International Journal of Advanced Technology in Engineering and Science
Vol. No.4, Issue No. 10, October 2016
www.ijates.com
ijates
ISSN 2348 - 7550

A SURVEY ON QUERY OPTIMIZATION IN CLOUD COMPUTING

Jaiveer Singh Rawat[1], Shyam Kishor[2], Madhu Kumari[3]

[1,2,3]*Computer Science, NIT Hamirpur, (India)*

ABSTRACT

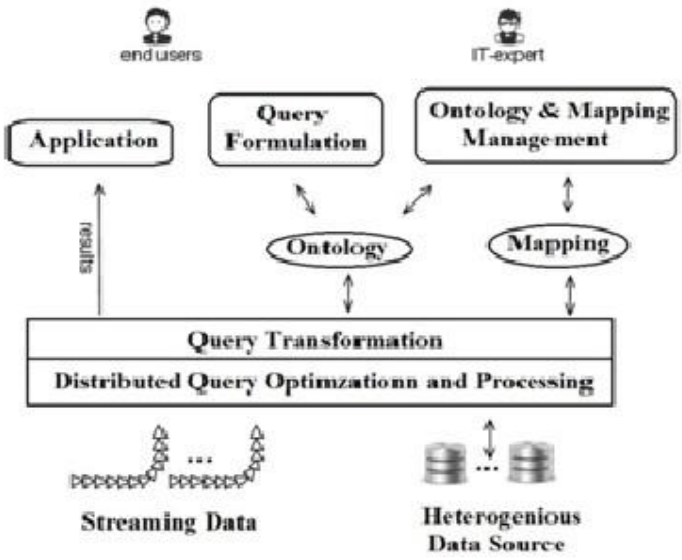
Multiple Query Optimization in the cloud has become a promising research direction due to popularity of cloud computing, which runs massive data analysis queries routinely. Many Cloud-based Distributed Data Processing platforms have been proposed to provide efficient and cost effective solutions for big data query processing, such as Hive, Hadoop etc. This paper presents a survey on query optimization based on their approaches.

Keywords Big data, Cloud Computing, Data Warehouse, Multi-Query Result Reuse, Query Optimization.

I. INTRODUCTION

The analysis of large collection of data is a routine activity in many commercial and academic organizations. Internet companies, for instance, collect massive amount of data such as content produced by web crawlers, service logs and click streams. Analyzing these data sets may require processing tens or hundreds of terabytes of data. To perform this task, many companies rely on highly distributed software systems running on large clusters of commodity machines. So there is a need of Query Optimization for analyzing the large amount of data in order to minimize the thousands of query.

Fig. 1 Cloud Query Processing



Query optimization is a method of many relational database management systems. The query optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans. Generally, the query optimizer cannot be used directly by users. Once queries are submitted to database server, and parsed

by the parser, they are then passed to the query optimizer where optimization occurs. However, some database engines allow guiding the query optimizer with hints. Query Optimization is a part of Query Processing. The Query Optimizer first generates a query plan. Each node in the query plan encapsulates a single operation that is required to execute the query. Then based on the query plan, the query optimizer generates an execution plan, defined as an ordering of the nodes in the query plan.

Query Optimization is well known problem in database research. It describes how to efficiently produce answers to a set of queries and becomes increasingly important to process SQL queries on cloud. In this Paper [1], the main aim was to achieve the interoperability through the integration of heterogeneous data sources using the Spatial On-line Analytical Processing. There are four layers in Spatial On-Line Analytical Processing: Internal layer, Designed layer, Operational layer and Display layer. In Internal layer, Data is fetched from the various data sources using the extract, transform and load process. In Designed layer, Spatial data structure is used to store the fetched data in DWs. Tools and users are connected in order to use spatial data in an efficient way. Data originator managed the data and metadata. Data warehouse schemas are constructed in this layer. The structure of the data warehouse is related to multidimensional data cube which provides the multidimensional view of the data. It allows pre-computation and fast accessing of the data. In Operational Layer, it performs the spatial On-line Analytical Process operations on data cubes by integrating both the spatial operations and multidimensional operations. Spatial operations and multidimensional operations are carried out of data of Spatial Data Warehouse with the help of Multidimensional data views and pre-computation of data. Multidimensional data view is provided by multidimensional operations. Spatial operations are functions that form important components for model which takes data related to location and then analyzes on it and then produces output information. These processes are known as Spatial On-Line Analytical Process. Display layer, this is the last layer where it displays the result of user request through interface, according to the requirements defined as the user interfaces. When user gives the request, firstly the request is processed and then displayed in the form map, web etc. Through this user can enhance the capacity to explore the underlined dataset only when the Spatial method is incorporated into On-Line Analytical processing.

The query optimization is carried out at compile time to reduce the encumbrance of optimization at run time, through this we can improve the performance of the code execution. We use the histogram [2] to get the assessments of selectivity of joins and predicate in a query. Based on those assessments we order query join and predicates in a query. The histograms are constructed using the frequency of objects called. The domain of the predicate is partitioned into intervals called windows. With the support of past queries the choosiness of predicate is derived with respect of its windows. The histogram approach would help us in the estimating the selectivity of a join and hence decide on order in which the joins have to be executed.

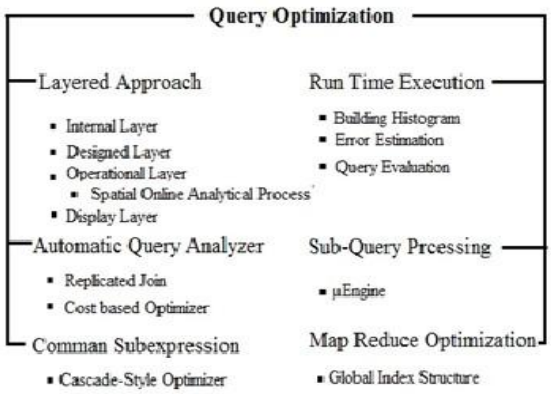
Data integration system (DIS) [3] assembles information from numerous remote sources, integrates and breaks down the information to get a query result. As Clouds/Grids are appropriated over wide-range systems, correspondence cost generally controls general query reaction time. In this way we can expect that query execution can be enhanced by minimizing correspondence communication cost. Disseminated information on incorporation of applications are constantly handled on dispersed bases, and correspondence cost gets to be the fundamental component of deciding query response time. Therefore we can expect that query execution can be enhanced by minimizing correspondence communication cost. The target of this paper was to propose a way to

enhance the query processing performance of information combination frameworks by improving sub-query processing.

Numerous investigation scripts are complex and contain common sub expressions, that is, halfway results that are consequently joined and collected in various distinctive ways execute a typical sub expression numerous times, once for every purchaser, which is obviously inefficient. The optimization procedure is reached out with another re-improvement stage that upholds physical properties at the mutual gatherings. The methodology accommodates contending physical prerequisites in a way that prompts an internationally ideal arrangement. The system was intended to be coordinated with analyzers that utilize the common Cascades model.

The global index [4] can be used to eliminate the unnecessary map tasks for the ranged queries so it can reduce the overhead of data I/O and task scheduling, which not only reduces the query response time but also optimizes the system resource utilization. In Hadoop, MapReduce consists of two phase: map phase and reduce phase. Firstly user submits a MapReduce Job through Hadoop scripts, including all the associated resource such as input, output, map function, configuration parameters, reduce function, libraries etc. Then Hadoop client divides the job into several tasks based on the distribution of input data to determine the total no of map tasks and each input.

Fig. 2 Query Optimization Background



In cloud computing fragmentation [5] is another approach for optimizing the queries. Here data is stored on the different physical system. Other than the data processing time, accessing the data requires inter-node communication time. To reduce the inter-nodes communication time we placed a Map on the various nodes leased by client. Each node uses its card to quickly identify the location of other fragments involved in the query. Each node will contain two things, firstly the architecture for dispersion of all the fragments in the cloud and secondly the fragmentation algorithm to generate a new fragmentation pattern which are built by frequent queries.

In COSMOS [6] if naive partition algorithm is used then it gives a straightforward solution which is to statically partition the query space into Query Reuse Unit. The basic idea of this algorithm was to decide the partition granularity of each partitioning column by analyzing the WHERE clauses among the query workload and divide the whole query space according to the given partition granularity. Then we can achieve the performance for result reusing but will suffer from the drawback of dimensionality i.e. too many queries will be generated. The cost of Query Reuse Unit management will increase.

II. RELATED WORK

In this paper, we have analyzed the methods applied to the Query Optimization. It comprises the theoretical analysis of the methods of Optimization. We have classified different Methodologies used in Query Optimization.

2.1 Spatial Query Optimization

Spatial On-Line Analytical Processing [1] performs operations on data cubes by integrating both the spatial operations and multidimensional operations. Spatial operations and multidimensional operations are carried out using data of Spatial Data Warehouse with the help of Multidimensional data views and Pre-Computation of data. Multidimensional data views are provided with multidimensional operations. Spatial operations are functions that form important components for model which takes data related to location and then analysis on it and produce output information. These processes are known as Spatial On-Line Analytical Process. The spatial query is processed in two steps, first is the filter step and other one is the refinement step, because of a large volume and high complexity of the spatial data. Here a query optimization strategy is discussed which takes the characteristics of SDBs into account. Select-merge rule of relational algebra optimization rules is used for combining refinement steps, and the Oid-intersection technique and the Oid-join technique is used for combining filter steps. Also they used the Spatial Object Algebra (SOA) to represent the input query and Intermediate Spatial Object Algebra (ISOA) to optimize the spatial queries. Implementation has been done on the sample database using OLapCube and Miner3D software. OlapCube analyze the data and create data cubes locally with .cub extension and Minor3D is used for visualization and create two types of 3D chart Scatter 3D and Bar3D. Some of the operations of SOA like UNION, PROJECT are still to be solved and no real data experiments have been carried out.

2.2 Selectivity Using Histograms

1) Building of Histogram: Histogram contains the occurrence of values assigned to different buckets. In numerical data we can assign some range and then assign the bucket accordingly. For the categorical data we have to partition the data into range with respect to letters.

2) Incremental Maintenance of Histograms: Here we find the estimation error for every attribute, if the error is greater than some threshold value then we need to update the histogram, otherwise we use the same old histogram to provide the selectivity estimate. The high frequency of occurrences are known as popular queries.

3) Method Outline for Error Estimation: Here we find the error estimate by using standard deviation between the old data value and the updated data value in the histogram buckets.4) Query Evaluation: Histogram is used to get the estimate for selectivity of the predicate query and joins.

5) Split and Merge algorithm: It helps us to reduce the cost of building and maintaining the histograms for the large tables. So estimating the selectivity of a join and predicates we get the join and predicate ordering at compile time.

According to this paper proposed technique reduces the burden of optimization at run time. This proposed work using histogram, get the estimates of selectivity of joins and predicates in query. Based on those estimates it orders query joins and predicates in a query. Then finally find the query plan at compile time.

2.3 Data Flow Style

In this approach [3] Data Integration System (DIS) uses a data flow style query execution mode. Runtime model of this data flow execution has four kinds of elements.

1) Query Plan: Contains a set of sub-queries formulated over the data sources and operators which specifies how to merge the result of sub-query.

2) Request: Group of operations are generated according to query plan and sent to the µEngine.

3) Dispatcher: Sends request to µEngine.

4) µEngine: Processes request.

In our strategy, DIS utilizes an information stream style query execution model. Every query arrangement is mapped to a group of µEngines, each of which is a project related to a specific administrator. Thus, multiple sub-queries from simultaneous questions can share µEngines. Therefore, all the sub-queries can acquire their outcomes, and general correspondence overhead can be lessened. Exploratory results demonstrate that, when DIS runs a group of parameterized queries, this remaking calculation can lessen the normal query culmination time by 32 to 48%, when DIS runs a group of non-parameterized queries, the normal query finishing time of questions can be diminished by 25 to 35%.

2.4 Automatic Query Analyzer

In this paper [7] AQUA (Automatic Query Analyzer) is proposed for Query optimization in MapReduce based on the Massively Parallel Processing. For every query AQUA generates the sequence of MapReduce jobs, which minimizes the cost query processing, Here AQUA adopts two phase of query optimizer, in the first phase users query is parsed into a join graph based on which we adaptively group the join operators. Each group may contain more the one join operator and one MapReduce job is generated for each group. In the second phase intermediate results of the groups are joined together to generate the final query results and then select the one that minimizes processing cost. AQUA has list of advantages. First, it used replicated join so that it reduces the number of MapReduce jobs and also avoids generating large volumes of intermediate results. Second, it adjusted the join sequence by using a cost based optimizer.

For implementation of AQUA, HIVE is used. The expression tree is forwarded to HIVE analyzer, which applies the metadata of tables to translate the tree into a set of MapReduce. It can implement the Shared Table Scan by modifying the MapReduce jobs generated by HIVE. Firstly in MapReduce job the job description is modified by replacing its key-value pairs to composite key-value pairs. Secondly two new operators are fulfilled for HIVE. One is designed for mappers to write back key-value pairs to HDFS and second is used in reducers to load key-value pairs from HDFS. For calculating the cost of the query, the cost model applies some pre computed histograms to estimate the selectivity of the joins and predicates. To build the histogram our native approach is to apply n MapReduce jobs, one for each column. In Map phase the partition tuples and table are scanned according to the value in specific column. In reduce phase, each reducer generates a cell for column histogram. To the histogram, it generates the composite key for each tuple in the map phase. In the reduce phase, it classifies key-value pairs by their column ID and combine the results from the multiple mappers. Then metadata of a histogram bucket are written back to HDFS. After that cost of the MapReduce jobs is evaluated. There are two types of MapReduce jobs: map-only jobs and Map-reduce jobs. In map-only jobs single table scan and map-side join are transformed into map-only jobs, each jobs will be processed by each mapper individually. In map-

reduce jobs, it is more costly than map-only jobs because here they used triggers sort operation at both the map and reduces sides. The cost model can also be configured for the query response time as the metric to measure how good the query plan is.

2.5 Extended Cascade Style Optimizer

In this paper [8] it is demonstrated to extend a Cascade-style optimizer to accurately enhance scripts containing common sub expression. The methodology has been prototyped in SCOPE, Microsoft's framework for monstrous information investigation.

These are the four steps in processing a query.

1) Step 1: Identifying normal sub expressions.

2) Step 2: Recording physical properties.

3) Step 3: Propagating data about shared gatherings and recognizing LCAs (least common ancestor group).

4) Step 4: Re-optimizing the query upholding physical properties.

The authors have expanded Microsoft's SCOPE stream- lining agent with the algorithms for misusing regular sub expressions. They ran the SCOPE enhancer on an Intel Dual Core 2GHz machine with 2GB RAM utilizing Microsoft Windows as working framework.

They prototyped the structure in SCOPE and the exploratory examination demonstrates that it lessened altogether (from 21 to 57%) the evaluated expense of basic and large real-world scripts.

2.6 Global Index

The main goal for Global index is the using of the global index structure to optimize the ranged queries. Priori global index knowledge and ranged queries are the two premises. However if not both conditions are satisfied then we should use the default procedure. The overall courses can be divided into four stages: query pattern analysis, opportunities of optimization detection, tasks division and task scheduling. In the query pattern analysis, when the user submits the MapReduce job, it will determine whether a user is performing a ranged query request, if it is not then we can't use the global index structure and then execution should be based on default process. In opportunities of optimization detection, here we can determine whether the submitted queries can be optimized or not. This process fully depends on global index structure. In task division based on global index, the main goal of using global index is to minimize the overhead of data I/O and task scheduling during the ranged query processing. To obtain this we can implement the customized task division before the MapReduce Jobs execute. The task division method is optimized for ranged queries which will cut down the number of map tasks conducting to minimize the overhead of I/O and map tasks scheduling. In task scheduling, this is the last step where the Hadoop expects for the task need to be scheduled for execution. By this we can reduce the response time of ranged queries and also improve the resource utilization.

2.7 Cluster Based Partition Algorithm

Cluster-based Partition Algorithm is to reduce the number of QRUs efficiently and guarantee the effectiveness of the partition result as far as possible. The basic idea behinds the CPA is to partition the partial query instead of the entire query by considering the query pattern and the locality of the data. The Cluster-based Partition

Algorithm can be described as four phases: Firstly, a partitioning columns selection strategy is needed to construct the entire query space. Secondly, we will classify and integrate the queries of workload according to their search range in the whole query space. Cold and hot are the two divisions of queries of workload. When the region contains lots of query, it means this region is hot region for now and vice versa, it is cold region. Thirdly we chose the set of hot regions as the candidate partial query for partitioning instead of partitioning the whole space. Then the number of Query Reuse Unit can be cut down by ignoring the less important region. And finally we adopt the naive algorithm to conduct the partition and ensure the partition result of each candidate partial query space can satisfy the query requirement.

2.8 Multi Query Reuse Dependence Graph:

Multi Query Reuse dependence graph is constructed after partition of the query space. Thus depicting the dependence between the multiple queries provides a basis for achieving multi-queries result reuse. The result reuse dependence graph is directed acyclic graph in which each node represents a query and the directed node between the two nodes represents the dependence between the two queries. Qi is used to denote the node I and the query Qi. Q0 refers the data source which is the root node of Multi Query Reuse Dependence Graph. An edge QiQj indicates that the data stream from Qi to Qj and share the similar query type. Suppose Sq denote the current query and we have to find the uplink (Qi) and downlink (Qi). Qi accepts the data streams from the nodes in uplink (Qi) and its query results will be reused by the nodes in the downlink (Qi). Each node announces its required reuse units in its uplink nodes and records where it is getting these units from. Suppose if a new query node Qi joins the graph then we search the query in Sq and connect Qi to the optimal queries which are able to provide maximum reuse utility but with minimal overloads.

2.9 Robust Heuristic Algorithms

An arrangement of powerful heuristic algorithms, Branch-and-Bound, Genetic, HillClimbing, and Hybrid Genetic-Hill Climbing, are proposed [9] to discover (close) ideal query execution plans and augment the advantages. They proposed four unique algorithms to improve the Cloud-MQO issue. The calculations are Branch-and-Bound (B and B), Genetic Algorithm (GA), Hill-climbing (HCA), and hybrid Genetic-Hill Climbing algorithm (Hybrid GHCA). B and B is a thorough algorithms that quests the entire arrangement space of the queries, others apply heuristic strategies to create and find (close) ideal arrangement of QPs. The tpc-h decision support benchmark database (10 GB) is utilized as a part of the investigations. The database has eight relations: Linetitem, Orders, PartSupp, Part, Customer, Supplier, Region, and Nation. Two criteria are utilized to assess the execution of the proposed calculations, optimization times and the quality of the solutions. Calculation of optimization times of the proposed algorithm B and B is seen to be the most tedious calculation. It gets to be restrictive when the quantity of queries is more than 7. B and B can give accurate answers for the question sets; notwithstanding, it is not a decent calculation for large number of queries that have numerous options due to its exponential advancement time. Optimization time of the genetic algorithms relies on its end condition. HCA proceeds with its optimization process until it can't locate any better arrangement (or gets stuck in a neighborhood ideal worth). Hybrid-GHCA incorporates both of the optimization times of GA and HCA because the beginning arrangements (each one in the population) are enhanced with GA. The improvement time of Hybrid-GHCA takes less time than beginning with irregular arrangements. Solution quality of the proposed

algorithms B and B calculation produces accurate arrangements; however, it can't advance issues with expansive pursuit space multifaceted nature. For every one of the situations, Hybrid-GHCA algorithm is seen to locate the best results. GA discovers arrangements that are for the most part the same or marginally more terrible (05%) than the arrangements of Hybrid-GHCA for issue sets with substantial multifaceted nature. HCA is great and quick for little issues; be that as it may, results found by HCA deteriorate when the query space is very complex. As for the advancement time and the nature of the arrangements, GA calculation is assessed to be the best performing calculation among the others.

2.1.1 Scalable Query Optimization

[10] Query Processing consists of two parts, query optimization and query generation. A query optimizer first generates the optimized query plan. It is the structure of deep left tree or bushy tree. Query optimization constructs the execution plan by breaking the optimal query plan into sequence of binary join operations so that in parallel system a multi way join must be broken down to binary join to execute, since each parallel job is able to execute only one binary job. Scalable Optimizer of SQL takes the SQL query as an input and generates the query execution plan that consists of sequence of MapReduce Job where the MapReduce Job can be binary or multi way join. Here for finding the optimal execution plan is NP-Hard. In this paper a polynomial algorithm is used to optimize the query execution plan. It will take the time complexity of O (n^2) where n is no of table in the SQL query. SQL query is represented by the join graph where vertex stands for table and edge stands for join between the two tables. We have to find the optimal way to break the join graph into set of sub-graph. For the sub-graph to be executed with minimal time, each sub-graph is assigned to a MapReduce Job. In the MapReduce framework, there are four types of join algorithm: Semi-join, repartition join, directed join, broadcast join. Repartition join supports the multi-way join that improve the efficiency of query processing.

III. CONCLUSION

Spatial query optimization is expected to be efficient in raster data and image, but some of the SOA operations like union and projection are still to be solved and need real data experiments. Using histogram we can make optimization plan at compile time rather than run time to reduce the execution time. In data flow style execution, data integration system uses a group of μEngines, processes sub query and reduces the communication cost. This approach is well suited for parameterized query as compared to non-parameterized query. Also data flow style execution model achieves 40% less compilation time of query as compared to iteration model. AQUA is MapReduce based query optimizer, generates sequence of MapReduce jobs which minimize the cost query processing. Extended cascades-style optimizer optimizes scripts, which contain common sub-expressions. This approach is already prototyped in SCOPE, Microsoft's system and gives plan with 21- 57% lower estimated costs. Global index based optimization strategy is well suited for range query and analysis but still evaluation in multi-user and multi-job is pending. Fragmentation approach is not validated under any tool, so performance cannot be evaluated. Multi-queries optimization framework based on MapReduce-oriented cloud environment utilizes the dependencies between multiple queries to realize query result reuse. This approach gives the 39.3% better performance as compared with Hive. A set of robust heuristic algorithms like Branch-and-Bound, Genetic, Hill Climbing and Hybrid Genetic-Hill Climbing can be used to generate optimal (near to) query execution plans to maximize benefits. This should be extended in MapReduce-based cloud environments using

hadoop and Hive. SOSQL is also an approach for query optimization and query execution and it is evaluated in Google Cloud Platform.

REFERENCES

[1] A. Tripathy, L. Mishra, and P. K. Patra, "A query optimization strategy for implementing multi-dimensional model in spatial database system," in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 2, pp. 64–68, IEEE, 2010.

[2] V. K. S. Nerella, S. Surapaneni, S. K. Madria, and T. Weigert, "Exploring query optimization in programming codes by reducing run-time execution," in Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual, pp. 407–412, IEEE, 2010.

[3] G. Chen, Y. Wu, J. Liu, G. Yang, and W. Zheng, "Optimization of sub-query processing in distributed data integration systems," Journal of Network and Computer Applications, vol. 34, no. 4, pp. 1035–1042, 2011.

[4] H. Zhao, S. Yang, Z. Chen, S. Jin, H. Yin, and L. Li, "Mapreduce model-based optimization of range queries," in Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on, pp. 2487– 2492, IEEE, 2012.

[5] A. Ettaoufik and M. Ouzzif, "Query's optimization in data warehouse on the cloud using fragmentation," in Next Generation Networks and Services (NGNS), 2014 Fifth International Conference on, pp. 145–148, IEEE, 2014.

[6] S. Wu, B. C. Ooi, and K.-L. Tan, "Continuous sampling for online aggregation over multiple queries," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 651– 662, ACM, 2010.

[7] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi, "Query optimization for massively parallel data processing," in Proceedings of the 2nd ACM Symposium on Cloud Computing, p. 12, ACM, 2011.

[8] Y. N. Silva, P.-A. Larson, and J. Zhou, "Exploiting common sub expressions for cloud query processing," in Data Engineering (ICDE), 2012 IEEE 28th International Conference on, pp. 1337–1348, IEEE, 2012.

[9] T. Dokeroglu, M. A. Bayir, and A. Cosar, "Robust heuristic algorithms for exploiting the common tasks of relational cloud database queries," Applied Soft Computing, vol. 30, pp. 72–82, 2015.

[10] Y. Shan and Y. Chen, "Scalable query optimization for efficient data processing using mapreduce," in Big Data (BigData Congress), 2015 IEEE International Congress on, pp. 649–652, IEEE, 2015.

[11] D. Ding, F. Dong, and J. Luo, "Multi-q: Multiple queries optimization based on mapreduce in cloud," in Advanced Cloud and Big Data (CBD), 2014 Second International Conference on, pp. 100–107, IEEE, 2014.

[12] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[13] H. Kllapi, D. Bilidas, I. Horrocks, Y. E. Ioannidis, E. Jimenez-Ruiz, E. Kharlamov, M. Koubarakis, and D. Zheleznyakov, "Distributed query processing on the cloud: the optique point of view (short paper)," in OWLED, Citeseer, 2013.