Effective Technique for Optimizing Timestamp Ordering in Read-Write/Write-Write Operations

Obi, Uchenna M[1], Nwokorie, Euphemia C[2], Enwerem, Udochukwu C[3], Iwuchukwu, Vitalis C[4]

[1]Lecturer, Department of Computer Science, Federal University of Technology, Owerri, Nigeria
[2]Senior Lecturer, Department of Computer Science, Federal University of Technology, Owerri, Nigeria
[3]Lecturer, Department of Computer Science, Federal University of Technology, Owerri, Nigeria
[4]Lecturer, Department of Computer Science, Federal University of Technology, Owerri, Nigeria

---

Abstract - *In recent times, the use of big data has been the trending technology, most enterprises tend to adopt large databases, but the inherent problem is how to ensure serializability in concurrent transactions that may want to access the data in the database so as to maintain its data integrity and not to compromise it. The aim of this research is to develop an efficient Timestamp Ordering algorithm and model in conflicting operations in read-write/write-write data synchronization. In read-write synchronization, one of the operations to perform is read while the other is write operation. While in the write-write synchronization, both operations are trying to access the same data item and both of them are write operations. If multiple transactions modify the same data item, the integrity of the database might be compromised if there is no proper control mechanism. Timestamp and C4.5 machine learning algorithm was used to develop an algorithm that ensures serialization and consistency of database. We adopted object Oriented Analysis and Design Methodology in the analysis of effective technique for optimization timestamp ordering scheduler in RW/WW synchronization. The proposed hybrid algorithm was able to carry out the RW and WW transactions in minimal time. It optimizes the cost of reading or writing in large datasets and enable speed access in the database system with huge volume of data; thus, eliminating the problem of coordinating concurrent access to a database system.*

Key Words: **Timestamp, Concurrency, Transactions, Serializable, Conflict, Read, Write, C4.5.**

1. INTRODUCTION

A transaction is a set of instructions or user program that comprises of set of read and write operations in a database. In other words a transaction is an atomic unit of processing, that is completed entirely or not done at all [1]. In a multiuser system with large databases and large number of users executing database transactions at the same time, such as in airline reservations, railway reservations, electronic fund transfer, stock markets, online retail purchasing, online job applications and many other applications, conflicts might arise among transactions and that may create problems of database consistency. There are two kinds of synchronizations that will require concurrency control measures, these are: read-write (RW) synchronization and write-write (WW) synchronization.

Concurrency control techniques are employed for managing concurrent access of transactions on a particular data item by ensuring serializable executions or to avoid interference among transactions and thereby helps in avoiding errors and maintain consistency of the database. Various concurrency control techniques have been developed by different researchers and these techniques are distinct and unique in their own methods and representations [2].

Over the last few decades, processing of transactions and concurrency control issues have played a vital role in modern databases and have been a crucial research area. After studying relevant proposed algorithms, Arun and Ajay (2010) found out that there are compositions of only a few algorithms used by DBMS, two basic techniques used to form various concurrency control algorithms are: 2-phase locking (2PL) and timestamp ordering (TO) [3].

2-Phase Locking(2PL) algorithm employs the mechanism of preventing multiple transactions from accessing data items simultaneously by locking data items, 2PL protocols are mostly used in commercial DBMSs but transactions encounter problems like long waiting time (blocking) and deadlock which requires deadlock detection, prevention and avoidance mechanisms. While timestamps ordering (TO) does not use locks, therefore deadlock cannot occur.

A timestamp is a distinct identifier for every transaction created by the database management system (DBMS). At the start of every transaction, Timestamps values are generated base on system clock or a logical counter.

Elmasri and Navathe[4] states that timestamp ordering protocol ensures that transactions with older timestamps get higher priority in the event of conflicting operations on a particular data item and for every data item, two timestamp are given: W-timestamp which is the last timestamp of write operation performed successfully on a data item and R-timestamp which is the last timestamp of read operation performed successfully on a data item [5]. Read/write operations can only take place if the last update on that data item was done by a transaction with an older timestamp value. Else, transaction requesting read/write is aborted and given a new timestamp. The rule is that whenever we have a data item (X) and some transactions T1, T2 and T3 try to issue a read operation on data item R(X) or a write operation on data item W(X), the timestamp of T1 will be compared with the R-timestamp of data item R-TS(X) and the W-timestamp of data item W-TS(X) to make sure that the transaction timestamp order is not violated. If the order is violated, then transaction T1 will be aborted, rolled back and restarts as a fresh transaction with a new generated timestamp value, the next transaction T2 which might have used the value written by transaction T1 will also be rolled back. On the same note, T3 which might have made use of a value written by transaction T2 will also be rolled back, and so on. This resulting effect is called cascading rollback and it is one of the major problems of basic timestamp [2]. Timestamp ordering algorithm can give efficient results for concurrent control problems when provided with relevant information of transaction of the database [6]. Therefor a hybrid algorithm will be developed using intelligent query optimizer and timestamp ordering techniques to control concurrency problems in a database system. The intelligent query optimizer will be developed with C4.5 Machine Learning Algorithm. The intelligent query optimizer will provide timestamp ordering algorithm with previous information about the transaction in the database system needed to efficiently control concurrent problems.

1.1 Overview of Timestamps and their origin

Generally, timestamps are digital representation of specific process recorded in time and are created at the start of executing codes running on a processor, the executed codes obtains the value based on the local clock assigned to the processor and are stored on the hard disk or can be added in the packets sent through the network. Time identification of when the enclosing code was carried out is normally the function of the timestamp [7]. Timestamp uniquely identifies a specific process on computer systems. Large numbers of timestamps are stored in the computer system memory and typically store them in manner that the process can be clearly identified.

Sources of timestamps include the following:

(i) **File systems**: Each file in a computer system is associated with a timestamp base on the user actions on the file either when the file was created, last read, last written or otherwise modified.

(ii) **Email**: SMTP severs add timestamps to emails that will be transmitted, therefore timestamps are associated with email messages and some messaging protocols, such as social network and SMS/MMS used in GSM also generate timestamps to each message sent across the network [8].

(iii) **Logs**: The logging mechanism keeps the record of every activity taking place in the system. System logging facilities usually log events from system processes in system logs. Every event has its own timestamp value.

(iv) **Database**: Timestamp are used to coordinate concurrent transactions accessing data item in database management system (DBMS) so as to avoid conflicts among transactions, timestamps are given to every transaction at the start of execution by the DBMS to identify each transaction. Typically, transactions obtain timestamp values in the order in which they are submitted to the system and priority of getting access to the data item is based on the timestamp order. One of the possible way timestamps are created is by the use of the current date/time value of the local clock assigned to the system processor and ensure that no two transactions are assigned with the same timestamp value. Another way to implement timestamps is the use of a logical counter that increases each time a value is assigned to a transaction. The maximum value of the logical counter may be finite. So periodically, whenever transactions stop executing for short period of time, the system reset the counter to zero [4].

1.2 Distributed Database Management System (DDBMS)

Large databases that are accessed from remote machines or remote areas are to be place in sections and stored on different machines or sites for fast and reliable data retrieval and access. Distributed Database Management System comprises of different number of sites interconnected together by a communication network [3]. Each of the sites is a centralized database that consists of:

1. Transactions (T)
2. Transaction Manager (TM)
3. Data Manager (DM)
4. Data.

Users communicate with the DBMS by the execution of transactions that may be embedded queries in an application programs written in a high level programming language, while TM supervises and manages transactions between users and the DBMS while DM manages the actual data (database).

In Distributed Database Management System, transactions interact with TMs and TMs interact with DMs while DMs interact and manage the Data. Meanwhile the interaction between TMs with each other and DMs with each other is not possible. Fig - 1.1 shows the components of DDBMS.

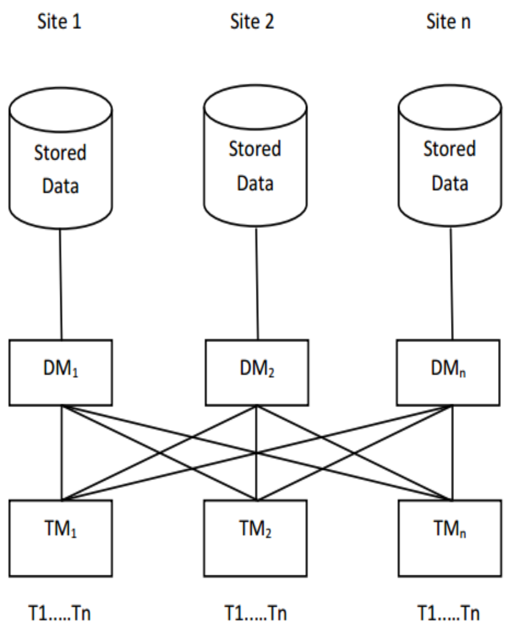


Fig - 1.1: DDBMS Architecture [9].

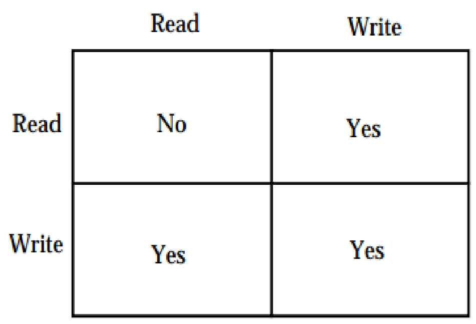
1.3 Conflicting Operations

Two transactional operations T1(A) and T2(A) are in conflict if and only if they access the same data item (A) and at least one of the operations is a write operation, i.e.

1. T1 = read (A) and T2 = write (A).
2. T1 = write(A) and T2 = read(A)
3. T1 = write(A) and T2 = write(A)

Here, read(A) is equivalent to the time of operation by transaction T_1 to read data item (A), write(A) means the time of write operation on data item (A) by transaction T1 etc.

The conflict between two operations shows that their order of execution is important, because if there is conflict between the operations of more than one transaction, their execution may result to a database inconsistency state. Note, the ordering of read operations does not matter, because they do not conflict with each other.



	Read	Write
Read	No	Yes
Write	Yes	Yes

Fig -1.2: Conflict relation of read and write operations (Design of an object-oriented framework for atomic transactions).

1.4 Schedule

A schedule is a list of operations (read, write, abort, or commit) ordered by time, performed by a set of transactions. Fig-1.2 shows a Schedule involving two Transactions T1 and T2. T1 contains two operations, read data item (A**)** and write on data item (A**)**, while T2 also contains two operations, read data item (A**)** and write on data item (A**)**. The Fig-1.3 shows two transactions executed in serial order, while Fig- 1.4 shows all the possible order of executing T1 and T2. In schedule1, the operations of T1 are executed entirely before T2 start execution. In schedule2, the operations of T2 are executed entirely before T1, which shows that schedule1 and schedule2 are serial execution. While the operations in schedule3 and schedule4 are interleaving, this means that their execution are serializable.

So far, the most accepted criterion for correctness in concurrent transactions in database system is serializabilty [10]. The concept of serializability is based on the fact that any sequential execution of transactions will leave the database in a consistence state. Serializability may be defined as the execution of certain schedules on the database which will produce the same output and effect to at least one serial execution of the same transactions [18].

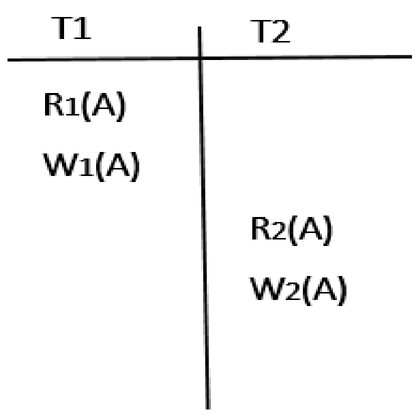


Fig-1.3: A schedule involving two Transactions T1 and T2 [11]

Schedule1 = { $R_1(A)$ $W_1(A)$ $R_2(A)$ $W_2(A)$ }

Schedule2 = { $R_2(A)$ $W_2(A)$ $R_1(A)$ $W_1(A)$ }

Schedule3 = { $R_1(A)$ $R_2(A)$ $W_1(A)$ $W_2(A)$ }

Schedule4 = { $R_2(A)$ $R_1(A)$ $W_2(A)$ $W_1(A)$ }

Fig - 1.4: The sequence of execution T1 and T2 [12]

1.5 Serial schedule

A schedule is serial if all the operations of one transaction are executed and then all the operations of another transaction, and so on. That is, for a schedule to be serial the sequence of transactions T1, T2, T3, ..., Tn, the transaction T1 must be executed entirely (committed) before the next transaction T2 starts execution. Serial schedules are naturally consistent and result in correct execution and the operations of transactions do not interleave among each other [13]. Let's consider the example where two accounts X and Y have the balances of 500 and 1000 naira, respectively, and two active transactions T1 and T2, in which funds are transferred from one account to another. Fig-1.5 shows a schedule involving T1 and T2.

T1	T2
R(x)	
x=x-100	
W(x)	
	R(x)
	T=x*0.2
	x=x-T
	W(x)
R(y)	
y=y+100	
W(y)	
	R(y)
	y=y+T
	W(y)

Fig-1.5: A schedule involving two Transactions T1 and T2 in a serializable form [12].

2. Review of Related Work

Arun et al[3] proposed a distributed architecture in distributed database system for synchronization in transaction. In their analysis, problems of concurrency control were decomposed into read-write and write-write synchronization. They stated that the performance of their proposed system will be evaluated in their future work [3].

Concurrency control method based on commitment ordering in mobile databases was proposed by Karami et al [14]. They introduced OPCT concurrent control algorithm based on optimistic concurrency control method. They provide their proposed OPCT concurrent control algorithm using serialization graph. From the results, it was shown that there was a decrease in abortion rate and waiting time when using OPCT algorithm compared to 2PL and Optimistic algorithm, but the problem of their proposed system is overhead of timestamps and their calculation.

Dagar et al. [2] in his analysis of effectiveness of concurrent control techniques in databases, they analyzed different techniques of concurrency control such as locking, timestamp and optimistic-based mechanism. They proposed optimistic-based mechanism is deadlock free techniques. It is prior to other techniques as it assumes that not too many transactions will conflicts with each other, but its waiting time is less than locking and more than timestamp ordering.

Kamal [12] analyzed transaction concurrent control for resource constrained application. They explore the previous techniques relating to timely transactional systems for remote clients and centralized database. They used the first techniques developed to decrease disk access time via local caching of state to tackle the problems of prevalent in real-time databases. They results brings gave efficient throughput to improve battery life for mobile device and minimized time complexity of the system. They suggested that it would be worth investigating the proposed approach with the new transactional phase order in non-blocking software transaction memory, order to achieve further advance in the field.

Saeid et al [15] in his proposal analyzed the modeling of timestamp ordering method with the use of coloured Petri Net. The analysis using state space shows that timestamp ordering methods illustrated in some text books may be faced with starvation. They also analyzed model explosion of state space and they observed that timestamp ordering method falls in infinite loop and therefore has starvation. They proposed concept of precedence graph produces an efficient throughput of the system. They stated that it is an open problem to start bringing up new variations of Timestamp Ordering method that inherently eliminates starvation.

In Shefali et al [16] work, issues of performance in concurrent transaction execution in DDBMS was revisited. They stated that for a distributed data, it is very difficult to find an absolute solution with respect to the following challenges: Network cost, usage of memory, processing time, response time, access time, resources, etc. They review shows that to obtain optimum solution these challenges should be minimized which would be done with the use of better algorithm for different principles of DDBMS.

Jaypalsinh et al. [6] made a proposal of a study and comparative analysis of basic Optimistic and Pessimistic management system. They highlighted on the pros and cons of pessimistic and optimistic approaches for concurrency controls. They observed that Pessimistic locking based approach is suitable for update-intensive for read operation and the optimistic method is a backward oriented concurrency control method which works on three different phases: Read, Write and Validation. They did not compare the performance of all basic methods of concurrent control to provide optimum performance of their proposed optimistic concurrency control methods.

3. Analysis of the Existing System

The problem of coordinating concurrent access in a database system has been area of interest by many researchers and several concurrent controls have been introduced. The intention of concurrency control is to ensure that the integrity of the database is preserved while allowing the execution of transactions in the system. Lomet [17] proposed a consistent time stamping for transactions in a distributed system. He stated that achieving timestamp ordering in a distributed system is potentially troublesome. In Dagar et al. [2] analysis of effectiveness of concurrent control techniques in Databases, they analyzed different techniques of concurrency control such as locking, timestamp and optimistic-based mechanism. The result showed that the latter was more efficient than lock and timestamp ordering but its waiting time is less than locking and more than timestamp ordering.

3.1 Limitations of the Existing System

The disadvantages of the existing system are as follows:

1. The traditional query engine of the database system with timestamp mechanism is time wasting, it takes a lot of time to complete transaction processes.

2. These mechanisms lack intelligence to minimize time of processing transaction in a distributed database system.

3. The problem with timestamp ordering techniques of Transaction Processing is the time complexity of its execution, most times transaction keep on waiting for the assigned timestamp of the other jobs.

4. The whole database has to be re-evaluated when data resides in main-memory, which makes the system to be lazy.

The architecture of the existing system is shown in Fig- 3.1.

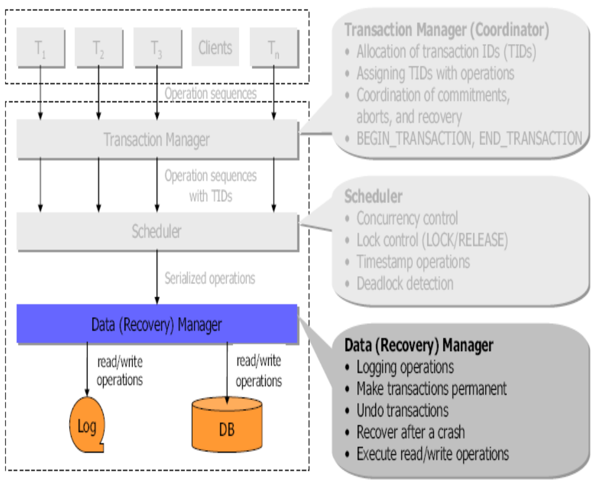


Fig-3.1: Architecture of the Existing System [2].

3.2 Analysis of the Proposed System

This work examines effective technique for optimizing timestamp ordering scheduler in Read-Write/Write-Write synchronization. We proposed a hybrid intelligent decision rule algorithm for Read-Write/Write-Write in a distributed database system using C4.5 and Timestamp ordering scheduler algorithm. A distributed relational database is a database that comprises of a number of relations and multiple physical locations or sites. Relations may be fragmented or replicated at different sites in the system. With the idea that query optimization in a distributed system should be systematic just like in a centralized system, in the sense that the optimizers should be able to use cost models to make smart and cost effective decisions such as, to determine where to fetch data from, where to execute evaluation operators and in what order. The intelligent system will ensure that data are stored at or close to the location where they are needed most. This will reduce the cost of reading or writing in a large datasets. Which is the limitation of Dagar et al. analysis and it will enable speed access in the central database distributed system with huge volume of data and complicated data structure to provide necessary information of transaction for efficient control of concurrent problems. The proposed hybrid system will eliminate the problem of coordinating concurrent access to a database system [2].

3.3 Advantages of the proposed system

1. It will produce efficient serialization order among the transactions which will be determined by the intelligent query algorithm that is sorting the corresponding timestamp vectors.

2. It will enforce efficient concurrency control base on optimal solution of intelligent query of C4.5 algorithm that provides precise dependency information obtained from the operations of the transaction.

3. The hybrid system consisting of C4.5 and Timestamp algorithm will enhance the conventional timestamp ordering eliminates the premature determination of the serialization order.

3.4 Architecture of the Proposed System

The architecture of the proposed system comprises of the transactions, transaction manager, timestamp operations, C4.5 algorithm, the data manager and the database. The diagram is shown below

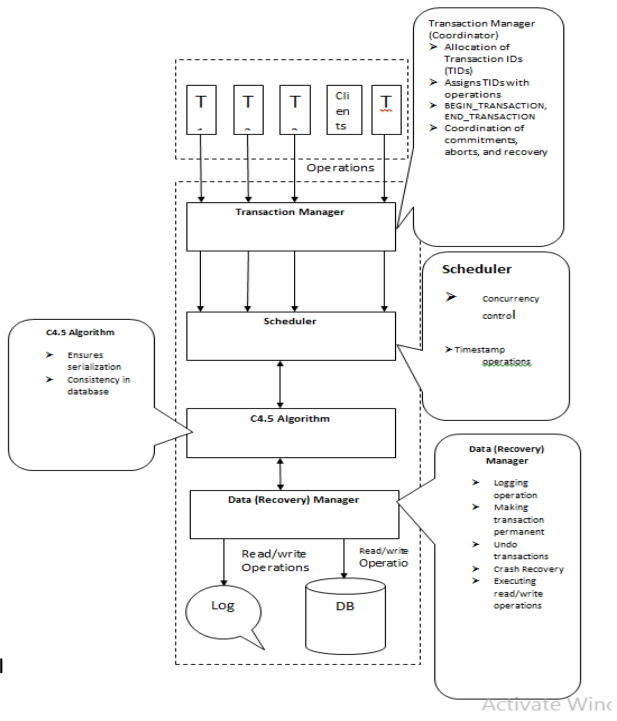


Fig -3.2: Architecture of the proposed System

From the mechanism of the Architecture, it shows that data manager ensures that starting process is given copies of all the files which it has to access and files are only to be read and not to be copied, but only set a pointer to the parents' workspace. In the log, the read/write files are modified in place, but before changing a block, record is then written to a log. This log contains details of transaction that is making a change, the file to be changed, then the old and new values. Sometimes in the cases of an abort, the write-head logs makes a simple rollback, then read the log from the end and undo all recorded changes. The transaction manager is the coordinator that allocates transaction IDs (TIDs), Assigns TIDs with operations, Coordination of commitments, aborts and recover; it begin transaction and end transaction. The scheduler ensures concurrency control and timestamp operation, it provides efficient serialized operations.

3.5 Algorithm for the proposed system.

A: Timestamp Algorithm

Case1: For read(X) of transaction T1

(a) If Ts(T1) < W-Ts(X), then T1 needs to read a value of X that was overwritten by newer transaction with larger timestamp value, hence, the read operation is rejected, and T1 is rolled back and start as a new transaction.

(b) If Ts(T1) ≥ W-Ts(X), this implies that the read operation is executed, and R-Ts(X) is set to the maximum of R-Ts(X) and Ts(T1).

Case2: For write(X) of transaction T1

(a) If Ts(T1) < R-Ts(X), this implies that the value of X that T1 is generating was needed initially, and the system assumed that the value might not be generated again. Then the write operation will be rejected and T1 is rolled back.

(b) If Ts(T1) < W-Ts(X), then T1 is attempting to write an obsolete value of X. Hence, the write operation is rejected and T1 is rolled back.

If Ts(T1) ≥ W-Ts(X) and Ts(T1) ≥ R-Ts(X), then the write operation can be performed and sets W-Ts(X) to Ts (T1).

B: Optimization Algorithm

1) Let 'T' be the set of training instances
2) Choose an attribute that will properly differentiates the instances contained in T.
3) With the chosen attribute, create a tree node whose value is the same with the attribute.
4) Form child links from this node where each individual link represents a distinct value for the chosen attribute.
5) Create subclasses with further subdivision of the Instances with the use of a child link value.
6) Do the instances in subclass satisfy predefine criteria? If yes go to step 7 else go to 2.
7) Following the decision path, specify the classification for new instances.

Stop.

3.6 The Activity Diagram of the Proposed System

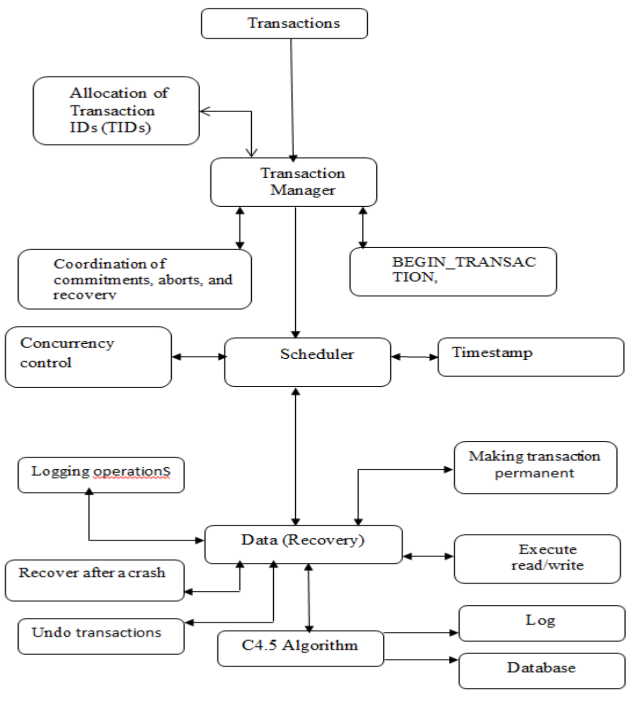


Fig - 3.3: Activity Diagram of the System

The activity diagram shows the connectivity of the various components of the proposed system. It consists of transactions to be executed, transaction manger, scheduler, data manager, C4.5 algorithm, database and log. As committed version, requests are been received by a coordinator, it will always be able to execute it because consistency check has been carried out for all operations. The mechanism of the proposed hybrid system ensures that the committed versions of an object must be created in timestamp order. The proposed hybrid system is strict such that each read operation is delayed until previous transactions that had written the object have committed or aborted.

4. Result Analysis

Transactions	Arrival time	Operations	Proposed Hybrid System Completion Time (per sec)	Existing system with basic timestamp completion time (per sec)
Transaction 1	Same	Write	0.21	0.29
Transaction 2		Write	0.28	0.30
Transaction3	Same	Write	0.21	0.31
Transaction 4		Write	0.21	0.29
Transaction 5	Same	Write	0.27	0.32
Transaction 6		Write	0.27	0.31

Table 4.1: **Analysis of Job Completion Time of Transactions**

Table 4.1 shows the analysis of job completion time of transactions. It has two columns. Column one contains transaction processed in the database. Column two contains Arrival time. Column three contains Operations performed and column four and five contains the job completion Time per milliseconds. The results are represented graphically in Fig - 4.1.

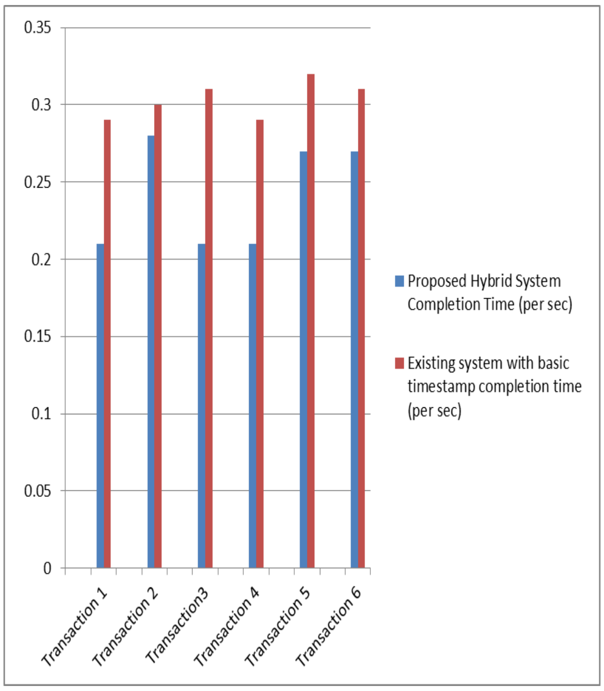


Chart 4.1: Analysis of Job Completion Time of Transactions

4.1 Discussion of Results

Transactions were simulated simultaneously to analyze the performance of the proposed system, where Write/Write operations were performed. Six (6) transactions were simulated which we calculated the transaction completion time. From the result of Read/Write operations of the transactions, the Read and Write transactions arrived at the same time. The system enables the Read transaction to read the values of database and make copy to a private workspace place where it does the modifications and makes local copies of its modifications in the private workplace only and not to the database. This is done in the Read phase of the proposed system. After the conflict checking is done in the read phase, new Timestamp Ts(Tv) is given to the active transaction, it enters into the write phase where the modified values in the private workspace updates the database, once a transaction is in the write phase, it is considered to be complete. In the write phase, transaction makes all its updates permanent in the database. This is applicable to the rest of the simulations. The experimental results show significant performance of the proposed system with respect to system throughput, and response time as the transaction operation of the Read-Write-Validate approach is deployed in the database system.

5. Conclusion

This work examines effective technique for optimization timestamp ordering of transactions in RW/WW synchronization. We proposed a hybrid intelligent decision rule algorithm for RW/WW in a distributed database system using C4.5 and Timestamp ordering algorithm. The results enabled speed access in the central database distributed system with huge volume of data and provide necessary information of transaction for timestamp efficient control of concurrent problems in complicated data structure. The proposed hybrid system eliminates the problem of coordinating concurrent access in a database system.

5.1 Recommendation

An improvement on the optimistic concurrency control algorithm (OCCA) will be recommended for similar projects. The basic aim of OCCA is to let transactions execute freely with the presumption that most of its transactions will not conflict, but in areas of high contention of data this assumption might fail. An improvement in the OOCA technique will be a good suggestion for future research.

REFERENCES

[1] A. Habes, and R. Hasan, "Multiversion Thomas' Write Rule Timestamp-based Concurrency Control," International Journal of Computer Science Issues, Vol. 12, ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784, 2015.

[2] R. Dagar and B. Rachna, "Analysis of Effectiveness of Concurrency Control Techniques in Databases," International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181. Vol. 1(5), 2012.

[3] K. Arun and A. Ajay, "A Distributed Architecture for Transactions Synchronization in Distributed Database Systems," International Journal on Computer Science and Engineering Vol. 02(6); 84-91, 2010.

[4] R. Elmasri and B. Navathe, "Fundamentals of Database Systems," sixth edition. Includes bibliographical references and index. ISBN 13: 978-0-136-08620-8, 2011.

[5] K. Sonal, and R. Morena, "Analysis and Comparison of Concurrency Control Techniques," International Journal of Advanced Research in Computer and Communication Engineering Vol. 4(3); ISSN (Print): 2319-5940, 2015.

[6] A. Jaypalsinh, and M. Prashant , " Study and Comparative Analysis of Basic Pessimistic and Optimistic Concurrency Control Methods for Database Management System," International Journal of Advanced Research in Computer and Communication Engineering Vol. 5(1), 2016.

[7] Y. Svein, "Methods for Enhancement of Timestamp Evidence in Digital Investigations," Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering Department of Telematics, 2008.

[8] D. Brian, "A hypothesis-based approach to digital forensic investigations," CERIAS Tech Report, Purdue University, 2006.

[9] C. Rinki, M Suman, R. Rathy and B. Preeti, "Recoverable Timestamping Approach for Concurrency Control in Distributed Database," International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, Vol. 3(7); 2707-2711, 2011.

[10] C. Papadimitriou, "The serializability of concurrent database updates," J, ACM Vol. 26(4): 131-153, 1979.

[11] R. Raghu and G. Johannes, "Database Management Systems," Third Edition, Includes index. ISBN 0-07-246563-8, 2003.

[12] S. Kamal, "Transactional Concurrency Control for Resource Constrained Applications," School of Computing Science, Newcastle University, 2014.

[13] U. Aydonat, "Relaxing Concurrency Control in Transactional Memory," Department of Electrical and Computer Engineering University of Toronto. Vol. 6(45); ISSN (Print). 2010, Pp.074-145.

[14] A. Karami, and B. Dastjerdi, "A Concurrent Control Method Based on Commitment Ordering in Mobile Databases," International Journal of Database Management System (IJDMS) vol.3, No. 4. 2011, pp. 39-53.

[15] P. Saeid and R. Mahd, "Modeling Timestamp Ordering method using Coloured petri net," Indian Journal of Science and Technology, Vol. 8(35); ISSN (Print): 0974-6846, ISSN (Online): 0974-5645, 2015.

[16] N. Shefali and K. Samarat, "Revisited Performance Issues in Concurrent Transactions Execution in Distributed Database Management System," International Journal of Current Engineering and Scientific Research(IJCESR), Vol. 2, ISSN (PRINT): 2393-8374, 2015.

[17] D. Lomet, "The Consistent Timestamping for Transactions in Distributed Systems," Digital Equipment Corporation Research Lab, 1990.

[18] P. Bernstein, V. Hadzilacos and N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison Wesley, Reading, MA, 1987.