A Simulation System for Distributed Database Design

Sagnkyu Rho

College of Business Administration
Seoul National University

Abstract

Although numerous distributed database design models and solution algorithms have been developed, very few have been validated. Validation is critical to the successful application of such models to distributed database design. In this paper, we develop a simulation system for distributed database design. We then analyze and validate an average response time model using simulation. The simulation results demonstrate that the average response time model is reasonably accurate.

1. INTRODUCTION

With the emergence of commercial distributed database management systems [Richter, 1994, The, 1994], distributed database systems are becoming more common. Properly designed, distributed databases can yield significant performance advantages over centralized systems for geographically distributed organizations. However, the design of a distributed database is an extremely complex process. Given a computer network consisting of nodes with given processing and storage capacities, connected by links with given data transmission capacities, 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].

The determination of units of data to allocate (termed file *fragments*) and the placement of copies of those units on nodes in the network is termed *data allocation*. The choice of when, where, and how retrieval and processing operations are performed is termed *operation allocation* or *distributed query optimization*. A *concurrency control mechanism* [Bernstein and Goodman, 1981] insures that update operations are performed correctly and consistently. Having an efficient and effective

concurrency control mechanism is particularly important when data is allocated redundantly. Collectively, operation allocation and concurrency control are termed *operating strategies*.

Data allocation and operation allocation are interdependent problems [Apers, 1988]. The optimal set of file fragments and their optimal allocation depend on how queries are processed (i.e., the operation allocation). However, the optimal operation allocation depends on where file fragments are located (i.e., the data allocation). Therefore, although operations are not actually allocated until run-time, an *ideal* operation allocation strategy must be developed at design time to obtain an effective distributed database design. The concurrency control mechanism is typically dependent upon the selected distributed database management system.

Although numerous distributed database design models and solution algorithms have been developed (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. Ideally, a real 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. In this paper, we develop a simulation system for distributed database design and validate the response time model of Rho [1995] using simulation.

Although there have been many simulation studies of distributed database systems and database systems in general, most prior work investigates the performance of concurrency control mechanisms (e.g., Agrawal et al. [1987], Carey and Livny [1988], Ciciani et al. [1990], Huang, Hwang, and Srivastava [1993], Thanos et al. [1988]). These simulation systems do not model comprehensive operation allocation strategies such as those in Rho [1995]. Since our main objective is to understand the performance of data and operation allocation strategies, the previous simulation systems are not appropriate for the purposes of our study. Therefore, we develop a simulation system that models the comprehensive operation allocation strategy of Rho [1995].

The average response time model developed in Rho [1995] and summarized in Appendix 1 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. In this paper, 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. The next section describes our simulated distributed database system. The following section compares the simulation results with the analytical results.

2. 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.

2.1 A Distributed Database System Model

A simplified distributed database system architecture (adapted from Jenq et al. [1988]) is shown in

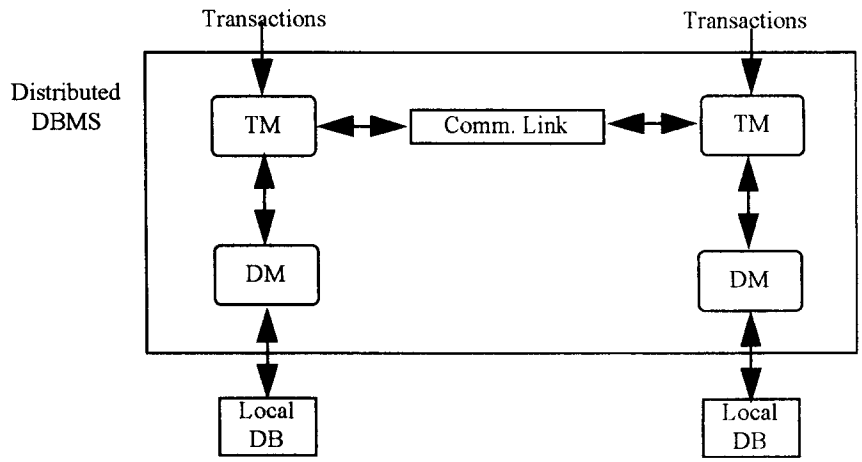


Figure 1. A Distributed Database System Architecture

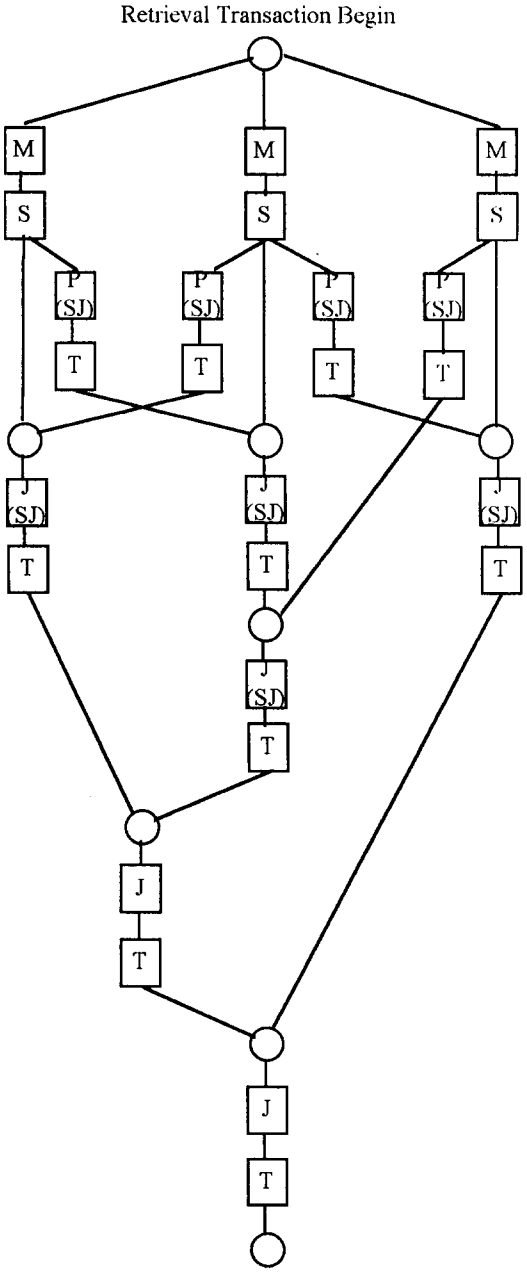
Figure 1. It consists of two levels of server processes: Transaction Manager (TM) and Data

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.

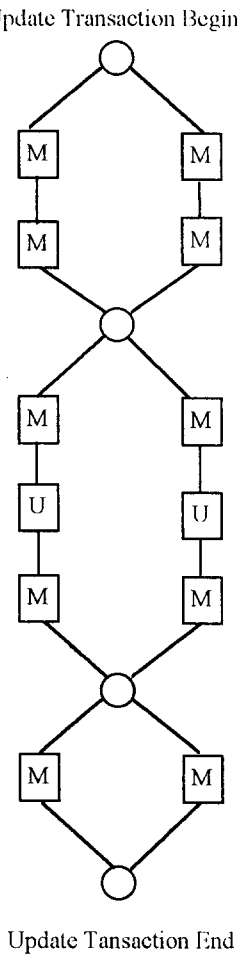
2.2 Transaction Models

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. Figure 2.a shows the execution model for a two-join query when all four semijoins are performed. A rectangle represents an operation. We define 6 operation types for retrieval transactions: selection/projection, join, projection of semijoin, join of semijoin, message, and fragment transmission [Rho, 1995]. Note that message and fragment transmission operations may not need to be performed if their previous and following operations are performed at the same node. A circle represents a synchronization point. At a synchronization point, all the previous operations must be finished before the next operation(s) can begin. For example, the last join operation in Figure 2.a cannot begin until both the result of the previous join operation and the result of the semijoin operation are transmitted to the node at which it is performed.

Figure 2.b shows the execution model for an update query based on distributed two phase locking where two copies of the affected fragment are allocated [Bernstein and Goodman. 1981]. We define 2 operation types for update queries: message and update.



a. Retrieval Transaction Model (2 Join)



b. Update Transaction Model (2 copy)

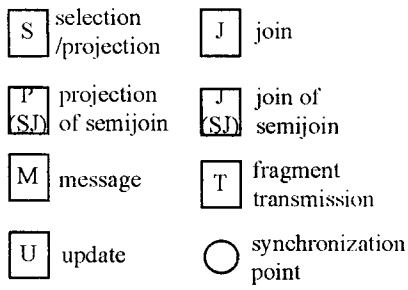


Figure 2. Transaction Models

2.3 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 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.

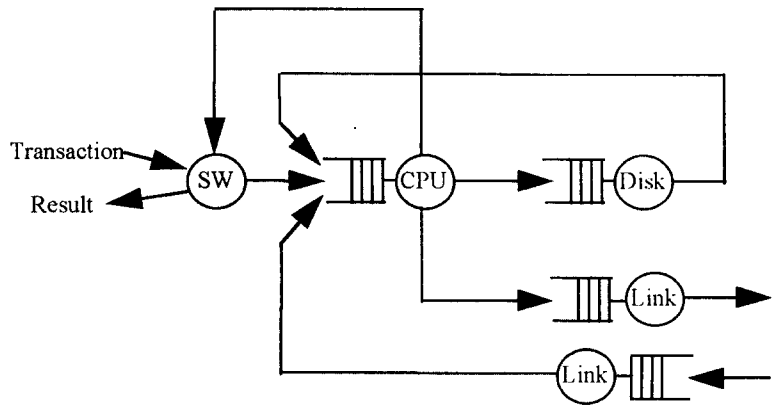


Figure 3. Physical Queueing Model

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 will be addressed in future research.

2.4 A Simulation System

The simulation system was developed using CSIM [Schwetman, 1986, 1988, 1991] based on the model described above. Figure 4 shows the architecture of the simulation system. Each component is briefly described below.

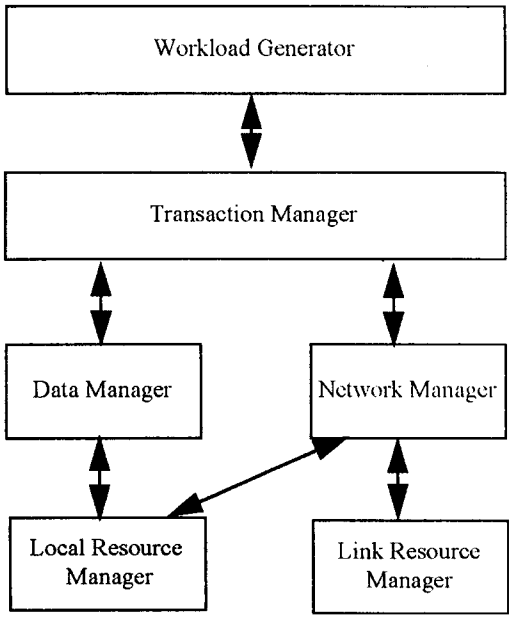


Figure 4. Simulation System Architecture

Workload Generator: simulates the arrival of transactions based on their relative frequencies.

Transaction Manager: models the execution of transactions and ensures their synchronization based on the operation allocation produced by the genetic algorithm. It uses the services provided by the Data Manager and the Network Manager.

Data Manger: models local database operations such as selection/projection and join. It uses the services provided by the Local Resource Manager.

Network Manager: models network transmission operations such as message and fragment transmission. 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.

Network Resource Manager: models the use of communication links.

3. SIMULATION RESULTS

We simulated the set of solutions for the problems developed in Rho [1995]. The simulation results are reported in Appendix 2 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 presented in Rho [1995]. The analytical average 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.

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

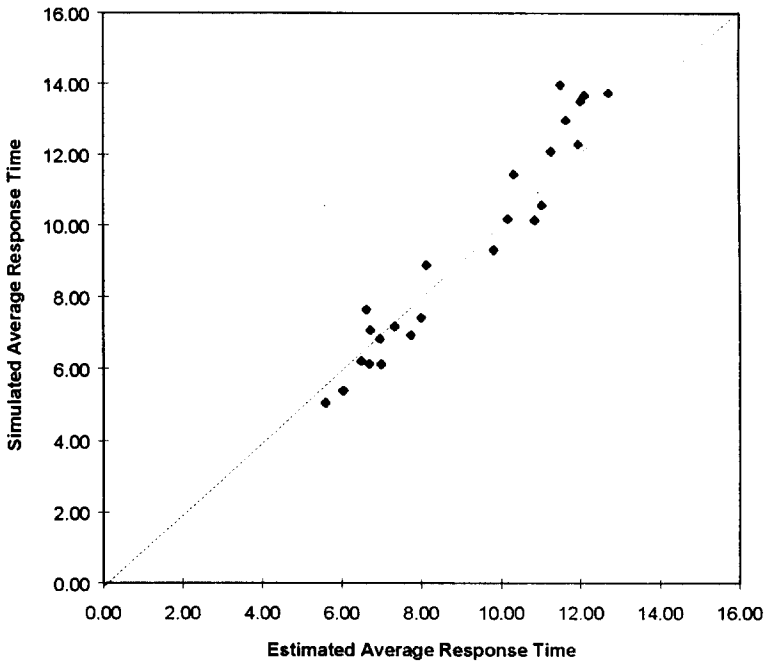


Figure 5. Analytical vs. Simulated Average Response Time

results demonstrate that Rho and March's [1995] analytical response time model is reasonably accurate.

4. SUMMARY

In this paper, we presented a simulation system for distributed database design. 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. Future work will enhance the simulation system to include data contention as well as resource contention. More rigorous validation of response time models including Rho's [1995] will be performed.

5. 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 Allocation 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.

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.

Schwetman, H., *CSIM Users' Guide*, Microelectronics and Computer Technology Corp., 1991.

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.

Appendix 1. Average Response Time Model [Rho, 1995]

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

where $R_{COM}(k)$, $R_{IO}(k)$, and $R_{CPU}(k)$ are the times spent by query k in communication, disk I/O, and CPU, respectively. These response time components are summarized below.

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

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:

For message steps of retrievals,

$H(k,m,t,p) = L^M$ if t = node(k) and p = node(a(k,m))
$H(k,m,t,p) = 0$ otherwise

where L^M 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,

$H(k,m,t,p) = L_{a(k,m)} + L_{b(k,m)}$ if t = node(a(k,m)) = node(b(k,m)) and p = node(k,m)
$H(k,m,t,p) = L_{a(k,m)}$ if t = node(a(k,m)) and t ≠ node(b(k,m)) and p = node(k, m)
$H(k,m,t,p) = L_{b(k,m)}$ 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 L_i 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,

$H(k,m,t,p) = L_{a(k,m)}$ if t = node(a(k,m)) and p = node(k)
$H(k,m,t,p) = 0$ otherwise.

For send-message steps of updates,

$H(k,m,t,p) = L^M$ 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,

$H(k,m,t,p) = L^M$ if copy(a(k,m), t) = 1 and p = node(k)
$H(k,m,t,p) = 0$ otherwise.

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

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,

$O(k,m,t) = D_{kmt}$ if $t = node(a(k,m))$

$O(k,m,t) = 0$ otherwise

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

For join steps,

$O(k,m,t) = F_{a(k,m)t}$ if $t \neq node(k,m)$ and $t = node(a(k,m))$ and $t \neq node(b(k,m))$

$O(k,m,t) = F_{b(k,m)t}$ if $t \neq node(k,m)$ and $t \neq node(a(k,m))$ and $t = node(b(k,m))$

$O(k,m,t) = F_{a(k,m)t} + F_{b(k,m)t}$ if $t \neq node(k,m)$ and $t = node(a(k,m))$ and $t = node(b(k,m))$

$O(k,m,t) = D_{kmt}$ if $t = node(k,m) = node(a(k,m)) = node(b(k,m))$

$O(k,m,t) = D_{kmt} + E_{a(k,m)t}$ if $t = node(k,m) = node(b(k,m))$ and $t \neq node(a(k,m))$

$O(k,m,t) = D_{kmt} + E_{b(k,m)t}$ if $t = node(k,m) = node(a(k,m))$ and $t \neq node(b(k,m))$

$O(k,m,t) = D_{kmt} + E_{a(k,m)t} + E_{b(k,m)t}$ if $t = node(k,m)$ and $t \neq node(a(k,m))$ and $t \neq node(b(k,m))$

$O(k,m,t) = 0$ 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 $E_{a(k,m)t}$ 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,

$O(k,m,t) = E_{a(k,m)t}$ if $t \neq node(a(k,m))$ and $t = node(k)$

$O(k,m,t) = F_{a(k,m)t}$ if $t = node(a(k,m))$ and $t \neq node(k)$

$O(k,m,t) = 0$ otherwise.

For update requests,

$O(k,m,t) = D_{kmt}$ if $copy(a(k,m), t) = 1$

$O(k,m,t) = 0$ otherwise

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

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,

$U(k,m,t) = S_t$ if $t = node(k)$ and $t \neq node(a(k,m))$

$U(k,m,t) = R_t$ if $t \neq node(k)$ and $t = node(a(k,m))$

$U(k,m,t) = 0$

where S_t and R_t are the expected CPU units required to send and receive a message.

For selection and projection steps,

$U(k,m,t) = W_{kmt}$ if $t = node(a(k,m))$

$U(k,m,t) = 0$ otherwise

where W_{kmt} 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 $t \neq node(k,m)$ and $t = node(a(k,m))$ and $t \neq node(b(k,m))$

$U(k,m,t) = F'_{b(k,m)t}$ if $t \neq node(k,m)$ and $t \neq node(a(k,m))$ and $t = node(b(k,m))$

$U(k,m,t) = F'_{a(k,m)t} + F'_{b(k,m)t}$ if $t \neq node(k,m)$ and $t = node(a(k,m))$ and $t = node(b(k,m))$

$U(k,m,t) = W_{kmt}$ if $t = node(k,m) = node(a(k,m)) = node(b(k,m))$

$U(k,m,t) = W_{kmt} + E'_{a(k,m)t}$ if $t = node(k,m) = node(b(k,m))$ and $t \neq node(a(k,m))$

$U(k,m,t) = W_{kmt} + E'_{b(k,m)t}$ if $t = node(k,m) = node(a(k,m))$ and $t \neq node(b(k,m))$

$U(k,m,t) = W_{kmt} + E'_{a(k,m)t} + E'_{b(k,m)t}$ if $t = node(k,m)$ and $t \neq node(a(k,m))$ and $t \neq node(b(k,m))$

$U(k,m,t) = 0$ 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 $t \neq node(a(k.m))$ and $t = node(k)$
$U(k,m,t) = F'_{a(k,m)t}$ if $t = node(a(k.m))$ and $t \neq node(k)$
$U(k,m,t) = 0$ otherwise.

For send-message steps of updates,

$$U(k.m.t) = \sum_{p \neq t} copy(a(k,m), p)\ S_t \text{ if } t = node(k)$$

$U(k,m,t) = R_t$ if $t \neq node(k)$ and $copy(a(k.m), t) = 1$
$U(k,m,t) = 0$ otherwise

For receive-message steps of updates,

$$U(k.m.t) = \sum_{p \neq t} copy(a(k,m), p)\ R_t \text{ if } t = node(k)$$

$U(k,m,t) = S_t$ if $t \neq node(k)$ and $copy(a(k.m), t) = 1$
$U(k,m,t) = 0$ otherwise

For update steps,

$U(k,m,t) = W_{kmt}$ if $copy(a(k.m), t) = 1$
$U(k,m,t) = 0$ otherwise

Appendix 2. 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