A Review on Concurrency Control In Mobile Computing Framework

Arti Nag, Sambhav Agrawal

Computer Science and Engineering,

Bansal Institute of Engineering and Technology, Lucknow, India

Abstract: In modern time the mobile computing based applications needs advanced data synchronization. They works on a complete set of data management services that uses very accurate data modeling , mobile and server-side support systems that can handle deployment and versioning, rules-based data distribution, bi-directional data transfers in a very fast and secure environment. The mobile device-based database services and tight transaction-level integration with multiple enterprise information sources are in great demand. Due to these reasons the mobile computing environment is developed as a distributed computing.

Keywords: Mobile Computing, MDRTDBS, Optimistic Concurrency Control , WSN.

1. Introduction:

Recent advances in wireless communication networks and portable computers have led to the emergence of a new research area called mobile computing systems. An important part of the research conducted in mobile computing systems has been done on mobile data management. What make the mobile data management different from the conventional data management are the mobility of the users or the computers connected to the system, and the resource constraints such as wireless bandwidth and battery life. As a result of such distinctive features of mobile systems, the data management techniques developed for conventional distributed database systems may not work well in a mobile environment. Research contributions are required in a variety of areas, such as distribution of data on mobile and/or non-mobile computers, processing of queries and transactions submitted by mobile users, maintaining the consistency of data cached on mobile computers, and so on. Another important issue that needs to be considered in mobile data management is the requirement of processing queries and transactions within certain time limits in order to maintain the temporal validity of the data accessed by those queries and transactions. Our basic objective in this project is a thorough investigation of the issues to develop various types of methods for mobile data management in response to the requirements mentioned.

Many current researchers in the mobile computing arena share the same vision: ubiquitous access to information, data, and applications. Ubiquitous access refers to the ability of users to access these computing resources from almost any terminal. The idea behind the research is to provide dissemination of larges amount of useful and needful information to different mobile user by designing the efficient data management policies. Recent developments relating to the Internet are establishing solid foundations for wide-area ubiquitous computing systems. [1, 2] Universal access and management of information has been one of the driving forces in the evolution of computer technology. Central computing gave the ability to perform large and complex computations and advanced information manipulation. Advances in networking connected computers together and led to distributed computing. Web technology and the Internet went even further to provide hyper-linked information access and global computing. However, restricting access stations to physical locations limits the boundary of the vision. The real global network can be achieved only via the ability to compute and access information from anywhere and anytime. This is the fundamental wish that motivates mobile computing. This evolution is the cumulative result of both hardware and software advances at various levels motivated by tangible application needs.[3]

2. Related Work:

Kam-Yiu Lam et. al. (2000) [1] they proposed a distributed real-time locking protocol, called Distributed High Priority Two Phase Locking (DHP-2PL), for MDRTDBS. With the rapid advances in mobile computing technology, there is an increasing demand for processing real-time transactions in a mobile environment. Based on the High Priority Two Phase Locking (HP-2PL) scheme. In the protocol, the characteristics of a mobile computing system are considered in resolving lock conflicts. Two strategies are proposed to further improve the system performance and to reduce the impact of mobile network on the performance of the DHP-2PL: (1) A transaction shipping approach is proposed to process transactions in a mobile environment by exploring the well-defined behavior of real-time transactions. (2) We explore the application semantics of real-time database applications by adopting the notion of similarity in concurrency control to further reduce the number of transaction restarts due to priority inversion, which could be very costly in a mobile network. A detailed simulation model of a MDRTDBS has been developed, and a series of simulation experiments have been conducted to evaluate the performance of the proposed approaches and the effectiveness of using similarity for concurrency control in MDRTDBS.

The design of mobile distributed real-time database systems (MDRTDBS) is receiving growing interests in recent years. Due to the poor quality of services provided by a mobile network, it is not easy to meet the deadlines of the transactions in a MDRTDBS. In this paper, we define a detailed model for MDRTDBS, in which the mobility of the mobile clients and characteristics of the mobile network, e.g., disconnection and low bandwidth, are modeled explicitly. We have designed a distributed real-time locking protocol, called Distributed High Priority Two Phase Locking (DHP-2PL), where the characteristics of the mobile network are considered in resolving the conflicts in data accesses. Then, we propose two strategies to improve the system performance and to reduce the impact of mobile network on the performance of the adopted concurrency control protocol. We first propose the concept of transaction shipping to reduce the dependency of a concurrency control protocol on the performance of the

underlying network. With the transaction shipping approach, the communication overheads for processing a transaction can be much reduced. A data pre-fetching mechanism is included in the transaction shipping approach to deal with the dynamic properties of transactions and inaccuracy of prediction in the pre analysis. We then adopt the notion of similarity to resolve conflicts among data access that can be very costly over a mobile network. Different issues in the design of similarity-based real-time locking protocol are discussed. In the design of similarity-based locking protocol, special attention should be paid in resolving a lock conflict in which some of the lock holders are similar to the lock requester while some of them are not. Two methods, the aggressive and conservative approaches, are suggested to resolve the conflicts. Simulation experiments have been conducted to investigate the performance of the DHP-2PL protocol, the effectiveness of the transaction shipping approach and the similarity-based protocols. With the transaction shipping approach, the number of deadline violations is greatly reduced as the contention for channels, the time spent on communication, the probability of lock conflict, and the amount of resources wasted on restarted transactions are much reduced. The transaction shipping approach can also help balance the workload in the system (between the channels and the base stations). The use of similarity-based algorithm further improves the system performance by reducing the number of lock conflicts. However, the experimental results show that the effectiveness of similarity depends very much on the values of the similarity bounds.

The distributed transaction commit problem requires reaching agreement on whether a transaction is committed or aborted. The classic Two-Phase Commit protocol blocks if the coordinator fails. Fault-tolerant consensus algorithms also reach agreement, but do not block whenever any majority of the processes are working. The Paxos Commit algorithm runs a Paxos consensus algorithm on the commit/abort decision of each participant to obtain a transaction commit protocol that uses $2F + 1$ coordinators and makes progress if at least $F+1$ of them are working properly. Paxos Commit has the same stable-storage write delay, and can be implemented to have the same message delay in the fault-free case, as Two-Phase Commit, but it uses more messages. The classic Two-Phase Commit algorithm is obtained as the special $F = 0$ case of the Paxos Commit algorithm proposed by **Jim Gray et al. (2004) [2]**.

Two-Phase Commit is the classical transaction commit protocol. Indeed, it is sometimes thought to be synonymous with transaction commit [2]. Two-Phase Commit is not fault tolerant because it uses a single coordinator whose failure can cause the protocol to block. We have introduced Paxos Commit, a new transaction commit protocol that uses multiple coordinators and makes progress if a majority of them are working. Hence, $2F + 1$ coordinators can make progress even if F of them are faulty. Two-Phase Commit is isomorphic to Paxos Commit with a single coordinator. In the normal, failure-free case, Paxos Commit requires one more message delay than Two-Phase Commit. This extra message delay is eliminated by Faster Paxos Commit, which has the theoretically minimal message delay for a non-blocking protocol. Non-blocking transaction commit protocols were first proposed in the early 1980s [3, 4]. The initial algorithms had two message delays more than Two-Phase Commit in the failure-free case; later algorithms reduced this to one extra

message delay [3]. All of these algorithms used a coordinator process and assumed that two different processes could never both believe they were the coordinator an assumption that cannot be implemented in a purely asynchronous system. Transient network failures could cause them to violate the consistency requirement of transaction commit. It is easy to implement non-blocking commit using a consensus algorithm an observation also made in the 1980s. However, the obvious way of doing this leads to one message delay more than that of Paxos Commit. The only algorithm that achieved the low message delay of Faster Paxos Commit is that of Guerraoui, Larrea, and Schiper [11]. It is essentially the same as Faster Paxos Commit in the absence of failures. (It can be modified with an optimization analogous to the sending of phase 2a messages only to a majority of acceptors to give it the same message complexity as Faster Paxos Commit.) This similarity to Paxos Commit is not surprising, since most asynchronous consensus algorithms (and most incomplete attempts at algorithms) are the same as Paxos in the failure-free case. However, their algorithm is more complicated than Paxos Commit. It uses a special procedure for the failure-free case and calls upon a modified version of an ordinary consensus algorithm, which adds an extra message delay in the event of failure. With $2F + 1$ coordinators and N resource managers, Paxos Commit requires about 2FN more messages than Two-Phase Commit in the normal case. Both algorithms incur the same delay for writing to stable storage. In modern local area networks, messages are cheap, and the cost of writing to stable storage can be much larger than the cost of sending messages. So in many systems, the benefit of a non-blocking protocol should outweigh the additional cost of Paxos Commit. Paxos Commit implements transaction commit with the Paxos consensus algorithm. Some readers may find this paradoxical, since there are results in the distributed systems theory literature showing that transaction commit is a strictly harder problem than consensus [10]. However, those results are based on a stronger definition of transaction commit in which the transaction is required to commit if all RMs are nonfaulty and choose to prepare even in the face of unpredictable communication delays. In contrast, our Non-Triviality condition requires the transaction to commit only under the additional assumption that the entire network is non faulty meaning that all messages sent between the nodes are delivered within some known time limit. (Guerraoui, Larrea, and Schiper stated this condition more abstractly in terms of failure detectors.) The stronger definition of transaction commit is not implementable in typical transaction systems, where occasional long communication delays must be tolerated.

Salman Abdul Moiz et al. (2010), [3] they worked for any database environment either wired or wireless, if multiple host access similar data items it may lead to concurrent access anomalies. As disconnections and mobility are the common characteristics in mobile environment, preserving consistency in presence of concurrent access is a challenging issue. Most of the approaches use locking mechanisms to achieve concurrency control. This leads to increase in blocking and abort rate in mobile environments. However the dynamic timer adjustment strategies may use locking mechanism to efficiently implement concurrency control. To reduce deadlocks and blocking of resources an enhanced optimistic approach for concurrency control is proposed by **Salman Abdul Moiz et al. (2010) [3]**. To show the effectiveness of the

commit protocols in mobile environments, a simulator is designed and implemented to demonstrate how the transactions are committed and how the data consistency is maintained when the transactions are executed concurrently. The simulator was tested for both pessimistic and optimistic approaches.

As long as the mobile clients are not involved in the concurrent access of data items, the database consistency can be preserved. When multiple mobile hosts initiate the transactions requesting for the same data item, it can be locked by only one of the transaction. After the execution of the transaction, the data items are unlocked and the same are acquired by the waiting transaction. When one transaction is being executed, the other transaction that needs the same data items, locked by the former has to wait for invariant time. The delay in acquiring the data items may further increase due to disconnections of mobile hosts for longer time. To solve these problems, following concurrency control techniques are proposed. The basic idea is that a transaction has to be executed within certain time period (Execution time). This information is maintained by fixed host. To achieve concurrency control, two phase locking protocol was used in the traditional environment. However this protocol requires clients to communicate continuously with the server to obtain locks and detect the conflicts. Hence it is not suitable for mobile environments. In [3], A Timeout based Mobile Transaction Commitment Protocol uses timeouts to provide non-blocking protocol with restrained communication. It faces the problem of the time lag between local and global commit. In [4] the proposed Mobile 2PC protocol preserves the 2PC principle and minimizes the impact of unreliable wireless communication. This protocol assumes that all communicating partners are stationary hosts, equipped with sufficient computing resources and power supply with permanently available bandwidth.

In the pessimistic approaches, the items may be blocked for certain period of time. To avoid the blocking of data items and allowing multiple users to access the shared data items requires strong conflict resolution strategies. For this reason, an optimistic concurrency control technique is frequently used in wireless environments [13, 14, 15].

An optimistic concurrency control technique detects and resolves data conflicts in the phase of transaction validation. In a mobile environment the transaction validation is done on the server, it may lead to delayed response causing overhead at the server. An Optimistic Concurrency Control with Dynamic Time stamp Adjustment Protocol requires client side write operations. However because of the delay in execution of a transaction, it may never be executed [2]. In [2], the conventional optimistic concurrency control algorithm is enhanced with an early termination mechanism on conflicting transactions. However because of early termination a transaction need to be initiated again and again.

Optimistic concurrency control protocols (OCC) [4, 5, 7] are non-blocking and deadlock free, which make them efficient to use in mobile computing and have been adopted in the Disconnected Operation [6] and Kangaroo Transaction model. However, without locks to data items, transactions might access conflicting data items under an optimistic concurrency control protocol (OCC). Two concurrent transactions conflict if one of them performs a write on similar data items. Therefore, approaches to terminate conflicting transactions are proposed [2,3]. In these approaches if the conflict rate increases, more and more transactions get aborted. Pessimistic

commit protocols suitable for mobile environments are presented from [5] to [8]. Further a Real Time optimistic Commit protocols with a conflict resolution strategy is presented in [9]. The Design & Implementation of Pessimistic Commit protocols is described in [10]. The design and implementation of the optimistic commit protocol is presented in this chapter. The Optimistic Concurrency Control Strategy doesn't use any locking which doesn't block the shared resources. Further concurrency can be guaranteed by first executing transactions locally and later on propagating the results. In this scheme whenever a fixed host detects a concurrency violation, it propagates the updated shared data item to the mobile host using the same data item without aborting it. The mobile host which successfully completes the transaction locally will be committed irrespective of its arrival time. In this scheme there could be a possibility that the transaction which arrived quiet early might not get executed because the other mobile hosts are executing faster. The future work may introduce a priority field to give chance to the transaction which requested first or a hybrid approach for concurrency control that enters into pessimistic approach by partially locking data items to complete its execution is needed.

Managing the transactions in real time distributed computing system is not easy, as it has heterogeneously networked computers to solve a single problem. If a transaction runs across some different sites, it may commit at some sites and may failure at another site, leading to an inconsistent transaction. The complexity is increase in real time applications by placing deadlines on the response time of the database system and transactions processing. Such a system needs to process Transactions before these deadlines expired. A series of simulation study have been performed to analyze the performance under different transaction management under conditions such as different workloads, distribution methods, execution mode-distribution and parallel etc. The scheduling of data accesses are done in order to meet their deadlines and to minimize the number of transactions that missed deadlines. A new concept is introduced to manage the transactions in dynamic ways rather than setting computing parameters in static ways. With this approach, the system gives a significant improvement in performance this approach is proposed by **Y. Jayanta Singh et.al. (2010) [8]**.

A series of simulation study have been performed to analyze the performance under different transaction management situation such as different workloads, distribution methods, execution mode-Distribution and Parallel, impact of dynamic slack factors to throughput. The scheduling of data accesses are done in order to meet their deadlines and to minimize the number of transactions that missed deadlines. Parallel execution of the cohorts reduces the transaction response time than that of serial or distributed execution. The time required for the commit processing is partially reduced, because the queuing time is shorted in parallel and so there are much fewer chances of a cohort aborting during waiting phase. The throughput initially increases with increase in slack factor. But it drops rapidly at very high workloads. The slack factors can be providing by static or dynamics ways. The new approach dynamic management either dynamic intelligent agent or dynamic slack management gives a significant improvement to the performance of the system. Dynamic intelligent agent keeps tracks of timing of the transactions to help them from

aborts. This agent gives advance information about the remaining execution time of the transactions. This helps the system to inject extra time to such transactions. In all the conditions the arrival rate of transaction plays a major role in reducing number of miss percentage and improved performance.

Recent advances in wireless communication networks and portable computers have led to the emergence of a new research area called mobile computing systems proposed by **Vishnu Swaroop et. al. (2011) [9]**. An important part of the research conducted in mobile computing systems has been done on mobile data management. What make the mobile data management different from the conventional data management are the mobility of the users or the computers connected to the system, and the resource constraints such as wireless bandwidth and battery life. As a result of such distinctive features of mobile systems, the data management techniques developed for conventional distributed database systems may not work well in a mobile environment. Research contributions are required in a variety of areas, such as distribution of data on mobile and/or non-mobile computers, processing of queries and transactions submitted by mobile users, maintaining the consistency of data cached on mobile computers, and so on. Another important issue that needs to be considered in mobile data management is the requirement of processing queries and transactions within certain time limits in order to maintain the temporal validity of the data accessed by those queries and transactions. Our basic objective in this project is a thorough investigation of the issues to develop various types of methods for mobile data management in response to the requirements mentioned.

To deal with the characteristics of mobile computing, especially with wireless connectivity and small devices, various extensions of the client/server model have been proposed. Such extensions advocate the use of proxies or middleware components. Proxies of the mobile host residing at the fixed network, called server-side proxies, perform various optimizations to alleviate the effects of wireless connectivity such as message compression and re-ordering. Server-side proxies may also perform computations in lieu of their mobile client. Proxies at the mobile client undertake the part of the client protocol that relates to mobile computing thus providing transparent adaptation to mobility. They also support client caching and communication optimizations for the messages sent from the client to the fixed server. Finally, mobile agents have been used with client/server models and their extensions. Such agents are initiated at the mobile host, launched at the fixed network to perform a specified task, and return to the mobile host with the results. Another concern in terms of software architectures is adaptability. The mobile environment is a dynamically changing one. Connectivity conditions vary from total disconnections to full connectivity. The resources available to mobile computers are not static either, for instance a "docked" mobile computer may have access to a larger display or memory. Furthermore, the location of mobile elements changes and so does the network configuration and the center of computational activity. Thus, a mobile system is presented with resources of varying number and quality. Consequently, a desired property of software systems for mobile computing is their ability to adapt to the constantly changing environmental conditions. Despite the complete challenges and stress that mobile and wireless computing places on organizations are quickly developing strategies for their mobile workforces. Location dependent information services have great promise for mobile and pervasive computing environments. They can provide local and non local news, weather, and traffic reports as well as directory services. Before they can be implemented on a large scale, however, several research issues must be addressed. The scope of this paper is to raise the data management in terms of operation and management of application software and management services within the mobile distributed systems and the impact of advanced computing and networking technologies on management [9].

Ashish Srivastava et. al. (2012), [10] they worked on a homogenous distributed real time replicated database system that is a network of two or more DBMS that reside on one or more machines. A distributed system that connects two or more databases are Homogenous Distributed Database Systems (HDDBS) create different problems when accessing distributed and replicated databases. Particularly, access control and transaction management in HDDBS require different mechanism to monitor data retrieval and update to databases. Current trends in multi-tier client/server networks make DDBS an appropriated solution to provide access to and control over localized databases. This paper highlights the basic concepts underlying distributed database system including transaction management in in HDRTDBS. Distributed database systems (DDBS) create different problems when accessing distributed and replicated databases. Particularly, access control and transaction management in DDBS require different mechanism to monitor data. Retrieval and update to databases and reduces the communication traffic and also achieves a good transaction response time. An example is given to demonstrate the step involved in executing the two-phase commit.

Scheduled the present time Transaction management is an fully grown thought in distributed data base management systems (DDBMS) for research area. Our Homogenous Distributed Database Systems based replication proposal is able to inherit and reduces the communication traffic the best characteristics of the Database Systems. However, Oracle was the first commercial DBMS to implement a method of transaction management: the two-phase commit. Though it was very difficult to obtain in order on homogenous DBMS implementation of this method were able to pull together sufficient in sequence to put in writing homogenous transaction for the database system. Many associations do not implement distributed databases because of its difficulty. They simply resort to centralized databases. However, with global organizations and multi-tier network architectures, distributed implementation becomes a necessity. It is hoped that this paper to will assist organization in the implementation of distributed databases when installing homogenous DBMS, or give confidence organizations to journey from centralized to distributed DBMS. Organisations could also contribute to this process by having graduates with the knowledge of homogeneous DBMS capability. With DBMS making so much effort on incorporating this and other advanced features in its database software, academicians should also play a major role in exposing beneficiary to these superior element transaction management.

3. Conclusion:

In this paper we have worked on the review of complete database distributed among wireless components as in mobile switching stations. We found that the entire database is being distributed in wireless components of the computer systems. Some of the parameters that influence and complicate database management will design of database and replication of database. We will developed a mobile environment protocol that can handle the distributed database of several clients using priority based concurrency control mechanism with considerations of Hand Off situation.

References:

[1] kam-yiu lam et. al., "Concurrency Control In Mobile Distributed Real-Time Database Systems", Information Systems Vol. 25 No. 4, pp. 261–286, 2000.

[2] Jim Gray and Leslie Lamport, "Consensus on Transaction Commit" 1 January 2004 revised 19 April 2004, 8 September 2005.

[3] Salman Abdul Moiz et. al., "Commit Protocols in Mobile Environments: Design & Implementation" International Journal of Database Management Systems (IJDMS) Vol.2, No.3, August 2010.

[4] Bernstein, P.A, Hadzilacos, V. and Goodman, N, "Concurrency Control and Recovery in database System", Addison-Wesley 1987

[5] H. T. Kung and J. T. Robinson, "On Optimistic Methods for Concurrency Control," ACM TODS, 6(2), June 1981.

[6] J. Kisler, and M. Satyanarayanan, Disconnected Operation in the Coda File System, ACM Transactions on Computer Systems, 10(1), 1992.

[7] T. Härder. "Observations on optimistic concurrency control schemes". Information Systems, 9(2):111–120, 1984.

[8] Y. Jayanta Singh et. al., "Dynamic management of transactions in distributed real-time processing system" 10.5121/ijdms.2010.2210.

[9] Vishnu Swaroop, and Udai Shanker, "Data management in Mobile Distributed Real Time Database Systems: Reviews and Issues" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1517-1522.

[10] Ashish Srivastava, et. al., "International Journal of Advanced Research in Computer Science and Software Engineering" IJARCSSE, Volume 2, Issue 6, June 2012.