Plan Bouquet based Techniques for Variable Sized Databases

Achint Chaudhary

November 3, 2019

Mid-term MTech Project Report

Abstract

OLAP applications require a certain set of queries to be fired on databases with the change in certain values in queries. For optimal execution of these queries, the query optimizer has to select an optimal strategy known as the query execution plan. These choices are based on selectivity estimates of various predicates that differ from actual selectivity values encountered during execution. Due to this reason, we get bad choices of execution plans that led to high variance in actual execution cost as compared to predict during optimization phase. An altogether different approach for query processing is proposed in 2014, named Plan Bouquet. Basis of which is selectivity discovery at run-time by repeated execution of multiple plans. This technique provides strong bounds independent of data distribution.

Plan Bouquet on other hand is not robust against large updates in the database. The research work focuses on providing incremental algorithms that can use past information about plan bouquet and iso-cost contours, to provide further robust execution without incurring entire compilation overhead of plan bouquet.

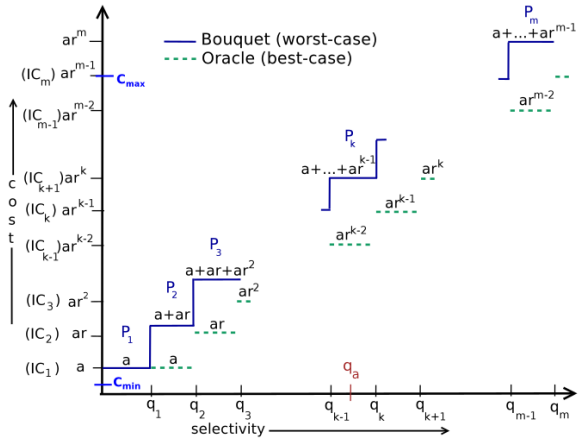
1 INTRODUCTION

Query Optimization is done as choosing the cheapest query execution plan, which comes from various structural choices of logical and physical operators for query execution. Decision-based in these choices are based on the cost of each operator which is calculated using number of tuples it has to process known as cardinality. Cardinality normalized is known as selectivity throughout our analysis.

These selectivity values are estimated before query execution based on some statistical models used in classical cost-based optimizers. An entirely different approach based on run-time selectivity discovery is proposed known as Plan Bouquet, which provides for first time theoretical strong bounds on worst-case performance as compared to optimal performance possible from all of the available plan choices.

For each given query, predicates having the potential of selectivity error contributes as a dimension in Error-Prone Selectivity Space (ESS). The set of Optimal plans over the entire range of selectivity values in ESS is called Parametric Set of Optimal Plans (POSP). POSP is generated by asking optimizer's optimal plans for plans at various selectivity values using Selectivity injection module.

A subset of POSP is identified as Plan bouquet, which is obtained by the intersection of plans trajectory with iso-cost surfaces, each of which is placed at some ratio proportion of cost from the previous surface.



Since these executions form geometric progress, the total cost of which is can be also derived using the sum of geometric progression. The figure above [1]

shows the performance of Bouquet w.r.t to optimal oracle performance.

In above figure, various plans up to actual selectivity value q_a are executed. Each plan has a limit provided by the next iso-cost surface. This yields total execution cost of [1]

$$C_{bouquet}(q_a) = cost(IC_1) + cost(IC_2) + ... + cost(IC_k)$$

$$= a + ar + ar^2 + ... + ar^{k-1} = \frac{a(r^k - 1)}{r - 1}$$

This leads to suboptimality (which is ratio of incurred cost to optimal cost) of bouquet approach of [1]

$$SubOpt(*, q_a) \leq \frac{\frac{a(r^k-1)}{r-1}}{ar^{k-2}} = \frac{r^2}{r-1} - \frac{r^{2-k}}{r-1} \leq \frac{r^2}{r-1}$$

This value is minimized using r=2, which provides a theoretical worst case bound of 4 times the optimal execution time for any query invocation.

2 Present Directions

2.1 Using Previous Plans

During the compilation phase of Plan Bouquet, multiple calls are made to query optimizer, by changing values of selectivities. These are costly operations, and since plan structures depends on cardinality values, not selectivity directly. Plans at initial compilation can still be used for reducing the overhead of bouquet re-compilation for database instance with different scale values.

2.2 Using Previous Contours

Iso-cost surface detection algorithm *NEXUS* [1], is another costly operation for contour detection and placing plans within the Least Upper Bound of those surfaces. After database scale up these contours can be totally changed, while re-execution of entire algorithm will be costly. On the other hand usage of previous contours while adding few more in case of database scale-up will help reduce compile time overhead, and performance bound in that case can still be provided with some marginal threshold.

3 Experimental Setup

Present experiments are done on PostgreSQL-9.4.1 with the additional functionality of foreign plan costing and external plan forcing for execution. Standard workload testing benchmark TPC-DS is used, a database instance of 1GB is created with all schematic constraints like foreign keys, and indexes. Metadata of this database is copied into multiple databases and scaled to 10 different values ranging from 25 to 400 GB in geometric progression.

Plan Bouquet is implemented as Python API externally, not inside Postgres which is used to emulate all components of the bouquet algorithm. FPC is used to identify the range of selectivity captured from old bouquet's plans in a scaled version of the database. Selectivities are also taken as a geometric progression.

4 Conclusions and Future Work

Providing a general framework for handling multi-dimensional queries with the update in data is what we are looking for. Also, work is to be done towards providing theoretical bounds during updates in the database. Geometry based [2, 3] or Machine learning [4] techniques can be used to extrapolate plans beyond there region of optimality on which they are evaluated on the database before the update in size.

References

[1] Anshuman Dutt and Jayant R Haritsa. Plan bouquets: query processing without selectivity estimation. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pages 10391050. ACM, 2014

[2] Srinivas Karthik, Jayant R. Haritsa, Sreyash Kenkre, and Vinayaka Pandit. A Concave Path to Low-overhead Robust Query Processing. PVLDB, 11 (13): 2183-2195, 2018.

[3] Numerical Optimization (Jorge Nocedal and Stephen J. Wright), Springer, 2006.

[4] Hastie, T., Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). The elements of statistical learning: Data mining, inference, and prediction. New York: Springer.