경영정보학연구
제7권 3호
1997년 12월

A Simulated Distributed Database System for Response Time Evaluation*

노 상 규**

응답시간평가를 위한 분산데이터베이스 시뮬레이션시스템

Although numerous models and solution algorithms to design efficient distributed databases have been developed, very few have been validated for their effectiveness. In this paper, we develop a simulation system which can be used to analyze and validate the average response time of distributed database designs. Our simulation system models comprehensive query processing strategies such as semijoin as well as a concurrency control mechanism. We analyze and validate an average response time distributed database design model using our simulation system.

* This research was partially supported by S.N.U. Research Fund, Seoul National University, Korea.
** College of Business Administration Seoul National University

이 논문은 1996년 7월 10일 접수하여 1차 수정을 거쳐 1997년 11월 28일 게재확정되었습니다.

Ⅰ. Introduction

Distributed database systems are becoming more common in geographically distributed organizations [Richter, 1994, The, 1994]. Properly designed, distributed databases can yield significant performance and cost advantages over centralized systems. However, the design of a distributed database is an extremely complex process. A distributed database design must allocate data to nodes (possibly with redundancy) so that retrieval and update operations can be efficiently carried out at run time.

Inappropriate placement of data or poor choices of data access or processing strategies can result in poor system performance [Edelstein, 1995a, 1995b, Ozsu and Valduriez, 1991].

Although numerous distributed database design models and solution algorithms have been developed to support the design process (e.g., Apers [1988], Blankinship et al. [1991], Cornell and Yu [1989], March and Rho [1995], Ram and Narasimhan [1994], Rho [1995]), relatively few response time models have been developed (e.g., Cornell and Yu [1989], Lee and Sheng [1992], Rho [1995]).

Furthermore, very few researchers have validated the performance (i.e., response time) of the solutions (i.e., distributed database designs) obtained using their models and algorithms. Ideally, a commercial distributed database system should be used to validate the performance of distributed database designs. However, experiments with a real system would be too costly and/or too difficult to control. Thus, simulation is the only viable alternative to validate the performance of distributed database design. In this paper, we develop a simulation system for distributed database design and validate the response time model of Rho [1995] using simulation.

The remainder of the paper is organized as follows. The next section discusses prior research on the performance studies of distributed database systems. The following section describes our simulated distributed database system. The following section describes a simulation study and compares the simulation results with the analytical results. The final section discusses the limitations of the current study and presents directions for future research.

Ⅱ. Prior Research

There have been many performance modeling studies of distributed database systems and database systems in general.

Agrawal et al. [1987] developed a simulation system for a centralized database system to study the performance of different concurrency control mechanisms. Therefore their simulation system does not model the network components of distributed database systems. Furthermore, it does not model comprehensive query processing strategies.

Jenq et al. [1988] developed a queueing network model to analyze the performance of a distributed database testbed system. The model includes a concurrency control mechanism and a transaction recovery mechanism. Ciciani et al. [1990] developed an analytical model to analyze the effects of replicating data in distributed database

systems under different concurrency control mechanisms. However, both models ignore a comprehensive query processing model.

Carey and Livny [1988] and Thanos et al. [1988] analyzed the effect of concurrency control mechanisms and that of replication using simulation.

Huang et al. [1994] developed a simulation system to analyze the performance of concurrency control mechanisms in federated database systems. These simulation systems do not model comprehensive query processing strategies, either.

Although there have been many performance studies of distributed database systems, most prior work investigates either the performance of concurrency control mechanisms or the effects of replication. These performance models do not include comprehensive query processing strategies such as join site selection and semijoins. Since our main objective is to understand the performance of data allocation and query processing strategies, the previous simulation systems are not appropriate for the purposes of our study. Therefore, we develop a simulation system that models a comprehensive query processing model which includes join site selection and semijoins.

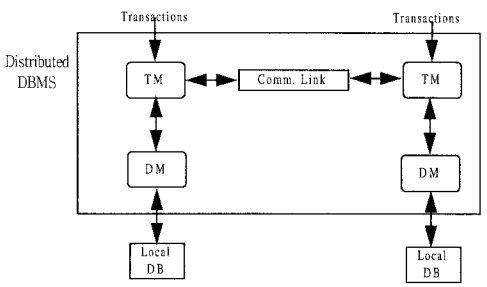
Ⅲ. A Simulated Distributed Database System

We first describe the distributed database system architecture on which our simulation system is based. The following subsection describes query processing (or transaction) models of the system. The physical queueing model underlying the system is described next. Finally, the implementation of the simulation system is described. In the following discussion the terms query and transaction are used synonymously.

3.1 A Distributed Database System Model

A simplified distributed database system architecture (adapted from Jenq et al. [1988]) is shown in <Figure 1>. It consists of two levels of server processes: Transaction Manager (TM) and Data Manager (DM). TMs and DMs work cooperatively to service transactions submitted to the system.

A user application process submits a transaction to the system. A transaction (query) consists of multiple subtransactions (query steps), each of which must be sent to and executed at a local database. A TM acts as a transaction coordinator. It sends each subtransaction to an appropriate DM and synchronizes the execution of the transaction. A request to a remote DM is routed through a remote TM. A DM manages the execution of subtransactions at a local database.



<Figure 1> A Distributed Database System Architecture

3.2 Transaction Models

There are two types of transactions: retrieval (read only) and update. A retrieval transaction consists of a set of query steps or

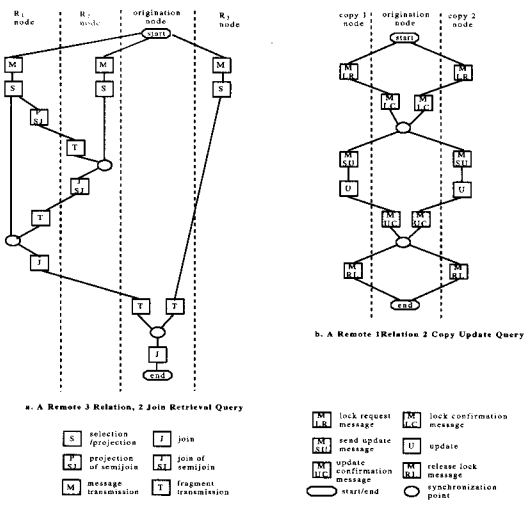
operations (e.g., message, select/project, join, semijoin), some of which can be processed in parallel and some of which must be processed sequentially (See Yu and Chang [1984] and Rho and March [1997] for detailed description of distributed query processing).

3.2.1 Retrieval Transaction Model

We define 6 operation types for retrieval queries: selection/projection, join, projection of semijoin, join of semijoin, message, and fragment transmission. Consider a three-relation, two-join SQL query as follow:

```
SELECT <attribute list>
FROM R1, R2, R3
WHERE R1.A0 = R2.A0 AND R2.A1 = R3.A1
```

<Figure 2> a shows an example execution schedule for the above query where each relation is located at a remote node from the query origination node. A rectangle represents an operation, lines represent control or data flow, and circles represent a synchronization point. At a synchronization point, all the previous operations must be finished before the next operation(s) can begin.



a. A Remote 3 Relation, 2 Join Retrieval Query

b. A Remote 1Relation 2 Copy Update Query

<Figure 2> Transaction Models

The example query execution schedule begins by sending a message from the query origination node to each node containing a relation. Upon receiving the message the appropriate selection/ projection operations are performed on each relation. The join attribute is projected from relation R1 and transmitted to the relation R2 node where a semijoin is performed (reducing relation R2 by relation R1).

The reduced relation is transmitted to the relation R1 node where it is joined with relation R1. The result is transmitted to the query origination node. Relation R3 is also transmitted to the query origination node where it is joined with the prior join result. Note that join operations cannot begin until all needed data is at the join node. For example, the last join operation cannot begin until both the result of the previous join operation and relation R3 are transmitted to the query origination node.

3.2.2 Update Transaction Model

We define 6 operation types for update transactions: lock request message, lock confirmation message, send update message, update, update confirmation message, and release lock message. <Figure 2> b shows the execution schedule for a remote update transaction based on distributed two phase locking [Bernstein and Goodman. 1981] where two copies of the affected relation are allocated.

The example transaction execution schedule begins by sending a lock request message from the query origination node to each node containing a copy of the relation.

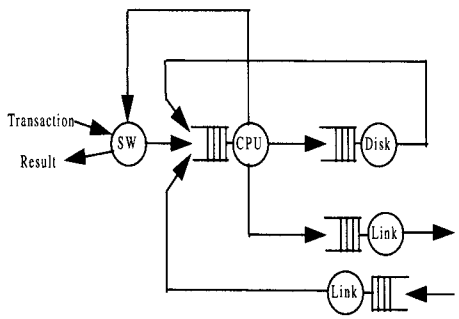
Upon receiving the message, each copy is

locked and a lock confirmation message is sent to the query origination node. When both lock confirmation messages arrive at the query origination node, a send update message is sent to each node containing a copy of the relation and update is performed there. Once update is finished, an update confirmation message is sent to the query origination node. Finally, a release lock message is sent and the lock on the relation is released.

3.3 Physical Queueing Model

<Figure 3> shows the physical queueing model of a node and its links to another node underlying the simulation system.

We assume a node consists of a single CPU and a single disk (See Agrawal et al. [1987] for detail) and is connected to another node via two communication links (one for each direction). A synchronization wait center (SW) is introduced to model the synchronization requirements of the transaction models described in the previous subsection.



<Figure 3> Physical Queueing Model of a Node

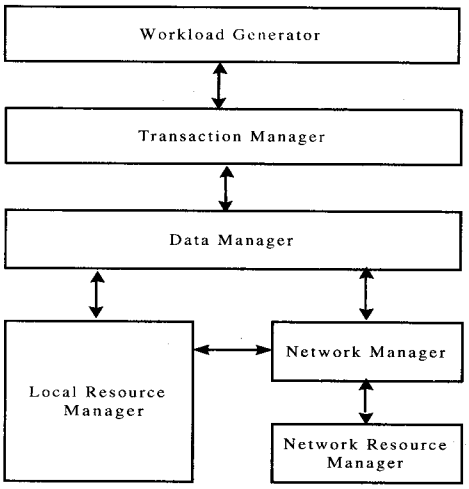
A transaction of a particular type arrives at a particular node based on its relative frequency. We assume a Poisson transaction arrival process (i.e., with negative exponential transaction interarrival times). Once a transaction (a set of operations with synchronization requirements) arrives, it enters the synchronization wait center (SW).

Some of its operations can be immediately put into queues. However, others must wait until other operations (i.e., their previous subtransactions) are completed before they can be put into queues. An operation receives services from CPUs, disk, and/or links depending on its type. Each operation has CPU, disk I/O, and/or transmission requirements associated with it. They are calculated in the same way as presented in Rho [1995]. The requests in CPU and disk queues are serviced using a round robin scheme with a time slice discipline. Those in link queues are serviced using a first-come-first-served (FCFS) discipline. The current model ignores queueing delays due to data contention (e.g., waiting for lock release). This wil be addressed in future research.

3.4 A Simulation System

The simulation system was developed using CSIM [Schwetman, 1986, 1988, 1991] based on the model described above. CSIM is a process-oriented discrete-event simulation software for use with C or C++ programs. The basic entity in CSIM is a process, which is used to model active components of a system (e.g., transactions in distributed database system). A process uses one or more facilities (e.g., disks) which are used to model passive components of a system. Events are used to implement synchronization requirements among processes (e.g., a join operation

cannot begin until both select operations are completed). CSIM was chosen because it is flexible enough to model complex dependencies among operations in the transaction models. <Figure 4> shows the architecture of the simulation system. The architecture was designed to support modularity so that changes made in one component will not affect others. For example, different types of networks can be modeled by changing only Network Resource Manger. Each component is briefly described below.



<Figure 4> Simulation System Architecture

Workload Generator: simulates the arrival of transactions based on their relative frequencies. In the current implementation, the Poisson arrival process is assumed. Once a query arrives, it launches one of the processes in the Transaction Manager corresponding to the query type. For example, when a three-relation two-join query arrives, a two_join_query process is launched.

Transaction Manager: models the execution of transactions and ensures their synchronization based on the query processing schedules obtained by the optimization algorithm. Distributed two phase commit is employed as a concurrency control mechanism. It has the following processes:

no_join_query(query_id),
one_join_query(query_id),
two_join_query(query_id), and
update(query_id).

They implement the transaction models discussed above and uses the processes (e.g., restrict, semijoin, etc.) provided by the Data Manager.

Data Manger: models local database operations such as selection/projection and join. It provides the following services to the Transaction Manager:

restrict (query_node, restrict_node, input_size, output_size, fr, restrict_done),
semijoin (reducer_node, reducee_node, destination_node, input_size, reducer_ size, reducee_size, output_size, fr, semijoin_ done),
join (join_node, destination_node, input1_size, input2_size, output_size, fr, join_done),
get_lock (query_node,frag_node,lock_secured),
do_update (query_node,frag_node, num_block, fr, updated), and release_lock (query_ node, frag_node, lock_released).

Each service combines a set of operations in the transaction model <Figure 2> that must be processed sequentially. For example, restrict combines send message and select

operations. Each service utilizes the services provided by the Network Manager and the Local Resource Manager.

Network Manager: models network transmission operations such as message and fragment transmission. It provides the following services:

transmit_data(from_node, to_node, size, fr)
transmit_msg(from_node, to_node)

It uses the services provided by the Local Resource Manager and the Network Resource Manager.

Local Resource Manager: models CPU processing and disk I/O of a node. It provides CPU and I/O services to other components. Requests in the CPU and disk queues are serviced using a round robin scheme with a time slice discipline. Together with the Network Resource Manager, it implements the physical queueing model above.

Network Resource Manager: models the use of communication links. A fully connected point-to-point network is assumed in the current implementation.

Requests in link queues are serviced using a first-come-first-served (FCFS) discipline.

Ⅳ. A Simulation Study

In this section, we analyze and validate an average response time model [Rho, 1995] using simulation.

The average response time model developed in Rho [1995] includes queueing delays in local database operations as well as those in network communication. They assume M/M/1 queueing models and do not explicitly model the synchronization and possible parallel processing of query steps. We examine the effects of these limitations using simulation [Kobayashi, 1978, Law and Kelton, 1991, Sauer and MacNair, 1983]. Our goal is to gain a preliminary understanding of the accuracy of their analytical model.

4.1 Simulation Problems

We simulated a set of solutions for the problems developed in Rho [1995].

The Problems have 17 fragments and 54 query types (37 retrievals and 17 updates) in a fully-connected 5-node network.

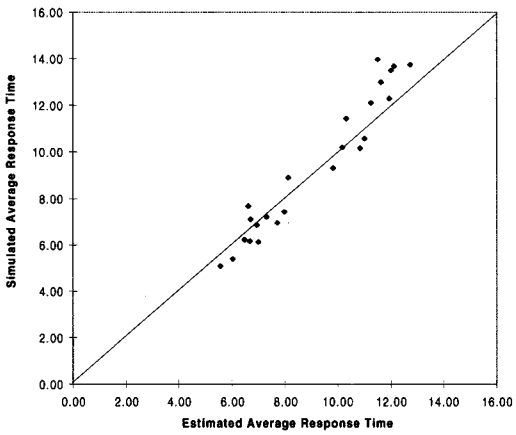
To assess the simulation system in a variety of environments, we varied the characteristic of the above problems along two dimensions: query mix and query selectivity. Query mix is the relative execution frequency of retrieval compared to update queries. Three classes of query mix are considered: Retrieval Intensive, Balanced, Update Intensive. Query selectivity is the proportion of records selected by a query.

Two environments are considered: Low Selectivity and High Selectivity.

4.2 Simulation Results

The simulation results are reported in Appendix 1 and summarized in <Figure 5>. In <Figure 5>, simulated average response times are plotted against estimated average

response time using the analytical response time model developed in Rho [1995] and summarized in Appendix 2. The analytical avrage response time model estimated the simulated average response time with reasonable accuracy (R = .9684, p = .0000). The analytical model tended to slightly underestimate the simulated time as the analytical average response time increased. This is somewhat unexpected because ignoring parallelism should result in overestimation.



<Figure 5> Analytical vs. Simulated Average Response Time

A closer examination revealed, however, that assuming an M/M/1 queueing model in estimating the average delay in the queue should result in underestimation. The average queueing delay for an M/G/1 queue increases as the variability of the service time increases even though the mean service time stays the same [Law and Kelton, 1991]. Intuitively, this is because a highly variable service time random variable will have a greater chance of taking on a large value, which means the server will be tied up for a long time, causing the queue to build up. It is likely that the variability of the actual service time distribution in the sample problem is larger than that of an exponential service time distribution with the same mean (recall that the problem has two types of transactions: retrieval, which requires a large amount of processing time, and update, which requires a very small amount of processing time). A larger variability should result in the underestimation of average response time, especially for transactions with very small processing time requirements (i.e., update transactions).

In fact, further analysis of the underestimation cases revealed that the underestimation was mostly due to the underestimation of update transaction response time. Although limited in scope, the results demonstrate that Rho's [1995] analytical response time model is reasonably accurate.

V. Summary and Future Research

In this paper, we presented a simulation system for distributed database design. This is the first simulation system that models complex query processing strategies of distributed database systems. We analyzed and validated Rho's [1995] average response time model using simulation. The simulation results demonstrate that Rho's average response time model is reasonably accurate. However, the results must be interpreted with caution since simulated response time is another estimate of the true system response

time.

There are several directions for future research.

First, the simulation system will be enhanced to model more general cases of distributed database systems. Different networks (e.g., bus topology, general topology) and computer systems (e.g., multi-disk systems) can be modeled. The simulation system can be enhanced to include data contention as well as resource contention.

Secondly, the simulation system will be more tightly integrated into optimization algorithms, resulting in a decision support system for distributed database design.

Finally, more rigorous validation of response time models including Rho's [1995] will be performed. Response time models will be validated using a wide variety of problems. Such validation will result in more realistic distributed database design models.

⟨REFERENCES⟩

Agrawal, R., Carey, M. J. and Livny, M., "Concurrency Control Modeling: Alternatives and Implications," *ACM Transactions on Database Systems*, Vol.12,No.4, December 1987, pp. 609-654.

Apers, P. M. G., "Data Allocation in Distributed Database Systems," *ACM Transactions on Database Systems*, Vol.13, No.3, September 1988, pp. 263-304.

Bernstein, P. A. and Goodman, N., "Concurrency Control in Distributed Database Systems," *ACM Computing Surveys*, Vol.13, No.2, June 1981, pp. 185-222.

Blankinship, R., Hevner, A. R. and Yao, S. B., "An Iterative Method for Distributed Database Design," *Proceedings of the 17th International Conference on Very Large Data Bases*, Barcelona, Spain, September 1991, pp. 389-400.

Carey, M. J. and Livny, M., "Distributed Concurrency Control Performance: A study of Algorithms, Distribution, and Replication," *Proceedings of 14th International Conference on Very Large Data Bases*, Los Angeles, CA, August 1988, pp. 13-25.

Ciciani, B., Dias, D. M. and Yu, P. S., "Analysis of Replication in Distributed Database Systems," *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, No.2, June 1990, pp. 247-261.

Cornell, D. W. and Yu, P. S., "On Optimal Site Assignment for Relations in the Distributed Database Environment," *IEEE Transactions on Software Engineering*, Vol.15, No.8, August 1989, pp. 1004-1009.

Edelstein, H., "The Challenge of Replication," *DBMS*, March 1995a, pp. 46-49, 52.

Edelstein, H., "The Challenge of Replication, Part 2," *DBMS*, April 1995b, pp. 62-70, 103.

Huang, J., Hwang, S.-Y. and Srivastava, J., *Concurrency Control in Federated Database Systems: A Performance Study*, Technical Report TR93-15, Department of Computer Science, University of Minnesota, February 1993.

Jenq, B. C., Kohler, W. H. and Towsley, D., "A Queueing Network Model for a Distributed Database Testbed System," *IEEE Transactions on Software Engineering*, Vol.14, No.7, July 1988, pp. 908-921.

Kobayashi, H., *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*, Addison Wesly Publishing, 1978.

Law, A. M. and Kelton, W. D., *Simulation Modeling and Analysis*, McGraw-Hill, 1991.

Lee, H. and Sheng, O. R. L., "A Multiple Criteria Model for the llocation of Data Files in a Distributed Information Systems," *Computers and Operations Research*, Vol.21, 1992, pp. 21-33.

March, S. T. and Rho, S., "Allocating Data and Operations to Nodes in Distributed Database Design," *IEEE Transactions on Knowledge and Data Engineering*, Vol.7, No.2, April 1995, pp. 305-317.

Ozsu, M. and Valduriez, P., *Principles of Distributed Database Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991.

Ram, S. and Narasimhan, S., "Database Allocation in a Distributed Environment: Incorporating a Concurrency Control Mechanism and Queuing Costs," *Management Science*, Vol.40, No.8, August 1994, pp. 969-983.

Rho, S., *Distributed Database Design: Allocation of Data and Operations to Nodes in Distributed Database Systems*, Unpublished Ph.D. thesis, University of Minnesota, May 1995.

Rho, S. and March, S. T., "Designing Distributed Database Systems for Efficient Operation," *Proceedings of the 16th International Conference on Information Systems*, December 1995. pp. 237-253

Rho, S. and March, S. T., "Optimizing Distributed Join Queries: A Genetic Algorithm Approach," *Annals of Operations Research*, Vol.71, 1997, pp. 199-228.

Richter, J., "Distributing Data," *Byte*, June 1994, pp. 139-148.

Sauer, C. H. and MacNair, E. A., *Simulation of Computer Communication Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Schwetman, H., "CSIM: A C-Based, Process-Oriented Simulation Language," *Proceedings of the 1986 Winter Simulation Conference*, 1986, pp. 387-396.

Schwetman, H., "Using CSIM to Model Complex Systems," *Proceedings of the 1988 Winter Simulation Conference*, 1988, pp. 246-253.

Thanos, C., Bertino, C. and Carlesi, C., "The

Effects of Two-Phase Locking on the Performance of a Distributed Database Management System," *Performance Evaluation*, Vol.8, 1988, pp. 129-157.

The, L., "Distribute Data Without Choking the Net," *Datamation*, Vol.40, January 7, 1994, pp. 35-36.

Yu, C. T. and Chang, C. C., "Distributed Query Processing," *ACM Computing Surveys*, Vol.16, No.4, December 1984, pp. 399-433.

Appendix 1. Simulation Results

Solution	Average Response Time	
	Analytic	Simulation
1	8.118	8.900
2	7.987	7.426
3	7.726	6.934
4	6.605	7.652
5	6.708	7.083
6	6.024	5.390
7	7.317	7.191
8	6.948	6.834
9	6.983	6.126
10	6.482	6.216
11	6.681	6.138
12	5.581	5.054
13	12.114	13.677
14	12.725	13.742
15	11.956	12.290
16	12.013	13.511
17	11.643	12.977
18	11.501	13.975
19	10.850	10.160
20	10.321	11.446
21	11.028	10.580
22	10.168	10.194
23	11.259	12.092
24	9.812	9.318

Appendix 2. Average Response Time Model [Rho, 1995]

Min RT =

$$\frac{\sum_k f(k)\,(R_{COM}(k) + R_{IO}(k) + R_{CPU}(k))}{\sum_k f(k)}$$

where RCOM(k), RIO(k), and RCPU(k) are the times spent by query k in communication, disk I/O, and CPU, respectively. These response time compo- nents are summarized below.

RCOM(k) =

$$\sum_t \sum_p \sum_m \left(\frac{W(t,p)TL(t,p)N(k,m,t,p)}{(UL(t,p))^2 - UL(t,p)TL(t,p)} + \frac{H(k,m,t,p)}{UL(t,p)} \right)$$

where UL(t,p) is the capacity of the communication link from node t to node p (bytes per unit time), $TL(t,p) = \sum_k f(k) \sum_m h(k,m,t,p)$, $W(t,p) = \frac{TL(t,p)}{\sum_k f(k) \sum_m N(k,m,t,p)}$, and N(k,m,t,p) is 1 if H(k,m,t,p) > 0 and it is 0 otherwise. H(k,m,t,p) is defined as follows: W(t,p) =

For message steps of retrievals,
H(k,m,t,p) = LM
if t = node(k) and p = node(a(k,m))
H(k,m,t,p) = 0 otherwise

where LM is the size of a message, node(k) is the origination node of query k, node(i) is the node at which file fragment i is located.

For join steps,
H(k,m,t,p) = La(k,m) + Lb(k,m)
if t = node(a(k,m)) = node(b(k,m)) and p = node(k,m)
H(k,m,t,p) = La(k,m)
if t = node(a(k,m)) and t ≠ node (b(k,m)) and p = node(k, m)
H(k,m,t,p) = Lb(k,m)
if t ≠ node(a(k,m)) and t = node (b(k,m)) and p = node(k, m)
H(k,m,t,p) = 0
otherwise.

where Li is the size of file fragment i (in characters), a(k,m) and b(k,m) are the file fragment referenced by step m of query k, and node(k,m) is the node at which step m of query k is processed.

For a final step,
H(k,m,t,p) = La(k,m)
if t = node(a(k,m)) and p = node(k)
H(k,m,t,p) = 0
otherwise.

For send-message steps of updates,
H(k,m,t,p) = LM
if t = node(k) and copy(a(k,m), p) = 1
H(k,m,t,p) = 0
otherwise

where copy(i,t) is 1 if fragment I is stored at node t, and 0 otherwise.

For receive-message steps of updates,
H(k,m,t,p) = LM
if copy(a(k,m), t) = 1 and p = node(k)
H(k,m,t,p) = 0
otherwise.

RIO(k) =

$$\sum_t \sum_m O(k,m,t) \frac{1}{UIO(t) - TIO(t)}$$

where UIO(t) is the disk I/O capacity at node t (number of disk I/O' s per unit time) and TIO(t) = $\sum_k F(k) \sum_m O(k,m,t)$ is the total number of disk I/O' s at node t. O(k,m,t) is defined as follows:

For selection and projection steps,
O(k,m,t) = Dkmt
if t = node(a(k,m))
O(k,m,t) = 0
otherwise

where Dkmt is the number of disk I/Os required to process step m of query k at node t.

For join steps,
O(k,m,t) = Fa(k,m)t
if t ≠ node(k, m) and t = node(a(k,m)) and t ≠ node(b(k,m))
O(k,m,t) = Fb(k,m)t
if t ≠ node(k, m) and t ≠ node(a(k,m)) and t = node(b(k,m))
O(k,m,t) = Fa(k,m)t + Fb(k,m)t
if t ≠ node(k, m) and t = node(a(k,m)) and t = node(b(k,m))
O(k,m,t) = Dkmt
if t = node(k,m) = node(a(k,m)) = node(b(k,m))
O(k,m,t) = Dkmt + Ea(k,m)t
If t = node(k,m) = node(b(k,m)) and t node(a(k,m))
O(k,m,t) = Dkmt + Eb(k,m)t
if t = node(k, m) = node(a(k,m)) and t node(b(k,m))
O(k,m,t) = Dkmt + Ea(k,m)t + Eb(k,m)t
if t = node(k,m) and t node(a(k,m)) and t node(b(k,m))
O(k,m,t) = 0
otherwise

where F'a(k,m)t is the number of additional disk accesses needed at node t in order to send a(k,m) from node t to another node after having retrieved it and Ea(k,m)t is the number of disk access required to receive and store a(k,m) at node t (typically a file write and the creation of needed indexes).

For final steps,
O(k,m,t) = Ea(k,m)t
if t ≠ node(a(k,m)) and t = node(k)
O(k,m,t) = Fa(k,m)t
if t = node(a(k,m)) and t node(k)
O(k,m,t) = 0
otherwise.
For update requests,
O(k,m,t) = Dkmt
if copy(a(k,m), t) = 1
O(k,m,t) = 0
otherwise

RCPU(k) =

$$\sum_t \sum_m U(k,m,t) \frac{1}{UCPU(t) - TCPU(t)}$$

where UCPU(t) is the CPU capacity at node t (number of instructions per unit time) and TCPU(t) = $\sum_k F(k) \sum_m O(k,m,t)$ is the total number instructions at node t. U(k,m,t) is defined as follows:

For message steps,
U(k,m,t) = St
if t = node(k) and t node(a(k,m))
U(k,m,t) = Rt

if t node(k) and t = node(a(k,m))
U(k,m,t) = 0

where St and Rt are the expected CPU units required to send and receive a message.

For selection and projection steps,
U(k,m,t) = Wkmt
if t = node(a(k,m))
U(k,m,t) = 0
otherwise

where Wkmt is the number of CPU units required to process step m of query k at node t

For join steps,
U(k,m,t) = F' a(k,m)t
if t≠ node(k, m) and t = node(a(k,m)) and t node(b(k,m))
U(k,m,t) = F' b(k,m)t
if t ≠ node(k, m) and t node(a(k,m)) and t = node(b(k,m))
U(k,m,t) = F' a(k,m)t + F' b(k,m)t
if t ≠ node(k, m) and t = node(a(k,m)) and t = node(b(k,m))
U(k,m,t) = Wkmt
if t = node(k, m) = node(a(k,m)) = node(b(k,m))
U(k,m,t) = Wkmt + E' a(k,m)t
if t = node(k, m) = node(b(k,m)) and t node(a(k,m))
U(k,m,t) = Wkmt + E' b(k,m)t
if t = node(k, m) = node(a(k,m)) and t node(b(k,m))
U(k,m,t) = Wkmt + E' a(k,m)t + E' b(k,m)t
if t = node(k, m) and t node(a(k,m)) and t node(b(k,m))
U(k,m,t) = 0
otherwise

where F'a(k,m)t and E'a(k,m)t are the number of CPU operations required to send and receive a(k,m) from and to node t, respectively.

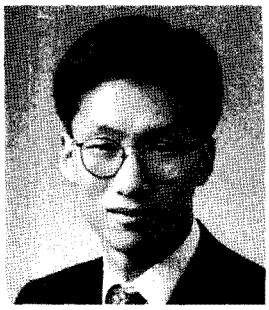
For final steps,
U(k,m,t) = E' a(k,m)t
if t node(a(k,m)) and t = node(k)
U(k,m,t) = F' a(k,m)t
if t = node(a(k,m)) and t node(k)
U(k,m,t) = 0
otherwise.

For send-message steps of updates,
$U(k,m,t) = \sum_{p \neq t} copy(a(k,m), p)\ St$
if t = node(k)
U(k,m,t) = Rt
if t node(k) and copy(a(k,m), t) = 1
U(k,m,t) = 0
otherwise

For receive-message steps of updates,
$U(k,m,t) = \sum_{p \neq t} copy(a(k,m), p)\ Rt$
if t = node(k)
U(k,m,t) = St
if t node(k) and copy(a(k,m), t) = 1
U(k,m,t) = 0
otherwise

For update steps,
U(k,m,t) = Wkmt
if copy(a(k,m), t) = 1
U(k,m,t) = 0
otherwise

◆ 저자소개 ◆



노 상 규 (Rho, Sangkyu)
서울대학교 경영학과를 졸업하고 미국 미네소타 대학에서 경영학 석사 및 박사 학위를 취득하였으며 현재 서울대학교 경영학과에 재직중이다. 주요 연구분야로는 분산 데이터베이스 시스템, 객체지향 시스템 개발, 데이터 마이닝 등이 있다.