


1990

Efficient Schemes to Evaluate Transaction Performance in Distributed Database Systems

R. Mukkamala
Old Dominion University

S. C. Bruell

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_fac_pubs

 Part of the [Databases and Information Systems Commons](#), [Software Engineering Commons](#), and the [Theory and Algorithms Commons](#)

Repository Citation

Mukkamala, R. and Bruell, S. C., "Efficient Schemes to Evaluate Transaction Performance in Distributed Database Systems" (1990). *Computer Science Faculty Publications*. 114.
https://digitalcommons.odu.edu/computerscience_fac_pubs/114

Original Publication Citation

Mukkamala, R., & Bruell, S. C. (1990). Efficient schemes to evaluate transaction performance in distributed database systems. *Computer Journal*, 33(1), 79-89. doi:10.1093/comjnl/33.1.79

Efficient Schemes to Evaluate Transaction Performance in Distributed Database Systems

R. MUKKAMALA* AND S. C. BRUELL**

* Computer Science Department, Old Dominion University, Norfolk, Virginia 23529, USA

** Department of Computer Science, University of Iowa, Iowa City, Iowa 52242, USA

Database designers and researchers often need efficient schemes to evaluate transaction performance. In this paper, we chose two important performance measures: the average number of nodes accessed and the average number of data items accessed per node by a transaction in a distributed database system. We derive analytical expressions to evaluate these metrics. For general applicability, we consider partially replicated distributed database systems. Our first set of analytic results are closed-form expressions for these two measures. These are based on some fairly restrictive simplifying assumptions. When these assumptions are relaxed, no closed-form expressions exist for these averages. Hence, we develop an efficient algorithm to compute these averages.

Received August 1988, revised July 1989

1. INTRODUCTION

A distributed database system is a collection of cooperating nodes that contains a set of data items at each node. A user transaction can enter such a system at any one of the nodes. The receiving node, also known as the *coordinating* or *initiating* node, undertakes the task of locating the nodes that contain the data items required by the transaction. (In this paper, the basic unit of access in a database is referred to as a data item.) In a partially replicated database system, the *set of nodes* in the execution of the transaction is called the *participating node set*. During the execution of the transaction, messages are exchanged between the participating nodes and the coordinating node.

The average transaction response time, the transaction reliability, and the system cost (e.g., communication cost, storage cost, etc.) are the three most important performance measures of interest to a distributed database system designer.

The transaction response time is generally defined to be the time between when a user's request is received by the system and the time the user is informed of the result. Response time depends on various factors such as the system's model of transaction execution, the distribution of data, the degree of data replication, the concurrency and commit protocols, and the characteristics of the given transaction. When we consider an environment where the cost (or delay) of a message is approximately independent of the size of the message, then the transaction response time (or the cost of transaction execution) may be expressed in terms of the *number of messages* exchanged between the coordinating and participating nodes and the *number of data items accessed* at each of the participating nodes^{8,12}. When we consider high-speed networks, where local processing costs dominate the communication costs, estimating the average number of data items accessed per node is beneficial for transaction cost estimations¹².

The transaction reliability defines the probability with which a transaction may be executed successfully in an error prone environment. This metric also depends on the data distribution, data replication, the concurrency

and commit protocols, and the characteristics of the given transaction^{2,5}.

The cost of a distributed database system includes one time costs such as the hardware and software costs as well as run time costs such as the transaction execution costs. Among the transaction execution costs, communication costs are predominant in a distributed database system. This cost also depends on all the above mentioned factors^{9,12,18}.

Even though it is possible to formulate equations expressing transaction response times, transaction reliability, and system cost in terms of the above mentioned factors, the evaluation of these measures is extremely cumbersome and requires unreasonably high computation time. The evaluation of the exact values for these measures generally involves both analysis and simulation. Evaluation tools with such large execution times are certainly not acceptable to a database designer who needs to evaluate a number of such possible database configurations before arriving at a final design. To overcome these problems, designers and researchers generally resort to approximation techniques^{4,5,8}. These techniques reduce the computation time by making simplifying assumptions regarding data distribution, data replication, and transaction execution.

Due to the importance of the size of the participating node set, and the number of data items accessed at each of these nodes, this paper develops evaluation techniques for these two measures. Even though it is possible to derive the exact distributions for these two measures, the average values of these metrics are the ones that are frequently used by designers either to make early design decisions or to estimate other performance measures using these two average values. Hence, in this paper we only derive the average values for these two metrics.

To help fix ideas, suppose we are given the specifications of a database in terms of the number of nodes n , the number of data items d , and the global data distribution matrix GD . (Table 1 summarizes the notation used in this paper.) Let us assume that we are interested in the system's performance (in terms of the average number of participating nodes and the average

Table 1. Notation

Symbol	Description
a_j	Number of data items accessed at the j^{th} node
c	Number of copies of each data item
d	Number of data items in the database
d_{GA}	Average number of data items accessed per node in the set A , corresponding to the given GA
d_s	Average number of data items accessed per node by a query of size s
g	Number of data items in each group, $g = d/n$
g_k	Number of data items referenced by query from the k^{th} group
h	Number of data items at each node
i	Data item index
i_k	Number of data items in the k^{th} group
j	Node index
k	Group index
n	Number of nodes in the database
\bar{n}_s	Average number of nodes accessed by a query of size s
s	Number of data items accessed by a transaction
A	Set of nodes accessed by a transaction
D	Index set of data items in the database: $D = \{1, 2, \dots, d\}$
G_k	Set of data items in the k^{th} group
GA	A group-access n -tuple with $g_k, k \in N$ as elements
GD_{ij}	Global data directory that indicates the presence (or absence) of the i^{th} data item at the j^{th} node
J	An n -tuple with $a_j, j \in N$ as elements
N	Index set of nodes in the database; $N = \{1, 2, \dots, n\}$
Q	A particular query
Q_k	Set of nodes to which the k^{th} group of data items is allocated
S	Index set of data items referred to by a transaction

number of data items accessed per node) for query transactions of a given read-set size, say s . For simplicity, let us further assume that all data items are equally likely to be accessed and that every node has the same probability of being the coordinator for a transaction. Under these assumptions the straightforward algorithm to compute the required average values would involve enumerating all possible $\binom{d}{s}$ query transactions and

evaluating the required metrics for each query transaction. For typical values of d and s , say $d = 10000$ and $s = 5$, the algorithm would require the generation and evaluation of approximately 10^{18} transactions. Clearly, this algorithm is an impractical tool for either a database designer or a database analyst.

The algorithms developed in this paper have a time complexity that depends on n and s and is independent of d . The techniques developed in this paper should aid database designers (and researchers) in their early stages of design where quick and approximate solution techniques are preferred to time consuming but exact techniques. For example, prior to solving the data distribution problem (also known as the file allocation problem⁶), which is an NP-hard problem, a database designer may like to determine (approximately) the impact of the number of copies of data items on transaction performance. The techniques suggested in this paper are extremely useful in answering these requests at considerably low computational cost (and hence quick response time). The designer may use these results to specify constraints (or bounds) on the number of copies for the data items in the database. Such constraints help the data distribution algorithms by reducing the search space.

The balance of this paper is outlined as follows. Section 2 formally defines the problem under con-

sideration. Section 3 describes the underlying assumptions and the analytic solution method as applied to query transactions. Section 4 extends these results to accommodate update transactions. Section 5 further extends the analysis to a distributed database system that has a modified majority consensus algorithm⁶ as the concurrency control policy. Section 6 then extends this technique to a generalized distributed database system with no restrictions on the number of copies or the data distribution. Finally, Section 7 provides some concluding remarks.

2. PROBLEM STATEMENT

We are given the following parameters (and quantities that are easily derivable from these parameters):

- n , the number of nodes in the database
- N , the index set for the nodes in the database; $N = \{1, 2, \dots, n\}$
- d , the number of data items in the database
- D , the index set for the nodes in the database; $D = \{1, 2, \dots, d\}$
- GD , the global data directory that contains the location of each of the d data items; the GD matrix contains d rows and n columns, each of which is either a 0 or a 1, i.e., $GD_{ij} = 0$ or 1, $\forall i \in D$ and $\forall j \in N$
- s , the size of the read-set for a query transaction; clearly, $s \leq d$.

From these input parameters we are to determine (1) the average cardinality of a set A (that represents the set of nodes accessed by a transaction), and (2) the average number of data items accessed from the nodes in the set A . Note that A is not necessarily unique. For

example, if a transaction requires access to data items 1, 4, and 5 each of which is located at nodes 2, 3, and 4, some of the 7 possible choices for A are $A = \{2\}$, $A = \{3, 4\}$, and $A = \{2, 3, 4\}$.

In order to enumerate all possible sets A , we employ the following auxiliary quantities:

- S , the index set of the data referred to by a transaction; in the previous example $S = \{1, 4, 5\}$. The cardinality of S , $|S|$, must match our input parameter s , i.e., $|S| = s$. In addition, $S \subseteq D$.
- J , an n -component vector that represents the number of data items accessed by the transactions at different nodes in the database; a_j , the j^{th} element of J , represents the number of data items accessed by the transaction at the j^{th} node. In our previous example, if the number of nodes in the database were 5, then $J = \langle 0, 3, 0, 0, 0 \rangle$ corresponds to $A = \{2\}$; either $J = \langle 0, 0, 2, 1, 0 \rangle$ or $J = \langle 0, 0, 1, 2, 0 \rangle$ corresponds to $A = \{3, 4\}$; and $J = \langle 0, 1, 1, 1, 0 \rangle$ corresponds to $A = \{2, 3, 4\}$.

Our problem effectively reduces to enumerating all possible vectors J that satisfy the constraints of the problem (see below) and of determining the set A that corresponds to each vector J . By enumerating all vectors J and assigning proper weights to each vector, we can then compute the average cardinality of A as we will see later. For now, let us return to the constraints that each J vector ($J = \langle a_1, a_2, \dots, a_n \rangle$) must satisfy. Since the size of the read-set is s , the number of data items accessed must be s , i.e., $\sum_{j \in N} a_j = s$. In addition, the number of data items accessed at the j^{th} node must be less than or equal to the number of data items present at that node; i.e., given $J = \langle a_1, a_2, \dots, a_n \rangle$, $\sum_{i \in S} GD_{ij} \geq a_j \geq 0$, $\forall j \in N$. The set A is now easy to specify; again given $J = \langle a_1, a_2, \dots, a_n \rangle$, $A = \{j | j \in N \text{ and } a_j > 0\}$ with the stipulation that every data item from the read-set, S , is present at at least one node from the set of nodes to be accessed (A), i.e., $\sum_{j \in A} GD_{ij} \geq 1$, $\forall i \in S$.

We now formally summarize the statement of the problem under consideration: *Given a matrix GD with d rows and n columns, and a positive integer s ($s \leq d$), determine the average cardinality of a set A that contains a subset of the n columns such that the following conditions are satisfied:* (The average is over all possible sets S where S is as defined in Equation (4)).

$$\forall J = \langle a_1, a_2, \dots, a_n \rangle, \sum_{j \in N} a_j = s, \quad (1)$$

$$\sum_{i \in S} GD_{ij} \geq a_j \geq 0, \forall j \in N, \quad (2)$$

$$GD_{ij} = 0 \text{ or } 1, \forall i \in D \text{ and } \forall j \in N \quad (3)$$

$$S \subseteq D, |S| = s, \quad (4)$$

$$A = \{j | j \in N \text{ and } a_j > 0\}, \quad (5)$$

$$\sum_{j \in A} GD_{ij} \geq 1, \forall i \in S \quad (6)$$

$$N = \{1, 2, \dots, n\} \text{ and } D = \{1, 2, \dots, d\} \quad (7)$$

Two important problems are addressed in this paper. The first is to find the average value of the cardinality of the set A , over all sets S , such that Equations (1)–

(7) are satisfied. The second is to determine the average number of data items accessed from the nodes in set A .

2.1 An illustrative example

Consider a distributed database with 6 nodes ($n = 6$, $N = \{1, 2, \dots, 6\}$) and 12 data items ($d = 12$, $D = \{1, 2, \dots, 12\}$). The global distribution of data is given by the matrix GD of Table 2. Given a query Q with four data items in its read-set ($s = 4$), we are to determine the average number of nodes accessed by the query.

As mentioned in the introduction there are $\binom{12}{4}$ different possible values for S . Suppose we consider a query Q with a read-set $S = \{2, 5, 8, 9\}$. Then there are several possible values for J that satisfy Equations (1) and (2). Suppose we choose $J = \langle 2, 0, 1, 1, 0, 0 \rangle$. From Equations (1) and (5), we see that $A = \{1, 3, 4\}$ (since a_1, a_3 , and $a_4 > 0$). Hence, the number of nodes accessed, or the cardinality of A ($|A|$), is 3. The average number of data items accessed from the nodes in this set A is $4/3$.

Since for a given read-set S and a given data distribution matrix GD there are several possible values for J , the above computed averages (for the given S) would not be correct unless they are averaged over all possible values of J that satisfy the constraints of the problem. Enumerating these vectors and assigning proper weights to each vector, in order to compute the average number of nodes and data items accessed, is the subject of the next section.

The computation structure of the obvious approach that enumerates all possible vectors of S to compute the averages is shown in Figure 1. In this figure, each node represents computational information: input, output, or intermediate. The node with label (s, d) represents the input of s, d to the computation process. The nodes $S_1, S_2, \dots, S_{d'}$ represent distinct choices of data item sets (each of size s) to be accessed by a transaction of size s . Certainly, $|S_k| = s$. Given a value of s , there are $\binom{d}{s}$ (indicated by d' in Figure 1) distinct ways of choosing the S_k sets. Given a set of data items, say S_k , these items can be accessed in a number of ways from the nodes in the distributed system. The exact number

Table 2. Global Data Distribution Matrix, GD

Data items	Node number					
	1	2	3	4	5	6
1	1	1	1	0	0	0
2	1	1	1	0	0	0
3	0	1	1	1	0	0
4	0	1	1	1	0	0
5	0	0	1	1	1	0
6	0	0	1	1	1	0
7	0	0	0	1	1	1
8	0	0	0	1	1	1
9	1	0	0	0	1	1
10	1	0	0	0	1	1
11	1	1	0	0	0	1
12	1	1	0	0	0	1

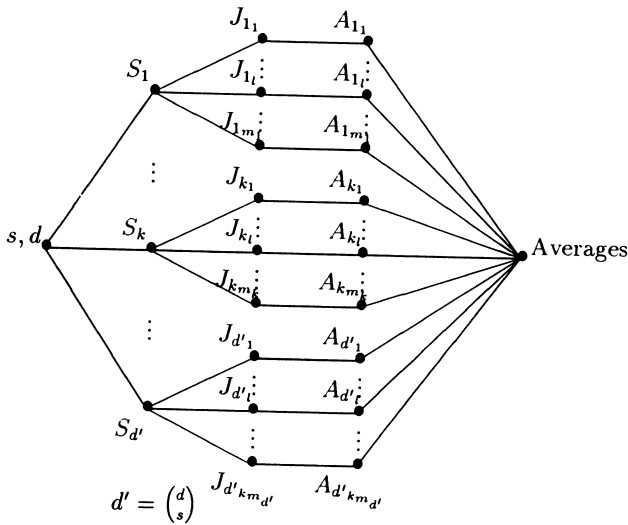


Figure 1. General computation structure

of such distinct ways is determined by the data set, the data distribution, and the search strategy adopted to locate the data items. For the data set S_k , we indicate this number by m_k . As before, we let J represent the data access vector. Thus, J_{kl} represents the l^{th} data access vector for the data item set S_k . Given a data access vector J_{kl} , the number of nodes accessed, or A_{kl} , is uniquely determined by Equations (1) and (5). Finally, the node labeled 'Averages' represents the final output of the computations.

3. A PROBABILISTIC APPROACH TO THE PROBLEM

Before attempting to solve the given problem, a number of key issues need to be considered.

It should be clear from the above description of the general computation structure that for any specific value of S , there are a number of values that can be assigned to J (i.e., the mapping $S_k \Rightarrow J$ is one-to-many). Each such assignment results in a specific value for A that satisfies Equations (1)–(7). However, the average number of nodes accessed (as defined in Section 2) is only defined over all possible values of S . This implies that it assumes a unique value of J (and hence A) associated with each of the possible values of S . To make the mapping $S_k \Rightarrow J$ one-to-one, we need to specify a *data copy selection policy*. This selection policy would remove one level of enumeration in the computation process (i.e., S_k to multiple J_{kl} s in Figure 1).

Similarly, if different values of S (i.e. S_k s), satisfying Equation (4), occur with different probabilities, then the average node computation should take these into account by assigning proper weights to different values of $|A|$ that result from the different values of S .

As discussed above, we need to determine the value of A for each possible value of S . But any algorithm

that enumerates all $\binom{d}{s}$ possible values for S in order to compute the average number of nodes to access, will be prohibitively expensive.

In order to develop a computational algorithm that is *efficient* and a *close* approximation to the exact value,

we make the following assumptions about the data distribution, the data copy selection policy, and the query distribution. Section 6 shows how the more restrictive of these assumptions can be relaxed.

3.1 Underlying assumptions

3.1.1 Data distribution assumptions

- All nodes in the database have the same number of data items, i.e.,

$$\sum_{i \in D} GD_{ij} = h, \forall j \in N \text{ and } h > 0 \quad (8)$$

- All data items in the database have the same number of copies, i.e.,

$$\sum_{j \in N} GD_{ij} = c, \forall i \in D \text{ and } 1 \leq c \leq n \quad (9)$$

- The data items in the database are categorized into n disjoint groups, each group G_k , $k \in N$, having the same number of data items $g = d/n$. The groups are allocated to the nodes in the following fashion: Group G_k is allocated to nodes in the set Q_k where

$$Q_k = \begin{cases} \{k, (k+1), \dots, (k+c-1)\}, \\ \{k, (k+1), \dots, n, 1, 2, \dots, (c+k-n-1)\}, \end{cases} \quad (10)$$

if $k+c-1 \leq n$ otherwise

Hence, each group corresponds to a set of c nodes. (This follows from Equation (9) which assumes that all data items have c copies.) This, our most stringent assumption, will be relaxed in Section 6.

All these three conditions will be removed in Section 6.

3.1.2 Data copy selection assumptions

- Each node has a copy of the matrix GD . The coordinator node of a query transaction searches this matrix in a *cyclic* manner starting from its own column in GD , in order to locate data items needed by the query. For example, if a query is received at node 1, then:

$$a_j = |B|, \text{ where } B = \{i | GD_{ij} = 1 \text{ and } GD_{il} = 0, 1 \leq l < j\}, \forall j \in N \quad (11)$$

where B represents the set of data items of the given query accessed from the j^{th} node. This means that a data item is always obtained from the node 'closest' (in cyclic order) to the coordinator node, if the data item was not found at the coordinator node.

3.1.3 Data item selection assumption

- All data items in the database are accessed with equal probability by any query transaction, i.e., $Pr(i \in S) = |S|/d, \forall i \in D$.

This is a commonly made assumption, and its implications are very well discussed by Chrisdoulakis^{3,4}.

3.1.4 Query distribution assumption

- A given query Q is equally likely to be received at any of the n nodes. Accordingly, the proposed analysis is

carried out for query transactions received at node 1. The average statistics computed for this node are equally applicable for any other node in the system.

3.2 The illustrative example revisited

Our illustrative example of Section 2.1 meets the underlying assumptions described above because:

- Each node has the same number of data items, $h = 6 (= d \cdot c/n)$.
- All data items have the same number of copies, $c = 3$.
- Each of the $n = 6$ groups has $g = 12/6 = 2$ data items. For example, $G_1 = \{1, 2\}$, $G_2 = \{3, 4\}$, $G_3 = \{5, 6\}$, $G_4 = \{7, 8\}$, $G_5 = \{9, 10\}$, and $G_6 = \{11, 12\}$.
- Each of the $\binom{d}{s}$ or $\binom{12}{4}$ possible choices for the read-set of Q are equally likely to occur.

Group G_k , $k = 1, 2, \dots, 6$, is allocated to the nodes in the set Q_k , where $Q_1 = \{1, 2, 3\}$, $Q_2 = \{2, 3, 4\}$, $Q_3 = \{3, 4, 5\}$, $Q_4 = \{4, 5, 6\}$, $Q_5 = \{5, 6, 1\}$, and $Q_6 = \{6, 1, 2\}$.

Suppose a query Q with a read-set $S = \{1, 3, 7\}$ is to be executed at node 1. Then following the 'cyclic' search order mentioned previously, we initiate the search from column 1 of GD (since we assumed that the query was received at node 1). Data item 1 is available at node 1, but data items 3 and 7 are not. Continuing the search for data items 3 and 7, we find that node 2 is the closest node that has data item 3. Finally, data item 7 is found at node 4. Hence, $J = \langle 1, 1, 0, 1, 0, 0 \rangle$, and $A = \{1, 2, 4\}$.

3.3 Computation of the averages for a query transaction

In developing an efficient procedure to compute the average performance (i.e., the number of nodes accessed and data accessed per node) our aim is to *reduce* the number of S values that need to be evaluated. This is achieved by introducing the concept of group access vectors. For ease of explanation, let us consider a query Q with a read-set size s that is received at node 1.

First, we introduce a new vector, the *group access vector*, $GA = \langle g_1, g_2, \dots, g_n \rangle$ where g_k represents the number of data item references by query Q from group G_k . The determination of J from this vector GA is straightforward and will be explained later.

Second, let Pr_{GA} represent the probability of occurrence of the n -tuple $GA = \langle g_1, g_2, \dots, g_n \rangle$ due to the query. There are $\binom{d}{s}$ ways of choosing s data items out of d data items. Since the n data groups (each with g data items) are distinct, there are $\binom{g}{g_1} \binom{g}{g_2} \dots \binom{g}{g_n}$ ways of choosing s data items to form a given $GA = \langle g_1, g_2, \dots, g_n \rangle$. Thus, Pr_{GA} may be expressed as:

$$Pr_{GA} = \frac{Pr_{\langle g_1, g_2, \dots, g_n \rangle}}{\binom{d}{s}} = \frac{\binom{g}{g_1} \binom{g}{g_2} \dots \binom{g}{g_n}}{\binom{d}{s}}$$

$$= \frac{\prod_{k=1}^n \binom{g}{g_k}}{\binom{d}{s}} \quad (12)$$

From Equation (10), when $c > 1$, the data groups distributed to node 1 are: $G_1, G_{n-c+2}, G_{n-c+3}, \dots, G_n$. From here, the probability that node 1 is accessed by query Q is derived as:

$$\begin{aligned} P_1 &= \text{Prob}(\text{node 1 is accessed}) \\ &= \text{Prob}(g_1 > 0 \vee g_{n-c+2} > 0 \vee g_{n-c+3} > 0 \vee \dots \vee g_n > 0) \\ &= 1 - \text{Prob}(g_1 = 0 \wedge g_{n-c+2} = 0 \wedge g_{n-c+3} = 0 \wedge \dots \wedge g_n = 0) \\ &= 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \end{aligned} \quad (13)$$

Since the query Q is received at node 1, using the searching rule in Equation (11), we can conclude that node 2 is accessed by this transaction only when $g_2 > 0$ (from Equation (10), Group 2 is the only group that is present in node 2 and not present in node 1). Thus,

$$\begin{aligned} P_2 &= \text{Prob}(\text{node 2 is accessed}) \\ &= \text{Prob}(g_2 > 0) \\ &= 1 - \text{Prob}(g_2 = 0) \\ &= 1 - \frac{\binom{d-g}{s}}{\binom{d}{s}} \end{aligned} \quad (14)$$

Using similar arguments we can show that,

$$P_3 = P_4 = \dots = P_{n-c+1} = 1 - \frac{\binom{d-g}{s}}{\binom{d}{s}} \quad (15)$$

Since the data items from the other groups (e.g., Groups $(n-c+2), (n-c+3), \dots, n$) are already at node 1, the corresponding nodes are never accessed by this transaction. Thus,

$$P_{n-c+2} = P_{n-c+3} = \dots = P_n = 0 \quad (16)$$

From Equations (13)–(16), the expected number of nodes to be accessed by Q may be computed as:

$$\bar{n}_s = \sum_{j=1}^n P_j = \left[1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \right] + (n-c) \left[1 - \frac{\binom{d-g}{s}}{\binom{d}{s}} \right] \quad (17)$$

\bar{n}_s is the desired average participating node set size for Q . From this equation, the expected number of data items accessed from each node in the participating set may be approximated as s/E_s . Thus,

$$\bar{d}_s \approx \frac{s}{\bar{n}_s} \quad (18)$$

From Equations (13)–(16), we can also compute the variance of these two estimated averages.

As mentioned above, even though we have performed this analysis for an arbitrary transaction received at node 1, the same average applies to every node, as well as to the entire database. The analysis may be carried out for all possible s values. By averaging these values (when transactions of different read-set sizes arrive into the system with different frequencies, then this would be a weighted average), we can obtain the average node set size for any transaction in the distributed database system.

4. COMPUTATIONS FOR UPDATE TRANSACTIONS

The analysis described in Section 3 pertains to a *query* transaction that requires accessing *any copy* of each of the data items in the read-set. This section deals with *update* transactions. (For simplicity, in this paper, we assume that an update transaction only performs update operations on the data items in its write-set.) Since the query analysis assumes a *read-any-one* copy policy, we assume *write-all* copies for update transactions.

The average node analysis for query transactions is extended to accommodate update transactions as follows. Since $GA = \langle g_1, g_2, \dots, g_n \rangle$ represents the distribution of write-set data items among the groups, we can make use of this information for updates also. If a data item, say $i \in S$, belongs to a group G_k , $k \in M$, then all the nodes in the database that contain the data items in G_k are to be accessed by the update transaction.

Again let us consider the transactions received at node 1. Let U be an update transaction that updates s data items. Since we assume a read-one/write-all policy, U has to access all c copies of each data item that it has to update. Under these conditions, a node has to be accessed whenever it contains a data item that U updates. Now, the expressions for the probability of node 1 being accessed by U may be written as:

$$\begin{aligned} P'_1 &= \text{Prob}(\text{node 1 is accessed by the update transaction}) \\ &= \text{Prob}(g_1 > 0 \vee g_{n-c+2} > 0 \vee g_{n-c+3} > 0 \vee \dots \vee g_n > 0) \\ &= 1 - \text{Prob}(g_1 = 0 \wedge g_{n-c+2} = 0 \wedge g_{n-c+3} = 0 \wedge \dots \wedge g_n = 0) \\ &= 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \end{aligned} \quad (19)$$

Since each of the n nodes have the same number of data items, and since each data item is equally likely to be updated by a transaction,

$$P'_1 = P'_2 = P'_3 = \dots = P'_n = 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \quad (20)$$

Thus, the expected number of nodes accessed by U is

written as:

$$\bar{n}'_s = n \left[1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \right] \quad (21)$$

5. COMPUTATIONS UNDER MAJORITY CONSENSUS

The analysis described in Sections 3 and 4 pertains to transaction executions under a *read-one/write-all* concurrency control policy. There are other replication control protocols, referred to as quorum consensus protocols, that are used in distributed database systems to improve the fault-tolerance of such systems^{10,17}. Under these protocols, each copy of a data item is assigned a positive integer indicating the number of votes assigned to it. In order to obtain the value of a data item, a set of copies of the data item that form a quorum, have to be accessed. The most up-to-date value of the data item is determined after processing values from this set of copies. Generally, a time-stamp or a version number is used to identify the most up-to-date value. We consider a simple scheme, referred to as the Majority Consensus Algorithm, that assigns one vote to each copy of a data item. Hence, a quorum is formed by any set containing a majority of the copies. For example, for a data item with c copies, any set of $\left\lceil \frac{c+1}{2} \right\rceil$ copies, would form a quorum. Similarly, updating the value of a data item requires the updating of at least a majority of the copies of that data item.

In this section, we extend our previous results to handle database systems that employ a majority consensus concurrency control algorithm. Since read as well as write operations on a data item require accessing a majority of the copies, the analysis dealing with query transactions is also applicable to update transactions. In the following discussion we shall simply refer to these transactions as query transactions.

Since $GA = \langle g_1, g_2, \dots, g_n \rangle$ represents the distribution of data items in the read-set among the groups, we shall use this information in the following computations. If a data item, say $i \in S$, belongs to a group G_k , $k \in N$, then a majority of the nodes in Q_k have to be accessed by the transaction to read the most up-to-date value of i .

Again let us consider query transaction Q received at node 1. Let $m = \left\lceil \frac{c+1}{2} \right\rceil$ represent the majority of copies that need to be accessed to read a data item. Let $m' = c - m$. We now need to analyze two cases: (i) $3 \leq c \leq n$ (ii) $c = 2$.

Let us consider case (i). Here $m' > 0$ and $m < c$. Using the copy selection criteria in Equation (11), we may classify the n nodes into three categories:

1. Nodes 1, 2, ..., m are accessed when any of the c groups that they contain are referred to by Q ;
2. Nodes $n - m' + 1$, $n - m' + 2$, ..., n are never accessed by Q since a majority of copies corresponding to their data groups are already available at nodes 1, 2, ..., $n - m'$;
3. Nodes $m + 1$, $m + 2$, ..., $n - m'$ are accessed when m of the c data groups that they contain are accessed.

We now compute the probability of access for nodes in each of these three categories.

Let us consider category 1. Since query Q has to access m copies of each of the s data items in its read-set, and since the search of the data item copies starts at node 1, nodes 1, 2, ..., m will be accessed by Q whenever they contain a data item (or group) in this read-set. For example, consider node 1. Since node 1 contains cg data items, the probability that it contains one of the s data items in the read-set is given by:

$$P_1 = 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \quad (22)$$

Since all nodes of category 1 are accessed with the same probability,

$$P_j = 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}}, 1 \leq j \leq m \quad (23)$$

Now consider node $n - m' + 1$ in category 2. It contains the groups: $G_{n-m'+2-c}$, $G_{n-m'+3-c}$, $G_{n-m'+1}$. For all these groups, the majority of copies (m) are available at nodes (1, 2, ..., $n - m'$). Thus, node $n - m' + 1$ is never accessed. Other nodes in category 2 have the same property. Thus,

$$P_j = 0, n - m' + 1 \leq j \leq n \quad (24)$$

Now consider node $m + 1$ of category 3. It contains c groups including group 1 (since $m < c$). Since it is the $m + 1^{\text{st}}$ node in the searching sequence, this node will never be accessed by query Q for data items in group 1. However, it should be accessed for any of the other $c - 1$ data groups. From Equation (13), the probability that node $m + 1$ is accessed by Q may be computed as:

$$P_{m+1} = 1 - \frac{\binom{d-(c-1)g}{s}}{\binom{d}{s}} \quad (25)$$

Similarly, we can argue that each node of category 3 is accessed only when at least one of the $c - 1$ groups are accessed by Q . Thus,

$$P_j = 1 - \frac{\binom{d-(c-1)g}{s}}{\binom{d}{s}}, m + 1 \leq j \leq n - m' \quad (26)$$

In summary, when $3 \leq c \leq n$,

$$P_j = \begin{cases} 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}}, & \text{if } 1 \leq j \leq m \\ 1 - \frac{\binom{d-(c-1)g}{s}}{\binom{d}{s}}, & \text{if } m + 1 \leq j \leq n - m' \\ 0, & \text{otherwise} \end{cases} \quad (27)$$

We now consider the case when $c = 2$. Here, $m = 2$, and $m' = 0$. Thus all copies of a data item are to be accessed whenever it is referred to by Q . Since each node has the same number of groups ($= c$), the probability that a node is accessed is given by

$$P_j = 1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}}, 1 \leq j \leq n \quad (28)$$

Using Equations (27)–(28) the expected number of data items may be computed as:

$$\bar{n}_s = \begin{cases} m \left(1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \right) \\ + (n - c) \left(1 - \frac{\binom{d-(c-1)g}{s}}{\binom{d}{s}} \right), & \text{if } c \geq 3 \\ n \left(1 - \frac{\binom{d-cg}{s}}{\binom{d}{s}} \right), & \text{if } c = 2 \end{cases} \quad (29)$$

The average data items accessed per node may be derived using Equations (18) and (29).

6. COMPUTATIONS FOR GENERAL DATABASES

In Sections 3–5, we described analysis methods to determine the average number of nodes accessed by a transaction in a database under certain restrictive assumptions (cf. Section 3.1). In fact, the implications of the assumptions such as uniformity of attribute values and uniformity of queries (in a centralized database system) is very well summarized by Christodoulakis⁴. In this section, we shall extend the previous analysis to enable us to compute the average statistics for a database system with no restrictions on the number of copies of a data item, the distribution of data copies, or the number of data items at each node.

In a partially replicated distributed database, the number of copies of a given data item in the database may be determined either by the application environment or by the user's access pattern to the data item. In essence, it is not necessary that all data items in the database have the same number of copies. However, in practical distributed database systems, the decision on the number of copies of a data item may not really be arbitrary. Similarly, in a practical database system the data item copy distribution is not entirely a random choice. The data distribution is generally determined based on the applications running on the system and, in fact, may follow some form of a group pattern. For example, if data items i , j , and k belong to a certain application, then they may be located together at the required nodes^{1,11,13,16}. Sometimes de-clustering techniques are applied to group the data items⁷. Hence, this

distribution may neither be as systematic as the one in our earlier analysis (cf. Section 3.1) nor be completely random. We now extend the previous analysis for databases with these characteristics.

Let us group the d data items in the distributed database based on the number of copies and on the allocation of these copies among the nodes. A data group, say G_k , is characterized by the two parameters x_k and y_k , where x_k represents the number of replication copies of each of the data items in the k^{th} group, and y_k represents the number of data items in this group, i.e., $y_k = |G_k|$. Each group of data items, say G_k , is allocated to a set of nodes, say Q_k ($x_k = |Q_k|$). The actual number of data groups in a database, ng , depends on the number of copies and the distribution of the copies among the nodes in the database. The distribution discussed in the previous sections is a special case of this general distribution wherein all data items have the same number of copies (i.e., $x_k = c$, for all k), and the size of each group G_k is the same (i.e., $|G_k| = g$, for all k).

The probability of occurrence of the ng -tuple $GA = \langle g_1, g_2, \dots, g_{ng} \rangle$ due to the read-set S is given by:

$$\begin{aligned} Pr_{GA} &= Pr_{(g_1, g_2, \dots, g_{ng})} \\ &= \frac{\binom{y_1}{g_1} \binom{y_2}{g_2} \dots \binom{y_{ng}}{g_{ng}}}{\binom{d}{s}} \\ &= \frac{\prod_{k=1}^{ng} \binom{y_k}{g_k}}{\binom{d}{s}} \end{aligned} \quad (30)$$

Since we are considering an arbitrary grouping of data items, and also an arbitrary distribution of groups, it is not possible to derive closed-form expressions for \bar{n}_s and \bar{d}_s as before. Instead, we use the computation structure shown in Figure 2.

In the modified structure, given the data grouping information (G_1, G_2, \dots, G_{ng}), and the s value, we generate all possible values for $GA = \langle g_1, g_2, \dots, g_{ng} \rangle$ so that $0 \leq g_i \leq y_k$ and $\sum_{i=1}^{ng} g_i = s$. These vectors are referred to as $GA_1, GA_2, \dots, GA_{d'}$. If Φ represent the

```

Procedure Determine_JQ( $GA$ : Integer_Array; var  $J$ : Integer_Array);
var
   $j, k$ : integer;
begin
  for  $j$ : = 1 to  $n$  do
     $J[j]$ : = 0;
  for  $k$ : = 1 to  $ng$  do
    if  $G[k] > 0$  then
      begin
        {Determine the minimum element of the set  $Q_k$ }
         $j$ : =  $\min\_element(Q_k)$ ;
         $J[j]$ : =  $J[j] + GA[k]$ 
      end
  end;

```

Figure 3. Generation of Vector J from GA for Query Transactions (General)

set of all these vectors, then $d' = |\Phi|$. For each value of GA , the corresponding node access vector J is computed. (For example, the procedure *Determine_JQ* in Figure 3 may be used to compute J vector for query transactions.) Having computed the value of J_k for a given GA_k , we may use Equations (1) and (5) to derive the value of A_k . Now, $|A_k|$ represents the number of nodes accessed by a given query that corresponds to the given GA_k ; let n_{GA_k} represent this quantity. Similarly, the average number of data items accessed per node in the set A_k , corresponding to the given GA_k , is denoted by d_{GA_k} ($= s/n_{GA_k}$).

For a given vector GA , the probability of its occurrence is computed using Equation (30). Having computed n_{GA} and d_{GA} for this vector, the overall averages \bar{n}_s and \bar{d}_s are computed as:

$$\bar{n}_s = \sum_{GA \in \Phi} Pr_{GA} n_{GA} \quad (31)$$

$$\bar{d}_s = \sum_{GA \in \Phi} Pr_{GA} d_{GA} \quad (32)$$

Figure 4 illustrates the overall structure of our algorithm and Figure 5 shows a Pascal procedure to generate all possible values of GA . When the number of data items in each group is greater than the read-set size of query Q (or write-set size for an update transaction U), i.e.

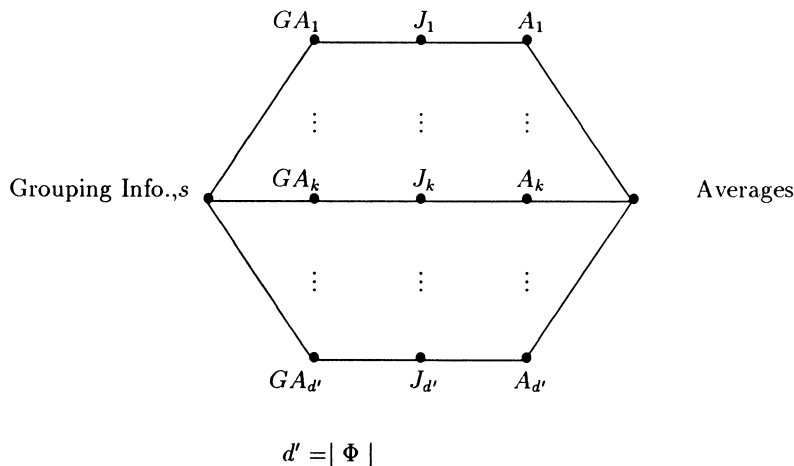


Figure 2. Modified computation structure

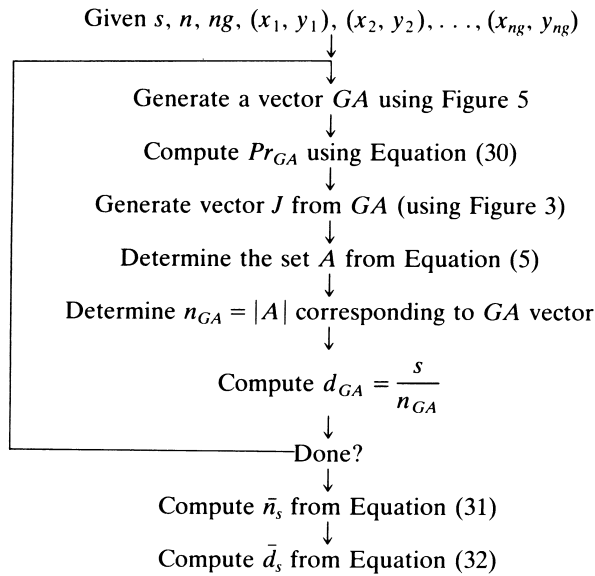


Figure 4. Algorithm to compute \bar{n}_s and \bar{d}_s

$y_k \geq s$ for all k , then the number of these vectors is given by $\binom{n+s-1}{s-1}$. The derivation of this expression is straightforward and not included here.

Since in a generalized database, it is not necessary for all nodes to have the same number of data items, the average statistics derived for transactions at one node may not be valid at other nodes. Accordingly, we may need to repeat this analysis for transactions at every node in the database.

6.1 Another illustrative example

Consider a distributed database with 6 nodes ($n = 6$) and 12 data items ($d = 12$). The global distribution of data is given by the matrix GD of Table 3. Given a query Q with $s = 4$ data items in its read-set, we are to determine the average number of nodes accessed by the query.

Suppose we consider a query Q with a read-set $S = \{2, 5, 8, 9\}$. Then there are several possible values for J

```

Procedure Generate_GA(s: integer);
var
  k: integer;
begin
  for k: = 1 to ng do
    GA[k] := 0;
    GA[1] := s;
    {The first GA vector is available here}
    while GA[ng] ≠ s do
      begin
        k := 1;
        while GA[k] = 0 do
          k := k + 1;
        GA[k] := GA[k] - 1;
        GA[k + 1] := GA[k + 1] + 1
        {The next GA vector is available here}
      end;
    {The last GA vector is available here}
  end;
end;
  
```

Figure 5. Generation of GAs (for $y_1, y_2, \dots, y_{ng} > s$)

Table 3. GD – Global Data Distribution Matrix

Data items	Node number					
	1	2	3	4	5	6
1	1	1	0	0	0	0
2	1	1	0	0	0	0
3	0	1	0	0	0	0
4	0	0	1	1	1	0
5	0	0	1	1	1	0
6	0	0	1	1	1	0
7	1	0	0	1	1	1
8	0	0	0	0	1	1
9	1	0	0	0	1	1
10	1	0	0	0	0	1
11	1	0	0	0	0	1
12	0	1	0	0	0	0

that satisfy Equations (1) and (2). Suppose we choose $J = \langle 2, 0, 1, 0, 1, 0 \rangle$ where data items 2 and 9 are read from node 1, data item 5 from node 3, and the data item 8 from node 5. From Equations (1) and (5), we see that $A = \{1, 3, 5\}$. Hence, the number of nodes accessed, or the cardinality of A ($|A|$), is 3.

One possible grouping of the 12 data items and the corresponding node sets is shown in Table 4. The table of generation and evaluation of the n -tuples GA for query transactions is summarized in Table 5. This table illustrates the generation and evaluation of GAs for query transactions received at node 1. Using this table and Equations (31) and (32), we can compute the average number of nodes accessed, \bar{n}_s , and the average number of data items accessed per node, \bar{d}_s . Similar results may be derived for queries received at other nodes in the database by modifying the procedure for determining the vector J from GA and computing similar statistics. Table 6 summarizes these statistics for requests received at the six nodes in the database.

The computation of these average statistics for update transactions as well as other concurrency control schemes, such as the majority consensus scheme, is very similar to the above procedure.

7. CONCLUSION

We have introduced a probabilistic method to compute the average number of nodes and the average number of data items accessed per node by a transaction in a partially replicated distributed database system. With constraints on data distribution and copy selection

Table 4. Grouping of Data Items for GD in Table 3

k	G_k	Q_k
1	$\langle 3, 12 \rangle$	$\langle 2 \rangle$
2	$\langle 1, 2 \rangle$	$\langle 1, 2 \rangle$
3	$\langle 10, 11 \rangle$	$\langle 1, 6 \rangle$
4	$\langle 8 \rangle$	$\langle 5, 6 \rangle$
5	$\langle 4, 5, 6 \rangle$	$\langle 3, 4, 5 \rangle$
6	$\langle 9 \rangle$	$\langle 1, 5, 6 \rangle$
7	$\langle 7 \rangle$	$\langle 1, 4, 5, 6 \rangle$

Table 5. Evaluation of GA for Query Transactions (General Model)

GA	Pr _{GA}	J	A	n _{GA}	\bar{d}_{GA}
$\langle 2, 2, 0, 0, 0, 0, 0 \rangle$	0.0020	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 1, 1, 0, 0, 0, 0 \rangle$	0.0081	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 1, 0, 1, 0, 0, 0 \rangle$	0.0040	$\langle 1, 0, 2, 0, 5, 0 \rangle$	{1, 3, 5}	3	1.33
$\langle 2, 1, 0, 0, 1, 0, 0 \rangle$	0.0121	$\langle 1, 0, 3, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 1, 0, 0, 0, 1, 0 \rangle$	0.0040	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 1, 0, 0, 0, 0, 1 \rangle$	0.0040	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 2, 0, 0, 0, 0 \rangle$	0.0020	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 1, 1, 0, 0, 0 \rangle$	0.0040	$\langle 1, 0, 2, 0, 1, 0 \rangle$	{1, 3, 5}	3	1.33
$\langle 2, 0, 1, 0, 1, 0, 0 \rangle$	0.0121	$\langle 1, 0, 3, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 1, 0, 0, 1, 0 \rangle$	0.0040	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 1, 0, 0, 0, 1 \rangle$	0.0040	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 0, 1, 1, 0, 0 \rangle$	0.0061	$\langle 0, 0, 3, 0, 1, 0 \rangle$	{3, 5}	2	2.0
$\langle 2, 0, 0, 1, 0, 1, 0 \rangle$	0.0020	$\langle 1, 0, 2, 0, 5, 0 \rangle$	{1, 3, 5}	3	1.33
$\langle 2, 0, 0, 1, 0, 0, 1 \rangle$	0.0020	$\langle 1, 0, 2, 0, 1, 0 \rangle$	{1, 3, 5}	3	1.33
$\langle 2, 0, 0, 0, 2, 0, 0 \rangle$	0.0076	$\langle 4, 0, 0, 0, 0, 0 \rangle$	{3}	1	4.0
$\langle 2, 0, 0, 0, 1, 1, 0 \rangle$	0.0152	$\langle 1, 0, 3, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 0, 0, 1, 0, 1 \rangle$	0.0076	$\langle 1, 0, 3, 0, 0, 0 \rangle$	{1, 3}	2	2.0
$\langle 2, 0, 0, 0, 0, 1, 1 \rangle$	0.0025	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0
⋮	⋮	⋮	⋮	⋮	⋮
$\langle 0, 0, 0, 0, 2, 1, 1 \rangle$	0.0455	$\langle 2, 0, 2, 0, 0, 0 \rangle$	{1, 3}	2	2.0

policy, we have derived closed-form expressions to compute these performance measures. When these constraints are relaxed and a general distribution is assumed, we developed an efficient algorithm for computing the desired measures. The general structure of our algorithm is summarized in Figure 4, with the high-level computational structure depicted in Figure 2.

Our results can be used to provide

- an approximation to a transaction’s execution time,
- an approximation to a transaction’s reliability,
- the probability that a transaction will require data items from the coordinating node, and
- the average number of data items accessed (locally) by the coordinating node.

These measures are extremely useful in the reliability and performance studies of replicated distributed database systems^{14,15}. The availability of the suggested approximation tools should ease the task of system designers and system analysts.

Acknowledgement

The authors thank Prof. Douglas Jones and Prof. Chaganty N. Rao for their critical comments on this

paper. The authors would also like to thank the anonymous referee for his comments which improved the readability of this paper.

REFERENCES

1. D. A. Bell, F. J. McErlean, P. M. Stewart and W. A. buckle, Clustering related tuples in databases. *The Computer Journal* **31** (3), 253–257 (1988).

2. B. K. Bhargava and L. Lilien, A review of concurrency and reliability issues in distributed database systems, in *Concurrency Control and Reliability Issues in Distributed Systems*, Edited by B. K. Bhargava, Van Nostrand Reinhold Co, pp. 1–84 (1987).

3. S. Christodoulakis, Estimating block selectivities. *Information Systems* **9** (1), 105–115 (1984).

4. S. Christodoulakis, Implications of certain assumptions in database performance evaluation. *ACM Trans. on Database Systems* **9** (2), 163–186 (1984).

5. S. B. Davidson, Analyzing partition failure protocols. *Technical Report*, MS-CIS-86-05, Dept. of Computer Science, University of Pennsylvania (1986).

6. L. W. Dowdy and D. V. Foster, Comparative models of the file assignment problem. *ACM Computing Surveys* **14** (2), 287–313 (1982).

7. M. T. Fang, R. C. T. Lee and C. C. Chang, The idea of de-clustering and its applications. *Proc. Twelfth Intl. Conf. on Very Large Data Bases*, Kyoto, Japan, pp. 181–188 (1986).

8. H. Garcia-Molina, Performance evaluation of the update algorithms for replicated data in a distributed database. *Ph.D. Dissertation*, Computer Science Department, Stanford University (1979).

9. B. Gavish and H. Pirkul, Computer and database location in distributed computer systems. *IEEE Trans. on Computers* **C-35** (7), 583–590 (1986).

10. D. K. Gifford, Weighted voting for replicated data. *Proc. 7th ACM Symposium on Operating System Principles*, pp. 150–162 (1979).

11. J. A. Hoffer and D. G. Severance, The use of cluster

- analysis in physical database design. *Proc. of 1st Very Large Databases Conference*, Boston, pp. 69–86 (1975).
12. L. F. Mackert and G. M. Lohman, R* Optimizer validation and performance evaluation for distributed queries. *Proc. Twelfth Intl. Conf. on Very Large Data Bases*, Kyoto, Japan, pp. 149–159 (1986).
 13. R. Mukkamala, S. C. Bruell and R. K. Shultz, Design of partially replicated distributed database systems: an integrated approach. *Proc. ACM SIGMETRICS Conference of Modeling and Measurement*, pp. 187–196 (1988).
 14. C. S. Raghavendra, V. K. Prasanna Kumar and S. Hariri, Reliability analysis in distributed systems. *IEEE Trans. on Computers* 37 (3), 352–358 (1988).
 15. M. Singhal and A. K. Agarwala, Performance analysis of an algorithm for concurrency control in replicated database systems. *Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 159–169 (1986).
 16. L. E. Stanfel, Applications of clustering to information system design. *Information Processing and Management* 19 (1), 37–50 (1983).
 17. R. B. Thomas, A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. on Database Systems* 4 (2), 180–209 (1979).
 18. Y. L. Varol and S. V. Vrbsky, Distributed query processing for redundant data. *Proc. IEEE Conference on Distributed Computing Systems*, pp. 389–396 (1984).

Announcements

9–11 MAY 1990

NIVELLES, BELGIUM

Université Libre de Bruxelles

International Conference Computer, Man and Organization II

The Section Informatique et Sciences Humaines of the Université Libre de Bruxelles (Faculty of Social, Political and Economic Sciences) will be celebrating the tenth anniversary of its creation during the 1989–1990 academic year.

With this in view it has decided to organize a conference devoted to the transformations that the increasing use of information technology involves for people and organizations.

The conference aims at taking stock of the situation and sketching out the evolution prospect regarding the following questions: how does man define himself with regard to information technology? What are the main problems involved in the man-machine face-to-face encounter? What are the potential and real impacts of information technology on the organization and conversely? In what respects and how does information technology affect man's place in the organization?

Papers will focus on the following themes:

- man-machine communication;
- computer-assisted training;
- psycho-social representations of information technology;
- implementation strategies in big, small and medium-sized organizations;
- place of human sciences in the training of data processing specialists;
- actual and potential contributions of micro-processing, telematics and telecommunications in the field of management;
- problems arising from information technology for men and organizations;
- executives facing information technology;
- design, implementation and management policies regarding applications;
- myths and realities of the expert systems;
- information technology and public services;
- information technology assessment methods and practices;
- socio-economic analyses of information technology.

Location

The conference will take place at the centre cultural "Waux-Hall", Nivelles – Belgium. Nivelles is located some 25 km south of Brussels and is easily accessible through the motorway or by train. Access maps will be sent to the participants.

A special free (permanent shuttle) coach service will be available between Nivelles, where the conference takes place, and the centre of Brussels where it will serve the main hotels and communication centres.

Languages

The official languages of the conference are **French and English**.

Accommodation

Accommodation facilities will be communicated in due time. Rooms at special reduced rates will be booked in hotels located in the centre of Brussels.

Registration fee

The registration fee, which covers all working documents, coffees and refreshments, participation in a reception as well as the shuttle coach between Brussels and Nivelles, is 12,000 BEF.

We draw your attention to the fact that on 14, 15 and 16 May 1990, the 6th International French Speaking Congress on the Psychology of Work (*6ème Congrès International de Psychologie du Travail de Langue Française*) will be held in Nivelles as well.

People who take part in both the Conference and the Congress qualify for special reduced fees (see registration and call for papers form).

The amount of the registration fee for the Conference and/or the Congress is to be paid in Belgian currency into the account n° 001-1537599-31 of the L.I.S.H./U.L.B.

Please do not forget to mention the participant's name and the purpose of the payment: 'Conference' and/or 'Congress'.

It can also be paid by bank cheque. The account charges are payable by the drawer.

All correspondence or inquiries regarding the conference can be addressed to:

either

SISH-U.L.B., Colloque 'l'Ordinateur, l'Homme et l'Organisation II', Rue des Canoniers, 2, B-1400 Nivelles. Tel: +32.67.21.85.29 (from abroad) or (067) 21.85.29 (from Belgium)

or

Prof. L. Wilkin, Université Libre de Bruxelles/CP 140, 50, Av. F. Roosevelt. B-1050 Bruxelles. Tel: +32.2.642.41.24 (from abroad) or (02) 642.41.24 (from Belgium). Fax: +32.2.642.35.95 (from abroad) or (02) 642.35.95 (from Belgium).

23–25 MAY 1990

ESPOO (HELSINKI), FINLAND

Sixth International Conference and Exhibition on Information Security

Computer Security and Information Integrity in our Changing World

State of the art presentations of Information Security and EDP Auditing. Exhibition of Information Security software and hardware products.

For information contact:

Conference Secretariat: Congrex (Finland), Neitsytpolku 12 A, P.O. Box 151, SF-00141 Helsinki, Finland. Tel: +358-0-175355; Fax: +358-0-170122; Telex: 12358 cong sf.

Registration fees:

Before 1 April 1990 4000 FIM/900 USD

After 1 April 1990 4500 FIM/1000 USD

Speakers and students 2200 FIM/500 USD

Hotel accommodation available.

Interesting guided tours and excursions.

4–7 JUNE 1990

BRIGHTON

International Conference, Computer Technology in Welding

The Welding Institute is announcing preliminary details of a forthcoming International Conference:

Third International Conference on Computer Technology in Welding

Topics for discussions will include:

- Process and equipment control
- Production control and automation
- CAD, CAM and CIM software
- Sensors and instrumentation
- Modelling
- Information technology
- Artificial intelligence

A small exhibition is planned to run in parallel with the conference, and it will be staffed by technical personnel. Delegates will be able to see the latest developments in computing equipment and software in the area of welding and fabrication technology.

The Welding Institute invites offers of papers for the conference, and further information is available from Tony Gray, Conference Organiser, The Welding Institute, Abington Hall, Abington, Cambridge CB1 6AL, U.K. Tel: 0223 891162. Telex: 81183.