Intelligent Agent Based Approach for Transaction Processing in Mobile Database Systems

Sekar Ganesh, Mathan Vijayalakshmi, and Arputharaj Kannan
Department of Computer Science and Engineering, College of Engineering, Anna University, India

Abstract: *Transaction processing in a Mobile Database System (MDS) is more complex because of unlimited mobility of the Mobile Unit (MU). The handoff and frequent failure of mobile unit makes it tricky to store log records and access it for recovery. In this paper, we present a new log management scheme, which uses a mobile-agent-based framework to facilitate efficient transaction processing during handoff or MU failure. Instead of executing the transaction in mobile unit, we are giving it to base agent which is in the base station. By doing so, we save power in the mobile unit. Moreover, we are using announcement scheme during handoff which minimizes traffic of Home Location Register Database. Also we use Speculative Locking (SL) protocol, which improves the performance of fixed network transaction by trading extra processing power. In SL, a transaction releases the lock on the data object whenever it produces corresponding after-image during its execution. By accessing both before image and after images, the waiting transaction carries out speculative execution retains one execution based on the termination (commit or abort) mode of the preceding transactions. We have also compared waiting time of ordinary locking protocol and speculative locking protocol.*

Keywords: *Mobile databases, mobile agents, locking protocols*.

Received May 24, 2005; accepted April 12, 2006

1. Introduction

Mobile computers using wireless networks are a rapidly emerging trend. This will give user the information accessing capability regardless of the location of the user or the information. Wireless communication through Personal Communication Systems (PCS) or Global System for Mobile communication (GSM) has become a norm of present day society. Information processing system based on PCS or GSM architecture, which we refer to as the Mobile Database System (MDS). It is essentially a distributed client/server system where clients can move around freely while performing their data processing activities in connected, disconnected, or intermittent connected mode. It can process debit/credit transactions, pay utility bills, make airline reservations, and other transactions without being subject to any geographical constraints.

Since there is no MDS type of system available, it is difficult to identify the transaction volume at mobile units, however, the present information processing needs and trends in e-commerce indicate that transaction workload at each mobile unit could be high and MDS would be a useful resource to most of the organizations. Although MDS is a distributed system based on client/server paradigm, it differs from conventional centralized or distributed systems in transaction management schemes (concurrency control, database recovery, query processing, etc.), different logging scheme, different caching schemes, etc. For example, in MDS or in any distributed system, a number of activities related to transactions' execution, such as transaction arrival at client or at a server, transaction fragmentation and their distribution to relevant nodes for execution, dispatch of updates made at clients to the server, migration of a mobile unit to another cell (handoff), etc., have to be logged for recovery.

Application recovery, unlike database recovery, enhances application availability by recovering the execution state of applications. Application recovery is relatively more complex than database recovery because of:

- A large numbers of applications are required to manage database processing.
- Presence of multiple application states.
- The absence of the notion of the "last consistent state".

This gets more complex in MDS because of:

- The existence of random handoffs, and MU failures limited processing power of mobile units, and
- The presence of operations in connected, disconnected, and intermittent connected modes.

Because of handoff, it is not possible to store the entire log reliably at one location and retrieving it efficiently, which makes it very difficult to see the entire log for recovery. Further, if MDS uses conventional approaches for managing log, even with modifications,

would impose an unmanageable burden on the limited channel capacity.

In this paper, we take these challenges and present an efficient logging scheme, which stores, retrieves, and unify fragments of log for recovery within the constraints of MDS. We recognize and exploit the unique processing capability of mobile agents for dealing with geographical mobility and use them to develop our log management scheme, which is scalable, that is, any new application can be added or existing ones can be deleted dynamically. We claim that our contribution helps to develop robust and highly available mobile information management systems, which are the backbone of e-commerce and m-commerce platforms.

Like in conventional distributed systems we have a COordinator (CO), which coordinates the distributed transaction processing. In our CO, we use a new protocol called speculative locking protocol [8, 10] to improve the performance of mobile database. We extend speculation to 2PL to improve the transaction processing performance of Distributed DataBase Systems (DDBSs). In 2PL, we can observe that even though a transaction produces after-images during its execution, the locks on the data objects are released only after the completion of 2-Phase Commit (2PC).

Parallelism could be increased in the fixed network by allowing the waiting transactions to access the after-images, which were produced by the lock holding transaction during the execution. In the proposed Speculative Locking (SL) protocols, the waiting transaction is allowed to access the locked data objects whenever the lock-holding transaction produces corresponding after-images during execution. By accessing both before and after-images, the waiting transaction carries out speculative executions and retains one execution based on the termination decisions of the preceding transactions. This in turn reduces the waiting time of connected mobile unit.

The remainder of this paper is organized as follows. Section 2 discusses recovery problem in MDS. Section 3 discusses related previous work. Section 4 gives the main objective. Section 5 discusses our system architecture and details of each component. Section 6 gives the performance analysis and results. Section 7 presents the conclusion.

2. Recovery Problems

MDS recovery is more complex than conventional system mainly for the following reasons:

- *Random Handoff*: MUs may be subjected to handoff randomly. A handoff may affect recovery mainly because the location of the desired MU may not be immediately available for communication.
- *MU's Stability*: The unlimited portability of MUs makes them vulnerable to all kinds of failure. For example, it may run out of its limited battery power, it may run out of its disk space, it may be affected by airport security, or user may physically drop the MU, and so on. Any of these events affect its functionality and a recovery algorithm must take these into consideration.
- *Limited Wireless Bandwidth*: This severely affects its communication capability. During recovery, MU may require communicating with BS or with other MUs, but it may not get any free channel for communication. This limitation could seriously affect recovery.

3. Previous Work

The time-out scheme reported in [4], wastes power of the mobile unit because of frequent abort of the transaction. Lazy and Pessimistic schemes (asynchronous schemes) are reported in [1, 3, 9]. In lazy scheme, logs are stored in the BS and, if the MU moves to a new BS, a pointer to the old BS is stored in the new BS. The pointers can be used during failure to recover the log distributed over several BS. This scheme has the advantage that it has relatively less network overhead during handoff, as no log information needs to be transferred. But, this scheme has a large recovery time. In the pessimistic scheme, the entire log and checkpoint records, if any, are transferred at each handoff. Hence, the recovery is fast but each handoff requires large volumes of data transfer.

The work reported in [8] presents two schemes based on the MU's movement and uses independent check pointing and pessimistic logging. In these schemes, the list of BSs where the log is distributed is transferred during a handoff. In the distance-based scheme, log unification is done when the distance covered by MU increases above a predefined value. In the frequency-based scheme, log unification is performed when the number of handoffs suffered by the MU increases above a predefined value. After unifying the log, the distance or handoff counter is reset. These schemes are a trade off between the lazy and the pessimistic strategies.

The schemes discussed so far do not consider the case where a MU recovers in a BS different than the one in which it crashed. In such a scenario, the new BS does not have the previous BS information in its VLR and it has to access the HLR to get this information [6], which is necessary to get the recovery log. HLR access may increase the recovery time significantly if it is stored far from the MU.

4. Objectives

The main objective of this paper is to develop agent-based framework, which provides a platform for implementing our scheme, based on the distributed

logging approach, which reduces recovery time while keeping the total network cost manageable. Also, we develop a new locking protocol called Speculative Locking (SL) protocol, which reduces the waiting time of transactions, by using extra processing power.

5. System Architecture

Figure 1 depicts high level architecture of our Mobile Database System (MDS). Mobilaction fragments the user query and sends the required query to home agent, which is in base station. This transaction is then forwarded to coordinator, which takes care of query distribution and commit protocols. In our system, coordinator uses speculative locking protocol. Event agent captures different events (handoff, failure, registration) of mobile unit. While handoff, we use announcement scheme, which is to reduce the HLR traffic. Along with log we add trace, which we explain in following sections. The detail functions of each process are given in the following sections.

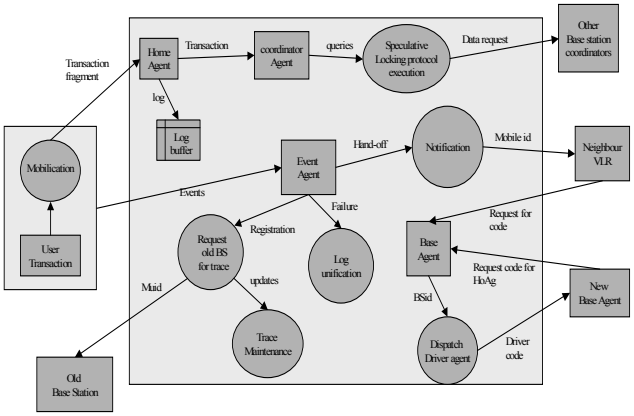

Figure 1. System Architecture.

5.1. Mobilaction

Here we have used transaction model referred to as *"Mobilaction*"[4]. A Mobilaction (Ti) is defined as Ti = {e1, e2, ….., en} where ei is an "execution fragment". Each ei represents subset of the total Ti processing. A transaction Ti is requested at a MU, it is fragmented, and is executed at the MU and at a set of BSs, which is in fixed network. We refer to the MU where a Ti originates or initiates as H-MU (Home MU) and the BS where H-MU initially registered as H-BS (Home BS).

In MDS, like conventional distributed database systems, a COordinator (CO) is required to manage the commit of Ti and its role can be illustrated with the execution of a Ti. A Ti originates at H-MU and the H-BS is identified as the holder of the CO of Ti. H-MU fragments Ti, extracts its ei, sends Ti-ei to the CO and begins processing ei. H-MU may move to other cells during the execution of ei, which must be logged for recovery. At the end of the execution of ei, H-MU updates its cache copy of the database, composes update shipment, and sends it to the CO. CO logs the updates from H-MU. Upon receipt of Ti-ei from H-MU, the CO splits Ti-ej's and sends them to relevant DBSs for execution. The H-MU may suffer a handoff and the CO may change. Handling this situation is explained in forth coming situation.

By forwarding this execution fragment to fixed network, saves power of the mobile unit. Since MUs have some limitation (limited storage, limited power, handoff, frequent failure etc), it is better to forward the execution fragment to fixed network instead of executing in MU itself.

5.2. Speculative Locking Protocol

CO uses a new protocol called speculative locking protocol to improve the performance of mobile database. We extend speculation to 2PL to improve the transaction processing performance of DDBSs. In 2PL, it can be observed that even though a transaction produces after-images during its execution, the locks on the data objects are released only after the completion of 2PC. Parallelism could be increased in the fixed network by allowing the waiting transactions to access the after-images, which were produced by the lock holding transaction during the execution.

In the proposed Speculative Locking (SL) protocols, the waiting transaction is allowed to access the locked data objects whenever the lock-holding transaction produces corresponding after-images during execution. By accessing both before and after-images, the waiting transaction carries out speculative executions and retains one execution based on the termination decisions of the preceding transactions. This in turn reduces the waiting time of connected mobile unit.

The processing of a transaction Ti in DDBS is depicted in Figure 2-a. For a transaction Ti, the notations si, ei, and ci denote the start of execution, completion of execution, and completion of commit processing, respectively. The notation ai denotes the abort of Ti. (Note that an abort can happen at any time during processing). Consider T1 and T2 that access (X, Y) and (X, Z), respectively. Figure 2-b illustrates the processing with conventional 2PL. In this figure, an arc a to b, indicates that b happens after a. Also, ri [X] and wi [X] indicate read and write operations on X by Ti, respectively.

It can be observed that even though T1 produces the after-image of X during execution, T2 accesses X only after the completion of T1's commit processing. Figure 2-c illustrates the processing with SL. Whenever T1 produces X' (X' denotes an after-image of X), T2 accesses both X and X' and starts the speculative executions: T21 and T22. However, T2 commits only after the termination of T1. If T1 commits, T22 is retained. Otherwise, if T1 aborts, T21 is retained. By doing so we reduce the waiting time of transaction which tries to access the locked data is reduced. In

turn, it reduces the bandwidth used by mobile unit. Moreover, the speculative execution is taken place in fixed network, where the processing power is abundant.

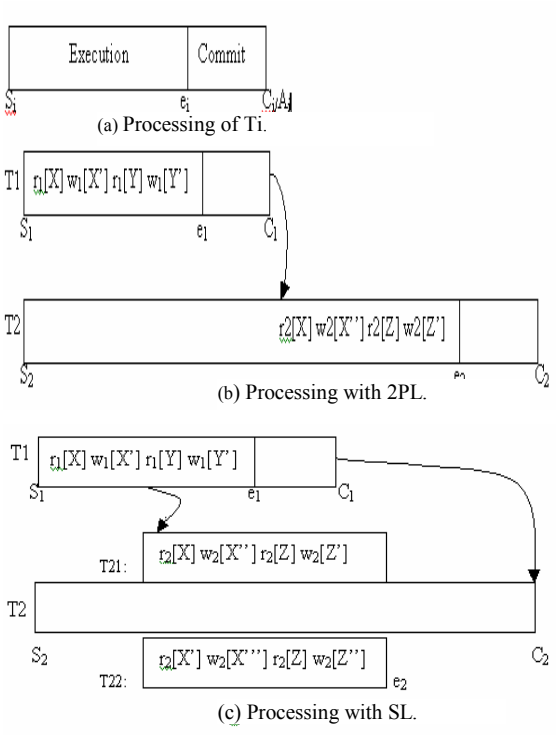

Figure 2. Speculative locking protocol.

5.3. MobileAgents While Handoff

First, let us see various agent used in our framework:

- *Base Agent (BaAg)*: This agent decides which logging scheme to use in current environment. For every MU, the BA creates an instance of agent that handles Mobilaction based on the relevant logging scheme.
- *Home Agent (HoAg)*: This agent handles Mobilaction for each H-MU. It is responsible for maintaining application states.
- *Coordinator Agent (CoAg)*: This agent will act as coordinator in each base station.
- *Event Agent (EvAg)*: This agent will records different events such as registration, handoff, failure of MUs.
- *Driver Agent (DrAg)*: This agent is responsible to move the code and associated data of mobile agents.

MU sends Mobilaction to its HoAg, Which is forwarded to its corresponding CoAg. All interaction of CoAg and MU is sent via HoAg. The HoAg moves along with the MU to the new BS in handoff. When MU gets handoff and registered in new BS, an instance of DrAg is created and it is sent to old BS. DrAg function is to dispatch the HoAg code from old BS to new BS without changing its execution state. Along with HoAg code, log data required to continue its execution is also brought by DrAg. Log data referred above is loaded into HoAg in old BS and its downloaded in new BS. HoAg will execute from its current state without rebooting. HoAg will activate BaAg, which will decide which logging scheme to use. We use trace based logging scheme, discussed in next section.

5.4. Trace Management

During mobile unit handoff, logs should be transferred. If there is frequent handoff, then there will be lot of traffic in the network for log transfer. So, instead of transferring the entire log we use trace based scheme. Trace is a table with two fields namely base Id and log size. By seeing the trace we can take decision of unification of distributed log in different sites. Unification is discussed in next section.

When a mobile unit registers to a base station, base station will send subscriber Id of mobile unit to old base station as a request for trace. Then the old base station HoAg will send the appropriate trace to new base station. Trace is stored in local trace buffer and it adds a new entry with current base station Id in first field and second field is zero. On further execution of transaction, log and trace are updated.

5.5. Log Unification

When the mobile gets failed in any site, the logs are unified on getting the trace. Unification is decided on two values names Expected Recovery Time (ERT) and Unification Time (UT). Expected Recovery Time is found using the previous statistic. Unification Time is calculated by:

Unification Time (UT) = function (Trace (log size), network speed, propagation delay)

ERT is found from previous statistical values. After computing these two values, compare it. If ERT is greater than UT then start unification. If ERT < UT it has to wait until MU recovery. This is because; MU may recover in different base station. If the recovery happens in the same BS, log unification starts, but if the MU reboots in a different BS, then HoAg transfers the trace information and the log stored at this BS when requested.

5.6. Announcement Scheme

When the mobile unit gets handoff, the old base station Id is required for getting the trace and logs. This old BS is found by querying the HLR. For each handoff HLR is queried. If there is more number of MUs and if handoff rate is high then traffic (queries) at HLR will be high. To minimize this, we use announcement scheme.

When any mobile unit is handoff from current, VLR announcement starts. First, it selects adjacent VLR_list, in which mobile unit may register. This adjacent VLR_list selection is based on association

rule (data mining algorithm) got from previous visitor database. Then ANN messages are sent to all VLR in the VLR_list. All these adjacent VLR in VLR_list will receive these ANN message and waits for mobile unit registration or any explicit DEL message. When any mobile unit gets registered, the corresponding VLR will send the message to old VLR about the registration. Then old VLR will send explicit DEL message to all other adjacent VLR_list, to tell that MU had registered in some other VLR. If MU registered any VLR other than adjacent VLR_list then "registered message" will passed via Home Location Register (HLR).

6. Performance Analysis

We have compared the performance of our agent based framework with normal MDS. Also, we have compared the performance of our SL protocol with ordinary locking protocol. In the following section, we have presented the details about our simulation model and then the performance results.

6.1. Simulation Model

As there is no simulator available for MDS, we have built a simulation model MDBSim using Jsim [2]. Initially, we have assumed 36 base stations, in 6X6 grid fashion. To this 36 base station, we have distributed 108 mobile units. Here, all the base stations have an equal area and each BS has at most eight neighbors. These BS's are combined into groups of nine (3X3 grid), where each group represents a MSC and a VLR attached to it. Each mobile unit suffers handoff and failure, with rates *h* and *f*, respectively.

Agents discussed in the above sections are designed by IBM Aglets. In aglets, the agent framework is first tested with single mobile unit. After testing, we have integrated Aglet work and MDBSim by inter-process communication. The database servers are connected through JDBC connectivity. Initially, we gave transaction to 20 mobile units, and the handoff rates (*h*), failure rates *(f)* are varied. Also we have analyzed performance of SL protocol, by varying transmission time.

6.2. Simulation Results

6.2.1. Handoff Rate (*h*) Vs. Transaction Time

Figure 3 shows the relationship between handoff rate and the average transaction time of 20 mobile units. As the handoff rate increases above 6 then transaction time is less to our scheme. So if the handoff rate is high our scheme is fair.

6.2.2. Failure Rate (*h*) Vs. Transaction Time

Figure 4 shows the relation between failure rate and the average transaction time. Agent based logging scheme is always better than the normal schemes.

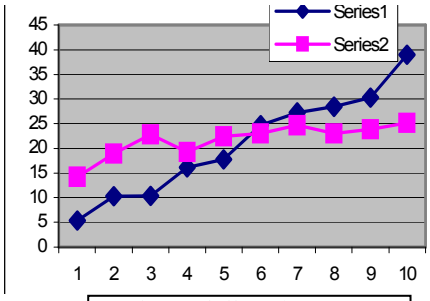

Figure 3. Handoff rate with transaction time.

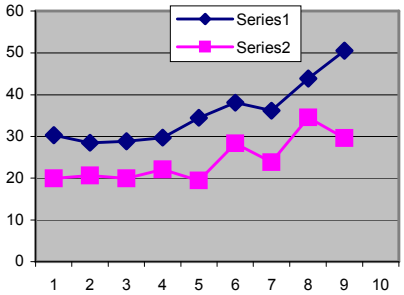

Figure 4. Failure rate with transaction time.

6.2.3. Speculation Vs. 2PL

Figure 5 shows the relation between transmission time and throughput of transaction. As the transmission time increases throughput gets decreases but speculative locking protocol has better throughput than 2PL.

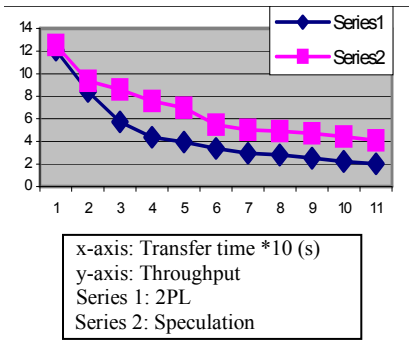

Figure 5. Transfer time with throughput.

7. Conclusion

In this paper, we have presented a mobile agent based framework with speculative locking protocol. The speculative locking protocol proposed with agents

reduces the waiting time of connected mobile unit and facilitates efficient transaction processing during handoff and mobile unit failure. Also, the announcement scheme proposed reduces the traffic at Home Location Register database during handoff by the mobile unit. The simulation result shows that transaction time and throughput are better during handoff and mobile unit failure than the other system.

References

[1] Gadiraju S. and kumar V., "Recovery in the Mobile Wireless Environment Using Mobile Agents," *IEEE Transactions on Mobile Computing*, vol. 3, no. 2, pp. 180-191, April 2004.

[2] JSim, available at: http://chief.cs.uga.edu/~jam /jsim/, 2001.

[3] Krishna P., Vaidya N. H., and Pradhan D. K., "Recovery in Distributed Mobile Environment," *in Proceedings of IEEE Workshop Advances in Parallel and Distributed Systems*, pp. 83-88, October 1993.

[4] Kumar V., Dunham M. H., Prabu N., and Seydim A. Y., "TCOT-A Time Out Based Mobile Transaction Commitment Protocol," *IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1212-1218, October 2002.

[5] Liao G., Liu Y., Wang L., and Peng C., "Concurrency Control of Real-Time Transaction with Disconnection in Mobile Computing Environment," *in Proceedings of the International Conference on Computer Networks and Mobile Computing*, pp. 205-212, 2003.

[6] Mao Z. and Douligers C., "A Distributed Database Architecture for Global Roaming in Next-Generation Mobile Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 146-160, February 2004.

[7] Ozsu M. T. and Valduriez P., *Principles of Distributed Database Systems*, Pearson Education, 2002.

[8] Park T., Woo N., and Yeom H. Y., "An Efficient Recovery Scheme for Mobile Computing Environments," *in Proceedings of the 8th International Conference Parallel and Distributed Systems*, pp. 26-29, 2001.

[9] Pradhan D. K., Krishna P., and Vaidya N. H., "Recovery in Mobile Environments: Design and Trade-Off Analysis," *in Proceedings of the 26th International Symposium on Fault-Tolerant Computing (FTCS'26)*, pp. 1-21, June 1996.

[10] Reddy P. K. and Kitsuregawa M., "Speculative Locking Protocol to Improve Performance for Distributed Database Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 2, pp. 154-169, February 2004.

[11] Schiller, *Mobile Computing*, Addison-Wesley Publications, New Delhi, 2003.

[12] Silberschatz, Korth, and Sunarshan, *Database System Concepts*, Addison-Wesley, 2002.

Sekar Ganesh received his Master in computer science and engineering from Anna University, India, in 2005. Currently, he is working as an assistant system engineer trainee in TCS, Chennai. His areas of interest include database, mobile communication, and algorithms.

Mathan Vijayalakshmi is a lecturer in the Department of CSE, College of Engineering, Guindy, Anna University, India. She has 5 years of teaching experience. Currently, she is working toward her PhD in computer science and engineering at Anna University, India. Her research areas include mobile computing, database, and artificial intelligence.

Arputharaj Kannan is an assistant professor in the Department of CSE, College of Engineering, Guindy, Anna University, India. He has 15 years of teaching experience. He received his PhD in computer science and engineering from Anna University, India in 2001. His research areas include software engineering, database management systems, and artificial intelligence.