International Journal of Database Theory and Application
Vol.10, No.6 (2017), pp.1-12
http://dx.doi.org/10.14257/ijdta.2017.10.6.01

Query Optimization for Databases in Cloud Environment: A Survey

Archana Bachhav[1*], Vilas Kharat[2] and Madhukar Shelar[3]

[1]*Ph.D. Research Scholar, Computer Science, S.P. Pune University, Pune, India*
[2]*S.P. Pune University, Pune, India*
[3]*KTHM College, Nashik, India*
[1]*77.archana@gmail.com*, [2]*laddoo1@yahoo.com*, [3]*mnshelar70@gmail.com*

Abstract

Now days in the field of service oriented technologies cloud computing plays an important role. The main aim of cloud computing is to make people compute and store the resources easily and efficiently. Recent focus is deal with data expressing and searching. To improve the performance in the cloud requires the optimization of data processing time. Our study gives a comprehensive survey on numerous models and approaches used for query optimization to minimize execution time and to improve resource utilization. We have reviewed various research work done on query optimization for conventional SQL and MapReduce platforms.

Keywords*: cloud computing, conventional SQL, Map-Reduce, query optimization, service level agreement*

1. Introduction

In service oriented computing cloud computing is an extremely successful paradigm [1,6]. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the most popular services of cloud computing. Extension of this concept is Database as a Service (DBaaS) or Storage as a Service [9]. Cloud computing enhances sharing computing power and storage for number of database applications." The propagation seen in the number of applications which influenced various cloud platforms resulting in tremendous increase in the scale of the data generated as well as consumed by such applications "[2]. How to organize and manage those huge amount of data so as to get the useful information for the users has been the new theme in the research area of cloud computing." Cloud data modeling is the foundation of cloud computing application and the searching algorithm upon it is the key issue of cloud computing application" [22]. How to get the data timely, accurately and reliably; plays an important role in the success of the cloud data model [12].

In cloud computing platforms, resources should be acquired and released automatically and quickly at runtime to guarantee the SLA (Service Level Agreements) between customer and cloud service provider [11]. With clusters of virtual machines cloud computing enables users rent large amount of resources for short duration in order to run complex queries efficiently on large scale data [7]. The rent duration can be decreased more with the help of better query optimization technique [8]. Hence there is a need of investigating efficient query optimization techniques so as to reduce query evaluation time and response time. It will also enhance better utilization of computing resources in cloud.

"Query optimization leads to resource rent time optimization in cloud environment".

Query optimization techniques in centralized as well as in distributed platform are extensively researched in the corner of conventional SQL and MapReduce technique. In our work we have navigated this area of research and analyzed different techniques of

ISSN: 2005-4270 IJDTA

query optimization. In remainder of the paper Section II describes the concepts of conventional SQL and MapReduce platforms. Section III represents our analysis of various techniques for query optimization after reviewing various research papers. Finally, our concluding remarks are given in Section IV.

2. Overview of Traditional SQL and MapReduce

RDBMS is a relational Database Management System in which data is stored in the form of tables. Every table has columns and rows The Structured Query Language (SQL) issued to retrieve necessary information from the stored data. Column entries can refer for another table so as to create relationships among them. The tables and relationships can be manipulated using join operation of SQL as shown in Figure 1.

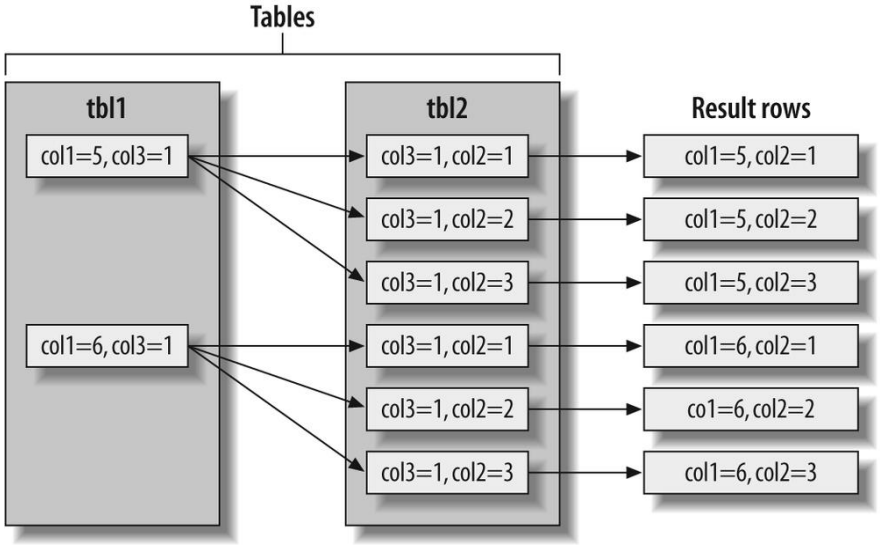

Figure 1. Query Processing Using Traditional SQL [42]

For large scale data it becomes cumbersome to manage it using traditional database systems. Now days Hadoop has become a solution to this problem. The applications which need high performance can be supported by Hadoop as it is scalable. Hadoop framework uses Distributed File Systems (DFS) for storing data. MapReduce is a new data processing technique which was proposed by Google. It is used to handle large scale data analysis jobs. It operates on the top of Distributed File systems (DFS). Data in DFS is partitioned into equal size chunks which facilitates parallel data processing.

As shown in Figure 2, in MapReduce parallel programs are split into two phases map and reduce. In map phase each mapper loads a data chunk from DFS and transforms it into a list of key-value pairs. The key value pairs are stored into n local files where n is no of reducers. In reduce phase files from different mappers are combined together for the values with same keys, a user defined processing function is applied by the reducer and new key-value pair is generated as the result. Finally, results are written back DFS. “The main challenge of query processing based on MapReduce is how to process join.” [21].

The implementation of traditional query optimization strategies in cloud platforms can have a poor performance, because they cannot predict future availability and release of resources [15]. MapReduce platforms that are used in cloud databases suffer from processing cost for join-intensive queries. Compare with MapReduce, SQL query has stronger expression ability such as join [19].

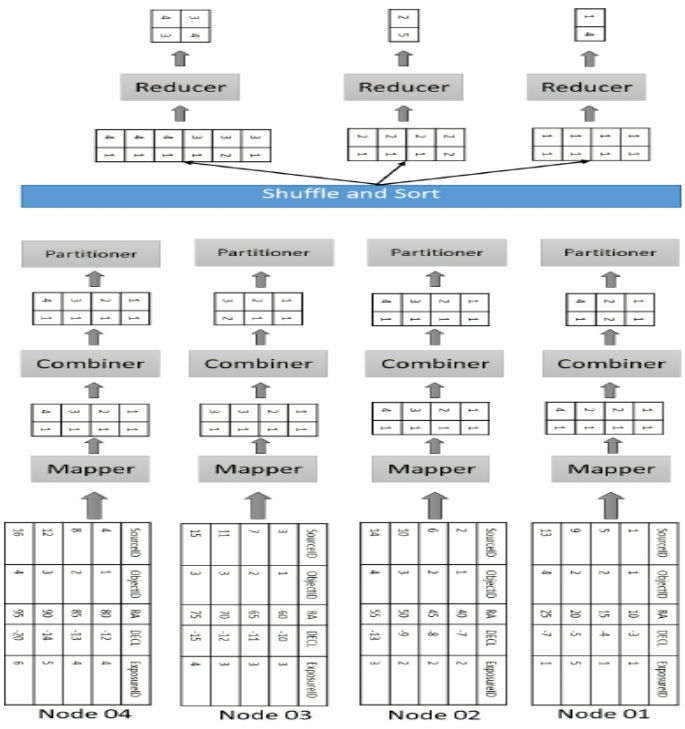

Figure 2. Query Processing Using MapReduce [43]

3. Navigation of Query Optimization Techniques

Either use centralized or distributed platforms with traditional SQL or MapReduce technique query optimization is obligatory to enhance high performance in large scale data systems. Query optimization is carried out in two phases in first phase search space is generated and in second phase an optimal plan from the search space is selected [8]. Numerous approaches are discovered by the researchers which are flavored with one or more following query optimization techniques.

- Elimination of Redundant Evaluation
- Continuous or Iterative processing
- Catching Intermediate Queries or Results
- Materialization
- Pipelining

Some of these techniques work on reducing communication cost, some focus on reducing execution time of the query and some focus on utilization of system resources properly in cloud environment.

3.1. Elimination of Redundant Evaluation

Complex queries consist of common sub-expressions. If they are evaluated only once it will reduce evaluation time of queries. An IGNITE system is developed by Lee R *et al.* [14] based on start-fetch wrapper with request window mechanism. It only sends the common sub-queries to the same data source so that redundant answers among the sub-queries can be eliminated. It detects for data sharing chances across concurrent distributed queries thereby reduces communication overhead but increases system throughput. Source wrappers in which multiple queries consist of common sub-queries are wrapped together, are decoupled by IGNITE and the execution engine sends sub-queries to the same source which enhance data sharing among sub-queries. Chen G *et al.* [5] also presented the research work based on IGNITE system; however it reduces the communication traffic which is increased in IGNITE by reconstructing original sub-

queries to alternative sub-queries. This system uses data integration system based on an operator-centric data flow execution model. Here µengine corresponds to every operator. By routing data through µengines queries are evaluated. As all these µengines work in parallel thereby taking advantage of parallelism. All similar query plans are allocated to the same group of µengines, sub-queries of different queries are placed together for processing in order to share data across the sub-queries. µengine processes the sub-queries after they are reconstructed with Merge-Partition algorithm. This reconstruction eliminates data redundancy. "It may not eliminate all redundant answers but never introduces unnecessary data as like in IGNITE. This method takes advantage of parallel sub-query processing whereas IGNITE cannot" [5]. Here new queries are evaluated by decreasing communication overhead. Dokeroglu *et al.,* [7] has built and used efficient sets of query execution plans. The system is based on evaluation of common tasks only once. Garofalakis *et al.,* [25] has proposed the resource usage models in which parallel query processing systems perform multiple queries scheduling to reduce the response time of queries. Decomposition of a query into parallel subtasks and rearrangement of the execution plan maximizes the reuse of cached data [26]. Query is divided into sub-queries that can be executed in parallel on many processors and reuse of already computed sub-query results improves the processing speed of new queries [27].

Alternative plans are not generated in above works. Dokeroglu *et al.* [24] has proposed a system in which alternative query plans are generated. Robust heuristic algorithms *e.g.* branch-and-bound algorithms, hill climbing, genetic hill-climbing algorithms etc are used to search optimal query execution plans and maximize benefits. After the extensive experiment authors analyzed that Branch & Bound algorithm produces exact solutions but cannot optimize problems with large search space complexity. Hybrid Genetic Hill Climbing algorithm is observed to find best results in all environments. Genetic Algorithm produces good results but slightly worse than the solution of Hybrid Genetic Hill Climbing algorithm for problem sets with complex search space. Hill Climbing Algorithm is good and fast for small problems. Genetic Algorithm is evaluated to be the best among others. The system is based on RDBMS. Authors in their future work are going to implement the system for MapReduce based cloud environments using Hadoop and Hive. Traditional distributed query engine processes a set of queries with best query plan. It executes that query plan by using cached results at different sites. On the other hand non optimal query plans may result in a smaller total execution time, if common sub expressions are evaluated only once. Giannikis *et al.* [28] developed a database architecture which is incorporated with batching queries and sharing computation across concurrent queries in a shared disk, shared L3 cache, multi-core and multiprocessor machine. In MRShare the different queries having common work to perform are grouped and treated as single job. That single job is executed once to optimize I/O [44]. [45] Proposed a technique in which different sharable jobs are rescheduled in order to access a set of files simultaneously. Aim accomplished by the system is to maximize the rate of processing. Authors proposed scheduling policies in which non-sharable scans are schedule after the sharable I/O work with future tasks. Silva *et al.* presents a cost based optimization of complex queries which shares common sub expressions. In this system local query plan is optimized that causes optimization of global plan. Using Incoop MapReduce jobs are executed in incremental manner. It observes changes to input and automatically change in the output is reflected [46].

3.2. Continuous or Iterative Processing

Continuous query optimization method is different from traditional one, which focuses on collection and propagation of data statistics before query execution. Bruno N *et al.* [4] has proposed a technique that continuously monitors query execution, collect actual runtime statistics and adapts execution plans as the query executes. The query optimizer is triggered whenever new runtime statistics become available. If a better plan is found, the

proposed technique intelligently adapt the current execution plan with minimal changes. Optimization techniques are based on accurate data statistics to select the best execution plan. But it becomes difficult to compute a better quality statistics on intermediate computation results. To deal with this problem Cole and Grafe [29, 32] proposed generating multiple plans at compile time. Among these plans one is selected to execute, when unknown quantities are discovered at run time using decision tree mechanism. Decision for how many plans are to be generated and stored at compile time is difficult.

Progressing query optimization (POP) detects cardinality estimation errors in mid execution. It compares the estimated cardinality values against the actual run time counts. If inconsistencies occur, the re-optimization of current plan is triggered. POP is designed for traditional centralized database systems [30, 33]. DB2L Earning optimizer waits until a query plan finishes execution. Then it compares actual row counts to the optimizers' estimates. Misestimates are used to learn and adjust the statics to improve the quality of optimizations in future queries [34]. These ideas are extended by Bruno N *et al.* The stubby systems [31] propose a cost based optimizer for MapReduce systems. It collects job statistics while job is running and uses it to optimize similar jobs later on. All these approaches focus on to improve future query executions using feedback, but not addressed improving query performance in mid execution which is addressed in the system proposed by Bruno N *et al.,* [4] in a distributed environment. Rankloud computes run time statistics that is used to derive the lowest score for top-k results [48]. Starfish system consists of two components, Profiler and What-if Engine. Profiler collects statistics about the size of data processed, usage of resources, time to execute each job. Using these parameter values What-if calculates benefits [52]. RoPE also collects statistics from running jobs and utilizes it for re-optimizing future execution of the same job [53].

3.3. Catching Intermediate Queries or Results

Many researchers also worked on query optimization by caching intermediate results in one sliding window [17]. Related work is presented by Safaeei A *et al.* [18] that optimize overlapping queries with common sub-expressions using multiple sliding windows. Executing and caching the shared sub-queries at the sites with the lowest communication cost and shipping the output to the sites that need them as input are important factors that improve the total execution time [24]. EARL supports for processing part of input data instead of entire data by using early results for the analytical queries in MapReduce. By uniform sampling and working iteratively EARL computes larger samples until desired accuracy reached [47]. BlinkDB retrieves approximate results based on samples which are computed earlier. Accuracy of the result depends upon the quality of samples. It is designed for interactive query processing on large volumes of data [55]. Shark is the system that uses a shared memory concept where inter query data cached in memory. It is accessed directly from memory instead of accessing from disk. It reduces I/O and if accompanied with Hive then increases its performance [54].

However, instead of maintaining intermediate results of overlapped queries, Theeten B *et al.* [20] proposed the CHive method for query optimization based on evaluating continuous queries in distributed clouds. Continuous query is a query that is repeatedly re-evaluated as new data comes in. It operates on data stream rather than previously stored database table. Yahoo developed a system NOVA which is incorporated with incremental processing of continuously arriving data. NOVA works on the top of the Pig and Hadoop [50]. CBP (Continuous Bulk Processing) is a system that is batch query processing. Here state is maintained during batch processing and work done previously is reused. It reduces data flow in the system [56]. REX is parallel query processing system in which incremental changes in state are used instead of using the state which is produced by earlier iteration whereas CBP uses state which is produced earlier. It increases efficiency of iterative processing [51]. DBaaS can provide a service to cloud users that willing to

receive results with weaker quality in exchange of lower cost. This feature can be provided through the use of new runtime partial result optimizer by modifying the execution of pipelines and operators to manage the balance between cost and quality [13].

3.4. Materialization

Materialized views can be used to improve the query processing time for aggregation query over large relations [10]. Success of materialization technique depends upon efficient view matching algorithm. Goldstein and Larson [10] devised a technique which is used for matching materialized view derived from Select-Project-Join queries. WATCHMAN [23] and DynaMat [35] systems integrated with the policies those protect the cache from storing large and unpromising tables so as to increase the hit ratio of its content. In architecture presented by Ivanova *et al.* [36] physical operations are matched with the entries in the cache during runtime which eliminates the need for changes to the query optimizer. ReStore [37] reuses the results of MapReduce jobs which are described in analytical query languages like Pig [38]. The output results of MapReduce jobs are maintained to recognize reuse chances by future jobs. Jonathan proposed an algorithm for determining whether part or all of a query can be computed from materialized views [31].

History aware query optimizer [16] is based on archiving intermediate results. It is capable of matching views which are the byproducts of previously executed queries. It uses those views for generating alternative execution plans and improves the query execution time. Prior evaluation of batch workloads in Cache-on-Demand [39] and MQT technique [40] determine elements to be cached for maximizing re-usage. In Cache investment framework [41] continuous analysis of historical information enables to identify relations which could be useful for evaluation of future queries.

3.5. Pipelining

In cloud environment using MapReduce, overall data processing cost for join-intensive workloads increases. Anyanwu K *et al.* [3] has introduced Nested Triple Group Data Model and Algebra (NTGA) that minimizes overall processing cost by reducing the number of MapReduce cycles. Hive, MapReduce based system incurs the cost of saving intermediate results. Task is divided into multiple jobs. One job reads the result of the previous job to continue processing. AQUA (Automatic Query Analyzer) designed for MapReduce reduces generation of too many intermediate results so as to decrease storage and network cost. It adopts 2-phase optimizer - phase 1 form groups of join operators for reducing total number of MapReduce jobs in order to evaluate the query. Phase 2 joins intermediate results of group to generate final query results [21]. MapReduce online is the system that uses pipelining of intermediate data from map tasks to reduce tasks instead of using materialization [49].

Following table shows the classification of approaches which we have studied in our survey according to their incorporated optimization techniques.

Table 1. Classification of Query Optimization Approaches

Approach	Materialization	Pipelining	Elimination of redundant evaluation	Continuous /Iterative processing	Caching intermediate queries/ results
[3]		✓			
[4]				✓	
[5]			✓		
[7]			✓		
[10]	✓				
[13]					✓
[14]			✓		
[16]	✓				
[17]					✓
[18]					✓
[20]					✓
[21]		✓			
[23]	✓				
[24]			✓		✓
[25]			✓		
[26]			✓		
[27]			✓		
[28]			✓		
[29]				✓	
[30]				✓	
[31]	✓			✓	
[32]				✓	
[33]				✓	
[34]				✓	
[35]	✓				
[36]	✓				
[37]	✓				
[38]	✓				
[39]	✓				
[40]	✓				
[41]	✓				
[44]			✓		
[45]			✓		
[46]			✓		
[47]					✓
[48]				✓	
[49]		✓			
[50]					✓
[51]					✓
[52]				✓	
[53]				✓	
[54]					✓
[55]					✓
[56]					✓

3. Conclusion and Future Work

In the presented survey, we have navigated the various approaches in centralized as well as distributed platform of query optimization based on conventional SQL and MapReduce technique. In cloud computing platforms, there should be an autonomic solution to acquire and release resources at runtime to provide fail-safe service to the customer. However, traditional query optimization strategies cannot forecast future availability and release of resources; hence it may suffer from poor performance as compare to MapReduce strategy. MapReduce may results in more processing cost regarding to join intensive queries. There is a future need for developing efficient techniques to bridge the gap between SQL query and MapReduce in cloud environment by strengthening the query optimization ability of SQL or join intensive ability of MapReduce.

References

[1] D. J. Abadi, "Data Management in the Cloud: Limitations and Opportunities", IEEE Data Eng. Bull., vol. 32, no. 1, **(2009)**, pp. 3-12.
[2] D. Agrawal, S. Das and A. E. Abbadi, "Big data and cloud computing: current state and future opportunities", In Proceedings of the 14th International Conference on Extending Database Technology, ACM, **(2011)**, pp. 530-533.
[3] K. Anyanwu, H. Kim and P. Ravindra, "Algebraic Optimization for Processing Graph Pattern Queries in the Cloud", Internet Computing, IEEE, vol. 17, no. 2, **(2013)**, pp. 52-61.
[4] N. Bruno, S. Jain and J. Zhou, "Continuous cloud-scale query optimization and processing", Proceedings of the VLDB Endowment, vol. 6, no. 11, **(2013)**, pp. 961-972.
[5] 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, **(2011)**, pp. 1035-1042.
[6] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati, "Integrity for join queries in the cloud", Cloud Computing, IEEE Transactions on, vol. 1, no. 2, **(2013)**, pp. 187-200.
[7] T. Dokeroglu, S. A. Sert and M. S. Cinar, "Evolutionary multi-objective query workload optimization of Cloud data warehouses", The Scientific World Journal, **(2014)**.
[8] P. Doshi and V. Raisinghani, "Review of dynamic query optimization strategies in distributed database", In Electronics Computer Technology (ICECT), 2011 3rd International Conference on IEEE, vol. 6, **(2011)**, pp. 145-149.
[9] 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 IEEE, **(2014)**, pp. 145-148.
[10] J. Goldstein and P. A. Larson, "Optimizing queries using materialized views: a practical, scalable solution", In ACM SIGMOD Record, ACM, vol. 30, no. 2, **(2001)**, pp. 331-342.
[11] H. Hacigümüs, J. Tatemura, W. P. Hsiung, H. J. Moon, O. Po, A. Sawires and H. Jafarpour, "CloudDB: One size fits all revived", In Services (SERVICES-1), 2010 6th World Congress on IEEE, **(2010)**, pp. 148-149.
[12] H. Kllapi, E. Sitaridi, M. M. Tsangaris and Y. Ioannidis, "Schedule optimization for data processing flows on the cloud", In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM, **(2011)**, pp. 289-300.
[13] W. Lang, R. V. Nehme and I. Rae, "Database optimization for the cloud: Where costs, partial results, and consumer choice meet", ACM, CIDR, **(2015)**.
[14] R. Lee, M. Zhou and H. Liao, "Request Window: an approach to improve throughput of RDBMS-based data integration system by utilizing data sharing across concurrent distributed queries", In Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment, **(2007)**, pp. 1219-1230.
[15] C. M. Costa and A. L. Sousa, "Adaptive Query Processing in Cloud Database Systems", In Cloud and Green Computing (CGC), 2013 Third International Conference on IEEE, **(2013)**, pp. 201-202.
[16] L. L. Perez and C. M. Jermaine, "History-aware query optimization with materialized intermediate views", In Data Engineering (ICDE), 2014 IEEE 30th International Conference on IEEE, **(2014)** March, pp. 520-531.
[17] P. Roy, S. Seshadri, S. Sudarshan and S. Bhobe, "Efficient and extensible algorithms for multi query optimization", In ACM SIGMOD Record, ACM, vol. 29, no. 2, **(2000)** May, pp. 249-260.
[18] A. A. Safaeei, M. Kamali, M. S. Haghjoo and K. Izadi, "Caching intermediate results for multiple-query optimization", In Computer Systems and Applications, 2007. AICCSA'07. IEEE/ACS International Conference on IEEE, **(2007)** May, pp. 412-415.

[19] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo and A. Rasin, "MapReduce and parallel DBMSs: friends or foes?", Communications of the ACM, vol. 53, no. 1, **(2010)**, pp. 64-71.
[20] B. Theeten and N. Janssens, "CHive: Bandwidth Optimized Continuous Querying in Distributed Clouds", Cloud [1] Abadi, D. J. Data Management in the Cloud: Limitations and Opportunities, IEEE Data Eng. Bull., vol. 33, no. 2, **(2009)**, pp. 3-12.
[21] 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, ACM, **(2011)** October, pp. 12.
[22] L. Zhou, K. He, X. Sheng and B. Wang, "A survey of data management system for cloud computing: models and searching methods", Research Journal of Applied Sciences, Engineering and Technology, vol. 6, no. 2, pp. 244-248.
[23] P. Scheuermann, J. Shim and R. Vingralek, "Watchman: A data warehouse intelligent cache manager", **(1996)**.
[24] 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, **(2015)**, pp. 72-82.
[25] M. N. Garofalakis and Y. E. Ioannidis, "Multi-dimensional resource scheduling for parallel queries", In ACM SIGMOD Record, ACM, vol. 25, no. 2, **(1996)** June, pp. 365-376.
[26] A. Sinha and C. Chase, "Prefetching and caching for query scheduling in a special class of distributed applications", In Parallel Processing, Software., Proceedings of the 1996 International Conference on IEEE, vol. 3, **(1996)** August, pp. 95-102.
[27] H. Andrade, T. Kurc, A. Sussman and J. Saltz, "Multiple query optimization for data analysis applications on clusters of SMPs", In Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on IEEE, **(2002)** May, pp. 154-154.
[28] G. Giannikis, G. Alonso and D. Kossmann, "SharedDB: killing one thousand queries with one stone", Proceedings of the VLDB Endowment, vol. 5, no. 6, pp. 526-537.
[29] R. L. Cole and G. Graefe, "Optimization of dynamic query evaluation plans", ACM, vol. 23, no. 2, **(1994)**, pp. 150-160.
[30] N. Kabra and D. J. DeWitt, "Efficient mid-query re-optimization of sub-optimal query execution plans", In ACM SIGMOD Record, ACM, vol. 27, no. 2, **(1998)** June, pp. 106-117.
[31] H. Lim, H. Herodotou and S. Babu, "Stubby: A transformation-based optimizer for mapreduce workflows", Proceedings of the VLDB Endowment, vol. 5, no. 11, **(2012)**, pp. 1196-1207.
[32] G. Graefe and K. Ward, "Dynamic query evaluation plans", In ACM SIGMOD Record, ACM, vol. 18, no. 2, **(1989)** June, pp. 358-366.
[33] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh and M. Cilimdzic, "Robust query processing through progressive optimization", Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM, **(2004)** June, pp. 659-670.
[34] M. Stillger, G. M. Lohman, V. Markl and M. Kandil, "LEO-DB2's learning optimizer", In VLDB, vol. 1, **(2001)** September, pp. 19-28.
[35] Y. Kotidis and N. Roussopoulos, "DynaMat: a dynamic view management system for data warehouses", In ACM SIGMOD Record, ACM, vol. 28, no. 2, **(1999)** June, pp. 371-382.
[36] M. G. Ivanova, M. L. Kersten, N. J. Nes and R. A. Gonçalves, "An architecture for recycling intermediates in a column-store", ACM Transactions on Database Systems (TODS), vol. 35, no. 4, **(2010)**, pp. 24.
[37] I. Elghandour and A. Aboulnaga, "ReStore: reusing results of MapReduce jobs", Proceedings of the VLDB Endowment, vol. 5, no. 6, **(2012)**, pp. 586-597.
[38] C. Olston, B. Reed, U. Srivastava, R. Kumar and A. Tomkins, "Pig latin: a not-so-foreign language for data processing", In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, **(2008)** June, pp. 1099-1110.
[39] K. L. Tan, S. T. Goh and B. C. Ooi, "Cache-on-demand: Recycling with certainty", In Data Engineering, 2001, Proceedings. 17th International Conference on IEEE, **(2001)**, pp. 633-640.
[40] T. Phan and W. S. Li, "Dynamic materialization of query views for data warehouse workloads", In Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on IEEE, **(2008)** April, pp. 436-445.
[41] D. Kossmann, M. J. Franklin, G. Drasch and W. Ag, "Cache investment: integrating query optimization and distributed data placement", ACM Transactions on Database Systems (TODS), vol. 25, no. 4, **(2000)**, pp. 517-558.
[42] S. Baron, Z. Peter, T. Vadim, D. Z. Jeremy, L. Arjen and J. B. Derek, "High Performance MySQL", **(2008)**.
[43] A. Mesmoudi, M. S. Hacid and F. Toumani, "Benchmarking SQL on MapReduce systems using large astronomy databases", Distributed and Parallel Databases, vol. 34, no. 3, **(2016)**, pp. 347-378.
[44] T. Nykiel, M. Potamias, C. Mishra, G. Kollios and N. Koudas, "MRShare: sharing across multiple queries in MapReduce", Proceedings of the VLDB Endowment, vol. 3, no. 1-2, **(2010)**, pp. 494-505.
[45] P. Agrawal, D. Kifer and C. Olston, "Scheduling shared scans of large data files", Proceedings of the VLDB Endowment, vol. 1, no. 1, **(2008)**, pp. 958-969.
[46] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar and R. Pasquin, "Incoop: MapReduce for incremental computations", In Proceedings of the 2nd ACM Symposium on Cloud Computing, ACM, **(2011)**, p. 7.

[47] N. Laptev, K. Zeng and C. Zaniolo, "Early accurate results for advanced analytics on mapreduce", Proceedings of the VLDB Endowment, vol. 5, no. 10, **(2012)**, pp. 1028-1039.
[48] K. S. Candan, J. Kim, P. Nagarkar, M. Nagendra and R. Yu, "RanKloud: scalable multimedia data processing in server clusters", IEEE MultiMedia, vol. 18, no. 1, **(2011)**, pp. 64-77.
[49] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy and R. Sears, "MapReduce online", In Nsdi, vol. 10, no. 4, **(2010)**, pp. 20.
[50] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson and C. Tian, "Nova: continuous pig/hadoop workflows", In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM, **(2011)**, pp. 1081-1090.
[51] S. R. Mihaylov, Z. G. Ives and S. Guha, "REX: recursive, delta-based data-centric computation", Proceedings of the VLDB Endowment, vol. 5, no. 11, **(2012)**, pp. 1280-1291.
[52] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of mapreduce programs", Proceedings of the VLDB Endowment, vol. 4, no. 11, **(2011)**, pp. 1111-1122.
[53] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing", In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association, **(2012)**, pp. 2-2.
[54] C. Engle, A. Lupher, R. Xin, M. Zaharia, M. J. Franklin, S. Shenker and I. Stoica, "Shark: fast data analysis using coarse-grained distributed memory", In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, **(2012)**, pp. 689-692.
[55] S. Agarwal, A. P. Iyer, A. Panda, S. Madden, B. Mozafari and I. Stoica, "Blink and it's done: interactive queries on very large data", Proceedings of the VLDB Endowment, vol. 5, no. 12, **(2012)**, pp. 1902-1905.
[56] D. Logothetis, C. Olston, B. Reed, K. C. Webb and K. Yocum, "Stateful bulk processing for incremental analytics", In Proceedings of the 1st ACM symposium on Cloud computing, ACM, **(2010)**, pp. 51-62.

Authors

Archana S Bachhav, completed Master of Computer Science degree from Savitribai Phule Pune University, formerly known as University of Pune, India in 2001 and M.Phil in Computer Science from Yashwantrao Chavan Maharashtra Open Univesity(YCMOU) in 2009. Archana has qualified National Eligibility Test (N.E.T.) and State Eligibility Test (S.E.T.) for Lectureship. She is Pursuing Ph.D. in Computer Science at Department of Computer Science, Savitribai Phule Pune University. Currently, she is working as Assistant Professor in Computer Science Department, CMCS College, Nashik. Archana has more than 10 years of teaching experience for undergraduate and postgraduate programmes in Computer Science and Information Technology. Her research interest includes Query Optimization in Cloud Environment and Distributed Database Management Systems.

Vilas S Kharat, graduated with Physics, Chemistry and Mathematics, subsequently did his Master of Science from the Dr.B. A. Ambedkar Marathwada University, Aurangabad and Doctor of Philosophy from the University of Pune. During late Eighties he joined as a lecturer and in mid Nineties University of Pune, became full Professor in 2005. He is recipient of four consecutive best research papers awards by Indian Mathematical Society for the years 1998, 1999, 2000 and 2001. Professor Kharat has published number of research papers in different international journals. Five students have obtained their Ph.D. degrees and work of seven is in progress and importantly he has to his credit a partial solution to a very famous Frankl's Conjecture in collaboration with his research student. Kharat is also member of various learned societies including IEEE, American Mathematical Society member of various academic bodies

which includes Board of studies, Research and Recognition Committee, Faculty of Science of various universities.Ph.D.

Madhukar N Shelar, received Master of Computer Science degree from Savitribai Phule Pune University, formerly known as University of Pune, India in 1993 and also qualified State Eligibility Test (S.E.T.) for Lectureship. He is Pursuing Ph.D. in Computer Science at Department of Computer Science, Savitribai Phule Pune University. Currently, he is working as a Head of Computer Science Department, KTHM College, Nashik. Madhukar has more than 20 years of teaching experience for undergraduate and postgraduate programmes in Computer Science and Information Technology and he has been member of Board of Studies in Computer Science and Faculty of Science at Savitribai Phule Pune University. He has authored five books includes System Programming and Operating System, OOP and Java Programming. He has successfully completed the research work funded by Savitribai Phule Pune University under minor research scheme and also published research papers at conferences. His research interest includes Cloud Computing, Operating System and Distributed Database Management Systems.

