Green Query Optimization: Taming Query Optimization Overheads through Plan Recycling

Parag Sarda   Jayant R. Haritsa*

Database Systems Lab, SERC/CSA
Indian Institute of Science, Bangalore 560012, INDIA.

Abstract

PLASTIC [1] is a recently-proposed tool to help query optimizers significantly amortize optimization overheads through a technique of plan recycling. The tool groups similar queries into clusters and uses the optimizer-generated plan for the cluster representative to execute all future queries assigned to the cluster. An earlier demo [2] had presented a basic prototype implementation of PLASTIC. We have now significantly extended the scope, useability, and efficiency of PLASTIC, by incorporating a variety of new features, including an enhanced query feature vector, variable-sized clustering and a decision-tree-based query classifier. The demo of the upgraded PLASTIC tool is shown on commercial database platforms (IBM DB2 and Oracle 9i).

1 Introduction

Query optimization is a computationally intensive process, especially for complex queries. Recently, in [1], we presented a tool called PLASTIC (PLAn Selection Through Incremental Clustering), that can be used by query optimizers to amortize the optimization overheads through a technique of "plan recycling". Specifically, PLASTIC groups similar queries into clusters and uses the optimizer-generated plan for the cluster representative to execute all future queries assigned to the cluster. Queries are characterized by a feature vector that captures both query structures and statistics, and query similarity is evaluated using a distance function on these feature vectors. Experiments with a variety of queries (based on the TPC-H benchmark [5]) on a commercial optimizer showed that PLAS-

TIC predicts the correct plan choice in most cases, thereby providing significantly improved query optimization times. Further, even when errors were made, only marginal additional execution costs were incurred due to the sub-optimal plan choices.

Apart from the obvious advantage of speeding up optimization time, PLASTIC also improves query execution efficiency because optimizers can now always run at their highest optimization level – the cost of such optimization is amortized over all future queries that reuse these plans. Further, the benefits of "plan hints", a common technique for influencing optimizer plan choices for specific queries, automatically percolate to the entire set of queries that are associated with this plan. Lastly, since the assignment of queries to clusters is based on database statistics, the plan choice for a given query is *adaptive* to the current state of the database.

An earlier demo [2] had presented a basic prototype implementation of PLASTIC. We have, in the interim period, significantly extended the scope, useability, and efficiency of PLASTIC, by incorporating a host of new features, including:

- variable-sized clustering to match volatility in plan space;

- integration of a decision tree classifier for fast cluster identification;

- increased scope for query matching across resources and schemas; augmented feature vector to account for table access paths and organizations, as also index types

- computation and incorporation of cost estimations in plan generation;

- automated mechanisms for creating and visualizing integrated "plan-cost diagrams", which enumerate the plans chosen by the optimizer over the query space, and show the associated costs; and

- a plan analyzer module for comparing plan choices within and across database platforms.

In this demo, we present a walk-through of the upgraded PLASTIC tool, and explain how it helps to (a) significantly amortize the overheads of query optimization, and (b) serve

---

*Contact Author: haritsa@dsl.serc.iisc.ernet.in

as a research, educational and administrative tool for understanding the intricacies of query plan generation. The demo is hosted on commercial database platforms (IBM DB2 and Oracle 9i).

2   Overview of PLASTIC

The well-known inherent costs of query optimization are compounded by the fact that a query submitted to the database system is typically optimized afresh, providing no opportunity to amortize these overheads over prior optimizations. While current commercial query optimizers do provide facilities for reusing execution plans (e.g. "stored outlines" in Oracle 9i [6]), the query matching is extremely restrictive – only if the incoming query has a close *textual resemblance* with one of the stored queries is the associated plan re-used to execute the new query.

Our recently-proposed PLASTIC tool [1] attempts to significantly increase the scope of plan reuse. It is based on the observation that even queries which differ in projection, selection and join predicates, as also in the base tables themselves, may still have identical *plan templates* – that is, they share a common database operator tree, although the specific inputs to these operators could be different. By identifying such similarities in the plan space, we can materially improve the utility of plan cacheing.

PLASTIC captures these similarities by characterizing queries in terms of a feature vector that includes structural attributes such as the number of tables and joins in the query, as well as statistical quantities such as the sizes of the tables participating in the query. Using a distance function defined on these feature vectors, queries are grouped into clusters, which are built incrementally. Each cluster has a representative for whom the template of the optimizer-generated execution plan is persistently stored, and this plan template is used to execute all future queries assigned to the cluster. In short, PLASTIC recycles plan templates based on the expectation that its clustering mechanism is likely to assign an execution plan identical to what the optimizer would have produced on the same query.

A block-level diagram of the PLASTIC architecture is shown in Figure 1. The user query is first processed by the *Feature Vector Extractor* which also accesses the system catalogs and obtains the information required to produce the feature vector. The *SimilarityCheck* module establishes whether this feature vector has a sufficiently close match with any of the cluster representatives in the *Query Cluster Database*. If a match is found (solid lines in Figure 1), the plan template for the matching cluster representative is accessed from the *Plan Template Database*. The plan template is converted into a complete plan by the *Plan Generator* module, which fills in the operator inputs based on the specifics of the current query.

On the other hand, if no matching cluster is found (dashed lines in Figure 1), the Query Optimizer is invoked in the traditional manner and its output plan is used to execute the query. This plan is also passed to the *Plan Template Generator* which converts the plan into its template
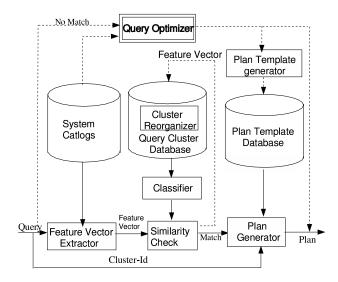

Figure 1: The PLASTIC Architecture

representation for storage in the *Plan Template Database*. Concurrently, the query feature vector is stored in the *Query Cluster Database*, as a new cluster representative.

3   New Features of PLASTIC

A basic prototype implementation of PLASTIC was demonstrated earlier in [2]. In this section, we describe in detail the new features of PLASTIC that are highlighted in the current demo. These new features significantly extend the scope, useability, and efficiency of PLASTIC, as explained below.

Variable-Sized Clustering. PLASTIC originally implemented *fixed-size* clusters, resulting in the twin problems of insufficient clusters in the *high-volatility* (rapid changes in plan choices) regions of the plan space and redundant clusters in the *low-volatility* (gradual changes in plan choices) regions. We have now incorporated *variable-sized* clustering in PLASTIC, providing several small-sized clusters in the high-volatility region and a few large-sized clusters in the low-volatility region. Our scheme is based on the observation that the high-volatility region is typically present in the highly-selective region of the plan space. Therefore, we have modified the distance function, used to compare similarities between queries, such that the cluster size thresholds are small in the highly-selective regions, and large in the less-selective regions – details are available in [3].

A sample output of variable-sized clustering for a TPC-H-based *query template*[1] is shown in Figure 2. Here, the axes represents the selectivities of a pair of the participating relations, namely PART and PARTSUPP, and the dots represent the cluster representatives. Our initial experimental results indicate that variable clustering can reduce plan prediction errors by almost 50 percent [3].

---

[1] A query template represents a query in which some or all of the constants in the where-clause predicates have been replaced by bind variables.
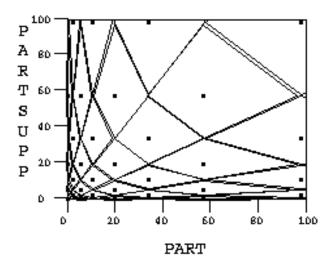
Figure 2: Variable-Sized clustering

Decision-Tree Classifier. In the original tool, identifying the matching cluster (if any) for the new query, was achieved by comparing the new query with the cluster representatives until either a similar representative was found, or all were found to be dis-similar. This process can become computationally expensive when a large number of clusters are present, as would often be the case. To hasten the process of cluster identification, we have now incorporated a new classifier module into the PLASTIC architecture (see Figure 1), which operates on the clusters in the database, after grouping them based on plan commonality – that is, clusters sharing the same plan are grouped together and a classifier is built on these groups. Specifically, the popular C 5.0 [7] decision-tree classifier has been integrated into our prototype.

To optimize the grouping process, initially the plan template of each query representative is traversed in post-order and an MD5 hash signature of this traversal is computed. Subsequently, these signatures, rather than the plans themselves, are compared to decide plan commonality among clusters. Such grouping significantly reduces the number of class labels in the cluster database, and has twofold advantages: Firstly, it increases the accuracy of the classifier, and secondly, results in a decision tree of lesser height, thereby requiring lesser time for classification. Quantitatively, our experiments show that the cluster identification time reduces by an *order of magnitude*, at only a small cost in the overall matching accuracy [3]. In fact, with the classifier, the identification cost is proportional to the *heterogeneity* of the clusters, whereas in the earlier version, the cost was proportional to the *cardinality* of the clusters.

The decision tree also helps to identify the attributes of the query feature vector that have the most impact on plan choices. We expect that this information will be especially beneficial for both database administrators and query optimization researchers/students. A sample output of the classifier module is shown in Figure 3 (here, ETS refers to Effective Table Size, a query feature vector component [1]).
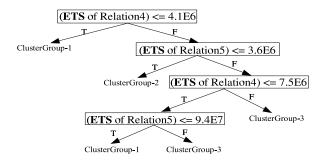

Figure 3: Classifier Module Output

Increased Scope for Query Matching. Our earlier work was effective only when all the queries were on a common schema. We have now extended the PLASTIC tool to share plan templates even for queries operating on different schemas. An example of the breadth of matching is shown in Figure 4, where Query 2 is correctly identified as having the same plan template as Query 1, which is a cluster representative in the database.

```
SELECT *
FROM EMPLOYEE, EMP_ACT, EMP_PHOTO
WHERE EMPLOYEE.empno = EMP_ACT.empno
   and EMP_ACT.empno = EMP_PHOTO.empno
   and EMPLOYEE.empno > 0
   and EMP_ACT.empno  < 400
   and EMP_PHOTO.empno between 10 and 390
```
Query 1

```
SELECT EMPLOYEE.name, EMPLOYEE.project,
       EMP_RESUME.resume
FROM EMPLOYEE, EMP_ACT, EMP_RESUME
WHERE EMPLOYEE.empno = EMP_ACT.empno
   and EMP_ACT.empno = EMP_RESUME.empno
```
Query 2

Figure 4: Breadth of Matching by PLASTIC

We have also augmented the PLASTIC feature vector to take into account different *table access paths* (table scans and index scans), *table organizations* (clustered and B-tree), and *index types* (unique, cluster, and reverse).

Further, it was earlier conservatively assumed that the presence of an index on a query attribute in the cluster representative meant that it would be used and therefore a new query that lacked an index on its corresponding attribute could not be matched to this representative. The system now checks for the *actual usage* of such resources in the representative and removes it from the representative's feature vector if not used, thereby increasing the scope for query matching and plan sharing.

Including Cost Estimations in Generated Plans. The plans generated by PLASTIC were incomplete in that they did not include the associated *operator costs*, although this is a standard feature of plans generated by the optimizer. Note that the costs of the cluster representative's plan cannot be directly used for this purpose since there can be

significant variation in the matched queries, although they share a common plan template. To address this issue, we have included an estimation module in the plan generator that scales and interpolates the costs of the cluster representative to reflect the costs of the new query, and includes this information in the generated plan.

Since the new query and its matched cluster representative may differ on selectivities of multiple relations, a *multivariate* strategy is required to compute the interpolated values. Currently, we have implemented first order multivariate interpolation based on barycentric coordinates [8], which has a low time complexity. Our initial experiments indicate that the interpolation is accurate to within 90 percent [3].

Automated Plan-Cost Diagram Generator. Generating a "plan diagram" [1], which is an enumeration of the plan choices of the optimizer over the query space, is a computationally expensive process since it involves firing of a large number of queries. Earlier, this diagram was constructed from scratch for each query, but we have now added a facility for persistently storing these diagrams directly in the database, from which they can be immediately retrieved at the desired time. We have also mapped these stored plan diagrams with the associated clusters, to provide better visual interfaces for generating, viewing, and reorganizing query clusters. Further, we have augmented the plan diagram, which is a qualitative picture, to include quantitative costs, thereby resulting in a combined *plan-cost* diagram. An example plan-cost diagram[2] corresponding to a TPC-H-based query, is shown in Figure 5. Note that there are eleven plan optimality regions (P1 through P11) spanning the entire selectivity spectrum on the PART and PARTSUPP relations, and the sizes of the dots indicate the relative plan costs.
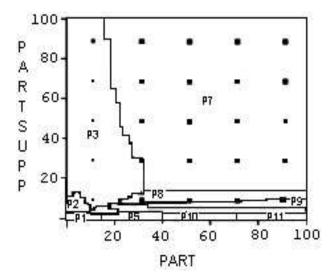

Figure 5: PLASTIC Plan-Cost Diagram

Plan Analyzer Module. A new Plan Analyzer module,

---

intended for analyzing the specific differences between a pair of query execution plans, has been incorporated in the Plastic tool. We expect that this module will be especially beneficial for database administrators and query optimization researchers/students to help understand plan choices made by the optimizer.

The module identifies differences between plans using an adaption of the X-Diff [4] algorithm (which was proposed for computing differences between XML documents). It can be used in two ways: (a) to compare plan choices for different versions of a query on a single platform, or (b) to compare choices for the same query across database platforms. Part of a sample output of the module, showing the differences between Plan 7 and Plan 9 of Figure 5, is shown in Figure 6 (the source plans are omitted because of their complexity and size). From this figure, we can see that these two plans differ only in the access method (*index scan* versus *table scan*) for the PARTSUPP relation.
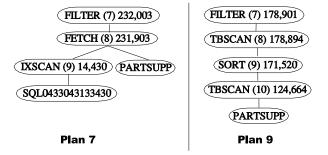

Figure 6: Plan Differences

References

[1] A. Ghosh, J. Parikh, V. Sengar and J. Haritsa, "Plan Selection based on Query Clustering", *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2002.

[2] V. Sengar and J. Haritsa, "PLASTIC: Reducing Query Optimization Overheads through Plan Recycling", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2003.

[3] P. Sarda, "Green Query Optimization: Taming Query Optimization Overheads through Plan Recycling", *Master's Thesis*, CSA, Indian Institute of Science, July 2004.

[4] Y. Wang, D. DeWitt and J.Cai, "X-Diff: A Fast Change Detection Algorithm for XMLDocuments" *Proc. of 19th IEEE Intl. Conference on Data Engineering*, March 2003.

[5] http://www.tpc.org

[6] http://download-east.oracle.com/otndoc/oracle9i/ 901_doc/server.901/a87503/toc.htm

[7] http://www.rulequest.com/see5-info.html

[8] www.library.uu.nl/digiarchief/dip/diss/2003-1028-125323/appd.pdf

---

[2]The diagram is based on plan selections obtained with the IBM DB2 query optimizer.