HYBRIDIZED CONCURRENCY CONTROL TECHNIQUE FOR TRANSACTION PROCESSING IN DISTRIBUTED DATABASE SYSTEM

Article *in* Artificial Life and Robotics · October 2020

DOI: 10.47760/IJCSMC.2020.v09i09.012

2 authors, including:

Ledisi G. Kabari
Ken Saro-Wiwa Polytechnic, Bori
93 PUBLICATIONS **199** CITATIONS

SEE PROFILE

Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 9, Issue. 9, September 2020, pg.118 – 127

HYBRIDIZED CONCURRENCY CONTROL TECHNIQUE FOR TRANSACTION PROCESSING IN DISTRIBUTED DATABASE SYSTEM

Gabriel, Bariyira Christopher[1]; Kabari, Ledisi G.[2]
[1]Computer Science Department, Ignatius Ajuru University of Education, Port Harcourt, Nigeria
[2]Computer Science Department, Ken Saro-Wiwa Polytechnic, Bori, Nigeria
[1] gabrielbariyira@gmail.com; [2] ledisigiokkabari@yahoo.com
DOI: 10.47760/IJCSMC.2020.v09i09.012

Abstract*— Concurrency control has been actively investigated for the past several years and two-phase locking used as a standard solution for transaction processing in a database management system but the system is saturated with conflicts due to frequent rollbacks, long time waiting and blocking, high rate of aborted transactions that leads to deadlock. Hence to solve these problems, this paper presented a hybridized concurrency control technique which combines two phase locking and timestamp ordering. This technique will enhance the performance of the concurrency control techniques for transaction processing in a distributed database management system. This work is developed in ASP.Net using C# programming language as the front end and Microsoft SQL Server 2008 as the back end that is the database. Our result shows that two-phase locking and timestamp ordering is slow, time consuming during execution while our hybridized approach optimizes faster and consumed less time. This approach will give confidence to database administrators when handling concurrency transactions and make them deliver their work timely.

***Keywords*— Concurrency, Distributed database, Transaction processing, Hybridized concurrency control, Timestamp ordering, Two phase locking**

I. INTRODUCTION

The world presently depends on information systems, and with the rising need for accessible, reliable and secure information in the business environment of preset decade, the need for client/ server applications and distributed databases is also increasing. A single logical database which spread physically across computers located in multiple sites and connected by communication links is called distributed database [1]. Distributed database can be seen as virtual database that has component parts physically stored in a separate distinct real database at a number of separate locations. Concurrency control is an act of coordinating concurrent accesses to a database in a multi-user Database Management System (DBMS). A situation where users of a database are able to access a database in a multi-programmed fashion provided individual user is executing alone on a dedicated system is called concurrency control [2]. A user accessing a database is called transaction; and a transaction management handles all transactions properly in DBMS. Database transactions are processes like series of data read/write activities on data object(s) stored in database system [3].

Many researchers have carried out several investigations on concurrency control for decades of years now and the problem for non-distributed Data Base Management Systems is well understood. Mathematical theories have been developed to analyze the problem, and an approach, called two-phase locking, has been established as a standard solution [2].

A lot of conflicts are associated with using two-phase locking to perform transaction processing in a distributed database management system the system. This is due to time waiting of transaction execution, frequent rollbacks of transaction and aborted rate of transaction. Due to deadlock with locking and blocked transaction storage overhead increases. Keeping tracks of locks and queue waiting for data access is difficult and consequently processing of these overheads is high. In distributed database management systems (DDBMS) where reliability of the system keeps on increasing as a result of impact of data fragment and replication, solving this conflict becomes too difficult.[4].

Hence to solve these problems stated we need to develop a hybridized concurrency control technique which combines two phase locking and timestamp ordering. This technique will enhance the performance of the concurrency control techniques for transaction processing in a distributed database management system. The system would be implemented using Object-Oriented-Analysis-Design (OOAD) methodology. C# programming language would be used as the front end and Microsoft SQL Server 2008 as the back end.

The two-phase locking technique is not free from deadlock in the distributed database system. It has a limitation of frequent rollbacks of transactions, time waiting of transaction execution and high rate of aborted transaction). Storage overhead is increased because of deadlock with locking and the blocked transactions, also induces communication cost because of the deadlock problem. Timestamp ordering technique is free from deadlock because there is no too much waiting, blocking and locking of transaction. We intend to address the above problem by developing a hybridized concurrency control technique for transaction processing using two phase locking and time-stamp ordering system to optimized time execution for transaction processing in distributed database system.

This paper presents a hybridized concurrency control technique for transaction processing using two phase locking and time-stamp ordering system. The designed system will make an improvement in handling conflicts for concurrent transaction processing in a distributed database management system. It will optimize time execution for transaction processing in distributed database system. It is implemented using C# programming language at the front-end and Microsoft SQL Server 2008 as the back-end.

II. RELATED WORKS

A. Distributed Database

When we have one rational database which can be extended materially over client machine in many areas that are linked together through a computer network, such situation can be said to be a distributed database (DDB). "A distributed database is a database in which storage devices are not all attached to a common processing unit such as the C.P.U." "It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers"[5].

In distributed databases, there has to be a software which is known as Database Management System (DBMS) which helps in the control of data entry to different sites, if such is achieved, then we call the system a distributed database management system. "Processors communicate with one another through a communication network" [6]**.** For an effective sending and receiving of data in distributed database, there is need to have database management system with the aid of the processor to manage the retrieval of data within several nodes.

Figure 2.1, depicts a design in support for client machine that has Distributed Database Management System competence. Where every node owns at least one neighbouring database management system which controls data in the database that are kept in their respective nodes. It can also be noticed that, every node contains one replica of distributed database management system in addition with a correlated distributed data directory. Distributed data directory has several positions of every data within the network computer, in addition to the data descriptions, and needs for data from end users that are made used of initially through the distributed database management system.

B. Two-Phase Locking Technique

Two-phase locking technique uses First Come First Serve (FCFS) scheduling method in handling concurrent transaction, that is to say that transactions are executed one after another. Two-phase locking implements its transaction processing using serial execution.

Let E denote an execution of transactions T1 . . .. Tn. E is a *serial execution* if no transactions execute concurrently in E; that is, each transaction is executed to completion before the next one begins. Every serial execution is defined to be *correct*, because the properties of transactions imply that a serial execution terminates properly and preserves database consistency.

C. Timestamp Ordering Technique

Timestamp ordering technique uses the concept of Shortest Job First (SJF) scheduling method in handling concurrent transaction, with each transaction T*i* in the system, we associate a unique fixed timestamp, denoted by TS (T*i*). This timestamp is assigned by the database system before the transaction T*i* starts execution. If a transaction T*i* has been assigned timestamp TS (T*i*), and a new transaction T*j* enters the system, then TS(T*i*)<TS(T*j*). There are two simple methods for implementing this scheme:

1. Use the value of the system clock as the timestamp; that is, a transaction's timestamp is equal to the value of the clock when the transaction enters the system.
2. Use a logical counter that is incremented after a new timestamp has been assigned; that is a transaction's timestamp is equal to the value of the counter when the transaction enters the system.

R-timestamp and W-timestamp values is associated with each data item Q:

Where:

i. W-timestamp (Q) denotes the largest timestamp of any transaction that executed write (Q) successfully.
ii. R-timestamp (Q) denotes the largest timestamp of any transaction that executed read (Q) successfully.

For instance; suppose that transaction T*i* issue read (Q).

a. If TS (T*i*) <W-timestamp (Q), then T*i* need to read a value of Q that was already overwritten. Hence, the read operation is rejected, and T*i* is rolled back.

b. If TS (T*i*) ≥ W-timestamp (Q), then the read operation is executed, and R-timestamp (Q) is set to the maximum of R-timestamp (Q) and TS (T*i*).

Also suppose that transaction T*i* issue write (Q).

a. If TS (Ti) < R-timestamp (Q), then the value of Q that Ti is producing was needed previously and the system q that value would never be produced. Hence, the system rejects the write operation and rolls Ti back.

b. If TS (Ti) < W-timestamp (Q), then Ti is attempting to write an obsolete value of Q. Hence, the system rejects this write operation and rolls Ti back.

c. Otherwise, the system executes the write operation and sets W-timestamp (Q) to TS (Ti).

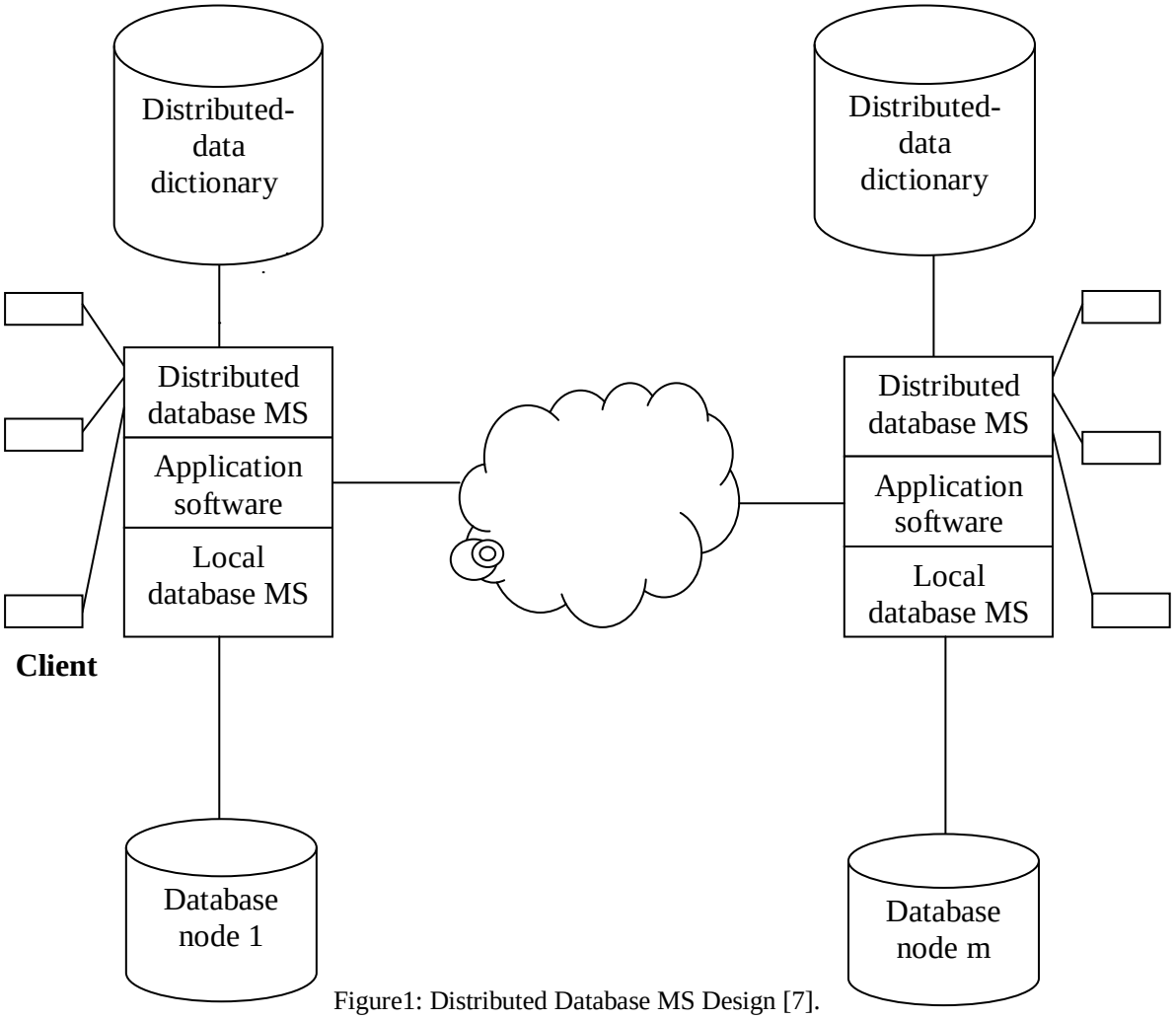

Figure1: Distributed Database MS Design [7].

III. METHODOLOGY

We simulated concurrency control techniques environment using C#. A database was created using MSSQL server 2008 where data are store. We develop a simple bank application program with a timer scheduler to interact with database. Six(6) different transaction process executions for 2 customers who want to check account balance, deposit and make withdrawal was used for demonstration. Figure2 shows the hybridized system as applied in the research.

The proposed hybridized system combines basic two-phase locking technique with Thomas Write Rule which is the modern technique for timestamp ordering as to improve the standard of controlling concurrent accesses by transactions processing in a distributed database system. The proposed system optimizes time such that Write Write (WW) conflicts never cause transactions to delayed or restarted against the existing system where there is a lot of restarting or delay.

It allows write operation to be performed on same data item by multiple transactions concurrently. The proposed system requires that every stored data item is associated with a lock time stamp, L-ts(x). It assigns a new timestamp called global timestamp to every transaction before execution, where transaction must finish within the time frame if eventually deadlock occurs then it will swap in other to finish within the time interval.

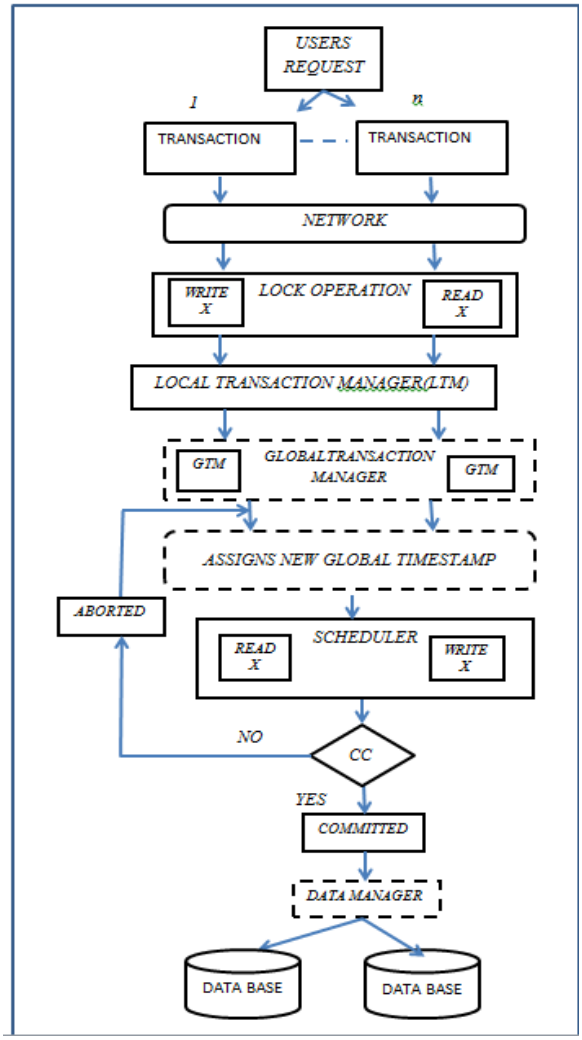

Figure2: Hybridized System

IV. RESULTS AND DISCUSSION

Transactions are been tested using individual technique, start time, end time, total time and the result were displayed which shows that two-phase locking is slow and time consuming and timestamp ordering is better than two-phase locking in terms of execution time while hybrid optimizes time that is the most fastest when dealing with time because is faster than other techniques when applied individually.

Figure3 shows transaction for Deposit and Checking Balance while Figure4 shows transaction for Deposit and Withdraw.

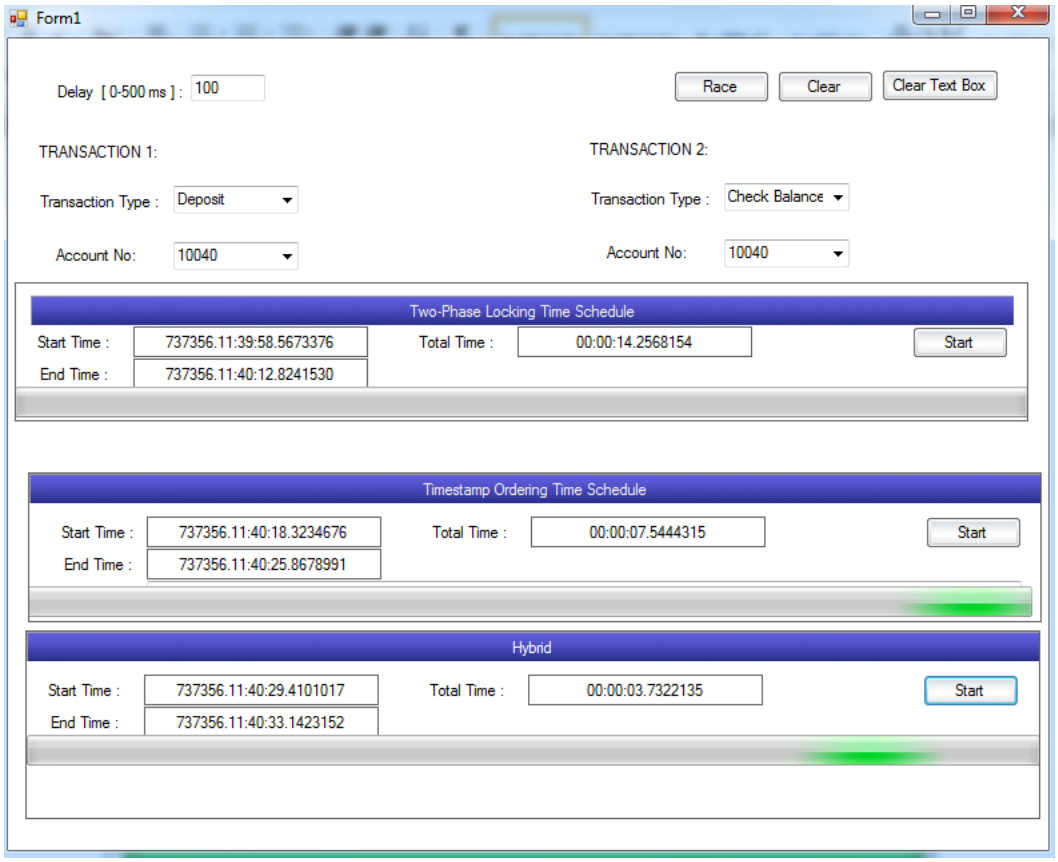

Figure3: Transaction for Deposit and Checking Balance

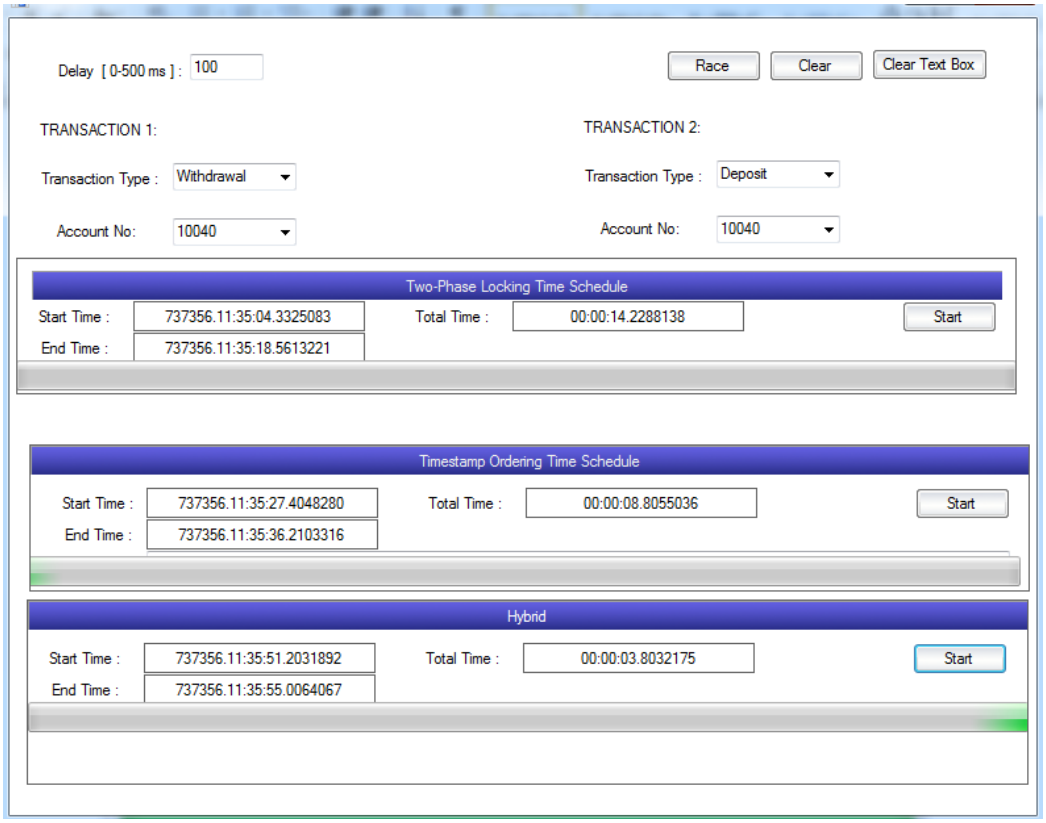

Figure 4: Transaction for Deposit and Withdraw

The total number of transactions that are accessing the database simultaneously have a timestamp been assigned to them by their transaction managers. The timestamp is categories into local and global. Local timestamp is the time taken for the user to send a transaction from its server location to the main server while global is the time taken for the main server to execute the transaction which is represented as the start-time and end-time and consequently, the execution time is calculated. Table1 shows the Two-phase locking which also contains field called committed which indicate whether transaction was committed or aborted. Committed transaction record the total number of successful transaction execution while aborted transaction records the uncompleted transactions.

Table1: Two-phase locking

Transaction	Start Time (ST) in second(s)	End Time (ET) in second(s)	Total Execution Time (TT) in second(s)	Committed
T1	31	44	13	Yes
T2	26	38	12	Yes
T3	35	47	12	Yes
T4	12	24	11.9≈12	Yes
T5	13	26	13	Yes
T6	12	25	13	Yes

Where;
T1= check balance, T2= Make deposit, T3=Make withdraw, T4= Deposit and withdraw, T5= Check balance and deposit, T6= check balance and withdraw.

In the table1, there are 2 customers who want to check the same account balance at the same time, the transaction in our context is called T1 which is a read operation. T2 is a transaction performs when more than 1 customer is trying to make a deposit into a single account number at the same time. That is performing an account update. T3 is a transaction performs when more than 1 customer is trying to make a withdrawal from a single account number at the same time. That is performing an account update. T4 is a transaction performs when more than 1 customer is trying to make a deposit and withdraw from a single account number at the same time. That is performing an account update. T5 is a transaction performs when more than 1 customer is trying to make a deposit and check balance from a single account number at the same time. That is performing read and write operation. T6 is a transaction performs when more than 1 customer is trying to withdraw and check balance from a single account number at the same time. That is performing read and write operation. The start time and the end time determines the total time taken for execution.

In figure3, Y is the number of transactions in a database; X is the total time taken for execution using two-phase locking which uses more time compare to others. In figure2 we plotted the value of time spent on process execution against Y. It shows that 2PL technique is time consuming because of first come first serve method used in the client side. Figure2 shows Total execution time for transactions using 2pl.

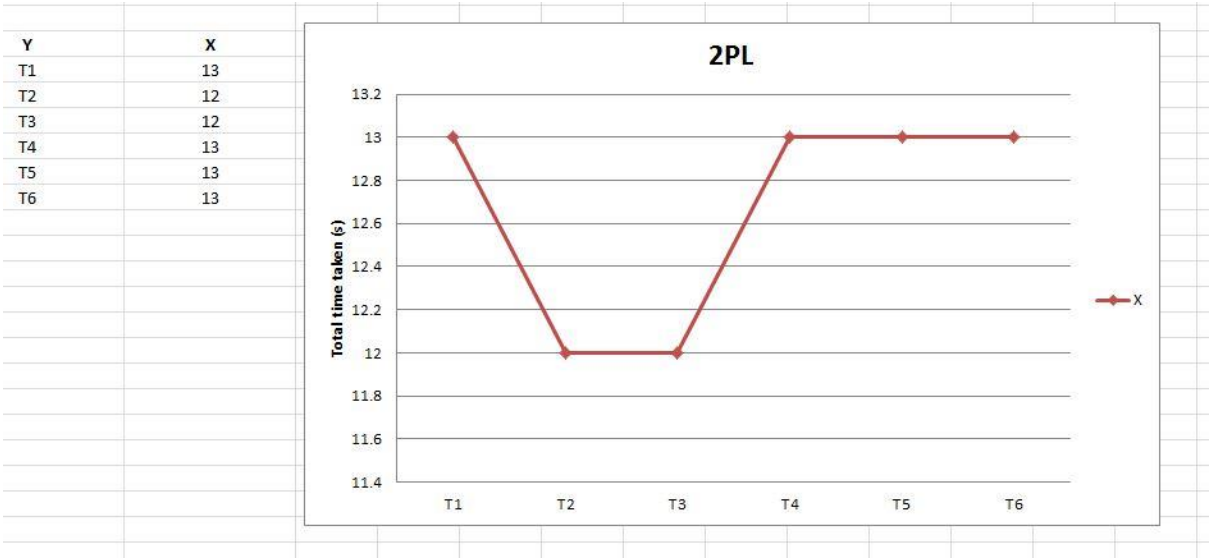

Figure3: Total execution time for transactions using 2pl

Table2 shows the Timestamp ordering with the following columns; transaction(T), StartTime(ST), EndTime(ET), TotalExecutionTime(TT) which is measured in seconds and a committed column. Table2 records

all the transaction perform by customers for read and write operation using a timestamp ordering technique. From the start time and the end time, the total time taken for execution is determined.

Table2: Timestamp ordering

Transaction	Start Time (ST) in second(s)	End Time (ET) in second(s)	Total Execution Time (TT) in second(s)	Committed
T1	03	12	08	Yes
T2	02	09	07	Yes
T3	04	11	07	yes
T4	25	32	6.8≈7	yes
T5	32	40	7.8≈8	yes
T6	45	51	06	yes

In figure4, Y is the number of transactions in a database; X is the total time taken for execution using Timestamp Ordering(T/O) which uses moderate time compare to two phase locking. In figure3 we plotted the value of time spent on process execution against Y. It shows that Timestamp Ordering(T/O) technique is moderately time consuming unlike the 2PL.

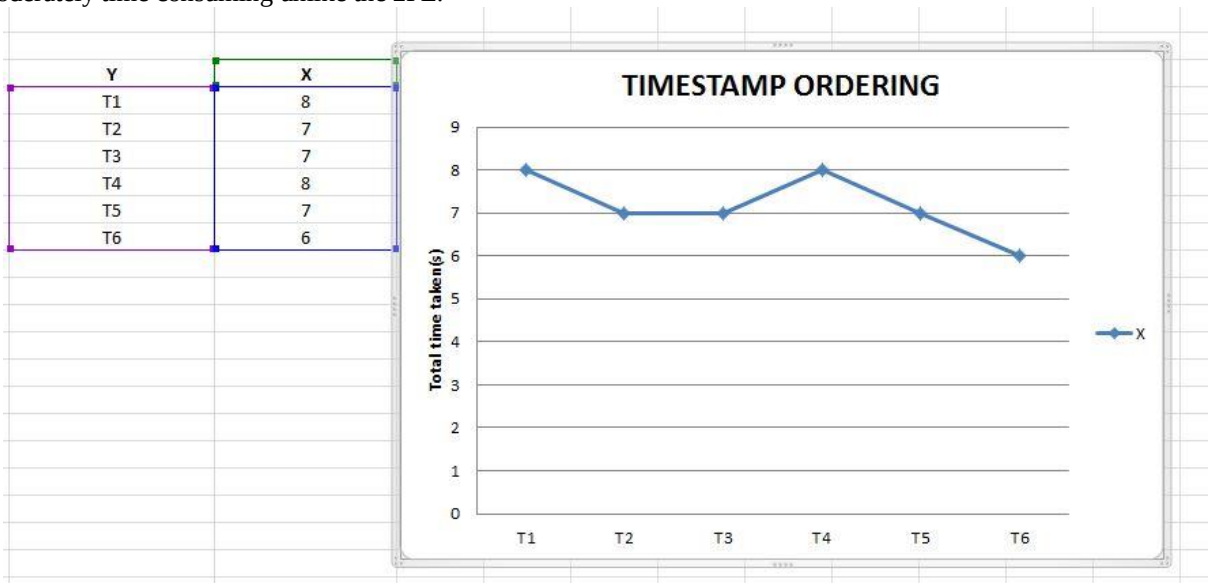

Figure4: Total execution time for Timestamp Ordering(T/O) technique

In table3 shows the Hybrid which has following columns; transaction(T), StartTime(ST), EndTime(ET), TotalExecutionTime(TT) which is measured in seconds and a committed column. Table3 records all the transaction perform by customers for read and write operation using a hybrid technique. From the start time and the end time, the total time taken for execution is determined.

Table 1.2: Hybrid

Transaction	Start Time (ST) in second(s)	End Time (ET) in second(s)	Total Execution Time (TT) in second(s)	Committed
T1	47	53	06	Yes
T2	29	34	05	Yes
T3	28	34	06	Yes
T4	34	38	3.7≈4	Yes
T5	42	46	3.7≈4	Yes
T6	03	08	05	Yes

In figure5, Y is the number of transactions in a database; X is the total time taken for execution using Hybrid Technique which is faster compare to two phase locking and timestamp ordering technique. In figure4, we plotted the value of time spent on process execution against Y. It shows that Hybrid technique is faster unlike the 2PL and T/O.

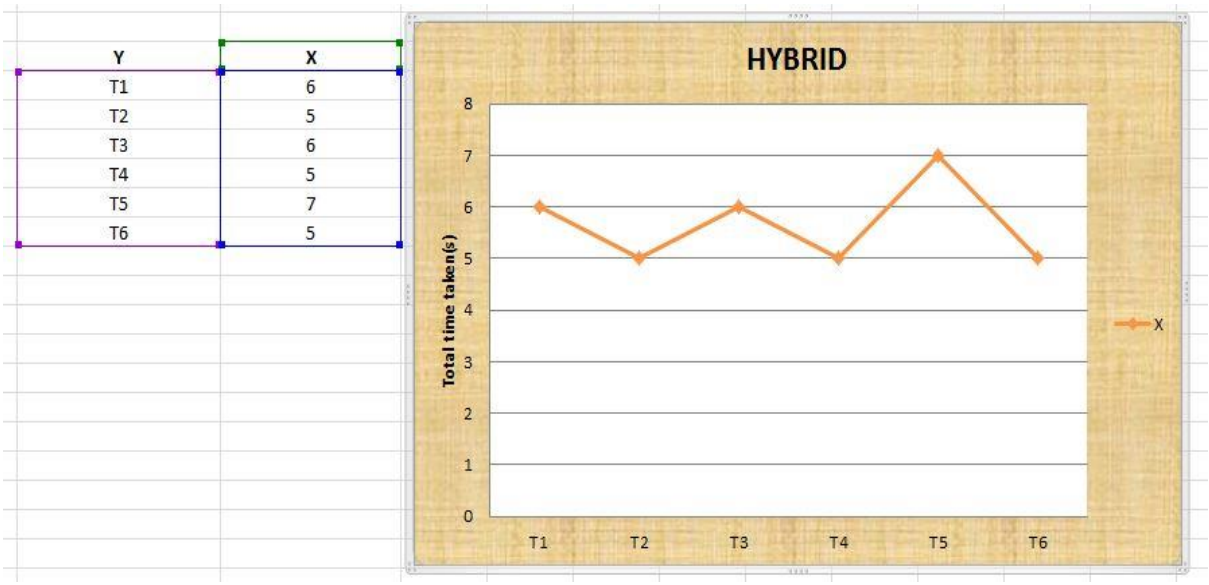

Figure5: Total execution time for transactions using hybridized technique

In table4, we describe the three (3) different technique used and then find the average execution time of each of the techniques when apply separately will be given as follows;

Average Execution Time = $\frac{Total\ number\ of\ execution\ time}{Total\ number\ of\ transaction}$

For two-phase locking:

AET $\frac{72}{6} = 12s$

For timestamp ordering:

AET $\frac{43}{6} = 7.16$

For hybridized:

AET $\frac{30}{6} = 5s$

Table4: Comparison of the three Techniques

	2PL	T/O	HYBRID
Transaction	Total Execution Time (TT) in second(s)	Total Execution Time (TT) in second(s)	Total Execution Time (TT) in second(s)
T1	12	08	06
T2	11	07	05
T3	11	07	06
T4	13	07	4
T5	13	08	4
T6	12	06	05
Total = 6	72	43	30

In table4 we show the total execution time for 2PL is high that is time consuming for all the transactions executed compare to timestamp ordering and hybrid, the total time taken when using hybrid is lesser and fast. Therefore, hybrid is best used when dealing with concurrent transactions execution in a distributed database system.

Table5 shows the average execution time for Two-phase locking, Timestamp ordering and the hybridized technique. We conclude that the hybridized technique is low, while Timestamp is medium and Two-phase is the highest in terms of average execution time.

Table5: Average Time

Technique	**Average Execution Time(AET)**	**Status**
Two-phase locking	12	High
Timestamp ordering	7.16	Medium
Hybridized	5	Low

In figure6 we indicate Y as the number of transactions in a database, X is the total time taken for two-phase locking technique, X_1 is the total time taken for Timestamp ordering technique and finally, X_2 is the total time taken for hybridized technique that uses lesser time.
In Graph 1.5 we plotted the value of time spent on process execution as represented as X, X_1 and X_2 against Y.

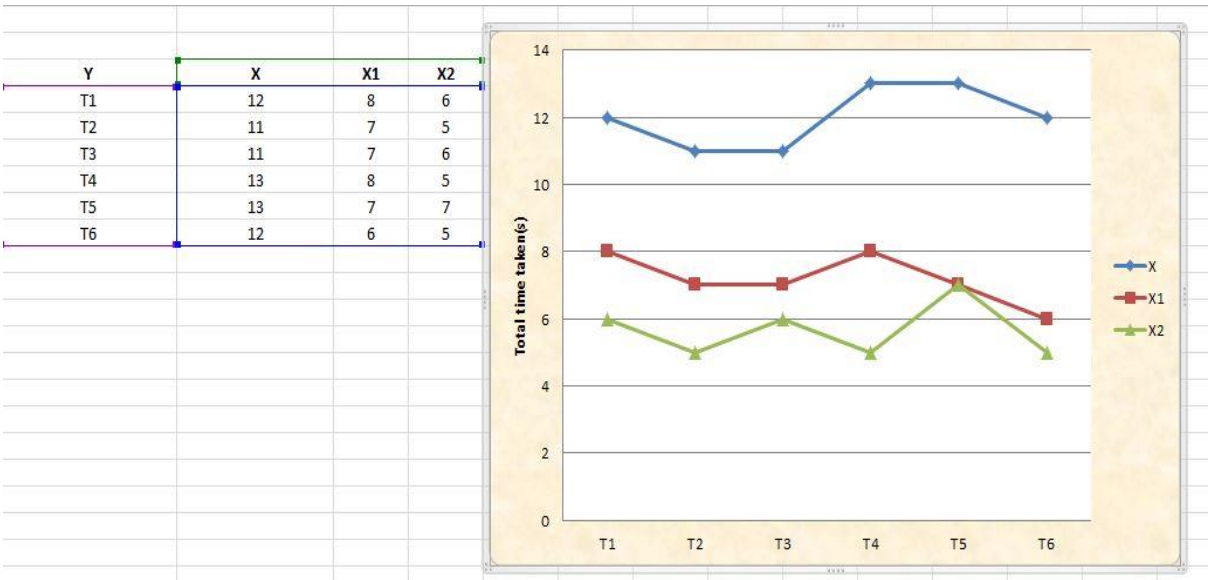

Figure6: shows the Total Execution Time for Transactions

V. CONCLUSIONS

In this research work, we dealt with concurrency control techniques for transaction processing in Distributed databases which involves client and server machines respectively. Concurrency control in distributed database is the activity of coordinating concurrent accesses to a database in a multi-user Database Management System (DBMS). Concurrency control permits users to access a database in a multi-programmed manner provided each user is executing alone on a dedicated system. We adopted Object Oriented Analysis and Design as the methodology in developing this distributed database system and C# programming language was used at the front end while Microsoft SQL server 2008 is the relational database management system used at the back-end.

This proposed project "A Hybridized Technique concurrency control technique for transaction Processing in a Distributed Database System" is a technique that optimizes time and improve on deadlock solution for managing concurrent transactions in a distributed database system. Therefore, we can conclude that the application has been developed using the recommended techniques in order to reduce waiting time for transaction processing to avoid deadlock problem.

REFERENCES

[1]. M. Kaur and H. Kaur, H., "Concurrency Control in Distributed Database System". International Journal of Advanced Research in Computer Science and Software Engineering. 3(7), 2015.

[2]. K. Arun and A. Agarwal, "A Distributed Architecture for Transactions Synchronization in Distributed Database Systems". International Journal on Computer Science and Engineering (IJCSE). 2(6)1984-1991, 2010.

[3]. A. Larson, S. Blanas, C. Diaconu, C. Freedman, M. Patel, and M. Zwilling, "High-performance concurrency control mechanisms for main-memory databases". Proceedings of the VLDB Endowment, 5(4) 298-309, 2011.

[4]. Z. Svetlana and P. Aleksandar, "Models of 2PL Algorithms with Timestamp Ordering for Distributed Transactions Concurrency Control". International Journal of Soft Computing and Engineering (IJSCE). 3(4), 2013.
[5]. S. Gupta, K. Saroba and K. Bhawna. (2011), "Fundamental Research of Distributed Database", International Journal of Computer Science and Management Studies, (11) 138-146, 2011.
[6]. Y. Clement and S. Meng, (1998). Principles of Database Query Processing for Advanced Application, 1998.
[7]. D. Bell and J. Grimson, "*Distributed Database Systems*", Reading, MA: Addison-Wesley, 1992.

