RECOVERY ALGORITHM FOR TRANSACTION FRAMEWORK IN WIRELESS MOBILE ENVIRONMENT

MIRACLIN JOYCE PAMILA J.C. [1]

Department of Computer Science and Engineering,
Government College of Technology,
Coimbatore, Tamilnadu, INDIA.
miraclin_2000@yahoo.com

THANUSHKODI K [2]

Akshaya College of Engineering and Technology,
Coimbatore, Tamilnadu, INDIA.
thanush_dr@rediffmail.com

ABSTRACT

Data processing capability and data storage capacity of mobile handheld devices facilitate transaction processing. Normally mobile internet transactions enable multiple users to involve in on-line transaction processing independent of their physical location. The mobile network is prone to frequent failures due to its constrained bandwidth available and mobility of mobile hosts. Users execute transaction with a web portal from a mobile device and the disconnection will make the transaction to fail completely or redo all the steps after reconnection to get back into the consistent state. Thus present mobile computing environment poses a new challenging problem in designing new recovery schemes for mobile transactions. So a recovery algorithm is proposed that ensures stable state recovery when mobile device gets reconnected after the disruption in execution of the transaction. The proposed framework ensures failure recovery algorithms that recover the stable application state with dependency preservation between transactions and with reduced recovery cost.

Keywords: Dependency preservation, Transaction recovery, Log based recovery, Check pointing, Time stamp ordering.

1. INTRODUCTION

The increasing popularity of mobile devices and improvement in the quality of service (QOS) provided by wireless mobile services, have made commercial transaction execution possible from the mobile devices. When more number of Mobile Clients (MC) involve in database transaction with a web portal from number of mobile devices (such as online ticket reservation), failure of any one transaction initiated at any mobile device may affect the transaction on another mobile device based on the state in which transaction failed. If the active transactions have conflicting operations with failed transaction, recovery strategies must be properly figured out to preserve dependencies between transactions.

In this paper a failure recovery model is proposed which recovers the failed transaction to a consistent state so that transaction execution may continue from a stable state.

The remainder of the paper is organized as follows: Section 2 focuses on recovery problem specification. Section 3 discusses about the reference architecture. Section 4 explains the transaction framework and the recovery algorithm while Section 5 analyses the performance. Section 6 concludes the presentation.

2. RECOVERY PROBLEM SPECIFICATION

2.1 Transaction Execution:

The transaction is initiated and terminated in the same mobile device and assumed to be short lived. The log is maintained in the mobile device itself to keep track of the progressive state of the transaction. Forced write operation is performed on to the log on handoff or device failure due to battery shut down.

The transactions run in mobile devices normally access the data items from the data base server. Simultaneous access requests made to database server overloads it. So to avoid such server overloading and to increase query response time, caches are used where data is replicated.

Transactions normally make Read Data Request RDR (Data_id, Time_Stamp) or Update Data Request UDR(Data_id, Time_Stamp, Value). So when read data request is made, the available data can be fetched from the cache itself. When update request is made the request is forwarded to the database server that confirms updation of data in the database server. As a result of updation, the Database server responds with Update Data Confirm (UDC). On receipt of this the transaction commits with Transaction Commit. So log keeps record of all requests made by the mobile device and all responses received by them.

2.2 Dependency between Transactions

All transactions that make RDR will be allowed to access the data simultaneously. But when a transaction issues UDR, then all other transactions that make read access to the same data become dependent transactions on the transaction that updates the shared data item. So the proposed recovery algorithm for failure recovery ensures preservation of this dependency property**.**

Here the transaction T2 starts before transaction T1 issues update request. Transaction T1 fails before update confirmation. Since T2 has not read the data item that has been modified, they are independent of each other. But Transaction T3 reads after transaction T2 has sent updation request. So Transaction T3 depends on T2.

If transaction T2 fails without updation confirmation then T3 should also roll back.

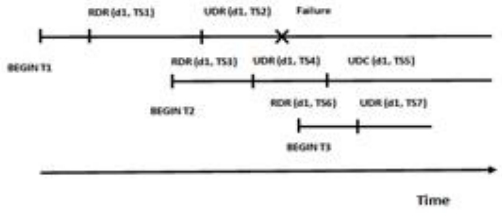

Fig 2.1Dependency between Transactions

The Recovery manager should check for dependency between transactions based on the data being accessed and the timestamp of the access.

3. REFERENCE ARCHITECTURE

Fig. 3.1 illustrates the reference architecture. It is a client/server architecture system based on a digital mobile communication system, a Global System for Mobile communication [12] network. GSM is improved to provide flexible and powerful packet oriented data service termed as General Packet Radio Service. As shown in the figure a mobile wireless network contains two basic components, a fixed backbone network and wireless network.

A host that can move while retaining its network connection with the fixed backbone is a mobile host. Mobile host (MH) comprises all user equipment and software needed for communication with a Global System for Mobile communication (GSM) network. The mobile host has notable storage capacity and computational capability.

A static network consists of fixed hosts and base stations (BS) that will interact with mobile host and with wired network. Base station is a fixed host in a radio subsystem of GSM for radio communication with mobile host and it will act as a gateway between wired and wireless networks. Base Station (BS) comprises all radio equipments such as transmitter, receiver and signal processing amplifiers necessary for radio transmission and reception. It forms a radio cell, a defined geographical region within which it can provide services to the mobile hosts. Base station is connected to the mobile host via radio link and is connected to Base Station Controller via a high speed wired link. Due to mobility the MH may cross the boundary of the cell and this process is known as handoff.

Two or more base stations are controlled by a Base Station Controller (BSC). It reserves radio frequencies, handles handovers from one base station to another within same Base station Subsystem. Similarly Mobile Station Switching Center (MSC) will control two are more BSC's.

Gateway GPRS support node is a networking unit between the GPRS network and external packet data networks (PDN). This node contains routing information for GPRS users, performs address conversion, and tunnels data to user via encapsulation. GGSN is connected external networks such as IP or X.25 via the G_i interface and transfers packets to the SGSN via an IP based GPRS backbone network (G_n interface). The other element Serving GPRS Support Node SGSN supports the mobile host via the G_b interface. It requests user addresses from the GPRS register (GR), keeps track of the individual mobile host's location, and is responsible for billing information and performs several security functions such as access control. The SGSN is connected to a BSC via a frame relay and is basically on the same hierarchy level as the MSC. So packet is transmitted from a packet data network, via GGSN and SGSN directly to the mobile host through base station.

Before sending any data over the GPRS network, a mobile host must attach to it , with mobility management procedures. This procedure assigns a temporal identifier called a temporal logical link identity (TLLI) and a ciphering key sequence number (CKSN) for data encryption. For each mobile host a GPRS context is set up and stored in the MS and in the corresponding SGSN. This context comprises the status of the mobile host , the CKSN, a flag indicating whether compression is used and routing data.

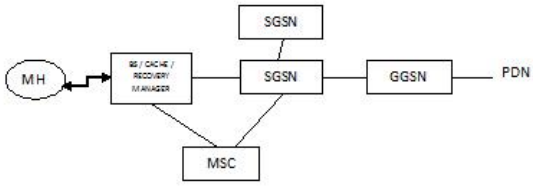

Fig 3.1. Reference Architecture

In the proposed architecture, the base station, a fixed host is found be suitable for storing a cache. A Cache Manager (CM) module in it monitors and coordinates the operations in a cache. The fixed host also has a Recovery Manager (RM) module that ensures recovery of failed transaction. A log manager module loaded in the mobile client itself aids in recovery of the failed transaction.

4. TRANSACTION EXECUTION FRAMEWORK

The mobile devices which may involve in simultaneous transaction execution, normally request data from the database server itself. Since many mobile clients may involve in simultaneous transaction processing (like railway reservation), multiple simultaneous access requests and response may increase network overhead. So to reduce such overhead, the proposed framework assumes that the fixed hosts that serve the mobile hosts may have cache in it. The cache manager module loaded in it will monitor concurrency control[2][3][4][5], cache data invalidation[6][7][8][9] and recovery of dependent transactions[10].

4.1 Caching Strategy

In the reference framework[12] the cache manager module implemented in base station initially fetches the data item from the database server and stores it in the cache. All subsequent requests for the data item will be serviced by the cache manager itself.

The cache manager stores the data as quintuple in the cache. Initially when the cache manager retrieves the data item from the server, it sets Last Updated Time to the current time, Predicted Life Period to optimal life time based on the nature of the data item (if it is stock quote then PLP may be few seconds or if it is available inventory on any commercial product then few minutes). The number of affiliated transactions is set to 1. When subsequent data access requests are made, Affiliated Transaction Count is incremented. The data in the cache is valid as long as it is not updated in the server.

Any mobile client that needs to update the data item makes the update request. When this request reaches the BS that holds the cache, it updates the data item locally. If simultaneous requests are made then it needs to resolve it in FIFO order. Once the cache manger updates the data item locally, it immediately sends the cache invalidation indication to all the mobile clients and fixed hosts that have already accessed the data item. If almost all mobile clients participate in transaction execution, the invalidation indication is broadcasted to all mobile clients. So all other mobile and fixed clients are forced to refresh the value of the data item.

4.2 Recovery Strategy

Recovery Manager implemented in base station aids in the recovery of the failed transaction. The mobile devices that participate in transaction execution will have a log in its permanent storage. The log manager module of the recovery manager is present in every mobile device participating in the transaction execution. The log manager module ensures that every action performed by the mobile device regarding transaction is stored in the local log.

The log entries are made and stored in the permanent storage of the device as shown in figure 4.1.

Trans_id	Data_id	Type_Opern	Time_Stamp

Fig. 4.1 Format of the log entry

The Transaction id normally uniquely distinguishes the transaction. Data_id is the identity of the data item that is accessed. The operation may be Read, Update or Commit. The Time Stamp is the time instance associated with every operation of the transaction.

The Recovery Manager in the base station keeps track of the status of affiliated transactions that access the same data item. If multiple transactions have read the same data item then, the list of affiliated transactions is kept in track. In addition to it if any Update Request is made then the status of the updation is kept in track. i.e. whether the updation is confirmed or not and if confirmed at what time stamp.

4.4 Algorithm

Recovery Manager

1. Store Transaction details such as dataid, lastUpdate_Timestamp(LUT), Transaction list.
2. Retrieve log entry from the log manager
3. if (RDR) and if (TS(read) < LUT) or
   (TS(RDR) < (PLP+TS(RDR)) then
   send redo read to log manager
   else allow transaction to continue.
4. if (UDR) and if update is confirmed then commit the transaction else send redo update to log manager.

Log Manager (Normal Operation)

1. Transaction start
2. If RDR then make entry in the log <did, Trans_id, Timestamp>
3. If UDR then make entry in the log <did, Trans_id, value, Timestamp>
4. On failure force write the state of the transaction.

On Recovery

1. Send the last entry (RDR or UDR) to Recovery Manager
2. Wait for the response from the Recovery Manager.
3. If redo then redo the operation specified else continue.

5. PERFORMANCE ANALYSIS

The proposed framework is simulated and performance metrics are analyzed. The following graphs show the results of simulation.

5.1 Analysis of Recovery probability

Recovery probability refers to the probaility of the failed transaction to restore the previous consistent state.
Recovery probaility is a function of log availability and wireless link quality.

$$P_{proprec} = Pl_{og} + P_{Stat} \quad (1)$$

where P_{log} is the probability that the log is available after failure recovery and P_{Stat} is the probability with which status inforamtion reaches the recovery manager module in the fixed host. Since the log is stored in the mobile device itself the probability of the availability of data is very high. Since only the last log entry is transaferred over the wireless link it is highly probable that the status information reaches the recovery manager. In the previous approaches log is stored in the base station itself , so the probability that the log is availabe in the base station is high and the recovery status will be sent by base station ensuring high recovery probability. So the proposed method works with good

recovery probability and comparison is shown in the graph.

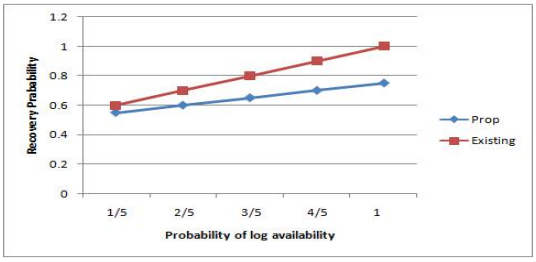

Fig 2 Analysis of Recovery Prabability

5.2 Analysis of Recovery Cost

Recovery cost of the mobile transaction is calculated as a function of log retrival cost, wireless link cost and recovery status retrieval cost.

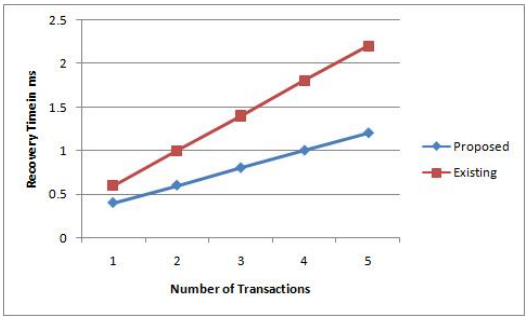

Fig. 5.2 Analysis of recovery cost

In the proposed approach the log is stored in the mobile host itself. So the log retrieval cost remains constant and the total recovery cost depends only on wireless link cost and cost of recovery status which increase in proportion to the number of transaction. But in the existing system the log is stored in the base station and the log retrieval cost is determined by the number of transactions against the constant value for the proposed framework. So the total recovery cost is decreased as the number of transactions increases.

5.3 Dependency Preservation

When transactions are simultaneously executed from various mobile hosts they are time stamped according to the Recovery Manager's Timestamp ordering.

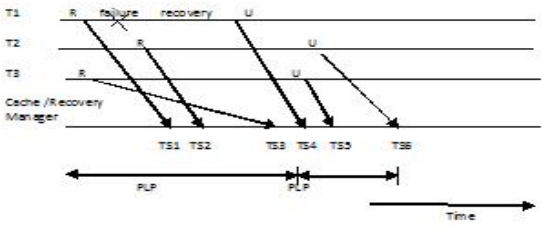

Fig 5.3 Dependency Preservation

Considering three transactions T1, T2 and T3 that access the data from the cache are assigned with time stamps as shown in fig.5.3. The messages are time stamped as given in the fig5.3. The log entries are made by the mobile hosts save these global time stamp for that recovery manager unit. As given in the fig 5.3, T1's read request is time stamped with TS1. Read operations of T2 and T3 are time stamped as TS2 and TS3. When T1 fails and recovers from the failure it sends a recovery status query to the recovery manager with the timestamp for the last operation i.e read with TS1. Now recovery manager checks whether TS1 is earlier than any recent update or whether current value of the data is valid (within PLP). Since the recovery manager finds the valid data it makes the transaction to continue with already read value. Hence the proposed framework ensures dependency preservation.

6. CONCLUSION

The proposed framework ensures concurrency control without locking, with dynamic PLP of data item that increases the transaction throughput. This proposed framework ensures acceptable recovery probability as compared to the previous framework. Also this framework significantly reduces the recovery time with all dependency preservation Thus proposed frame work assures increased throughput, reduced recovery time with dependency preservation.

7. REFERENCES

[1] Victor C.S., Kwok wa Lam and Son, S.H., "Concurrency Control Using Timestamp Ordering in Broadcast Environments", The Computer Journal, Vol.45 No.4 PP.410-422, 2002.
[2] Ho-Jin Choi, Byeong-Soo Jeong, "A Timestamp- Based Optimistic Concurrency Control for Handling Mobile Transactions", ICCSA 2006, LNCS 3981, pp. 796-805, 2006.
[3] P. Krishna Reddy, Masaru Kitsuregawa, "Speculative Lock Management to Increase Concurrency in Mobile Environments", MDA'99, LNCS 1748, pp. 82-96, 1999.
[4] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "Single Lock Manager Approach for Achieving Concurrency in Mobile Environments", Proceedings of 14th IEEE International Conference on High Performance Computing, 2007.
[5] Salman Abdul Moiz, Mohammed Khaja Nizamuddin," Concurrency Control without Locking in Mobile Environments", ICETET 2008.
[6] Barbara, D. and Imielinski, T., "Sleepers and Workaholics:Caching Strategies in Mobile Environments", Proceedings of ACM SIGMOD, 1994**.**
[7] Jing, J., Elmargarmid, A., Helal, A. and Alonso, F., "Bit-Sequences: an Adaptive Cache Invalidation Method in Mobile Client Server Environments", Technical Report CSD-TR-94-074.Purdue University, May 1995.
[8] Hu, Q. and Lee, D.L., "Adaptive Cache invalidation Methods in Mobile Environments", Proc. 6th IEEE International Symposium on High Performance Distributed Computing Environments, 1997.
[9] Joe Chun-Hung Yuen, Edward Chan, Kam-Yiu Lam, H. W. Leung, "An Adaptive AVI-based Cache Invalidation Scheme for Mobile Computing Systems", IEEE, 2000.
[10] Hien Nam Le , Mads Nygard, "A transaction frame work for mobile data sharing services", The second International conference on Mobile Ubiquitous Computing, Systems, Services, and Technologies, IEEE, 2008.
[11] Dunham, M. H. et al. ,"A Mobile Transaction Model thatCaptures Both the Data and Movement Behavior", MobileNetworks and Applications,1997.

[12] Miraclin Joyce Pamila, K. Thanush Kodi, "Frame work for Transaction Management in Mobile Computing Environment", ICGCT-CNIR, Volume II , December 2009.
[13] George Colouris, Jean Dollimore, Tim Kindberg, "Distributed Systems Concepts and Design", Pearson Education, 2002.

Miraclin Joyce Pamila J.C. is a senior Lecturer in Department of Computer Science and Engineering, Government College of Technology, Coimbatore, Tamilnadu, India. She received her Master degree in Computer Science and Engineering in the year 2005 from Anna University, Chennai, India. Her fields of interests are Mobile Computing, Database management Systems, Network Security and Recovery system design for mobile networks. She is a life member of ISTE. She also teaches and guides modules at both B.E. and M.E. levels in Computer Science and Information Technology. She has published 15 technical papers in national and international conferences and journals.

K. Thanushkodi is the Director of Akshaya College of Engineering and Technology, Coimbatore, Tamilnadu, India. His research interests are in the area of Computer Modeling and simulation, Computer Networking, Network security and Power Systems and Design. He received the B.E. in Electrical and Electronics Engineering, M.Sc. (Engg) from Madras University, and PhD in Electrical and Electronics Engineering from Bharathiar University, Coimbatore in 1972, 1976 and 1991 respectively. He has published 35 technical papers in National and International Journals