Query Cost Estimation through Remote System Contention States Analysis over the Internet

Weiru Liu
School of C&M
Univ. of Ulster
w.liu@ulster.ac.uk

Zhining Liao
School of C&M
Univ. of Ulster
z.liao@ulster.ac.uk

Jun Hong
School of C&M
Univ. of Ulster
j.hong@ulster.ac.uk

Abstract

Query processing over the Internet involving autonomous data sources is a major task in data integration. It requires the estimated costs of possible query plans in order to select the best one with the minimum cost. In this context, the cost of a query is affected by three factors: network congestion, server contention state, and complexity of the query. In this paper, we study the effects of both the network congestion and server contention state on the cost of a query. We refer to these two factors together as system contention states. We present a new approach to determining the system contention states by clustering the costs of a sample query. We construct two cost formulas for each of the system contention states respectively using the multiple regression process. When a new query is submitted, its system contention state is estimated first using either the time slides method or the statistical method. The cost of the query is then calculated using the corresponding cost formulas. The estimated cost of the query is further adjusted to improve its accuracy. Our experiments show that our methods can produce quite accurate cost estimates of the submitted queries to remote data sources over the Internet.

1 Introduction

To meet the growing needs for sharing pre-existing data sources over the Internet, data integration from a multitude of autonomous data sources has recently been a research focus. Query optimization is one of the major stages in data integration over the Internet. It requires the estimated costs of possible query plans to select the best query plan in terms of costs. The key challenges arise due to the dynamics and unpredictability of the workloads of both the network and remote servers, and the autonomy of remote data sources. These sources may not provide necessary metrics for accurate cost estimation. Therefore in such an environment, it has become necessary to estimate query costs [9].

Several methods have been proposed for query cost estimation (e.g., [1, 3, 8, 10, 13-15]). All these methods assume that both the network itself and remote servers do not change significantly over time. Therefore, the impact of these two factors has not been explicitly

considered in query cost estimation. In this paper, we refer to these two factors together as system contention states.

The significance of recognizing the impact of system contention states has recently been studied. In [16, 17], the effects of the workload of a server on the cost of a query are investigated and a method to determine the contention states of a server is developed. Cost models are derived through sampling queries for each contention state for estimating the costs of new queries. This research has however been concentrated only on the workload of a server (contention states) in the local network environment and it is assumed that the network is steady and transferring data over the network involves very little cost. In [5, 12], on the other hand, the importance of coping with the dynamic network environment is addressed. The effects of the network are investigated and a cost estimation model is proposed for estimating the costs of the same query in different network situations, e.g., different times of the day. This method splits the large time interval, over a period of seven days (of a week), into time slides, with each slide being further split based on the quantity of data being transferred over the network in the time slide. A Multi-Dimensional Table (MDT) is built to establish how the cost of a query varies in terms of *Day, Time,* and *Quantity of Data*. A *predict response time* is attached to each cell of the MDT which gives the estimation of the cost of a query belonging to the cell.

Table 1. An intermediate MDT structure

Day Monday-Friday								Saturday-Sunday
Time 8am-2pm				2pm-8pm			8pm-8am	12am-12am
Qty <200k	200k -400k	400k -600k	>600k	<200k	200k -400k	>400k	0-Max	0-Max

The main drawbacks of the method are twofold. First, the dimension of *Time* has the minimum scale of one hour. If a remote source is highly dynamic, hourly intervals may be too large to reflect the change of the server. Second, this approach considers only the quantity of data to be transferred and does not consider the variety of queries using different operators. The complexity of a query can significantly affect its response time even when the network has the same workload.

In this paper, we investigate how system contention states (the network behavior and the workload of a server) affect the cost of a query and propose our approaches to establishing the relationship between them. We assume that a higher system contention state implies a busier

system situation. For instance, if three system contention states are identified, then their corresponding system situations are *not busy*, *average*, and *busy* respectively.

We make three major contributions in the paper. First, we propose a clustering approach to determining the system contention states related to a remote server. A sample query is tested on a server at fixed time intervals over a period of 24 hours, and a set of costs are collected. These costs are then clustered into a number of groups, each determining one system contention state. Second, for each system contention state, two cost formulae for estimating the costs of unary and join queries respectively are constructed using the multiple regression model [2]. The estimated cost of a query is further adjusted in order to reflect the fact that the costs in each contention state can still be diverse. The adjustment leads to a more accurate cost estimation. Third, we determine the current system contention state using either the time slides method or the statistical approach at the time when a query is submitted to a remote server. The time slides method is adequate when the progression from one contention state to another is smooth. The statistical approach is more suitable when the progression is either fast or slow, but not beyond the scope of the two contention states involved. Our experiments show that our methods can produce quite accurate cost estimates of the queries submitted to remote data sources over the Internet. In addition, we have also carried out experiments on recording and simulating the network speed. Our initial study suggests that by modelling the patterns of the network speed, we will be able to estimate the pattern of the workload of a remote server. This knowledge would be useful when multiple sources are available, as we can then select a server with a lower workload.

The rest of the paper is organized as follows. Section 2 describes the clustering algorithm for determining the system contention states, and the extension of the contention state from discrete time points to continuous time points using the time slides method. Section 3 discusses how cost formulae can be obtained through the multi-regression process and how the adjustment is made to minimize the error of cost estimation. Section 4 investigates the selection of the right cost formula and introduces the statistical approach to determining the contention state when the time slides method is not applicable. In Section 5, we present our experimental results and investigate how to use the knowledge on the dynamics of the network to estimate the pattern of a server's workload. Section 6 concludes the paper.

2 Determining System Contention States

To establish the relationship between a system contention state and the cost of a query, a sample query is carefully designed, which is of reasonable complexity and can be quickly evaluated by the remote server. The sample query is tested on the remote server at fixed time intervals over a period of 24 hours, and the tests are repeated for many times. The (average) costs of the query at these time points are collected. Below, *Ti* denotes the cost of the query at time point *ti.*

2.1 Clustering query costs

Our clustering algorithm places each data object (a cost at a clock time point) in its own cluster initially and gradually merges clusters into larger ones until a desirable number of clusters are left. The criterion for merging two clusters C_1 and C_2 is that they have the minimum distance. A widely used distance measuring technique is to measure the distance between centroids or means of two clusters. Assume *mean* (C_1) and *mean* (C_2) are the means of two clusters, then the distance between them is $D_{mean}(C_1,C_2) = |mean(C_1)-mean(C_2)|$. When more than one pair of clusters has the same minimum distance, the pair containing more objects in its union than other pairs is merged first. Let K be the maximum number of possible system contention states. The clustering algorithm constructs clusters $\Omega_k = \{C_1, ..., C_k\}$ such that $mean(C_i) < mean(C_{i+1})$ for $i = 1, ..., k$ based on a set of costs (T_i) at time points (t_i).

The algorithm below creates K clusters to generate K system contention states, with a set of costs in *T_cost* (on a one-dimension axis in terms of cost time) as an initial input. In this algorithm, we assume that *Temp* is a set of records, each record consisting of six slots:

Temp[i].name: the name of the temporary cluster *i*,

Temp[i].min : the minimum cost (value) in cluster *i,*

Temp[i].max : the maximum cost (value) in cluster *i*,

Temp[i].mean : the average cost (value) of cluster *i*,

Temp[i].dist : the distance between the means of two neighboring clusters, that is *Temp[i+1].mean-Temp[i].mean.* For the last cluster, *Temp[n]. dist* is defined as the whole interval of all the costs of the sample query. (So this distance is larger than the distance of any pair and will never be selected.)

Temp[i].num : the total number of individual costs (values) in cluster *i*.

Clustering Algorithm

Input: *T_cost* // a set of costs of the sample query

Initialization:

Rank(*T_cost*) // rank the elements of *T_cost* by the cost in ascending order

FOR *i=1* to Size(*T_cost*) DO // Size(S) returns the total number of elements in S

{

Temp[*i*].min:=Temp[*i*].max:=Temp[*i*].mean := *T_cost* [*i*]

Temp[*i*].num:=*1*

Temp[*i*].name:=C_i

}// initialize the values in Temp, put each initial cost in one cluster

FOR *i=1* to Size(*T_cost*)-*1* DO

Temp[*i*].dist:=Temp[*i+1*].mean-Temp[*i*].mean

Temp[*n*].dist:= *T_cost* [*n*]- *T_cost* [*1*] // the whole interval of the costs

Begin

WHILE Size(Temp) >*k* DO

{IF (C_i, C_{i+1}) is the pair with the minimum distance, or the pair with the minimum distance and with the most elements in its union than any other pair with the same distance

THEN //merge cluster C_i and C_{i+1}

{

Temp[*i*].max:= Temp[*i+1*].max

Temp[*i*].mean := (Temp[*i*].max –Temp[*i*].min)*/2* +Temp[*i*].min

Temp[*i*].dist := Temp[*i+2*].mean - Temp[*i*].mean

Temp[*i*].num := Temp[*i*].num + Temp[*i+1*].num

Temp[*i -1*].dist := Temp[*i*].mean – Temp[*i -1*].mean

Delete(Temp[*i+1*])

}

FOR *j=i+1* to Size[Temp]-1 do // move the clusters after C_i up

Temp[*j*].name:=Temp[*j+1*].name

Temp[*j*].min:=Temp[*j+1*].min

Temp[*j*].max:=Temp[*j+1*].max

Temp[*j*].mean:=Temp[*j+1*].mean

Temp[*j*].dist:=Temp[*j+1*].dist

Temp[*j*].num:=Temp[*j+1*].num

}

End

End Algorithm

The whole interval of the collected costs is *T_cost* [*n*]- *T_cost* [*i*]. Each cluster derived from the algorithm covers a section of the interval, and there is no overlap between any two clusters. In fact, any two neighboring clusters would have a gap, $max(C_{i+1})$-$min(C_i)$, between them, where $min(C_i)$ and $max(C_i)$ represent the minimum and maximum values in cluster C_i respectively. To let the system contention states cover any possible value in the interval *T_cost* [*n*]- *T_cost* [*i*], we define *k* system contention states from the *k* clusters as follows.

System contention state *one* is defined as

$$[min(C_1), (max(C_1)+ min(C_2))/2].$$

System contention state *i* is defined as

$$[(max(C_{i-1})+min(C_i))/2, (max(C_i)+ min(C_{i-1}))/2], \quad for\ i=2, \ldots k-$$

System contention state *k* is defined as

$$[(max(C_{k-1})+min(C_k))/2, max(C_k)].$$

To make the notation simpler in the rest of the paper, we let

$$lower(C_1)= min(C_1),$$

$$lower(C_i)=(max(C_{i-1}) + min(C_i))/2 \quad for\ i=2, \ldots k,$$

and

$$upper(C_i)=(max(C_i) + min(C_{i-1}))/2 \quad for\ i=1, \ldots k-1,$$

$$upper(C_k) = max(C_k).$$

Then,

$$C= \{[lower(C_1), upper(C_1)], \ldots, [lower(C_k), upper(C_k)]\}$$

defines k system contention states.

The computational complexity of the algorithm is in the order of $O(n^2)$ where n is the initial number of costs in *T_cost*.

2.2 The relationship between contention states and time slides

Let $T\_cost=\{T_1, T_2, \ldots, T_n\}$ be a set of costs of a sample query collected at clock time points $T\_time=\{t_1, t_2, \ldots, t_n\}$. We assume that ten minutes pass the midnight is the first time point the sample query is sent and $t_n=24.00$ is the last time point, and the query is sent at the interval of every ten minutes. The time interval can be changed to adapt to a specific application. When the elements in *T_cost* are divided into clusters C_i, the elements in *T_time* can be divided into corresponding clusters C_i', where $C_i'=\{t_i | T_i \in C_i$ and T_i is collected at time point $t_i\}$.

To extend the contention states at discrete time points to any time points, for each time slide [t_i, t_{i+1}] and the corresponding costs T_i, and T_{i+1} collected at t_i, t_{i+1} respectively, there are two possible scenarios.

Scenario one: if costs T_i and T_{i+1} are in the same cluster C_i, then a contention state at any time point in time slide [t_i, t_{i+1}] is defined by contention state [*lower*(C_i), *upper*(C_i)] through cluster C_i.

Scenario two: if costs T_i and T_{i+1} are in two different clusters C_i and C_j, then a contention state at any time point in time slide [t_i, $t_i+\alpha$) is [*lower*(C_i), *upper*(C_i)], a contention state in time slide ($t_i+\alpha$, t_{i+1}] is [*lower*(C_{i+1}), *upper*(C_{i+1})], and a contention state in time point $t_i+\alpha$ can be defined by either [*lower*(C_i), *upper*(C_i)] or [*lower*(C_{i+1}), *upper*(C_{i+1})], where $\alpha=(t_i+t_{i+1})/2$.

However, using different contention states at time point $t_i+\alpha$ may result in a rather different estimated cost. In Section 4 we discuss how to choose the correct cost formula for this case.

Therefore, given any specific clock time point, we can determine the system contention state by finding the time slide it falls in. It needs to be pointed out that this method is useful only when the progression from one contention state to another is smooth. In Section 4, we discuss the statistical approach to determining a system contention state when the progression is not smooth.

3 Cost Formulae for Query Cost Estimation

To construct cost formulae of both unary and join queries for each system contention state, we carry out multiple regression analysis [2, 14]. The multiple regression process allows us to establish a statistical relationship between the costs of a class of queries and the relevant contributing (explanatory) variables, in the form of a cost formula. This formula is used to estimate costs for other queries of the same class in the same system contention state.

3.1 Factors affecting the cost of a query

Five main factors affect the cost of a query over the Internet.

1) The number of tuples in an operand table.
2) The number of tuples in the result table.
3) The cardinality of an intermediate table.
4) The tuple length of the result table.

5) System contention states. A system contention state represents factors from both a server (e.g., CPU, I/O, query index method, etc.), and the network (e.g., speed and congestion).

The first two factors are commonly used in query cost estimation. The second and fourth factors together decide the data volume to be transferred over the Internet in unary queries. For a join query, factors 2-4 should be considered for the same purpose. Factor 5 covers the overall environment affecting the cost of a query in addition to factors 1-4. Other possible factors are either less important in the wide area environment (e.g., the tuple length of an operand table), or not available from autonomous data sources (e.g., the index method of a database).

3.2 Multiple linear regression cost models

Let $X_1, X_2, \ldots, X_p$ be p explanatory variables (They do not have to represent independent variables. It is allowed, for example, that $X_3 = X_1 \times X_2$), which may correspond to the factors we discussed above. The response (dependent) variable Y (which is the query cost in this application) tends to depend on explanatory variables X_s in a systematic way. If this dependency is a statistical linear relationship, which we assume is true in our application, a multiple linear regression model is defined as:

$$Y = B_0 + B_1 X_1 + B_2 X_2 + \ldots + B_p X_p + \sigma. \qquad (1)$$

Assume that there are n observations (given n values) of dependent variable Y through p explanatory variables $X_1, X_2, \ldots, X_p$, each observation defines an equation,

$$Y_i = B_0 + B_1 X_{i1} + B_2 X_{i2} + \ldots + B_p X_{ip} + \sigma \qquad (i = 1,2, \ldots, n) \qquad (2)$$

where X_{ij} $(j = 1, 2, \ldots, p)$ denotes the value of the *j-th* explanatory variable X_j in the *i-th* observation (or experiment); Y_i is the value of the dependent random variable Y in *i-th* observation. $B_1, \ldots, B_p$ are regression coefficients for p explanatory variables which are unknown constants and will be determined by solving multiple (more than p) equations like Eq. (1) in each contention state. σ is the error caused by noisy data.

In this paper, we look into unary and join queries separately, because they require different explanatory variables to build their cost formulae.

Unary query cost formula. For a unary query, we define R_U as its operand table, S_U as its selectivity of a conjunctive term, N_U as its cardinality of R_U, N_{rt} as its cardinality of the result table, L_r as its tuple length of the result table. Then $LN_{rt} = N_r \times L_r$ is the data volume of the result table transferred back to where the query is submitted. The cost formula for unary queries from the regression model described above is as:

$$Y = B_0 + B_1 \times N_U + B_2 \times N_{rt} + B_3 \times LN_{rt} \quad (3)$$

Join query cost formula. For a join query with two operand tables both in the same server, we let R_{U1} and R_{U2} be the two operand tables, N_{U1} and N_{U2} be the cardinalities of them, N_r be the cardinality of the result table, and L_r be the tuple length of the result table. Then $LN_r = N_r \times L_r$ is the data volume of the result table that needs to be transferred. The cost formula is defined as

$$Y = B_0 + B_1 \times N_{U1} + B_2 \times N_{U2} + B_3 \times N_r + B_4 \times LN_r \quad (4)$$

For join queries with operand tables in different servers, the cost formula requires some additional variables, such as, the data volume of intermediate results. This is however not addressed in this paper.

Given a system contention state [$lower(C_i)$, $upper(C_i)$] containing l costs, it is possible to calculate B_0, B_1, B_2, and B_3 (and B_4), in the above two cost formulae for unary and join queries respectively. When l is much larger than the total number of coefficients that we need to calculate, there can be many sets of possible combinations of costs (among l) to enable the calculation. The coefficients calculated from these different sets are usually different and will result in different cost estimates of the same query.

3.3 Adjustments to cost formulae

In this section, we take the unary query cost formula as an example to discuss how to select the costs of a sample query to calculate coefficients and how to adjust an estimated cost. The query cost formula for join queries can be done similarly. Assume that l costs ($l>>4$) fall into cluster C_i (for contention state [$lower(C_i)$, $upper(C_i)$]) after the costs in T_cost are clustered. We choose 4 out of l costs in C_i which are among the closest to $mean(C_i)$. These four costs are then used to calculate the coefficients in Eq. (3) for contention state [$lower(C_i)$, $upper(C_i)$]. Obviously, if we had chosen four smallest (or largest) costs from C_i to get the coefficients, we would have had rather different values for them. To solve this problem, we modify Eq. (3) as follows.

$$Y' = B_0 + B_1 \times N_U + B_2 \times N_r + B_3 \times LN_r + \sigma(Y, C_i) \quad (5)$$

where $\sigma(Y, C_i)$ represents the adjustment to the estimated cost of a query and is defined as:

$$\sigma(Y, C_i) = ((T_i^{tj} - mean(C_i))/mean(C_i)) \times Y_Q \quad (6)$$

Here $Y_Q = (B_0 + B_1 \times N_U + B_2 \times N_r + B_3 \times LN_r)$ is the estimated cost of query Q at time point t (when the query is submitted) using Eq. (3) at system contention state [$lower(C_i)$, $upper(C_i)$]. T_i^{tj}

$\in$ *T_cost* is the cost within C_i at observation time point $t_j \in$ *T_time* which is the closest to time point *t*. The adjustment states what fraction of the estimated cost should be deducted from or added to the estimated cost. For example, when T_i^{tj} almost equals to *mean*(C_i), the adjustment is close to *0*. This means that using Eq. (3) itself can get the correct estimate. The adjustment is however significant when $|lower(C_i)-mean(C_i)|/mean(C_i) >= 10\%$ (or $((upper(C_i)-mean(C_i))/mean(C_i) >= 10\%$), since this indicates that in this contention state, there is a big difference between the lowest (highest) and average costs of the same query. In other words, in the same system contention state, the system is busier at certain times than other times. On the other hand, if $|lower(C_i)-mean(C_i)|/mean(C_i) < \varepsilon$ (a pre-defined threshold, e.g., 10%) (or $|(upper(C_i)-mean(C_i)|/mean(C_i) < \varepsilon$), the adjustment is not needed, since this indicates that there is no significant difference between any two estimated costs of the same query at any two time points in this same contention state, e.g., the workload of the system is almost constant.

4. Determining the Current Contention State and Selecting the Correct Cost Formula

4.1 Selecting the correct cost formula using the time slides method

Assume that query *Q* is submitted to a remote server, and the progression from one system contention state to another is smooth. Let *t* be the clock time when *Q* is submitted and fall between two time points t_i and t_{i+1} (or time slide $[t_i, t_{i+1}]$) in *T_time*. Let T_i and T_{i+1} be the two corresponding costs obtained at these two time points. Assume that the two clusters containing T_i and T_{i+1} are C_i and C_j respectively. We now use a unary query as an example to show how to select the correct cost formula (especially the variable in the adjustment).

- **Scenario one**: when $C_i = C_j$, the workload of the system is steady and the contention state is [*lower*(C_i), *upper*(C_i)]. Using Eq. (5), the cost of *Q* is estimated as

  $Y' = B_0 + B_1 \times N_U + B_2 \times N_r + B_3 \times LN_r + \sigma(Y, C_i)$,

  with $\sigma(Y, C_i) = \{(T_i^{ti} - mean(C_i))/mean(C_i)\} \times Y_Q$, where $T_i^{ti} = T_i$ when $|t_i - t| < |t_{i+1} - t|$; $T_i^{ti} = T_{i+1}$ when $|t_i - t| > |t_{i+1} - t|$; and $T_i^{ti} = mean(C_i)$ when $|t_i - t| = |t_{i+1} - t|$.
- **Scenario two-A**: when $lower(C_i) < lower(C_j)$, the workload of the system is increased from state [*lower*(C_i), *upper*(C_i)] to [*lower*(C_j), *upper*(C_j)]. Using Eq. (5), the cost of *Q* is estimated as $Y' = B_0 + B_1 \times N_U + B_2 \times N_r + B_3 \times LN_r + \sigma(Y, C_i)$, with $\sigma(Y, C_i) = \{(T_i^{ti} - mean(C_i))/mean(C_i)\} \times Y_Q$ where $T_i^{ti} = upper(C_i)$ when $|t_i - t| < |t_{i+1} - t|$; $T_i^{ti} = lower(C_j)$ when $|t_i - t| > |t_{i+1} - t|$; and $T_i^{ti} = (upper(C_i) + lower(C_j))/2$ when $|t_i - t| = |t_{i+1} - t|$.

- **Scenario two-B**: when $lower(C_i) > lower(C_j)$, the workload of the system is decreased from state [$lower(C_i)$, $upper(C_i)$] to [$lower(C_j)$, $upper(C_j)$]. Using Eq. (5), the cost of Q is estimated as $Y' = B_0 + B_1 \times N_U + B_2 \times N_r + B_3 \times LN_r + \sigma(Y, C_i)$, with $\sigma(Y, C_i) = \{(T_i^{ti} - mean(C_i))/mean(C_i)\} \times Y_Q$ where $T_i^{ti} = lower(C_i)$ when $|t_i - t| < |t_{i+1} - t|$; $T_i^{ti} = upper(C_j)$ when $|t_i - t| > |t_{i+1} - t|$; and $T_i^{ti} = (lower(C_i) + upper(C_j))/2$ when $|t_i - t| = |t_{i+1} - t|$.

4.2 Determining contention states using a statistical method

In this section, we develop a statistical model that uses a statistical function to describe the progression from system contention state C_i to C_j, when the progression is not smooth. Assume that we have collected a number of costs of the sample query within the two time points t_i and t_{i+1} (or time slide [t_i, t_{i+1}]), and the contention states at these two time points are represented by clusters C_i and C_j. The distribution of these costs can be illustrated using one of the diagrams below. Fig. 1a shows that in time interval (slide) [t_i, t_{i+1}] the cost increases, and Fig. 1b shows that the cost decreases. If some of the costs within the interval are above the maximum or below the minimum cost in C_i and C_j (Fig. 1c), there is at least another contention state occurring within the time interval. Therefore this time slide should be split. This topic is beyond the scope of this paper. Here, we assume that all the additional observed costs in the time slide [t_i, t_{i+1}] fall into interval [$min(lower(C_i), lower(C_j))$, $max(upper(C_i)$ $upper(C_j))$].

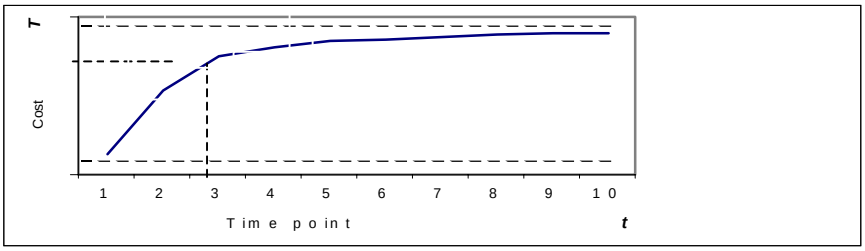

Figure 1a. Query costs increase from t_I to t_{i+1}

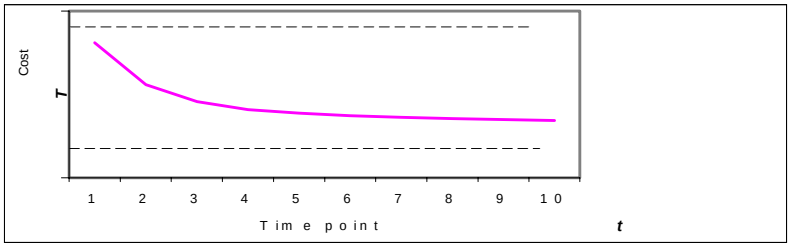

Figure 1b. Query costs decrease from t_I to t_{i+1}

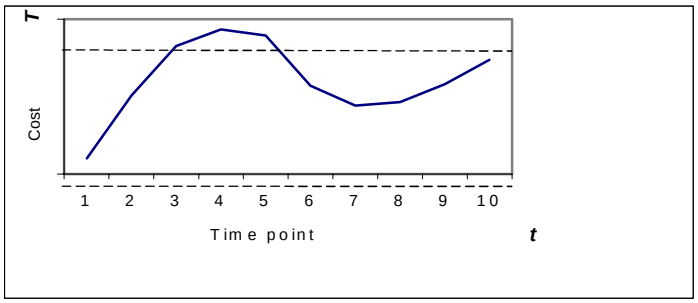

Figure 1c. More contention states between time points t_i to t_{i+1}

Figure 1. Possible progressions from one contention state to another. *t* and *T* represent time point and the cost of the sample query respectively. The dotted lines represent the lowest and the highest costs from the union of contention states C_i and C_j.

The best mathematical simulation of the observed data (as shown in Fig.1a and Fig. 1b) is a *function*

$$y = a \times e^{-b/t} \text{ with } (b>0) \qquad (7)$$

where *a* and *b* are two parameters, and *e* is the usual constant in mathematics with value *2.718282*. *t* is the time point when a query is submitted. When *t* is given, this function is used to estimate the cost *y* of the query at this time point. Parameters *a* and *b* are determined using the observed data (costs y_i) at certain time points (t_i). It is easy to see that this function is exponential. To reduce the computational complexity, we adopt the method proved in [5] and transform this function into a linear regression formula by changing the variables as $y' = \beta_0 + \beta_1 \times t'$ where $y' = log_e\, y$ and $t' = 1/t$. The relationship between these two pairs of variables is $\beta_0 = log_e\, a$, and $\beta_1 = -b$. β_0 and β_1 are calculated using equations below, known as *the least squares method*, for determining the parameters that best describe the relationship between the expected and observed sets of data minimizing the sums of the squares of deviation between observed and expected values.

Here $\beta_0 = \bar{y}' - \beta_1 \times \bar{t}'$

and

$$\beta_1 = (\Sigma_{i=1}^{n} (t_i' \times y_i') - n \times \bar{t}' \times \bar{y}') / (\Sigma_{i=1}^{n} t_i'^2 - n \times \bar{t}'^2)$$

with $\bar{t}' = 1/n\ (\Sigma_{i=1}^{n} t_i')$ and $\bar{y}' = 1/n\ (\Sigma_{i=1}^{n} y_i')$, and y_i ($y_i' = log_e\, y_i$) being the observed costs at time points t_i ($t_i' = 1/t_i$). Finally, the parameters (*a* and *b*) in the function are calculated by $a = e^{\beta 0}$

and $b=-\beta_1$. Assume n costs at n different clock time points are available in this time slide, then a and b can be calculated. We can then use Eq. (7) to estimate the cost of a sample query Q, denoted as y_Q, at time point t in time slide $[t_i, t_{i+1}]$. Based on the value of y_Q, we are able to decide which system contention state should be chosen to estimate the cost of a given query. For example, if a query is submitted just after time point 3 in the situation as shown in Fig. 1a, we choose the formula in cluster C_i with a suitable adjustment, based on the time slides method. However, the estimated cost of the sample query using this statistical method indicates that the contention state at time point 3 is closer to C_j and should use C_j rather than C_i. Therefore, in this circumstance, a statistical method is more accurate.

5 Experimental Results

5.1 Experimental settings

We have carried out experiments to evaluate the accuracy of our cost formulae in each system contention state. Our experiments were conducted between a remote server and a user. A server has been established at the University of Ulster using Windows 2000 as the operating system and Oracle 9.0 as database management system with 56 tables installed with tuple numbers ranging from 200 ~ 800,000. A sample query was sent from a user in China to this server a number of times at fixed time points and the corresponding query costs were collected. The costs collected at the same time points (in different runs) were averaged and a final set of (averaged) costs was used as an input to the clustering algorithm for deriving the system contention states. We obtained 4 systems contention states in our testing environment.

5.2 Analysis of cost estimation formulae

A number of unary sample queries were sent to the server to record the costs and the coefficients in the cost formula (Eq. (5)) were calculated for the unary query class in each contention state as shown in Table 2. Similarly, some sample join queries were sent by the user and the costs were used to calculate the coefficients of the cost estimation formula (Eq. (6)) for the join query class in each contention state as shown in Table 3.

A number of testing queries (both unary and join) were designed. We applied these cost formulae to estimate the cost of these queries in different system contention states before each of them was sent to the server for execution. The analysis below (Figures 2a-2d) shows the

comparison between estimated and observed costs in each contention state for both unary and join testing queries.

Table 2. Cost Estimation Formulae for Unary Query in each contention state

Con. States	Cost Estimation Formulae for Unary Queries in each contention state and a single state
State 1:	$Y = 1.84168657223 - 8.01832057E\text{-}5 * N_U + 0.24199654066 * N_r - 4.83872528E\text{-}6 * LN_r$
State 2	$Y = -5.1653349448 + 6.062131E\text{-}5 * N_U + 0.258267878 * N_r - 2.8606061E\text{-}4 * LN_r$
State 3	$Y = 2.09676928248E1 - 1.8766981E\text{-}4 * N_U - +0.272761212 * N_r + 3.9059394E\text{-}4 * LN_r$
State 4	$Y = 8.3451255319E1 - 7.1109199E\text{-}4 * N_U + 0.153626666 * N_r + 9.83466667E\text{-}4 * LN_r$
Single State	$Y = 2.4497292496E1 - 2.295809239E\text{-}4 * N_U + 0.2316508 * N_r + 2.707889437E\text{-}4 * LN_r$

Table 3. Cost Estimation Formulae for join Query in each contention state

Cont. States	Cost Estimation Formulae for Join Queries in each contention state and the single state
State 1:	$Y = 0.9538376 + 0.13936762E\text{-}3 * N_{U1} + 0.1707536E\text{-}3 * N_{U2} + 2.1415252E\text{-}4 * N_r + 6.0141673E\text{-}6 * LN_r$
State 2	$Y = 1.3361241 + 0.2215472E\text{-}3 * N_{U1} + 0.21611079E\text{-}3 * N_{U2} + 3.507236E\text{-}4 * N_r + 1.1179426E\text{-}5 * LN_r$
state 3	$Y = 1.91621544 + 0.5764138E\text{-}3 * N_{U1} + 0.7265162E\text{-}3 * N_{U2} + 9.0576512E\text{-}4 * N_r + 2.172937E\text{-}5 * LN_r$
state 4	$Y = 7.42515906E1 + 1.1109199E\text{-}3 * N_{U1} + 2.27620745E\text{-}3 * N_{U2} + 2.55655532E\text{-}3 * N_r + 7.717473E\text{-}5 * LN_r$
Single state	$Y = 6.977701891E1 + 5.07278115E\text{-}3 * N_{U1} + 0.91683423E\text{-}3 * N_{U2} + 1.19382323E\text{-}4 * N_r + 3.55945865E\text{-}5 * LN_r$

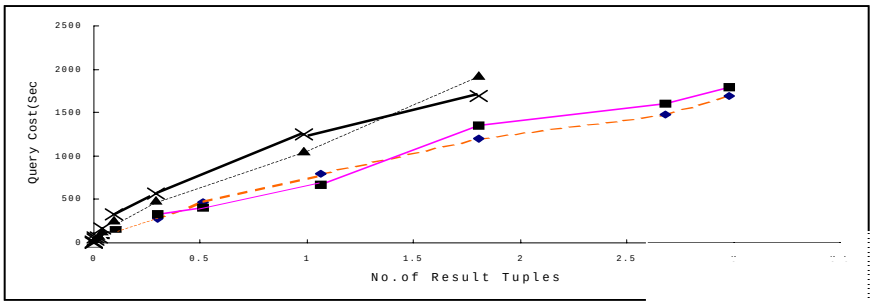

Figure 2a. Contention state 1

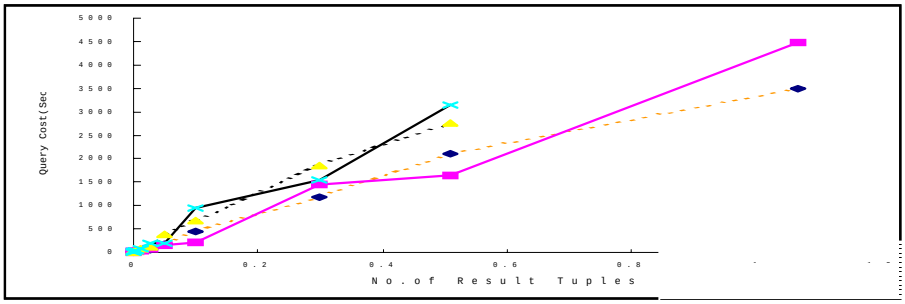

Figure 2b. Contention state 2

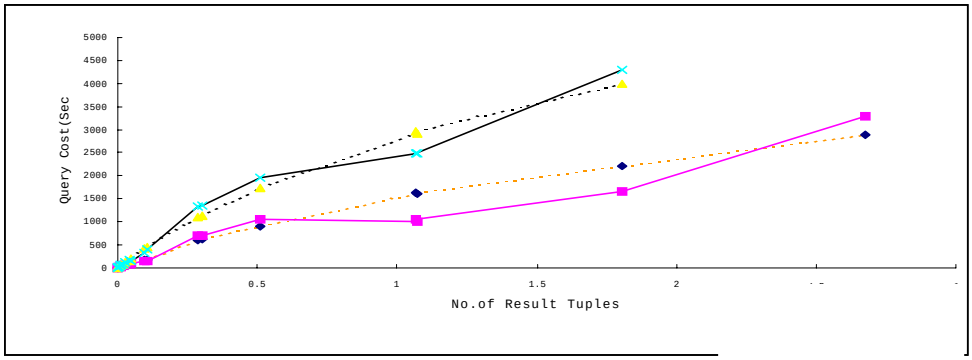

Figure 2c. Contention state 3

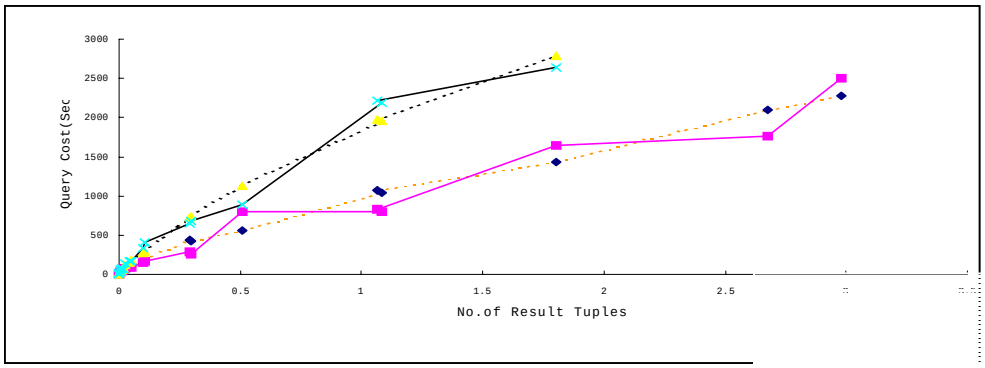

Figure 2d. Contention state 4

Figure 2. Estimated verses observed costs of test queries for both unary and join queries in each system contention state. Dashed lines with ♦ are for estimated costs of unary queries and solid lines with solid squares are for the observed costs of the same unary queries. Similarly, a dashed line with solid Δ and a solid line with × are estimated and observed costs for join queries respectively.

The error rate for each testing query is calculated by $\partial = 1/n \times (\Sigma_{i=1}^{n} ((Y_i - Y_e)^2)^{1/2}) / Y_a \times 100\%$, where Y_i –the i^{th} observed cost of the same query $(i = 1,2,\ldots,n)$, Y_a – the average observed cost of that query, Y_e – the estimated cost of the same query. The average error rates for unary testing queries in these four contention states are 6.912%, 7.517%, 12.272%, and 12.637% respectively. The average error rates for join queries are 9.162%, 14.838%, 21.314% and 13.686%. The error rate for join queries in contention state 3 is much higher than that in the other states. We are examining its causes.

On the other hand, we also investigated that if we did not distinguish system contention situations, and took the overall environment as almost steady, we obtained a single cost formula

for unary queries and one for join queries. Since the system is assumed static, a query will have exactly the same estimated cost at any time. The analysis below shows that a single cost formula gives much higher error rates at certain time points. This suggests that dividing a system into different workload based periods (contention states) is necessary to accurately estimate query costs.

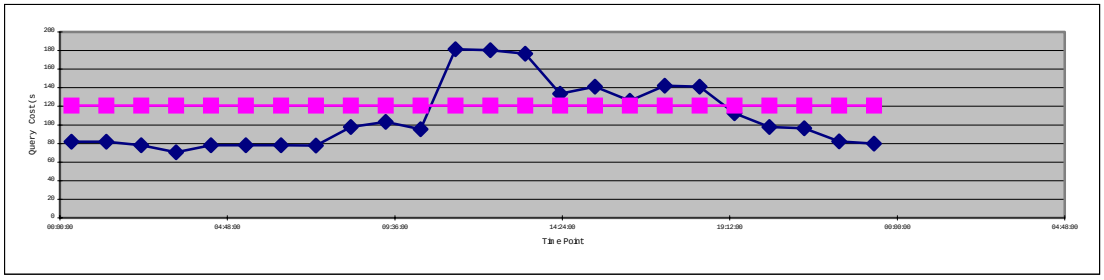

Figure 3. Estimated query costs verses observed costs in a single system contention state. The steady line shows the estimated cost of a unary query at any time point in it. The curved line shows the observed costs of the same query.

5.3 Analysis of network behavior

To find out how the network behavior affects the query cost, we conducted an experiment to measure the network speed at different time points (with a fixed interval 10 minutes). The network graph (Figure 4) reflects the average speed of a week's observation.

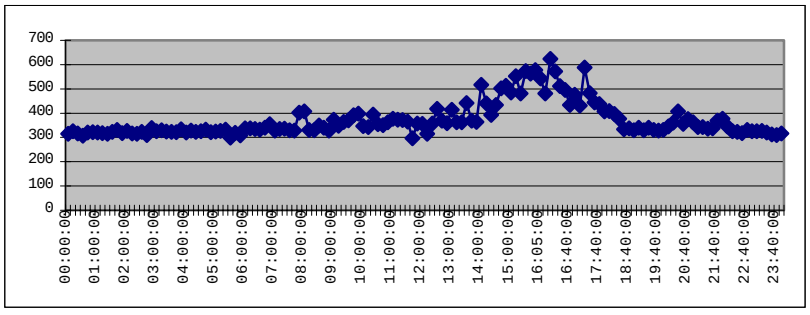

Figure 4. Network average speed data in a week

It is assumed that the recorded raw data about the speed of the network, *Nspeed(t),* is composed additively of a long-term signal *Nlspeed(t)* and a noise $n(t)$, that is *Nspeed(t)* = *Nlspeed(t)* + $n(t)$. If we are able to reduce $n(t)$ form *Nspeed(t),* then we can obtain the main trend of the network speed. In our study, the *Fourier transform* [6, 11] is applied to the raw

graph to identify the long-term signal *Nlspeed(t)* as it is constructed mainly from waves with low frequency (slow changes over time), while the noise signal is constructed from waves with high frequency (fast changes over time). The formulae are as follows.

The *n*-point (*n*=power of *2*) *Real Discrete Fourier Transform* of a signal

$$\vec{x} = [x_t],\ t=0,\ 1,\ \ldots,\ n\text{-}1$$

is defined to be a sequence $\vec{X}$ of $n/2+1$complex numbers X_f, f=0, 1, …, n/2, given by $X_f = R_f + i\, I_f$ where

$$R_f = \sum_{t=0}^{n-1} x_t \cos(2\pi ft/n) \text{ and } I_f = \sum_{t=0}^{n-1} x_t \sin(2\pi ft/\text{n}),\ \ f=0,\ \ldots,\ n/2$$

Here i is the imaginary unit. The signal $\vec{x}$ can be recovered by the inverse transform with the reduction of noise:

$$x_t = (R_0 + R_{n/2}\cos(\pi t))/2 + \sum_{f=1}^{n/2-1} R_f \cos(2\pi ft/n) + \sum_{f=1}^{n/2-1} I_f \sin(2\pi ft/n),\quad t=0,\ \ldots,\ n\text{-}1 \qquad (8)$$

A smooth graph of the revised network signals is illustrated in Figure 5.

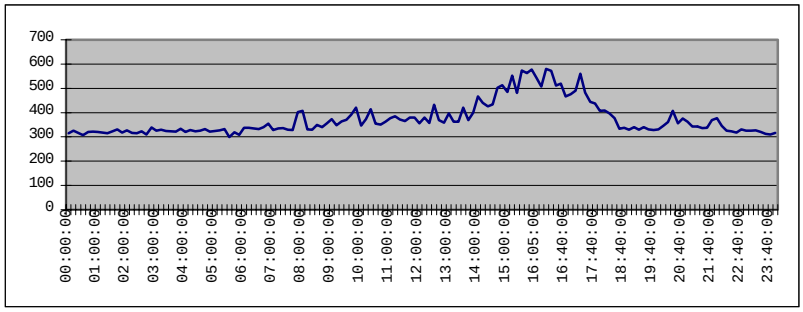

Figure 5. Using FT to transform the network speed data

With formula (8) simulating signal $\vec{x}$, we can use the *least squares method* [4] to generate a polynomial equation as shown below to describe a graph which best fits discrete data x_t, as illustrated in Figure 6.

$$Y = -0.0005*x^3 + 0.0863*x^2 - 2.5878*x + 334.7420 \qquad (9)$$

Based on formula (9), given any time (*x*), we will be able to estimate the likely state of the cost of the network on a query.

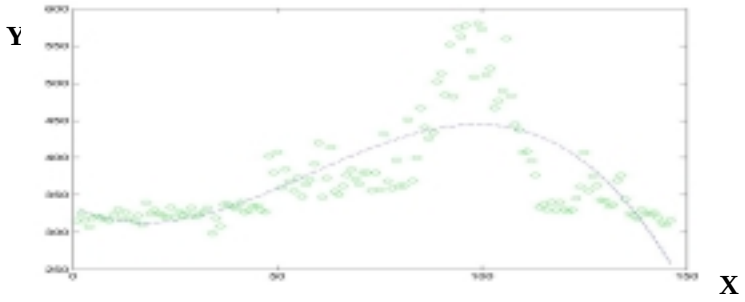

Figure 6 Distribution of discrete data (dotts) and their best simulation graph

Figures 6 and 7 show the comparison of the system contention states and the network behavior. The system contention states graph (Figure 7) is obtained by averaging the observed costs over the period of a week. From these graphs we can conclude that most of the time when the network is busy the system contention state is high. However, a situation that the network is not busy but the system contention state is still high indicates a busy state of the remote server. This comparison can be used to approximately model the pattern of the workload of a server.

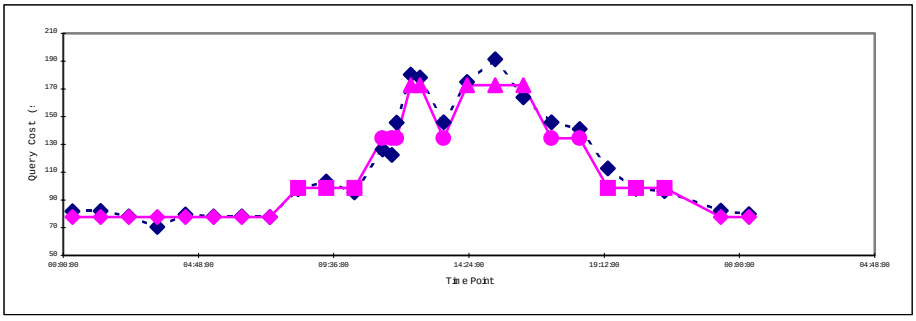

Figure 7. Summary of the system contention states

The contention states of a remote server is derived through the modelling of system contention states (Figure 7) and the pattern of the network behavior (Figure 6). The estimated server contention states, demonstrated in Figure 8, are consistent with the real situation of the server in our experiment, as we had run a number of application programs on the server irregularly during the testing period in order to avoid a steady server workload.

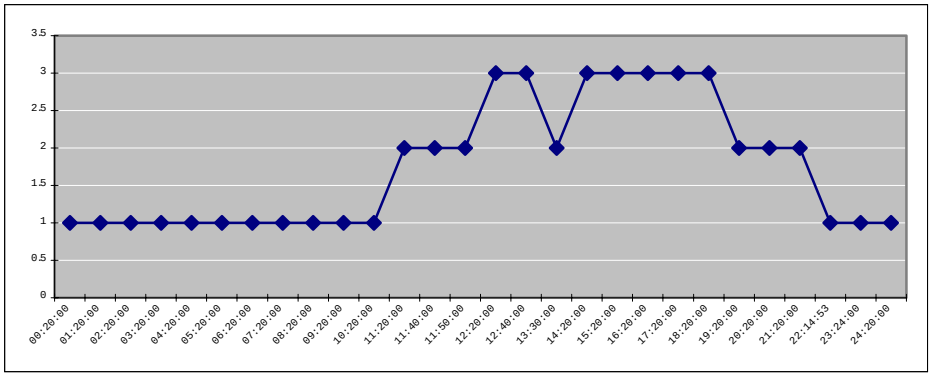

Figure 8. Estimated Server contention states

The estimation of the behavior of a remote server can be used at the query optimization stage to choose which server to use, when two (or more) servers are available with similar data. This knowledge can help to avoid the selection of a busier server.

6. Comparison with Related Work and Conclusion

In [16, 17], the relationship between the workload of a server and the cost of a query is investigated in terms of system contention states. A sampling query is sent to a remote server repeatedly to obtain sample costs. The cost interval determined by the minimum and maximum sample costs is divided into fix-length intervals first. Then some of these intervals are merged based on merging criteria until a certain number of intervals are left. Each interval determines one contention state. The sample costs in an interval are used to calculate not only the coefficients of regression cost formulae, but also the coefficients of a regression equation between the estimated costs of the sampling query and some of the major system parameters, such as, the CPU load, I/O utilization, and the size of used memory space of the remote server. When a query Q is submitted, these system parameters are obtained and the cost of sampling query Y is estimated. Based on the sub-interval that Y falls in, the correct cost formula is selected to estimate the cost. Both methods for determining system contention states and selecting the regression formula are different from our methods. Our clustering method with no fixed length is more accurate when determining contention states. Also, we do not require those system parameters for determining a contention state when a query is submitted, since these parameters may not be available in a wide area environment.

The common feature between our approach and the Multi-Dimensional Table (MDT) approach in [5, 12] is to use time slides to help decide the overall environment condition when a

query is submitted. However, the two approaches are completely different in terms of how the time slides are defined and used during the estimation of query costs. Our method takes into account more of the connection between the data volume and the query cost.

In summary, in this paper, we have proposed a method to determine the contention states of a system by clustering costs of a sample query. For each system contention state, we have established cost formulae for the unary query class and the join query class respectively. The system contention state, when a query is submitted, is determined using either the time slides method or the statistical method. As a consequence, the correct cost formula can be chosen for the query. The estimated cost is then adjusted to minimize estimation error. Our experimental results show that our methods can provide accurate query costs estimates.

Our immediate future work is to study the refinement of clusters when either two sets of clusters generated by two sample queries disagree with each other or the error rate of the estimated verses observed costs of queries is high. Our future work is to further investigate the behavior of the network and attempt to model the contention states of a server and the network more accurately.

References

1 Adali, S., Candan, K.S., Papakonstantinou, Y., and Subrahmanian, V.S. Query caching and optimization in distributed mediator systems. In Proc. of ACM SIGMOD'96, 137–48

2 Chatterjee, S. and Price, B. Regression Analysis by Example (2nd ed.) John Wiley & Sons, Inc. 1991

3 Du, W., Krishnamurthy, R., and Shan, M.C. Query optimization in heterogeneous DBMS. In Proc. of VLDB'92, 277–291

4 Folland, G.B. Fourier analysis and its applications Brooks/Cole Publishing Company', 1992

5 Gruser, J.R., Raschid, L, Zadorozhny, V., and Zhan, T. Learning response time for web-sources using query feedback and application in query optimization. VLDB Journal, 9(1), 18-37. 2000

6 Guo, G., Wang. H. and Bell, D.A., Data Reduction and Noise Filtering for Predicting Times Series. Proc. of WAIM'2002, 421-429

7 Hald, A. Statistical theory with engineering application. Jon Wiley & Sons, Inc. 1952

8 Muralikrishna, M. Dewitt, D.J. Equi-Depth histograms for estimating selectivity factors for multi-dimensional queries. In Proc. of SIGMOD'88, 28-36

9 Roth, M.T., Ozcan, F., and Haas, L.M. Cost models DO matter: providing cost information for diverse data sources in a federated system. In Proc. of VLDB'99, 599–610

10 Ling, Y. and Sun, W. A supplement to sampling-based methods for query size estimation in a database system. SIGMOD Record, 21(4), 12-15, Dec.1992

11 Wang, G., Huo, Z. and Shuo, J. Applied numerical method, Science Press in China, 1992

12 Zadorozhny, V., Raschid, L., Zhan, T., and Bright, L. Validating an Access Cost Model for Wide Area Applications. Cooperative Information Systems, Vol ? 371-385. 2001

13 Zhu, Q. and Larson, P.A. A Query Sampling Method of Estimating Local Cost Parameters in a Multidatabase System. Proc. of ICDE'94, 144-153

14 Zhu, Q. and Larson, P.A. Building Regression Cost Models for Multidatabase Systems. PDIS'96, 220-231.

15 Zhu, Q. and Larson, P.A. Solving local cost estimation problem for global query optimization in multidatabase system. Distributed and parallel databases, 6(4), 373-421, 1998

16 Zhu, Q., Motheramgari, S., and Sun, Y. Cost estimation for large queries via fractional analysis and probabilistic approach in dynamic multidatabase environments, Proc. of DEXA'2000, 509-525

17 Zhu, Q., Motheramgari, S., and Sun, Y. Developing cost models with qualitative variables for dynamic multidatabase environments. Proc. of ICDE'2000, 315-355