A PERFORMANCE STUDY OF CONCURRENCY CONTROL IN A REAL-TIME MAIN MEMORY DATABASE SYSTEM

Le Gruenwald **Sichen Liu**

School of Computer Science
The University of Oklahoma
Norman, OK 73019

ABSTRACT *Earlier performance studies of concurrency control algorithms show that in a disk-resident real-time database system, optimistic algorithms perform better than two phase locking with higher priority (2PL-HP). In a main memory real-time database system, disk I/Os are eliminated and thus more transactions are enabled to meet their real-time constraints. Lack of disk I/Os in this environment requires concurrency control be re-examined. This paper conducts a simulation study to compare 2PL-HP with a real time optimistic concurrency control algorithm (OPT-WAIT-50) for a real time main memory database system, MARS. The results show that OPT-WAIT-50 outperforms 2PL-HP with finite resources.*

1. INTRODUCTION

In a real-time database system (RTDBMS), transactions are not only executed correctly but also completed within their deadlines. This system is needed for many applications in which meeting transaction deadlines is crucial. ([Son, 1992], [Ulusoy, 1992]). For example, in telecommunications, transactions supporting call setups and timestamp for billing information must be completed in a very short amount of time. In radar systems, database operations involved in recognizing, tracking, and controlling objects must meet real-time requirements to be of any use. To ensure that transactions (or the majority of transactions) will meet their deadlines, among many issues that a RTDBMS designer must be concerned about is *concurrency control*. Specifically, the designer must provide an answer to each of the following questions: Should locking or time-stamping or optimistic be used? What kind of transaction priority assignment should be employed? If locking is used, then how priority inversion should be handled?

Earlier studies of concurrency control algorithms based on locking and optimistic approaches have addressed the problems of concurrency control in RTDBMS ([Abbott, 1988], [Abbott, 1989], [Carey, 1989], [Haritsa, 1989, 1990], [[Hung, 1992], [Son, 1993]). Recent papers [Haritsa, 1989; 1990] have conducted a performance comparison between two phase locking with higher priority (2PL-HP) and optimistic broadcasting (OPT-BC) algorithm, and a comparison among different versions of OPT-BC algorithm, including OPT-BC, OPT-Sacrifice, OPT-Wait, and OPT-Wait-50. They concluded that in a firm real-time environment, OPT-BC outperforms 2PL-HP and OPT-WAIT-50 is the best one among the various versions of OPT-BC.

Previous studies of concurrency control algorithms for RTDBMS are mostly based on disk-resident databases. We recognize that in a main memory database (MMDB) environment, the primary copy of the database is memory-resident. Disk I/Os are thus eliminated which subsequently allows many transactions to meet their real-time constraints. The performance of concurrency control algorithms in MMDB may be different from that in a disk-based database system since the time needed for accessing data objects is different in both environments. For example, the time for locking a page in a disk-resident database is much less than that for accessing a page. However, in MMDB the time for locking a page is compatible to that for accessing a page. Overhead incurred in obtaining/releasing locks may be unacceptable. Time needed for validation in an optimistic approach may be two high for MMDB transaction processing, which might cause many transactions

to miss their deadlines. Therefore it is important for us to examine the concurrency control issue for a RTDBMS in which the database is memory-resident. We call this a real-time main memory database (RTMMDB) system.

The objective of this paper is to conduct a performance comparison of two concurrency control protocols, 2PL-HP and OPT-WAIT-50, which have been proposed by [Abbott, 1988] and [Haritsa, 1990] for a main memory database system, MARS ([Eich, 1987], [Gruenwald, 1990], [Gruenwald, 1991]). In Section 2, we describe these two protocols. In Section 3, an overview of MARS is given. Section 4 describes our simulation model and analyzes the simulation results. Section 5 concludes the paper.

2. DESCRIPTIONS OF 2PL-HP AND OPT-WAIT-50

In 2PL-HP [Abbott, 1988], two-phase locking is augmented with a priority scheme to ensure that high priority transactions are not delayed by lower priority transactions. The locking process is divided into two phases [Korth, 1991]. In the growing phase, a transaction may obtain locks but may not release any lock; in the shrinking phase, a transaction may release locks but may not obtain any locks. When a transaction requests a lock on an object, if the lock requester's priority is higher than that of the lock holder, the holder is restarted, and the requester gets the lock. Otherwise, the requester has to wait for the holder to release the lock.

OPT-WAIT-50 is based on the optimistic approach [Haritsa,1990], Transactions are executed and validated before they commit. When a transaction reaches its validation phase, it is made to wait when the percentage of higher priority transactions is greater than 50%, that is, while half or more of its conflict set is composed of higher priority transactions [Haritsa,1990]. This gives a chance for higher prirority transactions to meet their deadlines first. When the percentage falls below 50%, the transactions in the conflict sets are restarted and the validating transaction is committed unless it has been restarted due to the commit of one of higher priority transactions in its conflict set.

3. MARS OVERVIEW

MARS (MAin memory with a Recoverable Stable log) is shown in Figure 1. It has two processors: database processor (DP) and recovery processor (RP). Both processors execute independently. The primary copy of the database is in a volatile main memory (MM). The backup copy of database is on an archive memory (AM). DP receives transaction requests from the host processor, performs database processing, and sends the results back to the host. RP is in charge of logging, transaction termination, checkpointing, and recovery. RP periodically performs a fuzzy checkpointing by copying dirty (modified) pages from MM to AM periodically. All updates take place in a stable memory (SM). When a transaction commits, then its updates are copied to the permanent database stored in MM.

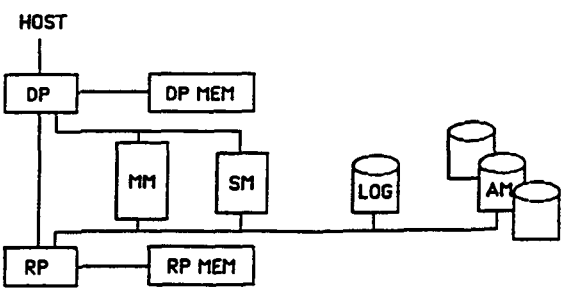

Figure 1. MARS Architecture

4. SIMULATION DESCRIPTIONS

We have written a simulation program using the simulation language SLAM II to measure the performance of 2PL-HP and OPT-WAIT-50 in terms of percentage of transactions that miss their deadlines (miss percentage), and overall system throughput. The simulation is conducted for a firm real-time environment in which late transactions are discarded. The *earliest deadline* policy is used for priority assignment [Abbott, 1988]. This policy gives a transaction that has the earliest deadline the highest execution priority. In this Section, we describe simulation parameters, transaction representation, and transaction runtime estimates.

4.1. Simulation Parameters

Tables 1 and 2 show the dynamic and

static parameters used in our simulation program. The majority of these parameters were adopted from [Gruenwald, 1990]. In each simulation run, the static parameters remain the same while the dynamic parameters keep changing within its specified range.

Parameter	Meaning	Default values
ArrivalRate	mean number of transaction arrivals per time unit	300
DatabaseSize	number of pages in database	1800
WriteProb	write probability	0.2
ReadProb	read probability	0.8
SlackFactor	slack factor in deadline formula	6
PriorityPolicy	transaction priority policy	earliest deadline
LatePolicy	firm or soft deadline	firm deadline
MPL	multiprogramming level	10
NumProcessor	number of processors	DP and RP

Table 1. Dynamic Parameters

Parameter	Meaning	Default values
SM_ACCESS	access an SM word	0.00011 ms
ALLOC_TM	Allocate a MM page	0.05 ms
AMREQ_TM	Request an I/O from AM	0.02 ms
PRETRAN	PREPROCESS A TRANSACTION	1.25 ms
PREOP	Preprocess an operation	0.005 ms
RELEASE_TM	Release an MM page	0.05 ms
BMAP_TM	Read until 1 in bit map	0.00011 ms
MM_ACCESS	Access an MM word	0.0001 ms
SM_SEAR	SM address translation	0.5 * MM_ACCESS
MM_SEAR	MM address translation	3 * MM_ACCESS
MSEEK	Minimum seek time	3 ms
REC_SZ	SM or log record	12 bytes
ET_TM	End transaction	1.25 ms
INTIO_TM	Initiate log I/O	0.01 ms
LOGIO_TM	Write a log page	12 ms
LOGPG_SZ	Log page size	2000 bytes
WORD_SZ	Bytes per word	4 bytes
TRACK_CYL	Tracks per Cylinder	15
SEEK	Average seek time	16 ms
LATENCY	Average latency	8.3 ms
INDN_TM	Initial down time	5 ms
LOCK_TM	Get one lock	0.025 ms
UNLK_TM	Release one lock	0.025 ms
NUM_AFIMS	Number of committed AFIM in log	10
CPU_POWER	Processor Power	2 MIPS

Table 2. Static Parameters

4.2. Transaction Representation and Runtime Estimate

In our simulation, the database is organized as a collection of pages. Transactions arrive in an exponential distribution. Each transaction consist of an identifier, arrival time (AT), deadline (DT), run time estimate (RT), operations (read/write), and

data pages processed by the operations. The deadline (DT) is computed using the following formula: DT =AT + SF * RT. Slack Factor (SF) is a positive number greater than or equal to 1 which is used to tight or slack the deadlines. If SF is larger, transactions are allowed to have more time to run. Otherwise, they have less time to complete. We represent the tightness or slackness of the deadline of a transaction by varying the slack factor. The run time estimate of a transaction is the worst time needed to complete all its operations.

Table 3 illustrates all processing steps a transaction must go through in MARS and times required to finish these steps when the 2PL-HP algorithm is being used for concurrency control. The runtime estimate for a transaction is the total time need to complete all these steps. In Table 3, NUMOPG represents the total number of operations; NUMPGS represents the number of pages needed; NUMWRT is the total number of write operations. Note that checkpoint time is not included in the runtime estimate since we assume that checkpoint is performed by RP with the lowest execution priority while DP performs normal transaction processing in parallel with RP. If RP is busy checkpointing and other activities such as transaction termination or logging need RP, checkpointing will be interrupted and RP will then be assigned to these activities immediately. Runtime estimate does not include logging time due to the following reasons. In MARS, logging only happens when the log buffer is full and log pages need to be flushed out onto log disks by RP. RP at this point will be assigned to logging which will cause some transactions to be delayed. Since logging does not occur very often in our system, and it is difficult to estimate which transactions are delayed by logging; we do not include logging time in transaction runtime estimate.

5. SIMULATION RESULTS

To compare the performance of 2PL-HP and OPT-WAIT-50, we conducted two testing cases: vary slack factor and vary arrival rate. Following are the results of these testing cases.

5.1. Vary Slack Factor

The slack factor is varied from 1 to 10 while the other parameters are kept the same for both 2PL-HP and OPT-WAIT-50 algorithms. The mean arrival rate is set to be 300. As shown in Figure 2a, with a lower slack factor, both algorithms show a higher miss percentage. This is expected since transactions are operating under very tight deadlines, cannot wait long for their turn to be executed, and thus have a higher chance to miss their deadlines. As slack factor increases, miss percentage decreases since transactions are given more time to complete. However, on average, OPT-WAIT-50 gives 45% less miss percentage than 2PL-HP does. One reason for this is that in the latter, transactions must wait to obtain locks on data objects . The time needed for lock requests is compatible to time for accessing data objects in MMDB. This overhead causes many transactions to miss their deadlines, while in the former technique, this overhead does not exist. Another reason is that in the latter, a transaction may be restarted by another transaction which later misses its deadline. This wastes CPU time, and in turn might yield many late transactions.

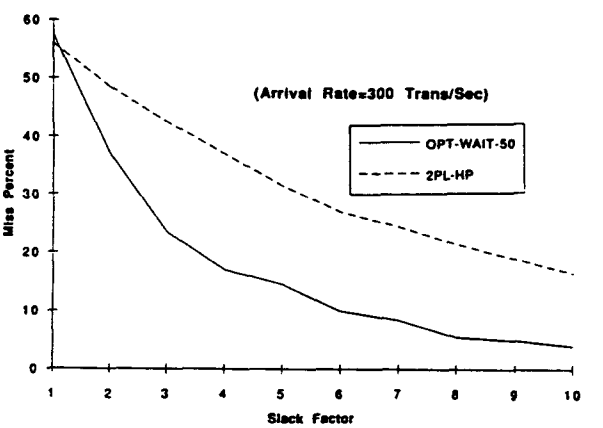

Figure 2a. Effect of Slack Factor on Miss Percentage

Figure 2b shows a comparison of throughput obtained in the two algorithms when changing slack factor. As slack factor increases, throughput also increases. However, when slack factor is in a lower range (between 1 to 6) the increase amount in throughput is much higher than that when slack factor is in a higher range (between 6 to 10). Throughput changes more drastically in OPT-WAIT-50 than in 2PL-HP. On average, OPT-WAIT-

50 yields about 20% better throughput than 2PL- HP.

Processing Steps	Time needed
(1) Preprocess of a transaction	PRETRAN
(2) Get all the locks of the data objects involved	NUMPGS *LOCK_TM
(3) Preprocess of operations	NUMOPS * PREOP
(4) Check whether the data object is in SM	SM_SEARCH
(5) If the data object is in SM, do one of the following:	
* read a SM word for each operation	SM_ACCESS
* write an entire SM record	SM_ACCESS * WORDS
(6) If the data object is not in SM, go to MM, and do one of the following:	MM_SEARCH + SM_SEARCH
* read a MM word for read operation	MM_ACCESS
* write an entire SM record for write operation	SM_ACCESS * WORDS
(7) repeat steps 3, 4, 5 and 6 until every operation is done	
(8) Commit time usage is only for write operations, for every write operation, do the following:	
* copy SM records to log buffer and also write BT and ET records	SM_ACCESS * NUMWRT * 2 * WORDS
* update Bit Map	BMAP_TM+SM_ACCESS*WORDS
(9) Unlock all locks	UNLOCK_TM*NUMPGS
(10) End transaction	ET_TM

Table 3. Steps and Times Involved in Runtime Estimate Calculation

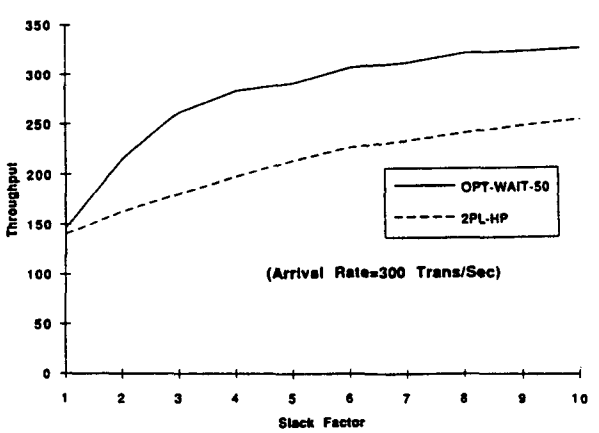

Figure 2b. Effect of Slack Factor on System Throughput

5.2. Vary Arrival Rate

As shown in Figure 3a, with a lower mean arrival rate, both algorithms show a lower value of miss percentage. As mean arrival rate increases, miss percent increases, and 2PL-HP increases sharply as mean arrival rate reaches about 250. As mean arrival rate is lower than 100, every transaction has enough time to complete, and no one missed its deadline. However, OPT-WAIT-50 performs about 40% better on average than 2PL-HP. As mean arrival rate increases, transactions have more conflicts on resources, and thus fewer transactions are committed. In the case of 2PL-HP, as mean arrival rate reaches 400, about 50% of the transactions missed their deadlines. At this arrival rate, transactions may have more conflicts on processors and other resources, and some of transactions have to wait for their pages, which subsequently might cause the transactions to miss their deadlines. In OPT-WAIT-50, since

transactions do not need to wait for pages, they are expected to run faster.

Figure 3b shows that the throughput in both algorithms increases as mean arrival rate increases within a range of 100 and 300, and decreases when mean arrival rate is higher than 300. This can be explained as follows. When mean arrival rate increases, more transactions are executed;thus system throughput increases. However when mean arrival rate gets above 300, conflicts among transactions are too high that cause transaction delays and restarts which in turn reduce system throughput. On average, throughput in OPT-WAT-50 is 34% better than that in 2PI-HP.

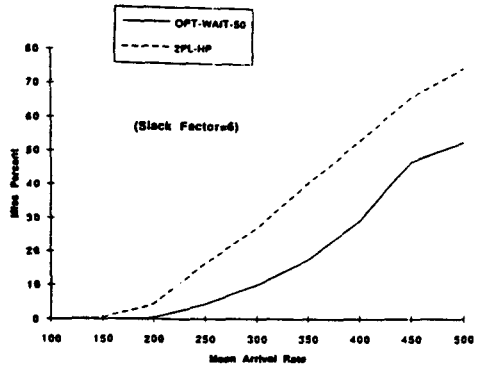

Figure 3a. Effect of Arrival Rate on Miss Percentage

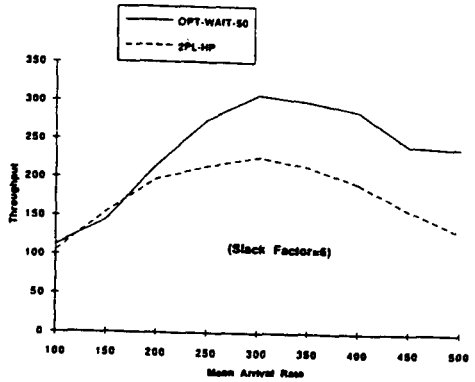

Figure 3b. Effect of Mean Arrival Rate on System Throughput

6. CONCLUSIONS

We have presented a simulation study of the relative performance of two concurrency control techniques, 2PL-HP and OPT-WAIT-50, using the earliest deadline priority assignment policy in a real-time main memory database system, MARS. The simulation experiments showed that in a firm real-time environment, as slack factor increases, deadline becomes slack so that transactions get more time to complete, and fewer transactions would miss their deadlines. As mean arrival rate increases, the conflicts on resources increase, transactions spend more time waiting for resources, and the number of late transactions also increases. Therefore, increasing slack factor and decreasing mean arrival rate would help more transactions to complete on time. On average, for a firm real-time environment, OPT-WAIT-50 performs about 50% better in terms of percentage of late transactions, and about 30% better in terms of system throughput than 2PL-HP in MARS. Our future research include conducting the simulation comparison for a soft real-time environment, and making use of different priority assignment policies.

7. REFERENCES

[Abbott,88] Abbott, R., and Garcia-Molina, H., "Scheduling Real Time Transactions: a performance evaluation", Proc. of the 14th VLDB conference, Aug. 1988.

[Abbott,89] Abbott, R., and Garcia-Molina, H., "Scheduling Real Time Transactions with Disk Resident Data", Proc. of the 15th VLDB conference, Agu. 1989.

[Carey, 89] Carey, M., Jauhari, R., and Livny, M., "Priority in DBMS Resource Scheduling," Proc. of the 15th VLDB Conference, Aug. 1989.

[Eich, 87] Margaret H., Eich, "MARS: The Design of A Main Memory Database Machine", Proc. of the 1987 International workshop on Database Machines, October, 1989.

[Eich, 89] Margaret H., Eich, "Main Memory Database Research Directions", Proc. of the 1989

International workshop on Database Machines, Deauville, France, June, 1989.

[Gruenwald, 90] Le Gruenwald, "Reload in a main memory database system: MARS", Ph.D. dissertation, Department of Computer Science, Southern Methodist University, Aug. 1990.

[Gruenwald, 91] Le Gruenwald, and Margaret H. Eich, :MMDB Reload Algorithms', ACM SIGMOD Record, Volume 20, No. 2, June 1991.

[Hung, 92] S. Hung, and K. Lam, "Locking Protocols for Concurrency Control in Real-Time Database Systems", SIGMOD RECORD, Vol. 21, No. 4, December 1992.

[Haritsa,89] Jayant R. Haritsa, Michael J. Carey and Miron Livny, "Dynamic Real Time Optimistic Concurrency control", Proc. of 11th Real Time Systems Symposium, Dec. 1990.

[Haritsa,90] Jayant R. Haritsa, Michael J. Carey and Miron Livny, "On Being Optimistic about Real Time Constraints", Symposium on Principles of Database Systems, Dec., 1990.

[Korth, 91] Henry Korth, and Abraham Silberchatz, "Database System Concepts", McGraw-Hill, NY, 1991.

[Son, 92] Sang Son, "Scheduling Real-Time Transactions Using Priority", Information and Software Technology, Vol 34, No. 6, June 1992.

[Son, 93] Sang Son and Seog Park, "Scheduling and Concurrency Control for Real-Time Database Systems", International Symposium on Database Systems for Advanced Applications, April 1993.

[Ulusoy, 92] Ozgur Ulusoy, "Current Research on Real-Time Databases", SIGMOD RECORD, Vol. 21, No. 4, December 1992.