International Journal of Computer Science and Business Informatics

IJCSBI.ORG

Design and Analysis of Concurrency Control Mechanism Using Modified SCC-2S Algorithm in Mobile Environment

Nyo Nyo Yee
Faculty of Information and Communication Technology,
University of Technology (Yatanarpon Cyber City)
Pyin Oo Lwin Township, Mandalay Division, Myanmar

Hninn Aye Thant
Faculty of Information and Communication Technology,
University of Technology (Yatanarpon Cyber City)
Pyin Oo Lwin Township, Mandalay Division, Myanmar

ABSTRACT
With the fast progress in mobile computing technology, there is growing strongly request for processing real-time transaction in a mobile environment. In real-time database use in mobile environments, mobile hosts (mobile users; mobile clients) can access shared data without regard to their physical location and can be updated by each mobile client independently at the same time. These conditions go to inconsistency of data. Real-time database system use in mobile environments, provide consistency of data items is a challenging issue in case of concurrent access. There are several concurrency control techniques that are proposed in literature to prevent data inconsistency. General characteristics of mobile environments like mobility, low bandwidth, limited battery power, limited storage, frequent disconnections etc. makes concurrency control more difficult. This paper proposed a method that based on Modified SCC-2S Algorithm in JEE architecture. Proposed method Concurrency Control Mechanism using Modified SCC-2S Algorithm solves write-write conflict for real-time database in mobile environment. In proposed system, Fixed Host (FH) has Database System module to perform database operation. Mobile Hosts (MHs) use On-Demand Mode to request data from FH. Therefore, MH can save storage and can live as thin client. Moreover, proposed method does not need compensating transaction for roll back transaction and can reduce memory usage in FH. Besides, proposed method can reduce the number of miss deadlines and improve effectiveness for concurrent transactions in mobile environment.

Keywords
concurrency control, modified SCC-2S Algorithm, fixed host, mobile host, mobile environment

1. INTRODUCTION
In today's Information Epoch, database is essential component of any Information system and in any environment either it is traditional, distributed, centralized, real-time or mobile. Database is a structured way to organize information. To manage the database, there are several methods for accessing the database in any system. Among them, centralized databases

are persistent but are inadequate of processing with dynamic data that constantly changes [6].

Real-time Database System (RTDBS) derived from traditional database systems and provide the same capabilities, but they are defined by timing constraints associated with the transactions of the data. Since the data is associated with a period of time for which the data is valid and can be considered to represent the true state of the system at a given time [2]. Moreover, real-time database system is processing system that is designed to manage workloads whose state is constantly changing and to stop trying reliable responses [6]. RTDBS must process transactions and guarantee database consistency [2].

There are two kinds of transactions in database management system. They are read-only transactions (ROTs) and update transactions (UTs). These transactions can have four ACID properties. These properties are Atomicity, Consistency (Concurrency), Isolation (Independence) and Durability (or Permanency) [1]. An update transaction (UT) is a transaction which can perform both read and write operations on database. An ROT is a transaction that contains only read operations which do not modify data. If an ROT conflicts with a UT, the processing of ROT is delayed till the corresponding UT terminates. Also, if a UT conflicts with an ROT, the processing of UT is delayed till ROT gets the access to the objects. Also, if a UT conflicts with another UT, the processing of UT is delayed till another UT gets the access to the objects. As a result, the throughput performance (number of transactions processed per second) deteriorates as data contention increases [1].

Real-time Database System (RTDBS) used in Mobile Environment provide information to Mobile Host (Mobile User). In mobile environment, Mobile Users (Mobile Host) can initiate transactions and that transactions may be executed at Mobile User (MU) or Fixed Host (FH). Most of the transactions used in mobile environment are flat transactions. In modern world, most applications are complex and long-running and flat transactions could not work properly in these applications. Moreover, flat transactions can perform only commit or rollback and cannot save intermediate results. If transactions were rollback, the whole transaction will be restarted. To solve this problem, proposed method (Concurrency Control Mechanism Using Modified SCC-2S Algorithm) based on closed-nested transactions model because nested transactions are suited for complex application and can save intermediate result. Nowadays, Airline Businesses are very popular and most of the users want to access this information from mobile environment. To reach the desire destination, most of the airlines use transit. So, there is an issue to control concurrent access in Airline Reservation System.

Proposed method aims for controlling concurrent access in airline reservation system.

The rest of this paper is organized as follows: Section 2 present proposed method. Section 3 provides general rules for proposed method and mathematical expression for proposed method. Section 4, analyses proposed method and other concurrency control algorithms and section 5 draws the conclusion.

2. PROPOSED METHOD

In proposed system, mobile user sends query using an uplink channel (pull process). To process the request, database server (including two databases) in Fixed Host (FH) use proposed method Concurrency Control Mechanism using Modified SCC-2S Algorithm that avoids conflict (access the same data). After the database operation had performed, FH returns the result back to corresponding MH. MH does not require having Database System (DBMS) module to perform database operations. So, MH acts as a thin client.

If two or more transactions enter the system concurrently, the system uses Concurrency Control Mechanism using Modified SCC-2S Algorithm to control concurrent access. Proposed system compared proposed method with Two-shadow Speculative Concurrency Control (SCC-2S). SCC-2S require standby shadow if conflicts occur between transactions. In SCC-2S, transaction with late time creates standby shadow. Standby shadow means the copy of the original query that does not contain the portion of the query that the primary shadow is already performed. Standby shadow creations require extra processing power and resources. It is not suitable for mobile environment either concurrency control is performed MHs or FH. Because MHs has limited storage, concurrency control is performed in MH is not suitable using SCC-2S. Similarly, concurrency control is performed in FH is not suitable because SCC-2S requires a lot of resource to consume so FH can become bottleneck when a lot of concurrent update occurs.

By using Modified SCC-2S Algorithm; it does not to require creating standby shadow. Hibernate is free open source software of the Object-Relational mapping (O/R mapping) tool for the Java language developed by Red Hat. O/R mapping is a programming technique for associating data between data type in Relational Databases and Object-Oriented programming languages. In Object-Oriented programming, data model is designed and implemented to manipulate objects, while Relational databases are structured for retrieving and saving data. The problem resides in how to convert the object values into database (and convert them back upon

retrieval) in spite of the difference of the principal and philosophical design. The purpose of O/R mapping is to solve the Object-Relational impedance mismatch issue and provides the seamless conversion between them reducing the typical and complicated work from developers. It can be used in matching with Struts, Spring, and Hibernate JEE (Java Enterprise Edition) framework. The work of the Synchronizer is to synchronize between threads. Transactions from mobile host enter the system as a thread. In JEE, synchronize method can be used to control the work of threads.

There are many kinds of lock used in database management system. They are database level, table level, page level, row level or field (attribute) level [4]. Proposed method performs concurrency control in row level. So, proposed system use row level lock. So, proposed method not need to lock database, table and page level. So, many transactions can perform database operation concurrently. In proposed method, transaction with late times block if conflict occurs between transactions. It cannot require creating save point and log files. Database server can perform this work automatically and can save intermediate result. If commit transaction release lock, it resumes its execution form the point that conflict occurs.

Illustration of SCC-2S works is , shown in "Fig. 1", Two mobile hosts MH1 and MH2 access the same data item x. MH1 execute Transaction T_1 to write data item x. MH2 execute Transaction T_2 to read data item x. Both transactions T_1 and T_2 start with one primary shadow, namely T_1^0 and T_2^0 respectively. When T_2^0 try to read object x, a potential conflict is discovered. At that time, a standby shadow, T_2^1, is created (means the transaction T_2^0 is blocked when transaction T_1^0 release its lock). If T_1^0 successfully validates and commits on behalf of transaction T_1, the standby shadow T_2^1 resumes its execution [5].

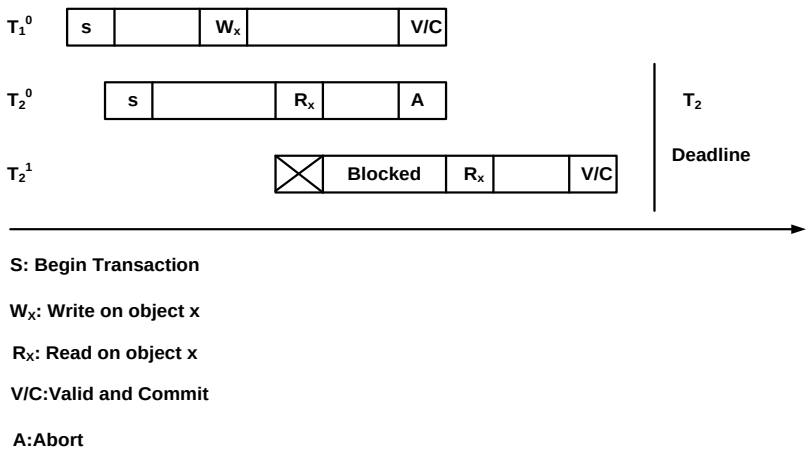

Fig. 1 Schedule with a standby shadow promotion for Read-Write conflict for SCC-2S

SCC-2S cannot solve Write-Write conflicts and it only solves Read-Write conflicts. In real world, read transactions is low priority than write (update) transactions. In SCC-2S, read transactions always create standby shadow.

On the other hand, if transaction T_2 reaches before transaction T_1, transaction T_2 reads operation cannot conflict with transaction T_1 write operation. So, transaction T_2 read data item x. But when transaction T_1 write data item x, at that time the result for transaction T_2 is wrong. At that time, transaction T_2 must create standby shadow for read data item x. When transaction T_1 commits, primary shadow for transaction T_2 is abort and standby shadow promote to become primary shadow and execution is resumed. So, transaction T_2 read the data item x two times and always check conflict with other transaction or not. Moreover, if many transaction conflicts with other transaction, there are many standby shadow creation and abortion. This leads to resource consumption. It can become more negative effect when write-write conflict occurs. The abortion for write transaction has more effect than read transaction. Really, most of the transactions used in Real-time Database System are update (write) transactions. Moreover, due to the characteristics of mobile environment the effects of abortion of write transaction more badly than other environment. So, proposed system solved Write-Write conflicts.

Illustration of how proposed method works is, shown in “Fig. 2”, The two mobile hosts MH1 and MH2 access the same data item x. MH1 execute Transaction T_1 to write data item x. MH2 execute Transaction T_2 to write data item x. In our proposed method, both transactions T_1 and T_2 start working. When T_2 attempts to write object x, a potential conflict is detected. At this point, T_2 is stop working and store previous perform result. This is done Database Management System automatically. If T_1 successfully validates and commits, transaction T_2 resumes its execution. There is no need for standby shadow creation. So, it reduces standby shadow creation time and memory usage.

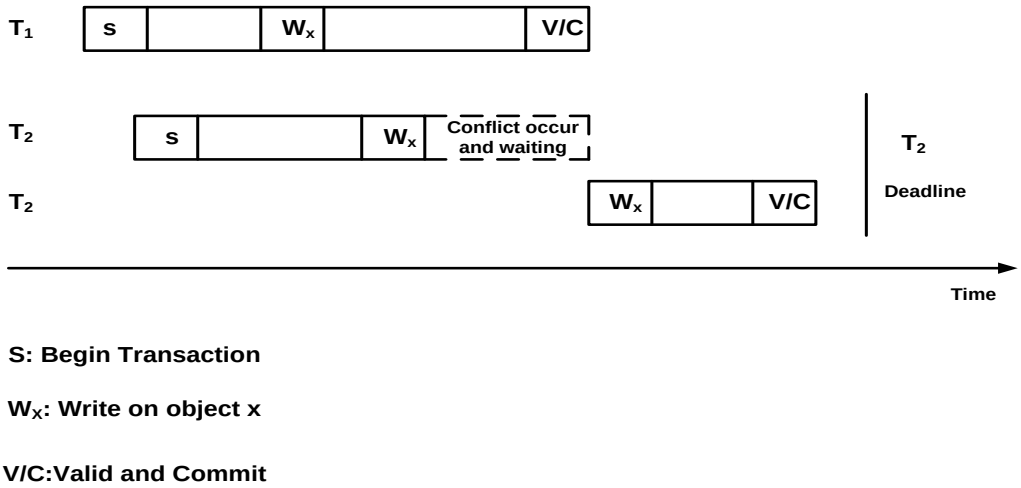

Fig. 2 Schedule with Write-Write conflict for proposed method

"Fig. 3" defines incoming transactions with their time. MH1 executes Transaction T_1 want to go Nay Pyi Taw (NPT) to Brunei Darussalam (BWN). There is no direct flight for NPT to BWN. So, use three transits: NPT to Yangon (RGN), RGN to Bangkok (DMK), and DMK to BWN. MH2 executes Transaction T_2 want to go Bagan Nyaung-U (NYU) to Bali (DPS).There is no direct flight. So, use three transits: NYU to RGN, RGN to DMK, and DMK to DPS. Conflict occurs in RGN to DMK transit.

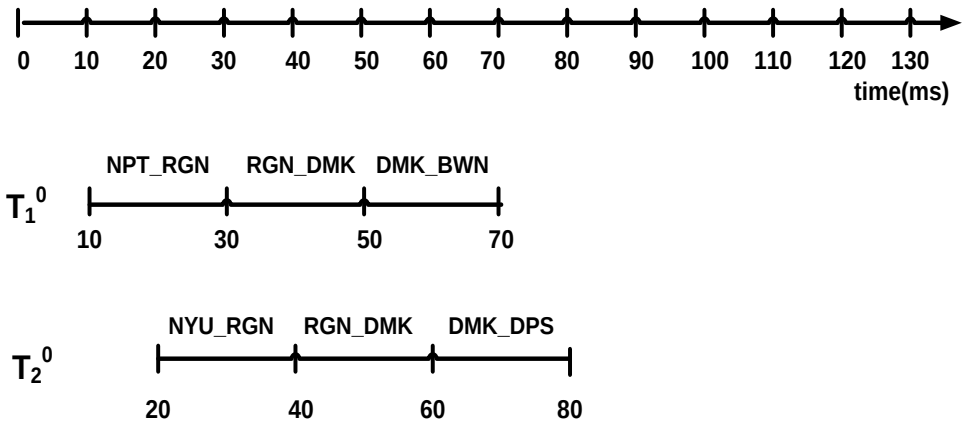

Fig. 3 Transaction with their time

For conflict RGN_DMK transit, proposed method Concurrency Control Mechanism using Modified SCC-2S Algorithm does not need to create standby shadow and wait the late conflict transaction to complete the first

transaction. The reach time for Transaction T_2 (MH2) for RGN-DMK transit is later than Transaction T_1 (MH1). So, MH2 wait MH1 finish it process at the point if conflicts occur. After finish transaction T_1 (MH1) transaction T_2 (MH2) resumes its operation is shown in fig 4.

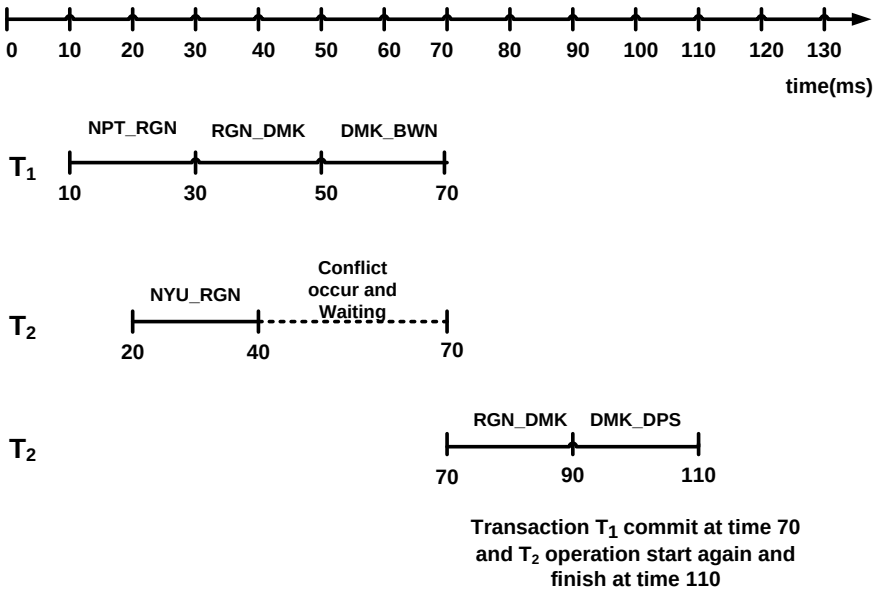

Fig. 4 Write-write conflict solves by proposed method

Due to the nature of network latency, processing speed and other nature, etc conflicts can occur different time or at the same time. Moreover, in these days most of the concurrency control mechanisms consider priority theory to provide high response and throughput. So, proposed method added rules for the conflicts occur at the same time. To illustrate proposed method, proposed system use two databases in FH. Proposed method assume all require data for transaction T_1 (MH1) can get only one database and require data for transaction T_2 (MH2) need more than one database. Transaction T_2 waits the end of transaction T_1 and save immediate results. ““Fig. 5” shows the two mobile hosts MH1 and MH2 access the same data item at the same time and “Fig. 6” illustrates proposed method with examples.

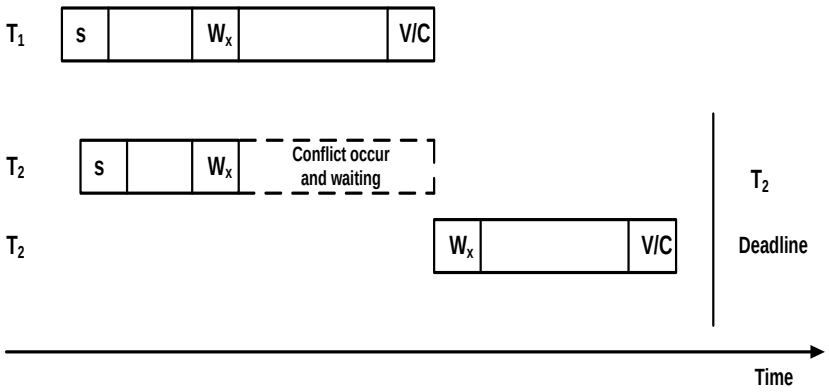

Fig. 5 Schedule with Write-Write conflict at the same time by proposed method

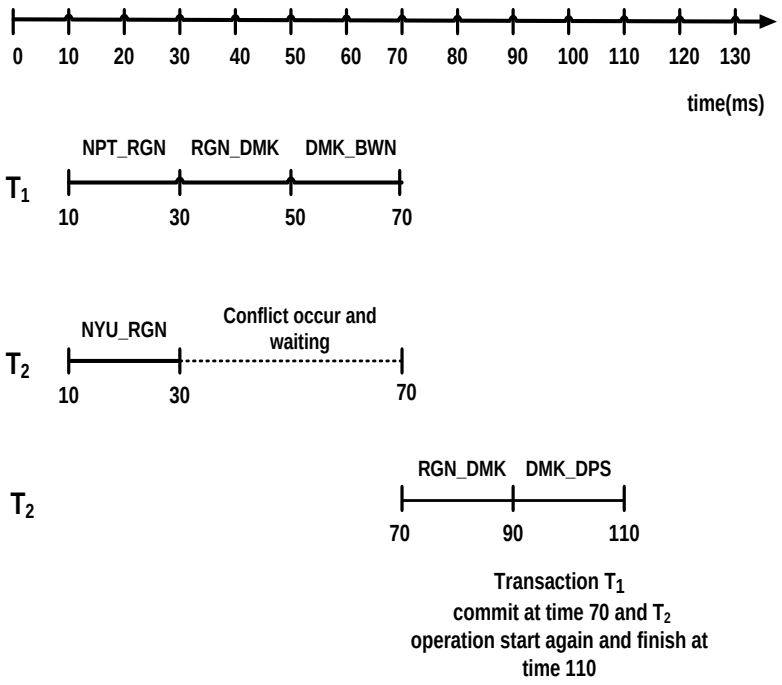

Fig. 6 Write-Write conflict at the same time solves by proposed method

3. GENERAL RULES FOR PROPOSED METHOD (CONCURRENCY CONTROL MECHANISM USING MODIFIED SCC-2S ALGORITHM)

Proposed method is applied in AirLine Reservation System and use two database for propose method. Proposed method use fId for flightId, rId for routeId, date for reservation_date, r for reach time and n for no of transit count.

```
Begin
  Income write lock request transaction Twt
 Add Twt into object array named list
 for (int  i=0;i<list.length-1;i++){
   for (int j=i+1;j<list.length;j++){
     If(list[i].fId&&rId&&date ≠ list[j]. fId&&rId&&date)
       Transactions run concurrently.
    Elseif (list[i].fId&&rId&&date==list[j]. fId&&rId&&date)
       If (list[i]. fId&&rId.r < list[j]. fId&&rId.r)
          wait list[j]
       Else if (list[i]. fId&&rId.r == list[j]. fId&&rId.r)
         Check data can get only one database or not
         If (list[i] access only one database and list[j] access more than
        one database)
            wait list[j]
         Else if (list[i] and list[j] access only one database)
            Check n
            If(n of list[i]> n of list[j])
              wait List[i]
            Else
              wait List[j]
         Elseif(list[i] and list[j] access more than one database)
             Check n
             If(n of list[i]> n of list[j])
                wait list[i]
             Else
                wait list[j]
         Else
            wait list[i]
      Else
         wait list[i]
     End if
    }
  }
End
```

3.1 Mathematical Expression for Proposed Method

Let $T = T_1, T_2, T_3, \ldots, T_m$ be the set of uncommitted transactions in the system. For each conflict transaction T_r in the system , a set WaitFor(T_r) is maintained, which contains a list of tuples of the form (T_s , x), where $T_s \in T$ and x is an object of the shared database. (T_s, x) $\in$ WaitFor (T_r) implies that T_r must wait for T_s before being allowed to read or write object x. The notation (T_s, -) $\in$ WaitFor(T_r) is used where there exists at least one tuple (T_s , x) $\in$ WaitFor(T_r), for some object x. Details of the Concurrency Control Mechanism using Modified SCC-2S Algorithm is defined as follows: detected, and the way they are resolved.

- New transaction T_r is requested for execution, it execute without any interrupt.

Mathematical expression for write/write conflict

- Whenever a transaction T_r wishes to write an object x that has been written by Transaction T_s, if the time of transaction T_r write an object x is a little late than the time of transaction T_s write an object x then,
  - If (T_s, x) $\notin$ WaitFor (T_r) then add transaction T_r in the waiting list such as (T_s, x) $\in$ WaitFor (T_r).
  - Transaction T_r must wait Transaction T_s commit time and save it intermediate result in the log file automatically (Database Server can perform this work automatically). When Transaction T_s commit, release all of it's acquire lock and transfer its locks to the transactions in the WaitFor list. At that time, transaction T_r resume its execution and acquire the require lock.

Mathematical expression for write/write conflict at the same time

- Whenever a transaction T_r wishes to write an object x that has been written by Transaction T_s. The two transactions T_r and T_s write the same data object x at the same time. Proposed method assume that transaction T_s access only one database module and transaction T_r access more than one database module.
  - If (T_s, x) $\notin$ WaitFor (T_r) then add transaction T_r in the waiting list such as (T_s, x) $\in$ WaitFor (T_r).
  - Transaction T_r must wait Transaction T_s commit time and save it intermediate result in the log file automatically (Database Server can perform this work automatically). When Transaction T_s commit, release all of it's acquire lock and transfer its locks to the transactions in the WaitFor list. At that time, transaction resume its execution and acquire the require lock.

- Whenever it is decided to commit transaction T_s, then release all of it's acquire lock and transfer it locks to the transactions in the WaitFor list.

4. ANALYSIS OF CONCURRENCY CONTROL USING PROPOSED METHOD AND OTHER CONCURRENCY CONTROL ALGORITHMS

Real-time database management system is a combination of conventional database management system and real-time system. Like other database system, real-time database system can process transactions and guarantee database consistency. Furthermore, this database system can operate in real-time that satisfy time constraints on each transaction.

Existing concurrency control algorithms for conventional database systems attempt to maximize concurrency, but ignore timing constraints. Deadline scheduling algorithms for conventional real-time systems do consider timing constraint, but ignore data consistency problems. Since concurrency control algorithms may introduce unpredictable delay due to transaction restarts and blocking, there is clearly a real need for a concurrency control model that combines the timeliness of deadline scheduling algorithms and the data consistency provided by conventional concurrency control algorithms [6].

Various concurrency control algorithms differ from the time when conflicts are detected, and the way they are resolved. Most of concurrency control method based on Pessimistic Concurrency Control (PCC) and Optimistic Concurrency Control (OCC). But in mobile environment, most of the method based on OCC. In OCC, each transaction perform database operation using three distinct phases- read phase, validation phase and write phase. Moreover, OCC can only detect conflicts at transaction commit time and resolve these conflicts by restarting conflict transactions. Moreover, most of the concurrency control method used in mobile environment lead to Mobile Ad-hoc Network. In this network MHs perform database operation. It is good for controlling central bottleneck. But it requires a lot of other things to increase performance and reduce system performance. So, proposed system performs database operation at fixed host. To use OCC, MHs have Database System Module to perform database operations. After finishing database operation, MHs send result back to FH to check conflicts or not. If conflicts occur between transactions, only one MH write request is performed and other MHs must perform database operations again.

Pessimistic Concurrency Control (PCC) is based on two phase locking protocol. In this method, concurrent users access a row and only one user

can get this row. That row is unavailable to other users until the acquired user release that row . So, if conflicts occur between transactions, conflicted transactions perform database operation again. PCC can detect conflicts immediately when they occur and resolve these conflicts using blocking. PCC algorithm is mostly useful in situations where it is harmful for the record to change during transaction processing time. But, PCC is not useful in a disconnected architecture. To use PCC in Mobile Environment, connections are open only long enough to read or update the data, so this method requires sustaining locks for long periods. PCC blocking based conflict resolution policy require a lot of resource. Moreover, this method can miss the deadlines as a result of unbounded waiting due to blocking.

When the original SCC-2S method is analyzed, it can only handle read-write conflicts. When compares with read and write conflict, read priority is lower than write priority. So, in this method, read transaction only executes as a standby shadow. In real time database system, concurrent users can encounter Read-Write conflict as well as Write-Write conflict. Therefore, proposed system manipulates this condition. Moreover, concurrency control methods use priority theory to increase performance and to reduce deadline. So, proposed method also added priority theory.

Proposed method Concurrency Control Mechanism using Modified SCC-2S Algorithm completely eliminates the complicated locking problems and delay commit. This approach is useful for critical real-time database system. And also according to the MySQL server nature, it can be manipulated the concurrent 1400 users [7]. When our proposed method is used in Dynamic Web Application Architecture, it can manage more than concurrent 1400 users expected. Moreover, it is very compatible methods for not only flat transactions but also nested transactions.

4.1. Performance Result

“Fig. 7” shows performance metric of memory usage for ten concurrent users and “Fig.8” shows performance metric of response time for ten concurrent users.

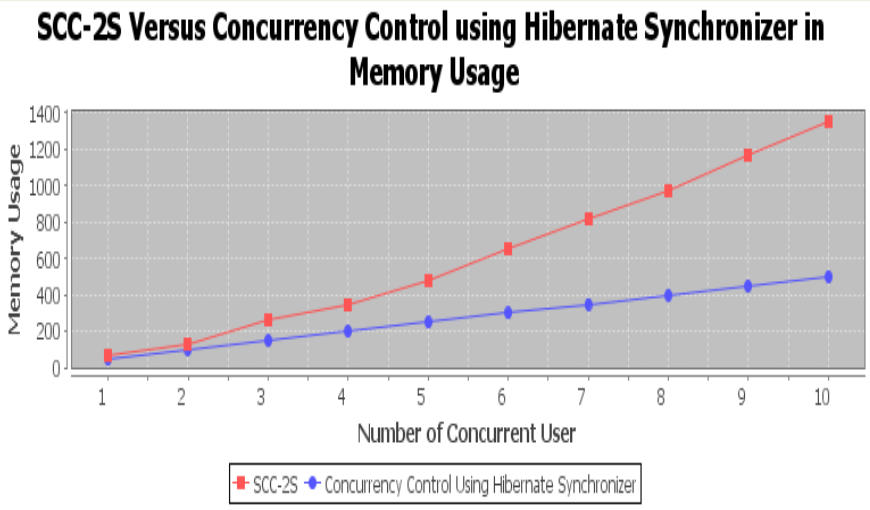

Fig. 7 Memory Usage for ten concurrent users

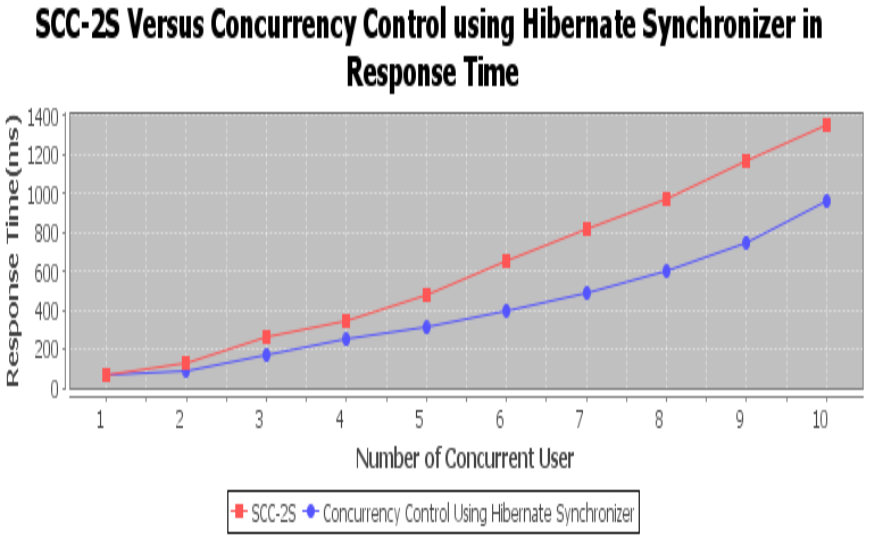

Fig. 8 Response time for ten concurrent users

5. CONCLUSIONS

Most concurrency control method used for real-time database system in mobile environment solves consistency issue (concurrent access). Proposed method provides high respond time and throughput. Moreover, proposed method decreases the number of missed deadlines; reduce battery power and memory usage in the system. Moreover, MHs cannot require to have database system module and MHs can live thin clients.

REFERENCES

[1] T. Ragunathan, *Speculation-Based Protocols for Improving the Performance of Read-Only Transactions*, Center for Data Engineering, International Institute of Information Technology, Hyderabad, India, December 2010.

[2] S. A. Bukhari, and S. R. Aparicio, *A Survey of Current Priority Assignment Policies (PAP) and Concurrency Control Protocols (CCP) in Real-Time Database Systems (RTBDS)*, 2012.

[3] S. A. Moiz, S. N. Pal, J. Kumar, P. Lavanya, D. C. Joshi, and G. Venkataswamy, *Concurrency Control in Mobile Environments: Issues & Challenges, International Journal of Database Management Systems (IJDMS)*, vol.3, no.4, November 2011.

[4] P. Rob, and C. Coronelm, *Database System: Design, Implementation, and Management*, 8th Edition, Course Technology, Cengage Learning, ISBN- 13: 978-1-4239-0201-0, ISBN- 10: 1-4239-0201-7, USA, 2009, pp.412-440, pp.494.

[5] A. Bestavros, S. Braoudakis, and E. Panagos, *Performance Evaluation of Two-Shadow Speculative Concurrency Control*, Computer Science Department, Boston University, Boston, MA 02215, February 1993.

[6] V. Swaroop, G. K. Gupta, and U. Shanker, *Issues In Mobile Distributed Real Time Databases: Performance And Review* , India, 2011.

[7] A. DIN, *Structured Query Language (SQL) A Practical Introduction*, University of Rome La Sapienza, May 1994, http://www.dis.uniroma1.it.

This paper may be cited as:

Nyo. N. Y. and Hninn. A. T., 2014. Design and Analysis of Concurrency Control Mechanism Using Modified SCC-2S Algorithm in Mobile Environment. *International Journal of Computer Science and Business Informatics, Vol. 14, No. 2, pp. 104-117.*