Received: February 17, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Assessment and Optimization of User Imprecise Queries in Cloud Environments

Pichaimuthu Mohankumar[1]* **Balusamy Balamurugan[1]**

[1]*Vellore Institute of Technology University, Vellore, India*
* Corresponding author's Email: pennarasimohan@gmail.com

Abstract: Data size growth rate arose massively day to day due to database and internet applications. It has become a challenging task to organize those data and to provide the user a relevant data in time with correct and compatible manner. Further choice was a pervasive feature of social life that profoundly affects people. They work with assumptions that stored data represent the proper subset of real world data and make a quick decision based on imprecise knowledge in daily life for survivals which tends to get irrelevant output .Sometimes for other input, this may be exact so it should not remove instead, managed to utilize appropriately to minimize the processing time. Moreover, optimism significance relies on user satisfactions. This paper provides a vision to tackle these issues by assessing the imprecise incoming query and reutilizing for future user instead of rejecting as wrong or irrelevant. To address optimization issues in this paper, we proposed the techniques for optimizing the queries to provide customers with fast data retrieval. In our model, Query processing: A 3-step process that transforms a high-level query (MongoDB) into an equivalent and more efficient lower-level query (relational algebra). Further MEAN stack based cooperative semantic approach was deployed in cloud environments as novelty to provide solution with the level of performance significance.

Keywords: Imprecise, Queries, Optimisation, MEAN stack, Cloud.

1. Introduction

Life is always full of choices. People work with assumptions that stored data represent the proper subset of real world data and make a quick decision based on imprecise knowledge[1] in daily life for survivals which tends to get irrelevant output .Sometimes for other input, this may be exact so it should not remove instead, managed to utilize appropriately to minimize the processing time. In most applications, database servers are queried by multiple clients. When using the classic semantic caching [2] approach, clients store and manage their own local caches independently. If the number of clients is high, the amount of data sent by database server and queries response times can rapidly increase even when caching is used. The performance can be further improved by allowing clients to share their entries in a cooperative way. Another limitation of existing semantic caching solutions is that they do not handle update queries. Modifications performed in the database are not propagated to cache entries stored by clients. Therefore, the Cooperative Semantic Caching [3] approach will extend the general semantic caching [4] mechanism by using a Peer-to-peer [5] approach in order to enable clients to share their local semantic caches in a cooperative manner. When executing a query, the content of both the local semantic cache and entries stored in caches of other clients can be used. A new query will be split into a probe, remote probes, and a remainder query. The probe retrieves the part of the answer which is available in the local cache. Remote probes retrieve those parts of the query which are available in caches of other clients. The remainder retrieves the missing n-tuples from the server. In order to execute the query rewriting, the cache entries of all clients will be indexed in a distributed data structure built on top of a P2P overlay that is formed by all clients which are interrogating a particular database server. Such an approach increases the performance of databases systems and presents economic

advantages when used in a cloud-computing environment. By using MEAN stack approach to execute a query, the content of both the local semantic cache and entries stored in caches of other clients can be used. A new query will be split into a probe, remote probes, and a remainder query. The probe recovers the part of in the local cache. Other regains their queries found caches. In order to execute the query rewriting, the cache entries of all clients are indexed in a distributed data for query optimization. The MEAN (Mongo, Express, Angular, Node) based cooperative semantic approach also contains a mechanism for handling operations that modify the content of the collections during the query handling process. Thus, Cooperative semantic caching will use Peer to peer technology in order to develop a cooperative caching solution. It will include a suitable and efficient mechanism for handling update queries, it will support select-project queries, where a query predicate is a n-dimensional range condition, it will design a heuristic to dynamically decide when using the cooperative cache is beneficial or not. As example Indian railway online reservation, is the very vast application which that deals many outcomes .To book a online ticket we need to have the database help, i.e.; trains timings, seats availability, nearby stations, source and destination etc., All these above mentioned information will change for every minute. The database has to update the information almost for every 5 to 10 seconds. Accessing this kind of information and booking the ticket and updating the database and providing the information to the user are both very difficult tasks. To overcome this cooperative semantic cache mechanism was deployed. Here query processing is both the predicates and the resulting tuples of queries are cached. In query processing semantic caching the availability and booking the ticket. Using this CoopSC we are reducing the query processing speed. Our database contains details of all the trains and seats availability, clients from all over the India share this information and book their tickets, this work can be simplified and the processing speed (response time to client query) can be greatly reduced.

To address optimization issues in this paper, we proposed the techniques for optimizing the queries to provide customers with fast data retrieval. In our model, Query processing has presented in 3-steps that transforms a high-level query (of Mongo DB) into an equivalent and more efficient lower-level query for fast optimization. 1. Parsing and translation (Check syntax and verify relations.., Translate the query (Dynamo dB)), 2.Optimization (Generate an optimal evaluation plan (with lowest cost) for the query plan.), 3. Evaluation (The query-execution engine takes an (optimal) evaluation plan, executes that plan and returns the answers to the query).On behalf of optimisation analysis a cloud environment was chosen as it provides a combination of parallel and distributed computing paradigms. It has the characteristics of on demand provisioning of shared pool of configurable computing resources as service. Further it provides a cost effective paradigm of computational, storage databases resources to users over internet. The increasing number of user query data deployed from virtual instances can lead to increase loads. Multiple queries compete for hardware resources causing resources contention within a rapidly changing in environment computational properties. Hence it has become a mandatory and major challenging task for researchers to provide efficient methodologies to execute concurrent queries. The road map of this paper initially discuss about the data analysis and imprecision assessment and query optimisation in terms parametric and multi-objective parametric aspects of user queries as extension.

2. Related work

In this paper, we envision to address that imprecise knowledge can add value to the Semantic Web technologies and information retrieval and with a stack of Semantic Web technologies that allow imprecise knowledge as an essential ingredient for building future applications. It is natural for a user to specify preferences on varied aspects of an entity in the pictured entity ranking task [6]. So, we will expect a user's question to carries with it preferences on multiple aspects; as an example, a preference question for an automotive can that consists of preferences on 3 completely different aspects (i.e., efficiency, price, and reliability). We ought to take apart a question to get preferences on completely different aspects. During this paper, we have a drift to specialize in finding out effectiveness of various ranking strategies, so we have a tendency to assume that the multiple aspects during a user's question have already been metameric in order to cypher the influence of question segmentation on retrieval accuracy. Such a query can even be naturally obtained by providing a multi-aspect question type or asking a user to use a delimiter (e.g., a comma) to separate multiple preferences. As an example, in Figure two, we have a tendency to show a system interface wherever the users will notice hotels in any town by stating their preferences on the assorted aspects of hotels. Although this ranking

downside closely resembles a, data retrieval downside where the reviews of AN entity are often thought to be an entity document, for vital variations in the collections. First, the question is supposed to precise a user's preferences in keywords; so, it's expected to be longer than regular keyword queries on the net. More significantly, the question typically would contain preferences on multiple aspects of AN entity. As we are going to show later within the paper, modelling these aspects will improve ranking accuracy. Second, the ranking criteria [8] square measure to capture however well an entity satisfies a user's preferences instead of the connection of a document to a question as within the case of regular retrieval. Therefore, the matching of narrow-minded words or sentimental analysis would be important. We are going to show that though ancient question enlargement works moderately. This approach, however, has some sensible limitations. First, these approaches assume a fixed range of aspects on a given entity. It's not solely impractical to outline or mine a set of aspects for every class of entities. However, a hard and fast range of aspects would additionally severely limit the kind of queries a user may issue. a lot of significantly, all the add this line, require some oversight therein they need the supply of ratings related to reviews, which cannot continually be gift. we have a tendency to take a lot of general stance, that is to assume restricted data on the opinions and also the aspects being queried and specialize in leveraging strong retrieval models to match the user's preferences for Associate in Nursing entity with opinions thereon entity. Dean et.al [9] proposed a programming model and a framework "Map Reduce" for processing large sets of raw data. A map-reduce program consists of two functions: Map and Reduce. The Map function processes the input data by distributing them to worker nodes for parallel computation and produces a set of intermediate results as key-value pairs, while the reduce function aggregates all the intermediate results with the same key from each node to produce the result. It can be used for structured data analysis of large sets. The limitations of Map Reduce as given in [10] are it produces the necessary secondary indices in an offline batch manner. Hence, secondary indexes are not up-to-date. So newly inserted rows cannot be queried until they are indexed. It does not provide data schema support, declarative query language and cost-based query optimizations. To optimize the execution of queries a number of greedy and approximation algorithms have been proposed earlier. But Kalnis et. al. [11] stated that they do not scale well for realistic workloads. They developed two greedy algorithms which emphasize on finding the most beneficial view in each step instead of finding most promising query. Their extensive experiments showed that their methods outperform the existing one. Expert finding, another relevant space of analysis is professional Finding. Instead of documents the goal is to retrieve a hierarchal list of specialists like an expert on a given topic. The techniques used vary from customary retrieval strategies just like the vector area model to progressive techniques that use probabilistic and language modelling approaches. Although our work is conceptually connected, therein we have a tendency to use data regarding Associate in tending entity to rank entities, in contrast to professional finding we are able to rank any form of entity that narrow content is offered. Also, rather than making an attempt to rank entities supported however well it matches a topic, we have a tendency to specialize in ranking entities supported however well a user's preferences are matched with opinions thereon entity. Opinion Retrieval, Opinion retrieval was initially explored in email form. The goal of opinion retrieval is to find documents (primarily blog posts) that have narrow content. The thought here is to check the flexibility to find opinion expressing posts as this is often essential in specialized searches like web log search. An opinion retrieval system is sometimes engineered on high of ordinary retrieval models where relevant content is initially retrieved, and so opinion analysis is completed on the retrieved content to come solely narrow documents. In distinction, our plan assumes that we have a tendency to have already got the narrow content for a given class of entities. The goal is so to rank the entities within the order of chance that the entity matches the user's preferences. Wang et.al analysed the LARA model without the pre-defined aspect keywords Presented the hybrid generative Latent Aspect Rating Analysis Model (LARAM) containing both aspects modelling and rating prediction, and demonstrated the model to performed the hotel and product review dataset. Ganesan et. al proposed the opinions summarization model and it generates the concise abstract summaries. Opinions graph produces abstractive summaries of highly redundant opinions. First it constructed a textual graph where the texts all summarized and graphs three unique properties of this graph tends to explore and score various sub paths that resulted candidate abstractive summaries. Ganesan & Zhai presented the rank based entities based on the user's preferences. It states the use of several retrieval models and experimentally demonstrates the two models and creates a bench

mark quantitative evaluation of opinion-based entity ranking.Group Lens Research Project at the University of Minnesota has analyzed the movie reviews, Movie Lens Dataset were used to establish the Simple demographic info for the users (age, gender, occupation, zip) and review text is not available. Micropinion Generation Dataset (CNET) was proposed by Ganesan & Zhai to solve the optimization problem by heuristic algorithm and produces the very concise phrases , at each phrase (micro opinion) is a summary of an opinion .

3. Data analysis

Data analysis [11] is needed in connection with query processing, to produce data summary information in the form of rules or assertions that allow semantic query optimisation or direct query answering without consulting the data itself. The purpose of the data analysis considered in this paper is to improve the speed of answering queries on that data. The summary information produced by analyser can either answer a query without consulting the data itself, or else modify the query to a form the data server will be able to process more quickly. Here this summary is used to identify precise or imprecise and to assess whether it's relevant for the processing query or not. Previous work in connection with meta-data discovery for use in query processing has used a machine-learning approach where the result of a query is treated as a set of positive instances (a training set). Let us see now the data analysis how far supportive for query processing.

3.1 Analysis and assessment of imprecise inputs

Handling imprecise data and knowledge in the semantic query processing is not a new issue. The most well-known approaches include fuzzy extensions, probabilistic extensions and possible extensions. But in order to manage to adapt during need, we should see the approach of supporting the imprecise by storing the processed results and uncovered error parameters i.e. imperfect error coverage and reliability. Most approaches prefer look up data and linked open data which support user interaction for resolving imprecise but it's not fit for complete results. In order to leverage the knowledge on the semantic engine requires efforts in different areas. Firstly need to create the possible imprecise knowledge base data sets. This is done by certain set of code which support rule base and implicit knowledge in order to foster a quick adoption, automatic approaches. Second, there need to be means to represent uncertain knowledge and uncertainty degrees on the level of individual axioms; Third, imprecise knowledge needs to be processed in a useful manner. It is clearly not enough to present the user a set of possible statements with their degrees of precision. Furthermore, using uncertain knowledge to derive new facts will be essential. A crucial point in creating and processing imprecise knowledge is the assignment of useful degrees of uncertainty. If uncertainty degrees are chosen badly, the entire approach can be flawed. Once imprecise information is discovered and created with appropriate measures of imprecision, it needs to be stored in order to allow for further processing. As imprecision can occur both on the instance and the schema level, mechanisms of an imprecise Semantic should be able to cope with both, ideally in a uniform way. Assigning useful degrees of imprecision is a crucial point in creating and using imprecise knowledge. The confidence degrees delivered by a machine learning algorithm can be a candidate, but are not necessarily the best measure. In our case we are not chosen a model for imprecise information, instead we deploy neighbour description concept. Here by establishing a distance among document descriptions, usually with some vector metric, and retrieving all the information in the neighbourhood of a request vector, the negative effects of imprecisions in the description are diminished. That is when a query specifies a value of this attribute, all the tuples would be retrieved, whose value for that attribute is in the neighbourhood of the specified value [12].We assumed here that descriptions are subject to imprecision (which is ignored) and requests are precise. The same solution applies when descriptions are precise and requests are imprecise. Such requests would be specified with apparent-precision, and would be answered with the neighbourhood of the request. Again, the negative effects of imprecision in the request would be moderated. We discuss this issue as follows.

3.1.1. Data distances

Data distances approach used to analyse and assess the precision and imprecision among the incoming data. An approach to imprecision that has been applied successfully to both databases systems and information retrieval systems handle imprecise information with distance. The basic idea is to model the real world with apparently-precise descriptions, to define the notion of distance among two descriptions, and thus to create neighbourhoods of descriptions. Thus, any imprecision about a real-

world object is ignored, and an apparently-precise description of it is stored. It is then hoped that this \negligence" would be compensated by having somewhere in the neighbourhood of the true description. When a request for information specifies this true description, would be retrieved, along with the other neighbours of the true description. As an example, consider an information retrieval system that describes documents with sets of keywords [13, 14]. Such systems often represent keyword sets with vectors: the dimension of each vector is the number of possible keywords, and a specific vector position is 1 if a particular keyword is in the set and 0 otherwise. Often, there is uncertainty whether a specific vector is the true description of a given document. By establishing a distance among document descriptions, usually with some vector metric, and retrieving all the information in the neighbourhood of a request vector, the negative effects of imprecisions in the description are diminished. As another example, consider relational database systems. Such systems describe objects with tuples, and often there is uncertainty regarding the value of some attribute in a given tuple. It is possible to [13] establish a distance among the elements of the domain of this attribute. Then, when a query specifies a value of this attribute, all the tuples would be retrieved, whose value for that attribute is in the neighbourhood of the specified value. We assumed here that descriptions are subject to imprecision (which is ignored) and requests are precise. The same solution applies when descriptions are precise and requests are imprecise. Such requests would be specified with apparent-precision, and would be answered with the neighbourhood of the request.

3.1.2. Completeness and accuracy

In order to measure the accuracy and completeness from imprecise can be viewed was discussed. Imprecise information declarations are made of the portions of the database that are prefect models of the real world (and thereby the portions that are possibly imperfect). Thus, like distances and unlike disjunctive values, confidence factors ,probabilistic values or possibilistic values , the descriptions themselves have no special features for representing imperfection (i.e., they appear perfect). However, meta-information provides the distinction between perfect and imperfect information. This approach interprets perfectness, which it terms integrity, as a combination of accuracy and completeness. A description is accurate, if it includes only information that occurs in the real world; a description is complete, if it includes all the information that occurs in the real world. Hence, a description has integrity, if it includes the whole truth (completeness) and nothing but the truth (accuracy). With this information included in the database, the database system 14 can qualify the perfectness of the answers it issues in response to queries: each answer is accompanied by statements that define the portions that are guaranteed to be perfect. A technique is described in [15] for inferring the views of individual answers that are guaranteed to have integrity, from the views of the entire database that are known to have integrity. The notion of view completeness is similar to an assumption that a certain view of the database is closed world [16]. Also, open nulls [17] are actually declarations of views that are non-complete. The notion of accuracy is shown to be a generalization of standard database integrity constraints. Relative accuracy and completeness are similar to the precision and recall measures used in information retrieval [18], and are used to guide the harmonization of inconsistent answers in a multi-database environment.

3.2 Imprecision manipulations and processing

While most of the work on imperfect databases has focused on description imperfection, transaction and processing imperfection also have important impact on the quality of the information delivered to users. In this section we discuss briefly issues and solutions that concern imperfections in the definition of transformations (e.g., queries), in the definition of modifications (e.g., updates or restructuring operations), and in the processing of such transactions.

3.2.1. Transformations

Transformations are operations that derive new descriptions from stored descriptions. The most frequent type of transformation is queries. Imperfect queries may occur for different reasons. At times, users of database systems have insufficient knowledge of the database and database system they are using: they might not have a clear idea of the information available in the database (or how it is organized), or they might not know how to formulate their requests with the tools provided by the system. Requests for information formulated by such naive users' exhibit a high level of imperfection. They range from requests that cannot be interpreted by the system (for reasons that are either syntactical or semantically) to requests that do not achieve correctly the intentions of the users (or

achieve them only in part). Regardless of their level of expertise, occasionally users may try to access a database system with only a vague idea of the information they seek. For example, a user may be accessing an electronic catalogue for a product that would be \interesting" or \exceptional". Alternatively, users could have a clear idea of the information they want, but might lack the information necessary to specify it to the system. An example is a user who wishes to look up the meaning of a word in a dictionary, but cannot provide its correct spelling. To summarize, we distinguish among (1) insufficient knowledge of the information available (or how it is organized), (2) vagueness with respect to the information needed (or how to denote it in terms acceptable to the system), and (3) insufficient knowledge of the system languages and tools that are used to formulate requests. To address all these, the approach has been to develop alternative access tools. Browsers allow users to access information in either situation discussed above [19-21]. Interactive query constructors. Conduct user-system dialogues to arrive at satisfactory formulations of user requests.

3.2.2. Processing

Even when a description D and a transformation t are free of imperfections, the result t(D) may be imperfect because of the methods used by the system to process requests. In certain applications, an information system might allocate only limited computational resources to process a request. For example, a recursive query to a genealogical database to list all the ancestors of a specific individual might be terminated after a predetermined period of time (presumably the number of ancestors retrieved by then would be sufficient. For example, a statistical database system might introduce perturbations into its answers deliberately, for reasons of security. In each case, the answers would exhibit imperfections. Finally, it is sometime considered advantageous to sacrifice accuracy for the sake of simplicity. Recent research on intentional answers has focused on the generation of abstract answers that describe the exhaustive answers compactly, albeit imperfectly. For example, a query to list the employees who earn over 50,000 might be answered simply and compactly \engineers", even when the set of engineers and the set of employees who earn over 50,000 are not exactly the same (e.g., when the two sets overlap substantially, or when one set contains the other).

4. Proposed work

With the increasing volume of data across the large number of applications, the challenge is to distribute the computations, responding to the query along with the distribution of data. As per Bobrowski (2011), relational database management systems have grown overly complex, difficult to manage, and are struggling today to take full advantage of cloud computing technology. So, there is need to reanalyse the design and processing of relational database technologies and refine the existing methods or develop new approaches exclusively (how and what is the method) for the cloud environment. Query processing and optimization are very important and necessary functions for any data base management system. The function of query processing is to transform the query written in a high-level language into a correct and efficient execution plan expressed in a low-level language. An important aspect of query processing is query optimization. As there are many equivalent transformations of the same high-level query, the aim of query optimization is to choose an efficient execution plan for processing a query. It chooses the one that minimizes the resource usage by using the information from the system catalog. The set of query plans selected for examination is formed by examining the possible access paths (e.g., index scan, sequential scan) and join algorithms. As the volume of the data is increasing since the last fifteen years at an exponential speed, research is continually needed to improve the performance and speed of data retrieval from the database management system. There is need for such database management system, which can process highly complex queries and handle terabytes or petabytes of data. Analytical applications are in demand, so that proper planning and decisions can be made from the data stored in the database. Most query optimizers represent query plans as a tree of "plan nodes". A plan node encapsulates a single operation that is required to execute the query.

4.1 Optimizing queries

Memory requirement is determined by query type. There are no simple and generic rules to determine the correlation between the maximum data size that a groups can process with its aggregated memory size. It does not load entire collection into memory, so the amount of available memory doesn't limit the collection size that it can handle. It builds crypt tables in memory, such as the right-hand side table of a join or the result set of an aggregation. In addition, memory as used in I/O buffers, where the number of processor cores on the

cluster and the speed of the scanners determine the amount of buffering that is necessary in order to keep all cores busy. For our m1.xlarge cluster in part 1 of our experiment, but when we performed single table scan, we were able to process tables of 128 GB and above. Because Impala didn't need to cache the entire result set of the query, it streamed the result set back to the client. In contrast, when performing a join operation, Impala may quickly use up a cluster's memory even if the aggregated table size is smaller than the aggregated amount of memory. To make full use of the available resources, it is extremely important to optimize your queries. In this section, we take Q3 used to illustrate some of the optimization techniques.

4.1.1. Parametric query optimization

Classical query optimization associates each query plan with one scalar cost value. Parametric query optimization assumes that query plan cost depends on parameters whose values are unknown at optimization time. Such parameters can for instance represent the selectivity of query predicates that are not fully specified at optimization time but will be provided at execution time. Parametric query optimization therefore associates each query plan with a cost function that maps from a multi-dimensional parameter space to a one-dimensional cost space.

4.1.2. Multi-objective query optimization

There are often other cost metrics in addition to execution time that are relevant to compare query plans. In a cloud computing scenario for instance, one should compare query plans not only in terms of how much time they take to execute but also in terms of how much money their execution costs. Or in the context of approximate query optimization, it is possible to execute query plans on randomly selected samples of the input data in order to obtain approximate results with reduced execution overhead. In such cases, alternative query plans must be compared in terms of their execution time but also in terms of the precision or reliability of the data they generate .Multi-objective query optimization models the cost of a query plan as a cost vector where each vector component represents cost according to a different cost metric. Classical query optimization can be considered as a special case of multi-objective query optimization where the dimension of the cost space (i.e., the number of cost vector components) is one. Different cost metrics might conflict with each other (e.g., there might be one plan with minimal execution time and a different plan with minimal monetary execution fees in a cloud computing scenario). Therefore, the goal of optimization cannot be to find a query plan that minimizes all cost metrics but must be to find a query plan that realizes the best compromise between different cost metrics. What the best compromise is depends on user preferences (e.g., some users might prefer a cheaper plan while others prefer a faster plan in a cloud scenario). The goal of optimization is therefore either to find the best query plan based on some specification of user preferences provided as input to the optimizer (e.g., users can define weights between different cost metrics to express relative importance or define hard cost bounds on certain metrics) or to generate an approximation of the set of Pareto-optimal query plans (i.e., plans such that no other plan has better cost according to all metrics) such that the user can select the preferred cost tradeoff out of that plan set.

4.1.3. Multi-objective parametric query optimization

Multi-objective parametric query optimization generalizes parametric and multi-objective query optimization. Plans are compared according to multiple cost metrics and plan costs may depend on parameters whose values are unknown at optimization time. The cost of a query plan is therefore modelled as a function from a multi-dimensional parameter space to a multi-dimensional cost space. The goal of optimization is to generate the set of query plans that can be optimal for each possible combination of parameter values and user preferences.

4.2 Problematic query

```
explain DELETE FROM xxxxx WHERE aggr_id = 3000010;
+----+-------------+--------------------------+------+-----
| id | select_type | table      | type | possible_keys | key  |
key_len | ref  | rows     | Extra       |
+----+-------------+--------------------------+------+-----
|  1 | SIMPLE      | xxxxx ALL  | NULL          | NULL |
NULL    | NULL | 46611048 | Using where |
+----+-------------+--------------------------+------+-----
1 row in set (0.05 sec)
```

Figure.1 Sample query

4.2.1. Snippet data

In order to align the real-world values to colonize with arbitrarily produced, the outline of the data generated same process in the form of sample data, and here we pasted few of them. The first sections describe that the total no of head books/

books records. It has education, body-mind-spirit, transportation, family relationship, travel, law, literary criticism.

```
$ head books/books
0|1-45812-668-3|EDUCATION|1986-06-14|Shinchosha|45.75
1|9-69091-140-1|BODY-MIND-SPIRIT|1983-07-29|Lefebvre-Sarrut|99.91
2|3-73425-809-9|TRANSPORTATION|1996-07-08|Mondadori|99.45
3|8-23483-356-2|FAMILY-RELATIONSHIPS|2002-08-20|Lefebvre-Sarrut|227.39
4|3-58984-308-3|POETRY|1974-06-13|ESKIMO|234.99
5|2-34120-729-8|TRAVEL|2004-06-30|cargage|120.99
6|0-38870-277-1|TRAVEL|2013-05-26|Education Group|173.99
7|8-74275-772-8|LAW|2012-05-01|Holtzbrinck|182.99
8|4-41109-927-4|LITERARY-CRITICISM|1986-04-
9|8-45276-479-4|TRAVEL|1998-07-04|Lefebvre-Sarrut|80.99

$ head transactions/transactions
0|360677155|84060207|4|2010-03-24 10:24:22
1|228662770|136084430|5|2009-07-03 14:53:09
2|355529188|26348618|9|2009-09-13 11:53:26
3|1729168|20837134|5|2006-01-05 19:31:19
4|196166644|99142444|19|2007-01-02 15:07:38
5|43026573|479157832|17|2010-04-14 16:42:29
6|306402023|356688712|12|2010-05-24 22:15:54
7|359871959|312932516|31|2000-04-03 11:06:38
8|379787207|265709742|45|2013-09-09 06:01:06
9|144155611|137684093|11|2010-06-06 17:07:07
```

Figure.2 Sample tested output

Table 1. Sample tested output 3

Instance type	Processor Architecture	vCPUs	ECU	Memory (GiB)	Internal storage (GB)
M1.xsmall	64-bit	4	8	15	4x420

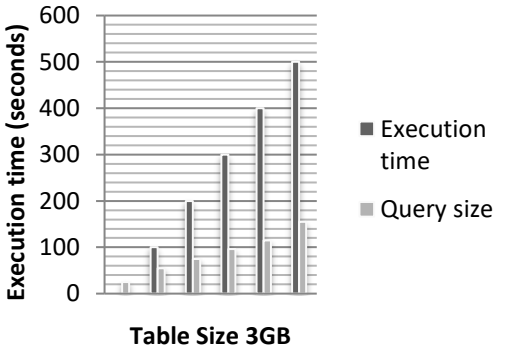

Figure.3 Existing system performance status

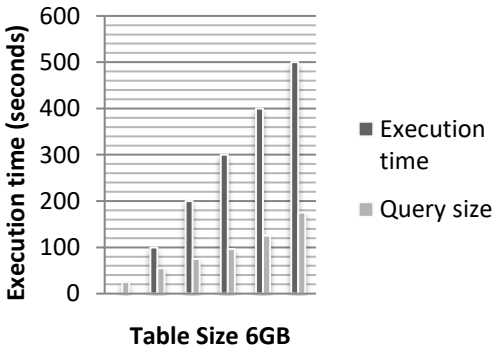

Figure.4 Existing system performance status

Table 2. Sample tested output 3

Instance type	Processor Architecture	vCPUs	ECU	Memory (GiB)	Internal storage (GB)
m2.4x large	64-bit	8	26	8	2x840

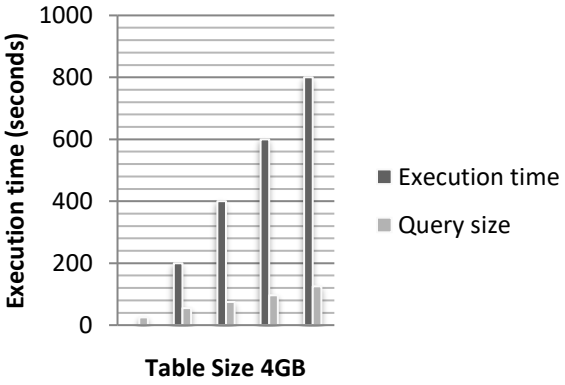

Figure.5 Proposed system performance status

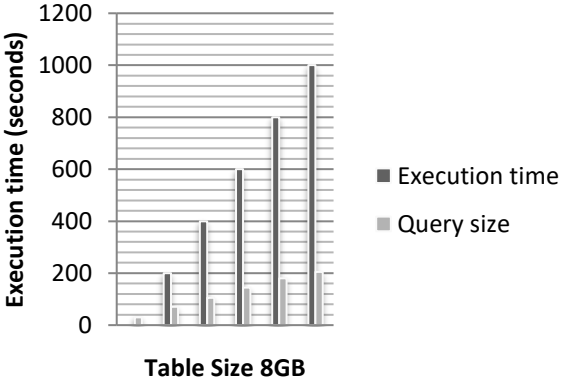

Figure.6 Proposed system performance status

5. Performance test results

During this experimental results were attained on various group of json data. Where as in amazon we use x1.large instance with ec2 instance to be installed as shown in Table 1. By comparing the query performance against the two metrics of the query execution time and output. From this Fig 3 and Fig 4 stated that y axis represents the average execution time has been measured using the time command from various trails. The missing data represented the unsuccessful of first approach due to impact of out-of-memory problematic scenario. Nonetheless, when the input data set is large enough such that the framework overhead is negligible compared to overall query time, initially tested output got only about 3 to 10 times faster. The second one was performed against this approach was tabulated in Table 2, it was not mutable with modified optimization technique. The goal of optimization is usually to generate all query plans that could be optimal for any of the possible parameter value combinations. This yields a set of relevant query plans. At run time, the best plan is selected out of that set once the true parameter values become known. The advantage of Multi-objective parametric query optimization is that optimization (which is in general a very expensive operation) is avoided at run time.

Figures 5 and 6 show goal of optimization is therefore either to find the best query plan based on some specification of user preferences provided as input to the optimizer (e.g., users can define weights between different cost metrics to express relative importance or define hard cost bounds on certain metrics) or to generate an approximation of the set of optimal query plans (i.e., plans such that no other plan has better cost according to all metrics) such that the user can select the preferred cost trade off out of that plan set.

6. Conclusion

Commercial database systems have been relatively slow to incorporate imprecise capabilities. Another hindrance for database system with imprecision capabilities may lie in the expectation of users. Users of database system have come to expect their queries to be interpreted unambiguously and answered with complete accuracy. Our works discussed in this paper provide solution for this .That is by assessing the user imprecise queries and reutilising for future use by cooperative answering aspect. (Where information is considered relevant to the queries delivered along with, or in place of, the standard answer) .Further the article presents an idea of gaining imprecise and incomplete information by novel methodologies. On behalf of performance issue optimization was deployed for parametric queries, multiple queries, multi objective queries in cloud environments. As conclusion our work reach the level of significance for daily database user and as future we extended the work in design of integrating the stand alone models for information retrieval which support the imprecise query processing system as a single one with AI mechanism and experiment on Big data over cloud environments as research .

References

[1] P. Mohankumar and J. Vaideeswaran, "Assessment on Precision-imprecision Essentials in Semantic Query Processing*", Indian Journal of Science and Technology*, Vol. 8 No. 13, pp.50-55, 2015

[2] P. Godfrey and J. Gryz, *"*Answering Queries by Semantic Cache*s", Database and Expert Systems Applications*, Florence, Italy, pp.485–498, 1999.

[3] A. Vancea and B. Stiller, "*Answering Queries Using Cooperative Semantic Caching*," In: *Proc. of IFIP International Conference on Autonomous Infrastructure, Management and Security*, pp 56-69, 2009.

[4] J. Colquhoun, P," *A Peer-to-Peer Server based on BitTorrent"*, Technical Report No. 1089, Newcastle University, pp. 1-28, 2008.

[5] M. J. Franklin and D. Srivastava, "Semantic Data Caching and replacement", *22th International Conference on VLDB*, pp. 330-341, 1996.

[6] Y. Tao, G. Liu, and J. Mottok, "*Georg Hagel Ranking task activity in teaching software engineering"*, IEEE Global Engineering Educations, pp. 1023 - 1027, 2016

[7] D S Guru,N Vinay Kumar," Novel feature ranking criteria for interval valued feature selection", *International Conference on Advances in Computing, Communications and Informatics*" , pp. 149 - 155, 2016 .

[8] J. Dean and S.Ghemawat, "*MapReduce: Simplified Data Processing on Large Clusters"* on To appear in OSDI 2004, pp1-31, Google, Inc.

[9] F. Li, B.C. Ooi, M. Tamer, and S. Wu, "Distributed Data Management Using

MapReduce", *ACM Computing Surveys*, Vol. 46, No. 3, pp. 31:1-31:42, 2014.

[10] J. Robinson, Universityof Essex, "*Data Analysis for Query Processing*, IDA report 1997.

[11] A. Motro, "*VAGUE A user interface to relational databases that permits vague queries",* ACM Transactions on Information Systems, Vol.6, No.3, pp. 187-214, 1988.

[12] H. R. Turtle and W. B. Croft, "Uncertainty in information retrieval systems", In: *Proc. of the Workshop on Uncertainty Management in Information Systems*, pp. 111-137, 1992.

[13] C. J. van Rijsbergen, *Information Retrieval*, Butterworths, pp. 21-30, press London, 1979.

[14] A. Motro, "*Integrity = validity + completeness* ", *ACM Transactions on Database Systems*, Vol.14, No. 4, pp. 480-502, 1989.

[15] R. Reiter. *On closed world data bases. In Logic and Databases*, pp 55-76. Press, 1978.

[16] G. Gottlob and R. Zicari, "Closed world assumption opened through null values", In: P*roc. of the 14th International Conference on Very Large Data Bases*, pp. 50-61, 1988.

[17] G. Salton and M. J. McGill, "*Introduction to Modern Information Retrieva*l", McGraw-Hill, NewYork, pp. 336-1337, 2004

[18] A. D'Atri, A. Motro, and L. Tarantino. "*ViewFinde*r", Technical report, George Mason University, pp.1-18, 1992.

[19] A. Motro, "BAROQUE: A browser for relational databases", *ACM23 on Information Systems*, Vol.4, No.2, pp. 164-181, 1986.

[20] T. R. Rogers and R. G. G. Cattell, "Object-oriented database user interfaces", *Technical report, Sun Microsystems*, 1987.

[21] P. Kalni and D. Papadias, "Multi-query optimization for on-line analytical processing", *Information Systems*, Vol. 28, No. 5, pp. 457-473, 2003.