Building Multistate Cost Models for Dynamic Multidatabase Environments *

Qiang Zhu Yu Sun Satyanarayana Motheramgari
Department of Computer and Information Science
The University of Michigan, Dearborn, MI 48128, USA
{qzhu, yusun, motheram}@umich.edu

Wen-Chi Hou
Department of Computer Science
Southern Illinois University, Carbondale, IL 62901, USA
hou@cs.siu.edu

Suyun Chen
Information Management
BMO Financial Group, Toronto, ON M1W 3E8, Canada
Suyun.Chen@bmo.com

Abstract

Local cost estimation is essential to global query optimization in a multidatabase system (MDBS). The techniques suggested in the literature to develop local cost models in an MDBS are suitable for a static environment. Many dynamic environmental factors have a significant effect on a query cost. In this paper, we introduce a multistate query sampling method to develop local cost models for a dynamic multidatabase environment. The key idea is to divide the system contention level, which reflects the combined net effect of dynamic factors on a query cost, in a dynamic environment into a number of discrete contention states based on the cost of a probing query and then incorporate a qualitative variable indicating the contention states into a cost model. To determine an appropriate set of contention states for a dynamic environment, two algorithms based on iterative uniform partition and data clustering, respectively, are introduced. To build an effective cost model with a qualitative variable for a dynamic environment, we extend the techniques from our previous (static) query sampling method, including query sampling, automatic variable selection, regression analysis, and model validation. Experimental results demonstrate that the presented multistate query sampling method is quite promising in developing useful cost models for a dynamic multidatabase environment. Some issues on query optimization based on the developed multistate cost models are also discussed. Our study shows that multistate cost models improve the effectiveness of query optimization. In addition, how to use the cost estimates from multistate cost models to estimate query costs in more complex situations is suggested.

Keywords: multidatabase, query optimization, cost model, dynamic environment, contention states, query sampling, regression analysis

---

* Research supported by the US National Science Foundation under Grant # IIS-9811980 and The University of Michigan under OVPR and UMD grants.

1. Introduction

A multidatabase system (MDBS) integrates data from multiple local (component) databases and provides users with a uniform global view of data. A global user can issue a (global) query on an MDBS to retrieve data from multiple databases without having to know where the data is stored and how the data is retrieved. How to process such a global query efficiently is the task of global query optimization.

A major challenge, among others [6,9,10,11,18], for global query optimization in an MDBS is that some necessary local information such as local cost models may not be available at the global level due to local autonomy preserved in the system. However, the global query optimizer needs such information to decide how to decompose a global query into local (component) queries and where to execute them. Hence, methods to derive cost models for an autonomous local database system (DBS) at the global level are required. Several such methods have been proposed in the literature.

In [5], Du *et al*. proposed a calibration method to deduce necessary local cost parameters. The key idea to construct a local synthetic calibrating database (with some special properties), and then run a set of special queries against this database. The access method used for executing such a query is known because of the special properties of the database and query. Cost metrics for the queries are recorded and used to deduce the coefficients in the cost formulas for the access methods supported by the underlying local database system by using the properties of the database and queries. In [7], Gardarin *et al*. extended the above method so as to calibrate cost models for object-oriented local database systems in an MDBS.

Although Du's calibration method was the first technique that demonstrates the possibility of estimating local cost parameters at the global level in an MDBS, it has several drawbacks. First of all, it may be impossible (or not allowed) to create a synthetic calibrating database at a local site in an MDBS. Unlike a traditional distributed database system (DDBS), local autonomy in an MDBS may prevent creating a test database at a local site. Secondly, cost parameters deduced by using a synthetic calibrating database may not be valid for real databases because of different data distributions, database sizes, file structures, adjustable local system parameters and so on. Thirdly, the deduced cost formulas cannot be applied if the local access method for a query is unknown, which is frequently the case in an MDBS. The calibration method deduces a set of cost formulas, one for each access method. To estimate the cost of a query, the access method used for the query needs to be known so that the right cost formula can be applied.

To overcome the above shortcomings, Zhu and Larson proposed a query sampling method in [25, 26, 27]. The key idea is as follows. Local queries that can be performed on a local DBS in an MDBS are grouped into homogeneous classes first, based on some information available at the global level in an MDBS such as the characteristics of queries, operand tables and the underlying local DBS. A sample of queries are then drawn from each query class and run against the actual user local database. The costs of sample queries are used to derive a cost model for each query class by multiple regression analysis. The cost model parameters are kept in the MDBS catalog and utilized during query optimization. To estimate the cost of a local query, the class to which the query belongs is first identified. The corresponding cost model is retrieved from the catalog and used to estimate the cost of the query. Based on the estimated costs, the global query optimizer chooses a good execution plan for a global query. Since the query sampling method (1) can capture the performance behavior of queries on the actual databases, (2) does not require the creation of synthetic databases, and (3) identifies the right cost model for a

query based on available information, it overcomes the aforesaid drawbacks of the calibration method.

There are several other suggested approaches to tackling the local query cost estimation problem in an MDBS. In [24], Zhu and Larson introduced a fuzzy method based on fuzzy set theory to derive cost models utilizing fuzzy information in an MDBS. In [12], Naacke *et al.* suggested an approach to combining a generic cost model with specific cost information exported by wrappers for local DBS's. In [1], Adali *et al*. suggested to maintain a cost vector database to record cost information for every query issued to a local DBS. Cost estimation for a new query is then based on the costs of similar queries. In [17], Roth *et al*. introduced a practical framework for costing in the Garlic federated system.

All the previously proposed methods only considered a static system environment, i.e., assuming no significant change over time. However, in reality, many factors in an MDBS environment such as contention factors (e.g., number of concurrent processes), database physical characteristics (e.g., index clustering ratio), and hardware configurations (e.g., memory size) may change significantly over time. Hence, a cost model derived for a static system environment may not give good cost estimates for queries in a dynamic environment. Figure 1 shows how the cost of a sample query is affected by the number of concurrent processes in a dynamic system environment. We can see that the cost of the same query can dramatically change (from 3.80 sec. to 124.02 sec.) in a dynamic environment. This raises an interesting research issue, that is, how to derive cost models that can capture the performance behavior of queries in a dynamic environment.

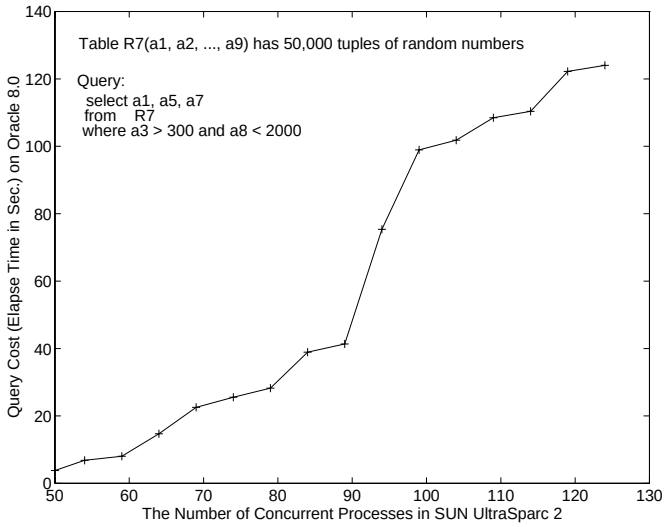


Figure 1: Effect of Dynamic Factor on Query Cost

In this paper, we propose a qualitative approach to deriving a cost model that can capture the performance behavior of queries in a dynamic environment. The key idea is as follows. We notice that there are numerous dynamic factors (such as the CPU and I/O loads) that affect a query cost. To simplify the development of a cost model for a dynamic environment, our approach considers the combined net effect of dynamic factors on a query cost together rather than individually. The system contention level that reflects such a combined effect is gauged by

the cost of a probing query. The larger the probing query cost, the higher the contention level. To capture such contention information in a cost model, we divide the system contention level (based on the cost of a probing query) in a dynamic environment into a number of discrete contention states (such as high, medium and low contention states) and use a qualitative variable to indicate the contention states in the cost model. This qualitative variable effectively allows the cost model to adopt different coefficients for different contention states so that its cost estimates can better reflect the dynamic environment. To determine an appropriate set of contention states for a dynamic environment, two algorithms, called the iterative uniform partition with merging adjustment (*IUPMA*) and the iterative clustering with merging adjustment (*ICMA*), respectively, are introduced. The former is used for general cases, while the latter is specifically designed for a dynamic environment with the contention level following a non-uniform distribution with clusters. Our previous query sampling method in [25, 26, 27] is extended so as to develop the regression cost model incorporating the qualitative variable (i.e., the qualitative cost model) for a dynamic environment. In other words, the coefficients of the qualitative cost model are obtained from the regression analysis based on observed sample query costs in the dynamic environment. Our approach in this paper is therefore an extension of our previous query sampling method. In this paper, we call our previous method as the static query sampling method and the new approach in this paper as the multistate query sampling method. In fact, the static method is a special case of the multistate one when only one contention state is allowed (so it can also be called the single-state query sampling method).

The rest of the paper is organized as follows. Section 2 analyzes the dynamic factors at a local site in an MDBS. Section 3 discusses how to develop a regression model with a qualitative variable and how to determine contention states of a qualitative variable for a dynamic environment. Section 4 extends our previous static query sampling method so as to derive cost models with a qualitative variable for different query classes in a dynamic environment. It also briefly discusses how to perform query optimization based on multistate cost models. Section 5 shows some experimental results. Section 6 summarizes the conclusions.

2. Dynamic Environmental Factors

In an MDBS, many environmental factors may change over time[1]. Some may change more often than others. They can be classified into the following three types based on their changing frequencies.

- *Frequently-changing factors*: The main characteristic of this type of factors is that they change quite often. Examples of such factors are the CPU load, I/O load, and size of used memory space, etc. The operating system at a local site typically provides commands (such as *top*, *ps* and *iostats* in UNIX) to display system statistics reflecting such environmental factors. Table 1 lists some system statistics in UNIX.
- *Occasionally-changing factors*: These factors change occasionally. Examples of such factors include local database management system (DBMS) configuration parameters (e.g., the number of buffer blocks, and the shared buffer pool size), local database physical/conceptual schemas (e.g., new indexes and new tables/columns), and local

[1] Since we concern ourselves with local cost models for an MDBS, only dynamic factors at local sites are considered. In general, there are also dynamic environmental factors for the network in an MDBS. Some of them were considered in [20].

hardware configurations (e.g., physical memory size). Note that some other factors such as the local database size, physical data distribution, and index clustering ratio may change quite frequently. However, they may not have an immediate significant impact on a query cost until such changes accumulate to a certain degree. Thus we still consider these factors as occasionally-changing factors. The changes of most occasionally-changing factors can be found via checking the local database catalog and/or system configuration files.

Type	Statistics for Frequently-Changing Environmental Factors	
CPU Statistics	• *rp* — number of running processes; • *tp* — number of stopped processes; • *us* — percentage of user time; • *id* — percentage of idle time • ld_1, ld_2, ld_3 — load averages for the past 1, 5, and 15 minutes, respectively	• *sp* — number of sleeping processes • *zp* — number of zombie processes • *sy* — percentage of system time
Memory Statistics	• *am* — available memory; • *sm* — shared memory; • *as* — available swap; • *fs* — free swap; • *si* — amount of memory swapped in;	• *um* — used memory • *bm* — buffer memory • *us* — used swap • *as* — cached swap • *so* — amount of memory swapped out
I/O Statistics	• *bi* — number of reads per sec.; • *du* — percentage of disk utilization	• *bo* — number of writes per sec.
Other Statistics	• *nu* — number of current users; • *cs* — number of context switches per sec.;	• *in* — number of interrupts per sec. • *sc* — number of system calls per sec.

Table 1: System Statistics for Frequently-Changing Factors in Unix

- *Steady factors*: These factors rarely change. Examples of such factors are the local DBMS type (e.g., relational or object-oriented), local database location (e.g., local or remote), and local CPU speed (e.g., 2.8GHz). Although these factors may affect a cost model, the chance for them to change is very small.

Clearly, the steady factors usually do not cause a problem for a query cost model. If significant changes for such factors occur at a local site, they can be handled in a similar way as described below for the occasionally-changing factors.

For the occasionally-changing factors, a simple and effective approach to capturing them in a cost model is to invoke the static query sampling method periodically or whenever significant changes for these factors have occurred. Since these factors do not change very often, rebuilding cost models from time to time to capture them is acceptable. However, this approach cannot be used for the frequently-changing factors because frequent invocations of the static query sampling method would significantly increase the system load and the cost model maintenance overhead. On the other hand, if a cost model cannot capture the dramatic changes in a system environment, poor query cost estimates may be used by the query optimizer, resulting in inefficient query execution plans.

Theoretically speaking, to capture the frequently-changing factors in a cost model, one approach is to include all explanatory variables that reflect such factors in the cost model. However, this approach encounters several difficulties. First, the ways in which these factors affect a query cost are not clear. As a result, the appropriate format of a cost model that directly includes the relevant variables is hard to determine. Second, the large number of such factors (see Table 1) makes a cost model too complicated to derive or maintain even if the first difficulty

could be overcome. In the rest of this paper, we introduce a feasible method to capture the frequently-changing factors in a cost model

3. Regression with Qualitative Variable

As mentioned before, the key idea of our method is to determine a number of contention states for a dynamic environment and use a qualitative variable to indicate the states. A cost model with the qualitative variable can be used to estimate the cost of a query in different contention states. The issues on how to include a qualitative variable in a cost model and how to determine an appropriate set of system contention states are discussed in this section.

3.1. Qualitative variable

To simplify the problem, we consider the combined effect of all the frequently-changing factors on a query cost together rather than individually. Although these dynamic factors may change differently in terms of the changing frequency and degree, they all contribute to the contention level of the underlying system environment. The cost of a query increases as the contention level. Hence the contention level can be gauged by the cost of a probing query. Depending on the observed probing query cost, the system contention level can be divided into a number of discrete states (categories) such as "*High Contention*" (S_H), "*Medium Contention*"(S_M), "*Low Contention*" (S_L), and "*No Contention*" (S_N). A variable W taking a value from such possible discrete states (e.g., S_H) is a qualitative variable[2] that can be used to indicate the current contention state for the environment. This qualitative variable, therefore, reflects the combined effect of foregoing frequently-changing environmental factors. A cost model incorporating such a qualitative variable can capture the dynamic environmental factors to a certain degree.

As shown in [25, 27], a statistical relationship between the query cost and its affecting factors such as operand and result table sizes can be established by multiple regression. The established relationship can be then used as a cost model to estimate query costs. However, only quantitative variables (e.g., operand and result table sizes) were considered in such regression and other types of existing cost models. To incorporate the foregoing qualitative variable W indicating system contention states into a query cost model, a new technique is required.

Notice that a qualitative variable can be represented by a set of indicator variables. For example, the above contention state variable W with four states can be represented by three indicator variables: Z_1, Z_2, and Z_3, where $Z_1 = 1$ indicates $W = S_H$, while $Z_1 = 0$ indicates $W \neq S_H$; $Z_2 = 1$ indicates $W = S_M$, while $Z_2 = 0$ indicates $W \neq S_M$; $Z_3 = 1$ indicates $W \neq S_L$, while $Z_3 = 0$ indicates $W \neq S_L$; and $Z_1 = Z_2 = Z_3 = 0$ indicate $W = S_N$. Note that no more than one indicator variable can be 1 simultaneously (i.e., W can take only one state at a time). Table 2 shows[3] how the indicator variables indicate different states of W. In general, a qualitative variable that has m categories (states) need $m-1$ indicator variables to represent it. As we will see, using such a set of indicator variables allows us to easily express the adjustment of a regression coefficient for each contention state in a cost model although other ways to use indicator variables are also possible.

[2] In general, variables can be classified as either quantitative ones or qualitative ones. A quantitative variable (e.g., monthly salary) takes values on a well-defined scale (e.g., 2300.00, 3100.50), while a qualitative variable (e.g., person's gender) only have several discrete categories (states) as its possible values (e.g., 'male', 'female').
[3] Only feasible combinations of the indicator values are allowed.

States	Z_1	Z_2	Z_3
S_H	1	0	0
S_M	0	1	0
S_L	0	0	1
S_N	0	0	0

Table 2: State Representations Using Indicator Variables

3.2. Qualitative regression cost model

The cost of a query usually consists of (1) initialization cost (for example, to move a disk head to the right position); (2) I/O cost (for example, to fetch a tuple from an operand table); and (3) CPU cost (for example, to evaluate the qualification condition for a given tuple). A typical cost model for a unary query class in a static environment may look like:

$$Y = B_0^0 + B_1^0 * N_U + B_2^0 * RN_U \quad (1)$$

where Y is the query cost; N_U and RN_U are the cardinalities (sizes) of the operand and result tables, respectively; and B_0^0, B_1^0 and B_2^0 are the parameters (coefficients) representing the initialization cost, the cost for retrieving a tuple from the operand table, and the cost for processing a tuple in the result table, respectively. Both B_1^0 and B_2^0 may reflect the I/O cost as well as the CPU cost. Therefore, the initialization cost affects the intercept coefficient (i.e., the 1st term) in a cost model, while the I/O and CPU costs affect the slope coefficients (i.e., the coefficients of the 2nd and 3rd terms) in the cost model.

In general, let Y be the response variable (i.e., query cost) and $X_1, X_2, \ldots\ldots, X_n$ be the (quantitative) explanatory variables (e.g., operand and result table sizes) in a regression query cost model. The regression cost model in a static environment is of the following general form:

$$Y = B_0^0 + \sum_{i=1}^{n} B_i^0 X_i \ , \quad (2)$$

where B_i^0's ($1 \le i \le n$) are regression coefficients that can be estimated based on observed costs of sample queries in the environment.

However, as mentioned before, a static cost model may not be useful for a dynamic environment. Notice that the contention level of a system can significantly affect not only the initialization cost but also the I/O and CPU costs of a query because the resources like the disk, I/O bandwidth and CPU are shared by multiple processes. As a result, both the intercept and slope coefficients in a query cost model may change when the system contention level changes. Assume that the underlying dynamic environment has m contention states: $S_m, S_{m-1}, \ldots, S_1$, with S_m being the lowest contention state and S_1 being the highest[4]. Our key idea to develop a cost model for such a dynamic environment is to adjust the coefficients of the cost model for different contention states in the environment. For example, if $B_0^0, B_1^0, B_2^0, \ldots, B_n^0$ are the coefficients of the cost model in the lowest contention state S_m (called the base cost model), then

[4] A descending index is used here to simplify the description of algorithms and derived cost models in the paper.

$B_0^0 + B_0^{m-1}, B_1^0 + B_1^{m-1}, ..., B_n^0 + B_n^{m-1}$ are the coefficients of the cost model in contention state S_{m-1}, where $B_0^{m-1}, B_1^{m-1}, B_2^{m-1}, ..., B_n^{m-1}$ are the adjustments (values) to the respective coefficients $B_0^0, B_1^0, B_2^0, ..., B_n^0$ of the base cost model to reflect the contention state change from S_m to S_{m-1}. Similarly, $B_0^0 + B_0^{m-2}, B_1^0 + B_1^{m-2}, B_2^0 + B_2^{m-2}, ..., B_n^0 + B_n^{m-2}$ are the coefficients of the cost model in contention state S_{m-2} and $B_0^{m-2}, B_1^{m-2}, B_2^{m-2}, ..., B_n^{m-2}$ are the relevant adjustments to the base cost model to reflect the contention state change from S_m to S_{m-2}. In general, $B_0^0 + B_0^i, B_1^0 + B_1^i, B_2^0 + B_2^i, ..., B_n^0 + B_n^i$ are the coefficients of the cost model in contention state S_i ($1 \le i \le m-1$), where $B_0^i, B_1^i, B_2^i, ...B_n^i$ are the adjustments to the coefficients of the base cost model for contention state S_i. The base coefficients and all the relevant adjustments of the cost model for different contention states can be estimated via the regression analysis based on observed data for sample queries.

Let qualitative variable W with the above m states be represented by indicator variables Z_{m-1}, $Z_{m-2}, \cdots, Z_1$, where the jth state ($1 \le j \le m-1$) is indicated by $Z_j = 1$ and $Z_i = 0$ for $i \ne j$ and the mth state is indicated by $Z_i = 0$ for all $1 \le i \le m-1$. To incorporate a qualitative variable representing the system contention states into a query cost model, we adopt the following regression model with a qualitative variable (represented by a set of indicator variables):

$$Y = \underbrace{(B_0^0 + \sum_{j=1}^{m-1} B_0^j Z_j)}_{\text{intercepts}} + \underbrace{\sum_{i=1}^{n} (B_i^0 + \sum_{j=1}^{m-1} (B_i^j Z_j) X_i}_{\text{slopes}} \quad . \qquad (3)$$

Specifically, the intercept coefficient for the jth state of the qualitative variable is $B_0^0 + B_0^j$ ($j = m, m-1, ..., 1$; and assuming $B_0^m = 0$), where B_0^j is the adjustment of B_0^0 for the jth contention state; and the ith slope coefficient (i = 1, 2, $\cdots$,n) for the jth state of the qualitative variable is $B_i^0 + B_i^j$ (j = 1, 2, $\cdots$, m; and assuming $B_i^m = 0$)), where B_i^j is the adjustment of B_i^0 for the jth contention state. Table 3 shows the adjusted cost model for different contention states.

State	$Zm-1$	Z_{m-2}	...	Z_1	Adjusted Cost Model In the Corresponding Contention State
S_m	0	0	…	0	$Y = B_0^0 + B_1^0 * X_1 + B_2^0 * X_2 + ...$
S_{m-1}	1	0	…	0	$Y = (B_0^0 + B_0^{m-1}) + (B_1^0 + B_1^{m-1}) * X_1 + (B_2^0 + B_2^{m-1}) * X_2 + ...$
S_{m-2}	0	1	…	0	$Y = (B_0^0 + B_0^{m-2}) + (B_1^0 + B_1^{m-2}) * X_1 + (B_2^0 + B_2^{m-2}) * X_2 + ...$
...	…	…	…	…	…..
S_1	0	0	…	1	$Y = (B_0^0 + B_0^1) + (B_1^0 + B_1^1) * X_1 + (B_2^0 + B_2^1) * X_2 + ...$

Table 3: Adjusted Cost Models in Different Contention States from Formula (3)

For example, we can extend cost model (1) from a static environment to a dynamic environment following (3). Assuming the underlying dynamic environment has three contention states (i.e., S_3, S_2 and S_1), cost model (1) is extended as the following based on (3):

$$Y = (B_0^0 + B_0^2 * Z_2 + B_0^1 * Z_1) + (B_1^0 + B_1^2 * Z_2 + B_1^1 * Z_1) * N_U + (B_2^0 + B_2^2 * Z_2 + B_2^1 * Z_1) * RN_U \; . \qquad (4)$$

When the current contention state is S_3 (i.e., $Z_2 = Z_1 = 0$), cost model (4) boils down to the base

model:

$$Y = B_0^0 + B_1^0 * N_U + B_2^0 * RN_U . \quad (5)$$

If the contention state is S_2 (i.e., $Z_2=1$ and $Z_1=0$), cost model (4) becomes:

$$Y = (B_0^0 + B_0^2) + (B_1^0 + B_1^2) * N_U + (B_2^0 + B_2^2) * RN_U . \quad (6)$$

We can see that B_0^2, B_1^2 and B_2^0 are the adjustments to the base coefficients B_0^0, B_1^0 and B_2^0, respectively, for contention state S_2.

Using indicator variables in the way illustrated by Table 2 allows us to easily identify the adjustments of the intercept and slope coefficients of the cost model for a contention state in the dynamic environment. Specifically, the adjustment to base coefficient B_i^0 in cost model (3) for the jth contention state is simply coefficient B_i^j of indicator variable Z_j ($0 \le i \le n$ and $1 \le j \le m-1$). Note that no coefficient adjustments are needed for the mth state since B_i^0's are assumed to be developed for that state. On the other hand, the way to use indicator variables for a qualitative variable is not unique. However, if one contention state S' were indicated by two indicator variables $Z' = Z'' = 1$, the adjustment of a (intercept or slope) coefficient of the cost model for contention state S' would have to be split between the coefficients of Z' and Z'', making their individual meanings unclear. Furthermore, if less than $m - 1$ indicator variables were used, the coefficient of one indicator variable might have to be used to adjust the relevant (intercept or slope) coefficient of the cost model for multiple consternation states. Since less parameters are used to adjust more cases, some tradeoffs will have to be made, which may lead to a less accurate cost model.

All the coefficients B_i^j's in cost model (3) can be obtained via the regression analysis, as we will see in the remaining sections. Since cost model (3) allows different adjustments for different contention states in a dynamic environment, it can give better cost estimates than those given by a static cost model in the dynamic environment, as we will see in Section 5.

3.3. Determining system contention states

Combining multiple dynamic environmental factors into a composite qualitative variable with a number of discrete contention states greatly simplifies the development of a cost model for a dynamic environment. The question now is how to determine an appropriate set of system contention states for a dynamic environment.

Two extremes

There are two extremes in determining a set of contention states. One extreme is to consider only one contention state for the system environment. A cost model developed in such a case is useful if the system environment is static. This, in fact, was the case that the static query sampling method assumed. However, as pointed out before, a real system environment may change dynamically over time. Using one contention state is obviously insufficient to describe such a dynamic environment. For a dynamic environment, usually, the more the contention states are considered, the better a cost model. In principle, as long as we consider a sufficient number of contention states for the environment, we can get a satisfactory cost model. Hence, the other extreme is to consider an infinite number of contention states. However, the more the contention

states are considered, the more the indicator variables are needed in the cost model. The number of coefficients that need to be determined in a cost model therefore increases. Hence, if too many contention states are considered, the cost model can be very complicated, which is not good for either its development or its maintenance. In practice, as we will see in Section 5, a small number of contention states (three to six) are usually sufficient to yield a good cost model.

Determining states via iterative uniform partition

Notice that, for a given query, its cost increases as the system contention level increases (see Figure 1). Based on this observation, we can use the cost of a probing query to gauge the system contention level[5]. The range of the probing cost (therefore, the contention level) is divided into subranges, each of which represents a (discrete) contention state for the dynamic environment.

Let the cost C_{Qp} of probing query Q_p fall in the range $[C_{min}, C_{max}]$ in a dynamic environment. A simple way to determine the system contention states is to partition range $[C_{min}, C_{max}]$ into subranges with an equal size. In other words, to determine m contention states $S_m, S_{m-1}, ..., S_1$, we divide range $[C_{min}, C_{max}]$ into m subranges $I_i = [C_{min}+(m-i)*D, C_{min}+(m-i+1)*D)$ and $I_1 = [C_{min} + (m-1)*D, C_{min} +m*D]$ where $i = m, m-1,..., 2$ and $D = (C_{max} - C_{min})/m$. The system environment is said to be in contention state S_i if $C_{Qp} \in I_i$ ($i = m, m- 1,..., 1$). To obtain more system contention states, we can simply increase m. Hence, $\{I_m, I_{m-1},..., I_1\}$ yields a set Γ_m of system contention states for the dynamic environment.

Using this partition, it is easy to determine the system contention state in which a query is executed. Let $SP = \{ Q_j \mid j = 1, 2, ..., t \}$ be a set of sample queries which are performed in the dynamic environment and whose observed data (e.g., execution time, operand size and result table size, etc.) are to be used to derive a regression cost model for a query class. To determine the system contention state $S_{Q\,j}$ in which sample query Q_j is executed, the cost C_{Qp} of probing query Q_p in the same environment is measured. If $C_{Qp} \in I_i$ ($1 \le i \le m$), we say $S_{Q\,j} = S_i$. We call the cost of a probing query associated with a sample query is a sampled probing query cost (value).

One basic question is how to determine a proper m. Another question is how to eliminate some unnecessary separations of subranges. Clearly, if queries in neighboring contention states S_{j-1} and S_j (for some j) have a similar performance behavior, separating S_{j-1} and S_j is unnecessary. The determination of system contention states should balance the accuracy and the simplicity (hence low maintenance overhead) of a derived cost model.

To solve these two problems, the following algorithm is used to improve the foregoing direct uniform partition:

Algorithm 3.1: Contention States Determination via Iterative Uniform Partition with Merging Adjustment (*IUPMA*)
Input: Observed data of sample queries and their associated probing query costs
Output: A set of system contention states[6]

---

[5] Our experiments showed that most queries, except the ones with extremely small cost (e.g., less than a second), can well serve as a probing query to gauge the system contention level.
[6] In fact, the algorithm integrates the contention states determination procedure with the cost model development procedure (to be discussed in Section 4). As a result, a cost model is also produced as part of the output of the algorithm.

Method:

1. **begin**
2. Derive a qualitative regression model with one contention state using sample query data;
3. Let R^2_{new} be the coefficient of total determination of the current regression model;
4. Let s_{new} be the standard error of estimation of the current regression model;
5. $m := 1$;
6. **do**
7. $R^2_{old} := R^2_{new}$; $s_{old} := s_{new}$
8. $m := m + 1$;
9. Obtain a set Γ_m of m contention states for the system environment via the straightforward uniform partition;
10. Derive a qualitative regression model with m contention states using sample query data;
11. Let R^2_{new} be the coefficient of total determination for the current regression model;
12. Let s_{new} be the standard error of estimation of the current regression model;
13. **until** ($\left|(R^2_{new} - R^2_{old}) / R^2_{old}\right|$ and $\left|(s_{new} - s_{old}) / s_{old}\right|$) are sufficiently small or m is too large;
14. $m := m - 1$;
15. Let S_j ($j = m, m-1, ..., 1$) represent the current m contention states in Γ_m;
16. Let $A^j_i = B^0_i + B^j_i$ ($i = 0, 1, ..., n$) be the adjusted coefficient of ith variable X_i for state S_j in the general model in (3), where $X_0 \equiv 1$ is a dummy variable for the intercept coefficient;
17. **for** $k = m$ **down to 2 do**
18. $E_k := \max_{i \in \{0,1,2,...,n\}} \left\{ \left| (A^{k-1}_i - A^k_i) / A^k_i \right| \right\}$
19. **if** E_k is too small **then**
20. tag that states S_k and S_{k-1} should be merged;
21. **end for**
22. **if** some states are tagged to be merged **then**
23. Derive a qualitative regression model with new merged states using sample query data;
24. **goto** step 15;
25. **end if;**
26. **return** the current set Γ_m of contention states;
27. **end.**

There are two phases in Algorithm 3.1. The first phase is to determine a set of contention states via the uniform partition. The algorithm iteratively checks each qualitative regression model with an incremental number of contention states until (1) the model cannot be significantly improved in terms of the coefficient of total determination[7] R^2 and the standard error of estimation[8] s; or (2) too many contention states have been generated. Condition (2) is used here to prevent that a derived cost model becomes too complicated (in terms of the number of variables involved). The

set of contention states obtained from the first phase are based on the uniform partition of the probing query cost range (see Figure 2). The partition does not consider whether two neighboring states actually have significantly different effects on the cost model or not. It is possible that some neighboring states have only slight different effects on the cost model. If so, the neighboring states should be merged into one to simplify the cost model. Such a merging adjustment is done during the second phase of the algorithm. If the maximum of relative errors of the corresponding pairs of adjusted coefficients (i.e., $B_i^0 + B_i^k$, and $B_i^0 + B_i^{k-1}$, $i = 0,1,...,n$) for two states S_k and S_{k-1} is too small, these two states are considered not to have significantly different effects on the cost model. The subranges in the final adjusted partition of the probing query cost range may not have an equal size.

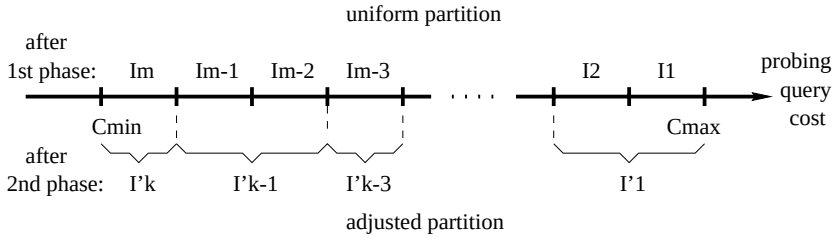


Figure 2: Contention States Determination via *IUPMA*

Determining states via data clustering

Note that, to apply the *IUPMA* algorithm, a set of sample queries need to be run at sampled system contention level points (measured by the sampled probing query costs). Typically, the sampled system contention level points (i.e., the sampled probing query costs) can be chosen randomly from their range [C_{min}, C_{max}]. However, in reality, the actual (occurred) system contention level (points) may not follow the uniform distribution. In other words, the actual system contention level may fall in some subranges more often than others.

Although we can still use uniformly sampled system contention level points (see Figure 3(a)) to run sample queries, the derived cost model may not be as good as the one derived by using a sample of system contention level points that follow the actual distribution in the underlying environment since more frequently occurred system contention level points will receive more weights. If the sampled system contention level points follow a non-uniform distribution, the uniform partition used in *IUPMA* may not be the best although the iterating and adjusting mechanisms sometimes mitigate the problem. For example, if sampled probing query costs in subranges I_{i+1} and I_i are clustered around their common boundary (see Figure 3(b)), the adjusting phase of *IUPMA* typically will merge I_{i+1} and I_i into one. However, if sampled probing query costs are clustered around both boundary ends of each subrange (see Figure 3(c)), the algorithm usually will not merge the two subranges into one. Unless the length of each subrange is small, the resulting classification may be poor since two strong opposite behaviors of sampled probing query costs exist in each subrange (contention state). To overcome the problem, an algorithm for

[7] The coefficient of total determination measures the proportion of variability in the response variable explained by the explanatory variables in a regression model [16]. The higher, the better.
[8] The standard error of estimation is an indication of the accuracy of estimation given by the model [16]. The smaller, the better.

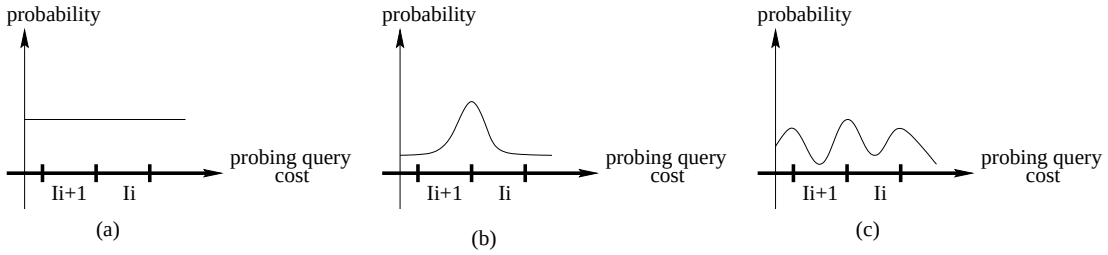


Figure 3: Some Distributions of Probing Query Costs

data clustering is incorporated into the contention states determination procedure as described below.

Let range $[C_{min}, C_{max}]$ of the cost $C_{Q\,p}$ of probing query Q_p be divided into N small intervals: $C_{min} = x_0 < x_1 < x_2 < ... < x_N = C_{max}$, where $|x_i - x_{i-1}| = \varepsilon > 0$ and $1 \le i \le N$. We choose ε to be sufficiently small so that the difference among probing query costs in each interval ($\eta_i = [x_{i-1}, x_i)$ for $1 \le i \le N - 1$ and $\eta_N = [x_{N-1}, x_N]$) is negligible. We then use the mean cost $\mu_i = (x_i + x_{i-1})/2$ to represent the probing query costs in interval η_i. Hence the frame for sampling probing query costs (i.e., system contention level points) is $\Delta = \{\mu_1, \mu_2, ..., \mu_N\}$.

Consider the probing query cost Y in the underlying dynamic environment as a random variable. Let f be the probability density function of Y. The probability of Y falling in interval η_i ($1 \le i \le N$) is:

$$p(\eta_i) = \int_{x_i-1}^{x_i} f(y)dy = F(x_i) - F(x_{i-1}),$$

where F is the cumulative distribution function of Y, In fact, $p(\eta_i)$ can be measured by observing frequencies in experiments. Based on the probability distribution of η_i's in the environment, we draw a sample S of probing query costs (i.e., system contention level points) from Δ to run sample queries in the environment.

Typically, the probing query costs in S are clustered. A natural classification of system contention states is to partition range $[C_{min}, C_{max}]$ according to the clusters in S so that less diverse performance behavior would occur in each contention state.

There are many techniques to analyze clusters in a data set [8]. The popular ones include the k-means method and the agglomerative hierarchical algorithm. Since the k-means method requires knowing the number of clusters in the set in advance, which cannot be predetermined in our case, we apply the agglomerative hierarchical algorithm to analyze the clusters in the above S of probing query costs. The key idea of the algorithm is to place each data object (i.e., sampled query cost) in its own cluster initially and then gradually merge clusters into larger and larger clusters until a desired number of clusters have been found. The criterion used to merge two clusters ω and ω' is to make their distance minimized. We employ a widely used distance measure: $D_{mean}(\omega, \omega') = |\mu(\omega) - \mu(\omega')|$, i.e., the distance between the mean $\mu(\omega)$ of cluster ω and the mean $\mu(\omega')$ of cluster ω'.

Let K be the allowed maximum number of system contention states. The above clustering algorithm can be used to obtain clustering $\Omega^m = \{ \omega_m^m, \omega_m^{m-1}, ..., \omega_1^m \}$ ($m = K, K-1, ..., 1,$

where ω_i^m's are clusters of sampled probing query costs such that $\mu(\omega_i^m) < \mu(\omega_{i-1}^m)$ for $i = m, m-1, ..., 2$).

For each clustering Ω^m (with m clusters), we can get m subranges by partitioning range $[C_{min}, C_{max}]$ of the cost of probing query Q_p as follows. Let $max(\omega_i^m)$ and $min(\omega_{i-1}^m)$ be the maximum and minimum of sampled probing query costs in clusters ω_i^m and ω_{i-1}^m, respectively. Since the sampled probing query costs are from Δ consisting of mean costs of the small intervals for $[C_{min}, C_{max}]$, $min(\omega_{i-1}^m) - max(\omega_i^m)$ is a multiple of interval length ε; that is, $min(\omega_{i-1}^m) - max(\omega_i^m) = n * \varepsilon$. Let

$$a_i^m = \begin{cases} C_{\min} & \text{if } i = m+1 \\ \dfrac{\min(\omega_{i-1}^m) + \max(\omega_i^m)}{2} & \text{if } 2 \le i \le m \text{ and } \min(\omega_{i-1}^m) - \max(\omega_i^m) \text{ is an odd multiple of } \varepsilon \\ \dfrac{\min(\omega_{i-1}^m) + \max(\omega_i^m)}{2} + \dfrac{\varepsilon}{2} & \text{if } 2 \le i \le m \text{ and } \min(\omega_{i-1}^m) - \max(\omega_i^m) \text{ is an even multiple of } \varepsilon \\ C_{\max} & \text{if } i = 1 \end{cases}$$

Then each a_i^m is a boundary point of some interval. Let subranges $I_i^m = [a_{i+1}^m, a_i^m)$ and $I_1^m = [a_2^m, a_1^m]$, where $i = m, m-1, ..., 2$. Each I_i^m $(1 \le i \le m)$ contains one cluster in clustering Ω^m. Clearly, $\{ I_m^m, I_{m-1}^m, ..., I_1^m \}$ defines a set Γ_m of the system contention states for the dynamic environment (see Figure 4); i.e., if probing query cost $C_{Q_p} \in I_i^m$ $(1 \le i \le m)$, we say that the environment is in contention state S_i. Such a classification of system contention states reflects the actual probability distribution of system contention level points (i.e., probing query costs) in the environment. If we use such Γ_m in Line 9 in Algorithm 3.1, we get a new algorithm, termed as the *Contention States Determination via Iterative Clustering with Merging Adjustment (ICMA)*.

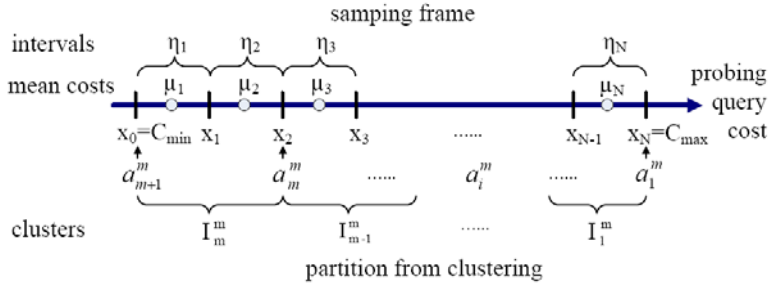


Figure 4: Contention States Determination via Clustering

Note that, for a clustering of sampled probing query costs, it is possible that some cluster(s) may not have a sufficient number of sampled query costs (contention level points) to meet the minimum sample size requirement for regression analysis. In such a case, we draw additional sample data points (therefore, executing more sample queries) to make the cluster meet the minimum requirement rather than simply treat the data points in the cluster as outliers and ignore them. Although this way may change the distribution of the contention level slightly, no useful contention level points are ignored in the derived cost model.

Probing costs estimation

To minimize the overhead for determining a system contention state, a query with a small cost is preferred as a probing query. To further reduce the overhead, an estimated cost (rather than the observed cost) of probing query Q_P can be used to determine the contention state of a dynamic environment. The idea is to first develop a regression equation between probing query cost Y_{Q_P} and some major system contention parameters[9] (such as CPU load ld_1, I/O utilization io, and size of used memory space um for a dynamic environment in Table 1), e.g.,

$$Y_{Q_P} = E_0 + E_1 * ld_1 + E_2 * io + E_3 * um \,, \qquad (7)$$

where E_i (i = 0, 1, 2, 3) are regression coefficients. Afterward, every time when we want to determine the system contention state in which a query is executed, we only need to check which subrange the estimated cost Y_{Q_P} of probing query Q_p lies in by using (7) without actually executing the probing query. Since obtaining the parameter values (ld_1, io, um) in (7) usually requires much less overhead than executing a probing query, using the estimated costs of a probing query to determine system contention states is usually more efficient. However, estimation errors may introduce certain inaccuracy.

4. Development of Cost Models

As mentioned before, we extend the query sampling method for a static environment in [25] so as to develop cost models for a dynamic environment via introducing a qualitative variable. Such extensions include determination of the minimum sample size for query sampling, strategies to add/remove quantitative and qualitative variables into/from a cost model, and statistical measures to evaluate developed qualitative cost models, which are to be discussed in this section.

4.1. Query classification and sampling

Similar to the static query sampling method, we group local queries on a local database system into classes based on their potential access methods to be employed. The previous classification rules and procedures in [25] can be utilized. For example[10],

$$G_{1\,1} = \{\, \pi_{\alpha}\, (\, \sigma_{R.a=C \,\wedge\, F}\, (R)\,) \mid R.a \text{ is a clustered-indexed column in table } R,\ C \text{ is a constant in the domain of } R.a,\ F \text{ is the remaining qualification (in the conjunctive normal form), } \alpha \text{ is a list of target columns from } R \,\} \qquad (8)$$

is a class of unary queries that are most likely performed by using a clustered-index scan access method in a DBMS. Hence a similar performance behavior is shared among the queries in the class and can be described by a common cost model. A classification can be further refined if an improvement of a derived cost model is required. In general, the smaller a query class is, the better the derived cost model, since queries in the class share more homogeneity.

---

[9] A standard statistical procedure can be used to determine the significant parameters for a system environment.
[10] π and σ denote project and select operations in relational algebra, respectively.

A set of sample queries are then drawn from each query class in a similar way as before. More specifically, sample queries are drawn in two steps. The first step is to use the judgment sampling to select a set of representative queries based on one's knowledge about the queries. The second step is to use one or more probability sampling techniques (e.g., simple random sampling, stratified sampling, and cluster sampling) to draw a sample of queries from the foregoing set of representative queries. For example, representative queries for $G_{1\,1}$ in (8) are of the following form:

$$\pi_{\alpha}(\sigma_{R.a = C \wedge R.b\ \omega\ C'}(R)) \tag{9}$$

where R is a table, $R.a$ is a clustered-indexed column in R, C is a constant in the domain of $R.a$, $R.b$ is another column in R, $\omega \in \{=, \neq, <, >\}$, C' is in the domain of $R.b$, and α is a list of columns from R. The first predicate $R.a = C$ in (9) is the key conjunct that determines the major performance behavior (via the chosen access method) of a query in $G_{1\,1}$, while the second one $R.b\ \omega\ C'$ is an auxiliary conjunct used to capture the secondary performance effect of remaining qualification F of a query in $G_{1\,1}$. Since there are a large number of such representative queries due to many possible values for parameters such as C in (9), they cannot all be used as sample queries. On the other hand, there is a minimum sample size requirement to avoid poor estimates of cost model coefficients. A commonly-used rule for sampling in statistics is to sample at least 10 observations for every parameter to be estimated [16]. A sampling procedure based on a mixture of simple random sampling, stratified sampling and cluster sampling is then used to determine one or more sets of values for parameters R, $R.a$, C, $R.b$, ω, C' and α from their relevant domains. It has been shown that our sampling procedures can meet the minimum sample size requirement and in the meantime do not draw too many extra sample queries so that the overhead of the cost model development is minimized [25].

For a dynamic environment, since the more parameters associated with the relevant indicator variables are included in a multistate cost model, the more sample queries are required. The following proposition gives a guideline on the minimum number of sample queries needed for regression analysis.

PROPOSITION 4.1 *For the qualitative regression cost model in (3) with n quantitative explanatory variables and one qualitative variable with m states, at least 10*(m*(n+1)+1) sampling observations are needed.*

PROOF: Notice that there are $(n + 1)$ groups of regression coefficients in the cost model, one for each independent quantitative variable plus the intercept term. Each group has m coefficients, one for each state of the qualitative variable. In addition, the variance of error terms need also to be estimated. Hence, there are totally $m * (n + 1) + 1$ parameters to be estimated, and each requires at least 10 sampling observations.

Sample queries drawn from a query class are performed in a dynamic environment. Their observed data as well as their associated probing query costs are recorded and used to derive a multistate cost model for the query class. A load builder, which is part of the MDBS agent for each local DBS [3], is used to simulate a dynamic application environment (with contention level points following any given distribution) at a local site in an MDBS during the query sampling procedure. The MDBS agent may also have an environment monitor which collects system statistics used for estimating the probing query costs when the estimation approach in Section 3.3 is employed.

4.2. Multistate cost models

A multistate cost model contains a set V_Q of quantitative explanatory variables and a set V_D of indicator variables for the qualitative variable indicating system contention states. Similar to the static query sampling method, we divide a cost model into two parts: *basic model + secondary part*. The basic model represents the essential part of the model, while the secondary part is used to further improve the model. The qualitative variable (i.e., the indicator variables) is included in both parts of the cost model to capture the dynamic environmental factors. Set V_Q is therefore split into two subsets V_B and V_S, where V_B contains basic (quantitative) explanatory variables in the basic model, while V_S contains secondary (quantitative) explanatory variables in the secondary part. Table 4 lists potential explanatory variables in each of the subsets for a unary query class and a join query class. If all variables (including indicator variables) are included, the full cost model is:

$$Y = \underbrace{[(B_0^0 + \sum_{j=1}^{m-1} B_0^j * Z_j) + \sum_{X \in V_B} (B_X^0 + \sum_{j=1}^{m-1} B_X^j * Z_j) * X]}_{\text{basic model}} + \underbrace{[\sum_{X \in V_S} (B_X^0 + \sum_{j=1}^{m-1} B_X^j * Z_j) * X]}_{\text{secondary part}}$$

However, usually, not all variables are necessary for a given cost model. Note that this model is obtained by regrouping the explanatory variables in Formula (3) into the basic and secondary sets.

Class	Basic Explanatory Variables	Secondary Explanatory Variables
Unary Query Class	N_U – size (cardinality) of operand table TN_U – size of intermediate table RN_U – size of result table	L_U – tuple length of operand table RL_U – tuple length of result table NZ_U – operand table length $N_U * L_U$ RZ_U – result table length $RN_U * RL_U$
Join Query Class	N_{J1} – size of 1st operand table N_{J2} – size of 2nd operand table TN_{J1} – size of 1st intermediate table TN_{J2} – size of 2nd intermediate table RN_J – size of result table TN_{J12} – size of Cartesian product of intermediate tables	L_{J1} – tuple length of 1st operand table L_{J2} – tuple length of 2nd operand table RL_J – tuple length of result table NZ_{J1} – 1st operand table length $N_{J1} * L_{J1}$ NZ_{J2} – 2nd operand table length $N_{J2} * L_{J2}$ RZ_J – result table length $RN_J * RL_J$

Table 4: Potential Explanatory Variables for Cost Models

To determine the variables to be included in the cost model for a query class, a mixed backward and forward procedure described below is adopted. We start with the full basic model which includes all variables in V_B and use a backward procedure to eliminate insignificant basic explanatory variables one by one. Note that, in our algorithm, if an explanatory variable X is removed from the model, its coefficients ($B_x^0 + \sum_{j=1}^{m-1} B_x^j * Z_j$) for all contention states (determined by indicator variables Z_j's) are removed. We then use a forward selection procedure to add more significant secondary explanatory variables from V_S into the cost model. This procedure tries to further improve the cost model. Similar to the backward procedure, if a

secondary variable X is added into the model, its coefficients ($B_x^0 + \sum_{j=1}^{m-1} B_x^j * Z_j$) for all contention states are included. Since it is expected that most basic variables are important to a cost model and only a few secondary explanatory variables are important, both the backward elimination and the forward selection procedures most likely terminate soon after they start. The above process is therefore quite efficient.

Assume that we have n_j sampling observations in contention state S_j *(1 ≤ j ≤ m)*, with $\sum_{j=1}^{m} n_j$ observations in total. Consider the simple correction coefficient [16] between variables X and Y in contention state S_j:

$$r_{X.Y}^{j} = [\sum_{i=1}^{n_j} X_{ij} Y_{ij} - (\sum_{i=1}^{n_j} X_{ij} \sum_{i=1}^{n_j} Y_{ij}) / n_j] \Big/ \sqrt{(\sum_{i=1}^{n_j} X_{ij}^2 - (\sum_{i=1}^{n_j} X_{ij})^2 / n_j)(\sum_{i=1}^{n_j} Y_{ij}^2 - (\sum_{i=1}^{n_j} Y_{ij})^2 / n_j)}$$

where X_{ij}, Y_{ij} are the values from the ith sampling observation ($1 \le i \le n_j$) in state S_j. For any explanatory variable X, if its maximum simple correlation coefficient $\max_{1 \le j \le m} \{r_{X.Y}^j\}$ with response variable Y is too small, it has little linear relationship with Y in any state. Such explanatory variables should be removed from consideration.

In the backward elimination procedure, the next variable X to be removed from the current model is the one which satisfies two conditions (a) its average simple correlation coefficient $\bar{r}_{X.Y} = (\sum_{j=1}^{m} r_{X.Y}^j) / m$ with response variable Y for all contention states is the smallest among all explanatory variables in the current model; (b) it makes $s' \le s$ or $|(s - s')/s| < \varepsilon$, where s' is the standard error of estimation [16] for the reduced model (i.e., with X removed) given by:

$$s' = \sqrt{[\sum_{j=1}^{m} \sum_{i=1}^{n_j} (Y_{ij} - \hat{Y}_{ij}')^2] / [\sum_{j=1}^{m} n_j - m * (k+1)]} \qquad (10)$$

where Y_{ij} , $\hat{Y}_{ij}'$, k denote the observed query cost, estimated query cost given by the reduced model, and number of explanatory variables in the model, respectively; s is the standard error of estimation for the original model given by a formula similar to (10); ε is a given small positive constant. Since the average simple correlation coefficient $\bar{r}_{X.Y}$ indicates the degree of linear relationship between X and Y on average in all states, foregoing condition (a) selects an explanatory variable X that contributes the least (on average in all states) in explaining the response variable Y. Since the standard error of estimation is an indication of estimation accuracy, foregoing condition (b) ensures that removing variable X from the model improves the estimation accuracy or affects the model very little. Removing a variable that has a little effect on the model can reduce the complexity and maintenance overhead of the model.

In the forward selection procedure, the next variable X from V_S to be added into the current model is the one satisfies (a) its average simple correlation coefficient $\bar{r}_{X.Y_S} = (\sum_{j=1}^{m} r_{X.Y_S}^j) / m$ with the residuals Ys of the current model for all states is the largest among all explanatory variables in the model; i.e., it can explain the most (on average for all states) about the variations that the current model cannot explain; and (b) it significantly improves the estimation accuracy, i.e., $s' < s$ and $|(s - s')/s| > \varepsilon$, where s', s denote the standard errors of estimation for the augmented model (i.e., with X included) and the original model, respectively; and ε is a given small positive constant.

Note that the exact number of explanatory variables in a cost model is determined after the above mixed backward and forward procedure is done. However, we need such information to determine the query sample size from Proposition 4.1 at the beginning of the cost model development. Since it is expected that most basic explanatory variables in V_B are selected and only a few secondary explanatory variables in V_S are used for a cost model, we expect the number of explanatory variables in a cost model usually not exceed $|V_B| + \lceil |V_S|/2 \rceil$. Based on experiments, the maximum number M of contention states for a dynamic environment in practice can also be estimated. Hence, a reasonable query sample size is:

$$10 * (M * (|V_B| + \lceil |V_S|/2 \rceil + 1) + 1) \qquad (11)$$

from Proposition 4.1.

4.3. Statistical measures for developing useful models

Multicollinearity occurs when explanatory variables are highly correlated among themselves. In such a case, the estimated regression coefficients tend to have large sampling variability. It is better to avoid multicollinearity.

The presence of multicollinearity is detected by means of the variance inflation factor (VIF) [13]. When an explanatory variable has a strong linear relationship with the other explanatory variables, its VIF is large. In a dynamic environment with multiple contention states, let VIF_j ($1 \le j \le m$) be the variance inflation factor of explanatory variable X in state S_j. If $\min_{1 \le j \le m} \{VIF_j\}$ is large, X is not included in a cost model to avoid multicollinearity.

In addition, F-test, the standard error of estimation s, the coefficient of multiple determination R^2, as well as the percentage of good cost estimates for some test queries are used to validate the significance of a developed regression cost model.

4.4. Query optimization using multistate cost models

In the previous sections, we have developed multistate cost models to estimate query costs in a dynamic multidatabase environment. Although estimating query costs is essential to query optimization, it is not the ultimate goal. The ultimate goal of query optimization is to choose a good execution plan for a query on the basis of the cost estimates by the cost models.

In general, there are two approaches to processing a query in a database system. The first one is called the interpretation approach. In this approach, simple query optimization is performed on the fly while a query is being executed. This approach is suitable for ad hoc/interactive queries, which are usually executed only once. The second one is called the compilation approach. In this approach, comprehensive query optimization is performed for a given query at compile time, resulting in an execution plan. The execution plan can be then executed repeatedly at run time. This approach is more suitable for stored and embedded queries, which are usually executed repeatedly. In an MDBS environment, both stored/embedded queries and ad hoc/interactive queries are expected. Another mixed compilation and interpretation approach was proposed for processing queries in an MDBS environment [28].

The multistate cost models can be easily applied to perform query optimization in the interpretation approach for a dynamic environment. The current system contention state at a local site can be detected by either running a small probing query or analyzing environmental statistics that can be used to estimate probing costs as in Section 3.3. Note that the overhead for detecting

the current system contention states is usually negligible, compared with the significant cost saving (e.g., hours or days) from a good execution strategy in a distributed multidatabase environment. The local costs of (component) queries in the current contention states at local sites can be estimated by the relevant multistate cost models. Based on the estimated local costs[11], the global query optimizer can determine a good execution strategy to run the given global query. Since the multistate cost models estimate query costs based on the current system states at local sites, they usually yield more accurate cost estimates comparing to the cost estimates given by the traditional static cost models, which in turn leads to more efficient query processing in the dynamic multidatabase environment.

It is not easy to perform query optimization in the compilation approach for a dynamic environment. This is because it difficult to predict the run-time system environment when a query is optimized at compile time. A traditional approach to improving performance for the execution plan chosen at the compile time in a dynamic environment is to dynamically modify the plan based on dynamic information observed at run time, which is called so-called dynamic/adaptive query optimization [2, 4, 14, 19]. Unfortunately, this approach may dramatically increase the query response time, which should be avoided for the embedded queries that need to be executed repeatedly.

To perform query optimization at compile time as much as possible and reduce query processing time at run time, we can employ the multistate cost models in the following way. We generate multiple versions (rather than one) of an execution plan for a query during query optimization at compile time. The idea is to use different versions to optimize the query for various system contention states. The query will be executed by invoking an appropriate version of the execution plan based on the detected system contention states at run time. Although the query optimizer takes more time to generate multiple versions of an execution plan for a query at compile time, which in fact does not affect the query response time, the cost saving at run time is usually very significant so that the overhead is well paid off. The other possible ways to apply the multistate cost models for query optimization include the optimistic approach, the environment predicting approach, and the lazy approach. Since the focus of this paper is to develop cost models for a dynamic multidatabase environment rather than to study query optimization strategies, we discuss the details of relevant query optimization techniques in separate papers [22, 29].

4.5. Estimating query costs for more complex cases using multistate cost models

Using the multistate cost models, we can directly estimate the cost of a query run in any contention state in a dynamic environment. However, there are two cases in which a query may experience multiple contention states during its execution: (1) the query is too large to be completed in one contention state although the environment changes gradually; and (2) the environment changes its contention states rapidly although the query may not be large.

To estimate the cost of a query experiencing multiple contention states in the first case, we employ a fractional analysis technique. The key idea is to analyze a query cost by fractionalizing it according to the contention states to be experienced. Notice that the system load in a particular application environment often demonstrates a certain pattern. For example, in a company, its system load is minimum off working hours, starts to grow in the morning when the working

[11] The query optimizer may also take some other factors such as network status into consideration during query optimization in an MDBS. These factors are not the topics to be discussed in this paper.

hours begin, and declines when the working hours are close to the end of the day. This load pattern repeats every working day in the company. Based on the pattern, the sequence of contention states in the system environment for each day can be determined. Let $\Delta = \{ S_1, S_2, ..., S_M \}$ be the set of all possible contention states in a given application environment; $\{ S^{(i)}, i = 1, 2, ... \}$ be the sequence of contention states occurring in the environment where $S^{(i)} \in \Delta$; and $t^{(i-1)}$ and $t^{(i)}$ be the starting and ending time instants for state $S^{(i)}$ ($i = 1, 2, ...$). Consider query Q starting its execution at time instant t_Q^s ($t^{(k-1)} \le t_Q^s \le t^k$) in state $S^{(k)}$. Let $C(Q)$ be the cost of query Q, which may experience multiple states. Let $C(Q, S^{(i)})$ $(i = k, k+1, ...)$ be the cost of query Q if the query is executed entirely in state $S^{(i)}$, which can be estimated by a multistate cost model introduced in the previous sections. Let $T^{(k)} = min\{ C(Q, S^{(k)}), (t^{(k)} - t_Q^s) \}$ and $T^{(i)} = (t^{(i)} - t^{(i-1)})$ for $i \ge k+1$. If $C(Q, S^{(k)}) \le (t^{(k)} - t_Q^s)$, query Q is to experience only one contention state $S^{(k)}$. Hence $C(Q, S^{(k)})$ is the cost for query Q, i.e., $C(Q) = C(Q, S^{(k)})$. If $C(Q, S^{(k)}) > (t^{(k)} - t_Q^s)$, query Q is to experience more than one contention state. Then $T^{(k)}/C(Q, S^{(k)})$ (clearly, < 1 in this case) is the fraction of work done for Q in state $S^{(k)}$. The remaining fraction $[1 - T^{(k)}/C(Q, S^{(k)})]$ of work for Q is to be done in the subsequent contention states. If $[1 - T^{(k)}/C(Q, S^{(k)})] * C(Q, S^{(k+1)}) \le (t^{(k+1)} - t^{(k)})$, all remaining work of Q can be done in state $S^{(k+1)}$. Thus the cost of Q is: $C(Q) = T^{(k)} + [1 - T^{(k)}/C(Q, S^{(k)})] * C(Q, S^{(k+1)})$. Following this fractional analysis procedure, similar cost estimates can be obtained if the last state that query Q experiences is $S^{(k+2)}, S^{(k+3)}$ ....

To estimate the cost of a query experiencing multiple contention states in the second case, we employ a probabilistic technique. The idea is to make use of the theory of Markov chains to derive a cost formula to estimate the query costs in such an environment. More specifically, the cost estimation formula for query Q in such a dynamic environment can be given as follows:

$$C(Q) = 1 / [\sum_{i=1}^{M} \frac{\pi_i}{C(Q, S_i)}] , \qquad (12)$$

where π_i is the limit probability of state S_i, which can be determined by solving a system of linear equations based on the transition probabilities for one state changing to another state in the Markov chain [15].

Clearly, the above two techniques make use of the cost estimates $C(Q, Si)$'s obtained from the multistate cost models to estimate the query cost for more complex cases. The detailed discussion of these techniques can be found in [21]. The multistate cost models provide a base to analyze query costs in more complex dynamic environments. Hence the multistate query sampling method together with the fractional analysis and the probabilistic techniques comprise a suite of techniques to estimate the query costs for various dynamic multidatabase environments.

5. Experimental Results

To verify the feasibility of our multistate query sampling method for developing cost models in a dynamic environment, experiments were conducted in a multidatabase environment using a research prototype named CORDS-MDBS [3]. Two commercial DBMSs, i.e., Oracle 8.0 and DB2 5.0, were used as local database systems running under Solaris 5.1 on two SUN UltraSparc 2 workstations. Figure 5 shows the experimental environment. Local queries are submitted to a local DBS via an MDBS agent. The MDBS agent provides a uniform relational ODBC interface

for the global server. It also contains a load builder which generates dynamic loads to simulate dynamic application environments.

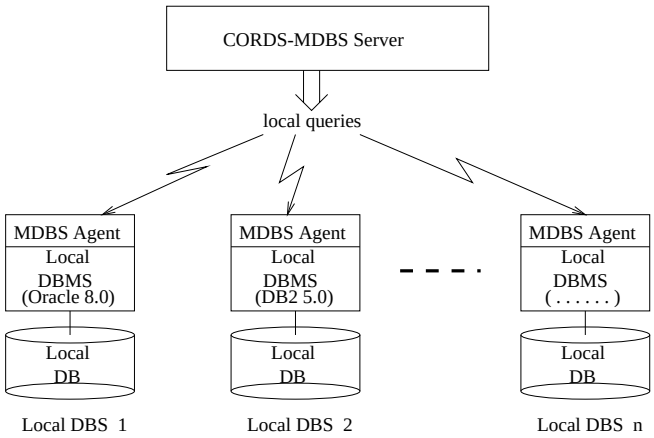


Figure 5: Experimental Environment

A synthetic experimental database was created for each local DBS in the experiments. Each experimental database contains 12 tables as shown in Table 5. Data in the tables are generated

Table	Indexed	Clustered-indexed	Cardinality
R1(a1, a2, a3)	a2, a3		250,000
R2(a1, a2, a3, a4, a5)	a2, a4	a3	200,000
R3(a1, a2, a3, a4, a5)	a3, a4		17,000
R4(a1, a2, a3, a4, a5, a6, a7)	a2, a6		3,000
R5(a1, a2, a3, a4, a5, a6, a7)	a2, a4	a1	30,000
R6(a1, a2, a3, a4, a5, a6, a7)	a4, a7	a1	10,000
R7(a1, a2, a3, a4, a5, a6, a7, a8, a9)	a1, a4, a7	a2	50,000
R8(a1, a2, a3, a4, a5, a6, a7, a8, a9)	a2, a6, a9	a3	150,000
R9(a1, a2, a3, a4, a5, a6, a7, a8, a9)	a1, a3, a5, a8		40,000
R10(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11)	a2, a4, a10	a1	100,000
R11(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11)	a2, a7	a1	7,000
R12(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13)	a2, a5, a11, a13		80,000

Table 5: Tables in Experimental Databases

as follows. The values for the nth column of a table are randomly generated from the domain [1, $100*n^2$]. Hence a predicate on different columns in a table has different selectivities. For an equality predicate, for example, the lower the column number is, the higher the relevant selectivity (closer to 1). The characteristics of such an experimental database include: (1) the number of columns in a table ranges from 3 to 13, with more tables having medium numbers of columns; (2) the cardinality of a table ranges from 3000 to 250,000; (3) different selectivities are provided in a table for different columns; and (4) various types of columns including indexed, clustered-indexed and sequential ones are considered. This experimental database allows us to test the feasibility of our technique to build cost models for capturing a variety of factors in a database environment. Note that, typically, not all factors have a large changing range in a real

database. The flat factors will not be considered in a cost model for a specific environment, which would lead to a simpler and more accurate cost model.

In the experiments, queries on each local DBS were first classified according to Section 4.1. A sample of queries with the size meeting condition (11) were then drawn from each query class and performed in the dynamic environments at the local sites. Their observed costs together with the associated probing query costs are used to derive a cost model with a qualitative variable for each query class using the techniques introduced in the previous sections. Some randomly-generated test queries in the relevant query classes were also performed in the dynamic environment, and their observed costs were compared with the estimated costs given by the derived cost models. The unary test queries are of the following form:

$$\pi_{\alpha}(\sigma_{F}(R))$$

where α is a list of columns from table *R*, qualification

$$F ::= P_R^1 \,\big|\, P_R^1 \wedge P_R^2 \,\big|\, P_R^1 \wedge (\, P_R^2\, [\vee P_R^3\,])\,,$$

P_R^i (*i = 1, 2, 3*) are valid predicates on table *R* for the relevant query classes. The join test queries are of the following form:

$$\pi_{\alpha}(R_1 \overset{\triangleright\triangleleft}{}_{F} R_2)$$

where α is a list of columns from tables R_1 and R_2, qualification

$$F ::= [(P_{R_1}^1 \mid P_{R_1}^1 \vee P_{R_1}^2)] \wedge P_{(R_1,R_2)} \wedge [\,(P_{R_2}^1 \mid P_{R_2}^1 \vee P_{R_2}^2)]\,,$$

$P_{R_j}^i$ (*j = 1, 2; i = 1, 2, 3*) are valid predicates on table R_j for the relevant query classes, and $P_{(R_1,R_2)}$ is a valid join predicate on tables R_1 and R_2 for the relevant query classes. These test queries cover typical forms of unary and join queries.

Note that, unlike scientific computation in engineering, the accuracy of cost estimation in query optimization is not required to be very high. The estimated costs with relative errors within 30% are considered to be very good, and the estimated costs that are within the range of one-time larger or smaller than the corresponding observed costs (e.g., 2 minutes vs. 4 minutes) are considered to be good. Only those estimated costs which are not of the same order of magnitude with the observed costs (e.g., 2 minutes vs. 3 hours) are not acceptable.

Table 6 shows the cost models derived by applying the multistate query sampling method suggested in this paper for three representative query classes for each local DBS, namely, a unary query class G_1 without usable indexes, a unary query class G_2 with usable non-clustered indexes for ranges, and a join query class G_3 without usable indexes[12]. Let us use the derived cost model for query class G_3/DB_2 in Table 6 as an example to explain why such a multistate cost model can usually give better cost estimates in a dynamic environment, compared to a static cost model. From the table, we can see that this cost model includes two indicator variables: Z_1 and Z_2, implying that three contention states are needed to capture the query performance behavior for query class G_3 in this DB_2 local database environment. Let the three contention states be: S_3 (i.e., $Z_2 = Z_1 = 0$), S_2 (i.e., $Z_2 = 1$ & $Z_1 = 0$) and S_1 (i.e., $Z_2 = 0$ & $Z_1 = 1$), where S_3 is the lowest contention state and S_1 is the highest contention state. The coefficients of indicator variables Z_2

[12] The three query classes correspond to $G_{1\,4}$, $G_{1\,5}$, and $G_{2\,4}$ in [25].

Query Class	Cost Estimation Model with Qualitative Variable (i.e., Multistate Cost Models)
G_1/DB2	$(-0.1522e+1-0.5784e+0*Z_2+0.1760e+1*Z_1)+(-0.1333e-4-0.2149e-3*Z_2$ $+0.4738e-4*Z_1)*TN_U+(0.5467e-5+0.8293e-4*Z_2+0.5145e-4*Z_1)*N_U$ $+(0.1378e-2+0.1469e-2*Z_2+0.3310e-2*Z_1)*RN_U+(0.2912e-7+0.1636e-4*Z_2$ $+0.8896e-5*Z_1)*NZ_U$
G_2/DB2	$(0.6758e+1-0.4563e+1*Z_5-0.1311e+2*Z_4-0.3462e+1*Z_3-0.1198e+2*Z_2$ $+0.3981e+1*Z_1)+(0.6701e-4-0.3545e-4*Z_5+0.6882e-4*Z_4+0.1225e-3*Z_3$ $-0.1153e-3*Z_2+0.1855e-3*Z_1)*N_U+(0.6153e-3+0.7202e-3*Z_5+0.1472e-2*Z_4$ $+0.2740e-2*Z_3+0.3729e-2*Z_2+0.4015e-2*Z_1)*RN_U+(0.5499e-1+0.8548e+0*Z_5$ $+0.8651e+0*Z_4+0.1126e+1*Z_3+0.2258e+1*Z_2+0.2269e+1*Z_1)*RL_U+(-0.1155e+1$ $+0.7266e+0*Z_5+0.1288e+1*Z_4-0.3080e+0*Z_3+0.1167e+1*Z_2-0.1680e+1*Z_1)*L_U$
G_3/DB2	$(0.1232e+2+0.6065e+2*Z_2-0.3505e+2*Z_1)+(0.5634e-7+0.6310e-8*Z_2$ $+0.3707e-6*Z_1)*TN_{J12}+(0.8489e-3+0.1586e-2*Z_2+0.3656e-2*Z_1)*RN_J$
G_1/Oracle	$(0.1648e-1-0.5209e+0*Z_2+0.2931e+1*Z_1)+(-0.3030e-3+0.1586e-3*Z_2$ $+0.3281e-3*Z_1)*TN_U+(-0.1549e-4+0.4146e-4*Z_2+0.6483e-4*Z_1)*N_U$ $+(0.2691e-2+0.1986e-2*Z_2+0.3791e-2*Z_1)*RN_U+(-0.5699e-4-0.1221e-4*Z_2$ $+0.8089e-4*Z_1)*RZ_U+(0.8557e-5+0.3322e-5*Z_2+0.1894e-6*Z_1)*NZ_U$
G_2/Oracle	$(0.5262e+1-0.7762e+1*Z_5-0.3278e+1*Z_4-0.7294e+1*Z_3-0.7848e+1*Z_2$ $-0.2817e+1*Z_1)+(-0.1034e-3+0.1738e-3*Z_5+0.1651e-3*Z_4+0.2833e-3*Z_3$ $+0.2526e-3*Z_2+0.2542e-3*Z_1)*N_U+(0.5224e-3+0.2012e-2*Z_5+0.2955e-2*Z_4$ $+0.4490e-2*Z_3+0.6067e-2*Z_2+0.6541e-2*Z_1)*RN_U$
G_3/Oracle	$(-0.1457e+2-0.4381e+2*Z_3+0.7830e+2*Z_2-0.7726e+2*Z_1)+(-0.9777e-7$ $+0.1322e-6*Z_3+0.2320e-6*Z_2+0.1373e-6*Z_1)*TN_{J12}+(0.1257e-2+0.1737e-2*Z_3$ $+0.3801e-2*Z_2+0.5704e-2*Z_1)*RN_J+(0.7793e-3-0.4290e-3*Z_3-0.8994e-3*Z_2$ $-0.1771e-6*Z_1)*TN_{J1}+(-0.1887e+1+0.4723e+1*Z_3+0.8677e+1*Z_2+0.9416e+1*Z_1)$ $*RL_J+(0.6537e+1+0.2638e-1*Z_3-0.2248e+2*Z_2+0.3179e+1*Z_1)*L_{J1}$

Table 6: Multistate Cost Models for DB2 and Oracle

and Z_1 in the cost model represent the coefficient adjustments for the corresponding explanatory variables in contention states S_2 and S_1, respectively. For example, the (integrated) coefficient of explanatory variable RN_J (i.e., the result table size, which is usually the most significant explanatory variable) for all contention states in the cost model is: $(0.8489e{-}3 + 0.1586e{-}2 * Z_2 + 0.3656e{-}2 * Z_1)$. When contention state is S_3, S_2, or S_1, the coefficient is adjusted to: 0.8489e−3 ($Z_2 = Z_1 = 0$), 0.2435e−2 ($Z_2 = 1$ & $Z_1 = 0$), or 0.4505e−2 ($Z_2 = 0$ & $Z_1 = 1$), respectively. Clearly, for a higher contention state such as S_1, its adjusted coefficient is larger; which implies that a larger overhead is needed to process a result tuple in a higher contention state. Note that the number of contention states needed and the adjustments of coefficients of explanatory variables for each state are determined automatically for a given environment by the statistical procedure presented in the previous sections. In this way, the technique can build cost model capturing the dynamic behavior of a multidatabase environment.

Table 7 shows some statistical measures for the derived cost models[13]. For the comparison purpose, two static experimental cases were considered. In the first case, cost models were derived by applying the static query sampling method to sampling data obtained from a static environment (Static Approach I). In the second static case, cost models were derived by applying the static query sampling method to sampling data obtained from a dynamic environment (Static

[13] The number in parentheses beside 'multistates' in Table 7 indicates the number of contention states used for the relevant cost model.

Approach II). This, in fact, is to restrict the multistate query sampling method to consider only one contention state.

Query class	Cost model type	R^2	s	Average cost	Very good estimates	Good estimates
G_1 for DB2	Multistates (3)	0.972	0.157e+2	0.528e+2	55%	78%
	One-state	0.798	0.363e+2	0.511e+2	30%	58%
	Static	0.972	0.672e+0	0.290e+1	3%	5%
G_2 for DB2	Multistates (6)	0.994	0.997e+1	0.620e+2	60%	76%
	One-state	0.779	0.620e+2	0.690e+2	24%	48%
	Static	0.986	0.733e+0	0.359e+1	7%	14%
G_3 for DB2	Multistates (3)	0.996	0.230e+3	0.735e+3	37%	62%
	One-state	0.910	0.254e+3	0.431e+3	27%	45%
	Static	0.992	0.116e+2	0.381e+2	9%	13%
G_1 for Oracle	Multistates (3)	0.982	0.160e+2	0.680e+2	69%	81%
	One-state	0.876	0.576e+2	0.865e+2	35%	60%
	Static	0.999	0.917e-1	0.402e+1	3%	6%
G_2 for Oracle	Multistates (6)	0.993	0.143e+2	0.873e+2	63%	74%
	One-state	0.901	0.672e+2	0.108e+3	35%	62%
	Static	0.999	0.301e+0	0.493e+1	4%	8%
G_3 for Oracle	Multistates (4)	0.999	0.148e+3	0.998e+3	51%	67%
	One-state	0.951	0.507e+3	0.882e+3	22%	44%
	Static	0.999	0.503e+1	0.492e+2	0%	1%

Table 7: Statistics for Cost Models

From the experimental results, we have the following observations:

- The multistate query sampling method presented in this paper can derive good cost models in a dynamic environment. The coefficients of total determination in Table 7 indicate that all derived models can capture 98.9% variations in query cost on average. The standard errors of estimation are acceptable, compared with the magnitude of the average cost of relevant sample queries (only 22% of average costs on average). The statistical F-tests at significance level $\alpha = 0.01$ were also conducted, which showed that all cost models are useful for estimating query costs in a dynamic environment. In fact, the more dynamic an environment is, the more a multistate cost model outperforms a one-state cost model.
- The (static) cost models derived by the static query sampling method for a static environment (i.e., Static Approach I) are not suitable for estimating query costs in a dynamic environment. Although such cost models may have good coefficients of total determination (99.1% on average in Table 7) for the sampling data in a static environment, they can hardly give good cost estimates in a dynamic environment (gave only 7.8% good cost estimates on average in Table 7 for the test queries in our experiments).
- The (multistate) cost models derived by using the multistate query sampling method for a dynamic environment significantly improve the (one-state) cost models derived by applying the static query sampling method for the dynamic environment (i.e., Static Approach II). In fact, compared with the one-state cost models, the multistate cost models

increase the number of very good cost estimates (i.e., with relative errors ≤ 0.3) and the number of good cost estimates (i.e., within one time range) by 27.0% and 20.2% (on average), respectively, for the test queries. Figures 6 ~ 11 show comparisons among the observed costs, estimated costs by the multistate cost models, and estimated costs by the one-state cost models for the test queries in a dynamic environment.

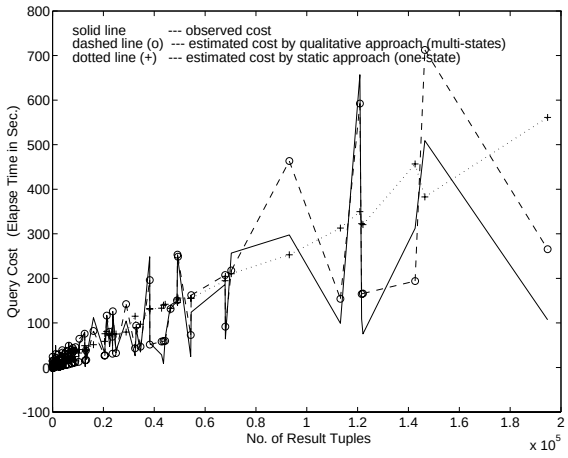


Figure 6: Costs for Test Queries in G_1 on DB2 5.0

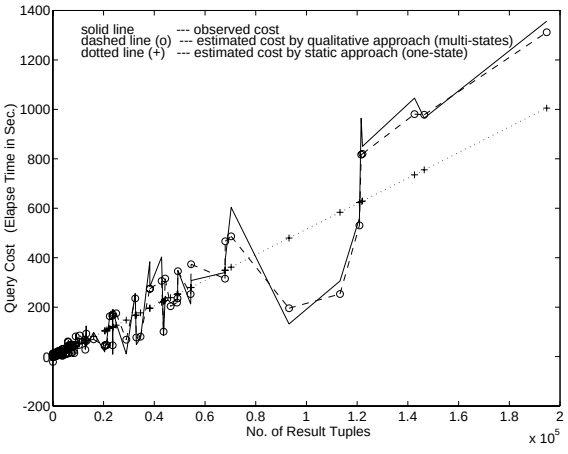


Figure 7: Costs for Test Queries in G_1 on Oracle 8.0

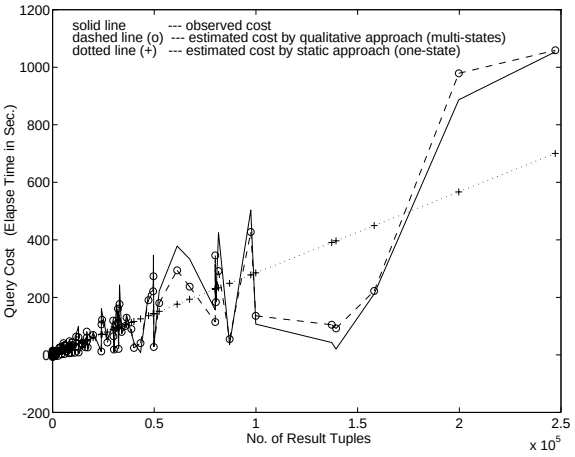


Figure 8: Costs for Test Queries in G_2 on DB2 5.0

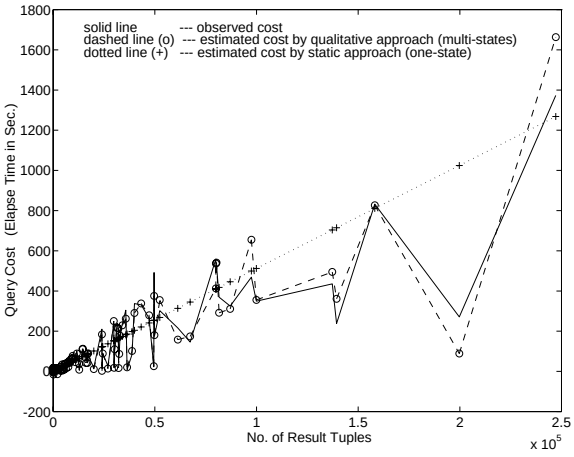


Figure 9: Costs for Test Queries in G_2 on Oracle 8.0

- The more contention states are considered, the better the derived cost model usually is. For example, the coefficients of total determination for the cost models for query class G_2/Oracle with 1 to 6 contention states are 0.901404, 0.974606, 0.978702, 0.990002, 0.990005, 0.993221, respectively. However, the improvement is very small after the number of contention states reaches certain point. Table 7 shows that usually considering

3 to 6 contention states for a dynamic environment is sufficient to obtain a good cost model.

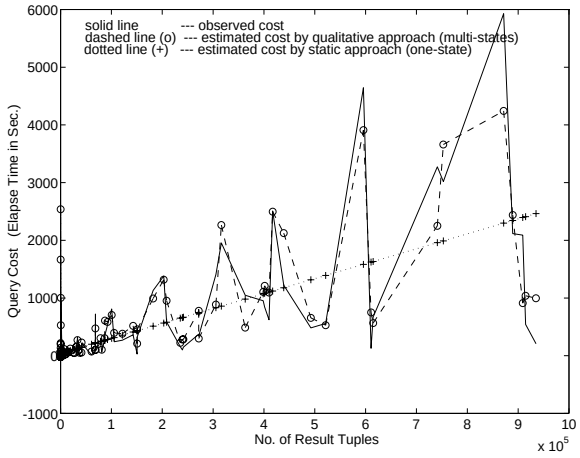


Figure 10: Costs for Test Queries in G_3 on DB2 5.0

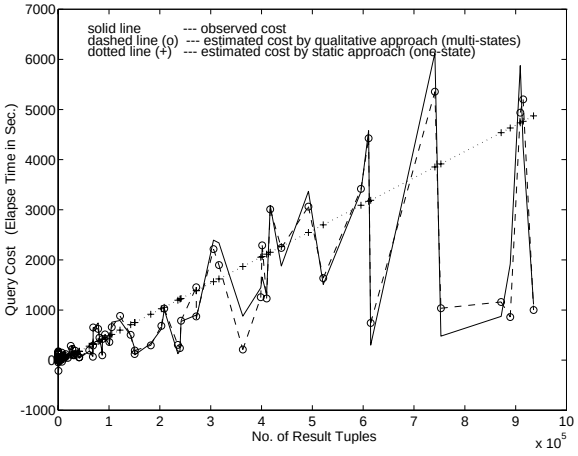


Figure 11: Costs for Test Queries in G_3 on Oracle 8.0

- Like static techniques [5, 25], it is also true for the multistate query sampling method that small-cost queries usually have worse cost estimates than large-cost queries. The main reason for this is that even a small momentary change in the system environment may have a significant impact on the cost of a small-cost query. It is not easy to capture all such small environmental changes in a cost model. Fortunately, estimating the costs of small-cost queries is not as important as estimating the costs of large-cost queries because it is more important to identify large-cost queries so that inefficient execution plans can be avoided.
- Contention states determination algorithm *IUPMA* works well for both uniformly-distributed and non-uniformly-distributed (clustered) probing query costs, while algorithm *ICMA* can determine an even better set of system contention states for the clustered cases. Note that the sampled probing query costs were drawn by following the distribution of the contention level in a dynamic environment. In fact, the experimental results shown in Tables 6 ~ 7 and Figures 6 ~ 11 were obtained for the uniform case. Extensive experiments were also conducted for clustered cases. The experimental results showed that, for a given query class, the cost model derived in the clustered cases is usually better than the one derived for the uniform case even if *IUPMA* is used. This is because the cost models for the clustered cases only need to capture performance behavior of queries in more focused and narrower subrange(s) of the contention level. Table 8 shows some typical experimental results for a query class in a dynamic environment with clustered contention level points (see Figure 12 for the corresponding frequency distribution of the contention level).

To check the effect of multistate cost models on query optimization, we also conducted some experiments. In the experiments, for simplicity, we considered only two dynamic sites A and B in the multidatabase environment. Site A ran Oracle 8.0, and Site B ran DB2 5.0. Each site can have a contention level point from 1 (least) to 96 (highest). Global test queries were randomly

generated and executed to retrieve information from local databases at the two sites. The following two execution plans were considered:

- Plan P_1: execute a local query at Site A, transfer the result to Site B, and perform the final integration (join) at Site B.
- Plan P_2: execute a local query at Site B, transfer the result to Site A, and perform the final integration (join) at Site A.

Query class	States determination	# of states	R^2	s	Average cost	very good estimates	Good estimates
G_1 for DB2	*IUPMA*	3	0.978	0.128e+2	0.488e+2	58%	82%
	ICMA	3	0.991	0.740e+1	0.465e+2	82%	95%

Table 8: Statistics for Cost Models in a Clustered Case

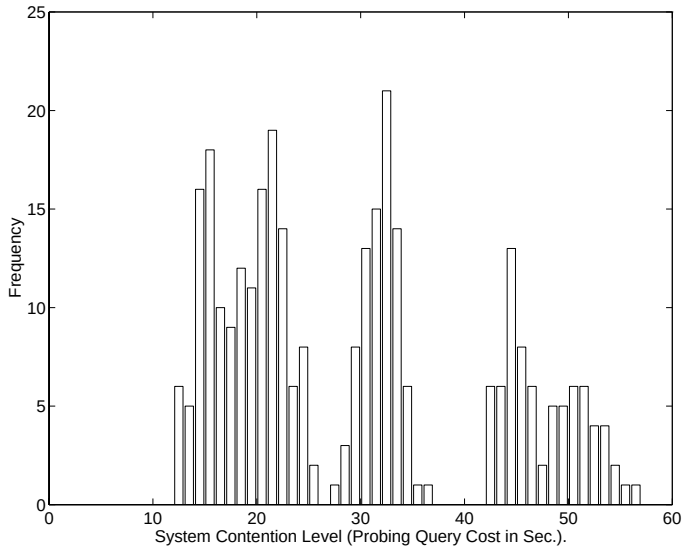


Figure 12: Histogram of Contention Level Points in a Clustered Case

A global query optimizer should choose a better plan between P_1 and P_2. Assume that data transferring cost is negligible in our fast local area network. Let $C(Pi)$, $M(Pi)$, $S_1(Pi)$, and $S_2(Pi)$ be the costs of plan Pi $(i = 1, 2)$ based on observations, multistate cost models, one-state cost models (Static Approach II), and static cost models (Static Approach I), respectively. Notice that, as long as the observed cost difference $C(P_1) - C(P_2)$ and the estimated cost difference (e.g., $M(P_1) - M(P_2)$) have the same sign (i.e., "+" or "–"), the corresponding cost models can help the query optimizer determine a correct plan. To include various dynamic environments in the experiments, we considered different ratios of contention level points between Site A and Site B as follows: 96:1, 84:1, 72:1, ..., 12:1, 1:1, 1:12, ..., 1:84, 1:96. Table 9 shows some typical experimental data. We use C, M, S1, and S2 to denote the observing, multistates, one-state, and static approaches in the table. Qi ($1 \leq i \leq 8$) denote test queries. From the experimental data, we can see that:

- The static cost models can only determine a correct plan (i.e., the better one) for 49.3% cases.

- The multistate cost models can determine a correct plan for 87.5% cases, which represents a significant improvement of 38.2% over the static models.
- Only for those cases in which the costs of two alternate plans are quite close to each other, the multistate cost models may fail to determine a better plan. Fortunately, choosing either plan does not make much difference in such cases.

These experimental results demonstrate that the multistate cost models are very promising in determining an efficient plan during query optimization.

		Ratio of contention level points at Sites A and B																
		96:1	84:1	72:1	60:1	48:1	36:1	24:1	12:1	1:1	1:12	1:24	1:36	1:48	1:60	1:72	1:84	1:96
Q1	C	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q2	C	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q3	C	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q4	C	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	+	+
	M	-	-	-	-	-	+	+	+	+	+	+	+	+	+	+	+	+
	S1	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	S2	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Q5	C	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q6	C	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q7	C	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q8	C	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
	M	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-
	S1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	S2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
"+" — cost difference is positive, i.e., P_2 is better than P_1 "-" — cost difference is negative, i.e., P_1 is better than P_2																		

Table 9: Determining Efficient Query Execution Plan Based on Multistate and Static Cost Models

6. Conclusions

Local query cost estimation is essential to global query optimization in an MDBS. The techniques proposed so far in the literature to develop local cost models in an MDBS are only suitable for a static environment. A query cost can be dramatically affected by dynamic factors in a multidatabase environment.

The main contribution of this paper is that we have proposed a novel qualitative approach (also called the multistate query sampling method) to building a cost model for a dynamic multidatabase environment, which includes: (1) a technique to incorporate a qualitative variable indicating the system contention states in the cost model; (2) a strategy to utilize the system contention level reflecting the combined net effect of dynamic factors on a query cost for the cost model development; (3) two algorithms to determine a good set of system contention states for dynamic environments with their system contention level following uniform and non-uniform distributions, respectively; (4) extensions of our previous static query sampling method including the determination of the minimum sample size for query sampling, the strategies and criteria to add/remove quantitative and qualitative variables, and the revised statistical measures to evaluate a multistate cost model; and (5) empirical studies on the accuracy of multistate cost models and the effectiveness of query optimization based on the models in dynamic environments.

A multistate cost model developed by the above qualitative approach is used to estimate the cost of a query run in any contention state in a dynamic environment. To estimate the cost of a query experiencing multiple contention states in a slowly-changing environment, a fractional analysis technique can be applied. To estimate the cost of a query experiencing multiple contention states in a frequently-changing environment, a probabilistic approach based on Markov chains can be adopted. The qualitative approach provides a base for the latter two techniques. These three techniques together comprise a complete suite of techniques to estimate the query costs for various dynamic multidatabase environments.

Although dynamic environmental factors have a significant effect on query cost, they were ignored in most existing cost model development techniques for MDBSs due to lack of appropriate techniques. This paper introduces some promising techniques to tackle the problem. However, our work is just the beginning of further research that needs to be done in order to fully solve all relevant issues for global query optimization in dynamic multidatabase environments.

Acknowledgments

The authors would like to thank Per-Åke Larson, Guy M. Lohman, and Frank Olken for their insightful suggestions for the work reported in this paper. Our grateful thanks are due to anonymous reviewers for their valuable comments and constructive suggestions for improving the paper. The preliminary work of this paper was presented at the 16th IEEE International Conference on Data Engineering [23].

References

[1] S. Adali, K.S. Candan, Y. Papkonstantinou and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of SIGMOD*, pp 137–48, 1996.

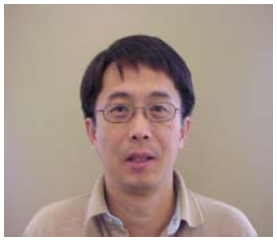
[2] L. Amsaleg, M.J. Franklin, A. Tomasic and T. Urhan. Scrambling query plans to cope with unexpected delays. In *Proc. of PDIS*, pp 208–19, 1996.
[3] G. K. Attaluri, D.P. Bradshaw, N. Coburn, P.-A. Larson, P. Martin, A. Silberschatz, J. Slonim and Q.Zhu. The CORDS multidatabase project. *IBM Systems Journal*, 34(1):39–62, 1995.
[4] L. Bouganim, F. Fabret, C. Mohan and P. Valduriez. Dynamic query scheduling in data integration systems. In *Proc. of ICDE*, pp 425–34, 2000.
[5] W. Du, R. Krishnamurthy and M.-C. Shan. Query optimization in heterogeneous DBMS. *In Proc. of VLDB*, pp 277–91, 1992.
[6] W. Du, M. C. Shan, and U. Dayal. Reducing multidatabase query response time by tree balancing. In *Proc. of SIGMOD*, pp 293 – 303, 1995.
[7] G. Gardarin, F. Sha and Z.-H. Tang. Calibrating the query optimizer cost model of IRO-DB, an object-oriented federated database system. In *Proc. of VLDB*, pp 378–89, 1996.
[8] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Academic Press, 2001.
[9] C. Lee and C.-J. Chen. Query optimization in multidatabase systems considering schema conflicts. *IEEE Trans. on Knowledge and Data Eng*., 9(6):941–55, 1997.
[10] W. Litwin, L. Mark and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Comp. Surveys*, 22(3):267–293, 1990.
[11] H. Lu and M.-C. Shan. On global query optimization in multidatabase systems. In *2nd Int'l workshop on Research Issues on Data Eng.*, pp 217, Tempe, Arizona, USA, 1992.
[12] H. Naacke, G. Gardarin and A. Tomasic. Leveraging mediator cost models with heterogeneous data sources. In *Proc. of 14th Int'l Conf. on Data Eng.,* pp 351–60, 1998.
[13] J. Neter, W. Wasserman and M.H. Kutner. Applied Linear Statistical Models, 3rd Ed. Richard D. Irwin, Inc., 1990.
[14] K. W. Ng, Z. Wang, R.R. Muntz and S. Nittel. Dynamic query re-optimization. In *Proc. of 11th Int'l Conf. on Sci. and Stat. DB Manag.*, pp 264–273, 1999.
[15] E. Parzen. Stochastic Processes. Holden-Day, Inc., 1962.
[16] R.C. Pfaffenberger and J.H. Patterson. Statistical Methods for Business and Economics. Richard D. Irwin, Inc., 1987.
[17] M. T. Roth, F. Ozcan and L. M. Haas. Cost models DO matter: providing cost information for diverse data sources in a federated system. In *Proc. of VLDB*, pp 599–610, 1999.
[18] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, Sept. 1990.
[19] Urhan, T., M. J. Franklin and L. Amsaleg. Cost-based Query Scrambling for Initial Delays. In *Proc. of SIGMOD*, pp 130–141, 1998.
[20] T. Urhan, M.J. Franklin and L. Amsaleg. Cost-based query scrambling for initial delays. In *Proc. of SIGMOD*, pp 130–41, 1998.
[21] Q. Zhu, S. Motheramgari and Y. Sun. Cost Estimation for Queries Experiencing Multiple Contention States in Dynamic Multidatabase Environments. In *Knowledge and Infor. Syst*., 5 (1): 26–49, 2003.
[22] Q. Zhu, J. Haridas and W. Hou. Global query optimization based on multistate cost models for a dynamic multidatabase system. In *Proc. of Int'l Conf. on Enterprise Infor. Syst. (ICEIS'03), Vol.1*, pp 144–155, 2003. Also in Enterprise Information Systems V (Selected Best Papers of ICEIS'03), pp. 117-128, Kluwer Publishers, 2004.

[23] Q. Zhu, Y. Sun and S. Motheramgari. Developing cost models with qualitative variables for dynamic multidatabase environments. In *Proc. of ICDE*, pp 413-424, 2000.
[24] Q. Zhu and P.-A. Larson. A fuzzy query optimization approach for multidatabase systems. *Int'l J. of Uncertainty, Fuzziness and Knowledge-Based Sys.*, 5(6):701 – 22, 1997.
[25] Q. Zhu and P.-A. Larson. Solving local cost estimation problem for global query optimization in multidatabase systems. *Distributed and Parallel Databases*, 6(4): 373 – 420, 1998.
[26] Q. Zhu and P.-A. Larson. Building regression cost models for multidatabase systems. In *Proc. of 4th IEEE Int'l Conf. on Paral. and Distr. Inf. Syst*., pp 220–31, Dec. 1996.
[27] Q. Zhu and P.-°A. Larson. A query sampling method for estimating local cost parameters in a multi database system. In *Proc. of 10th IEEE Int'l Conf. on Data Eng.,* pp 144–53, Feb. 1994.
[28] Q. Zhu. Query optimization in multidatabase systems. In *Proc. of the 1992 IBM CAS Conference*, Vol. II, pp 111–27, Toronto, Canada, Nov. 1992.
[29] Q. Zhu, J. Haridas and W. Hou. Query optimization via contention space partitioning and cost error controlling for dynamic multidatabase systems. *Distributed and Parallel Databases*, 23(2): 151-188, 2008.

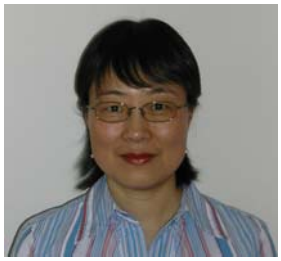
Qiang Zhu received his Ph.D. degree in computer science from University of Waterloo, Canada, in 1995. He also holds an M.Sc. degree from McMaster University, Canada, and an M.Eng. degree from Southeast University, China. He is presently a professor in computer and information science at The University of Michigan, Dearborn, USA. He is also an IBM CAS Faculty Fellow. His current research interests include query optimization, multidatabase systems, multidimensional indexing, self-managing databases, data streams, and Web information systems.

Yu Sun received an M.Eng. degree in computer science and a B.Sc. degree in mathematics from the Southeast University, China, in 1988 and 1982, respectively. He was a research associate in the department of computer and information science at The University of Michigan, Dearborn, USA. He was also a faculty member in computer science at Southeast University, China. His current research interests include query processing and optimization, multidatabase systems, and database statistical techniques.

Satyanarayana Motheramgari received an M.Sc. degree in computer and information science from The University of Michigan, Dearborn, USA, in 2002, another Master degree from Indian Institute of Technology, Kanpur, India in 1985, and a Bachelor degree with distinction from Osmania University, India, in 1983. He is presently a senior software engineer at Terumo CardioVascular Systems Corporation, USA. His current research interests include stochastic analysis of complex systems, database query cost modeling, and multidatabase systems.



Wen-Chi Hou received his M.Sc. and Ph.D. degrees in computer science and engineering from Case Western Reserve University, Cleveland, USA, in 1985 and 1989, respectively. He is presently an associated professor in computer science at the Southern Illinois University at Carbondale, USA. His current research interests include statistical databases, mobile databases, XML databases, and data streams.



Suyun Chen received her Ph.D. degree in applied mathematics from McMaster University, Canada, in 1995. She is presently a database specialist in Information Management at the BMO Financial Group, Canada. She was a database development analyst at the IBM Toronto Laboratory, Canada, and a faculty member at Jiangxi Normal Univeristy, China. Her current research interests include query optimization, data warehousing and mining, indexing methods, self-managing databases, and database performance.