International Journal of the Physical Sciences Vol. 6(7), pp. 1677-1684, 4 April, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.125
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

An improved 2PL-HP based concurrency control algorithm in real time database

Saeed Bahrami[1]*, Mohamadreza Shahrokhzadeh[2] and Seyyed Mohsen Mirhoseini[3]

[1]Department of Computer Engineering, Faculty of Engineering and Technology, Islamic Azad University, Branch of Abhar, Abhar City, Zanjan, Iran.
[2]Faculty of Electrical, Computer and IT Engineering, Islamic Azad University, Branch of Qazvin, Qazvin, Iran.
[3]Faculty of Computer, Islamic Azad University, Branch of Hidaj, Hidaj City, Zanjan, Iran.

Accepted 21 February, 2011

In the real time database management systems the mechanism of concurrency control 2PL-high priority, by applying a sacrificing approach of the low-priority transaction during the competition with high priority transaction, not only waste the resources of the system but also decrease the system efficiency due to the increase of the transaction failure because of miss deadline. The survey is focusing on the weak points of this mechanism and presenting a new approach that by omitting the unnecessary restarts greatly improve the efficiency of the system. We also present a new approach for determining the transaction's priority to provide the possibility of more successful transaction. This approach will be more prominent especially when there is more competition among the transactions for locking the resources of the system. By modeling the behavior system in Markov model, this increase in efficiency of the system comparing to the 2PL-HP mechanism has been improved.

Key words: Real time databases, concurrency control, transaction priority, performance evaluation.

INTRODUCTION

Regarding the type of application of real time databases management systems, there are considerable limitations and differences in the structure of this type of databases in comparison to conventional one. There is a serious time limit governing in real time databases. As a result real time transactions bear a determined performance due time in compliance with the importance rate of performance and would be divided into three groups of Hard, Soft and Firm. In case of any lack of performing the transactions of the first group at specified due time, there will be a serious danger to the system. But it is only the reduction of system efficiency in transactions to second and third transactions. The other difference of these two systems is that data in a management system of traditional database will be changed only through the transactions of the users, while it is necessary in such databases to have a complete compliance with environment since it is necessary to have relevant data that always reflect the realities of the environment. As a result since the transactions have time limits, we have only valid data in one time limit. Since the efficiency of database management system has a deep effect on applied mechanisms for concurrency control, this item was the center of focus of researchers simultaneous with creation of databases. The real duty of concurrency control mechanism is creating a complete adaptation with database at the time of performing a collection of transactions [Abu, 2006]. There are various mechanisms in traditional and common databases for concurrency controls. But regarding the mentioned basic differences, any application of these mechanisms in any real time databases will cause a serious reduction of its output. As a result there are a lot of new efforts for innovation of compliance methods with such a database in recent years. Generally it is possible to classify all efforts done into three optimistic classes as (Lindström, 2003; Squadrito, 1996; Abu, 2006; Barbosa, 2007; Sha et al., 1990; Lindström, 2000) pessimistic classes (Lindström, 2003; Braoudakis, 1995; Barbosa, 2007; Aldarmi, 1998)

*Corresponding author. E-mail: saeed.bahrami@abhariau.ac.ir. Tel: +989121828653.

and hybrid one (Agrawal et al., 1995). Due to the lack of providing a dead end and non-blocking property, optimistic class has a suitable situation for applying this type of databases but as a result of any delay in solving the conflicts among transactions up to the end of the minimum one of involved transactions may cause a reduction of efficiency and output of resources.

In contrast, pessimistic class has 2PL as the base of its work with some changes in it and trying for compliance of this mechanism with real time databases. Among all innovated algorithms in this class we have 2PL high priority as the most attractive one due to the easy implementation and lack of dead end and on time contrasts. By the way any removing policy of low-priority transactions with high-priority one may not only waste the system resources but increase the failure rate of transactions due to the delay in performance along with efficiency reduction of system. In order to increase the abilities and get more benefits from 2PL-HP, in this study we have proposed a method by which it is possible to focus on the number of restarts out of any contrast as a major weak point of mentioned mechanism. We could reduce considerably the number of these non-necessary restarts by applying some changes in the structure of this mechanism with the same rate of system efficiency. Then after a brief consideration of algorithms based upon locking, we have considered the operation manner of the proposed method. Then by modeling and mathematical proofs, we have evaluated the proposed mechanism along with a submission of the results at the end. According to the results, it is obvious that to improve the operation is so much considerable especially when the number of current transactions is more than total resources and as a result, it is more probable for arising any conflicts among them.

RELEVANT ALGORITHMS ON LOCKING FOR CONCURRENCY

Control in real time databases 2PL algorithm is the base of relevant mechanisms depending on the locking. A transaction in standard 2PL algorithm should obtain relevant reading (writing) lock before any reading (writing) from (on) a database. Then, writing lock is exclusive type and in case of any allocation to a transaction, it is necessary to release other application transaction up to the releasing time of the mentioned transaction. In case of any application of such algorithm for a concurrency control in a real time database management system, it is possible to have a phenomenon as "Priorities inversion", that is a high-priority transaction which may be blocked by a low-priority transaction. Further to have a probable occurrence a dead end in this algorithm, it may lead to a limitless waiting of transactions in a process which is so much dangerous in this type of database systems. The above mentioned problems are the base of relevant efforts applied up to now for optimization of 2PL algorithm and further application in a real time database management system. In this part we will point out to some of the most important cases. This is necessary to mention that it is important to consider this item base on the proposed method.

2PL-High priority

In case of any arising conflicts among different transactions against ownership of a resource, this algorithm is more related to the high-priority transactions. According to the function of this algorithm, if the priority of the applicant transaction is more than the transaction priority of the lock owner, there will be an end for the low-priority transaction and it is necessary to be restarted. Therefore, the lock is released and will be ready for high –priority transaction. But if we have a contrast scenario, the applicant transaction is waiting up to the end of transaction performance of the lock owner. One of the most important advantages of this method is solving the problem of "Priorities inversion" and removing any dead end conditions and it is easier to implement than operational overhead out of restart of low-priority transactions and "Useless restart "phenomenon with further reduction of system output. It is necessary to add that any useless restart means a transaction that may be stopped by a high-priority transaction which the same high-priority transaction will be omitted in future due to some similar reasons and before the completion.

Priority inheritance

This protocol is basically innovated by the aim of solving the "Priorities inversion" problem. According to this protocol, when the applicant of resource has a higher priority, the owner of the required resources with low-priority may obtain the requested transaction. Any increase in the priority level of current transaction may cause this transaction to find more required resources and this makes it difficult to block it by middle priority transactions and as a result it will finish sooner and may cause all available locks. In contrast with previous method (2PL-HP), in this method the transaction with higher priority may cause semi-finished situation of low-priority transaction and result in overhead increase of system operations not only make it possible to work the current transaction but may increase the level of priority and real time performance of it. In other words, the type of thinking of this protocol is winner -winner (Abu, 2006), but there are some defects with this method. The followings are the two most important items:

(i) Any transaction with higher priority may be blocked in the worst condition by all under-process transactions (Abu, 2006).

(ii) Since by increasing the competition rate there will be an equal situation for transactions priority, this protocol may lead the system towards a deadline (Sha et al., 1990).

Priority ceiling

The major goal of this protocol is to solve the problem of deadline and "Useless restarts "phenomenon which may present in the last two algorithms. In this protocol we have all resources with a priority ceiling equal to the transaction priority with the highest priority of future. As a result, either transactions and /or resources have specified priority before any performance. The idea of this protocol is the ability of a new transaction with higher priority to block an under-process transaction even and only even its priority is higher than the priority level of all sources which have been blocked by an under-process transaction. If this condition is not available, the new transaction will wait and obtain the highest priority among all stopped transactions and may continue the work up to the end and finally will be finished by releasing the locks. Then all stopped transactions will awake and one of them will start its work with the highest rate of priority. This reality shows that the priority of new transaction should be higher than the priority level of all data used by the current transaction may prevent from any occurrence of dead end. Also this reality that a new transaction with a higher priority would be blocked maximum with a time limit for performing a transaction with a lower priority level will prevent from any occurrence of the "Useless restarts" phenomenon. Of course there are some defects with this method. The most important of it is difficult (and /or impossible) situation for determining the required resources before its performance which may threaten any implement of this algorithm. Any lack of differentiation between reading and writing on data resources is another factor of reducing its efficiency which has been removed in the next versions.

Ordered sharing

By creating a sequence between the required locks in concurrency transactions, this algorithm will provide a new method for reducing the blocking rate of competitor transactions. In contrast with other methods based upon locking, there is no need in this method to keep the mutual exclusivity after the end of functions on a resource. It is possible to provide this resource for other applicant transactions. But in order to prevent from database adaptation, it is necessary to store the sequence of functions on the considered resource applied by the concurrency transactions. Assume the transaction of T_H requests a resource which has been locked in prior by T_R transaction. It is possible for T_R to benefit from this source when:

(i) All functions should be applied on the considered resource after completion of functions of T_R transaction.
(ii) T_H transaction may only be finished when we have the completion of T_R transaction before it. Otherwise, T_H will be aborted.

On the other hand, any reduction of blocking of competitor transactions as a major priority and any probable of occurrence the "useless restarts" phenomenon are the major problems of this algorithm (Agrawal et al., 1995). As it was described, all mentioned algorithms bear some weak points in addition to strong points. Like priority ceiling, some of them are only presentable as a theory due to the complex situation of it and the others are more welcome in spite of bearing some defects and due to the easy implementation. 2PL-high priority mechanism belongs to this group. In the following we have presented a method for reducing the weak points of this mechanism and increasing its advantages.

PROPOSED METHOD

As it was mentioned in introduction and relevant algorithms on locking for concurrency, the operational overhead out of restart of low-priority transactions and "Useless restarts phenomenon" are the major problems of 2PL-HP method. But on the other hand, there are some advantages for it such as easy implementation and lack of any dead end. Our proposed method has focused on two mentioned problems in 2PL-HP algorithm. Applying a different policy from one side at the time of facing on a transaction with high priority transactions reducing the number of restarts and obtained overhead and promoting the priority level of current transactions in compliance with number of transactions blocked by it on the other. Then there is a priority for successful completion of transaction and further reduction of blocked transactions in system. For more consideration of proposed method, assume the situation of transaction in each moment with pair (n, X) where the item X is related to transaction situation and may obtain B (Block) and/or P(Process) resulted in n for n^{th} required resource. Therefore, the pair (n, B) means that the transaction has been blocked after the n^{th} data resource and then pair (n, P) means a transaction process on n^{th} of data resource. Any transfer from one situation to the other is specified with an arrow sign. The situation of (0) is a sign of start point of the transaction. Assume that needs a data resource of d through its lifetime while it could obtain n resources of it up to now. In the proposed mode, if a transaction requests for a resource with higher priority (for example n)which has been locked at present by our considered transaction, it is only enough to backward to the previous closed situation (that is n, B) (Figure 1). It seems to take a snapshot from transaction condition through requesting the resource for encoding the transaction again.

It is necessary to restart this transaction in case of any compliance with 2PL-HP algorithm. By this policy and in addition to an increase in utilization of system resources, there will be an increase in completion opportunity of the transaction before the end of its performance due time. In case of any compliance with 2PL-HP algorithm. By this policy and in addition to an increase in utilization of system resources, there will be an increase in completion opportunity of the transaction before the end of its performance due time. On the other hand, it has been proved that the best method for specifying the transaction priority in real time databases is EDF method (Agrawal et al., 1995) according which we have closer opportunity for performing the transaction that makes more priority for its allocation. In order to increase the opportunity of successful

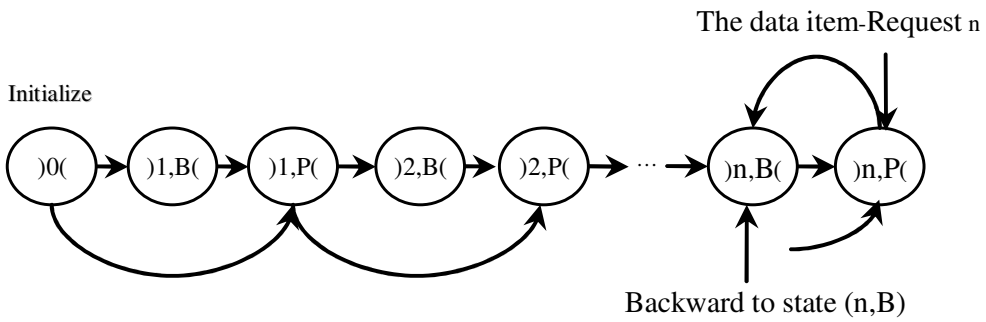

Figure 1. Transaction situation after requesting for a source with a condition for applying proposed method.

completion of transaction in our proposed method we should develop EDF and upgrade its priority in compliance with number of blocked transactions by the considered transaction. As a result it is possible to calculate the transaction priority rate, T_1, through the following formula:

$$P(T_i) \begin{cases} ((\min[\sum_{b=1}^{n} \frac{1}{D_b}, X])+1)*\frac{1}{D_i} & if\ n > 0 \\ \frac{1}{D_i} & otherwise \end{cases} \quad (1)$$

Where n is the number of blocked transactions, D_b is related to performance due time of these transaction and D_1 is the performance due time of T_1 transaction. Any specification of X is the result from the combination of the blocked transactions and the close end transactions.

While we consider a great amount of X, it is possible to have a great amount of considered transaction for further blocking of high-priority transactions near the final moment, as a result we will be far from our goal which is reducing the number of blocked transactions. Then we consider X equal to 1. Therefore, according to the above-mentioned formula, the number of blocked transactions may increase at most the priority of the transaction to two times more. Figure 2 is a pseudo code of proposed method for processing a presented transaction. It is necessary to note that there is an abortion of transaction only when it reaches to its end of performance due time. It is possible to prove that by applying the above-mentioned policies in 2PL-HP mechanism, there will be a considerable increase in its output. For this purpose, we will evaluate its operation b to apply the proposed mechanism and changing system behavior into a mathematical model.

Performance evaluation

In order to evaluate the output of a proposed algorithm, at first we should make a model of system behavior by Markov model and then evaluate its output in accordance with obtained model. The correctness of the mentioned output evaluation method for concurrency control mechanisms has been proved in real time databases. The obtained results have the same value resulted from simulation.

In this part, we will at first specify the governing situations on the problem and required parameters for evaluation of the comparison between proposed and basic methods (2PL-HP).

```
1 Transaction_ Process(T){
2       Init(T);        */Initialize transact       ion T   /*
3       i  =0;
4 while i < d or deadline(T)is not expired do
5  */       d presents  items that need to process
/*
6 if  item(i)is locked with T' then
7 if )priority(T)> priority(T'))
8                   or (priority(T)       = priority(T')and
9                   Lifetime(T)> lifetime(T')          (then
10                   {
11                       Backward(T')to state (i,B');
12                       Lock(item(i))with T;
13                             Process(item(i));          */go to
state (i,P)   /*
14                   }
15 elseif  )priority(T)< priority(T')        (
16                    or (priority(T)       = priority(T')and
17                                                Lifetime(T)<
lifetime(T')    (then
18                        Block(T)to state (i,B);
19                  */  go to state (i,B)      /*
20              endi  f
21 else
22       {
23                    Lock(item(i));
24                    Process(item(i);
25                  }
26 endif
27       i++;
28 end while;
29   Release all locks;
30   Terminate(T)*/Commit or Abort                /*31 }
```

Figure 2. pseudo code of proposed method.

Table 1. Required parameters for evaluation of proposed model.

D	Total number of current data resources in database.
d	Number of resources provided by transactions.
t	Number of processed transactions at any time by the system.
μ_0	Creation rate and primary rate of the transaction.
μ_p	Processing rate of transaction.
μ_b	Transaction rate in closing situation.
P_b	Probable situation for blocking of transaction.
$P_{a(i,B)}$	Probable situation for aborting of transaction while in (i,B) position.
$P_{a(i,P)}$	Probable situation for aborting of transaction while in (i,P) position.
$P_{bk(i,j,P)}$	Probable backward of transaction to (i,B)position in one of processes conditions of (P).
$P_{bb(i,j,B)}$	Probable backward of transaction to (i,B)position while it is in one of the blocked (B) positions.

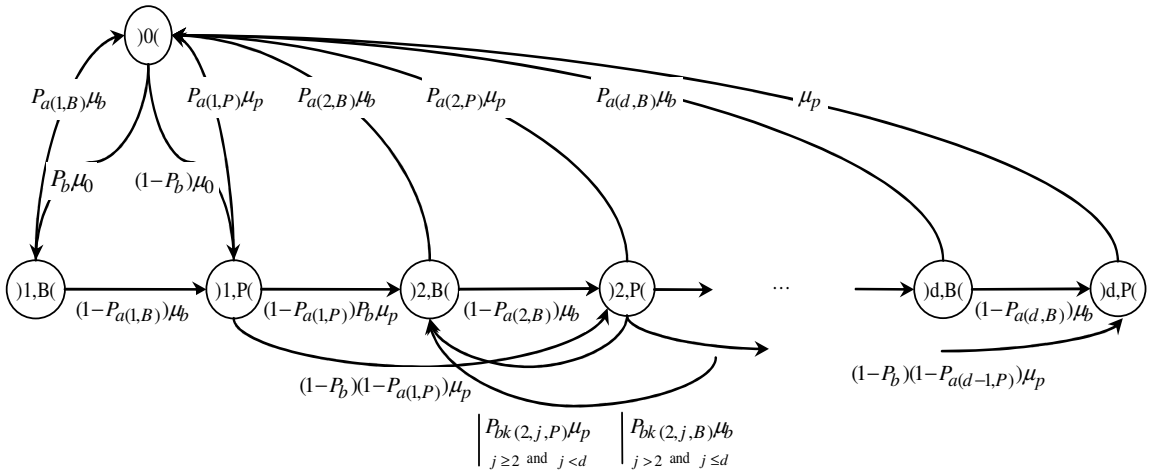

Figure 3. Diagram of different conditions and movements of a transaction.

Scenario and evaluation parameters

In order to make a model of proposed method, at first we should describe the theories and introduce the required parameters. The considered database is an independent one benefiting from a CPU for processing of transactions. In addition, it has been assumed that:

(i) A transaction will have access to the same number of data resources which would be specified by parameter D.
(ii) The locks have been assumed as exclusive type.
(iii) An accessible source by transactions would be accessible by total one and there is only one copy of it.

We benefited from Markov model for evaluating the proposed algorithm. This model making method is working according to the relevant probabilities between work situations. As a result, we need some parameters for expressing the further probable situations. Permitted delay against performance due time is one of the parameters mentioned which could be specified by symbol S. Table 1 shows the other required parameters along with relevant explanation.

Transaction model

Figure 3 shows any model of each transaction with 2d+1 and any relations among them. There are some hidden points in this figure. Firstly a transaction will be completed successfully when it is in (d, p) position. Therefore, any probable presence in this position is the evaluation criteria for further studies. Then we assume that if a transaction reaches the (d, p) position, certainly it will end successfully (p_a (n_1 p)= 0). Finally if a high-priority requests data resource no. 1, the current transaction would be aborted instead of referring to (1, B) situation. The following equations shows total relations that is required for calculation of P (d, p) as an evaluation criterion.

$$P(1,B) = (\mu_0/\mu_b)P_bP(0)$$

$$P(1,P) = (\mu_0/\mu_p)(1-P_bP_{a(1,B)})P(0)$$

$$P(2,B) = (\mu_0/\mu_b)P_b(1-P_{a(1,P)})(1-P_bP_{a(1,B)})P(0)+(\mu_p/\mu_b)P_{bk(2,j,P)}+P_{bk(2,j,B)}$$

$$P(2,P) = (\mu_0/\mu_p)(1-P_bP_{a(1,B)})(1-P_{a(1,P)})(1-P_bP_{a(2,B)})P(0)+(1-P_{a(2,B)})[P_{bk(2,j,P)}+(\mu_b/\mu_p)P_bP_{bk(2,j,B)}]$$

$$P(3,B) =$$

$$\begin{cases} (\mu_0/\mu_b)P_b(1-P_{a(1,P)})(1-P_bP_{a(1,B)})(1-P_{a(2,P)})(1-P_bP_{a(2,B)})P(0)+ \\ P_b(1-P_{a(2,P)})(1-P_{a(2,B)})[(\mu_p/\mu_b)P_{bk(2,j,P)}+P_{bk(2,j,B)}]+(\mu_p/\mu_b)P_{bk(3,j,P)}+P_{bk(3,j,B)} \end{cases}$$

$P(3,P)$ =

$$\begin{cases} (\mu_0/\mu_p)(1-P_bP_{a(1,B)})(1-P_{a(1,P)})(1-P_bP_{a(2,B)})(1-P_{a(2,P)})(1-P_bP_{a(3,B)})P(0)+ \\ (1-P_bP_{a(3,B)})(1-P_{a(2,P)})(1-P_{a(2,B)})[P_{bk(2,j,P)}+(\mu_b/\mu_p)P_{bk(2,j,B)}]+ \\ (1-P_{a(3,B)})[P_{bk(3,j,P)}+(\mu_b/\mu_p)P_{bk(3,j,B)}] \end{cases}$$

$P(d-1,B)$ =

$$\begin{cases} (\mu_0/\mu_b)P_b[\prod_{i=1}^{d-2}(1-P_{a(i,P)})(1-P_bP_{a(i,B)})]P(0)+ \\ \sum_{i=2}^{d-1}[[\prod_{k=i}^{d-2}(1-P_{a(k,P)})(1-P_{a(k,B)})][(P_{bk(i,j,B)}+(\mu_p/\mu_b)P_{bk(i,j,P)}]] \end{cases}$$

$P(d-1,P)$ =

$$\begin{cases} (\mu_0/\mu_p)(1-P_bP_{a(d-1,P)})[\prod_{i=1}^{d-2}(1-P_{a(i,P)})(1-P_bP_{a(i,B)})]P(0)+ \\ \sum_{i=2}^{d-1}[[\prod_{k=i}^{d-2}(1-P_bP_{a(k+1,B)})(1-P_{a(k,P)})][(1-P_{a(i,j,B)})[((\mu_p/\mu_b)P_{bk(i,j,B)}+P_{bk(i,j,P)}]] \end{cases}$$

$$P(d,B) \quad (\mu_p/\mu_b)(1-P_{a(d-1,P)})P_bP(d-1,P) \tag{2}$$

$$P(d-1,P) = (\mu_b/\mu_p)(1-P_{a(d,B)})P(d,B)+(1-P_b)(1-P_{a(d-1,P)})P(d-1,P) \tag{3}$$

$$P(0)+\sum_{i=1}^{d}\left[P(i,B)+P(i,P)\right]=1 \tag{4}$$

At first it is necessary to calculate μ_0, μ_p ,μ_B,P_b,$P_{a(i,B)}$, $P_{bb(i,P)}$, $P_{bk(i,B)}$, $P_{a(i,P)}$, then by applying (1) and (2) in Equation (3) it is possible to obtain relevant amount of P(0). Finally the amount of P (d, p) will be resulted which is equal to success probable of a transaction. Then we will point out to calculation manner of the mentioned parameters.

Variants $1/\mu_0$ and $1/\mu_p$

The required average time for processing a data resource or $1/\mu_p$ is equal to the average time of primary quantity of transaction ($1/\mu_0$) which would be assumed equal to 10 m/s. Since we are trying to compare any obtained results of proposed method with obtained results of 2PL-HP method, then it is enough for both variations to have equal amounts in both evaluations.

Variant P_b

P_b is the probable situation of blocking of transaction due to the locked of required resource by high-priority transaction. By assuming the independence of this probable situation, it is possible to calculate it as follows:

$$P_b = \frac{ALL\_LOCKS}{D} \tag{5}$$

Where All Locks is equal to the total number of locks obtained by high priority transaction at database. The average number of high-priority transactions is equal to: $(t-1)/2 = (0+1+2+...+(t-1))/t$

If we assume that all above-mentioned transactions may averagely have L amount of locks, then we will have:

$$ALL\_LOCKS = \frac{(t-1)*L}{2} \tag{6}$$

Then we can calculate L amount through the following formula:

$$L=\sum_{i=1}^{d}[(i-1)P(i,B)+iP(i,p)] \tag{7}$$

Therefore:

$$P_b = \frac{(t-1)*L}{2D} \tag{8}$$

Variant $1/\mu_b$

$1/\mu_b$ is the average waiting time of transaction up to the end. The manner of calculation of recent parameter has been presented in (Dogdu and Ozsoyo˘glu, 1997) but according to the obtained experiences out of continuous calculations, we can assume it equal to $d/2\mu_p$.

Variant P_a (i, P)

P_a (i, P) it is equal to any probable abortion of transaction out of expiration of performance due time in (i, P) situation. In order to calculate this variation, it is possible to use this method as follows:

First of all we assume that performance due time of a transaction is equal to the number of resources used by it. The variation $P_{a\ (i,\ P)}$ is resulted from dividing the lifetime of transaction in (i, P) situation on required time for processing of transaction. Following formula is for calculation of lifetime of a transaction:

$$AGE_{(i,P)} = 1\mu/_0 + i(P_b\ 1\mu/_b+1\mu/_P) \tag{9}$$

The required time for processing of transaction is obtained from the following formula:

$$P_{a(i,P)} = \frac{AGE_{(i,P)}}{ST} \tag{10}$$

Variant $P_{a\ (1,\ B)}$

$P_{a\ (1,\ B)}$ is similarly equal to abortion probability of transaction due to the expiration of performance due time while it is in position (i, B) and would be calculated like formula (10). But there is a difference in lifetime of the transaction as follows:

$$AGE_{(i,B)} = 1\mu/_0 + (i-1))P_b\ 1\mu/_b+1\mu/_P) + 1\mu/_b \tag{11}$$

Variant $P_{bk\ (I,\ J,\ p)}$

$P_{bk\ (I,J,p)}$ is equal to any backward probable of under-processing transaction out of any contrast with high-priority transaction referring from (j,P) situation to (i,B) which is:j={i, i+1,…,d}.

Any probable refer of transaction from (j, P) to (i, B) is as follows:

Table 2. Successful performance rate of transaction in both proposed method and 2PL-HP

T	d	Our approach	2PL-HP	Our approach	2PL-HP
5	5	12.21	11.62	11.85	11.74
	7	7.84	7.29	7.68	7.57
	9	5.2	4.78	5.22	5.13
	11	3.54	3.24	3.69	3.62
	13	2.46	2.24	2.66	2.61
	15	1.72	1.57	1.96	1.92
25	5	12.54	10.45	12.08	11.73
	7	7.24	5.79	7.73	7.40
	9	4.14	3.25	5.22	4.94
	11	2.42	1.81	3.59	3.40
	13	1.26	0.98	2.54	2.40
	15	0.68	0.52	1.81	1.71

$$\frac{1}{D} * \left[\frac{(t-1)}{2(P_b(1/\mu_b)+1/\mu_p)} \right] * \frac{1}{\mu_p} \quad (12)$$

Where its first term is referring to the considered resource rate of total resources.

Variant $P_{bk\,(i,\,j,\,B)}$

$P_{bk(i,j,B)}$ or the backward probable transaction in blocked situation (j,B) will be calculated like $P_{bk(i,j,B)}$. But there is a difference that we should put $1/\mu_b$ instead of $1/\mu_P$:

$$P_{bk(i,j,B)} = \sum_{j=i}^{d-1} \frac{\mu_p}{P_b \mu_b + \mu_p} * \frac{(t-1)}{2D} \quad (13)$$

Evaluation criteria

We want to obtain the rate of transactions which have been completed before the relevant due time. As obvious in Figure 2, a transaction that may reach to (d,P) situation was successful in passing all threatening obstacles and limitations. As a result the successful completion rate could be calculated by the following formula:

$$\text{Commit Rate} = P(d,P)\ \mu_p \quad (14)$$

This is necessary to mention that we considered the amounts of S and L respectively as 5 and d/2.

EVALUATION RESULTS

After considering nominal amounts in obtained relations, we compared the proposed method with 2PL-HP one. Table 2 shows the obtained results for databases with 1000 and 10000 data resources. We repeated the test on this database once with 5 transactions and then for 25 ones. As it is obviously mentioned in Table 2, if there is a great rate of required resources of under-process transactions to total current resources in database, and since there is an increase in any probable arising of contrast, our proposed method makes a considerable betterment in 2PL-HP method. In other words, more contrasts between the transactions will increase the efficiency of 2PL-HP method.

Conclusion

As was mentioned in this essay, there are some weak/power points for concurrency control algorithm 2PL-HP in real time databases. This is because the low-priority transaction may abort any conflict challenge and as a result may cause wasting of resources. In addition, this may cause any arising of "Useless restarts " phenomenon. But some of the major attractions of this method are solving the problem of "Priority inversion", lack of dead end conditions and also easy implementation. Our proposed method focused on two mentioned problems in 2PL-HP algorithm. On the one hand it may omit any useless restarts and resulted overhead by applying the backward policy at the time of facing the considered transaction with high-priority one and on the other hand it may increase the opportunity of successful completion of transaction by upgrading the priority level of current transaction in compliance with number of transactions blocked by it. As a result there is a reduction from number of blocked transactions. We had model making of system behavior in Markov model for evaluation of proposed algorithm efficiency. The results of this evaluation show a considerable increase in efficiency of proposed method against 2PL-HP which may cause an increase of competition between system transactions and efficiency of this method more than before.

REFERENCES

Abu AA (2006). On Optimistic Concurrency Control for Real time Database Systems. Am. J. Appl. Sci., 3(2): 1706-1710.

Agrawal D, Abbadi A, Jeffers R, Lin L (1995). "Ordered Shared Locks for Real Time Databases. FLDB J., 4: 87-126.

Aldarmi SA (1998). Real-Time Database System: Concepts and Design, YCS 303. Department of Computer Science, University of York.

Barbosa R (2007). An Essay on Real time Databases. SE-41296, Chalmers University of Technology, Goteborg, Sweden.

Braoudakis S (1995). Concurrency Control Protocols for Real time Databases. Ph. D Thesis, and University of Boston, USA.

Dogdu E, ¨Ozsoyo˘glu G (1997). Real time Transactions with Execution Histories: Priority Assignment and Load Control. Sixth International Conference on Information and Knowledge Management, pp. 301–308.

Lindström J (2000). Using Optimistic Concurrency Control in Firm Real time Databases. Next Millennium Computing Conference, Hanko, Finland.

Lindström J (2003). Optimistic Concurrency Control Methods for Real time Databases. Ph. D Thesis, University of Helsinki, Finland.

Sha L, Rajkumar R, Lehoczky JP (1990). "Priority Inheritance Protocols : An Approach to Real time Synchronization." IEEE Trans. Comput., 39(9): 1175 -1185.

Squadrito MA (1996). Extending the Priority Ceiling Protocol Using Read/Write Affected Sets. Submitted Master Thesis, University Of Rhode, Island.