This is the published version (version of record) of:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30005294

Copyright : ©2004 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
The Cost of Recovery Protocols in Web-based Database Systems

Adil M. Hammadi, John Casey, Wanlei Zhou

School of Information Technology, Deakin University
{amham, jacasey, wanlei}@deakin.edu.au

Abstract

The cost of recovery protocols is important with respect to system performance during normal operation and failure in terms of overhead, and time taken to recover failed transactions. The cost of recovery protocols for web database systems has not been addressed much. In this paper, we present a quantitative study of cost of recovery protocols. For this purpose, we use an experiment setup to evaluate the performance of two recovery algorithms, namely the, two-phase commit algorithm and log-based algorithm. Our work is a step towards building reliable protocols for web database systems.

1. Introduction

The purpose of recovery protocols is that in case of failures occur in a system, the state of the system should be restored to a consistent state [1]. Evaluation of the performance of recovery protocols is important to achieve better fault tolerant algorithms. In this paper, we evaluate the performance of recovery protocol of two-phase commit protocol and log-based recovery protocol for web database systems. Our paper presents quantitative approach for the evaluation of recovery algorithms for web database systems.

Recovery protocols for distributed systems in general and for distributed database systems in particular are studied by [4], [5], [6], [7], and [8]. But no such work is done for the web database systems. Our work is an effort towards building reliable protocols for the web database systems.

Two-phase commit (2-PC) protocol is well known atomic commitment protocol for distributed transactions. Log-based algorithm has many flavours. We establish an experiment for transaction-oriented web application like airline reservation system and we use this setup for comparison of 2-PC and revised log-based recovery algorithm.

Our result shows that log-based recovery algorithm has transparent recovery, low overhead cost, fast recovery rate and easy implementation as compared to 2-PC algorithm.

Rest of the paper is arranged as follows: Section 2 is about related work. Implementation of recovery protocols in web database systems is discussed in section 3. In section 4 we discuss our experimental setup and in section 5 we quantitatively evaluate the two protocols. Finally, Section 6 presents the summary and future work.

2. Related Work

Recovery algorithms given here are designed for transaction-oriented web applications like online reservation system. In literature, recovery protocols for distributed systems are given by [7] which are: uncoordinated checkpointing, coordinated checkpointing, communication-induced checkpointing, pessimistic logging, optimistic logging and casual logging. Variation of these protocols can be used for recovery algorithms of web database system. Coordinated checkpointing yields good performance and simplifies recovery than other checkpointing algorithms but its implementation is complicated. On the other hand, in log-based recovery algorithms, pessimistic logging protocol is simpler to implement and yields fast recovery but with performance overhead. Since log-based protocols are simpler to implement therefore, we chose that with an attempt to lower the overhead cost.

The cost of recovery in message logging system has been discussed by [8]. Based on their evaluation they conclude that it is bad idea to rely on other processes to provide the messages that have to be redelivered during recovery. For this reason, we used sort of optimistic protocol with fast recovery rate.
Our approach is very close to approach of Lomet and Weikum [10] where they have transparent recovery for client-server information systems. For their approach each request/reply pair need to be logged and on recovery two operations are possible, rollback request and execute again or redo/continue operation depending upon reply message that whether it has been sent to client or not. In our approach, there are logs for each database server, which starts functioning on the failure of server. For consistency, only those database requests are logged in the log of failed server that has been successfully executed on available database servers. In addition, our major contribution is the evaluation of recovery algorithms that is not given by [10].

3. Implementation

In our experiment, we have a web server/application server and two database servers. Application server and database servers form centralized topology with application server being the coordinator and database servers being the participants. We assume that algorithms presented in paper can handle only database server (site) failures and not the network portioning and application servers failure, which need separate algorithms.

3.1. Recovery Algorithms

We use two recovery algorithms in this paper. One is the two-Phase Commit (2-PC) protocol for the commitment of distributed transaction. The second algorithm is a variation of logging protocols for web-based database systems. Using the message logging protocols, this revised algorithm logs SQL statements during failure. This log is used to recover from failures if the database server fails. We compare recovery cost of both algorithms using an experiment environment, where both algorithms are implemented and their costs of recovery are evaluated.

3.1.1. Two-Phase Commit (2-PC) protocol. Two simple steps in this protocol are to send a message from the coordinator to all participants to commit a distributed transaction and if all transactions are ready to commit then the coordinator sends them message to commit finally or the whole transaction is aborted. This is commonly used protocol and therefore, we have not included its detail in the paper. The detail of 2-PC protocol can be found in [2].

This is a very simple algorithm for the commitment of distributed transaction in which recovery is implicit. Using timeouts, we can do recovery in this algorithm. Coordinator timeouts only if it received no reply from any participant.

The coordinator can timeout in following states: WAIT, where after sending prepare commit message to participants, it is waiting for their reply and COMMIT or ABORT; in the case where after sending global abort or global commit message to participants, it is waiting for their acknowledgement.

For timeout in WAIT state of the coordinator, it simply sends global abort message to all participants. For timeout in ABORT or COMMIT state, it keeps sending the global abort or global commit messages to participants who have not responded yet and waits for their acknowledgement. This is blocking algorithm during failure and recovery.

3.1.2 The Revised Log-based Recovery Algorithm. This algorithm is a variation of recovery protocols for distributed system to suit the recovery of transaction-oriented web-based database systems. SQL statements from client (web browser) are sent to the application server (web server & application server), which passes the request to DB servers during the normal operation of database servers. When a database server fails, the transaction manager starts logging the failed SQL statement and subsequent transactions in a log until the failed server becomes available (Figure 1). The servers that are available continue to process the transactions. Only those statements that have been executed successfully on available servers are logged in the log. Therefore, this algorithm performs no rollback operation. We accomplish recovery using the log by first blocking the available servers to achieve consistency and then by replaying all the SQL statements in the log. To make our recovery algorithm more efficient, we don’t need to replay ‘select’ statements. Only write transactions are replayed.

Transaction processing on a server and its failure and recovery has been explained in Figure 1. In this figure, a transaction consists of three SQL statements: select, update and insert. After the execution of the select statement, the server fails during processing of update. This statement and subsequent SQL statements are sent to log. As soon as the server becomes available, all the statements in the log are replayed till the transaction is completed (finish).
Figure 1. Log-based recovery algorithm

The details of this algorithm are highlighted below:

/* Initialization */
Let T be the transaction consisting of S1, S2,.. Sn SQL statements.
Let SF be the SQL statement of T in the system at the time of failure.
Let ServerAvailable be the available servers during the failure of one database server.
Let SS be the subsequent SQL statements during failure.

/* while there is no failure of database server continue*/
while(true)
{
    process T;
}

/* during failure, log SQL statements*/
while(sever is down)
{
    /* if SQL statement in the system at the time of failure successfully completed on available servers then log it*/
    if(SF==complete on ServerAvailable)
        Log SF;
    if(SS==complete on ServerAvailable)
        Log SS;
}

/* Recovery: Replay all statements in the log when server becomes available */
while(true)
{
    block ServerAvailable;
    /* ignore 'select' statements to speed up recovery */
    replay all write messages in the log on affected server; /* the one that had failed and recovering */
    unblock ServerAvailable;
    continue normal processing;
}

4. Experimental Evaluations

4.1. Experimental Settings

We conducted our experiment on a collection of 2.4 GHz Pentium 4 machines with 512 MB RAM running Windows XP in a LAN environment. For our experiment, we choose a transactional oriented web application like online reservation system, which allows travel agents to make reservation online, update customer details etc. For the database server, we use Oracle 9i Personal Edition on three nodes. For the implementation of recovery algorithms we use the Java programming language. We simulated web clients that send request to web server. We assume that only a database server can fail. Failures of database servers are generated randomly. We use simulated database server failures for database server failures.

4.2. Metrics

We use four important metrics in our experiment to compare 2-PC and revised log-based algorithms. These metrics are transaction arrival rate, number of transaction handlers, and mean failure rate and service outage against response time. Load on application server is directly proportional to transaction arrival rate [9]. With the increase of transaction handlers, we can execute requests concurrently and this helps us reducing response time. Another important metrics is the mean failure rate λ. Mean failure rate is the number of failures per unit time.
After failure, recovery algorithms apply repairing techniques to achieve the availability of the system. Repair rate is denoted by $\mu$. In our experiment, $\lambda$ is increasing and is given as failure per second. We have not given $\mu$ explicitly. Service outage is the time for which server remains out of service after failure. In our case, service outage is the time for which affected database server remains failed.

5. Experimental Results

5.1. Calculation of Recovery Time

Suppose one of the servers fails for time interval $\Delta t$. Then all other servers remain blocked for interval $\Delta t$ and will not be able to process any request in the 2-PC algorithm whereas in the log-based algorithm, all other servers continue processing transactions for interval $\Delta t$. The purpose of a recovery protocol is defined as to restore the system in a consistent state after the failure [3]. It means that using the 2-PC algorithm, the system is in the consistent state by recovering just only that transaction that was in the system at the time of failure whereas in log-based recovery algorithm many transactions might have processed during interval $\Delta t$. To bring system into consistent state, the transaction that was in the system at the time of failure as well as other transactions in the log are to be executed.

5.2. Transaction arrival rate vs. Response time

The graph in Figure 2 shows the performance of algorithms during normal processing of transactions (without failure). When there is no failure in the system, response time of the 2-PC algorithm is higher than that of the log-based algorithm due to the fact that 2-PC algorithm logs every SQL request in the log during normal operation whereas log-based algorithm has no log during normal operation. As soon as we introduce an error in the system, which is 0.05 errors per second, response time of 2-PC algorithm increases sharply because it is a blocking algorithm. This graph shows that the log-based algorithm has low performance overhead as compared to 2-PC algorithms.

5.3. No. of transaction handlers vs. response time

This graph (Figure 3) shows the response time of algorithms against the number of transaction handlers (threads). When there is no error in the system the response time of both algorithms are almost equal with the increase of transaction handlers. As soon as errors are introduced in the system, which is 0.01 errors per second, the response time of 2-PC is very high for 3 transaction handlers as compared to that of the log-based algorithm and it drops with the increase of transaction handler. As 2-PC is a blocking algorithm, therefore all transactions in the system are blocked due to error and have to wait for their turn. With the increase of transaction handlers, more transactions can be handled concurrently and therefore the response time decreases. Log-based algorithm performs well because it is non-blocking during failures.
5.4. Mean failure rates vs. response time

Another important observation in our experiment is the response time of algorithms against the mean failure rate. Graph in Figure 4 shows that with the increase of the failure rate, response time for the 2-PC algorithm remains above 0.3 seconds whereas the response time for the log-based algorithm is almost steady. This can be asserted from the graph that recovery of log-based algorithm is fast as compared to the 2-PC algorithm, which usually rolls back transaction and repeat the whole transaction again during recovery.

5.5. Service outage vs. response time

In this graph (Figure 5), the response time of the 2-PC algorithm increases due to the fact that during service outage all transactions in the system are blocked and have to wait for the system to restart whereas the response time of log-based algorithm is almost steady which is due to the fact that it is a non-blocking algorithm during failure. If server is unavailable for 3 seconds (say), during this time period the system will remain dormant for 2-PC algorithm. Although log-based algorithm is blocking during recovery, but log replay time is much less then service outage time during which the available server continue processing transactions.

6. Summary

There are several recovery algorithms for distributed systems and distributed database systems and there are quantitative studies of recovery cost of these algorithms. We compared these algorithms for a web database system to achieve quantitative study of recovery cost. Our work is an effort towards building reliable protocols for web database systems. Since log-based algorithm is easy to implement, therefore, we used a variation of log-based algorithm that suits for the web database systems. We compared log-based algorithm to commonly used two-phase commit algorithm and concluded that log-based protocol exhibits transparent recovery, low overhead cost, fast recovery and simple implementation as compared to 2-PC protocol. However, we achieved these results by assuming that there are no network or application server failures. Log-based algorithm can handle at most one failure in the system. To calculate recovery time precisely, we should consider network delay and time to replay messages from the log. In future, we need to extend our algorithm by considering above-mentioned limitations. Another important aspect that needs to be covered in the future is the calculation of the cost of recovery. That is, the precise calculation of time used by each algorithm to reach the consistent state.
7. Contributors

Adil Hammadi is a Masters student and undertook the initial literature search leading to this work. He developed the log based protocol algorithm and wrote the paper. John Casey a PhD student implemented the algorithms using Java, and JDBC technologies, generating the performance related statistics. Professor Wanlei Zhou supervised the overall process.

8. References


