Simulating MDBS Transaction Management Protocols

R. Lawrence K. Barker A. Adil
Department of Computer Science
University of Manitoba
Winnipeg, MB, R3T 2N2

Abstract

This paper investigates the performance of aggressive and pessimistic multidatabase (MDBS) transaction management protocols with different transaction submission characteristics. We have used a detailed simulation study in a realistic MDBS environment to study their performance. The aggressive algorithm displayed very poor performance and did not scale well when presented with higher database loads. The pessimistic algorithm had much better performance and scalability under all database loads. Thus, this work demonstrates that optimistic protocols suffer surprisingly worse performance than more pessimistic, serial-like protocols, due to the high abort rates and conflicts among global transactions present in optimistic protocols. This contradicts our intuition about such systems and demonstrates the unique challenges faced by researchers in this difficult environment.

Keywords: multidatabase ; transaction management ; integration ; interoperability

1 Introduction

A multidatabase system (MDBS) is a collection of autonomous databases participating in a global federation to exchange data. Multidatabases are important in industry because they allow diverse and distributed data sources to be integrated. Such an integrated system allows complex system-wide business management-level queries to be constructed. In short, they allow interoperatibility between data systems.

Ensuring consistency of the data in each autonomous database is difficult because the global level transaction manager cannot directly access local database objects. Several transaction management protocols have been proposed to allow global transactions access to data across numerous autonomous databases, but there has been no documented work comparing the performance of the proposed protocols. This work builds a MDBS simulator to investigate the performance of two such protocols: the Ticket Method algorithm [5], and the Global Serial Scheduler [2].

This work contributes to existing work by constructing a general, MDBS simulator capable of simulating any global transaction management protocol and MDBS configuration. The simulation system can be used to investigate new protocols under varying MDBS loads and organizations and determine an appropriate protocol for a given MDBS implementation. Finally, the simulation studies on the two algorithms yield many interesting results including the result that aggressive subtransaction submission results in worse performance than more serial submission approaches. This result is very important when designing a MDBS and new transaction management protocols.

2 Multidatabase Architecture

The basic MDBS architecture (see Figure 1) consists of a global transaction manager (GTM) which handles the execution of global transactions (GTs) and is responsible for dividing them into subtransactions (STs) for submission to the local database systems (LDBSs) participating in the MDBS. Each LDBS in the system has an associated global transaction server (GTS) which converts the subtransactions issued by the GTM into a form usable by the LDBS. Certain algorithms also rely on the GTS for concurrency control and simulating features, such as visible 2PC, that the LDBS does not provide. Each LDBS is autonomous so modifications are not allowed. Finally, local transactions not under the control of the GTM are allowed at each LDBS as if the LDBS was not participating in the MDBS.

3 Previous Work

There have been several algorithms proposed to handle the MDBS concurrency control problem. The

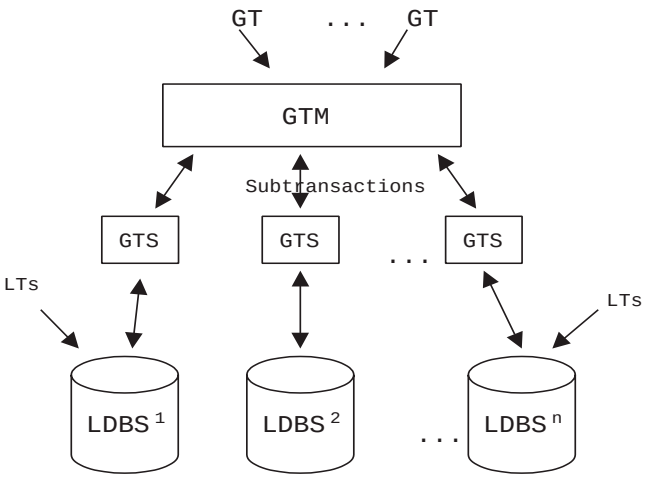

Figure 1: MDBS Architecture

two algorithms examined in this paper are the Ticket Method [5], and the Global Serial Scheduler [2].

The ticket GTM is a method for global concurrency control proposed by Georgakopolous *et al.*[5]. The basic idea is that normally undetectable local conflicts can be detected by forcing every subtransaction of a global transaction to "take-a-ticket" at the LDBS. Global transactions can then be serialized based on this ticket information. A global deadlock detection algorithm (based on timeouts) is necessary to prevent global deadlocks.

The Global Serial Scheduler (GSS) ensures MDB-serializable schedules by submitting global transactions serially[2]. It determines if all active global transactions access no more than one of the databases accessed by the global transaction (GT) being scheduled. The GT with only one subtransaction will always meet this condition and thus can always be submitted.

There has been a lot of work done on simulating database systems. Michael Carey has been involved in extensive simulation studies on concurrency control algorithms in both centralized databases [1] and distributed databases [4]. However, the amount of simulation work performed on multidatabase systems is extremely limited. To our knowledge, no general MDBS simulation system has been constructed to compare different transaction managers. Transaction management in multidatabases is a harder problem than in distributed databases because each database in the MDBS is autonomous.

4 Simulation Architecture

The MDBS simulator was developed using a C++ library designed at Vrije Universiteit [3] which provides a C++ interface for common simulation requirements including event scheduling, statistics gathering, and random number generation. The simulation system consists of approximately 15,000-20,000 lines of C++ code and is structured using object-oriented techniques to limit code duplication.

Simulating a MDBS requires simulation of both local databases (and their local transactions) participating in the MDBS, and the global-level transaction management protocols handling the submission and execution of global transactions. The following is a brief description of the simulation system with a more detailed description available in [6].

4.1 LDBS Simulation

We have developed a generic local database (LDBS) simulator designed to handle different transaction managers with varying database configurations. The database simulation is capable of modeling one database and its transactions. Collections of these (local) databases can then be combined into a MDBS simulation. Although it is typical to use a LDBS simulator as part of a MDBS simulation, it is possible to simulate a LDBS without considering global transactions.

For this study, relational databases using strict two-phase (strict-2PL) locking for transaction management (TM) were simulated. All LDBSs in the MDBS simulations use the relational strict-2PL protocol and provide a visible-2PC for use by GTM algorithms. Locking activities are assumed to take zero time. The simulation gathers statistics on the number of transactions (committed and aborted), throughputs in both transactions/sec and bytes/sec., and the average transaction execution time.

4.2 MDBS Simulation

The multidatabase simulator can handle multiple MDBS configurations and different global transaction managers (GTMs). The MDBS simulator is divided into a set of classes which provide the MDBS functionality. Testing different MDBS transaction managers only requires redefining the one class associated with transaction manager specific functions. This allows for greater code reuse and continuity across simulations.

For this study, the two GTMs simulated are the Ticket Method algorithm and the Global Serial Sched-

uler. Each algorithm is implemented exactly as detailed in their respective papers.

5 Simulation Parameters

A local database simulation was performed to validate the system and determine how increasing the transaction generation frequency (database load) affects database performance. These performance statistics were then used to configure the local databases in the MDBS simulations.

The database consists of 6 relations and a set of representative read and update queries. Based on the size of the relations, the probabilities of query occurence, and the database processing speed, the average time to execute a query should be 1.05 seconds if no data access conflicts are encountered.

The local database simulator was run 10 times for 1000 seconds. Statistics were gathered on transaction residence time and object lock times, wait times, and queue sizes. Interarrival times are generated using an exponential function. The simulation shows that the average transaction residence time increases rapidly as the mean interarrival time decreases below 0.3 transactions/sec. Note that the system does not get overloaded at an arrival rate of 1 transaction/sec as queuing theory would suggest because the increased parallelism provided by executing multiple read transactions simultaneously allows for a higher arrival rate than could normally be achieved.

5.1 MDBS Parameters

The simulated MDBS configuration consisted of 3 nearly identical LDBSs. Local databases 1 and 3 are identical. Local database 2 has the same object sizes as the other local databases, but some objects have different names. Each LDBS has a local transaction arrival rate of 1 transaction/sec. and a average query execution time of 1.05 seconds. Finally, all local database objects are available in the global view.

The set of global queries to the MDBS can access from 1 to 3 LDBSs. If all global subtransactions are executed serially, the average global transaction execution time is 3.1 sec and the maximum time is 5.3 sec. If all GSTs are executed in parallel, the average global transaction execution time is 1.5 sec with the maximum being 2 sec.

The MDBS simulator is run 10 times for 100 global transactions. A global transaction must commit before it is allowed to leave the system, so it may be restarted many times until it commits. After generating 100 global transactions, the global transaction generator (GT_gen) no longer submits global transactions, although the local database transaction generators continually submit local transactions. All LDBS arrival processes are exponential with mean of 1 sec. The GT_gen process is exponential and has the same starting seed for both GTM protocols. Thus, the set of 100 GTs generated will be exactly the same for both GTMs and both GTMs must handle the same load.

The simulation ran for 100 transactions because the Ticket Method algorithm failed when presented with high global transaction arrival rates or long simulation times. A value of 100 transactions was chosen in order to have a meaningful comparison. The GSS algorithm did not suffer these failings and could be run for 10,000 transactions or more with very high global transaction arrival rates. We must stress that the simulation failed because of the characteristics of the Ticket Method algorithm and not due to software or hardware problems in the simulation implementation.

Statistics were gathered on the number of global transaction aborts, and the mean and maximum global transaction residence times. Also, the number of aborts and average transaction execution time were recorded at each LDBS.

5.2 Scalability and Realism Arguments

The type and frequency of both local and global queries is completely defined in input files which can be varied as needed. Currently, both local and global level queries are read-only with about an 80% probability which tracks real DBMS query mixes.

Since the LDBSs are distributed, the time to communicate between the global level (MDBS) and the LDBSs through the GTS is dependent on the MDBS implementation and network-related issues. However, for the purpose of these simulations, this communication time is set to zero. Obviously in a real system, this would be unrealistic, but the value is the same across all simulation experiments so it would have a uniform impact.

The processing speed of each LDBS is considered an "infinite resource" as there are no restrictions on parallelism and dividing this resource. For example, many read transactions can be executing simultaneoulsy and each will receive the maximum processing speed. This infinite resource assumption has been used in other simulation papers.

The zero communication time and infinite resource assumptions are used to determine which GTM algorithm performs best under identical communication

conditions based solely on the restrictions on performance/parallelism inherent in the algorithm and its management procedures, and independent of the environment or MDBS configuration. Modeling non-zero communication times and resource contention can be easily added to the simulation to consider these performance aspects of the MDBS. However, communication and hardware delays actually represent a very small overhead in the system compared to the overhead resulting from the global transaction manager maintaining the consistency of the data.

6 Simulation Results

The simulations yielded some interesting results that were unexpected. The most important result is that the additional concurrency of the Ticket Method algorithm is more of a drawback than an asset.

6.1 Tuning the Ticket Method

The Ticket Method GTM has two parameters critical to its performance. They are the timeout value assigned to a global transaction, and the time for global transaction resubmission after abort. If these parameters are not properly configured, performance will decrease due to high transaction abort rates, global deadlocks, and LDBS overloading.

6.2 Comparing the Two Protocols

Both the Ticket Method algorithm and the GSS algorithm were run on the identical MDBS configuration outlined. Both algorithms were presented with the same local and global transactions. The results indicate the fundamental flaws with the Ticket Method algorithm, and the hidden benefits of the GSS algorithm. Figure 2 shows the residence time of global transactions for both protocols, and Figure 3 shows the residence time of all transactions at local database 1.

The major difference between the two algorithms is how the submission of the global transactions (and their subtransactions) are controlled. The Ticket Method submits all subtransactions when a global transaction is submitted. The ticket in each LDBS, and the GSG check insures global serializability, but there is no global flow control. This often results in lower concurrency as the LDBSs become overloaded which causes even more global transactions to timeout and abort. Restarted aborted transactions reenter the system competing for resources again, especially the

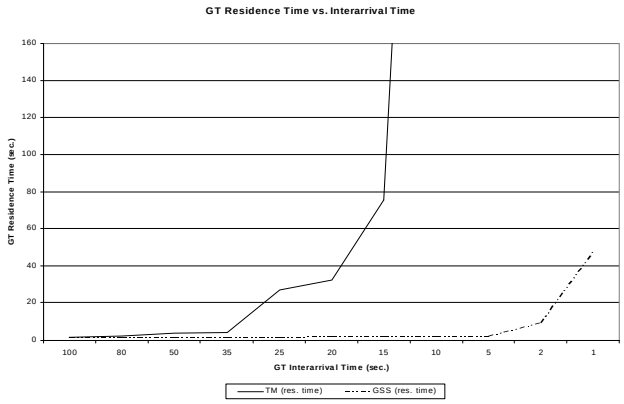

Figure 2: GT residence time vs. GT Arrival Rate

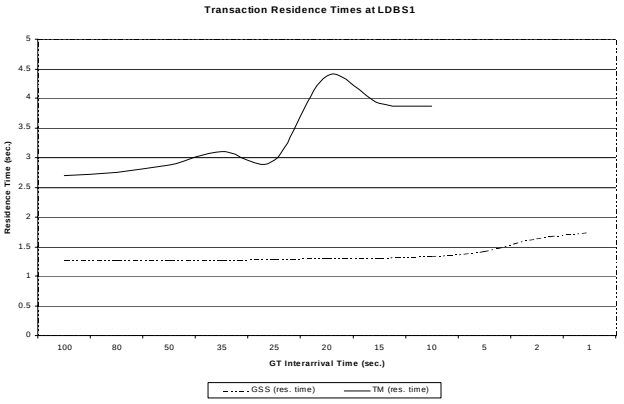

Figure 3: Transaction res. times at LDBS 1 vs. GT arrival Rate

ticket resource. Without global-level flow control, the Ticket Method cannot guarantee that transactions are committed in the order they are received which often results in highly variable global residence times and abort rates. This high variability explains the unexpected jump in residence time at LDBS 1 for a global interarrival time of 20 seconds. The Ticket Method algorithm is highly sensitive to timing issues, and in this case, suffered even worse performance than expected.

The GSS algorithm is deadlock-free and does not cause global transaction aborts. This allows the average and maximum GT residence times to be fairly consistent until the arrival rate passes the threshold of about 5 transactions/sec. The average GT residence time is very low and tends to be around 2 seconds

which is somewhere in between the average parallel execution time (1.5 sec) and the average serial execution time (3.1 sec). Also, transactions are committed in the order they arrive which further reduces the average residence time.

At the local level, the GSS algorithm causes no local aborts, and the global residence time at an LDBS does not depend on the global arrival rate. Thus, the total residence time of all LDBS transactions is not affected by the global arrival rate. The stability of the residence time at the LDBSs arises because the GSS algorithm implements both concurrency control and flow-control at the global level. Global subtransactions are only submitted when no conflicts can arise which limits the number of global subtransactions active at any LDBS, and consequently prevents the MDBS from overloading a LDBS with global subtransactions. Although performance may be decreased slightly by executing some STs serially, this performance is more than made up for by limiting the burden placed on the LDBSs by global transactions. Thus, global subtransactions that are submitted to a LDBS can execute faster than if they were competing for the same resources with other global subtransactions. Global-level flow-control is especially important when the global transaction arrival rate is high and when one or more LDBSs are heavily loaded.

In terms of implementation, the GSS is also much easier to build and configure. It is highly robust and is only slighted effected by increases in LDBS load, query mix changes, or varying global transaction submission rates. The Ticket Method algorithm is highly susceptible to performance concerns if the deadlock detection time and the restart time are not properly configured. Unfortunately, the performance varies greatly even within the same configuration. Also, it is next to impossible to properly configure the system to handle changing MDBS conditions. For example, the GSS can easily handle local transaction arrival rates of less than one second for the given MDBS configuration. The Ticket Method algorithm does not even complete the simulation for such values as it gets stuck in long cycles of global aborts and restarts. The Ticket Method with its high abort/resubmission rate and use of prepare-to-commit is slightly favored over the GSS algorithm which does not abort GTs or use prepare-to-commit in this simulation environment because of the zero communication time.

7 Conclusion and Future Work

This paper presented a general MDBS simulation system which was used to investigate the performance of two different GTM algorithms. The simulations showed that the GSS algorithm is superior because it robustly handles different global transaction and local transaction arrival rates. It has better performance in all situations and offers more consistent global transaction residence times. The Ticket Method algorithm is highly dependent on arrival rates and abort frequencies and suffers from poor performance and highly variable transaction residence times.

Future work involves extending the simulator to handle different local and global transaction managers and database models. It would be interesting future work to determine if other global transaction management algorithms which do not provide global-level flow control have the same performance liabilities as the Ticket Method GTM.

References

[1] Rakesh Agrawal, Michael J. Carey, and Miron Livny. Concurrency control performance modeling: Alternatives and implications. *ACM Transactions on Database systems*, 12(4):609–654, December 1987.

[2] K. Barker. *Transaction Management on Multidatabase Systems*. PhD thesis, University of Alberta, 1990.

[3] D. Bolier and A. Eliens. Simulation modeling support for discrete event simulation in C++. Technical report, Vrije Universiteit, Departement of Mathematics and Computer Science, October 1995.

[4] Michael J. Carey and Miron Livny. Conflict detection tradeoffs for replicated data. *ACM Transactions on Database systems*, 16(4):703–746, December 1991.

[5] D. Georgakopoulos, M. Rusinkiewicz, and A.P. Sheth. Using tickets to enforce the serializability of multidatabase transactions. *IEEE Transactions on Knowledge and Data Engineering*, 6(1), February 1994.

[6] R. Lawrence, K. Barker, and A. Adil. Simulating MDBS transaction management protocols. Technical Report TR-98-05, University of Manitoba, Department of Computer Science, March 1998.