Malaysian Journal of Computer Science, Vol. 10 No. 1, June 1997, pp. 64-74

A METHOD OF ESTIMATING ABORTED TRANSACTION IN THE DATABASE CONCURRENCY CONTROL SYSTEM

Mustafa Mat Deris
Faculty of Applied Science and Technology
University Putra MalaysiaTerengganu
21030 Mengabang Telipot
Terengganu, Malaysia
Tel.: 09-6696412,
Fax: 09-6696694

Ali Mamat
Department of Computer Science
University Putra Malaysia
43400 UPM Serdang
Malaysia
Tel.: 03-9486101
email: ali@fsas.upm.edu.my

ABSTRACT

Transactions may be aborted when they are unable to obtain a lock on a required data item. Estimating the proportion of transaction that aborts is one of the key issues in modelling a system which affect the performance measures of interest such as average response time and the throughput capacity of the system. This paper shows a method of estimating aborted transaction and performs a comparative study with other method given by Mitrani et al.

Keywords: ***aborted transaction, service demand, average response time, throughput***.

1.0 INTRODUCTION

The database environment can be described in terms of three components: the database itself, the transactions that act upon the data, and the computer system that supports the processing activities.

In a multi-user environment, many users will wish to access and update the database concurrently. Problems may arise if this concurrency is undisciplined, i.e. data may be lost due to uncontrolled updates and inconsistent retrievals may take place. Therefore, concurrency control is needed when transactions are executed concurrently to guarantee that the integrity and consistency of the database.

Various methods have been proposed to overcome these problems. The most popular method is the locking technique [1, 2, 3, 4, 6]. These techniques involve aborted transactions when they are unable to obtain a lock on a required data item where the proportion of transactions that aborted must be estimated besides the service demand of these transactions. However, some methods are better than the other based on the performance measures of interest in which a good method will deliver good performance.

Two of the major criteria involved in performance are:

i) The response time to a request, i.e.

   If R_k is the total residence time of a transaction at centre k (CPU or I/O), then the system response time, R, is the sum of residence time at various centres;

   $$R = \Sigma R_k$$

ii) The throughput capacity of the system.

   The system throughput can be calculated as;

   $$X = N / (z + R)$$

   where R is the system response time, z is the think time of the terminal to represent the delay and N is the number of transactions in the system.

Menasce and Nakanishi [4] proposed the idea of successful transaction by considering after transaction reads a set of database items (read set), it also updates the same set of items (write set) which is not quite true because the read set does not necessarily tend to update the database items. Assume that each transaction requires r read-locks and w write-lock of the I items in the database, if N transactions are active, they hold r * N read-locks but some of them do not tend to update the database items. Hence, r (read-locks) is not necessarily equal to w (write-locks). Therefore the formula given by Goodman et al. [2] for the probability of successful transaction should be accordingly modified where the number of active transactions as well as read set and write set should be considered. This idea will be adopted to develop the formula of the probability of aborted transaction (Pabort) and hence make comparison to the formula of Pabort developed by Mitrani et al [5] based on performance measures of interest.

The organisation of this paper is as follows :-
Section 2 describes the general model, parameters and the solution technique that are used to describe the formula of Pabort and the performance measures of interest. Section 3 describes the formula of the Pabort. Section 4 examines the results obtained from different approaches. Finally, Section 5 contains the conclusion reached due to the effect of aborted transactions, and also from the comparison between two methods.

2.0 MODEL, PARAMETERS AND SOLUTION TECHNIQUE

Database Environment

A computer system is composed of two major resources, the central processing units (CPU) and input-output (I/O) devices. In this work, computer system consists of one CPU, several identical I/O devices (disks), and a number of terminals. Our model for representing the executions of transactions is based on the closed queuing network model, where the number of transactions is constant and there are no external arrivals and departures. After receiving a certain amount of processing at the CPU, the transaction may require services from I/O devices. It then goes to the terminal if the transaction is successful or re-submitted after some delay during the conflict test (Fig. 1).

The following assumptions, regarding the computer system, are considered;

- After leaving the CPU the transaction selects, with equal probability, one of K I/O devices.
- The queuing discipline for resources of the computer system (CPU or I/O) is first come first served (FCFS).
- The proportion of time that a job completing service at centre j (CPU or I/O) proceeds directly to centre k is independent of the current queue lengths at any of the centres, for all j and k.
- The rate of completions of jobs from a service centre may not depend on the number of customers or transactions within the system.

A database is modeled as a collection of I resources, called items. An item is the smallest indivisible amount of information that may be acquired and locked [5].

The Transaction

The load on the system, that is the number of transactions in it, will be assumed constant. All transactions are assumed to be indistinguishable from one another and the requests are assumed to be uniform over the database.

The processing of a transaction consists of a locking phase, a processing phase, and a termination phase [3].

In the locking phase, the transaction requests a read-lock on each data item whose value is required. A read lock will be granted if no other transaction currently holds the write-lock on the item. The transaction requests a write-lock on each data item that is to be written. A write lock will be granted if no other transaction currently holds either a read-lock or a write-lock on the item. If any lock is refused, the transaction is aborted, releasing all locks previously granted to the transaction. The aborted transaction is re-submitted from the beginning.

In the processing phase, a transaction reads the values of the required data item. Based on these values, the transaction computes the values of the data items to be written. It then updates the values of the data items to be written.

In the termination phase, a transaction will release all read and write locks.

Parameter list.

I_n	-	the number of items in the database on disk n.
M	-	the number of devices in the system.
w_n	-	the number of write items on disk n.
r_n	-	the number of read items on disk n.
N	-	the number of transactions in the system.
D_{cpu}	-	the service demand of the CPU.
D_{succ_n}	-	the service demand of successful transaction at disk n.

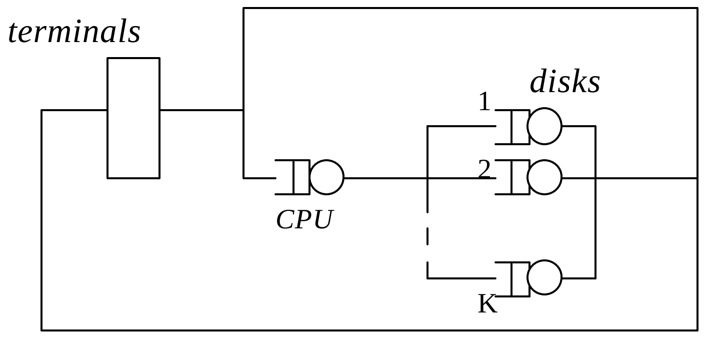

Fig. 1: The Computer System

Solution Technique for the performance measures of interest.

One of the techniques to evaluate the performance measures is known as Mean Value Analysis (MVA). It is based on the three equations:

(1) $X(N) = N / (z + \sum R_k(N))$

where $X(N)$ is the system throughput, $R_k(N)$ is the residence time at centre k when there are N transactions in the system, z is the think time of terminal to represent the delay.

(2) $Q_k(N) = X(N) * R_k(N)$

where $Q_k(N)$ is the average queue length at centre k, when there are N transactions in the system.

(3) $$R_k(N) = \begin{cases} D_k & \text{(delay centre)} \\ D_k(1+A_k(N)) & \text{(queuing centre)} \end{cases}$$

where D_k is the service demand at centre k, and $A_k(N)$ is the average number of transactions seen at centre k when a new transaction arrives.

3.0 THE FORMULA OF PABORT

The probability of non-conflict (successful) transactions given by Menasce and Nakanishi [4] is as follows:

$$\binom{I-r}{r} \Big/ \binom{I}{r}$$

But the proportion of successful transactions depends on factors such as the average number of active transactions and the average number of data items read and written by each transaction, relative to the total number of items in the database. We assume that each transaction requires read-locks on r and write-locks on w of the I items in the database. If N transactions are active, they hold N * w write locks. On arrival, a transaction will be able to acquire all of its r required read locks with probability

$$\binom{I - N*w}{r} \Big/ \binom{I}{r};$$

and hence,

$$\text{Pabort} = 1 - \binom{I - N*w}{r} \Big/ \binom{I}{r};$$

It is evident that the average number of active transactions is an output of the model. This is a key issue towards the estimation of Pabort.

Another approach to Pabort is given by Mitrani et al [5]. The discussion of Pabort is as follows:-

The influence of the processing-transaction is reflected by the parameter F: the probability that an item required by a transaction is unavailable. The formula for F is ;

$F = (K-1)m / (I-m)$,

where:

K is the number of transactions in the system,
m is the number of items held by a transaction,
I is the number of items on the disk.

Therefore, in this case, F is equivalent to Pabort. However this formula is not the same as the one discussed before. The aborted transactions are re-submitted after some delay.

The service demand of the transactions could be adjusted (by taking into account the Pabort) as follows:-

$D_k = (1 - \text{Pabort}_k) * D\mathit{succ}_k + \text{Pabort}_k * D\mathit{abort}_k$,

where:

$D\mathit{succ}_k$	is the service demand of the successful transaction at disk k.
$D\mathit{abort}_k$	is the service demand of the aborted transaction at disk k.

D*abort* can be estimated as one half of the lock manipulation overhead of a successful transaction (half of the required locks to be obtained before one is denied).

To compute the average response time required to successfully complete a transaction, the response time of each submission is multiplied by the average number of submissions required. The average number of submission is,

$$\begin{aligned} & 1 * (1 - \text{Pabort}) + \\ & 2 * (1 - \text{Pabort}) * \text{Pabort} + \\ & 3 * (1 - \text{Pabort}) * \text{Pabort}^2 + \\ & \cdot \\ & \cdot \\ & \cdot \\ & = 1/(1 - \text{Pabort}). \end{aligned}$$

The delay is very small. It is a constant which is dependent on the system. In this paper the average delay is to be 0.005 seconds. Hence the average delay of a transaction at the delay-centre is:

$$1/K\sum_{i=1}^{K}(1-Pabort_i) * z + Pabort_i * 0.005 \text{ seconds.}$$

where :

K	is the number of disks in the system,
$Pabort_i$	is the Pabort when it is unable to obtain a lock on a required data item on disk i.

4.0 RESULTS

The algorithm that is used to evaluate the performance measures by taking into account the Pabort in the database concurrency control is as follows:

Table 1, shows the average active transactions and Pabort at disks, the average queue length, the average response time and the throughput for the system when the service demands are varied and other parameters are constant. All disk cases have the same number of items, read items, write items, and successful service demand.

It shows that the average number of active transactions increases when service demand increases. Also, Pabort is increased due to the increase in the number of active transactions. Thus, it is noted that the average queue length increases when Pabort increases. Also, the average response time is increased due to the increase of Pabort, and the throughput is decreased (Fig. 2 to Fig. 5).

```
step 1.  Construct the traditional separable queuing network model with basic
         transaction service demands as calculated. Initially, assume that the
         average number of active transactions is zero. This will cause Pabort to
         be zero in the first iteration.

step 2.  Iterate as follows:
         2.1  Based on various parameters, determined the Pabort .
                   $Pabort_k = 1 - \binom{I_k - N_k * w_k}{r_k} \Big/ \binom{I_k}{r_k}$

         2.2  Calculate the service demands.
                   $D_k = (1 - Pabort_k) * Dsucc_k + Pabort_k * Dabort_k$

         2.3  Obtain the average number of active transactions using
              Mean value Analysis[MVA].

         Repeat step 2 until successive estimates of the average number of active
         transactions are sufficiently close. In this case the estimation is less
         than 0.001.
step 3.  Obtain performance measures from the final iteration.
```

Table 1: Pabort versus performance measures where service demand varies from 0.5 to 1.10

I = 700, N = 10, D_{cpu} = 0.6 , r = 30, w = 15					
Number of disks (K) = 3					
D*succ*	Ave. act. transactions	Pabort	Ave. Queue length	Response time	Throughput
0.50	0.5902	0.3274	5.3142	5.6703	0.9372
0.65	0.7057	0.3845	5.8830	7.1456	0.8233
0.80	0.8309	0.4113	6.3200	8.5858	0.7361
0.95	1.0372	0.5077	6.8762	10.9844	0.6256
1.10	1.1012	0.5293	7.1817	12.7403	0.5637

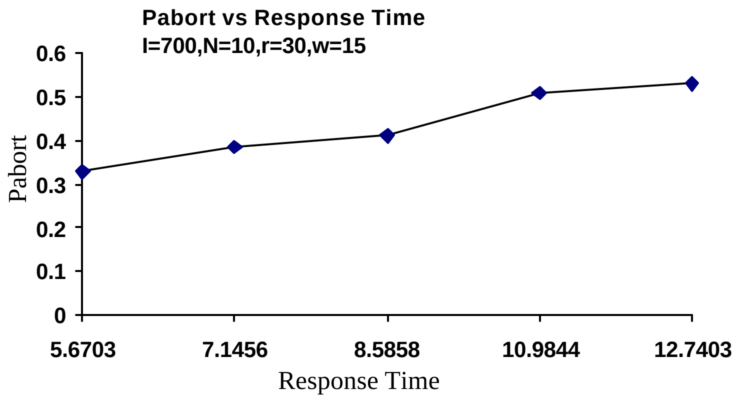

Fig. 2: Graph Pabort versus Response time

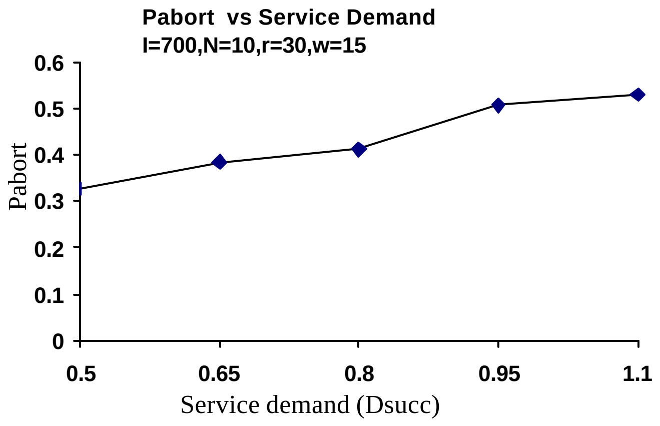

Fig. 3: Graph Pabort versus Successful service demand

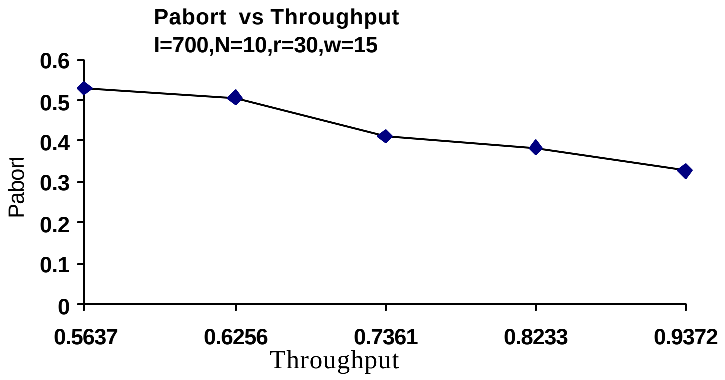

Fig. 4: Graph Pabort versus Throughput

Table 2 shows the average response time and the throughput performance measured when the number of write items (w) are varied and the other parameters remain constant. Table 2 also shows that Pabort increases when w increases. Again, the average response time increases and the throughput decreases since Pabort increases (Fig. 6 to Fig. 8).

Therefore, it can be concluded that Pabort plays an important role in evaluating the performance of database concurrency control mechanisms.

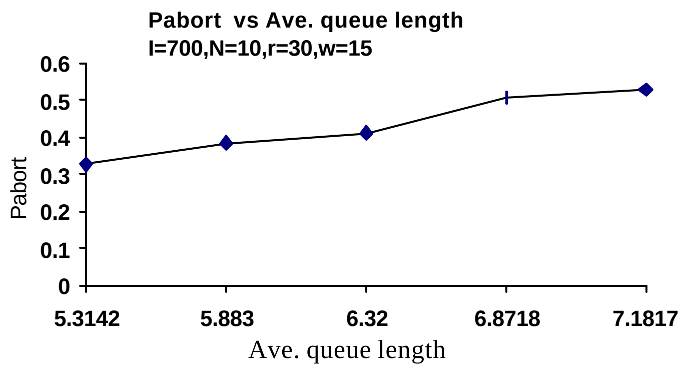

Fig. 5: Graph Pabort versus Average queue length

Table 2: Pabort versus performance measures where number of write items varies from 10 to 30

I = 700, N = 10, D_{cpu} = 0.6 , r = 30, Dsucc=0.5			
Number of disks (K) = 3			
w.	Pabort	Response time	Throughput
10	0.2651	5.2274	0.9778
15	0.3274	5.6703	0.9372
20	0.4113	6.5041	0.8693
25	0.4851	7.5830	0.7947
30	0.6242	1.2928	0.6138

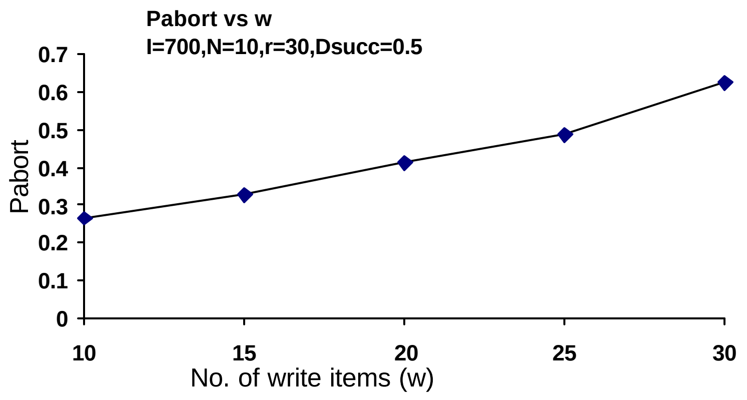

Fig. 6: Graph Pabort versus Number of write items

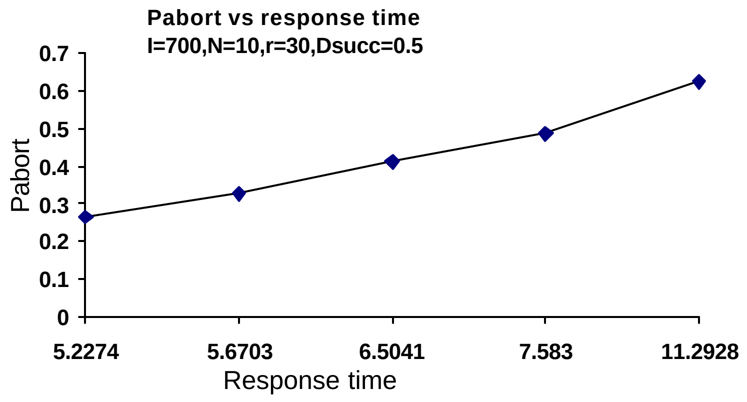

Fig. 7: Graph Pabort verses Response time

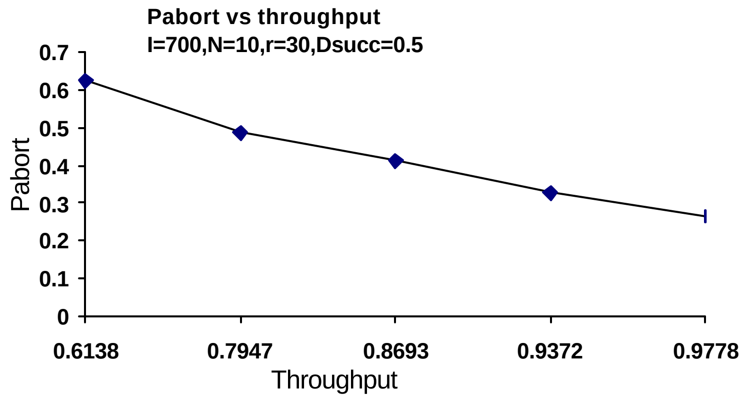

Fig. 8: Graph Pabort versus Throughput

Performance comparison

A comparison of performance measures between two methods based on different formula of Pabort will be carried out. The modified formula developed in section 3 for this paper, will be known as formula 1, and the formula developed by Mitrani et al [5], will be known as formula 2. The model of computer system consists of one CPU and single I/O. The algorithm which applies to formula 1 uses an iterative technique. The algorithm for formula 2 is different in that Pabort is evaluated from the constant parameters. The algorithm for formula 2 is given as follows:-

```
step 1. Input Parameters.
        $K, z, M, D_{cpu}, D_{succ}, I, m$
step 2. Calculate the Pabort.
        Pabort = $(K-1)*m/(I-m)$.
step 3. Revise the service demand
        $D = (1-\text{Pabort})* D_{succ} + \text{Pabort} * D_{abort}$
step 4. Evaluate the average response time
        and the throughput using MVA.
```

Table 3 shows the average response time and the throughput results obtained when the number of transactions(N) is increased from 5 to 20. Other input parameters are constant.

It was found that formula 1 shows a lower value of average response time when compared to formula 2. This is due to the lower value of Pabort of formula 1 when compared to the Pabort of formula 2 (Fig. 9). For instance, when N = 20, the average response time of formula 2 is 109.557 whereas the average response time given by formula 1 is 48.660 making the response time of formula 1 55.6% lower than that of formula 2 where the Pabort of formula 1 is 0.776 and Pabort of formula 2 is 0.895. Also the average queue length of formula 1 is smaller than that of formula 2 (Fig. 10). Hence, the throughput of formula 1 is larger than that of formula 2 since the average response time from formula 1 is smaller than that of formula 2 (Fig. 11).

Table 3: Performance measures for formula 1 (f 1) and formula 2 (f 2) where number of transactions varies from 5 to 20.

I=1000, r = 30, m = 45, z = 5, D_{cpu} = 0.6, D*succ* = 0.5								
N	Pabort		Ave. Queue length		Ave. Resp. time		Throughput	
	f 1	f 2	f 1	f 2	f 1	f 2	f 1	f 2
5	0.167	0.189	1.362	1.385	1.871	1.915	0.728	0.723
10	0.308	0.424	4.713	5.400	4.455	5.863	1.058	0.921
15	0.617	0.660	11.804	12.155	18.473	21.437	0.639	0.567
20	0.776	0.895	18.150	19.172	48.660	109.557	0.373	0.175

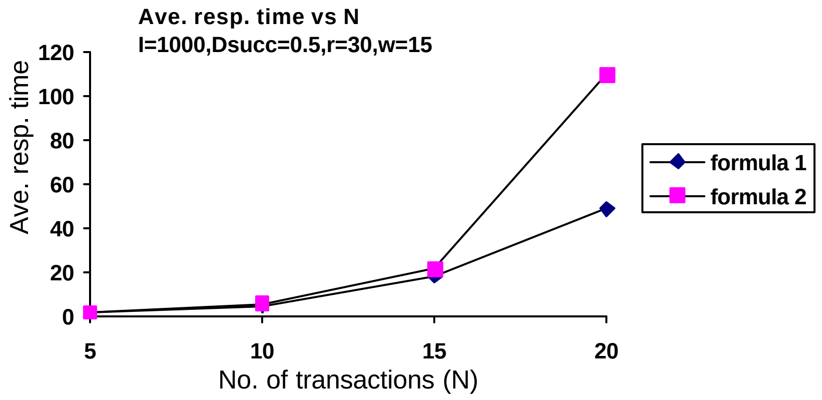

Fig. 9: Graph Average response time versus Number of transactions

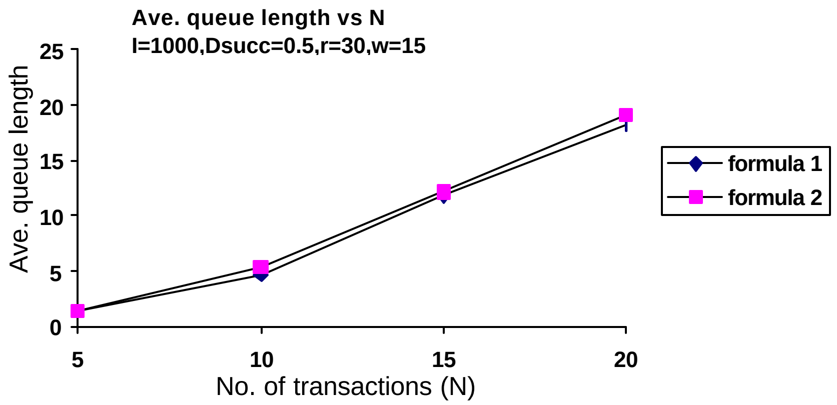

Fig. 10: Graph Average queue length versus Number of transactions

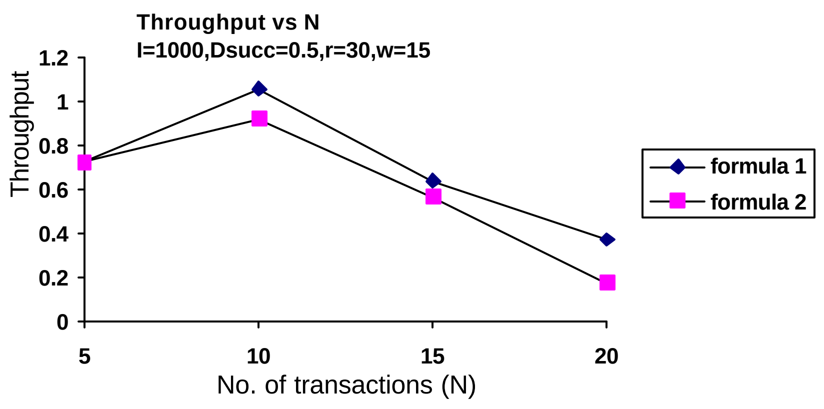

Fig. 11: Graph Throughput versus Number of transactions

Table 4: Performance measures for formula 1 (f 1) and formula 2 (f 2) when Number of Items varies from 500 to 2000

N=10, r = 30, m = 45, z = 5, D_{cpu} = 0.6, D*succ* = 0.5								
I	Pabort		Ave. queue length		Ave. response time		Throughput	
	f 1	f 2	f 1	f 2	f 1	f 2	f 1	f 2
500	0.678	0.890	7.326	9.069	13.668	49.560	0.536	0.183
700	0.411	0.618	5.315	6.883	5.672	10.794	0.937	0.633
1000	0.308	0.424	4.713	5.400	4.455	5.863	1.058	0.921
2000	0.204	0.207	4.226	4.242	3.659	3.682	1.155	1.152

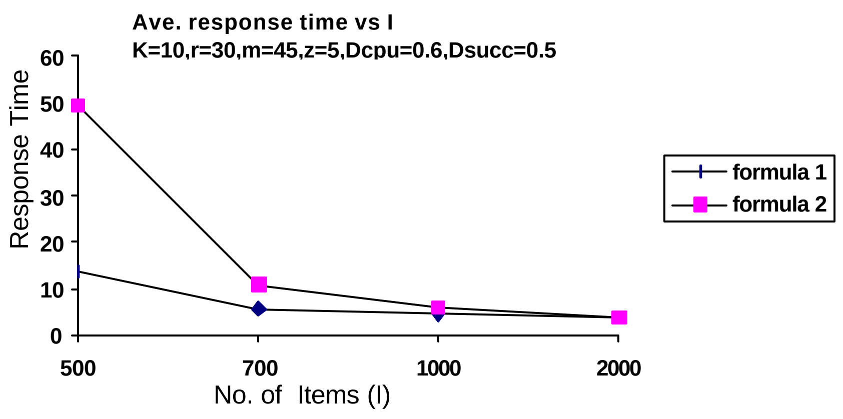

Fig. 12: Graph Average response time versus Number of items

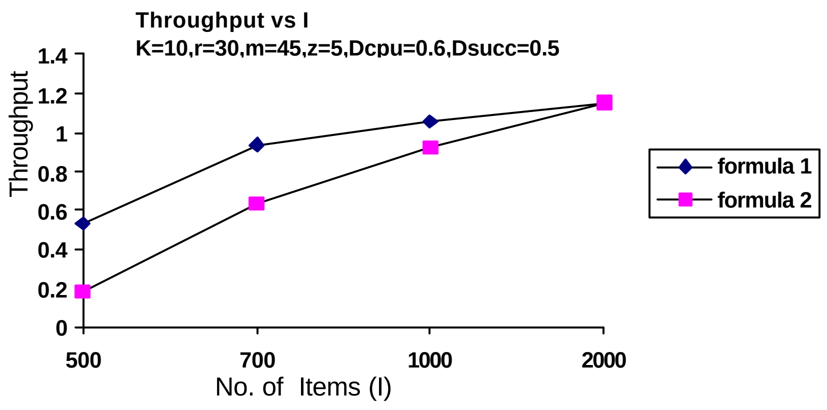

Fig. 13: Graph Throughput versus Number of items

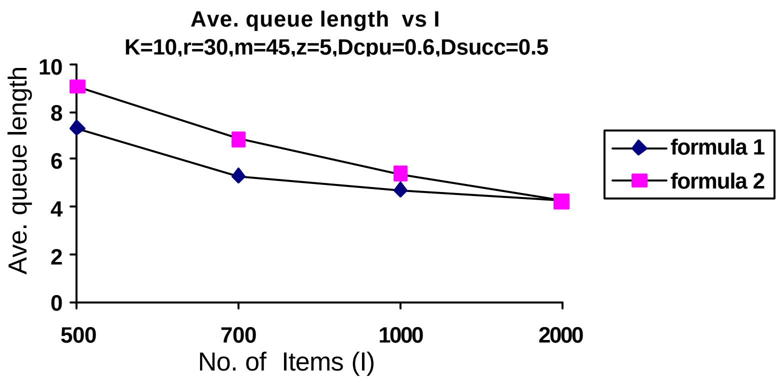

Fig. 14: Graph Average queue length versus Number of items

Table 4 shows the average queue length, the average response time and the throughput results obtained when the number of items (I) varies from 500 to 2000. Other input parameters are constant. For the case of I = 2000 the average queue length, the average response time and the throughput of formula 1 and formula 2 are almost the same. When I = 500 the average response time of formula 2 is much higher than that of formula 1, because Pabort for formula 2 is greater than Pabort for formula 1 (Fig. 12). Also, the average queue length of formula 2 is higher than that of formula 1 (Fig. 14) and the throughput of formula 2 is lower than that of formula 1 since the average response time from formula 2 is higher than that of formula 1 (Fig. 13).

5.0 CONCLUSION

This paper has shown how to develop the Pabort and it has also shown the results of the performance measure of interest. It was observed that the Pabort played an important role in the database concurrency control mechanism. Several parameters had played a key role in the characteristics of Pabort such as the number of transactions, the number of data items read and written by each transaction. It was concluded that the performance measure of interest for the database concurrency control was inversely related to the Pabort characteristic; i.e. as the Pabort increases the performance decreases.

Several cases were observed to identify the superiority of one formula to the other. The modified formula that was developed in this paper was found to be more favourable than the formula given by Mitrani et al [5]. Therefore, the two methods can be used to established the lower and upper performance bounds.

REFERENCES

[1] R. Agrawal, and M. J. Carey. "The performance of concurrency control and recovery algorithm for transaction-oriented database system". IEEE *Database Engineering*, 1987, pp. 58-67.

[2] N. Goodman, P. Bernstein. "Concurrency control in distributed database system". *ACM Computing Surveys*. 1981, pp. 185-221.

[3] E. D. Lazowska, J. Zahorjan, G. S. Grahan and K. C. Sevcik. *Quantative system performance: computer system analysis using queuing network model*. Prentice-Hall International. 1984.

[4] D. A. Menasce, T. Nakanishi. "Optimistic versus pessimistic concurrency control mechanisms in database management systems". *Inform. System*, Pergamon Press, Vol. 7, 1982, pp. 13-27.

[5] I. Mitrani, A. Chesnais and E. Gelenbe. "On modelling of parallel access to shared data". *Communication of the ACM*, Vol. 26, 1983, pp. 196-202.

[6] C. Ricardo, *Database system principles, design and implementation*. Mac Millan Pub. Company, 1990.

[7] K. C. Sevcik, "An analytical approach to the performance comparison of the database concurrency control methods". *World Computer Congress*, IFIP Congress, 1983.

BIOGRAPHY

Mustafa Mat Deris obtained his M.Sc in Computer Science from University of Bradford, England in 1989. Currently he is a lecturer at Faculty of Applied Science and Technology, University Putra Malaysia Terengganu. His research interest includes database and computer system performance.

Ali Mamat obtained his Ph.D in Computer Science from University of Bradford, England in 1991. Currently he is a lecturer in computer science in UPM. His research interest includes logic programming and database.