Continuous Weighted-Sum based Multi-Objective Query Optimization on Mobile Cloud DataBase

Chenxiao Wang[1] Florian Helff[1] Merrien Maxime P[1] Jason Arenson[1] Le Gruenwald[1] Laurent d'Orazio[2]

[1] School of Computer Science
University of Oklahoma
Norman, Oklahoma, USA
{fhelff, chenxiao, maxime.p.merrien-1, arensonjt, ggruenwald}@ou.edu

[2] CNRS, UMR 6074, IRISA
Rennes 1 University
Lannion, France
laurent.dorazio@univ-rennes1.fr

ABSTRACT

Optimizing queries on mobile cloud databases has to consider several objectives simultaneously, such as query execution time and monetary cost occurred on the mobile device to execute queries. There exist multi-objective query optimization techniques for cloud databases, but none deals with multi-objective query re-optimization that dynamically modifies the query plan to incorporate new statistics gathered during the query execution for performance improvement. Besides that, the existing techniques require user interaction during the query optimization process which produces extra cost. To fill this gap, in this paper, we vision an approach to optimize a query plan based on time and monetary cost and re-optimize the query plan during the execution without user interaction. In our approach, we first find a qualified query according to the objective preference set by the user before the query optimization process and adjust the query plan by using the actual running statistics collected during the execution.

1. INTRODUCTION

In a mobile-cloud database environment, a user issues queries from a mobile device to obtain data stored on the cloud . In order to execute a query, three different costs occur: the monetary cost of query execution on the cloud, the overall query execution time, and the energy consumption on the mobile device where the query might be executed. These three costs constitute the three cost objectives that the query optimizer needs to minimize in order to choose the optimal query execution plan (QEP). In traditional database systems, the query optimizer chooses the query plan with the minimum cost and the overall objective of the optimization is to minimize the query response time. However, mobile-cloud database systems are provided to users as on-demand services where users are charged for the actual usage of the services. The users do not need to pay a large amount of money to purchase the infrastructure to build the database system, but they pay a small amount of money by the time period of using the system. For example, Amazon Web Service (AWS) charges their users $0.0065 per hour for renting a certain type of servers [3]. The user then considers not only if the query response time is satisfied, but also whether the execution of the query is within the budget. Due to the elasticity of hardware in the mobile-cloud database system, the query optimizer should consider the monetary cost in addition to the query response time when deciding which QEP should be chosen to be executed in order to deliver the query result within the user's response time and budget constraints. This decision process under the traditional interaction model usually requires user's input of their objective preference before the right QEP is chosen. However, the process is slowed down on user's pending inputs.

There existing some techniques that optimize the query processing based on multi-objectives , but some of them ignore the elasticity of the hardware [5] which is essential in a cloud environment and some of them made assumption or require user's input [1] [6] [7] about the preferences of the objectives and users are not allowed to change these preferences. In our approach, the searching algorithm enable searching the optimal QEP on all the combination of containers and we introduce a new interaction model that allows user's preference be preset before the optimization process so that any user's pending input during the processing is not necessary.

2. CONTINUOUS WEIGHTED-SUM BASED MULTI-OBJECTIVE QUERY OPTIMIZATION

An example architecture of a mobile-cloud database environment is shown in Figure 1. The users use mobile devices to obtain data. This data is either stored in the cloud or retrieved from a cache on the mobile devices.
In our previous work, to improve the interaction model of the multi-objective query optimization, we presented the Normalized Weighted Sum Algorithm (NWSA) [10]. This algorithm allows the user to make a decision on the QEP based on multi-objectives but does not burden the user with the task of selecting a QEP out of a huge amount of options.

For example, suppose that a medical doctor would like to retrieve a patient's information from our database system. A typical following query is issued:

```
select *
from  patient_info, patient_data
where patient_info.name='Jone' AND patient_data.last_exam <=
'1998-09-16'
```

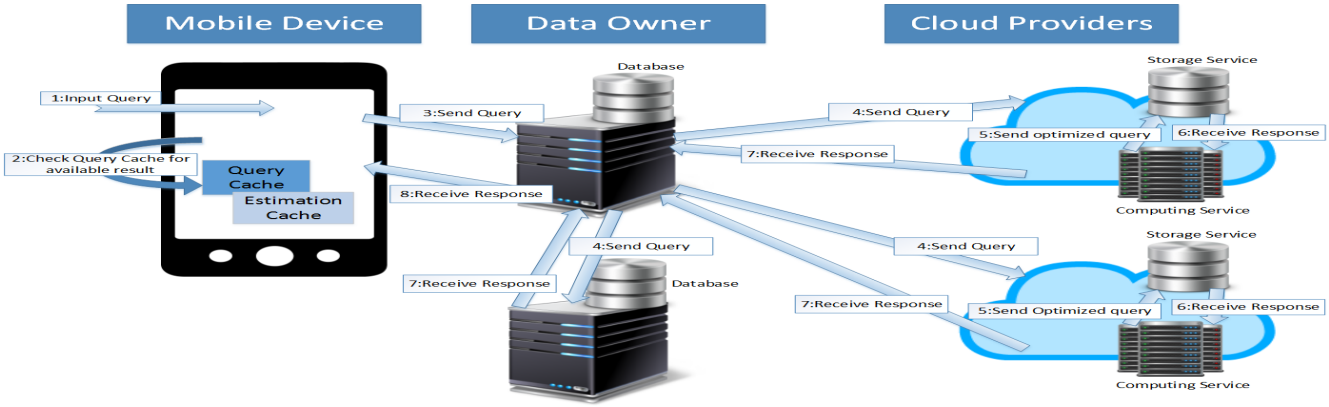

Figure 1. Mobile - Cloud Database Environment

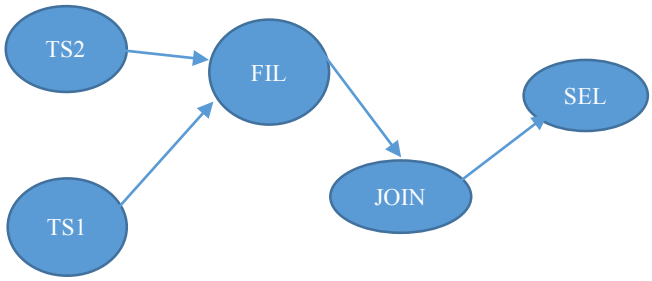

Figure 2. A Example of query operator DAG converted from the query

After this query is submitted, it will be first convert to different operators as seen in Figure 2. Each node represent an operator compiled from the query and will be then assigned to different containers. The arrows represent the data processing flow. Suppose we have 2 containers available to which each of the operator can be assigned. Container 1 is able to process the whole query within 10 seconds with a price of $20 per second, and Container 2 takes 20 seconds to process the query but with a cheaper price of $5 per second. Assigning these operators to Containers 1 or 2 will result in different times and monetary costs for the query plan. In addition, if we update the statistics of the query execution after the FIL operator, we will find that the actual data volume is larger than the estimation, so the JOIN operator might migrate from Container 1 to Container 2.

And in general cases the doctor need to decide which Container he/she would like to choose since one option is quicker in response time and another one is cheaper in price. Assuming the doctor is currently working in the ER room and has only several seconds to retrieve the patient's data, the first option might be a better choice. In our approach, we handle this situation. First, our algorithm is able to find suitable QEPs which meet the constraints of the user. Secondly, a QEP will be selected without interacting with the users. In this example, the doctor does not need to select which option to execute the plan. The QEP will be selected automatically according the previous Weight Profile settings.

In the following sections, we briefly present our previous research on the interaction model of the user and mobile-cloud database system. Additionally, we present an algorithm and describe how to optimize the queries on the cloud with the new interaction model.

2.1 Multi-Objective QEP formation

Assigning different operators to different containers is also known as a scheduling problem. A QEP is also known as a schedule and a schedule contains several assignments. An assignment includes the information of the operator and its assigned container plus the scheduled starting time and estimated ending time for executing this operator. Typically, a local optimal schedule is generated by a greedy algorithm and an optimizing algorithm is applied to optimize the allocation in the schedule so that a near optimal schedule can be found. However, in this work, we generate only a local optimized schedule. By referencing to the results reported in [1], a local optimal schedule is not significantly improved by applying the generic optimization algorithm, which is Simulation Annealing used in this work, and applying the generic optimization algorithm generates extra noticeable overhead. We adopt these results and do not apply any optimization algorithms after all the operators are assigned.
An assignment of an operator to a container can be modeled as follows:

$$\{\text{assign} < s_n^i, C_j, t_{start}, t_{end} >\}$$

where $\mathbf{s_n^i}$ denotes the operator $\mathbf{s_n}$ in the i-th optimization round; $\mathbf{C_j}$ denotes the container; and $\mathbf{t_{start}}, \mathbf{t_{end}}$ denote the start and end timestamps of the operator's execution.

Before the assignment, the Unit price of using each container is set and a set of system-wide parameters are set for each container and these parameters indicate the CPU usage and time consumption of each container to process a certain amount of data. For example, processing 1KB of data in Container 1 costs 0.1 seconds and occupies 1% of CPU usage. The estimation of how much execution time and percentage of CPU usage of executing an operator on each of the containers is made based on these parameters and the data size each operator is estimated to process. Since the elasticity of the cloud environment and the containers can be added or reduced at any time, this estimation has to be done when there are new incoming containers. In Figure 3, for example, the estimation of execution operator TS1 on Container 1 will cost 0.06 seconds and 12% of the CPU usage.

Operator	Container 1	Container 2	Container 3
TS1	(0.06, 12)	(0.07, 15)	(0.06, 20)
FIL1	(0.10, 16)	(0.20, 19)	(0.50, 17)

Figure 3. An example of the estimation

Assigning an individual operator to a container can be based on several different criterions. An operator can always be assigned to a container that has the minimum current CPU usage to balance the CPU utilization or to a container that has the minimum unit price to minimize the monetary cost. Here, each operator is assigned to the container which has the minimum estimated completion time. Note that, if the assigned container is congested, which means the CPU usage is over 100%, the completion time is adjusted by multiplying it with the CPU usage to influence the overlapping of multiple operators at the same time. The reason of choosing the container with the minimum estimated completion time is according to the experiment results reported in [1]. The results show that among 8 different assignment algorithms, the performance of this assignment stays between the best and worst cases in both time and monetary cost. This means this kind of assignment algorithm takes both time and monetary cost into consideration and balance the trader-offs between query response time requirement and budget.

Algorithm: Continuous Optimization
INPUT:
Sql: query
CONST: two-dimensional variable containing time and money constraints
C: a set of containers and each container has the percentage of current CPU usage and the network bandwidth.
OUTPUT:
Result: the result of the query

```
1.  Ops <- compile query to get its set of compiler-generated
2.  operators
3.  Operator_Tree <- generate a multi-staged Operator tree from
4.  the set of operators Ops
5.  for each stage in the multi-staged Operator-Tree
6.      G <- map each stage in Operator-Tree
7.      and form a dataflow graph
8.      Estimate the Operator Tree
9.      Candidate_Optimized_Schedule<-assign Operators to fastest container to form the schedule
10.     Optimized_Schedule<-apply Weight Profile to select optimized schedule
11.     Result <- execute the current stage of Optimized_Schedule
12.     Optimizer_Operators <- Eliminate the finished operators from the Operator_Tree
13.     if (Reoptimization_Policy = true)
14.       update constraints and operator statistics
15.     end if
16. end for
17. return Result
```

Figure 4. Continuous Optimization Algorithm

The assignment starts with the operators without any dependencies. The assigned container is chosen based on the above criteria. Then the operator is assigned the current timestamp as t_{start} and t_{end} is computed by propagating the estimated execution time to the t_{start}. Then these operators are eliminated and the new operators become ready to be assigned as the dependencies change. This process repeats until all the operators are assigned and finally, a schedule is formed and the total time and monetary cost of the schedule can be obtained. This procedure starts with using one container and repeats till all the containers has been used. Each time a schedule is formed, the estimated execution time and monetary cost are checked. If either of them violates the user's constraint, this schedule will not be considered as a candidate. A Weight Profile will be used at this time to select one schedule or QEP from the candidates and sent to execution. A Weight Profile is a set of weights that represents user's preference on the objectives and this setting is applied to a decision algorithm introduced in our previous work [10] which decides the selection of one QEP among the candidate QEPs. Candidate QEPs are Pareto optimal query plans and require further selection. For example, if a user set a high preference on monetary cost, a QEP with lower price but slow in query response time will be selected.

2.2 Continuous Optimization

The execution of the QEP in our approach is completed stage by stage. A stage is formed by several executing operators.
So far we have defined each stage to contain only one operator. The reason of executing the QEP stage by stage is to get more accurate estimation of the operators. As mentioned in Section 2.1, estimation is very crucial to the assignment of operators and containers which influences the overall cost the QEP. The estimation is made based on the statistics of the data, and the more accurate the statistics is, the more accurate the estimation can be. Using the actual statistics instead of the estimated statistics improves the estimation. After the completion of a stage, the statistics are updated with the actual running statistics collected from the finished stage. A new whole query plan is re-generated and combined with the old query plan. The statistic on data size largely impacts the performance of the query plan. As mentioned in Section 2.1, the assignment of each operator is based on its estimation and this estimation is highly based the data size. For example, when the "where a<500" clause in the query is processed, there is a Filter Operator generated for this clause. When the initial estimation is made before the query execution, the estimated number of rows for "a<500" could be 100,000 and the data size is 50MB. This number is changed if we update the statistics with the actual running statistics of 30,000 rows after the Filter operator is executed. By applying this change, the data size following the operator changes from 50MB to 16MB and the estimation of time and CPU usage on every containers is changed, which will impact the decision of assignment of this operator to the container.

2.3 Query processing re-optimization

When to pause the execution for re-optimization and how to form a stage are the issues that need to study further. This is because the re-optimization causes extra overheads especially in the queries that are executed within a short time. However, in this early version of our system, we force the query optimizer to do the re-optimization.

We have developed new algorithm that adopts the features of multi-objective optimization and continuous-optimization. As we can see in Figure 3, when a query is received, it is parsed, optimized and represented in an operator tree which contains the physical operators of the query plan (Line 1). This query processing so far is the same as the query processing in traditional database systems. After that, the operators are estimated for the cost and assigned to the containers to form a schedule(s) by using the technique discussed in Section 2.1 (Lines 8-10). One schedule is then selected by applying the Weight Profile (Lines 10) After that, each operator is executed based on the schedule and the execution is paused at the end of the stage and the statistics are updated. For each table, the updated statistics include the number of rows and the data size of the table, and for each column, the cardinality and the min and max values of the column is updated. Then the whole query plan is generated and the remaining operators in the unfinished stage are replaced with the new operators in the re-generated query plan. These steps repeat for each stage until all the operators are executed**.**

3. Research Challenges and Discussion

3.1 Scheduling Problem

As discussed in Section 2.1, a schedule is formed by assigning different operators to different containers. This is known as a NP-complete [8] problem in optimization and this problem is also applied to other research fields. Our approach only uses the local optimal solution as the answer and adopts the results in [1]. Thus there is a lot of space for improvement to get an optimal schedule at the initial place Although there are some algorithms targeting this problem [11][12][13], to get a global optimal schedule with little overhead at the beginning is still very challenging.

3.2 Accurate Estimation

In addition, since the operator-container assignment is largely influenced by the estimation of the data, to get an accurate estimation is not easy to do. Though we use continuous optimization to use actual running statistics to replace the estimated statistics, this procedure is very time-consuming and generates a significant overhead. How to predict the statistics of the data with a low overhead is another challenge.

3.3 Re-Optimization Point

Besides that, as mentioned in Section 2.2, how to efficiently form a "stage" is very crucial in continuous-optimization procedure. Too frequent re-optimization result in a large amount of overhead and less frequent re-optimization lose accurate statistics. There is research [4] on this topic and discussed when to do the re-optimization. However, this discussion is under traditional database system. In our case, there will be more challenges in mobile-cloud database environment.

4. Conclusion and Future Work

This paper present our apporach in query optimization with several features. First, a new user interaction model is introduced so that any user input during the query optimization process is not required. Insead we use preference settings known as Weight Profiles to decide QEP selection. Second, an algorithm is presented features in both multi-objetive and continous optimization. The goal of this algorithm is to search for a QEP both satisfies multi-objective and the execution cost is reduced by adjusting the QEP during the execution. We vision our algorithm here and there will be experiments to validate our algortihm in the future.

5. References

[1] Kllapi, H., Sitaridi, E., Tsangaris, M. M., & Ioannidis, Y. Schedule optimization for data processing flows on the cloud. *In Proceedings of the 2011 International Conference on Management of Data,SIGMOD'11.* (June 2011) 289-300

[2] Bruno, N., Jain, S., & Zhou, J. Continuous cloud-scale query optimization and processing. *In Proceedings of the VLDB Endowment, 6(11),* 961–972.

[3] Wu, W., Naughton, J. F., & Singh, H. Sampling-Based Query Re-Optimization. *CoRR,* abs/1601.05748.

[4] Y. Watanabe and H. Kitagawa, "Adaptive Query Optimization Method for Multiple Continuous Queries," 21st International Conference on Data Engineering Workshops (ICDEW'05)*,* 2005, 1242-1242.

[5] Meister, A., Breß, S., & Saake, G. (2014). Cost-aware query optimization during cloud-based Complex Event Processing. Lecture Notes in Informatics (LNI*), In Proceedings - Series of the Gesellschaft Fur Informatik (GI),* 705–716.

[6] Jongwuk Lee, Gae-won You, and Seung-won Hwang. Personalized top-k skyline queries in high-dimensional space. *Information Systems, 34(1)*, 2009

[7] Donald Kossmann, Frank Ramsak, and Steffen Rost. 2002. Shooting stars in the sky: an online algorithm for skyline queries. *In* Proceedings of the 28th international conference on Very Large Data Bases *(VLDB '02)*. 2002,VLDB Endowment 275-286.

[8] R. L. Graham. "Bounds on Multiprocessing Timing Anomalies". *SIAM Journal of AppliedMathematics,* 17(2):416–429

[9] Mengmeng Liu, Zachary G. Ives, and Boon Thau Loo. Enabling Incremental Query Re-Optimization. *In Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16). 1705-1720.*

[10] F. Helff, L. Gruenwald and L. d'Orazio, "Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments," *In Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference (March 2016)* Volume 1558

[11] Dokeroglu,T.,Sert, A.,Cinar, S."Evolutionary Multiobjective Query Workload Optimization of Cloud Data Warehouses," *The Scientific World Journal*, vol. 2014,

[12] T. M. Blackwell , Peter J. Bentley, Dynamic Search With Charged Swarms, *In Proceedings of the Genetic and Evolutionary Computation Conference,* (July 2002) 19-26

[13] Immanuel Trummer and Christoph Koch.Approximation schemes for many-objective query optimization. *In* Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data *(SIGMOD '14)*.1299-1310