The V4DB Testbed - Evaluating of Real-Time Database Transaction Processing Strategies

VÁCLAV KRÓL [1], JAN POKORNÝ [2]

[1] Institute of Information Technologies
Silesian University in Opava, Faculty of Business Administration in Karvina
Univerzitní náměstí 1934/3, 733 40 Karviná
CZECH REPUBLIC

[2] Department of Measurement and Control
VŠB Technical University Ostrava, Faculty of Electrical Engineering and Informatics
17. listopadu 15, 708 33 Ostrava
CZECH REPUBLIC

Abstract: - Previous research in real-time databases has focused primarily on evolution and evaluation of transaction processing algorithms, priority assignment strategies or concurrency control techniques. But for the most part the research efforts are based only on simulation studies with many parameters defined. It is very difficult to achieve guaranteed real time database services when putting a database into a real-time environment because various components can compete for system resources. So our objective was to design and implement an experimental real-time database system suitable for study of real time transaction processing. The experimental system was implemented as an integrated set of the most important functional parts of a veritable real-time database system. It serves as a support platform for performance evaluation of known and new algorithms of the particular processing components, including CPU scheduling, concurrency control and conflict resolution strategies. Because of the strong interactions among the processed components, proposed system can help us to understand their effect on system performance and to identify the most influencing factors. In this paper the overall system design is presented together with some experimental results showing the system testing possibilities.

Key-Words: - Real-time database, transaction processing, CPU scheduling, concurrency control

1 Introduction

Up to now the major part of RTDB research was focused on evolution and evaluation of transaction processing algorithms, priority assignment strategies and concurrency control techniques. Evaluation was usually based on simulation studies except a few exceptions ([4]). Simulations often consist of a number of parameters. The parameters specify maximal count of data items, average count of one transaction data pages, processor time needed to manipulate data items, average disk access time, probability of read vs. write transaction, etc. There is even a study where all the functional blocks are designed as object-oriented and described by means of classes with a number of attributes ([6]). Much less attention was paid to architecture aspects of the operating systems, developed especially for real-time systems and for better support of time critical operations. So two basic drawbacks of the presented research can be defined:

1. For the most part there is only one functional part considered for investigation without any interaction with other system parts. Because of the strong interactions among the various processing components in RTDBS, an integrated approach is necessary.
2. Research work at real-time transaction processing is based on simulation studies only. It is necessary to investigate the real-time transaction processing algorithms in their natural environment to achieve really relevant results. It means that the operating platform for RTDBS is a real-time operating system and the particular functional blocks communicate with each other by means of this operating system.

We developed real-time database testbed suitable for study of real time transaction processing named V4DB ([11]). In the next chapter the system design is briefly described.

2 The V4DB System

The system is currently implemented upon the real time operating system platform VxWorks as a centralized system with memory resident database. Overall design is presented on fig. 1. Oval blocks represent parallel processes while the square blocks are single functional blocks within processes. Some of the system parts contain grayed blocks. The blocks illustrate the possibility of functionality change of the parts. Their runtime behaviour can be changed.

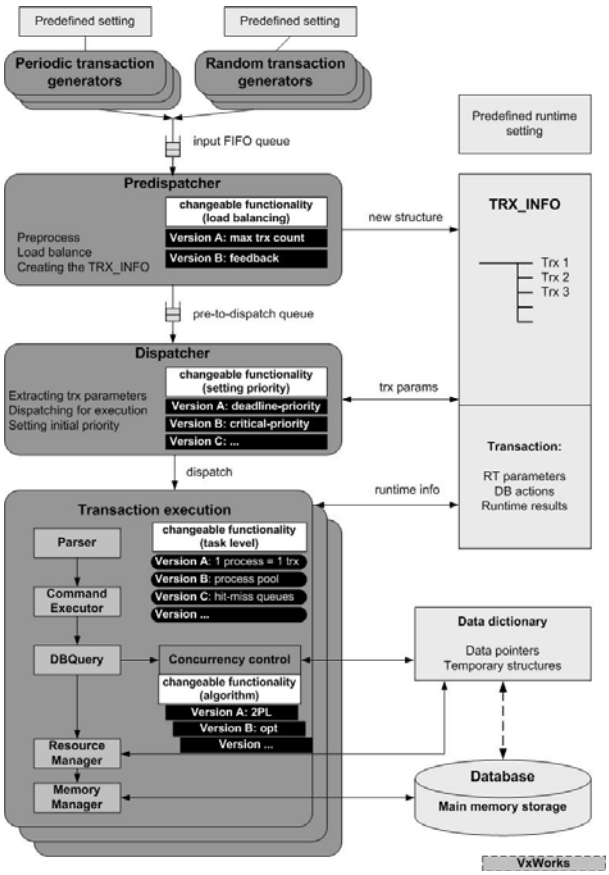


Fig.1.Experimental RTDB system

2.1. Predispatching

After the admission the transactions are predispatched. Predispatching includes admission control to avoid system overloading and creating the transaction info-structure. The structure fully describes the transaction definition and all its parameters.

2.2. Dispatching

In the next step the transaction parameters are extracted and dispatched for execution as to the priority assignment policy and the way of transaction processing. The priority assigned to a transaction execution process is mapped to a real operating system process priority and the context (transaction) switching is relied on an underlying operating system. This is one of the most important experimental system aspects.

2.3. Processing

When the transaction is scheduled for execution, first it is parsed into particular commands and then the commands are processed by the command executor. Database access must be synchronized through the concurrency control. The DBQuery block executes the commands on a logical level while the resource manager and memory manager work with physical data structures that are described by the data dictionary. To obtain reasonable performance, multiple transactions must be able to access data concurrently. So before a transaction performs an operation on a data object, it must be processed by concurrency control component in order to achieve the required synchronization.

2.4. Database

Regarding the project objective as an experimental system and application categories where RTDBS are used to advantage a simple schema is adopted in the following form: The database is divided into predefined count of memory areas. Each area represents some table and consists of predefined count of records. Records are of the same length for one memory area table just for simplicity. Database schema is outlined on fig. 2.

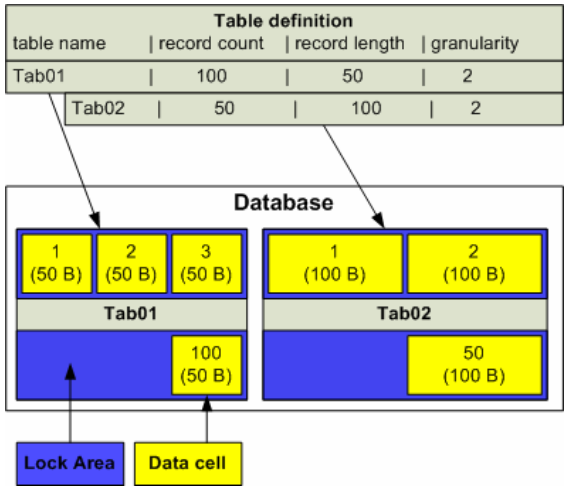


Fig.2.Database schema

The database schema can be defined by the notation:

tab name | rec count | rec byte length

for example

Tab01 | 100 | 50

- means that there exists a table named Tab01 which has 100 records each of 50 bytes in length.

2.4.1. Database granularity

The granularity parameter can be defined for each table mentioned above. The parameter stands for the count of logical areas into which each table is divided for the needs of concurrency control during transaction processing. The granularity is defined separately for

each table, so the parameter can be added to the table definition and the final database schema looks like that:

tab name | rec count | rec byte length | granularity

for example

Tab01 | 100 | 50 | 2

- means that there exists a table named Tab01 which has 100 records each of 50 bytes in length, the table is divided into two logical areas according to granularity 2.

2.5. Description of the transactions

Transactions are generated by the internal generators. To study the database transaction processing, it must be able to generate transactions which properties are known and set in advance. The parameters are described on fig. 3.

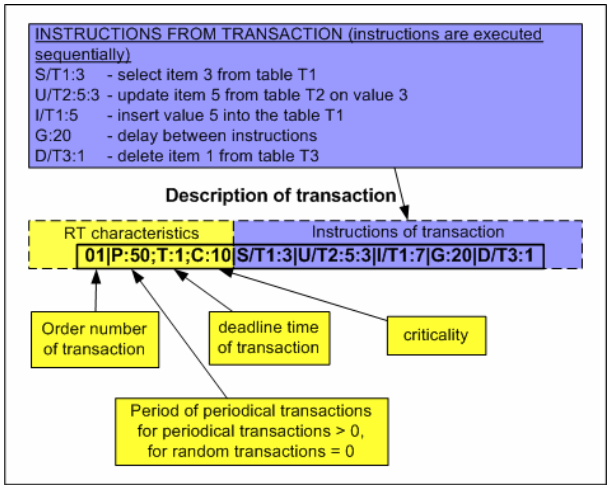


Fig.3.Description of the transactions

Logical database access results from physical database design. Access to the second record of table Tab01 can be written as Tab01:2, etc. Next four basic database access methods must be distinguished.

DB operation	shortcut
Select	S
Update	U
Insert	I
Delete	D

Table 1. Shortcuts of DB operations

For example, to select Tab01:2, it can be simply written as S/Tab01:2.

Besides these basic principles it is important to describe and work with some other, non-necessarily required parameters that further specify the transaction.

DB operation	shortcut
Deadline	T
Period	P
Criticality	C

Table 2. Shortcuts of transaction's RT characteristics

3 System test options

The system is implemented upon the real time operating system VxWorks. Currently it includes all necessary core database and transaction services, admission control, priority assignment and concurrency control. The way of operation of some system components can be changed according to project goals to enable testing the system behaviour under different conditions. The test options currently include:

3.1. Variable database definition and granularity

Database consists of tables defined by text lines in an external text file. Each table can be divided into predefined count of logical areas. Database schema is loaded during the system start.

3.2. Periodic and random transactions

Transactions are defined by simple text file. Each line represents the definition of one transaction as described above. The file is loaded before the initialization of the generators.

3.3. Priority assignment strategy

The priority assignment strategies make use of the RT characteristics of the transaction. There are four types implemented:

1) Deadline Monotonic (DM): Lower deadline = higher priority.

2) Most Criticality First (MCF): Higher criticality = higher priority.

3) Criticality Deadline First (CDF): Deadline-criticality 50-50 (%): Combination of 1and 2.

4) Random (RAND): uniformly generated random level of priority.

The priority assigned to a transaction is mapped to a real operating system process priority.

3.4. Transaction processing type

The way how the transactions are executed has certainly a significant impact on system performance. V4DB supports two types of transaction execution:

1) 1 transaction = 1 process: Each transaction is executed within its process. The process is created after transaction admission and destroyed after transaction commit. This processing type is used across all the experiments.

2) Process pool: The predefined count of processes executes the transactions. Each of the process executes transactions within the specified range of priorities. This processing type is currently under development.

3.5. Concurrency control mode

There are two types of pessimistic 2PL (two phase locking) and two types of optimistic protocols implemented, together with simple serial execution:

1) Strict 2PL (2PL-STRICT): Locking protocol. Hold all locks until the end of the transaction without any change of transaction priority.

2) 2PL Wait-Promote (2PL-WP): The scheme is identical to the basic 2PL in its resolution of conflicts. But with this mechanism, whenever a request is blocked behind a lower-priority lock holder, the lock holder's priority is promoted to that of the requester.

3) Optimistic locking - forward validation (OCC-FV): the transactions conflicted with the validating transaction are restarted.

4) Optimistic sacrified validation (OCC-SAC): If the validating transaction is in conflict with other transactions, it is restarted.

5) Strict serial (SERIAL): Transactions are executed in order of their admission. No transaction preemption can occur.

4 TEST ENVIRONMENT

During the process of testing it turned up how important detailed knowledge of the input of the transaction structure and overall description of the test environment are. So at first especially the target system, the used database structure, the way of transaction generation and performance metrics must be described.

4.1. Target system

The operating system VxWorks is supported on processor architecture Intel Pentium and that is why there was used a standard PC with an old Pentium2 processor as the target hardware. It is necessary to mention some essential settings of the operating system. There had to be priority pre-emptive task scheduling used. The time baseline was set to 100 microseconds which was sufficient with regard to the transaction process time. The system was rebooted before every experiment to achieve the same initial conditions.

Target system: PC, CPU Pentium2, 64MB RAM.

Operating system: VxWorks 5.5, kernel Wind 2.6.

4.2. The database

The database scheme was the same across all experiments. The database consists of 20 tables with 1000 records in each of them. The record size was also the same; every record had 4 bytes in length. Only granularity was different for various experiments conducted, its definition is therefore explicitly stated in the description of parameters used for particular experiments.

Database ~ 20 tables x 1000 records x 4 bytes

Granularity ~ from 1 to 100% (area size/database size)

4.3. Transactions

Input transaction setting is very variable in the created experimental system. The definition of every transaction consists of two parts: the first one contains transaction real-time properties while the second one contains required database operations with some system actions. Transactions can be generated periodically or randomly in predefined bounds.

4.4. RT characteristics

The RT characteristics were defined for all the experiments by the following way:

Trx0001 | L:20; H:80; D:10-30; C:1-10

Time range (parameters L, H)

It is obvious from the description that all the transactions are defined as random in some time range bounded by the parameters L and H. Within the time range (values are in milliseconds) the transactions are generated in a uniform distribution. It means that the transaction count within a time interval is known in advance. For the example stated above there is a lower time limit 20ms and the upper limit has a value of 80ms. The transaction generator defined by these limits generates 20 transactions in a second (1 transaction in 50 ms on an average). The transactions are generated by several generators to achieve some level of independence.

Deadline (parameter D)

Setting of the deadline parameter is also very crucial for experiment results. The deadline parameter D means the relative time from the system admission of the transaction to which the transaction must be processed to be concerned as successful. The parameter is set uniformly from some range defined by the values after the parameter D. The parameters do not mean the exact time values but rather the multiples of the "average" transaction process time which was found out experimentally. The same principle was used in [4]. The parameter is stated as the "deadline factor"in experiment descriptions.

Deadline factor ~ [3, 8]

Deadline ~ [3*avg_process_time, 8*avg_process_time]

Criticality (parameter C)

The last RT parameter means the transaction importance against the others in the system. The parameter was in all experiments randomly generated in a uniform way within the range [1, 10].

Criticality C ~ [1,10] (1 - the highest)

4.5. Database operations

The required database operations are defined in the second part of the transaction definition. There were used only the SELECT and UPDATE operations for simplicity. The following line describes the way of the text definition of used database operations:

S/?# U/?# U/?# S/?# S/?# U/?#...

Using the wildcards "?" inserted to the definition file in the place of tables and records was very useful. Since the real tables and records were chosen uniformly when using such wildcards, uniform database access was ensured.

It was also necessary to set the count of database actions in the transaction. Various settings were tested during experiments. Finally the value of 20 was established. It means that every transaction consisted of 20 database operations.

4.6. Update ratio (WR)

The update ratio means the ratio of UPDATE operations comparing to all operations performed. The ratio can be simply set up by the count of UPDATE operations within the transaction definition. The parameter was mostly set to 50%. The parameter is marked as WR.

$$WR \sim [0, 100] \quad [\%]$$

4.7. Evaluation metrics

The following chapter describes the indicators used as the evaluation criteria of performed experiments.

Deadline guarantee ratio [%]

Deadline guarantee ratio as the percentage of transactions that complete by their deadline:

Abort ratio [%]

The percentage of aborted transactions

Restart ratio []

The total value of restarted transactions divided by the total count of transactions. Every transaction can be restarted more than once.

Block ratio []

The total value of blocked transactions divided by the total count of transactions. Every transaction can be restarted more than once.

Every presented experiment was tested for 5000 input transactions.

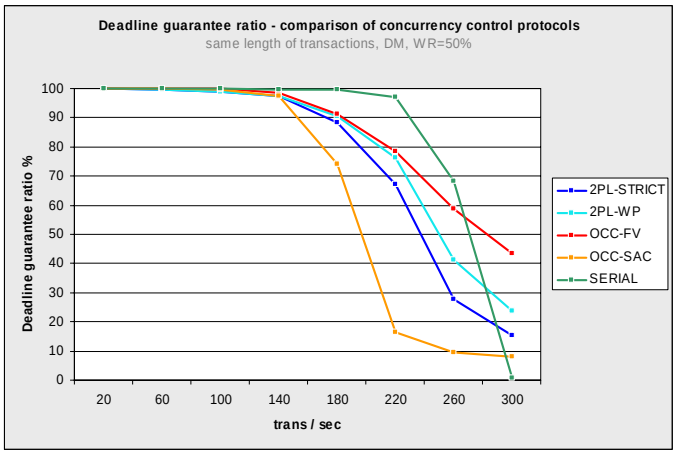
5 Experimental results

There were performed many experiments related to each of evaluation criteria mentioned. In this paper there are presented results of two experiments.

5.1. Comparison of concurrency control protocols depending on the transaction arrival rate

Parameter settings

Parameter	Value
Granularity	5%
Transaction arrival rate	20-300 trans/sec, same length
Count of DB operations	20 (within one transaction)
Deadline factor	3-8, uniform distribution
Update ratio (WR)	50%
Priority assignment	DM
Concurrency control	2PL-STRICT, 2PL-WP, OCC-FV, OCC-SAC, SERIAL



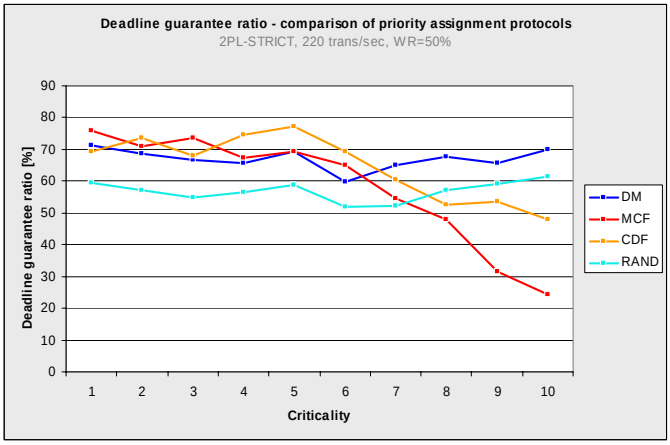
Comments:

The best results are achieved for serial execution of transactions. It is surprising but there are many reasons how to explain it. Serial transaction execution provides the best results because the database is completely in the main memory, thus the blocking time is very low, the size of logical areas can be very high, during serial execution the whole database is one big area. The next reason is the fact that the transactions are of the same average length so there is no blocking of short transactions by the long ones. In addition there is no system overhead during the transaction pre-emption or administration of the queues. This phenomenon was described in [13] and it can be stated that under some specific circumstances the serial transaction execution achieves the best results. We performed other tests with transactions in a different length and the results were really quite different.

5.2. Comparison of priority assignment strategies depending on criticality

Parameter settings

Parameter	Value
Granularity	5 %
Transaction arrival rate	220 trans/sec, same length
Count of DB operations	20 (within one transaction)
Deadline factor	[3-8], uniform distribution
Update ratio WR	50%
Criticality	[1-10] (1 = the highest), uniform distribution
Priority assignment	DM, MCF, CDF, RAND
Concurrency control	2PL-STRICT



Comments:

At first glance it can be seen that the protocol MCF (Most Criticality First) has the expected crucial impact on the transaction deadline guarantee ratio. It is especially noticeable for levels 8-10 where the guarantee ratio is the lowest in comparison to other protocols. The value of 10 is the lowest level of criticality. The combined protocol CDF (Criticality Deadline First) has the similar behaviour but not so strong because besides the value of criticality it takes into account also the value of the priority for decision making. As for the other protocols, the deadline guarantee ratio oscillates within the small value range regardless of the value of criticality. It is logical because the other protocols do not take into account the criticality value.

6 Conclusion

We have developed an experimental real-time database system suitable for study of real-time transaction processing. Since the implementation process is very difficult, it is impossible to include all the functional parts of the real RTDBS. It is intended to create the system in such way that it would be possible to extend it and to continue its development later. Future directions can be suggested right now: variable database granularity, indexing algorithms, backup possibilities or database movement to a disk media. It is assumed that the created system will be used as the experimental basis for future research and for future advancement to study another, more complex properties of RT databases.

Acknowledgement

The support for this research work has been provided by the project 102/06/1742: Experimental real-time database testing system and the project 102/05/H525: The Postgraduate Study Rationalization at Faculty of Electrical Engineering and Computer Science VSB-TU Ostrava, both provided by the Czech Science Foundation.

References:

[1] Kam-Yiu Lam, Tei-Wei Kuo, *Real-time database systems: Architecture and Techniques,* Kluwer Academic Publishers, 2001.

[2] Krishna C. M., Kang G. Shin, *Real-Time Systems,* McGraw-Hill, New York, 1997.

[3] Aldarmi S.A., *Real-Time Database Systems : Concepts and Design,* The University of York, 1998.

[4] Huang J., *Real-Time Transaction Processing : Design, Implementation, and Performance Evaluation*, PhD thesis, University of Massachusetts, 1991.

[5] Matthew R. Lehr, Young-Kuk Kim, Sang H. Son, Managing Contention and Timing Constraints in a Real-Time Database System, *16th IEEE Real-Time Systems Symposium,* Pisa, Italy, 1995.

[6] Taina J., Son S.H., *Towards a General Real-Time Database Simulator Software Library,* University Of Virginia, 1999.

[7] Kim S., Son S.H., Stankovic J.A., Performance Evaluation on a Real-Time Database, *Proc. IEEE Real-Time Technology and Applications Symp.*, 2002.

[8] Sivasankaran R., *Design of RADEx : Real-Time Active Database Experimental System,* Technical report, University of Massachusetts, 1994.

[9] Krol V., Pokorny J., Cernohorsky J., Towards the evaluation of algorithms used in real-time databases, *Proceedings of the 7th WSEAS Int .Conf. on Automatic Control, Modelling and Simulation,* Prague, Czech Republic, 2005.

[10] Krol V., Pokorny J., Design of V4DB – Experimental Real-Time Database System, *Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society***,** Paris, France, 2006.

[11] Lee J. : *Concurrency Control Algorithms for Real-Time Database Systems.* PhD thesis. University of Virginia. 1994.

[12] Kao B., Garcia-Molina H., An Overview of Real-Time Database Systems, *Advances in Real-Time Systems*, Prentice Hall, ISBN 0-13-083348-7, pp. 463-486, 1995.