Predicting Query Execution Time

Name

Mid-term ME Project Report

Abstract

The ability to estimate the query execution time is central for a number of tasks in database system such as query scheduling, progress monitoring and costing during query optimization. Recent work has explored the use of statistical techniques in place of the manually constructed cost models used in query optimization. Such techniques, which require as training data along with the actual execution time, promises superior accuracies for they being able to account the for factors such as hardware characteristics and bias in cardinality estimates. However, such techniques fail to generalize i.e., produce poor estimates for queries that are not seen during the training.

In this work, we propose and evaluate predictive modeling techniques that learn query execution behavior at a fine grained operator level. For each operator, we consider different sets of features and build different models for them. Since there are only finitely many operators in database, this approach is practical and will be able to estimate any query as its a composition of many operators. We evaluate our approaches using TPC-H and TPC-DS workloads on PostgreSQL.

1 INTRODUCTION

Database systems can greatly benefit from accurate execution time predictions including:

- Admission control: Resource managers can use this metric to perform workload allocations such that the specific QoS are met [2].
- Query Scheduling: Knowing the execution time is crucial in deadline and latency aware scheduling
- Progress monitoring: Knowing the execution time of an incoming query can help avoid *rogue queries* that are submitted in error and take an unreasonably long time to execute [3].

Currently, execution time estimation is based on manually constructed models, which are part of the query optimizer and typically use combinations of weighted estimates of the number of tuples flowing through operators, column widths, etc. Unfortunately, such models often fail to capture several factors that affect the actual resource consumption. For example, they may not include detailed modeling of all of the various improvements made to database query processing such as nested loop optimizations [4, 7] which localize references in the inner subtree, and introduce partially blocking batch sorts on the outer side, thereby increasing the memory requirements and CPU time and reducing I/O compared to the traditional iterator model. Similarly, they may not accurately reflect specific hardware characteristics of the current production system or the impact of cardinality estimation errors. Analytical cost models predominantly used by the current generation of query optimizers cannot capture these interactions and complexity; in fact, they are not designed to do so. While they do a good job of comparing the costs of alternative query plans, they are poor predictors of plan execution latency. Recent work [4] showed this result for TPC-DS [10], and in this work we do the same for TPC-H [9] data and queries.

In this work, we utilize learning based models and prediction techniques which are promising reasonable accuracies in recent work [4, 5, 6]

2 Background:Model Based Prediction

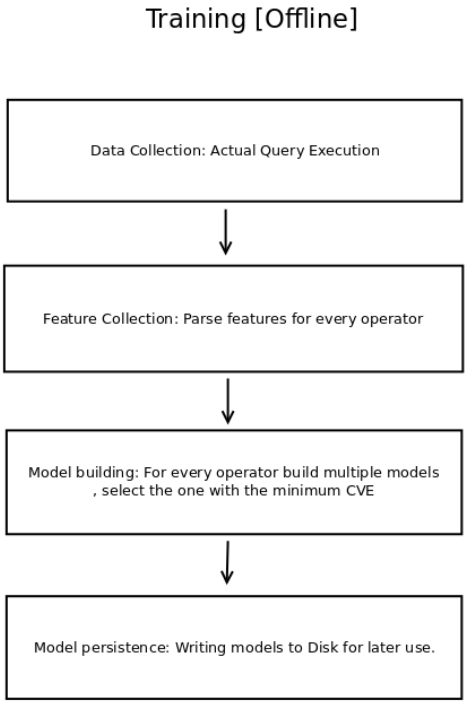
COMEBACK: Write about predictive stuff .. to give more intro. lets skip to imp parts now.

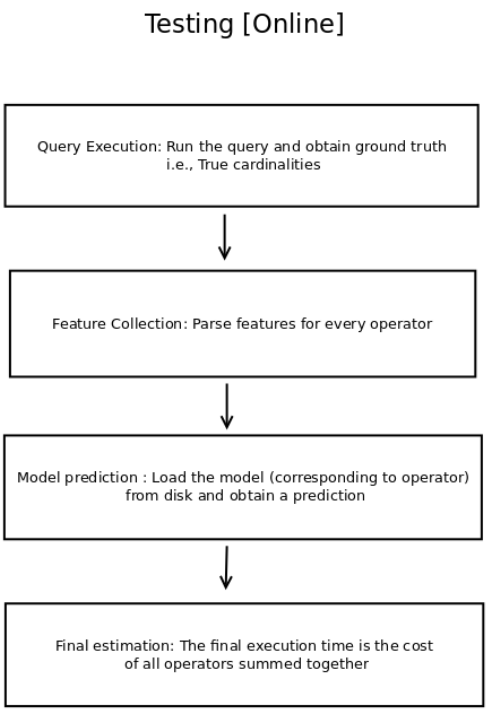
We use the term model to refer to any predictive function such as Multiple Regression, Bayesian Nets, and Support Vector Machines. Training a model involves using historical data sets to determine the best model instance that explains the available data. For example, fitting a function to a time series may yield a specific polynomial instance that can be used to predict future values. Model training (or building) requires selecting (i) the feature at- tributes, a subset of all attributes in the data set, and (ii) a prediction model, e.g., Linear Regression and Support Vector Machines (SVMs), to be used for modeling. In general, we cannot know which model type and feature set will produce the most accurate model for a given data set without building and testing multiple models. In some cases, a domain expert can manually specify the feature attributes. In other cases, this step is trivial as the prediction attribute(s) directly determine the feature attribute(s), e.g., in auto- regressive models. Alternatively, feature attributes can be learned automatically; however, given a set of n attributes, trying the power set is prohibitively expensive if n is not small or training is expen- sive [4, 3, 2] thereby requiring heuristic solutions. Most approaches rank the candidate attributes (often based on their correlation to the prediction attribute using metrics such as infor- mation gain or correlation coefficients) and use this ranking to guide

a heuristic search [4] to identify most predictive attributes tested over a disjoint test data set. In this paper, we will use a similar For- ward Feature Selection algorithm based on linear correlation coef- ficients [4]. This algorithm essentially performs a best-first search in the model space. It starts with building models using small num- ber of features and iteratively builds more complex and accurate models by using more features. The features are considered with respect to their correlation with the target/prediction attribute. The training data set may be sampled to speed up the process. While we use a feature selection algorithm for building accurate models using relevant features, we do not consider building multi- ple models of different types for solving the model selection prob- lem. Instead in each one of our experiments we use a single type of prediction model, either Linear Regression or SVMs, that performs well. Hypothesis testing and confidence interval estimations are two com- mon techniques for determining predictive accuracy [2]. As men- tioned, it is not possible to estimate a priori what model would be most predictive for a given data set without training/testing it. One form of hypothesis testing that is commonly used is K-Fold Cross Validation (K-CV). K-CV divides the observed data up into k non- overlapping partitions. One of the partitions is used as validation data while the other k 1 partitions are used to train the model and to predict the data in the validation interval. In this study, we will use cross-validation techniques to estimate the accuracy of our prediction models.

3 Overview of Proposed Approach

To-Do : improvise diagrams with creately.com





In this section, we elaborate on the overview of the approach which consists of 2 phases. Like most other machine learning approaches, the first phase is an off-line training phase and the second is on-line testing phase. During the training phase, we construct number of different models for each type of physical database operator(e.g., Sort-Merge Join, Nested Loop, Index Nested Loop,..). Each operator will be associated with so called 'Best-Fit' model which will be determined based on the Operator model selection described in section 3.1. This model will be invoked for the purpose of estimating the execution time for a certain operator. Since there are only finite number of operators and only a couple of physical implementations for them, the space consumption is not excessive.

During the testing phase, we actually run the query to get the true cardinality estimates. While this seems to contradict the whole estimation problem, right now this is required because the focus is on solving the modeling problem given the actual cardinalities. The more practical scenario i.e., where we we only the know the estimated cardinalities, is a much harder problem as now the models also need to be Robust to the errors in cardinality estimates. This is the natural follow-up work to do. Once after the execution of the query, we need to extract features depending on the operator and invoke the Best Fit model (which is already computed during training phase) to get an estimate. We sum up costs for all the operator present in the plan tree to finally give an estimate of execution time.

TO-DO : 2. Describe the concept of multiple models for a single operator. 3. How do we select the best one among them. (Mention that current experiments don't use this idea, only one a simple non-linear model is used for all models with 2 features)

Analytical approaches problems:

counting the numbers of pages read is not sufficient as the discrepancy between random and sequential I/O is tremendous.

3.1 Model selection and Training

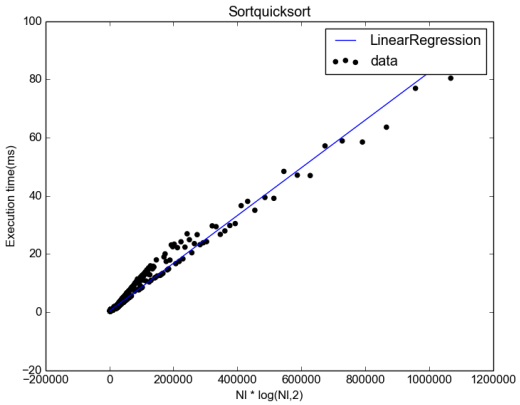
Before we describe the need for multiple models for an operator, we shall present the motivation for doing so. While the statistical models can find the complex non-linear dependencies between features and the target, they cannot model the interaction among operators which might affect the target value.

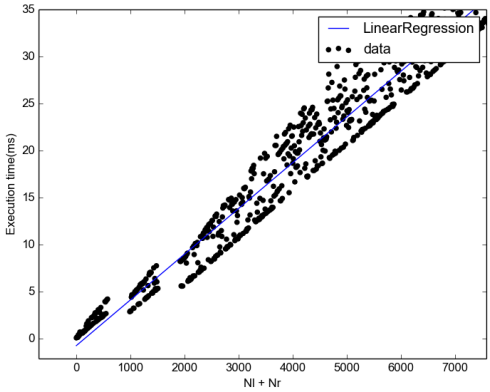
Consider the classic example of Sorting, where in the target variable (execution time) is proportional to $N_l \ logN_l$. Unless this is used as a feature, the accuracy of the resulting model can be disappointing. Since we don't know the right function to be used, we explore the possible set of functions listed bellow and select the one with the minimum L2 error. We've used functions that are similar to the one's defined in [8]

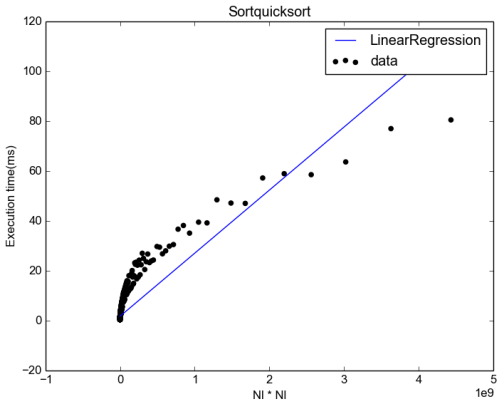
TO-DO : Describe more about the L2 error in this context

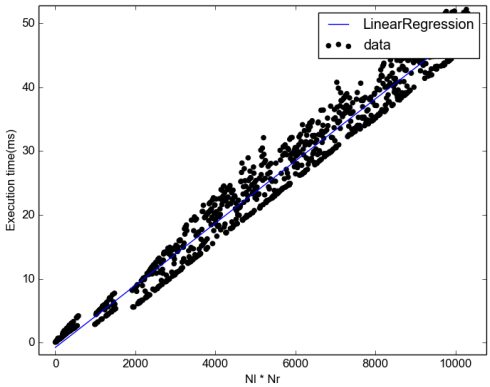
- Linear: $a_0 N_l + a_1 N_r$
- Quadratic: $a_0 N_l N_r$, $a_0 N_l N_l$, $a_0 N_r N_r$ ...
- Logarithmic: $a_0 N_l log(N_r), a_0 N_r log(N_l)$ ..
- Exponential: $a_0 N_l^{N_r}$

TO-DO: Describe the queries and the procedure used to generate graphs. We now show the observations for logarithmic and quadratic functions.

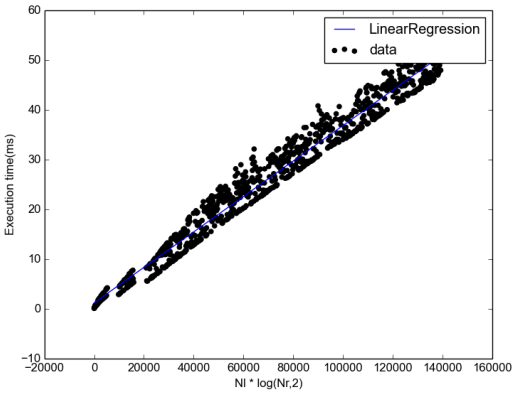








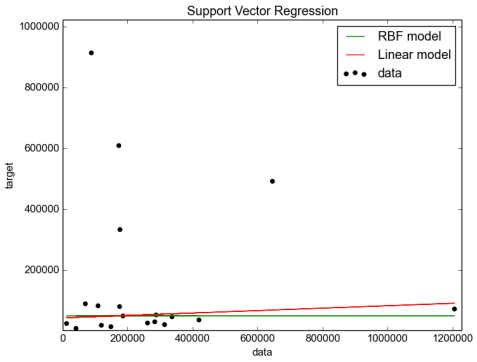
Unsurprisingly, logarithmic function fits better than the quadratic. While in some cases (Such as QuickSort) the appropriate function is obvious from SQL query processing, it does not hold for all operators. For example, consider Nested Loop Join which can be modeled with quadratic functions in general but when the inner relation has an index the same model produces estimates poor estimates which are "off" by orders of magnitude. In the following example, we take an exploratory approach to determine the best function for the Index Nested Loop operator.



What model is considered good ? choice of model and kernel If possible prove linear systems are not sufficient. Training data ,specification. Does more training data helps? training time etc.. Essentially all operators need to be covered. generation QGEN tool.

4 Preliminary Experiments

If possible plot a non-linear SVR for the data. will the optimizer's cost itself is enough 1D regression using linear, polynomial and RBF kernels



Give a background about Static and dynamic work-

load. do experiments at 2 stages 1. Act vs Act 2. Est vs Est(Bias towards estimated cardinalities) For each type of above scenario, do testing for tpc-h queries (tpc-ds in future) At different scales to see the model's ability to predict. For midterm you should be having result for 1 as well as 10 GB. For each above dataset, Results should convey L1 (MRE) as well as queries that are within [0-0.5] [0.5-1] [1-1.5]

5 Conclusions and Future Work

Talk about generalization. operator specific features capturing query interaction i.e., pipeline source of over-estimation Optimizers provides sufficient information to detect a pipeline, can we use it to bind the operators and predict execution time for a set of operators instead of a single operator.

References

[1] A. Ganapathi, H. A. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. I. Jordan, and D. A. Patterson. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In ICDE, 2009.

[2] Li, Jiexing, et al. "Robust estimation of resource consumption for sql queries using statistical techniques." Proceedings of the VLDB Endowment 5.11 (2012): 1555-1566.

[3] Akdere, Mert, et al. "Learning-based query performance modeling and prediction." Data Engineering (ICDE), 2012 IEEE 28th International Conference on. IEEE, 2012.

[4] S. Guirguis, M. A. Sharaf, P. K. Chrysanthis, A. Labrinidis, and K. Pruhs. Adaptive scheduling of web transactions. In ICDE, 2009.

[5] Harish, D., Pooja N. Darera, and Jayant R. Haritsa. "Identifying robust plans through plan diagram reduction." Proceedings of the VLDB Endowment 1.1 (2008): 1124-1140.

[6] TPC-H benchmark specification, http://www.tpc.org/tpch/

[7] R. Othayoth and M. Poess, The making of tpc-ds, in VLDB 06: Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006, pp. 10491058

[8] Du, Weimin, Ravi Krishnamurthy, and Ming-Chien Shan. "Query optimization in a heterogeneous dbms." VLDB. Vol. 92. 1992.

[9] Xiong, Pengcheng, et al. "ActiveSLA: a profit-oriented admission control framework for database-as-a- service providers." Proceedings of the 2nd ACM Symposium on Cloud Computing. ACM, 2011.

[10] Mishra, Chaitanya, and Nick Koudas. "The design of a query monitoring system." ACM Transactions on Database Systems (TODS) 34.1 (2009): 1.