EAST WEST UNIVERSITY

Theory Report

Course Title: Advance database

Course Code: CSE464

Section: 02

Project Title: Hybridized Concurrency Control Technique For Transaction Processing In Distributed Database System

Submitted To

Sadika Islam Sneha

Lecturer, Department of Computer Science and Engineering

Submitted by

ID	Student Name
2018-2-60-010	Syeda Tamanna Sheme
2018-3-60-075	Nabila Islam
2018-3-60-032	Lubaba Alam Chhoa

Date of Submission: 08-09-2022

Introduction

Two-phase locking is the industry standard for transaction processing in database management systems, and Concurrency control has been actively researched over the past few years. However, the system is overloaded with conflicts as a result of frequent rollbacks, prolonged waiting and blocking, and a high rate of aborted transactions that result in a deadlock. The paper we have chosen presented a hybridized concurrency management strategy that combines two-phase locking and timestamp ordering in order to address these issues. In a distributed database management system, this strategy improves the performance of concurrency control techniques for transaction processing. The outcome demonstrates that the hybridized strategy optimizes more quickly and uses less time than two-phase locking and timestamp ordering during execution [1].

Methodology and Results

The database for this work is Microsoft SQL Server 2008, with C# serving as the front end's programming language and ASP.Net serving as the back end. To connect with the database, the authors create a straightforward bank application software with a timed scheduler. For the demonstration, six possible transaction process executions for two clients who wanted to deposit money, check their account balance, and withdraw money were used. The hybridization system used in the research is depicted in Figure 1 [1].

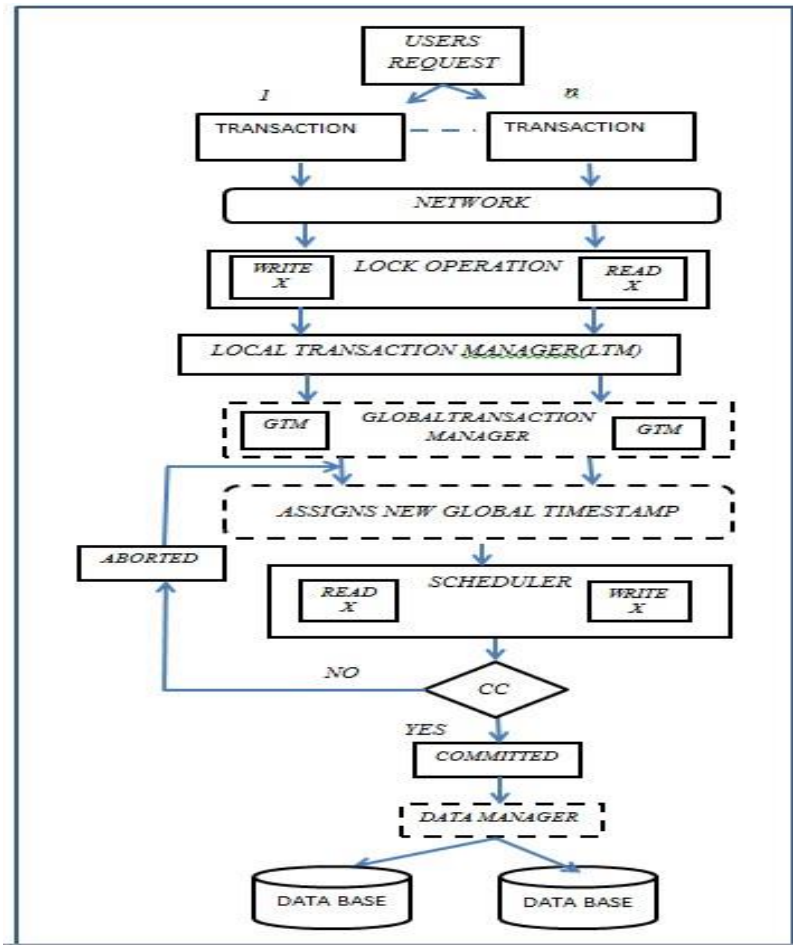

Figure 1: Hybridized System

The proposed hybridized system combines the basic two-phase locking technique with the Thomas Write Rule which is the modern technique for timestamp order to improve the standard of controlling concurrent accesses by transactions processing in a distributed database system. In contrast to the current system, which frequently restarts or delays transactions due to Write Write (WW) conflicts, the suggested approach optimizes time so that this never happens. It allows writing operations to be performed on the 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). Every transaction is given a brand-new timestamp before it is executed, known as the global timestamp. The transaction must complete within the time frame; if a deadlock finally develops, another transaction will be switched in to finish within the time limit [1].

Two-phase locking is slow and time-consuming, timestamp ordering is better than two-phase locking in terms of execution time, and hybrid optimizes the time that is the fastest when dealing with time because it is faster than other techniques when applied individually. Transactions are tested using the individual technique, start time, end time, and total time, and the results were displayed.
Figures 2 and 3 depict transactions for deposits and checking balances and withdrawals, re spectively [1].

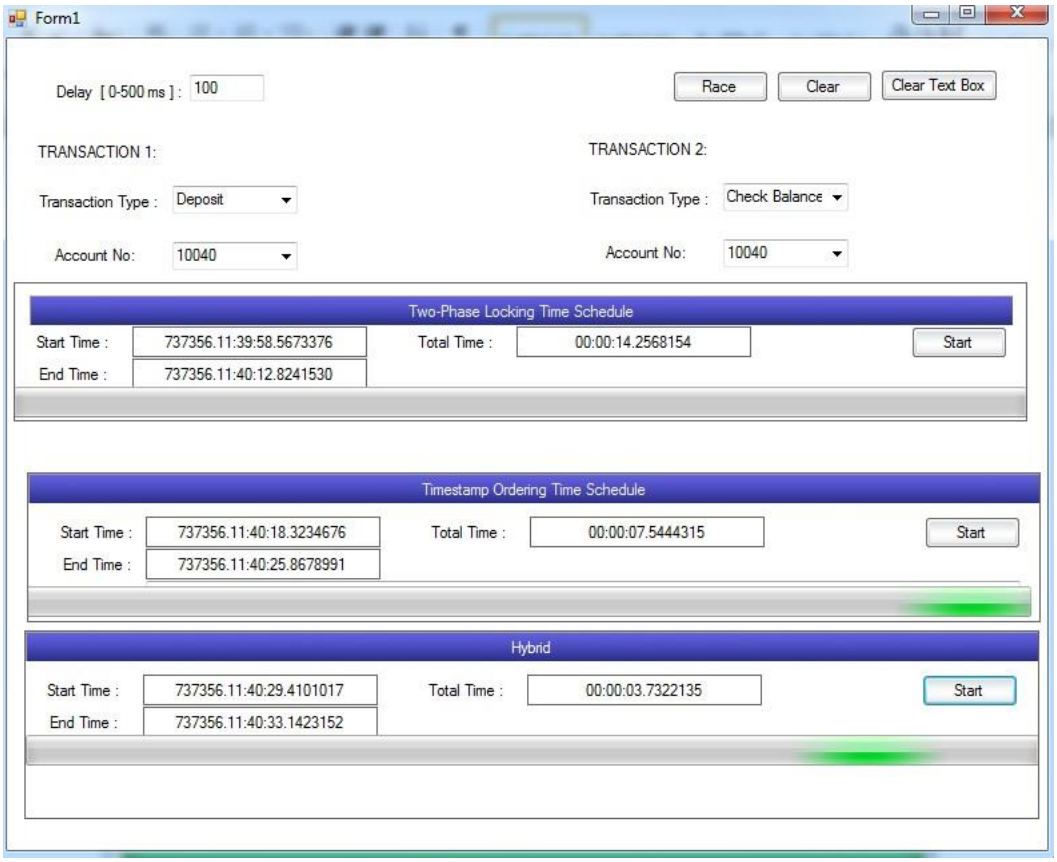

Figure 2: Transaction for Deposit and Checking Balance

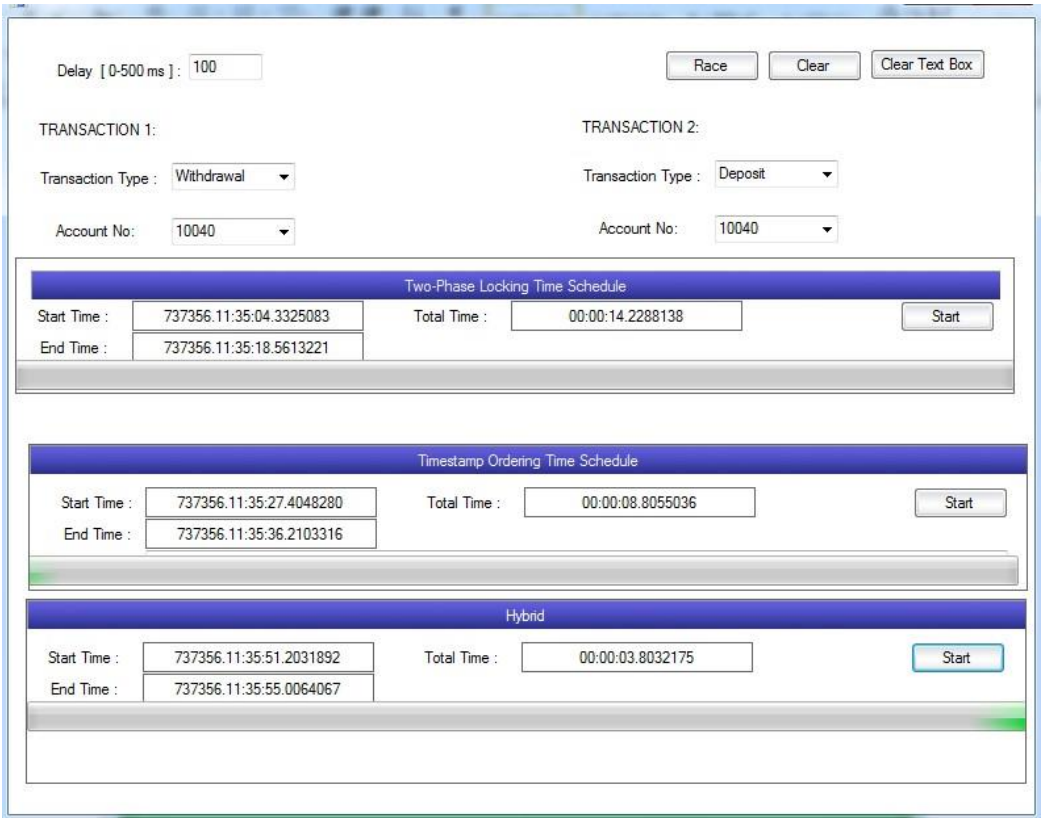

Figure 3: Transaction for Deposit and Withdraw

The transaction managers of each transaction have allocated a timestamp to each of the transactions that are concurrently accessing the database. There are two types of timestamps: local and international. The global timestamp is the amount of time it takes for the main server to execute the transaction, which is represented as the start-time and end-time, after which the execution time is calculated. The local timestamp is the amount of time it takes for the user to send a transaction from its server location to the main server. The Two-phase locking is displayed in Table 1 along with a field called committed that shows whether the transaction was committed or aborted. A committed transaction records the total number of successfully executed transactions, whereas an aborted transaction records the transactions that were not successfully executed [1].

Table 1: 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 a deposit, T3=Make withdraw, T4= Deposit and withdraw, T5= Check balance and deposit, T6= check balance and withdraw.

Table 2: 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

Table 2 shows the number of database transactions (Y) and the overall execution time (X), which uses Timestamp Ordering (T/O), a method that takes less time than two-phase locking.

Table 3: 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 table 3, we can see that in contrast to 2PL and T/O, a hybrid approach is faster.

Table 4: 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

Table 4 lists the three different techniques that were employed, followed by the average execution times for each of the strategies when utilized separately:

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

Table 4 shows that, when compared to timestamp ordering and hybrid, the overall execution time for 2PL is high, making it time-consuming for all of the transactions that are carried out. When using hybrid, however, the entire execution time is low and quick. So when dealing with concurrent transaction execution in a distributed database system, a hybrid is preferable.

Table 5: Average Time

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

Table 5 displays the typical execution times for the hybridization method, timestamp ordering, and two-phase locking. In terms of average execution time, we find that the hybridized approach is low, the Timestamp is medium, and the Two-phase is the highest [1].

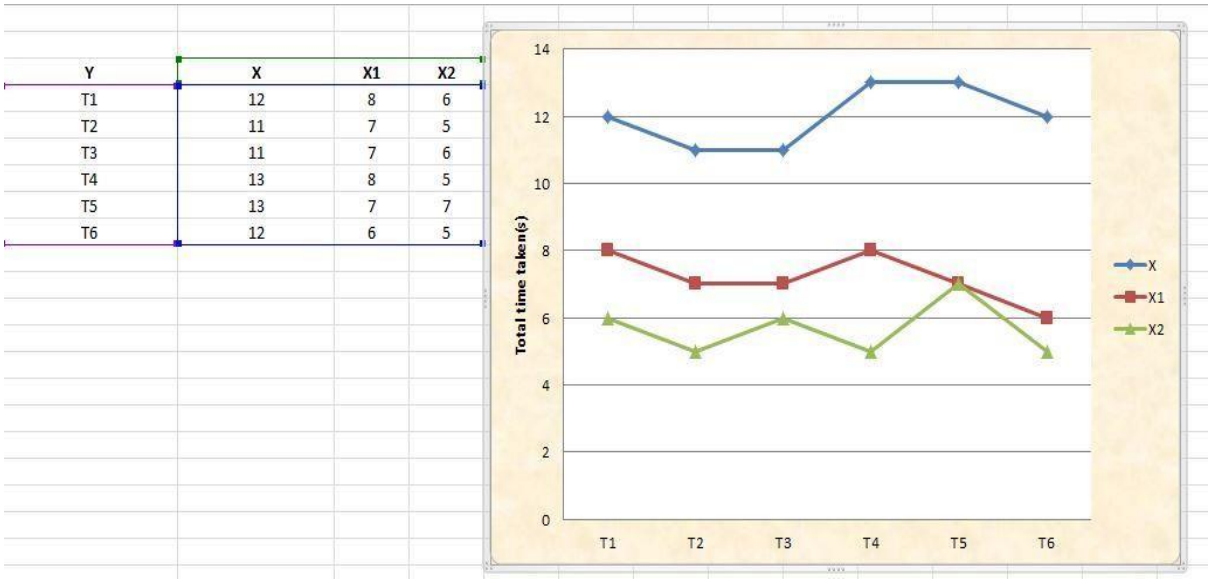

Figure 4: shows the Total Execution Time for Transactions

In figure 4 we indicate (left side) Y as the number of transactions in a database, X is the total time taken for the two-phase locking technique, X_1 is the total time taken for the Timestamp ordering technique and finally, X_2 is the total time taken for a hybridized technique that uses lesser time [1].
In Graph, we plotted (right side) the value of time spent on process execution as represented as X, X_1, and X_2 against Y.

As a result, it is clear that the system as intended improves the way conflicts are handled for concurrent transaction processing in a distributed database management system. Additionally, it streamlines the process of processing transactions in distributed database systems.

Related work

The Morpheus concurrency testing approach for Erlang's distributed real-world systems was introduced in the article. Morpheus employs POS to take advantage of the straightforward underlying causes of concurrency issues, targets high-level concurrency for Erlang systems to concentrate on protocol-level errors, and adds conflict analysis to further cut down on pointless attempts to reorder non-conflicting actions. In popular real-world distributed systems written in Erlang, Morpheus successfully discovered protocol-level problems, according to the authors' evaluation [2].

Morpheus has the same drawbacks as earlier randomized or systematic testing methods. In addition to fail-stop faults and infinite loops, it also depends on test cases to supply inputs to the system being tested and high-level invariants to check. Limited methods of communicating with non-Erlang code are supported [2].

The paper proposed an improvement of 2PL to achieve deadlock-free cell locking (DFCL), which improves the processing of concurrent transactions. DFCL improves committed transactions, response time, throughput, concurrency, and consequently database performance and reduces rolled-back ones. In addition, it eliminates the deadlock problem of locking algorithms such (as 2PL). So DFCL is a good algorithm to be applied in situations with a high degree of concurrency [3].

In order to store every cell with its individual status and every transaction with all of its relevant information, DFCL has significant storage needs. As a result, DFCL achieves both increased space complexity and decreased time complexity [3].

The locking activities between several transactions are carried out by a novel locking protocol that is proposed in the paper. The locking and versioning system has been updated by the authors' novel contribution. As a result, synchronous locking activity between various transactions is made possible. The advantage of multiple versions and different granularity locking protocols are integrated, which is the core insight underlying the novel approach that is being proposed [4].

It is suitable for a database environment such that simultaneous execution of reading and write operations can be carried out by the multiple transactions on the same data item in the hierarchical structure. As a result, the study proposes to enhance system performance by removing the waiting requirement of read-only transactions [5].

Reasons behind Choosing the paper

The world currently depends on information systems, and the client as well as server applications and distributed databases are becoming more and more necessary as the need for easily accessible, trustworthy, and secure information in the commercial environment of the present decade develops.

Using two-phase locking for transaction processing in a distributed database management system results in a lot of conflicts. This is brought on by lengthy transaction execution times, frequent transaction rollbacks, and a high rate of transaction cancellations. Storage overhead rises as a result of stalemate with locking and stopped transactions. Because it is challenging to keep track of locks and queues awaiting data access, processing of these overheads is significant. Therefore, the authors claimed to create a hybridized concurrency control strategy that combines two-phase locking and timestamp ordering in order to address these issues. In a distributed database management system, this strategy improves the performance of concurrency control techniques for transaction processing.

The two-phase locking technique is not free from the deadlock in the distributed database system. It has a limitation of frequent rollbacks of transactions, time waiting for transaction execution, and a high rate of an aborted transactions. Storage overhead is increased because of deadlock with locking and blocked transactions, which also induces communication costs because of the deadlock problem. The timestamp ordering technique is free from deadlock because there is not too much waiting, blocking, and locking of transactions. The paper intends to address the above problem by developing a hybridized concurrency control technique for transaction processing using two-phase locking and a time-stamp ordering system to optimize time execution for transaction processing in the distributed database system.

The designed system makes an improvement in handling conflicts for concurrent transaction processing in a distributed database management system. It optimized time execution for transaction processing in the distributed database system.

The Object-Oriented-Analysis-Design (OOAD) technique is used to implement the system. C# programming language is used as the front end and Microsoft SQL Server 2008 as the back end. The implementation process is very easy and understandable to anyone. These are the reasons behind choosing this paper.

Conclusion

The authors of this study focused on client and server machines' respective roles in distributed databases' transaction processing using concurrency control approaches. Coordinating concurrent visits to a database in a multi-user Database Management System is the task of concurrency control in a distributed database. Concurrency control allows users to access a database using multiple programs as long as each user is running independently on a separate system. This project's proposal, "A Hybridized Technique Concurrency Control Technique for Transaction Processing in a Distributed Database System," proposes an approach to handling concurrent transactions in a distributed database system that reduces wait times and outperforms the deadlock solution. In order to solve this problem, future work can focus on ways to reduce the high overhead of various concurrent algorithms. Additionally, it might improve the timestamp, multi-version, and/or optimistic performance measures. In order to provide deadlock-free cell locking and improve the handling of concurrent transactions, this study presented a 2PL upgrade. The hybrid method decreases rolled-back transactions while increasing committed transactions, response time, throughput, concurrency, and ultimately database performance. Additionally, it resolves the deadlock issue using locking techniques such (as 2PL). Therefore, using a hybrid approach is a smart idea when there is a high degree of concurrency.

Reference

[1] G. Bariyira Christopher and K. Ledisi G., “HYBRIDIZED CONCURRENCY CONTROL TECHNIQUE FOR TRANSACTION PROCESSING IN DISTRIBUTED DATABASE SYSTEM,” International Journal of Computer Science and Mobile Computing, vol. 9, no. 9, pp. 118–127, Sep. 2020, doi: 10.47760/ijcsmc.2020.v09i09.012.

[2] X. Yuan and J. Yang, “Effective concurrency testing for distributed systems,” in International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS, Mar. 2020, pp. 1141–1156. doi: 10.1145/3373376.3378484.

[3] M. Mohamed, M. Badawy, and A. EL-Sayed, “An improved algorithm for database concurrency control,” International Journal of Information Technology (Singapore), vol. 11, no. 1, pp. 21–30, Mar. 2019, doi: 10.1007/s41870-018-0240-y.

[4] SCAD College of Engineering and Technology and Institute of Electrical and Electronics Engineers, Proceedings of the 4th International Conference on Trends in Electronics and Informatics (ICOEI 2020) : 15-17, June 2020.

[5] P.-Å. Larson, S. Blanas, C. Diaconu, C. Freedman, J. M. Patel, and M. Zwilling, “High-Performance Concurrency Control Mechanisms for Main-Memory Databases,” 2150.