sm as value. Each of the map sm referred by the RDF reference rid as key contains key, value pairs, where the key is a subject or object of a triplet that addresses again a map uem as value. Each of these maps uem addresses unique entries of the subject or object as key and their occurrence count as value. On other dimension, the Query Index Map maintains other map rm, which denotes RDF reference as key that refers a map $spom$, which further refers the predicate as key and the value is another map som having each subject as key and the value is the hash-set that representing the respective objects, which linked by the predicate that used as key in the map $spom$ to refer this map som. The syntax to explore the required information about a triplet from QIM is as follow

$QIM[rid] \Rightarrow sm[subject/object] \Rightarrow uem[unique\ entry\ of\ the\ uem] \Rightarrow occurrence\ count$ //delivers the occurrence count of each unique entry for the given element of subject/object

$$rm[rid] \Rightarrow spom[predicate] \Rightarrow som[subject] \Rightarrow \{objects\}$$

// extracting the all objects linked to the given subject under given predicate.

This Query Index Map (QIM) will be available as local resource to the process that optimize the queries, which are buffered from the query stream in a given stipulated time.

B. Query Graph Formation for Access Cost Optimization

In order to this, each given query is reframed as a graph that connects the triplets using join options (Subject and Predicate are interpreted as keys and Object can be either a key or an arbitrary data in a given triplet). Further, these graphs will be merged that portrays single graph (see Figure 1).

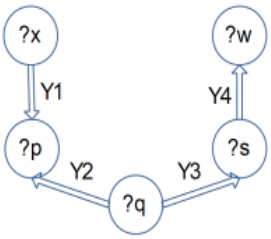

Figure 1(a): Input Query Q1

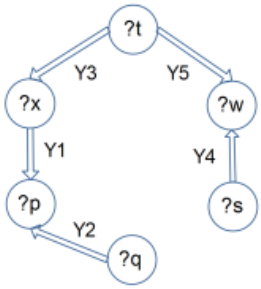

Figure 1(b): Input Query Q2

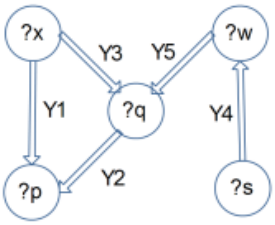

Figure 1(c): Input Query Q3

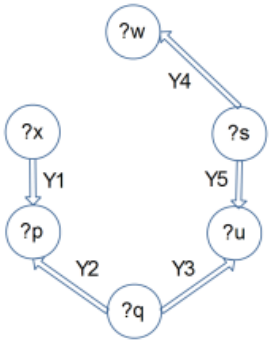

Figure 1(d): Input Query Q4

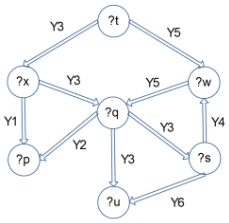

Figure 1: Reformation of the queries and their optimization in regard to Query Index Map from the resultant graph is following.

Qid1: $?t[Y3]?x[Y3]?q \Rightarrow ?t[Y3]?q$ // if the subject $?t$ and the object $?q$ has connected under the same predicate $[Y3]$ that appear in common in the source query

Qid2: $?t[Y3]?x[Y3]?q[Y3]?u \Rightarrow Qid1[Y3]?u$

Qid3: $?t[Y3]?x[Y3]?q[Y2]?p \Rightarrow Qid1[Y2]?p$

Corresponding Author: K.SHAILAJA, ***Email Id:*** *shailajamtech2006@yahoo.co.in*
History: Received: September 04, 2018, Revised: November 25, 2018, Accepted: December 28, 2018

Qid4: $?t[Y3]?x[Y3]?q[Y3]?s \Rightarrow Qid1[Y3]?s$ // if the subject $?t$ and the object $?s$ has connected under the same predicate $[Y3]$ that appear in common in the source query

Qid5: $?t[Y3]?x[Y3]?q[Y3]?s[Y6]?u \Rightarrow Qid4[Y6]?u$

Qid6: $?t[Y3]?x[Y3]?q[Y3]?s[Y4]?w[Y5]?q[Y2]?p \Rightarrow Qid4[Y4]?w[Y5]?q[Y2]?p$ Qid7: $?t[Y3]?x[Y1]?p$

Qid8: $?t[Y5]?w[Y5]?q[Y2]?p \Rightarrow ?t[Y5]?q[Y2]?p$ // if the subject $?t$ and the object $?q$ has connected under the same predicate $[Y5]$ that appear in common in the source query.

The above practice of forming query chains intends to optimize the access cost. However, the optimization of search space is not in the context of the Query graph formation. In regard to this the query reformation by Query Index Map (QIM).

C. Query Reformation under QIM for Search Space Optimization

In general scenario, a query attempts to discover the results from the query built by a triplet under multiple conditions that are interlinked by any of the join condition. Hence, a query template can be built in the form of a graph structure such that, a subject and object as source and target vertices in respective order, which connected by a predicate as edge. In order to this graph can be built with multiple edges that linked by unions, optional, filters, value aggregations, path expressions, sub-queries, value assignment such that the target vertex of an edge can be the source vertex of the other edge in sequence. Further, the unique graph sequences will be collected such that each sequence will have the unique pair of source-vertex (the vertex that begins the graph sequence) and end-vertex (a vertex that ends the graph sequence). The predicates found in each graph sequence often can be recursive. Further, these graph sequences are optimized in regard to access cost and search space. Each edge of the graph sequence denotes predicate and unidirectional. Most of the existing contributions attempts to optimize the query by considering each predicate are unidirectional. In contrast to this, the proposal can swap the subject as object and object as subject of the corresponding edge referred by the given predicate, if the overall counts of the values presented in triple store for a subject are less than the values considered for an object under defined predicate. An illustrative example of this argument is follows.

Let the subject is department id, and the employee id is object representing given set of employee ids. If there is query that explores the department id of the employee having employee id given as input for object. In a general scenario, if the overall department ids are only three, and the count of employee ids given as input for object of the query are greater than the overall department ids, then the optimal query can be as follows.

Select all department ids, for each department id, find employee ids having that department id. Find the intersection of the “resultant employee ids” and “employee ids given as input” is not empty, then the department id is one of the results.

Here the search iterates only for the department ids, which is less than the count of employee ids given as input to the query. If the department ids are only 3, and the input employee ids are 10, then the department id as subject, employee id as object under the predicate that discovers the department id of the each employee id would iterates 10 times, whereas, if you swap the department id as object for all possible department ids (count in 3), and the employee ids as subject, then the iterations limits to maximum 4 (3 iterations for all three department ids, one is to extract all department ids). Let explore a SPARQL query as illustrative example that follows

```
SELECT? country Name
WHERE {
? country a typr: Land Locked Countries.
? country rdfs: label? country Name.
? country prop: population Estimate? population.
FILTER (? population > 15000000).
}
```

The above stated query is framed using SPARQL that fetches name of the countries having no coastline as boarder on any side also having population more than 1.5 crore.

The search space initially fetches the countries that are land locked (countries that are not having coast line) and then filters the countries having population greater than 15000000.

If the number of countries having population greater than 15000000 is lesser than the number of countries landlocked then search space can be optimized by selecting countries having population greater than 15000000 and then filter the countries that are land locked.

According to the description given, further, the process of the proposal denotes the search space in terms of number of iterations required to access the store corresponding to the respective triplet. If the counts of inputs related to the subject are less than the given inputs of the object, then the subject and object will be swapped, but the corresponding predicate remains same. In order to this the statistics related to the subject and object in the corresponding query will be fetched and determines that the query is intended to optimize the search space in current state of the triplet or by swapping the subject and object under same predicate.

The process flow of the proposal is devised in the form of algorithmic approach that follows:

Let Q be the set of queries buffered in a given stipulated time

For each query $\{q \exists q \in Q\}$ Begin

For each triplet $\{t \exists t \in q\}$ Begin

Prepare graph node $g = \{p,[s,o] \exists s \in t, o \in t, p \in p\}$ that connects subject $\{s \exists s \in t\}$ and object $\{o \exists o \in t\}$ as vertices by the predicate $\{p \exists p \in t\}$ as edge.

End

The interlinked set of these graph nodes further depicts the graph G_q of the query q

End

Perform union operation on all of these graphs that determines the final graph G

For each sequence of the graph nodes gs that begins with vertex $?a$ and ends with $?b$ Begin //Perform the query formation from the resultant graph

If all of the edges depicted in gs are referring the predicate P Begin

Search QIM to identify the $?a[P]?b$ is a valid triplet or not

If vertex $?a$ that begins the graph sequence gs and the vertex $?b$ that ends the graph sequence gs and the predicate P are valid triplet Begin

Optimize the query represented by graph sequence gs as $?a[P]?a$.

End

End

End

For each resultant query rq of the graph G Begin //

If reformation of the query rq is optimal to search space Begin //Search QIM to identify that the query rq reformation by changing the order of the vertices optimize the search space or not

Accept the reformation of the query rq

End

End

IV. EXPERIMENTAL STUDY

The suggested approach COPQ and benchmark approach Lusail [26] presented in contemporary literature that intended to deal with similar objective of the COPQ are implemented through Java interface and performed on Intel-5G processor. The distributed environment of different triple-stores and different users is performed through Java-RMI. This enables simultaneous entry of SPARQL queries by the users. Efficiency metrics of the outcome from the proposed method and benchmark methods are evaluated through expression-language R [36].

Corresponding Author: K.SHAILAJA, ***Email Id:*** *shailajamtech2006@yahoo.co.in*
History: Received: September 04, 2018, Revised: November 25, 2018, Accepted: December 28, 2018

A. Input Query Pattern

The experimental setup continuously produces the synthesized group of SPARQL query chains and streams towards the proposal COPQ and contemporary model Lusail [26] to explore results from distributed triple stores FACTBOOK, LUBM and FOAF [37], [38], [39]. These chains are generated so that every chain consists of inconsistent count of query sequences, which obtains the outcomes of triple-stores. Further, the queries buffered for stipulated time will be optimized continuously by the proposal COPQ and contemporary model Lusail. The volume of query components in every chain varied inconsistently between 5 and 25.

B. Performance Analysis

The parameter "time duration for query optimization" is measured for efficiency evaluation of the proposed method that compared to the contemporary model Lusail. This metric denotes the time taken to optimize the set of queries. Further, the "ratio of memory used in query execution" is also used as a key performance metric. This metric denotes the memory used to execute optimized queries against the memory used to execute the source queries without applying any of the optimization processes. In addition to these parameters, "ratio of triple store access cost" is also used for evaluation. This ratio determines the ratio of visits to triple stores required for optimized query against actual number of visits to triple stores required for given input query.

In order to scale the performance under metrics portrayed above, the overall experiments carried in ten folds, such that each fold again having ten windows, where each window denotes the set of queries buffered from a query stream in a given stipulated time, which is 30 sec.

Performance analysis is portrayed by comparing the results obtained for above stated metrics for the proposed COPQ and contemporary model Lusail.

The Table 1 and Table 2 denote the statistics obtained for the metric time duration taken for the query optimization.

Table 1: time in seconds taken for query optimization carried on each query window of each fold under proposed COPQ.

	COPQ									
	#F1	#F2	#F3	#F4	#F5	#F6	#F7	#F8	#F9	#F10
#w1	69.83	71.22	73.35	69.49	46.31	33.84	28.81	40.09	38.76	28.88
#w2	70.24	73.31	75.7	77.99	48.25	36.98	36.01	30.67	33.45	31.17
#w3	67.88	69.6	74.98	75.44	45.65	41.83	31.12	40.24	35.61	30.5
#w4	77.71	80.47	81.14	82.15	58.13	52.53	40.17	41.99	45.33	42.9
#w5	82.06	78.29	78.58	78.49	61.75	48.46	38.51	50.15	49.37	45.75
#w6	82.28	76.29	82.11	86.39	55.54	51.48	46.73	46.24	49.42	41.34
#w7	84.42	82	79.3	81.04	64.35	43.61	43.4	40.58	47.21	46.42
#w8	74.59	71.13	73.14	74.33	51.87	32.91	34.35	38.83	37.49	34.02
#w9	82.47	80.38	80.93	86.99	63.69	51.6	45.71	40.73	44.06	48.04
#w10	70.02	75.31	72.17	70.09	54.46	33.96	27.79	34.58	33.4	35.58
Mean	76.15 ± 6.06	75.8 ± 4.19	77.14 ± 3.52	78.24 ± 5.77	55 ± 6.61	42.72 ± 7.56	37.26 ± 6.46	40.41 ± 5.13	41.41 ± 6.06	38.46 ± 6.87

Table 2: Time in seconds taken for query optimization carried on each query window of each fold under proposed COPQ.

LUSAIL

	#F1	#F2	#F3	#F4	#F5	#F6	#F7	#F8	#F9	#F10
#w1	69.4	67.51	78.63	67.79	68.05	70.02	71.01	73.9	76.53	65.25
#w2	82.03	85.1	78.9	86.69	87.89	87.92	89.43	88.54	84.59	83.68
#w3	71.81	67.04	70.85	70.73	71.2	73.45	79.47	73.28	71.82	72.69
#w4	69.54	69.83	70.07	75.05	76.26	72.53	71.29	76.57	72	66.85
#w5	79.62	85.57	86.68	83.75	84.74	84.49	80.97	89.16	89.3	76.24
#w6	69.3	66.68	77.5	75.9	71.88	73.24	70.91	79.48	79.42	69.03
#w7	68.56	72.75	71.71	77.06	73.89	74.72	72.65	72.85	74.52	71.39
#w8	81.89	82.77	87.47	79.43	79.69	85.41	89.14	85.87	89.11	82.08
#w9	82.13	85.93	80.04	78.58	84.06	84.7	89.53	82.96	81.7	79.9
#w10	82.87	79.86	85.82	77.42	82.05	83.22	87.79	89.59	86.6	77.54
Mean	75.72 ± 6.09	76.3 ± 7.88	78.77 ± 6.16	77.24 ± 5.26	77.97 ± 6.34	78.97 ± 6.37	80.22 ± 7.86	81.22 ± 6.5	80.56 ± 6.38	74.47 ± 6.08

The average time taken to query optimization for fold#1 to fold#4 are almost same in regard to the both methods COPQ and LUSAIL, which is since the query index map that used in the proposal is having limited or no statistics of the earlier queries submitted for execution. Hence the both methods are relied on ASK queries to optimize the queries submitted for execution. Over the time, since the fold#5 to fold#10, the average time taken to optimize the queries submitted in respective folds is observed as considerably low in regard to proposed model COPQ that compared to the contemporary model LUSAIL. This is since; the proposal is relying on query index map, which is in the local scope of the proposed model COPQ whereas the contemporary model relies on ASK queries (see Figure 2).

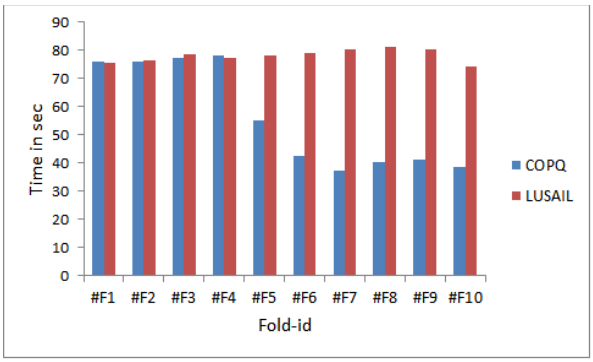

Figure 2: Graphical representation of average time duration for query optimization of COPQ and LUSAIL.

The statistics of the other metric "ratio of memory used for executing the optimized queries" of each window of each fold are portrayed in Table 3 and Table 4, which are obtained for proposed model COPQ, and contemporary model LUSAIL in respective order.

Corresponding Author: K.SHAILAJA, ***Email Id:*** *shailajamtech2006@yahoo.co.in*
History: Received: September 04, 2018, Revised: November 25, 2018, Accepted: December 28, 2018

Table 3: The ratio of memory used by COPQ for optimal query execution against the memory required to execute the queries as in the state of they submitted.

COPQ

	#F1	#F2	#F3	#F4	#F5	#F6	#F7	#F8	#F9	#F10
#w1	0.69	0.49	0.58	0.49	0.45	0.43	0.67	0.56	0.45	0.51
#w2	0.7	0.5	0.7	0.6	0.63	0.61	0.7	0.57	0.62	0.57
#w3	0.77	0.56	0.72	0.6	0.6	0.54	0.73	0.56	0.59	0.61
#w4	0.62	0.43	0.56	0.49	0.48	0.5	0.64	0.56	0.48	0.46
#w5	0.7	0.54	0.66	0.64	0.64	0.58	0.69	0.57	0.6	0.56
#w6	0.75	0.56	0.74	0.56	0.59	0.56	0.71	0.6	0.62	0.62
#w7	0.69	0.46	0.62	0.48	0.45	0.45	0.64	0.48	0.45	0.54
#w8	0.65	0.44	0.54	0.52	0.49	0.48	0.66	0.52	0.45	0.46
#w9	0.69	0.46	0.63	0.45	0.44	0.46	0.68	0.56	0.47	0.52
#w10	0.71	0.53	0.66	0.6	0.63	0.59	0.73	0.65	0.62	0.54
Mean	0.7 ± 0.04	0.5 ± 0.05	0.64 ± 0.06	0.54 ± 0.06	0.54 ± 0.08	0.52 ± 0.06	0.69 ± 0.03	0.56 ± 0.04	0.54 ± 0.08	0.54 ± 0.05

Table 4: The ratio of memory used by LUSAIL for optimal query execution against the memory required to execute the queries as in the state of they submitted.

LUSAIL

	#F1	#F2	#F3	#F4	#F5	#F6	#F7	#F8	#F9	#F10
#w1	0.84	0.64	0.9	0.66	0.57	0.77	0.84	0.74	0.7	0.67
#w2	0.87	0.62	0.86	0.64	0.66	0.76	0.84	0.75	0.65	0.71
#w3	0.88	0.6	0.87	0.64	0.67	0.74	0.87	0.73	0.73	0.65
#w4	0.87	0.64	0.9	0.71	0.63	0.72	0.8	0.71	0.68	0.69
#w5	0.83	0.6	0.86	0.72	0.64	0.71	0.8	0.71	0.71	0.72
#w6	0.72	0.57	0.81	0.63	0.47	0.64	0.71	0.59	0.54	0.65
#w7	0.76	0.53	0.77	0.61	0.56	0.61	0.74	0.59	0.57	0.64
#w8	0.73	0.55	0.81	0.64	0.48	0.62	0.74	0.58	0.62	0.59
#w9	0.78	0.58	0.81	0.55	0.5	0.66	0.78	0.61	0.57	0.59
#w10	0.74	0.53	0.78	0.57	0.51	0.66	0.78	0.62	0.59	0.61
Mean	0.8 ± 0.06	0.59 ± 0.04	0.84 ± 0.04	0.64 ± 0.05	0.57 ± 0.07	0.69 ± 0.06	0.79 ± 0.05	0.66 ± 0.07	0.64 ± 0.06	0.65 ± 0.04

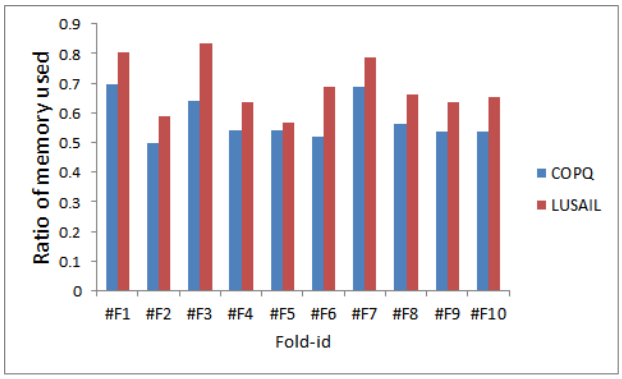

Figure 3: Graphical representation of average ratio of memory used in query execution of COPQ and LUSAIL.

The average ratio of memory used for each fold of optimal queries execution observed for COPQ are considerably low that compared to the counterpart LUSAIL (see Figure 3). This is since the COPQ is optimizing the queries by conditionally swapping the subjects and objects in the given query.

The statistics obtained for other metric "ratio of triple store access cost" of optimal queries against the access cost required to execute the queries in the state of as they submitted are portrayed in Table 5, and Table 6, which are obtained from COPQ and LUSAIL in respective order.

Table 5: The ratio of triple store access cost of the COPQ for optimal query execution against the access cost observed to execute the queries as in the state of they submitted.

	COPQ									
	#F1	#F2	#F3	#F4	#F5	#F6	#F7	#F8	#F9	#F10
#w1	0.81	0.84	0.78	0.76	0.74	0.8	0.76	0.79	0.83	0.81
#w2	0.83	0.77	0.86	0.78	0.79	0.81	0.79	0.75	0.84	0.85
#w3	0.76	0.81	0.85	0.8	0.74	0.79	0.79	0.72	0.75	0.84
#w4	0.66	0.77	0.68	0.69	0.63	0.63	0.65	0.67	0.65	0.67
#w5	0.73	0.74	0.7	0.68	0.68	0.65	0.64	0.69	0.73	0.68
#w6	0.69	0.71	0.77	0.71	0.68	0.63	0.68	0.63	0.66	0.72
#w7	0.78	0.82	0.81	0.81	0.81	0.74	0.79	0.72	0.84	0.78
#w8	0.72	0.68	0.74	0.64	0.67	0.68	0.69	0.7	0.66	0.7
#w9	0.77	0.86	0.81	0.84	0.76	0.76	0.75	0.72	0.83	0.82
#w10	0.72	0.72	0.74	0.66	0.61	0.7	0.65	0.69	0.65	0.74
Mean	0.75 ± 0.05	0.77 ± 0.06	0.78 ± 0.06	0.74 ± 0.07	0.71 ± 0.06	0.72 ± 0.07	0.72 ± 0.06	0.71 ± 0.04	0.74 ± 0.08	0.76 ± 0.07

Corresponding Author: K.SHAILAJA, ***Email Id:*** *shailajamtech2006@yahoo.co.in*
History: Received: September 04, 2018, Revised: November 25, 2018, Accepted: December 28, 2018

Table 6: The ratio of triple store access cost of the LUSAIL for optimal queries execution against the access cost observed to execute the queries as in the state of they submitted.

	LUSAIL									
	#F1	#F2	#F3	#F4	#F5	#F6	#F7	#F8	#F9	#F10
#w1	0.84	0.79	0.9	0.84	0.8	0.91	0.79	0.8	0.87	0.9
#w2	0.96	0.92	0.93	0.95	0.87	0.95	0.88	0.95	0.94	0.97
#w3	0.85	0.77	0.84	0.9	0.84	0.88	0.82	0.79	0.9	0.92
#w4	0.85	0.76	0.91	0.87	0.86	0.89	0.83	0.78	0.9	0.88
#w5	0.84	0.82	0.94	0.89	0.81	0.84	0.76	0.82	0.86	0.85
#w6	0.94	0.85	0.92	0.96	0.87	0.96	0.88	0.88	0.93	0.93
#w7	0.97	0.9	0.98	0.92	0.92	0.92	0.91	0.94	0.98	0.99
#w8	0.87	0.84	0.86	0.89	0.85	0.87	0.82	0.86	0.91	0.87
#w9	0.97	0.87	0.94	0.97	0.9	0.99	0.94	0.92	0.99	0.94
#w10	0.95	0.93	0.97	0.99	0.86	0.93	0.87	0.96	0.94	0.92
Mean	0.9 ± 0.06	0.84 ± 0.06	0.92 ± 0.04	0.92 ± 0.05	0.86 ± 0.04	0.91 ± 0.04	0.85 ± 0.05	0.87 ± 0.06	0.92 ± 0.04	0.92 ± 0.04

The average ratio of access cost for each fold of optimal queries execution observed for COPQ are considerably significant that compared to the counterpart LUSAIL (see Figure 4). This is since the COPQ aimed to minimize the count of resultant triples by reformation of the queries and conditionally swapping the subjects and objects in the given query.

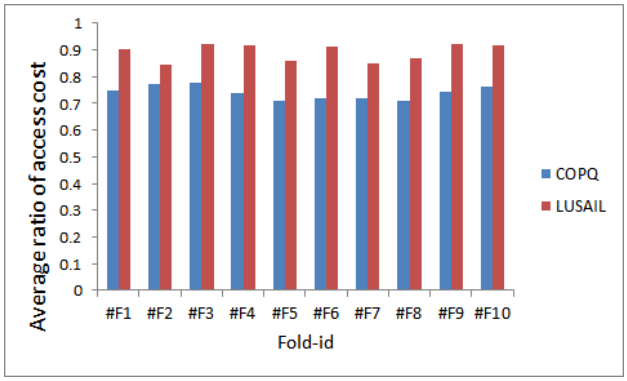

Figure 4: Graphical representation of average ratio of triple store access cost of COPQ and LUSAIL.

Hence, the statistics of the experimental results emerged in this study evincing that the query optimization approach COPQ put forward in this manuscript recorded superior performance as compared to its counterpart method called LUSAIL.

V. CONCLUSION

This research work suggested a novel approach COPQ for determining efficient patterns towards parallel query planning. Contrary to the contemporary model Lusail [25], [26], this approach defined history log called Query Index Map (QIM) that buffers the statistics of the queries executed on the target distributed RDF stores. These statistics are further utilized to optimize the queries buffered in a given stipulated time from the query streams. The contemporary approach is using ASK queries to optimize the target queries, contrary to this, the proposed method uses query index map that available as local to the query processing engine. The usage of this Query Index Map improvises the query optimization in regard to "time duration for query optimization", "ratio of memory used in query execution", and "ratio of triple store access cost". For determining the efficiency of the proposed model, a

simulation study has been performed by scaling the experimental results that compared with the results obtained from contemporary model Lusail. The future research can extend the COPQ to optimize the search process related to Query Index Map, which is portrayed as the major contribution of this manuscript.

REFERENCES

[1] Lassila, Ora, and Ralph R. Swick. "Resource description framework (RDF) model and syntax specification." (1999).
[2] Prud, Eric, and Andy Seaborne. "SPARQL query language for RDF." (2006).
[3] Jena, A. "Apache jena. jena. apache. org {Online}." (2013).
[4] Gurajada, Sairam, Stephan Seufert, Iris Miliaraki, and Martin Theobald. "TriAD: a distributed shared-nothing RDF engine based on asynchronous message passing." In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 289-300. ACM, 2014.
[5] Schätzle, Alexander, Martin Przyjaciel-Zablocki, Thorsten Berberich, and Georg Lausen. "S2X: graph-parallel querying of RDF with GraphX." In Biomedical Data Management and Graph Online Querying, pp. 155-168. Springer, Cham, 2015.
[6] Xin, Reynold S., Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. "Graphx: A resilient distributed graph system on spark." In First International Workshop on Graph Data Management Experiences and Systems, p. 2. ACM, 2013.
[7] Dickinson, Ian. "Jena Ontology API." On the WWW, at http://jena. sourceforge. net/ontology/index. html (2009).
[8] Jiang, Lili, and Jie Luo. "Schema-Based Query Rewriting in SPARQL." In International Conference on Knowledge Science, Engineering and Management, pp. 275-285. Springer, Cham, 2016.
[9] Ahmeti, Albin, Diego Calvanese, and Axel Polleres. "Updating RDFS aboxes and tboxes in SPARQL." In International Semantic Web Conference, pp. 441-456. Springer, Cham, 2014.
[10] Calvanese, Diego, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Tractable reasoning and efficient query answering in description logics: The DL-Lite family." Journal of Automated reasoning, Vol: 39, no. 3 (2007): 385-429.
[11] Shi, Jiaxin, Youyang Yao, Rong Chen, Haibo Chen, and Feifei Li. "Fast and Concurrent RDF Queries with RDMA-Based Distributed Graph Exploration." In OSDI, vol. 16, pp. 317-332. 2016.
[12] Montoya, Gabriela. "Answering SPARQL Queries using Views." PhD diss., Université de Nantes, 2016.
[13] Sequeda, Juan F., and Daniel P. Miranker. "Ultrawrap: SPARQL execution on relational data." Web Semantics: Science, Services and Agents on the World Wide Web Vol: 22 (2013): pp. 19-39.
[14] Abdelaziz, Ibrahim, Razen Harbi, Semih Salihoglu, and Panos Kalnis. "Combining Vertex-centric Graph Processing with SPARQL for Large-scale RDF Data Analytics." IEEE Transactions on Parallel and Distributed Systems Vol: 28, no. 12 (2017): 3374-3388.
[15] Harbi, Razen, Ibrahim Abdelaziz, Panos Kalnis, Nikos Mamoulis, Yasser Ebrahim, and Majed Sahli. "Accelerating SPARQL queries by exploiting hash-based locality and adaptive partitioning." The VLDB Journal—The International Journal on Very Large Data Bases Vol: 25, no. 3 (2016): 355-380.
[16] Galárraga, Luis, Katja Hose, and Ralf Schenkel. "Partout: a distributed engine for efficient RDF processing." In Proceedings of the 23rd International Conference on World Wide Web, pp. 267-268. ACM, 2014.
[17] Zeng, Kai, Jiacheng Yang, Haixun Wang, Bin Shao, and Zhongyuan Wang. "A distributed graph engine for web scale RDF data." In Proceedings of the VLDB Endowment, vol. 6, no. 4, pp. 265-276. VLDB Endowment, 2013.
[18] Lee, Kisung, and Ling Liu. "Scaling queries over big RDF graphs with semantic hash partitioning." Proceedings of the VLDB Endowment Vol: 6, no. 14 (2013): 1894-1905.
[19] Huang, Jiewen, Daniel J. Abadi, and Kun Ren. "Scalable SPARQL querying of large RDF graphs." Proceedings of the VLDB Endowment Vol: 4, no. 11 (2011): 1123-1134.
[20] Abdelaziz, Ibrahim, Razen Harbi, Zuhair Khayyat, and Panos Kalnis. "A survey and experimental comparison of distributed SPARQL engines for very large RDF data." Proceedings of the VLDB Endowment Vol: 10, no. 13 (2017): 2049-2060.
[21] Özsu, M. Tamer. "A survey of RDF data management systems." Frontiers of Computer Science Vol: 10, no. 3 (2016): 418-432.
[22] Saleem, Muhammad, and Axel-Cyrille Ngonga Ngomo. "Hibiscus: Hypergraph-based source selection for SPARQL endpoint federation." In European Semantic Web Conference, pp. 176-191. Springer, Cham, 2014.

Corresponding Author: K.SHAILAJA, ***Email Id:*** *shailajamtech2006@yahoo.co.in*
History: Received: September 04, 2018, Revised: November 25, 2018, Accepted: December 28, 2018

[23] Acosta, Maribel, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. "ANAPSID: an adaptive query processing engine for SPARQL endpoints." In International Semantic Web Conference, pp. 18-34. Springer, Berlin, Heidelberg, 2011.

[24] Görlitz, Olaf, and Steffen Staab. "Splendid: Sparql endpoint federation exploiting void descriptions." In Proceedings of the Second International Conference on Consuming Linked Data-Volume 782, pp. 13-24. CEUR-WS. org, 2011.

[25] Abdelaziz, Ibrahim, Essam Mansour, Mourad Ouzzani, Ashraf Aboulnaga, and Panos Kalnis. "Query optimizations over decentralized RDF graphs." In 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 139-142. IEEE, 2017.

[26] Abdelaziz, Ibrahim, Essam Mansour, Mourad Ouzzani, Ashraf Aboulnaga, and Panos Kalnis. "Lusail: a system for querying linked data at scale." Proceedings of the VLDB Endowment Vol: 11, no. 4 (2017): 485-498.

[27] Schwarte, Andreas, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. "Fedx: Optimization techniques for federated query processing on linked data." In International Semantic Web Conference, pp. 601-616. Springer, Berlin, Heidelberg, 2011.

[28] Ambite, Jose Luis, and Craig A. Knoblock. "Flexible and Scalable Query Planning in Distributed and Heterogeneous Environments." In AIPS, pp. 3-10. 1998.

[29] Duschka, Oliver M., and Michael R. Genesereth. "Query planning in infomaster." In Proceedings of the 1997 ACM symposium on Applied computing, pp. 109-111. ACM, 1997.

[30] Roth, Mary Tork, and Peter M. Schwarz. "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources." In VLDB, vol. 97, pp. 25-29. 1997.

[31] Tomasic, Anthony, Louiqa Raschid, and Patrick Valduriez. "Scaling heterogeneous databases and the design of DISCO." PhD diss., INRIA, 1995.

[32] Ouzzani, Mourad, and Athman Bouguettaya. "Query processing and optimization on the web." Distributed and Parallel Databases Vol: 15, no. 3 (2004): 187-218.

[33] Halevy, Alon Y., Zachary G. Ives, Peter Mork, and Igor Tatarinov. "Piazza: data management infrastructure for semantic web applications." In Proceedings of the 12th international conference on World Wide Web, pp. 556-567. ACM, 2003.

[34] Franconi, Enrico, Gabriel Kuper, Andrei Lopatenko, and Ilya Zaihrayeu. "Queries and updates in the coDB peer to peer database system." In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pp. 1277-1280. VLDB Endowment, 2004.

[35] Bonifati, Angela, Elaine Chang, Terence Ho, Laks V. Lakshmanan, Rachel Pottinger, and Yongik Chung. "Schema mapping and query translation in heterogeneous P2P XML databases." The VLDB Journal—The International Journal on Very Large Data Bases Vol: 19, no. 2 (2010): 231-256.

[36] Ihaka, Ross, and Robert Gentleman. "R: a language for data analysis and graphics." Journal of computational and graphical statistics Vol: 5, no. 3 (1996): 299-314.

[37] Miller, L., and D. Brickley. "The friend of a friend (foaf) project." (2000).

[38] Factbook, C. I. A. "The World Factbook; 2010." See also: http://www cia gov/library/publications/the-world-factbook, accessed January 30 (2015).

[39] Guo, Yuanbo, Zhengxiang Pan, and Jeff Heflin. "LUBM: A benchmark for OWL knowledge base systems." Web Semantics: Science, Services and Agents on the World Wide Web 3, no. 2-3 (2005): 158-182.