Computer and Information Science
www.ccsenet.org/journal.html
Vol. 2, No. 1
February 2009

A Study on Query Optimization for Federated Database Systems

Xinhua Xu
Computer Science & Technology Division
Hubei Polytechnic Institute
Xiaogan City, Hubei 432000, China
E-mail: uxinua@gmail.com

Abstract

In this paper, we explore the design space for a query optimizer in this environment and demonstrate the need for decoupling various aspects of the optimization process. We present minimum-communication decoupled variants of various query optimization techniques, and discuss trade-offs in their performance in this scenario. We have implemented these techniques in the Cohera federated database system and our experimental results, somewhat surprisingly, indicate that a simple two-phase optimization scheme performs fairly well as long as the physical database design is known to the optimizer , though more aggressive algorithms are required otherwise.

Keywords: Query Optimization, Federated database, Optimization Quality, Two-phase

1. Introduction

The need for federated database services has increased dramatically in recent years. Within enterprises, IT infrastructures are often decentralized as a result of mergers, acquisitions, and specialized corporate applications, resulting in deployment of large federated databases. Perhaps more dramatically, the Internet has enabled new inter-enterprise ventures including Business-to-Business Net Markets (or Hubs) (L. Knight), whose business hinges on federating thou-sands of decentralized catalogs and other databases. These decouplings are often forced by administrative constraints, since federations typically span organizational boundaries; decoupling is also motivated by the need to scale the administration and performance of a system across thousands of sites. Federated query processors need to consider three basic decouplings:

Decoupling of Query Processing;

Decoupling of Cost Factors;

Decoupling of Cost Estimation.

In this paper, we consider a large space of federated query optimizer design alternatives and argue the need for taking into consideration the high "cost of costing" in this environment. Accordingly, we present minimum-communication decoupled variants of various well-known optimization techniques. We have implemented these algorithms in the Cohera federated database system (J. M. Hellerstein, M. Stonebraker, & R. Caccia. 2003.) and we present experimental results on a set of modified TPC-Hbenchmark queries.

Our experimental results, somewhat surprisingly, suggest that the simple technique of breaking the optimization process into two phases (W. Hong andM. Stonebraker. 2001.)—first finding the best query plan for a single machine and then scheduling it across the federation based on run time conditions — works very well in the presence of fluctuations in the loads on the underlying data sources and the communication costs, as long as the physical database design is known to the optimizer.

We also present a preliminary analysis explaining this surprising success of the two-phase optimizer for our cost model and experimental settings later in the paper (Section 4.3). Our analysis suggests that this behavior may not merely be a peculiarity of our experimental settings, but may hold true in general.

2. Architecture and Problem Definition

We base our system architecture on the Mariposa re-search system (A. Tomasic, R. Amouroux, P. Bonnet, O. Kapitskaia,H. Naacke, & L. Raschid. 2007.), which provides the decouplings discussed in the earlier section through the use of an economic paradigm. The main idea behind the economic paradigm is to integrate the underlying data sources into a computational economy that captures the autonomous nature of various sites in the federation. A significant and controversial goal of Mariposa was to demonstrate the global efficiency of this economic paradigm, e.g., in terms of distributed load balancing. For our purposes here, controversies over economic policy are not relevant; the long-term adaptivity problem that Mariposa tried to solve is beyond the scope of this paper. The main benefit of the economic model for us is that it provides a fully decoupled costing API among sources. As a result, each site has local autonomy to determine the cost to be reported for an operation, and can take into account factors such as resource consumption, response time, accuracy and staleness of data, admin- istrative issues, and even supply and demand for specialized data processing.

For query optimization purposes, the most relevant parts of the system are the query optimizer in the middleware, and the bidders at the underlying sites (Figure 1). As in a centralized database system, the query optimizer could use a variety of different optimization algorithms, but the federated nature of the system requires that the cost estimates be made by the underlying data sources or in our case, by the bidders. The optimizer and the bidder communicate through use of two constructs: (1) Request for Bid (RFB) that the optimizer uses to request cost of an operation, and (2) Bid through which a bidder makes cost estimates.

2.1 The Federated Query Optimization Problem

The federated query optimization problem is to find an execution plan for a user-specified query that satisfies an optimization goal provided by the user; this goal may be a function of many variables, including response time, total execution cost, accuracy and staleness of the data. For simplicity, we concentrate on two of these factors, response time and total execution cost (measured in abstract cost units), though it is fairly easy to extend these to include other factors, assuming they can be easily estimated. Since we assume that the only information we have about the costs of operations is through the interface to the bidders, the optimization problem has to be restated as optimizing over the cost information exported by the bidders. Before describing the adaptations of the known query optimization algorithms to take into account the high cost of costing, we will discuss two important issues that affect the optimization cost in this framework significantly.

2.2 Simplifying Assumptions

To simplify the discussion in the rest of the paper, we will make the following assumptions :

.Accurate Statistics: We assume that statistics regarding the cardinalities and the selectivities are available. This information can be collected through standard protocols such as ODBC/JDBC that allow querying the host database about statistics, or by caching statistics from earlier query executions .

.Communication Costs: We assume that communication costs remain roughly constant for the duration of optimization and execution of the query, and that the optimizer can estimate the communication costs incurred in data transfer between any two sites involved in the query.

.No Pipelining Across Sites : We assume that there is no pipelining of data among query operators across sites. The main issue with pipelining across sites is that the pipelined operators tend to waste resources, especially space shared resources such as memory (M. N. Garofalakis & Y. E. Ioannidis. 2005.).

3. Adapting the Optimization Techniques

In this section, we discuss our adaptations of various well-known optimization techniques to take into account the high “cost of costing”. Aside from minimizing the total communication cost, we also want to make sure that the plan space explored by the optimization algorithm remains the same as in the centralized version of the algorithm. In general, we will break all optimization algorithms into three steps:

Step 1: Choose subplans that require cost estimates and prepare the requests for bids.

Step 2: Send messages to the bidders requesting costs.

Step 3: Calculate the costs for plans/subplans. If possible, decide on an execution plan for the query, other wise, repeat steps 2 and 3.

Clearly we should try to minimize the number of repetitions of steps 2 and 3, since step 2 involves expensive communication.

3.1 Exhaustive with Exact Pruning

An optimizer may be able to save a considerable amount of computation by pruning away subplans that it knows will not be part of any optimal plan. A top-down approach is more suitable for this kind of pruning than the bottom-up dynamic programming approach we described above, though it is possible to incorporate pruning in that algorithm as well. Typically, these algorithms first find some plan for the query and then use the cost of this plan to prune away those subplans whose cost exceeds the cost of this plan.

The main problem with using this kind of pruning to reduce the total number of bid requests made by the optimizer is that it requires multiple rounds of messages between the optimizer and the data sources. The effectiveness of pruning will depend heavily on the number of rounds of messages and as such, we believe that exact pruning is not very useful in our framework..

3.2 Dynamic Programming with Heuristic Pruning

Since dynamic programming requires the costs for all the feasible joins, we can not reduce the number of bid requests without compromising the optimality of the technique. Heuristic pruning techniques such as Iterative Dynamic Programming (IDP) (D. Kossmann & K. Stocker. 2006.) can be used instead to prune sub-plans earlier so that the total number of cost estimates required is much less. The main idea behind this algorithm is to heuristically choose and fix a subplan for a portion of the query before the optimization process is fully finished.

We experiment with two variants of the iterative dynamicprogramming technique that are similar to the variants described in (D. Kossmann & K. Stocker. 2006.), except that the bid requests are batched together to minimize the number of rounds of messages:

IDP(k) : We adapt this algorithm as follows :

1) Enumerate all feasible k-way joins, i.e., all feasible joins that contain less than or equal to k base tables. K($\leq$n) is a parameter to the algorithm.

2) Find costs for these by contacting the data sources using a single round of communication.

3) Choose one subplan (and the corresponding k-way join) out of all the subplans for these k- way joins using an evaluation function and throw away all the other subplans.

4) If not finished yet, repeat the optimization procedure using this intermediate relation and the rest of the relations that are not part of it.

IDP-M(k,m) : This is a natural generalization of the earlier variant . It differs from IDP in that instead of choosing one -way join out of all possible k-way joins, we keep such joins and throw the rest away,where is another parameter to the algorithm. The motivation behind this algorithm is that the first variant is too aggressive about pruning the plan space and may not find a very good plan in the end.

Table 1 shows the number of bid requests required for different query graph shapes. The total cost of costing here also depends on the number of rounds of communication required(n/k).

3.3 Two-phase Optimization

Two-phase optimization has been used extensively (M. N. Garofalakis & Y. E. Ioannidis. 2005.) in distributed and parallel query optimization mainly because of its simplicity and the ease of implementation.

This algorithm works in two phases:

Phase 1: Find the optimal plan using a System R-style algorithm extended to search through the space of bushy plans as well. This phase assumes that all the relations are stored locally and uses a traditional cost model for estimating costs. If the physical database design is known (e.g., existence of indexes or materialized views on the underlying data sources), then this information is used during the optimization process.

Phase 2: Schedule the optimal plan found in the first phase. This is done by first requesting the costs of executing the operators at the involved data sources from the bidders and then finding the optimal schedule using an exhaustive algorithm.

4. Experimental Study

In this section, we present our initial experimental results comparing the performance of various optimization algorithms that we discussed above. The main goals of this experimental study are to motivate the need for dynamic costing as well as to understand the trade-offs involved in the optimization process.

4.1 Experimental Setup

We have implemented the algorithms described earlier in a modified version of the Cohera federated database system, a commercialization of the Mariposa research system. The experiments were carried out on a stand-alone Windows XP machine running on a 466MHz Pentium with 256 MB of Memory. Both the optimizer and the underlying data sources connect to a Microsoft SQLServer running locally on the same machine. A set of bidders was started locally as required for the experiments. We simulate a net work by using the following message cost model: A mes-sage of size N bytes takes $\alpha+\beta*N$ time to reach the other end, where $\alpha$is the startup cost and$\beta$is the cost per byte. We experimented with two different communication settings corresponding to a local area network (LAN) withα=10ms, β=0.001ms and a wide area network (WAN) with α=120ms, β=0.005.

4.2 Optimization Quality

In this section, we will see how the different optimization algorithms perform under various circumstances and further motivate the need for better costing in the optimization process. The algorithms that we compare are Exhaustive (E)(Section 3.1), Two-Phase (2PO)(Section 3.4) and four variants of Iterative Dynamic Programming(Section 3.3),IDP(4), IDP(3), IDP-M(4,5) and IDP-M(3,5).

Figure 2(i) showsthe mean for these scaled costs as well as the standard deviation for these 40 random runs for each of the algorithms. We show only the results for a wide-area network since the trends observed are similar in the local area network. As we can see, though the two-phase algorithm performs some what worse than the exhaustive algorithm, in many cases it does find the optimal plan.

Figure 2(ii) shows the relative performance of these optimization algorithms in this case. As we can see, for two of the queries, Query 1 and Query 4, 2PO finds a much worse plan than the optimal plan. For queries 2 and 3, on the other hand, it performs almost as well as the optimal plan. The main reason for this is that since we have extended the first phase of the two-phase optimizer to search through bushy plans as well, it finds bushy plans for these queries and as such, they can be effectively parallelized. The IDP variants perform almost the same as the earlier experiment, though with higher deviations in some cases.

Figure 2(iii) shows the results from this experiment for queries Q1 and Q2. The results for Q3 were similar, whereas Q4 is not affected by this view. As we can see, 2PO consistently produces a plan much worse than the optimal plan. IDP variants once again show unpredictable results with IDP-4 performing very well, since it is able to take advantage of the view, whereas IDP-3 performs almost as bad as two-phase optimizer, since the restriction of choosing the lowest cost subplan of size 3 makes it choose the wrong plan.

4.3 Discussion

As we can see, the IDP variants are quite sensitive to their parameters, but in almost all cases, at least one of IDP(3) and IDP(4) performed better than the two-phase optimizer. This suggests that a hybrid of two such algorithms might be the algorithm of choice in the federated environment, especially when the physical designs of the underlying data sources may be hidden from the optimizer.

The most surprising fact that arises from our experiments is that the two-phase optimization algorithm does not perform much worse than the exhaustive algorithm for total cost optimization if the physical database design is known to the optimizer.

References

A. Tomasic, R. Amouroux, P. Bonnet, O. Kapitskaia, (2007). H. Naacke, & L. Raschid. The distributed information search component (DISCO) and the world wide web. In SIGMOD.

D. Kossmann & K. Stocker. (2006). Iterative dynamic programming: a new class of query optimization algorithms. ACM TODS.

J. M. Hellerstein, M. Stonebraker, & R. Caccia. (2003). Independent, open enterprise data integration. IEEE Data Engineering Bulletin.

L. Knight.(2006). “the e-market maker revolution”, dataquest inc. http://www.netmarketmakers.com/documents/perspective1.pdf, 2006.

M. N. Garofalakis & Y. E. Ioannidis. (2005). Parallel query scheduling and optimization with time- and space-shared resources. In VLDB.

W. Hong andM. Stonebraker. (2001). Optimization of parallel queryexecution plans in xprs. In PDIS.

Table 1. Number of Bid Requests

Shape of the query	DP w/o multi-joins	DP with multi-joins	IDP(k)[5] (for k $\ll$ n)	IDP-M(k,m)[5] (for k $\ll$ n)
Chain	$O(n^3)$	$O(n2^n)$	$O(n^2k)$	$O(mn^2k)$
Star	$O(n2^n)$	$O(3^{n+1})$	$O(n^k)$	$O(mn^k)$
Clique	$O(3^n)$	$O(2^{2^n})$[5]	$O((2n)^k)$	$O(m(2n)^k)$

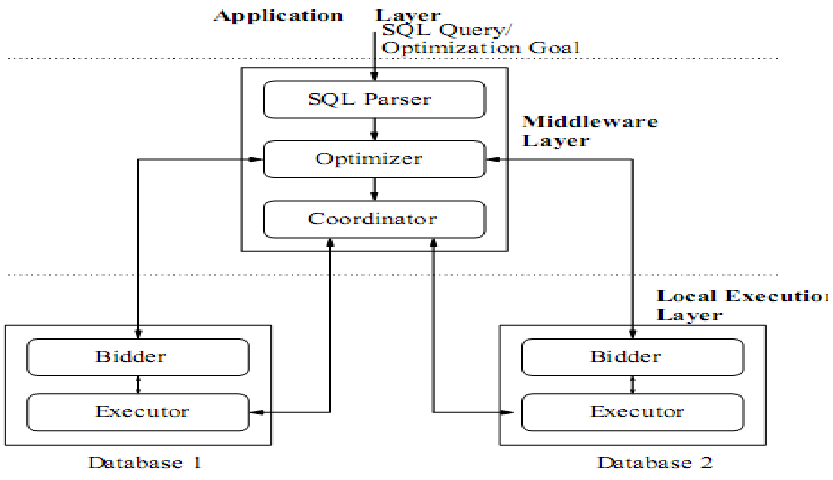

Figure 1. System Architecture

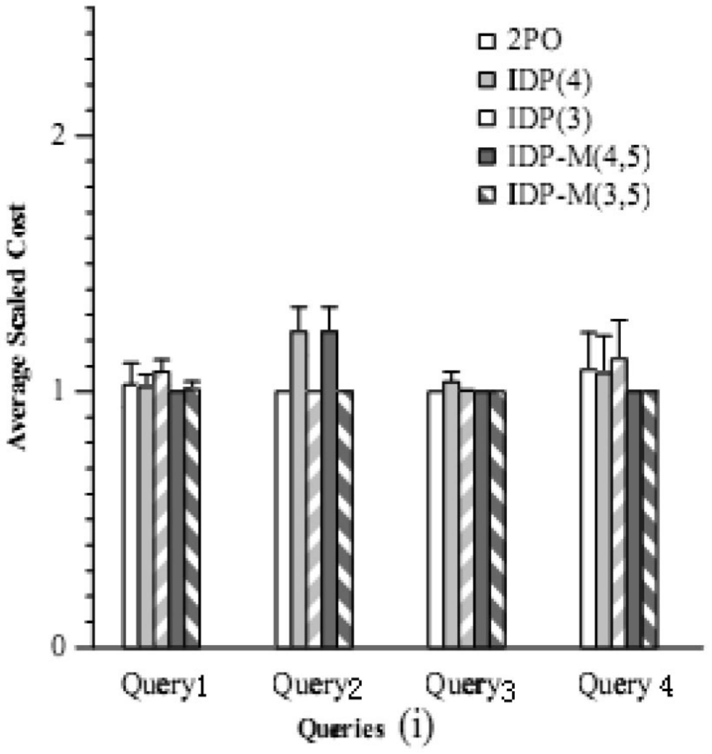

Figure2(i) Total cost optimization

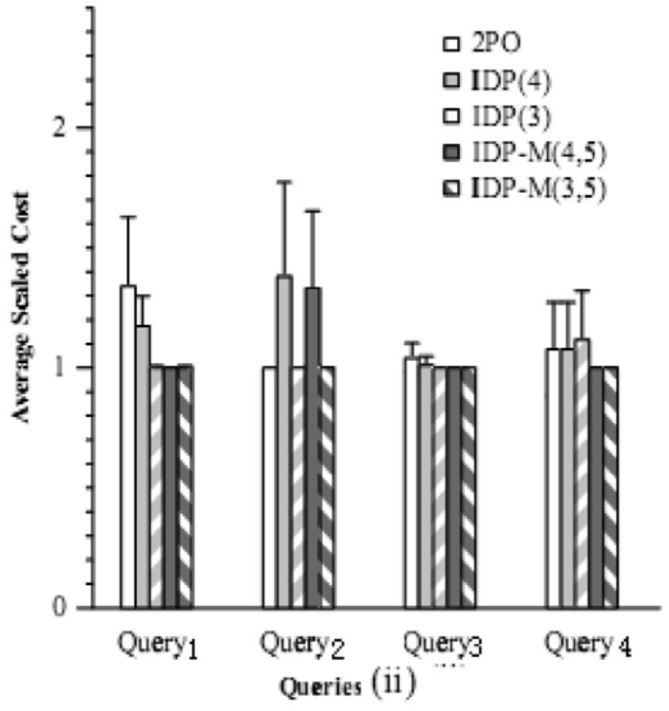

Figure2 (ii) Response time optimization under uncertain load conditions

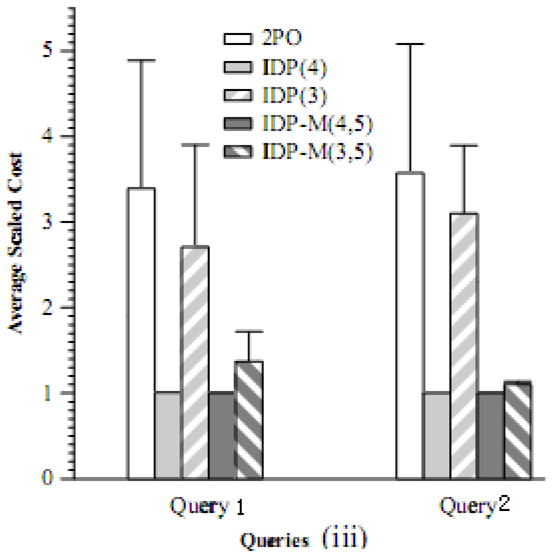

Figure 2 (iii) Total cost optimization in presence of materialized views