International Journal of Computer and Electrical Engineering, Vol. 1, No. 3, August 2009
1793-8163

On Analytical Performance Measurement of Concurrency Control Protocols in DBMS

Samir Kumar Ghosh, Md. Shohidul Islam, S. Member, IEEE, Md. Anisur Rahman

Abstract— **In this paper some commonly used concurrency control protocols have been implemented through simulation. It is well known that the transactions have mainly four properties: atomicity, consistency, isolation and durability, which are known as the ACID properties. The objective of concurrency control is to ensure consistency when a shared database is updated by multiple concurrent transactions. It is also used to increase the database object utilization. Among all the existing concurrency control protocols, standards are locking, two phase locking, graph based protocols, time stamp-based protocol, wait-die scheme and wound-wait scheme. This paper analyze implementation of two phase locking, wait-die and wound-wait schemes. To implement these protocols, a PC based model has been developed first. The outcome of the model is quite satisfactory. The relative performance of those protocols has been compared in terms of utilization. For each protocol a concurrency control manager has been designated which maintains all lock buffers, detects deadlock, and takes necessary action for deadlock recovery. Analyzing the whole experiment it is clear that wait-die scheme and wound-wait scheme protocols show better performance than two phases locking.**

***Index Terms*— Transaction, Serializability, Deadlock, Protocol**

I. INTRODUCTION

In view of the prominence of multiprocessing and multi-user environments in computer system, it is inefficient to restrict large database to sequential operation. If concurrent transactions are allowed, then throughput, resource utilization are improved and response time is reduced [4]. But all the usual problems associated with parallel accessing of data updating will arise. For example update of one transaction may be overwritten by another.

Update value of some uncommitted transaction may be read by another transaction that commit earlier and the first transaction rollbacks; and while processing some queries, some other transaction may update the desired records partially [1], [6]. Many algorithms have been developed to concurrently handle those situations in multiprocessing and multitasking environments. One of the fundamental properties of transaction is the isolation [1]. When several transaction execute concurrently in the database, the isolation property may no longer preserved. To ensure that, the system must control the interaction among the current transaction. This control is achieved through one of a variety of mechanisms called concurrency-control schemes. The concurrency-control schemes that we discuss in this paper are based on the Serializability property i.e. all the schemes presented here ensure that the schedules are Serialize. Among various protocols one way to ensure Serializability is to require that data items be accessed in a mutually exclusive manner i.e. while one transaction is accessing a data item no other transaction can modify that. Existing concurrency control protocols are standard locking, two phase locking, graph based protocols, time stamp-based protocol, wait-die scheme and wound-wait scheme [1], [2], [9]. We have implemented two phase locking, wait-die and wound-wait scheme. The relative performance of those protocols has been compared in terms of resource utilization.

Manuscript received on March 12, 2009. This work was supported in part by the Department of Computer Science & Engineering, RUET, Rajshahi-6204 and DUET, Gazipur-1700, Bangladesh respectively.

Samir Kumar Ghosh is with Dept. of Computer Engineering, National Institute of Technology Karnataka, Surathkal-575025, India.

Md. Shohidul Islam is with the Department of Computer Science & Engineering in Dhaka University of Engineering & Technology, Gazipur-1700, Bangladesh.

Md. Anisur Rahman completed graduation from Department of CSE in Rajshahi University of Engineering & Technology, Rajshahi-6204, Bangladesh.

.

II. CONCURRENT EXECUTIONS

A transaction (Txn) is a unit of program execution that accesses and possibly updates various data items. A Txn usually results from the execution of a user program written in a high-level data-manipulation language or programming language. The Txn consists of all operations executed between the begin and end indication.

To clarify concurrent execution let, T_1 and T_2 are two Txns that transfer funds from one account to another. Txn T_1 transfers \$50 from account A to account B and is defined as:

T_1	T_2
Read(A)	Read(A)
A:=A-50	Temp:=A*0.1
Write(A)	A:=A-temp
Read(B)	Write(A)
B:=B+50	Read(B)
Write(B)	B:=B+temp
	Write(B)

Let the current values of accounts A and B be \$1000 and \$2000 respectively. Suppose that the two Txns are executed one at a time in the order T_1 followed by T_2 as depicted in Table 1. The final values of accounts A and B after the execution take place, are \$855 and \$2145 respectively. Thus, the total amount of money in accounts A and B that is, the

sum A+B is preserved after the execution.

TABLE 1: SCHEDULE1, <T1,T2> (A=$855; B=$2145 & A+B=$3000)

T1	T2
Read(A) A:=A-50 Write(A) Read(B) B:=B+50 Write(B)	
	Read(A) Temp:=A*0.1 A:=A-temp Write(A) Read(B) B:=B+temp Write(B)

Similarly, if the Txns are executed one at a time in the order T_2 followed by T_1 then the corresponding execution sequence is that of Table 2. Again, as expected, the sum A+B is preserved, and the final values of accounts A and B are $850 and $2150 respectively. Here Table 1 and Table 2 are executed serially.

TABLE 2: SCHEDULE2<T2, T1> (A=$850; B=$2150 & A+B=$3000)

T_1	T_2
	Read(A) Temp:=A*0.1 A:=A-temp Write(A) Read(B) B:=B+temp Write(B)
Read(A) A:=A-50 Write(A) Read(B) B:=B+50; Write(B)	

When several Txns are executed concurrently, the corresponding schedule no longer needs to be serial. If two Txns are running concurrently, the operating system may execute one Txn for a little while, then perform a context switching, execute the second Txn for some times, and then switch back to the first transaction for some time, and so on. With multiple Txns, the CPU time is shared among them.

TABLE 3: SCHEDULE3 (A=&855; B=&2145&A+B=&3000)

T_1	T_2
Read(A) A:=A-50 Write(A)	
	Read(A) Temp:=A*0.1 A:=A-temp Write(A)
Read(B) B:=B+50 Write(B)	
	Read(B) B:=B+temp Write(B)

TABLE 4: SCHEDULE 4 (A=&900; B=&2150&A+B=&3050)

T_1	T_2
Read(A) A:=A-50	
	Read(A) Temp:=A*0.1 A:=A-temp Write(A) Read(B)
Write(A) Read(B) B:=B+50 Write(B)	
	B:=B+temp Write(B)

Suppose the two Txns are executed concurrently. One possible schedule is shown in Table 3. After this execution takes place, we arrive at the same state as the one in which the Txns are executed serially in the order T_1 followed by T_2. The sum A+B is indeed preserved.

Not all concurrent executions result in a correct state. To illustrate, consider the schedule of Table 4. After the execution of this schedule, we arrive at a state where the final values of accounts A and B are $900 and $2150 respectively. The final state is an inconsistent state, since we have gained $50 in the process of the current execution. Indeed, the sum A+B is not preserved by the execution of the two Txns.

III. SIMULATION MODEL

Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output. Simulation embodies the principle of "learning by doing" to learn about the system we must first build a model of some sort and then operate. Several methods of modeling systems exist, which do not involve simulation but involve solution of a closed-form system (such as a system of linear equations). Simulation is often essential in the following cases:

1) The model is very complex with many variables and interacting components.

2) The underlying variables relationships are nonlinear.

3) The model contains random variants.

4) The model output is to be visual as in a 3D computer animation.

Fig.1 shows the chronological order or working principles to ensure concurrent execution that consists of three stages- Scheduling, Run function and action of Concurrency Control Manager.

Scheduling:

Scheduling takes all the available transactions to make a schedule. At first randomize the random number generator, to get more random values. Generate a random number to select any transaction to run it till not all transactions completed their tasks. Then call Run().

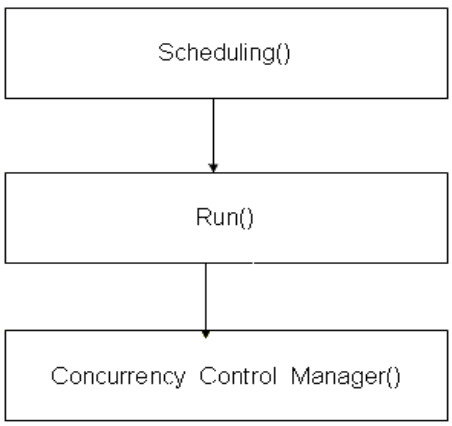

Fig. 1 Working principle of simulation models

Run function mainly depends on Lexical Analysis. Lexical Analyzer analyzes the instructions; detect the specified commands and data. Concurrency control manger maintains all lock buffer, wait for graph, detects deadlock and takes necessary action for deadlock recovery.

IV. Analysis of Simulation Result

In concurrent execution, increasing the no. of transaction increases possibility of CPU Utilization bounded by equation 1.

$$Utilization = \frac{total\_service\_time}{service\_time + waste\_time + wait\_time}$$

……..…(1)

Where Service time refers the slice of time taken by transaction instructions to be executed. Wait time implies the duration any transaction waits for data item hold by another transaction and Waste time is the time any transaction wastes for removing deadlock.

From equation 1 it is observed that increasing of waste time and wait time, decrease Utilization for specific transaction. But total concurrent execution time will be less than total serial execution time.

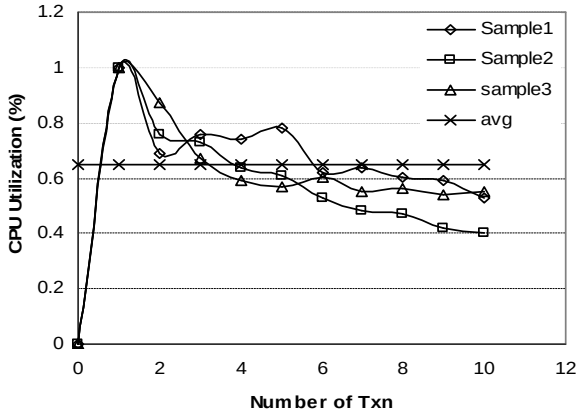

Fig. 2 CPU utilization for Two Phase scheme

Fig.2, Fig.3 and Fig.4 depicts the proportion of CPU utilization for Two Phase Locking, Wait Die and Wound Wait protocol respectively wherein X-axis shows the no. of transaction and Y-axis show the percentage of CPU utilization. We have taken 3 times sample data and calculated their average with a view to have flagrant understanding.

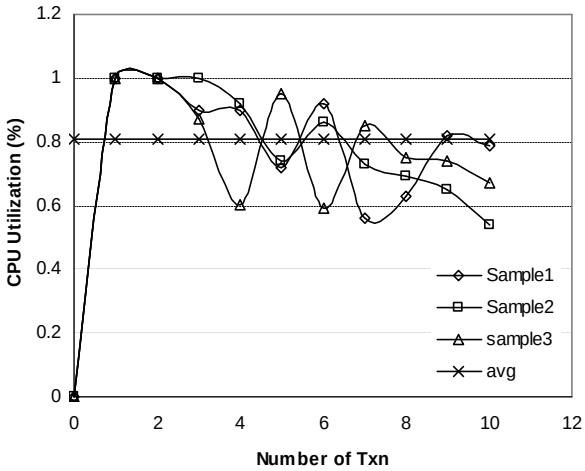

Fig. 3 CPU utilization for Wait Die scheme

For Two Phase Locking protocol average utilization is 0.65, for Wait Die Scheme average utilization is 0.81and for Wound Wait scheme it is 0.83. Hence Wait Die and Wound Wait scheme show better performance in comparison to Two Phase Locking which is also perceptible from Table 5.

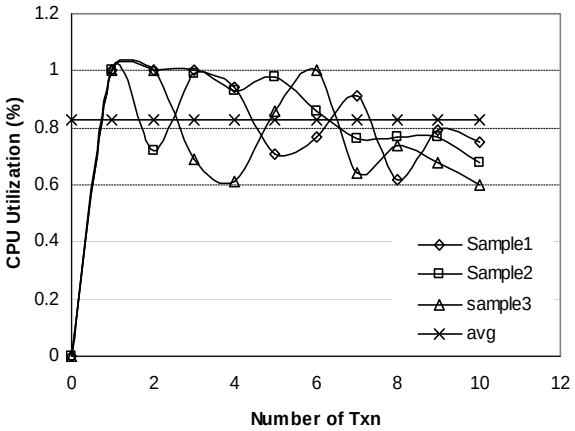

Fig. 4 CPU utilization for Wound Wait scheme

Table 5. CPU utilization by protocols over transaction

No. of Transaction	Utilization of CPU by		
	Two Phase	Wound Wait	Wait Die
0	0%	0%	0%
1	100%	100%	100%
2	69%	100%	70%
3	76%	78%	100%
4	74%	80%	77%
5	78%	86%	93%
6	62%	73%	92%
7	64%	61%	80%
8	60%	82%	39%
9	59%	80%	73%
10	53%	81%	72%

V. Conclusion

The simulation protocols can be applied for the distributed database. Future researchers may increase the no of transactions and analyze the results to improve performance of the simulated protocols. Using the concepts of our problem job scheduling, random number generation and other difficult problems can be solved as well.

References

[1] Abraham Silberschatz, Henry F.Korth and S.Sudarshan, DATABASE SYSTEM CONCEPTS, Third edition, McGraw-Hill.
[2] Jeffry D.Ullman, PRINCIPLES OF DATABASE SYSTEMS, Second Edition, Galgotia publications.
[3] H.T.Hung and John T.Robinson, "On optimistic Methods for Concurrency control" ACM transaction on database system, vol.6.n0.2, June 1981.
[4] Peter A.Franaszek, John T.Robinson and Alexander Thomsian, "Concurrency control for high contention environments", ACM transaction on database systems, vol.17, no.2, june-1992.
[5] Alexander Thomsian, "Concurrency Control: Methods, Performance and Analysis", ACM Computing Surveys, vol.30, no.1, march 1998.
[6] S.F Andler, Hansson, J.Ericson, J.Mellin.Berndtsson, and B.Eftring, "Deeds towards a distributed and active real-time database systems",.ACM Sigmoid Record, 15(1):38-40, March 1996.
[7] K.Ramamritham, "Real-time database", International Journal of Distributed and Parallel Database, 1(1),1992.
[8] L.Sha,R.Rajkumar, and J.P.Lehoczky, "Concurrency control for distributed real-time database", ACM SIGMOID Record, March 1988.
[9] Lee Juhnyoung and H.Son Sang, "Performance of concurrency control Mechanisms in Centralized Database Systems", Prentice-Hall,1996.
[10] O.Ulusoy, "Analysis of concurrency control protocols for real-time database systems", Technical Report BU-CEIS-9514,Bilkent University,1995.

Samir Kumar Ghosh completed graduation in Computer Science and Engineering from Rajshahi University of Engineering and Technology, Rajshahi 6204, Bangladesh in March 07, 2007. He is now postgraduate student of Dept. of Computer Engineering, National Institute of Technology Karnataka, Surathkal-575025, India.. His research interest includes different fields of Database and Data Mining. He is at samir023041@yahoo.com

Md. Shohidul Islam completed B.Sc Engineering in Computer Science and Engineering from Rajshahi University of Engineering and Technology, Rajshahi 6204, Bangladesh in March 07, 2007. He is a member of IEEE-USA, ACM-USA, IAENG-Hong Kong, IACSIT-Singapore, BCS-Bangladesh and IEB-Bangladesh and serving as lecturer of Computer Science and Engineering department in Dhaka University of Engineering and Technology, Gazipur-1700, Bangladesh. He is keen on research fields-Wireless Networking, Image Processing, Algorithm and Database. His email is shohidulcse@yahoo.com.

Md. Anisur Rahman completed B.Sc Engineering in Computer Science and Engineering from Rajshahi University of Engineering and Technology, Rajshahi 6204, Bangladesh in March 07, 2007. He is now research associate at DEBII in Curtin University of Technology, Pert, Western Australia. He is keen on research fields-Component Based Software, Service oriented architecture, Cloud Computing, Ontology, Wireless Networking, Mobile Computing, Database and, VoIP. His email is anis_javedd@yahoo.com.