VISUALIZING CONCURRENCY CONTROL ALGORITHMS FOR REAL-TIME DATABASE SYSTEMS

Olusegun Folorunso[1], H.O.D. Longe[2], and Adio T. Akinwale[1]

[1]Department of Computer Science, University of Agriculture, Abeokuta, Nigeria
E-mail: folorunsoo@unaab.edu.ng
[2]Department of Computer Sciences, University of Lagos, Nigeria

ABSTRACT

This paper describes an approach to visualizing concurrency control (CC) algorithms for real-time database systems (RTDBs). This approach is based on the principle of software visualization, which has been applied in related fields. The Model-View-controller (MVC) architecture is used to alleviate the black box syndrome associated with the study of algorithm behaviour for RTDBs Concurrency Controls. We propose a Visualization "exploratory" tool that assists the RTDBS designer in understanding the actual behaviour of the concurrency control algorithms of choice and also in evaluating the performance quality of the algorithm. We demonstrate the feasibility of our approach using an optimistic concurrency control model as our case study. The developed tool substantiates the earlier simulation-based performance studies by exposing spikes at some points when visualized dynamically that are not observed using usual static graphs. Eventually this tool helps solve the problem of contradictory assumptions of CC in RTDBs.

Keywords: Real-time database systems, Visualization, Concurrency control, Algorithms

1 INTRODUCTION

Real-time database applications are characterized by their time-constrained access to data and access to data having temporal validity. In the literature, considerable work has been devoted to transaction scheduling and concurrency control in RTDBs (Abbott & Garcia-Molina, 1988; Abbott & Garcia-Molina, 1989; Buchman et al., 1989). Most of the proposed algorithms are based on two basic mechanisms: two-phase locking (2PL) (Eswaran, et al., 1976) and optimistic concurrency control (OCC) (Kung & Robinson, 1981) and use priority information in resolving data conflicts. However, RTDBs are significantly different from conventional DBMSs with respect to performance goals and processing considerations. Lee (1996) pointed out that some results from previous performance studies on concurrency control protocols for RTDBs, instead of being definitive, are contradictory (e.g. between Haritsa et al., 1990 and Huang, 1991 and between Abbott & Garcia-Molina, 1989 and Huang, et al., 1991).

Haritsa and Ramamritham (2000), in "RTDBs in the New Millennium," agreed among the communiqués that a framework that substantiates the earlier experimental results of simulation-based performance of RTDBs need to be investigated. Furthermore, in an empirical evaluation of software visualization carried out by Lawrence et al. (1994), it was found that students who were able to control and interact with a variety of algorithm animations gained better understanding of the algorithm's behavior than those who could only passively observe the visualizations.

For a designer of real-time database concurrency control algorithms to be able to conceive imaginative solutions and understand the ideas presented by other members of the design team, he or she needs to possess outstanding visual thinking and design modeling abilities. Keim (1997) found that application of visualization techniques to support people's understanding of a database structure has been receiving growing attention during the last few years. By enabling RTDBs users and designers to observe and interact with their algorithms, it is hoped that a better understanding of the behavior of the algorithms is achieved. However, visualization can be achieved only with a good understanding of the structural meaning of the components of the RTDBs.

The major problem that needs to be addressed is how to visualize the runtime behaviour such as concurrency control algorithms. This study focuses on designing a framework that substantiates earlier experimental results of simulation-based performance using a visualization approach. To resolve the conflicts among different CC algorithms, as in the examples above, a method of visualization is now proposed.

The rest of this paper is organized as follows. Related work is presented in Section 2. The architecture is described in Section 3. The implementation procedure and evaluations of the design are presented in Section 4. In Section 5 we present conclusion and future works.

2 A REAL-TIME DATABASE MODEL

In a real-time database system, transactions travel through various components until their termination. Here a RTDB system model is presented with all its various components described as shown in Figure 1 below. This model is adopted with minor modifications from Stankovic (1991). In contrast to tasks in conventional real-time operating systems, the model assumes *transactions* to be the schedule unit. Thus, it is unlike the model proposed in Bauchmann et al. (1989), where the system's load consists solely of tasks, which in turn may contain a number of transactions within it. The model here contains transactions as schedulable-units counter to tasks (Bauchmann, et al., 1989). In the Bauchmann model, where tasks are the main schedulable unit, the following issues must be considered:

- (i) The type of deadline associated with a task, that is, soft, firm or hard.
- (ii) Type of deadline associated with the transaction within a task, that is, soft, firm or hard.
- (iii) Derivation of a deadline for each of the transactions within a task such that all of them are initially feasible.
- (iv) Determination of whether transactions within a task are cooperative or independent.
- (v) Determination of whether the execution of transactions within a task is serial or concurrent.
- (vi) Consideration of when a transaction's deadline is soft, what happens if a transaction misses its deadline? That is, could a transaction's behavior jeopardize the execution of other transactions within the task if the transaction is allowed to execute past its absolute deadline.
- (vii) If one of its transactions misses its deadline, would the tasks abort or continue its execution? If a transaction's failure signals the task's failure, it implies that the failure of one transaction jeopardizes the execution of the remaining transactions within the task.

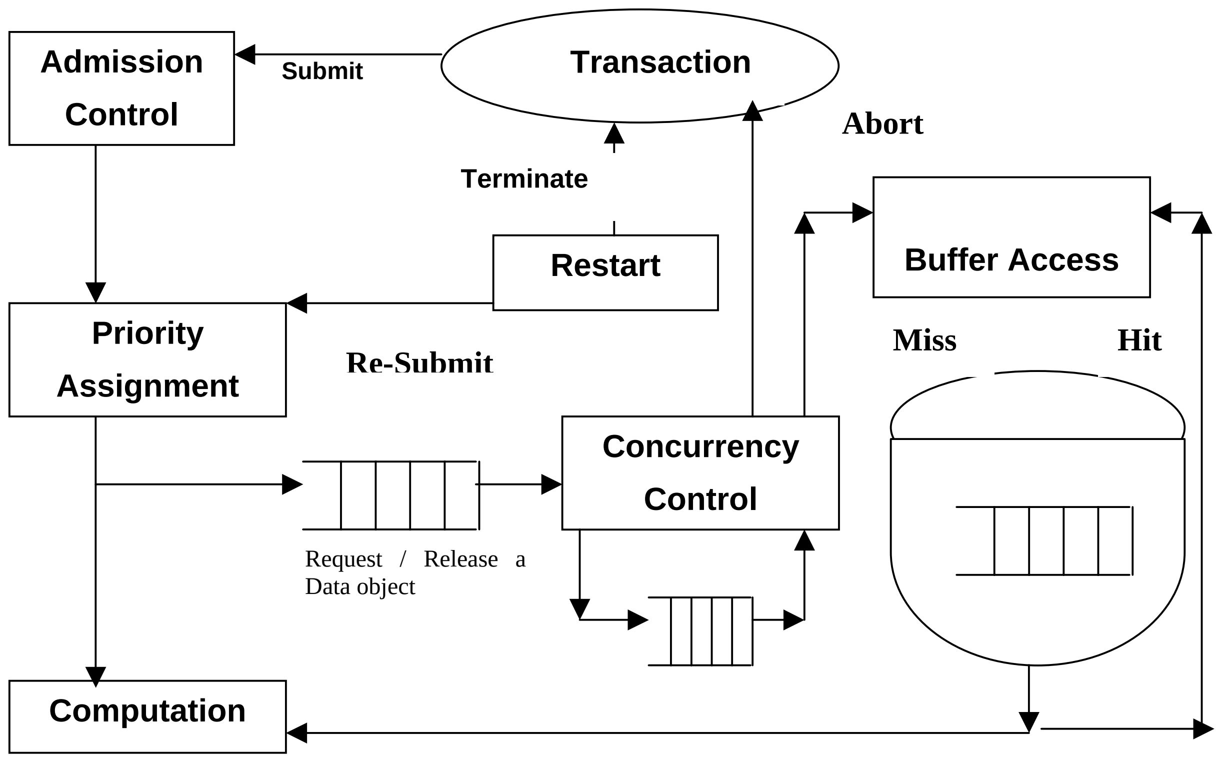


Figure 1. Real-time database system model (developed from Stankovic, 1991)

Our model consists of transactions only, with each transaction having attributes just as tasks in conventional real-time operating systems, for example, deadlines, periodicity, criticalness, priority, etc. However, the database itself is considered the main resource. Therefore, transaction's scheduling in this study as opposed to CPU scheduling (which exists at a lower level below that of database) is concerned with scheduling of transaction access to the database.

2.1 Concurrency Control

Ramamritham (1993) and Stankovic (1991) stated that the fundamental challenge of RTDB systems is the unification of priority-driven CPU scheduling, and database control protocols in order to maximize both concurrency and resource utilization, while subjected to data consistency (logical and temporal), transaction correctness, and transaction timing constraints. Some of the several techniques that can implement concurrency control protocols for conventional non-real-time database systems are: loading, time stamping, multi-version, and validation (also known as certification and as optimistic concurrency control protocols). Different assumptions are used to design each of these mechanisms, all of which have the same goal, enforcing serializability. Existing concurrency control protocols ensuring serializability are based on either *blocking* and/or *restarting transactions*. Blocking periods might outlive the prescribed deadlines in addition to introducing deadlocks and the priority-inversion problem, while restarting transactions wastes processing time and system resources. In addition, the previous works of Graham (1992), Kao (1995), Ulusoy (1992), and Ulusoy (1995) suggested that it might cause the restarted transactions to miss their prescribed deadlines. Deep investigations have been carried out on the performance and characteristics of those mechanisms for conventional database systems. However, Haritsa, et al. (1992) showed that there is need for modification of such data access protocols and proper re-evaluation of their trade-offs under RTDB systems.

2.2 Conflict Resolution

Conflict resolution protocols determine which of the conflicting transactions will actually obtain access to a data item while priority assignment governs CPU scheduling. The result of concurrent execution of transactions performing incompatible operations is usually conflict, that is read or write, on the same data item at the same time. Schedulers differ in detecting resource conflicts among transactions and the manner in which the conflicts are resolved once they are detected. Abbott (1989) suggested that the preempted transaction usually must be rolled back if in place. Updates are employed for shared database resources.

If a request by a transaction is a lock on a data item and the lock on the data item is in a conflicting mode from another transaction, both transactions have time constraints, yet only one can hold the lock. The conflict should be resolved according to the characteristics of the conflicting transactions.

2.2.1 Priority-base Wound-Wait Conflict Resolution

The wound-wait technique was originally proposed by Rosekrantz,, et al. (1978) for avoiding deadlocks. The original scheme was designed to use timestamps. However, the scheme was modified by Abbott and Garcia-Molina (1988) and Abbott (1989) so that it uses priorities instead of timestamps. The modified version is applied to resolve conflicts in RTDB systems. Abbott (1988) and Abbott (1989) modified it into versions known as High-Priority (HP) and as Priority-Abort (PA). Below is the outline of the general algorithm

```
Let: P(Ti) be the priority of transaction Ti.
      Ti requests a lock on data item D.
If (no conflict) then Tr accesses D
    Else – Th is holding the requested data item; resolve the conflict as follows
If (P(Tr)> P(Th) then Th is aborted
   Else Tr waits for the lock, that is, blocks.
```

If two transactions are involved in a conflict in the HP scheme, then the lower-priority transaction is aborted in order to free up the required resources for the higher-priority transaction. To preserve serializability, the preempted lower-priority transaction is rolled back, and when restarted, it must execute from the beginning.

3 A FRAMEWORK FOR VISUALIZING A CONCURRENCY CONTROL ALGORITHM FOR A RTDBS

3.1 The Visualization Cycle

This model was first proposed by Upson et al. in 1989. It draws on the similarities between the methods of producing results and the analysis of those results. It recognizes that the model used to produce a visual representation of the data can be thought of in the same manner as the model used to obtain the data in the first place if we consider the model used to describe the production of the results.

In order to simulate such a system, it is necessary to produce a mathematical approximation of the system that we are interested in. The mathematical approximation needs to include information that is relevant to the simulation and exclude information that is not. It is vital to the success of the simulation that the correct information is included. Once the simulation has been formulated, it is run, and results are produced. ***Analysis of the results of such a simulation frequently exposes either failings in the mathematical approximation itself or reveals the need for a greater level of detail***. The system is subsequently updated, the simulation repeated, and the results re-analyzed.

This process continues until a satisfactory result is obtained. The process is summarized in the diagram below.

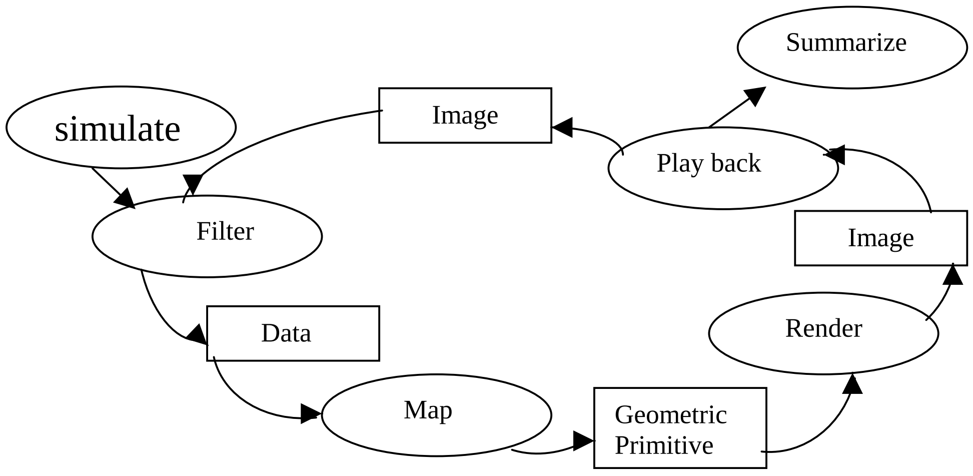


Figure 2. The Computational Cycle (adapted from Upson et al., 1989)

The model proposed by Upson et al. (1989) for visualization follows from the notion of repeatedly performing the simulation, perhaps with an increasingly accurate model, until a satisfactory set of results is produced. However, in the case of the visualization cycle, it is the analysis, rather than the production of the results, that is repeated. The cycle involved in analyzing the results of the numerical simulation is divided into a number of steps. These steps are repeated until a satisfactory representation of the data is obtained. The individual steps involved are as follows.

(i) The data from the simulation are *filtered* into an appropriate format. This may be simply a case of reordering the data, or it may involve selecting a subset of data, which is of most interest at any one time.

(ii) The filtered data are then *mapped* into a form that can be displayed. Upson et al. (1989) considers the case where the data are mapped onto a number of geometric primitives, such as triangles, rectangles, etc., which are used to build up the final picture.

(iii) The geometric primitives produced in the previous stage are then *rendered* into pictures which we can examine.

Once the image (or perhaps even a series of images) has been produced, it can be compared with the expected, or experimental, result. If the image is not sufficiently detailed or is not as expected, the analysis cycle can be repeated with more data included or with a finer mapping until the results obtained are satisfactory.

3.2 CC-RTDBS Model

In this section, we consider alternatives of modeling assumptions, which have significant impacts on the performance of concurrency control protocols in RTDBS. Combining the alternatives of those assumptions, we design our experiments for this study.

3.2.1 Concurrency Control Model

Following the Upson et al. (1989) postulation of discussing mathematical representation of the domain area, Lee (1996) modeled a soft deadline system of RTDBS as a single M/M/1 queue. The computation of the miss percentage for this system can be done as follows: Consider a transaction with slack time, *S*. Then the probability that it misses its deadline is equal to the probability that it must wait in the server queue more than *S* time units. Given an M/M/1 system with mean inter-arrival time $1/\lambda$ and mean service time 1.0, the waiting time distribution W(s) is given by Kleinrock (1975):

$$\begin{aligned} W(s) &= \Pr(\text{time spent in queue} \subseteq s) \\ &= 1 - \lambda e^{-(1-\lambda)s} \end{aligned} \qquad (1)$$

The probability that the transaction waits more than S time units and misses its deadline is 1 - W(s). To compute the expected probability of a missed deadline, we need the distribution of *S*. However, for simplicity we replace S by some constant, D and take this probability to be analogous to the miss percentage of the soft deadline system denoted as $MP_{SD.}$ Thus we obtain the following equation:

$$MP_{SD} = \lambda e^{-(1-\lambda)D} \qquad (2)$$

Furthermore, a firm deadline system can be modeled by a M/M/1/N queue, which is identical to a M/M/1 system except that the size of the queue is finite, and hence customers are turned away and not accepted by the queue if it is full. The probability customers are turned away is called the probability of *blocking*. We take the probability to be roughly analogous to the miss percentage of the firm deadline system and denote it by MP_{FD}. Given an M/M/1/N system with mean inter-arrival time $1/\lambda$ and mean service time 1.0, the blocking probability MP_{FD} is given by Kleinrock (1975):

$$MP_{FD} = ((1-\lambda)/(1-\lambda^{(N+1\lambda)}))\lambda^{N} \qquad (3)$$

Experimental Environment

This section outlines the structure and details of the simulation model and experimental environment, which were used to evaluate the performance of concurrency control algorithms for RTDBSs.

Parameter Setting

Table 1 gives the names and meanings of the parameters that control system resources. The parameters, CPUTime and DiskTime capture the CPU and disk processing times per data page. Our simulation system does not explicitly account for the time needed for transaction management, data operation scheduling, data management, or context switching. We assume that those costs are included in the CPU on a per data object basis.

The use of a database buffer pool is simulated using a probability rather than tracing each buffer page individually. When a transaction attempts to read a data page, the system determines whether the page is in the memory, so the transaction can continue processing without disk access. Otherwise, an I/O service request is created.

Table 2 summarizes the key parameters that characterize system workload and transactions. Transactions arrive in a Poisson stream, i.e. their inter-arrival times are exponentially distributed. The ArriRate parameter specifies the mean rate of transaction arrivals. The number of data objects accessed by a transaction is determined by a triangular distribution (as an approximation of a normal distribution) with mean TranSize, and the actual data objects are determined uniformly from the database. A page is updated with the probability, WriteProb.

Table 1. System Resource Parameters

Parameter	Meaning	Base Value
DBSize	Number of data pages in database	400
NumCPUs	Number of processors	2
NumDisks	Number of disks	4
CPUTime	CPU time for processing a page	15 msec
DiskTime	Disk Service time for a page	25 msec
BufProb	Probability of a page in memory buffer	0.5

Table 2. Workload Parameters

Parameter	Meaning	Base Value
ArriRate	Mean transaction arrival rate	-
WaitingTime	Mean waiting time	-
TranSize	Average transaction size (in pages)	10
WriteProb	Write Probability per accessed page	0.25
MinSlack	Minimum slack factor	2
MaxSlack	Maximum slack factor	8

The assignment of deadlines to transactions is controlled by the parameters MinSlack and MaxSlack, which set a lower and upper bound, respectively, on a transaction's slack time. We use the following formula for deadline-assignment to a transaction:

Deadline=AT + Uniform (MinSlack, MaxSlack)*ET

AT and ET denote the arrival time and execution time respectively. The execution time of a transaction used in this formula is not an actual execution time but a time estimated using the values of parameters TranSize, CPUTime, and DiskTime. In this system, the priorities of transactions are assigned by the Earliest Deadline First policy (Liu & Layland, 1973), which uses only deadline information to decide transaction priority but not any other information on transaction execution time.

Mapping a Design to Code

At the very least, design class diagrams depict the class name, super classes, method signatures, and simple attributes of a class. This is sufficient to create a basic class definition in an object-oriented programming language.

Animation Control Code

The animation codes available in CC-RTDBS can be applied either during the RTDBS run (online) or disabled while off-line visualization is being displayed. Below is the pseudocode used for animation.

```
Dim time01, time02, timeDiff
Private Sub Form_Load()
time01 = Time
End Sub
Private Sub tmDelay_Timer()
  time02 = Time
 timeDiff = DateDiff("s", time01, time02)
 If timeDiff >= 3 Then
 Unload Me   'unload the form
      End If
      End Sub
```

4 EXPERIMENTS AND RESULTS

The simulation module was included inside the back-end program using Visual-Basic 6.0. The data collection in the experiments is based on the method of replication. For each experiment, we ran the simulation with the same parameter values for at least 25 different random number seeds. Each run continued until 400 transactions were executed. For each run, the statistics gathered during the first few seconds were discarded in order to let the system stabilize after initial transient conditions.

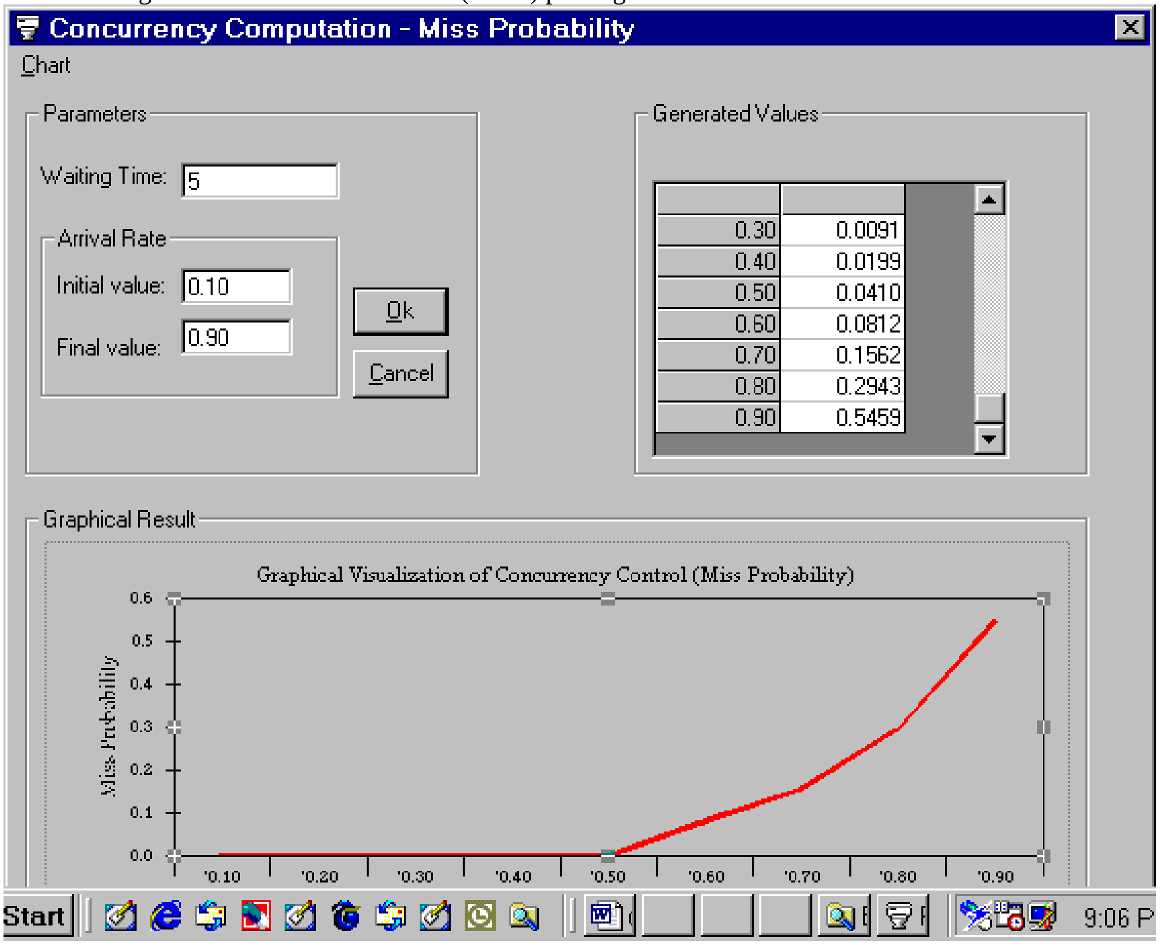
The experiments conducted for this study were designed to investigate the impact of data contention, resource contention, and the two policies for dealing with tardy transactions on the performance of concurrency control protocols in RTDBs. The system condition considered for experiment is a soft deadline policy under a finite resource situation. For the condition chosen, system performance was evaluated over a wide range of workloads. The system workload was controlled by the arrival rate of transactions.

As stated earlier, the bridge between the conventional and real-time databases lies on the algorithms and proper evaluation. Visualization assists in evaluating CC-RTDBs performance by studying the characteristics involved. However, a snapshot of a representative portion of the output can be examined via a graphical user interface.

Program visualization supports user understanding of the program's code and data values, whereas algorithm visualization supports user understanding of the algorithm's instructions and generic data structures.

Interface Design

This subsection describes the individual views and navigators available in CC-RTDBs. Figure 3 shows an example of a screen image taken from CC-RTDBs and contains a coarse-grained miss ratio versus number of transactions. The top right panel shows a fine-gained visualization, and the top left contains a text box range selector using the Model-View-Controller (MVC) paradigm.



Concurrency Computation - Miss Probability

Chart

Parameters

Waiting Time: 5

Arrival Rate

Initial value: 0.10

Final value: 0.90

Generated Values

0.30	0.0091
0.40	0.0199
0.50	0.0410
0.60	0.0812
0.70	0.1562
0.80	0.2943
0.90	0.5459

Graphical Result

Figure 3. Description of an interface design of visualizing miss probability (as a ratio)

Figure 4 shows offline visualization.

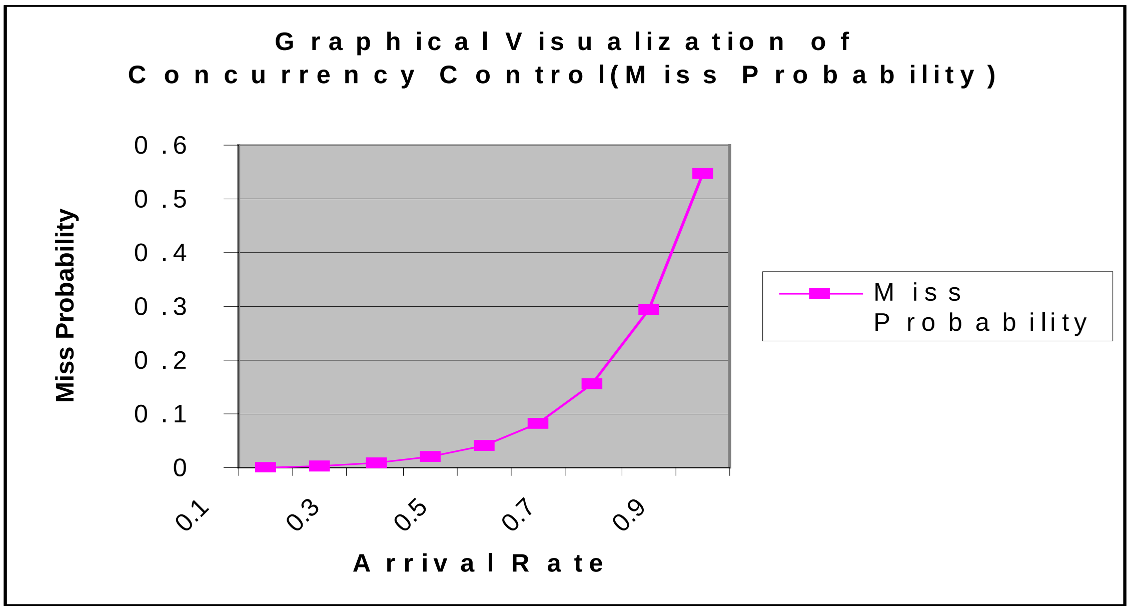


Figure 4. Offline visualization of concurrency control miss probability (as a ratio)

Results from Graphical Package: MS-Excel

For comparative studies, Figures 3 and 4 show the former method of displaying performance behaviour graphically through electronic spreadsheet program of which the conflicting features are not easily monitored dynamically.

One of the main characteristics of performance evaluation in RTDBs is concurrency control. One objective of this work relates to the understanding of the actual behaviour of algorithm of choice and the determination of quality of the performance evaluation. This is the essence of the next set of experiments. In all the experiments, parameters are entered, and the behaviour during the execution is monitored. When needed, the parameters are varied, and the visualization tool is re-run.

The graphical user interfaces with their components are displayed in Figures 3 and 4 respectively. From detailed analysis, we observe that when the number of transactions (NR) equals 8 and the computed miss ratio (MR) equals 0.5, the slope increases; whereas when the number of transactions (NR) equal or exceed 16 and the miss ratio(MR) equals 0.9±0.01, the slope fluctuates. However, when the parameters are doubled, the number of transactions equals 16, the miss ratio equals 0.6, and the slope increases. Finally, with the number of transactions equal to or exceeding 32 and the miss ratio equaling 0.8±0.01, the slope smoothes, tending to a linear graph. This example shows the power of visualization in understanding the algorithm and its results.

The same experiments were performed for miss probability and data blocking for both dynamic visualization and offline visualization with various test parameters.

5 CONCLUSIONS AND FUTURE DIRECTIONS

The work studies concurrency control protocols performance of RTDBs, which covers a wide range of applications. On one hand, there is the question of understanding the actual behaviour of the algorithms of choice, and on the other is the question of knowing actual behaviour in determining the quality of the performance evaluation of algorithms related to RTDBs. In the former case, the decision to understand actual behaviour is largely a voluntary one taken to know the dynamics of the algorithms. In the latter case, the RTDBs designers are unaware of the algorithms' behaviour. They have no way of identifying the contribution of the different components of the algorithm and, therefore, no direct way of analyzing the algorithm design and assigning credit to good algorithm components, e.g. parameters, or locating and improving ineffective algorithm components. In this investigation, the two extremes were subsumed as special cases for CC-RTDBs algorithms' behaviour.

Furthermore, the study has also derived a "Cc-RTDBs" framework that enables RTDBs designers to specify their visualization in terms of players, views, mapping and navigators. As a result, a designer can avoid much of the low-level graphics programming associated with visualization. The study presented the "Cc-RTDBs" framework that provides designers with an object-oriented view hierarchy from which they can select and apply a view to produce a standard visualization or inherit the attributes of an existing view and adapt them to a new view. Generally, it has been demonstrated that using our flexible CC-RTDBs visualization tool helps substantiate the earlier experimental results of simulation-based performance studies with visualization techniques.

6 REFERENCES

Abbott, R. & Garcia-Molina, H. (1988) *"Scheduling Real-Time Transactions. SIGMOD Record, Vol. 17, No. 1.*

Abbott, R. & Garcia-Molina, H. (1989) Scheduling Real-Time Transactions with Disk Resident Data. *Proceedings of the Fifteenth International Conference on Very Large Databases*, pp. 385-396.

Buchman, A., McCarth, D. R., Hsu, M., & Dayal, U. (1989) Time Critical Database Scheduling: A framework For Integrating Real-time Scheduling and Concurrency Control. *Proceeding of Real-time Systems Symposium*, pp. 470-480.

Eswaran, K.P., Gray, J., Lorie, R. & Traiger, I. (1976) The Notions of Consistency and Predicate Locks in a Database System. *Communications of the ACM, 19(11).*

Graham, M. H (1992) Issues in Real-Time Data Management. *The Journal of Real-Time Systems, 4*, pp. 185-202.

Haritsa, J. R., Carey, M. J, & Livny, M (1990) On Being Optimistic about Real-Time Constraints. *Proceedings of the 1990 ACM SIGACT-SIGART-SIGMOD Symposium on Principles of Database Systems (PODS).*

Haritsa, J. R., Carey, M. J, & Livny, M (1992) Data Access Scheduling in Firm Real-Time Database Systems. *The Journal of Real-Time Systems, 4*, pp. 203-241.

Haritsa, J. R., & Ramamritham, K. (2000) Real-Time Database Systems in the New Millenium. *Real-Time Systems, 2000, 19(3*): 205 - 215.

Huang, J. (1991) *Real-Time Transaction Processing: Design, Implementation, and Performance Evaluation,* PhD thesis.

Kao, B. & Garcia-Molina, H. (1995) An Overview of Real-Time Database Systems. In S. H. Son (Ed.) *Advances in Real-Time Systems*. Prentice-Hall: Englewood Cliffs, NJ.

Keim, D. (1997) Visual Data Mining. Tutorial in *International Conference on Very Large Databases (VLDB'97),* Athens Greece.

Kleinrock, L (1975) *Queueing Systems, Volume 1*. John Wiley and Sons: New York

Kung, H.T. & Robinson, J. (1981) On Optimistic Methods for Concurrency Control. *ACM Transactions on Database Systems 6(2)*: 213-226.

Lawrence, A., Badre, A., & Stasko, J. (1994) Empirically evaluating the use of animations to teach algorithms. *Technical Report GIT-Gvu-94-07*, Graphics Visualization and Suability Center, Georgia Institute of Technology, Atlanta GA.

Lee, J. (1996) *Concurrency Control Algorithms for Real-Time Database Systems*, Ph.D. thesis, University of Virginia, Charlottesville, VA.

Liu, C., & Layland, J. (1973) Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *J. ACM 10(1).*

Ramamritham, K. (1993) Real-Time Databases. *Intl. Journal of Distributed and Parallel Databases*.

Stankovic, J., Ramamritham, K., & Towsley, D. (1991) Scheduling In Real-Time Transaction Systems. In van Tilborg & Koob (Eds.) *Foundations of Real-Time Computing: Scheduling and Resource Management*. Kluwer Academic Publishers.

Ulusoy, O. (1992) Current Research on Real-Time Databases. *SIGMOID Record Vol. 21, No.4*.

Ulusoy, O. (1995) Research Issues in Real-Time Database Systems. *Information Sciences 87,* pp.123-151.

Upson, C., Faulhaber Jr., T., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., & Van Dam, A. (1989) The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics & Applications* (July) 30-42.