Query Optimization to Meet Performance Targets for Wide Area Applications *

Vladimir Zadorozhny
University of Pittsburgh
Pittsburgh, PA 15260
vladimir@sis.pitt.edu

Louiqa Raschid
University of Maryland
College Park, MD 20742
louiqa@umiacs.umd.edu

Abstract

Recent technology advances have enabled mediated query processing with Internet accessible WebSources. A characteristic of WebSources is that their access costs exhibit transient behavior. These costs depend on the network and server workloads, which are often affected by the time of day, day, etc. Given transient behavior, an appropriate performance target (PT) for a noisy environment will correspond to "at least X percentage of queries will have a latency of less than T units of time". In this paper, we propose an optimizer strategy that is sensitive to the objective of meeting such performance targets (PT). For each query plan, a PT sensitive optimizer uses both the expected value of the cost distribution of the plan, as well as the expected delay of the plan. We validate our strategy using a simulation based study of the optimizer's behavior. We also experimentally validate the optimizer using traces of access costs for real WebSources.

1 Introduction

Recent technology advances have enabled mediated query processing in the wide area environment with remote relations (objects) that are resident on Internet accessible WebSources. There are many characteristics of mediated query processing in this environment that pose substantial challenges. One characteristic is that typically there are several alternate WebSources for each remote relation. Thus, a mediator subquery (on a relation) can be submitted to multiple candidate WebSources. In order to choose between alternate WebSources, an optimizer has to compare their access costs. However, the second characteristic of Internet accessible WebSources is that their access costs exhibit *transient* behavior. The costs may be described as *spatio-temporal cost distributions*. Variation in response time may depend on many factors including the topology of the client and the remote server, the network and server workloads, which are often affected by the time of day, day, etc. and points of congestion between the client and remote server [21].

There has been extensive prior research on query evaluation techniques to accommodate transient behavior. Reactive query evaluation techniques at the plan level are described in [1, 4, 18, 25]. Alternately, adaptive evaluation techniques at the plan and operator implementation level are described in [3, 11, 12, 14, 24]. Research in [2, 7, 13, 15, 29] have addressed various aspects of the task of modifying an optimizer to handle distributions for various parameters, e.g., available memory, or intermediate join cardinality. The Tukwila project [14] provides an adaptive framework composed of rules and adaptive operators that are sensitive to transient factors. Its optimizer also has a re-optimization capability based on pipelined fragments of query execution. The Telegraph project explores adaptive fine-grained data flow based query processing techniques using rivers, eddies, ripple join, XJoin, etc. [3, 11, 24]. They also focus on continuous and high frequency response to changing feedback. Query scrambling [1, 25] is a query optimization and evaluation technique to deal with transient conditions, e.g., unexpected delay. It is based on plan re-organization to avoid idle time, and the synthesis of new operators to execute in the absence of other work. It makes a key contribution of a response time based query optimizer to deal with delay, instead of the more traditional cost based approach.

While adaptive techniques are best suited to overcome transient behavior, an optimizer must address the companion issue of *planning* to choose among noisy sources, to determine those sources that are most likely to meet a performance target (PT). A performance target (PT) will correspond to "at least X percentage of queries will have a latency of less than T units of time" for some particular choice of source(s). This is in contrast to "100% of the queries must execute in less than 400 seconds".

The task of planning to choose among noisy sources is not straightforward. The transient behavior of WebSources and the resulting cost distributions would challenge a traditional cost based optimizer which expects exact costs. This

*This research has been partially supported by the National Science Foundation under grants DMI9908137 and IIS0135142.

would make it difficult to directly compare the cost of plan execution on different (combinations of) WebSources. LEC (least expected cost) is an approach [7] to compare multiple candidate plans when they are associated with cost distributions. LEC uses the expected values of cost distributions for each plan to determine the best plan. LEC is defined formally in Section 4.1. However, the expected value reflects aggregate behavior, and is not a good reflection of the actual response time. (Response time is typically defined to be the time to execute the query and download results from the source.) A noisy source with high variance in its response time will behave differently from a stable source that is less noisy. Meanwhile, both sources could have the same, or similar expected values of the response time. Thus, the difference in actual performance may be independent of the expected values of their access cost distributions.

In this paper, we present a PT sensitive optimizer that explores the relationship between the choice of sources and plans, and the likelihood of meeting a performance target. Our approach to PT sensitive optimization is as follows: for each combination of sources (and cost distributions), we determine a plan and its cost distribution. We characterize the plan with the *expected cost* of the plan, as well as the *expected delay*. The *expected delay* represents the deviation *in excess* of the expected cost. We combine the *expected cost* and the *expected delay* using a *cost factor*, and a *delay factor*, to obtain a combined measure, the *Cost–Delay–Measure* CDM.

Using a simulation based study, we demonstrate that the LEC optimizer is typically unable to choose among sources/plans to meet a desired target, whereas the PT optimizer is able to do this successfully. We validate that the PT sensitive optimizer's behavior is predictable and scalable with respect to the following: alternate sources with varying values of expected cost and expected delays; a mix of queries of varying complexity; a large search space of plans; and traces of access cost distributions obtained from real WebSources.

2 Architecture

Wrapper mediator architectures have been proposed for interoperability of heterogeneous information sources [16, 17, 19, 26]. In this paper, we consider Internet accessible WebSources. A WebSource is accessible via the `http` protocol; a form-based interface provides a limited query capability, and returns answers in XML or HTML. The mediator has the task of decomposing a mediator query into subqueries; identifying relevant WebSources that can answer a subquery; and providing query optimization and evaluation functionalities. Wrappers reflect the limited query capability of WebSources and handle mismatch between the mediator and the WebSource. Our mediator is an extension of the Predator ORDBMS [22].

A major challenge for our optimizer is the unpredictable behavior of WebSources, dictated by network and server workloads and points of congestion in the network. These workloads may be affected by the time of day, the day of the week, etc. We developed a tool, the Web (P)rediction (T)ool, WebPT, based on an online learning and prediction technique [5, 10, 28]. The WebPT is able to exploit the significance of Time and Day on workloads, to construct a more accurate access cost distribution, even in cases where the sources are noisy.

3 Motivating Example

Consider two alternate sources, S_1 and S_2, for a remote mediator relation. The access cost (response time) distributions for these sources are represented by normal (Gaussian) distributions [1] with different values for the expected value μ, and the variance σ. One choice for the optimizer is to choose a source with the *least expected value* (μ), in this example it is source S_1. However, the performance of S_1, i.e., the actual response time of S_1, is not always better than the performance of S_2. Consider Figure 1 that represents a quantile plot of the percentage of queries versus the response time for the two sources. To construct the quantile plots, we used the distributions of S_1 and S_2 to generate the response time for some statistically significant (large) number of queries submitted to these sources.

Figure 1 indicates the risk and the benefit with respect to a performance target of 400 seconds. Source S_1 has the greater benefit with respect to the target. However, it also has the greater risk of queries that exceed the target. For example, if we consider the 90-th percentile, the response time of 90 % of the queries to source S_2 will not exceed 600 seconds, whereas for source S_1, 90 % of the queries will not exceed 800 seconds. This illustrates that in noisy environments, the expected value alone is insufficient to compare sources. The optimizer needs a more flexible strategy that would consider both the *risk* and *benefit* of its choice.

Different applications may favor different targets. An application may only benefit from some information within a certain time period, e.g., less than 400 seconds. Source S_1 has a higher percentile of queries that meet this target. Thus, it has a higher *utility* with respect to meeting this target, and should be the optimizer's choice. The optimizer must emphasize the benefit of each source of meeting this target - this is an *optimistic optimizer strategy*. A utility function can be appropriately constructed where the utility of exceeding the target is close to 0. A different application may have a restriction that exceeding a deadline, e.g., 600 seconds, will be very expensive. Thus, the strategy should be to choose

[1] We use Gaussian distributions only for purposes of explanation.

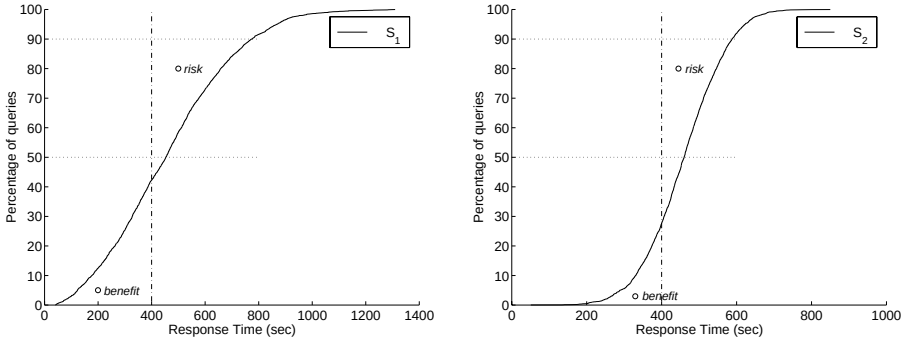

Figure 1. Risk and benefit of Sources S_1 and S_2 for a performance target of 400 seconds

a source where we minimize the percentile of the queries that exceed the target. Source S_2 has a lower percentile of queries (10%) that do not meet this target, and it should be the optimizer's choice. In order to choose S_2, the optimizer must emphasize the risk of each source of not meeting the target. This is defined to be a *conservative strategy*. We consider the utility of close to 100% of the queries in selecting a source(s) to reflect a conservative strategy.

4 A Performance Target Sensitive Optimizer

Our performance target (PT) sensitive optimizer considers both the expected cost and the expected delay of a plan. We briefly review Least Expected Cost (LEC) based optimization [7]. We introduce the concept of *Expected Delay* for a cost distribution, and define the *Cost–Delay–Measure* (CDM) for a query plan.

4.1 Cost–Delay–Measure CDM

Accurate cost estimation for a plan is very difficult since it requires accurate a priori knowledge of costs and details of the run-time environment. The cost of a plan is typically determined using a cost formula and (specific) value(s) for some vector of parameters $\overline{V}$ that affect the cost of the plan. Research in [7] has proposed an alternative *least expected cost* (LEC) approach. Let us assume that the probability distribution of values for each parameter in $\overline{V}$ is independent. Suppose $\overline{V}$ corresponds to K values where each $\overline{V}_K$ represents a combination of values, one for each of the parameters in $\overline{V}$. Each $\overline{V}_K$ has probability of occurrence $Prob(\overline{V}_K)$. Now, the expected cost*EC(p)* = $\sum_K$ (Cost-plan-p($\overline{V}_K$) $\times$ $Prob(\overline{V}_K)$).

The LEC-optimizer compares the expected cost of each plan, and chooses the candidate plan with the least expected cost. The expected cost reflects aggregate behavior but it may be insufficient to determine actual performance. We define the concept of the *Expected Delay (ED)* of a plan for some access cost distributions. Together, EC and ED more accurately reflect the actual performance.

We explain the concept using a plan accessing a single WebSource. Figure 2a illustrates a normal (Gaussian) distribution with expected value μ. Suppose the access cost distribution $distS$ is approximated by k discrete values, where each value val_S_k has a probability of occurrence $prob\_val\_S_k$. Then, the *expected value* [7] of this distribution, also referred to as the *mean*, is $EC(distS) = \sum_k val\_S_k \times prob\_val\_S_k$. Next, we compute the *excess* of each of the k particular values val_S_k of the distribution, compared to the expected value $EC(distS)$. This is $excess\_S_k$. Suppose we only consider the positive excess, when the value val_S_k is greater than $EC(distS)$ (or μ); this deviation represents a *delay*. The probability of this value of delay is $prob\_val\_S_k$. Then, the *expected delay* of the distribution $distS$ is $ED(distS) = \sum_k delay\_S_k \times prob\_val\_S_k$.

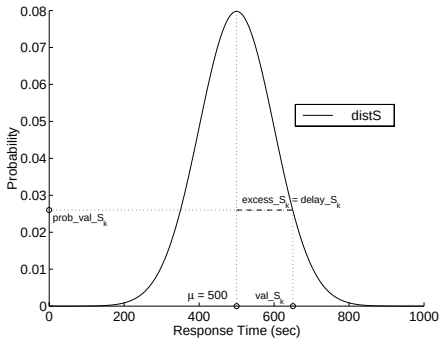

Figure 2. Explanation of Expected Delay

Given a plan p, with an expected cost EC(p) as defined previously, the expected delay for this plan, *ED(p)* = $\sum_K$ (Delay-plan-p($\overline{DelV}_K$) $\times$ $Prob(\overline{V}_K)$).
We describe how Delay-plan-p($\overline{DelV}_K$) for a plan p is calculated from the cost formula for calculating the cost of the plan or Cost-plan-p($\overline{V}_K$) in the next section.

The *Cost–Delay–Measure* (CDM) uses a *cost factor* cf, and a *delay factor* df. The CDM for plan p *CDM(p)*= $cf \times EC(p) + df \times ED(p)$.
The choice of the PT sensitive optimizer is as follows:

> Given a set of candidate plans, and values for cf and df, the optimizer chooses among the candidate plans for one that minimizes the CDM value, for corresponding values for cf and df.

4.2 Calculating the CDM for a Query Plan

The cost of a plan is estimated in a bottom-up manner, starting from the terminal nodes of the plan tree. A terminal node that is a scan on a mediator relation that is resident on a remote WebSources is associated with an access cost distribution. We use these distributions to calculate the cost distribution for each interior node of the plan tree. The cost formula combines the distributions of child nodes to obtain the distribution for the parent node.

Consider a three-way join query on relations R1, R2, and R3. Suppose that WebSources S1, S2, and S3 have been assigned to R1, R2, and R3, respectively. Figure 3 shows the plan tree and the cost distributions for each node. For each leaf node, `RT` is the response time and `prob` is the corresponding probability. For the interior nodes, `Cost` is the cost of evaluating the corresponding operator at the node. First, the scan on relation R1 (or R2 or R3) is assigned the cost distribution for the corresponding source. Next, we calculate the cost distribution for the join of R1 and R2. For simplicity, we assume that the join is implemented as a hash join, and we ignore the local (in memory) cost of the hash join implementation. Then, the cost of the join is the sum of the costs of the two scans on R1 and R2. We also assume that the cost distributions for S1 and S2 are independent, and we use this assumption to calculate the joint probability distribution. For example, a scan cost of 120 seconds for R1 with probability 0.2 (from S1) and a scan cost of 25 seconds for R2 with probability 0.4 (from S2) will result in a join cost of 145 seconds with probability 0.08.

This calculation of cost distributions may result in a distribution that has values with a very low probability, or adjacent values that are very close in value. Such a fine granularity in the distribution is not very useful to the optimizer. In addition, it can significantly decrease the optimizer performance for multi-way join queries. We use a special technique to merge values in the calculated distribution that do not meet some threshold. This is illustrated in Figure 3. Consider the initial cost distribution for the root node of the plan tree (a 3-way join); it is labeled D1. Values whose probability is below a probability threshold, in this case a value of 0.05, are merged to obtain the distribution labeled D2. Finally, adjacent values whose cost difference is less than a threshold, in this case 10 seconds, are also merged, to obtain the final distribution labeled D3. Details are in [27].

Using the final cost distribution D3 for the root node of the plan, we can calculate the expected cost *EC(p)* and the expected delay *ED(p)* for plan p. Finally, using the values for the cost factor cf and the delay factor df we calculate the *CDM* for plan p: $cf \times EC(p) + df \times ED(p)$.

4.3 Generating Candidate Plans

Given a mediator query where N of the relations are remote, we assume that it is the *choice of a particular WebSource and its access cost distribution, for each of these N relations*, that has the greatest impact on the performance target(s). Based on this assumption, we (informally) describe our strategy to produce plans that minimize CDM as follows:

- In the first phase, we enumerate all choices of combinations of WebSource(s) and access cost distribution(s), for each of the remote relations.
- In the second phase, we vary the values of cf and df in the range [0.0,1.0]. We note that for the results reported in this paper, we display the optimizer choices for values of cf and df adding up to 1.0, i.e., the value of cf = (1.0 - df). These values were found to work well to illustrate the optimizer strategy. For each pair of values (cf, df)and its choice of WebSource(s), we use a modified relational optimizer to generate *the best plan* with the least value for CDM.
- For each pair of values (cf, df), we rank the best plans based on their CDM values¿. The winner(s) for each pair of values (cf, df) are selected.

Details of how the best plan is generated as well as the cost model is reported in [27].

4.4 Utility and Optimizer Strategy

Utility functions [6, 20, 23] can be used to quantify how well the response time for a query met some target, and to quantify the utility when a target is not met.

$$\texttt{Utility(T, x, K)} = \begin{cases} 1.0 & \text{if } \texttt{x} \leq \texttt{T} \\ K/(x - T + K) & \text{otherwise} \end{cases}$$

Let `x` be the actual response time and T be the target response time. K is a constant $\geq$0 that is used to tune the rate at which the utility value decreases from 1.0 when x exceeds the target T.

To determine the utility of an aggressive optimistic strategy one would only consider queries that *met the target* and whose utility = 1. In general, we can also associate a positive utility (less than 1) with those queries that exceed the target. We chose a value of K in our experiments so that the utility is 0.25 when the response time $x = 2T$. Other values for K may also be considered with no prejudicial effect. To evaluate a conservative strategy one could consider the utility of *up to 100%* of the queries submitted to the source.

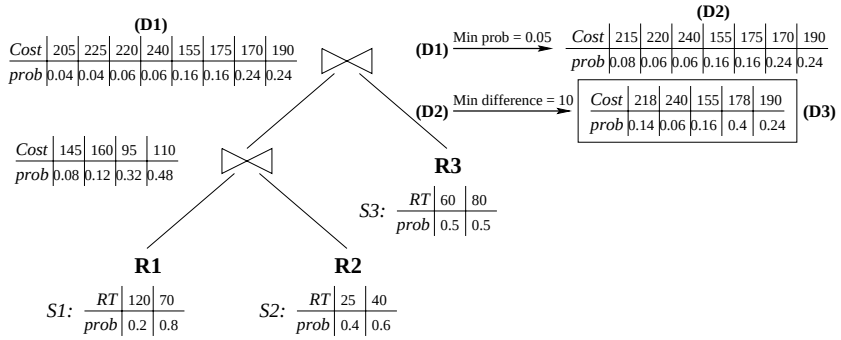

Figure 3. Cost Distributions associated with query plan; RT: Response Time; prob: probability

5 Behavior of the PT Optimizer

5.1 Simulation Environment

For the simulation, our mediator was executed on a Sun Ultra SPARC 1 with 64MB of memory, running Solaris 2.6. While validating access to WebSources, this machine was connected via a 10 Mbps Ethernet cable to the domain umiacs.umd.edu. This domain is connected to its ISP via a 27 Mbps DS3 line.

The mediator queries were complex N-way join queries and we used up to 12 queries in the simulation. They were generated to reflect the complex decision support queries and statistics of the TPC-D Benchmark. For each query, up to 5 relations were identified as relations resident on remote WebSources. For each such relation, we identified up to 3 alternate WebSources, with their dependencies and access cost distributions. All costs are end-to-end delays for downloading data. The mediator statistics were modified so as to vary the percentage of remote processing cost versus the total cost of the queries. We report on behavior for specific queries as well as the *aggregate* behavior over several queries.

5.2 Comparison of the LEC Optimizer and the PT Optimizer

For our first experiment, we chose a five-way join query. Two of the relations were identified as remote relations. For each of the remote relations we considered 2 or 3 different synthetic Gaussian access cost distributions with different values for μ and σ. For various values of cf and df, the PT optimizer chose a combination of sources and the plan with minimum CDM value.

Figure 4(a) reports on CDM values for selected sources and plans, as we varied the delay factor from 0.0 to 1.0, i.e., the cost factor varied from 1.0 to 0.0. The y axis with df = 0.0 represents the LEC optimizer choice. As can be seen, the *winner* as decided by the LEC optimizer is labeled (□) and the worst case plan for the LEC optimizer is labeled (x). As we increase the value of df, and consider the CDM value, the situation changes, and with df = 1.0, the plan labeled (x) has the lowest CDM value.

Figure 4(b) reports on the utility of the actual response times for each of the plans. Suppose we consider a particular target response time $T = 1.25E + 6\ ms$; this value was chosen to represent an average response time across several plans. We note that the trends that we discover hold independent of the particular value of T. Suppose we only consider the %age of those queries with utility = 1.0 that met this target. From (the top of) Figure 4(b), the winner of the LEC optimizer (□) has a smaller %age of queries that met this target. Instead the plan (∗) has the greatest %age of queries with utility = 1.0 followed by the plan (∘). Neither (∗) nor (∘) would have been chosen by the LEC optimizer. Suppose we consider the utility of the queries that do not meet this particular target. Recall that we chose a value of K such that the utility has a value of 0.25 when the response time is $2T$. As we consider more queries, the LEC winner (□) eventually has a higher utility than (∗) or (∘). This is consistent with the PT optimizer; as df increases, the value of CDM for plan (□) decreases rapidly, in comparison to (*) or (∘). If we consider 100% of the queries in Figure 4(b), the highest utility is associated with plan (x). This too is consistent with the PT optimizer, since with df = 1.0, plan (x) has the lowest CDM value. To summarize, the LEC optimizer is not a good predictor of different performance targets (or utility) for sources/plans, whereas the PT optimizer can indeed differentiate between sources/plans using the df and CDM.

5.3 PT Optimizer Behavior on Traces from Real Sources

In the next set of experiments we explore the optimizer behavior on cost distributions obtained from query traces from real WebSources. The sources are EPA [8] and Aircraft [9]. We recognize that these sources are not replicas, so the semantics of these queries are meaningless. Nevertheless, it illustrates the behavior of the PT optimizer with trace data. The source EPA had a low level of noise, whereas

Aircraft was a noisy source. We first report on a single cost distribution for each source. We then consider the impact of distributions that are sensitive to time and day.

The optimizer used the (single) distribution for Aircraft and for EPA to generate cost distributions for the query plans $P1$ and $P2$, respectively. Figure 5a represents quantile plots of response time for $P1$ and $P2$ that were generated from those distributions. We observe that there is a significant risk when choosing the noisy Aircraft (P1) since the response time of a large percentile of the queries exceed the response time of queries on EPA (P2). However, there is indeed a small benefit associated with choice of noisy Aircraft (P1) since the response time of a small percentile of queries is lower than EPA (P2).

CDM values for $P1$ and $P2$ are shown in Figure 5b for different values of delay factor df. Note that as df increases, the values of CDM also decrease. To explain, we chose the value of cf to be = (1.0 - df). Hence the value of CDM can either increase of decrease, and the trend is determined by the relative values of EC and ED for some source. The plan $P2$ based on the less noisy EPA is the winner for all ranges of df, i.e., for all optimizer strategies. This is consistent with the significant risk and a small benefit associated with noisy Aircraft (P1). If we consider the relative CDM ($P1$ - $P2$), it is $\approx$ 1.4E+6 ms when df = 0.0 and increases to $\approx$ 1.9E+6 ms when df = 1.0. This indicates that while $P2$ was always the winner, as df increases, and the PT optimizer becomes more conservative and emphasizes delay, $P2$ wins by an increasing margin (of the relative CDM). Thus, the behavior of the optimizer is consistent with respect to the amount of risk and benefit in choosing query plans on real WebSources. We note that in this case, since the expected cost of $P2$ was less than the expected cost of $P1$, the LEC optimizer would also choose $P2$.

Next, we consider cost distributions that are sensitive to both time interval and day. We report on the CDM for the two plans noisy Aircraft (P1) and less noisy EPA (P2) in Figure 6. We consider the CDM obtained from using a single cost distribution to represent access costs (labeled whole sample). We compare this against the CDM obtained when the cost distribution was constructed to be sensitive to time and day variations. The CDM labeled day-based reflects a cost distribution constructed to represent access costs on a specific day of the week, and time-based reflects a specific N hour time interval over different days of the week. In prior research [5, 10, 28], both time of day and day of week were significant predictors of access costs for these sources, and time and day were exploited to more accurately predict their access costs.

Since Aircraft is a noisy source compared to EPA, the variance of CDM as we increase the value of df is much greater for Aircraft. If we consider the graph labeled day-based for Aircraft, we observe much less variance compared to the whole sample for Aircraft. Thus, the PT optimizer is able to differentiate the situation when knowledge of time and day variations are used in constructing the cost distribution. For the less noisy EPA, the variance of CDM for the whole sample as df increases is much less. Here too, the graphs labeled day-based and time-based show slightly less variance. These experiments indicate that the PT optimizer behavior consistently reflects the behavior of more or less noisy sources, as well as the situation when time and day variations are used in constructing access cost distributions.

5.4 Cost–Delay Trade-Off

We report on the CDM trade-off of the PT optimizer, compared to the LEC optimizer, as we varied sources and queries. In these experiments, we report on the *aggregate* behavior over the query mix. The LEC optimizer is represented by the value of 0.0 for df.

First, we consider one remote relation and 4 alternate WebSources. The EC for the sources was comparable to the ideal source, but the values of ED varied widely. Figure 7a reports on the quantile plots for the response times of query plans, one for each of the sources. Figure 7b compares the relative CDM for each plan/source, compared to the CDM of the plan/ideal source $*$, as we vary the delay factor df. As is seen in Figure 7b, the LEC optimizer (df = 0.0) is not able to distinguish a clear winner among the non ideal sources, since they have similar values for EC. The plan labeled $\triangleright$ in Figure 7a has the greatest benefit and the greatest risk in comparison to the ideal source. The plan labeled $\circ$ has the least benefit and the least risk. From Figure 7b, as we increase the delay factor df, we see that the relative CDM for the plan $\triangleright$ (compared to the ideal source) increases very rapidly which is consistent with the risk. On the other hand, the relative CDM for the plan $\circ$ only increases slightly which is consistent with the low benefit and low risk compared to the ideal source. This further illustrates that while the LEC optimizer was unable to differentiate between the sources/plans, the PT optimizer is able to vary the df and predict behavior, in this case, in comparison to the ideal source.

6 Controlling Optimizer Strategy

In this section, we explore how the value of CDM impacts the optimistic or conservative choices of plans, and therefore the performance targets. Recall, that a conservative strategy corresponds to higher values of df, while the optimistic strategy corresponds to lower values of df.

We consider an experiment with only one remote relation in the query so that we can clearly understand the trade-off

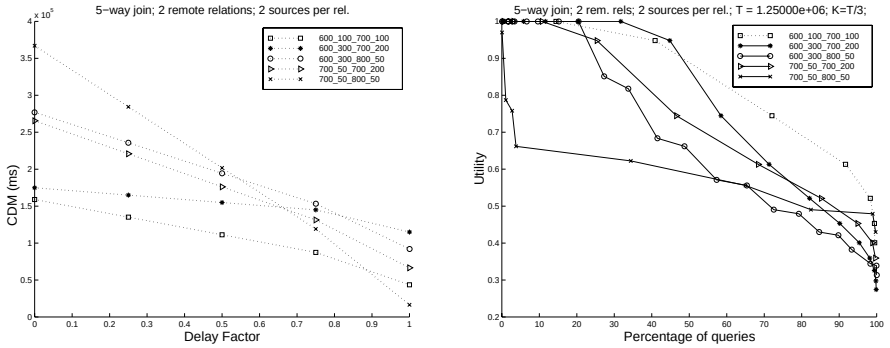

Figure 4. Utility Based Comparison of the LEC and the PT Optimizer

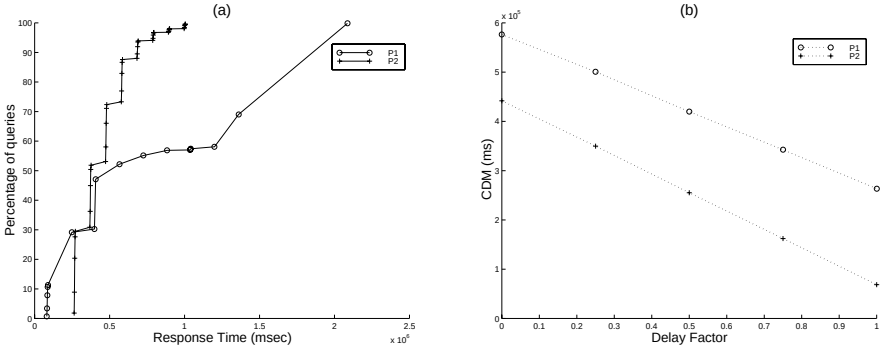

Figure 5. Behavior of Plans with Real WebSources

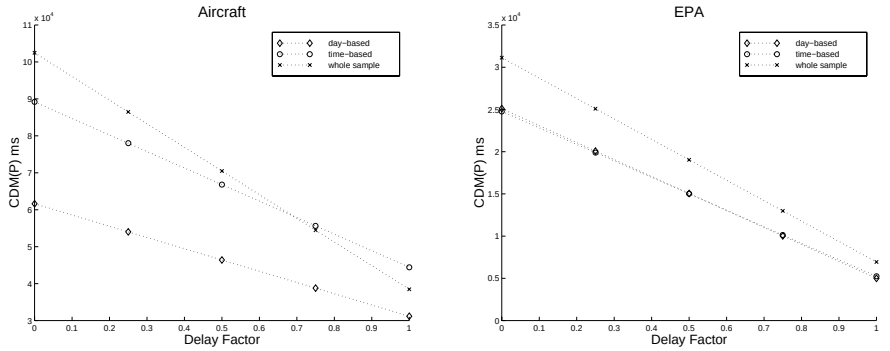

Figure 6. Effect of day and time interval on CDM for Real WebSources

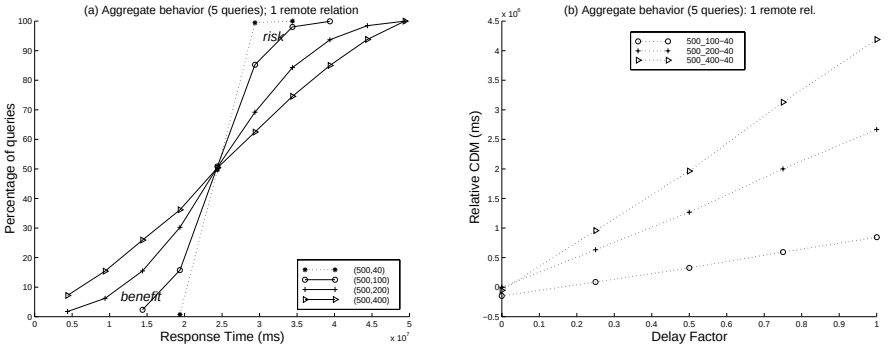

Figure 7. Aggregate Behavior over Multiple Queries; Relative ECD; One Remote Relation

between optimistic and conservative choices. We report on the aggregate behavior over multiple queries. We chose 3 alternate WebSources. The ideal source had ED close to 0 ($\triangleright$) and the other more noisy sources (+ and $\circ$) had lower values for EC, and higher values for ED. Figure 8a reports on the quantile plots which were generated using the synthetic cost distributions. Figure 8b reports on CDM.

First, we make a pairwise comparison of (+,$\triangleright$) and ($\circ$, $\triangleright$) The plan + has significant benefit compared to $\triangleright$; however it also has significant risk. The plan $\circ$ has significant benefit compared to $\triangleright$ and it has little risk. The PT optimizer should favor plan + instead of plan $\triangleright$ only under a *very optimistic* strategy, where it ignores risk and emphasizes the benefit of meeting a target. The plan $\circ$, on the other hand, has significant benefit and little risk compared to $\triangleright$. This implies that plan $\circ$ should be chosen by the optimizer for a wide range of behavior, spanning from optimistic to conservative. Finally, plan $\triangleright$ should be favored when the optimizer is conservative.

Figure 8b confirms that our PT optimizer choices based on values for cost factor, delay factor, and CDM, are consistent with the conservative and optimistic strategies that we have described. In Figure 8b, if we consider an optimistic value for the delay factor, df = 0.25, we observe that the winners of the very optimistic to optimistic strategy ($\circ$ and +) both have lower values of CDM, compared to the conservative choice $\triangleright$. Further, the plan $\circ$ has low CDM values irrespective of the value of df. This plan is the winner for a wide range of behavior from optimistic to fairly conservative.

To complete our analysis, we consider the utility of these plans with respect to two targets. Figure 9(a) represents a lower target (2.5E+7 ms) and Figure 9(b) represents a higher target (3.5E+7 ms). Suppose we desire an aggressive optimistic strategy with respect to the lower target. The choice(s) of the PT optimizer with lower df values ($\circ$ and +) have a larger % of queries with utility=1.0 that meet this target, as observed at the top of Figure 9(a). (+) had the highest percentage of queries, while $\triangleright$ had 0% of queries with utility = 1.0. Suppose we want a conservative strategy with respect to the higher target and we consider the utility of 100% of the queries. In this case, $\triangleright$ is the winner, followed by $\circ$ and +, and this is consistent with df = 1.0.

7 Conclusions and Future Work

We have developed a PT sensitive optimizer based on values of cost factor cf, delay factor df and the Cost-Delay-Measure CDM. We show that the LEC optimizer is unable to differentiate between sources and plans with respect to some performance target. The PT optimizer is able to use the CDM measure to predict the behavior of plans with respect to some target. We also validate that varying the values of cf and df reflect more optimistic or conservative optimizer strategies.

Acknowledgments: We are grateful to Michael Franklin and Fatma Ozcan for their feedback, and to Laura Bright, Tolga Urhan, María Esther Vidal and Tao Zhan for their contributions to the optimizer infrastructure.

References

[1] L. Amsaleg, M. Franklin, A. Tomasic, and T. Urhan. Dynamic query operator scheduling for wide-area remote access. *Journal of Distributed and Parallel Databases*, 6(3), 1998.

[2] G. Antoshenkov. Query processing in dec rdb: Major issues and future challenges. *Data Engineering Bulletin*, 16(4):42–52, 1993.

[3] R. Avnur and J. Hellerstein. Eddies: Continuously adaptive query processing. *SIGMOD Conf.*, 2000.

[4] L. Bouganim, F. Fabret, C. Mohan, and P. Valduriez. A dynamic query processing architecture for data integration systems. *IEEE Data Engineering Bulletin*, 2000.

[5] L. Bright, L. Raschid, V. Zadorozhny, and T. Zhan. A comparison of a web prediction tool and a neural network in learning response times for websources using query feedback. *CoopIS Conf.*, 1999.

[6] G. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, 1997.

[7] F. Chu, J. Halpern, and P. Seshadri. Least expected cost query optimization: An exercise in utility. *Proc. of the Symposium on the Principles of Database Systems*, pages 138–147, 1999.

[8] EPA Toxic Releases Inventory Database. *http://www.epa.gov/enviro/html/tris/tris_query_java.html*.

[9] Landings Aviation Search Engines. *http://www.landings.com/_landings/pages/search.html*.

[10] J.-R. Gruser, L. Raschid, V. Zadorozhny, and T. Zhan. Learning response time for websources using query feedback and application in query optimization. *VLDB Journal*, 9(1), 2000.

[11] P. Haas and J. Hellerstein. Ripple joins for online aggregation. *SIGMOD Conf.*, 1999.

[12] J. Hellerstein et al. Adaptive query processing: Technology in evolution. *IEEE Data Engineering Bulletin*, 23(2), 2000.