Plan Bouquet based Techniques for Variable Sized Databases

Achint Chaudhary
February 2, 2020

Mid-term MTech Project Report (CSA)

Abstract

OLAP applications often require a certain set of canned queries to be fired on database with the varying constants in queries. For optimal execution of these queries, query optimizer does select a strategy known as the query execution plan. These choices are based on cardinality estimates of various predicates that often hugely differ from actual cardinality values encountered during execution. Due to this reason, optimizer choice leads to high inflation in actual execution cost as compared to predicted cost during optimization.

An altogether different approach for query processing was proposed in 2014, named Plan Bouquet. Basis of which is selectivity discovery at run-time by repeated cost bounded execution of carefully chosen to set of plans. This technique provides strong bounds independent of data distribution.

However, Plan Bouquet on cost suboptimality is not robust against large updates in the database. This work focuses on providing incremental algorithms that can use information from plan bouquet compiled in past and extend it, to provide further robust execution without incurring overhead of re-compiling entire plan bouquet.

Sec.1 Introduction

Database query optimizer choose a plan covering various structural choices of logical and physical operators for query execution. These choices are based on the cost of each operator which is calculated using number of tuples it will process known as $cardinality$. Cardinality normalized in range of [0, 1] 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 called Plan Bouquet, which provides for the first time strong theoretical bounds on worst-case performance as compared to optimal performance possible from all 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 chose plans at various selectivity values using Selectivity injection module. An $Iso - cost\ surface$ is collection of all points from ESS which have same optimal plan cost at that location.

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 (r_{pb}) of cost from the previous surface.

Since each plan on an iso-cost surface has a bounded execution limit, and incurred cost by execution using bouquet will form geometric progress. The figure below shows the performance of bouquet w.r.t to optimal oracle performance.

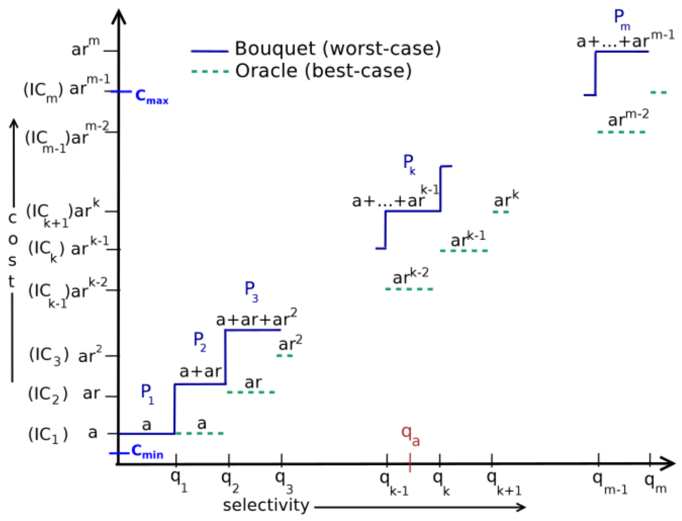

Fig 1. Cost incurred by Oracular vs Bouquet Execution

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

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

$$= a + ar_{pb} + ar_{pb}^2 + \cdots + ar_{pb}^{k-1} = \frac{a(r_{pb}^k - 1)}{r_{pb} - 1}$$

This leads to Sub-Optimality (ratio of incurred cost to optimal cost) of bouquet approach as

$$SubOpt(*, q_a) \leq \frac{\frac{a(r_{pb}^k - 1)}{r_{pb} - 1}}{ar_{pb}^{k-2}} = \frac{r_{pb}^2}{r_{pb} - 1} - \frac{r_{pb}^{2-k}}{r_{pb} - 1}$$

$$\leq \frac{r_{pb}^2}{r_{pb} - 1}$$

This value is minimized using $r_{pb} = 2$, which provides theoretical worst case bound of 4 times the optimal execution time.

Sec. 2 Notations, Notes & Assumptions

2.a Notations used in this work

Notation	Description
p	Query Predicate
QP	Set of all Query Predicates
AKP	Set of Actually Known Predicates
EPP	Set of Error Prone Predicates
TP	Set of Trivial predicates
ESS	EPP Selectivity Space
Dim	Dimension of ESS
$Cost_{sel}$	Cost of Optimal Plan at selectivity sel
$(1+\Delta)$ or S	Scaling Factor of Predicate in Database
$\mathcal{E}$	Minimum Selectivity on Predicate
RES	Resolution on Each axis of ESS
m	Number of Iso-cost Contours
r_{pb}	Cost Ratio of Iso-cost contours
$(0, 1.0]$ or $[\mathcal{E}, 1.0]$	Discretized interval for each axis of ESS
$POSP$	Set of Optimal Plans over entire ESS
$Cost(P, q)$	Cost of plan P at selectivity location q in ESS
$Card(p, q, size)$	Cardinality of predicate p at location q with database size scale s
d_{sel}	Difference in Selectivity values
r_{sel}	Ratio of Selectivity values
β	Worst case slope of Plan Cost Function
α	Tolerance of contour thickening

2.b Notes

2.b.1 Interplay of Selectivity & Cardinality

Selectivity is the fraction of tuples out of maximum possible tuples that can come out of a query predicate. Notation of selectivity is devised to make study of ESS independent of cardinality values.

2.b.2 Distribution of Selectivity values on axis of ESS

ESS is $(0,1]^{Dim}$ or $[\mathcal{E}, 1.0]^{Dim}$ and will be discretized at finite resolution RES during initial compilation with number of points RES^{Dim}. Each axis denoting selectivity for each of predicate in EPP will have RES points. One choice that will matter is whether to take these points in Arithmetic progression (AP) or Geometric progression (GP) in interval $(0,1]$.

GP gives more focus on lower selectivities (near origin) since the plans and cost rapidly changes in that region of ESS. Choice of GP also helps in designing efficient incremental bouquet algorithm.

Ɛ is minimum considered selectivity, sooner we will see bounds on this also under some assumptions.

2.b.3 Trivial Predicates

These are predicates never seen when dealing with a fixed size database, as they always have maximum selectivity of 1.0. Example of this is scanning of entire relation without any filter.

These predicates are useful when comparing old and new database with reference to each other.

$$QP = EPP \cup AKP \cup TP$$

2.c Assumptions

2.c.1 Plan Cost Monotonicity (PCM)

This assumption implies that if location b dominates location a in selectivity component of each predicate, processing more tuples will have more cost

$$(b \succ a) \rightarrow \ Cost(P, b) > Cost(P, a)$$

2.c.2 Perfect Cost Model of Optimizer

This assumption states that poor choices of plan come only from cardinality estimation error of optimizer and not from cost model itself. While, we have assumed perfect cost model of optimizer, an optimizer with bounded cost model will also work well. Improving, which is an orthogonal problem, one work on static tuning [4], proves most actual cost values to lie in 30% of estimated cost values after tuning.

2.c.3 Axis Parallel Concavity (APC)

This assumption is on Plan Cost Function (F_p) which is not just monotonic but exhibit a weak form of $concavity$ in their cost trajectories. For 1D ESS, F_p is said to be concave if for any two selectivities locations q_a, q_b from ESS and any $\theta \in [0,1]$ below condition holds

$$F_p(\theta * q_a + (1-\theta) * q_b) \geq \ \theta * F(q_a) + (1-\theta) * F(q_b)$$

Generalizing to D dimensions, a PCF F_p is said to be $axis\ parallel\ concave\ (APC)$ if the function is concave along every axis-parallel 1D segment of ESS.

Which simply states that each PCF should be concave along every vertical and horizontal line in the ESS

Further, an important and easily provable implication of the $PCFs$ exhibiting APC is that the corresponding $Optimal\ Cost\ Surface\ (OCS)$, which is the infimum of the PCFs, also satisfies APC. Finally, for ease of presentation, we will generically use concavity to mean APC in the remainder of this work.

2.c.4 EPP are only query predicates

For present analysis, we have considered that all of query predicates are Error-prone, there is also no trivial predicate, which means each relation has some filter applied over it.

Rational behind this assumption is that, if we supply same selectivity value to both old and new database, their outputs will be of different cardinalities. Also, we need to determine change of scale for AKP and TP, in that case, maybe re-compilation of bouquet is needed.

Sec. 3 Challenges in database size change

Plan bouquet is suitable for canned queries as compilation overhead of $O(RES^{Dim})$ is amortized over repeated invocations. Under situation of change of database size, placements of ideal contours can be totally different from existing contours built for old database size, which may result that old bouquet plans are totally different from new bouquet contours.

Under above mentioned conditions it seems that a re-compilation will be needed. While, in this work we will focus on how efforts done in earlier compilation can be utilized for incremental compilation, and to provide performance guarantees like Plan Bouquet Approach.

Sec. 4 Present Directions

4.a Using Old Contours

Compilation phase of Bouquet based approach needs $O(RES^{Dim})$, this much cost cannot be afforded if database size changes frequently, also what we care is change of scale of each predicate as compared to what it was when bouquet was initially compiled.

$$Cost(\, P_{opt}\, ,(1.0)\,) = Cost_{max} = Cost(IC_{m_{old}})$$

Placement of contours in bouquet works like we first create a contour of $Cost(IC_m)$ and keep placing other contours from r_{pb} cost ratio less than the cost of previously created contour.

In this section we prove that we can:

1. Use previous complete contours
2. Extend previous incomplete contour into extended ESS
3. Create new contours in extended region of ESS when needed

Focusing on a 1-Dimensional version, where a predicate $p \in ESS$ is scaled up, scale change can be evaluated as

$$(1+\Delta)\ or\ S = \frac{Card(p, 1.0, size_{new})}{Card(p, 1.0, size_{old})}$$

With $only\ EPP$ assumption, we can view predicate on new database, as scaled up version of predicate on old database by extending maximum selectivity up to $(1+\Delta)$.

From now onwards, our entire formulation takes old database as reference, because bouquet is compiled on that.

Formula for number of contours

$$m = floor\left(\frac{\log\left(\frac{Cost_{max}}{Cost_{min}}\right)}{\log(r_{pb})} + 1\right)$$

Using above expression, number of contours in old and new database will be

$$m_{old} = floor\left(\frac{\log\left(\frac{Cost\left(P_{opt}, (1.0)\right)}{Cost\left(P_{opt}, (\mathcal{E})\right)}\right)}{\log(r_{pb})} + 1\right)$$

$$m_{new} = floor\left(\frac{\log\left(\frac{Cost\left(P_{opt}, (1.0+\Delta)\right)}{Cost\left(P_{opt}, (\mathcal{E})\right)}\right)}{\log(r_{pb})} + 1\right)$$

If ESS is extended to $(1+\Delta)$, in any dimensional ESS, the last contour is always a point, a single optimal plan at that location just needs an optimizer call, so that last contour need not be saved during incremental compilation.

IF [$\frac{\log\left(\frac{Cost\left(P_{opt}, (1.0+\Delta)\right)}{Cost\left(P_{opt}, (1.0)\right)}\right)}{\log(r_{pb})}$ is an INTEGER] :

$$m_{extra} = m_{new} - m_{old} - 1$$

ELSE:

$$m_{extra} = m_{new} - m_{old}$$

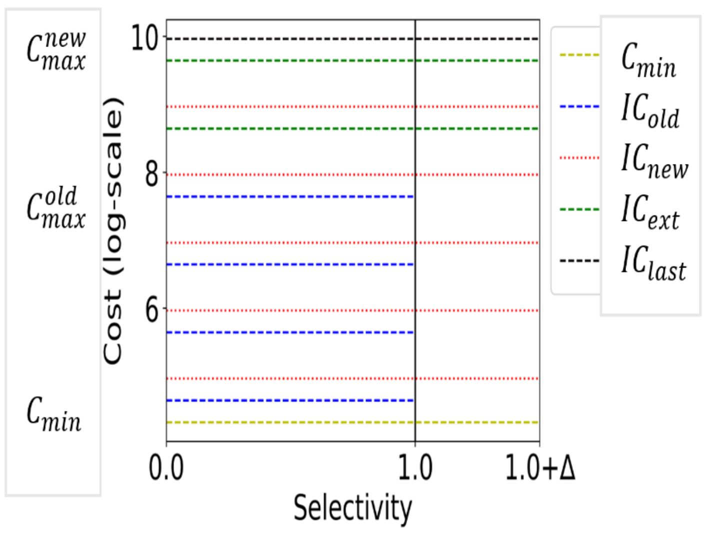

Fig 2. Placement of Ideal re-drawn contours vs Extended contours of initial bouquet

4.b Changes in existing contours

During re-compilation it can be case that most of effort in existing ESS are redundant and utilizing those earlier effort is crucial work that lead to decrease overheads.

While utilizing previous contours on a scaled version of database, plans at same selectivity location may change and even shape of Iso-cost contour may change significantly, in that case plans already on old contour may result in sub-optimality.

Since, both plans on previous contours and shape of contours are amenable to change, what we seek is an Algorithm for identifying plans on iso-cost contours in scale-up version using information from ESS which is already there before scaling.

4.c Proof of Performance Guarantee

Having evaluated m_{extra}, which is number of new contours to place at cost ratio r_{pb} from last contour of previously compiled bouquet (IC_{old} in Fig 2). These extended contours are presented as IC_{ext} in Fig 2, also plan under entire re-compilation are shown as IC_{new}.

Also, if last contour of m_{extra} has cost value lesser than C_{max}^{new} (which is $Cost(\,P_{opt}\,, (1.0)\,)$), then an additional last contour is constructed which will always contain an optimal plan at maximum selectivity location.

If this contour exist, it will have cost ratio r_{last} which is less than r_{pb} from last constructed contour out of m_{extra} contours built during incremental compilation.

Proof will be carried out in two cases.

Case 1. When selectivity discovery of q_a using bouquet execution can be completed up to $IC_{m_{old}+m_{extra}}$ contours.

$$SubOpt(*, q_a) \leq \frac{\frac{a(r_{pb}^{k}-1)}{r_{pb}-1}}{ar_{pb}^{k-2}} = \frac{r_{pb}^{2}}{r_{pb}-1} - \frac{r_{pb}^{2-k}}{r_{pb}-1}$$
$$\leq \frac{r_{pb}^{2}}{r_{pb}-1}$$

Above case satisfies same Suboptimality condition as original bouquet and will give performance guarantee of 4 using $r_{pb} = 2$.

Part 2. When selectivity discovery of q_a using bouquet execution will need execution of IC_{last}

Since, we have shown in figure that last contour if it exists will have lesser cost ratio from previous compiled iso-cost contour as $r_{last} < r_{pb}$. Under same consideration expression for suboptimality is

$$SubOpt(*, q_a) \leq \frac{ar_{pb}^{k-2}r_{last} + \frac{a(r_{pb}^{k-1}-1)}{r_{pb}-1}}{ar_{pb}^{k-2}}$$
$$= r_{last} + \frac{r_{pb}}{r_{pb}-1} - \frac{r_{pb}^{2-k}}{r_{pb}-1}$$
$$\leq r_{last} + \frac{r_{pb}}{r_{pb}-1}$$

Sine r_{last} has upper bound of r_{pb}, when $r_{pb} = 2$ is substituted in above expression and upper bound of r_{last} is used for r_{last}. Suboptimality in that case is also upper bounded by 4.

Hence, we can state that incremental bouquet compilation under assumption of $EPP\ are\ only\ query\ predicates$, will provide same worst-case performance guarantee.

This shows that in the region of ESS for which Bouquet is already compiled no redundant effort is required.

4.d Incremental Bouquet Compilation Approach

4.d.1 Constructing New contours

Given m_{extra} which is number of extra contours that needs to be drawn in new ESS. Portion of ESS which is newly constructed should be searched for these contours and bound on those efforts are shown in later sub-section.

4.d.2 Completing existing contours

It can be case that existing contours, needs to be completed in extended ESS. Contours that have either intersection on ESS boundary beyond anti-diagonal should be completed during incremental compilation.

In both constructing of new contours and completing new one w.r.t new ESS, we will require an effort of

$$O(ESS_{Volumne_{new}} - ESS_{Volumne_{old}})$$

With above formulation, we should try to minimize that volume of new ESS should not be too high w.r.t old ESS.

4.e Number of Extra Points Needed on each axis

Ratio for Geometric progression of selectivity values of old data base and new database to be created are

$$r_{sel} = \left(\frac{1}{\mathcal{E}}\right)^{1/(RES-1)}$$

Extending same relation to new scale of predicate and total points

$$r_{sel} = \left(\frac{1+\Delta}{\mathcal{E}}\right)^{1/(K-1)}$$

Re-arranging to find K, we get

$$K = ceil\left(\frac{\log\left(\frac{1+\Delta}{\mathcal{E}}\right)}{\log(r_{old})}\right) + 1$$

$$= ceil\left(\frac{\log\left(\frac{1+\Delta}{\mathcal{E}}\right)}{\log\left(\frac{1}{\mathcal{E}}\right)} * (RES-1)\right) + 1$$

To solve this equation further, we need to find what $\mathcal{E}$ is, this minimum selectivity possible should also have some lower bound otherwise there are too large jumps on selectivity values on axis in values close to maximum selectivity of $1+\Delta$ for incremental bouquet.

To derive lower bound on $\mathcal{E}$, we can consider worst case of concavity assumption, which is linear F_{PIC}

$$r_{pb} \le \beta * r_{old}$$

$$\mathcal{E} \ge \left(\frac{\beta}{r_{pb}}\right)^{1/(RES-1)}$$

Using extreme lower bound value of $\mathcal{E}$, we get

$$K = ceil\left(\frac{\log(1+\Delta)}{\log\left(\frac{r_{pb}}{\beta}\right)}\right) + RES$$

Extra points needed on axis will be $K - RES$, which will be

$$ceil\left(\frac{\log(1+\Delta)}{\log\left(\frac{r_{pb}}{\beta}\right)}\right) or\ ceil\left(\frac{\log(S)}{\log\left(\frac{r_{pb}}{\beta}\right)}\right)$$

$$\boldsymbol{\theta}(\mathbf{log}(\mathbf{1}+\Delta))\ \boldsymbol{or}\ \boldsymbol{\theta}(\mathbf{log}(\boldsymbol{S}))$$

This bound on Number of points needed is independent of RES. While if we have done through

Arithmetic progression-based range of selectivities on each axis, it will be

$$\theta(\Delta * RES)$$

4.f Efforts needed for incremental compilation

Since done with previous section, which claims that linear number of more points will be needed with even exponential increase in database size, incremental compilation will need

$$\boldsymbol{O}\left(\ \left(\boldsymbol{RES} + \boldsymbol{\theta}(\mathbf{log}(\boldsymbol{S}))\right)^{\boldsymbol{Dim}} - \boldsymbol{RES}^{\boldsymbol{Dim}} \ \right)$$

Which is much less than, what we get using arithmetic progression for selectivity values on each axis for arbitrarily large scale-up on each predicate axis of ESS.

$$O(\ RES^{Dim} * \left((1+\Delta)^{Dim} - 1\right))$$

5 Experiments

6 Conclusion and Future Work

Each predicate axis in ESS will demand extra points that are only logarithm of scale change of that predicate from old to new database, which is also independent of RES taken to discretize ESS.

Given proof of concept is not able to handle the change of scale in AKP, for now, to handle this we can go with a conservative assumption of $QP = EPP$, and consider $AKP \ and \ TP$ also as EPP comes with curse of dimensionality. we can apply dimensionality reduction techniques [3], if not anything else to handle $AKP \ and \ TP$.

Also, there can be some trivial predicates (TP) within a query, like a simple scan of a relation without any filter will result different cardinalities on different sized databases, and hence will impact optimal plans generated for on both old and new sized database. To take care of this, Trivial predicates should be created and combined with AKP in future experiments.

7 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 D. Pandit. **A Concave Path to Low-overhead Robust Query Processing**, In Proc. of the VLDB Endow., 11(13), pages 2183-2195, 2018

[3] Sanket Purandare, Srinivas Karthik and Jayant R. Haritsa. **Dimensionality Reduction Techniques for Robust Query Processing**, Technical Report TR-2018-02, DSL CDS/CSA, IISc, 2018.

[4] Wentao Wu, Yun Chi, Shenghuo Zhu, Jun'ichi Tatemura, Hakan Hacigms, and Je_rey F.
Naughton. Predicting query execution time: Are optimizer cost models really unusable?
In Proc. of 29th Intl. Conf. on Data Engg., ICDE '13, pages 1081{1092, 2013. 5, 78, 115