International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.3, MayJune. 2013 pp13531356 ISSN: 22496645

Analysis of Rollback Recovery Techniques in Distributed Database Management System

Akanshika

Abstract: *As system continue to grow in size and complexity, they pose increasingly greater safety and risk management challenges. In this paper, we analyzed different methods of rollback recovery techniques and compare their performance. Idea that are used in the design, development, and performance of rollback recovery have been summarized. Independent check pointing, coordinate check pointing, communication induced checkpointing.*

Keywords: *Distributed database management system, rollback recovery, domino effect, checkpointing.*

I. Introduction

The database is a collection related data in an organized manner. This is the best way of storing the data .a distributed system can be visualised as a set of sites, each site consisting of a number of independent transactions. A distributed database is a database in which storage devices are not all attached to a common CPU. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. A database state is represents the values of database objects that represent some real world entity. The database state is changed by the execution of a user transaction. Individual transaction running in isolation are assumed to be correct. When multiple user access multiple database objects residing on multiple site in distributed database system, the problem of recovery and keep the system in consistent state arises. This paper presents rollback recovery techniques to restore the system in most consistent state.

II. Recovery of Data

Recovery from transaction failures usually means that the database is restored to the most recent consistent state just before the time of failure. To do this the system must keep information about the changes that were applied to data items by various transactions.

2.1Issues in Recovery Protocol in ddbms

With distributed databases, guaranteeing atomicity and durability becomes more complicated. Transactions usually span more than one site, so if a transaction commits, then all the sites that are involved in the transaction have to commit. Also, if the transaction aborts, then all subtransactions have to abort. The problem is how to restore the data in most recent consistent state.

Rollback recovery is suitable where system availability requirement can tolerate the outage of computing system during recovery. It offers a resource efficient way of tolerating failures.

A [checkpoint] entry is recorded in the log periodically, when the system writes out to the database on disk all DBMS buffers that have been modified Consequence: all transactions whose [commit T] entry appears in the log before the [checkpoint], do not need to have their WRITE operations redone in case of a system crash (because all these updates have been recorded to disk during check pointing).

2.2consistency issue in distributed checkpoints

Local checkpoints: a local checkpoint is a snapshot of a process. A local state is not necessarily recorded as a local checkpoint, so the set of local checkpoints is only subset of the set of local state global checkpoint: a global checkpoint is a set of local checkpoints, one from each process. A local checkpoint can be part of global checkpoint if it does not contain any orphan message.

Definition for consistency criteria are provided by [3]:

"Given set of local checkpoints, can this set be extended to a global checkpoint that satisfies the consistency criterion P?" (Where P is traditional consistency, transitlessness, or storing consistency).

Traditional consistency : a global checkpoint is consistent if all its pairs of local checkpoints are consistent means does not exhibit any orphan message(a message m sent by a process Pi to a process Pj delivery of m is belong to Cj,y while its sending event not belong to Ci,x).

transitless global checkpoints: a global checkpoint is transitless if all its pairs of local checkpoints are transitless means a message m is intransit with respect to an ordered pair of local checkpoints (Ci,x, Cj,y) if send(m) belong to Ci,x and deliver(m) not belong to Cj,y.

Strong consistent global checkpoints: a global checkpoint is strongly consistent if all its pairs of local checkpoints are consistent and transitless.

Acceptability: let (Ci, Cj) be an ordered pair of checkpoints, Ci belonging to Pi and Cj belonging to Pj with i!= j. The ordered pair (Ci, Cj) is aceeptable if there is no cedge (e1, e2) with e1 issued by Pi, e2 issued by Pj, Cj> e1, and e2>Cj. A cedges(e1, e2) is such that e1 and e2 are two communication events that belong to different processes and concern the same message. We correspond to different message properties (orphan or intransit) and lead to different intentions of this generic graph.

Stable storage: rollback recovery uses stable storage to save checkpoints, event logs, and other recovery related information. Stable storage must ensure that the recovery data persist through the tolerated failures and their corresponding recoveries.

Garbage collection: checkpoints and event log consume storage resources. As the application progresses and more recovery information is collected, a subset of stored information may useless for recovery. Garbage collection is the deletion of such recovery information. A common approach to garbage collection is to identify the most recent consistent set of checkpoints which is called recovery line and discard all information related to events that occurred before line.

III. Checkpoint Based Rollback Recovery

Upon a failure, checkpoint based rollback recovery restores the system state to the most recent consistent state to the most recent consistent set of checkpoints, i.e. the recovery line. It does not rely on the PWD assumption, and so does not need to detect, log, or repaly non deterministic events. Checkpoint based protocols are therefore less restrictive and simpler to implement than log based rollback recovery. Checkpoint based rollback recovery techniques can be classified into three categories: uncoordinated checkpointing, coordinated checkpointing and communication induced check pointing.

3.1Uncoordinated Checkpointing

Uncoordinated checkpointing allows each process the maximum autonomy in deciding when to take checkpoints. The main advantage of this autonomy is that each process may take a checkpoint when it is most convenient. A process may reduce the overhead by taking checkpoints when the amount of state information to be saved is small. In uncoordinated check pointing possibility of the domino effect , which may cause the loss of large amount of useful work, possibly all the way back to the beginning of the computation uncoordinated check pointing forces each process to maintain multiple checkpoints, and to invoke periodically a garbage collection algorithm to reclaim the checkpoints that are no longer useful. It is not suitable for applications with frequent output commits because these require global coordination to compute recovery line, negating much of the advantage of autonomy.

If failure occurs, the recovering process initiates rollback by broadcasting a dependency request message to collect all the dependency in formation maintain by each process. The initiator then calculates the recovery line based on the global dependency information and broadcasts a rollback request message containing the recovery line. Upon receiving this message, a process whose current state belongs to the recovery line simply resumes execution otherwise its rolls back to an earlier checkpoint as indicated by the recovery line.

3.2 Coordinate check pointing

Coordinate checkpointing requires processes to orchestrate their checkpoints in order to form a consistent global state. Coordinate check pointing simplifies recovery and is not susceptible to domino effect, since every process restarts from its most recent checkpoint. Also, coordinated checkpointing requires each process to maintain only one permanent require each process to maintain only one permanent checkpoint on stable storage, reducing storage overhead and eliminating the need for garbage collection. Coordinate checkpointing is the large latency involved in committing output, since a global checkpoint is needed before message can be sent to OWP.

A straight forward approach to coordinated checkpointing is to block communication while the checkpointing executes. A coordinator takes a checkpoint and broadcasts a request message to all processes, asking them to take a checkpoint when process receive this message, it stop its execution flushes all communication channels, take a tentative checkpoint, and send an acknowledgement message back to the coordinator. After the coordinator receive the acknowledgements from all processes, it broadcasts a commit message that completes the two phase checkpointing protocol. After receiving the commit message, each process removes the old permanent checkpoint and atomically makes the tentative checkpoint permanent. The process is then free to resume exchange messages with other processes.

Minimal checkpoint coordination: coordinated check pointing requires all processes to participate in every checkpoint. This requirement generates valid concern about its scalability. It is desirable to reduce the number of processes involved in a coordinated checkpointing session. This can be done since the processes that need to communicated with the checkpoint initiator either or indirectly since the last checkpoint. Two phase protocol achieves minimal checkpoint coordination. During the first phase, the checkpoint initiator identifies all processes with it has communicated since the last checkpoint and send them a request, and so on, until no more processes can be identified. During the second phase, all processes identified in the first phase take a checkpoint. The result is a consistent checkpoint that involves only the participating processes.

3.3 communication induced checkpointing.

Communication induced check pointing (CIC) protocols avoid the domino effect without requiring all checkpoint to be coordinated. In this protocol processes take two kinds of checkpoints, loc al and forced. Local checkpoints can be taken independently, while forced checkpoints must be taken to guarantee the eventual progress of the recovery line. CIC protocols take forced checkpoint to prevent the creation of useless checkpoints, i.e. checkpoints that will never be part of a consistent global state. Useless checkpoints are not desirable because they do not contribute to the recovery of the system from failures, but they consume resources and caused performance overhead.

As opposed to coordinated checkpointing, CIC protocol s do not exchange any special coordination messages to determine when forced checkpoint should be taken: instead, they piggyback protocol specific information on each application message; the receiver than use this information to decide if it should take a forced checkpoint, this decision based on the receiver determining if past communication and checkpoint patterns can lead to the creation of useless checkpoint. CIC proto cols have been classified in one of two types. Model based check pointing and index based checkpointing

Model based protocol: model based check pointing relies on preventing pattern of communication and checkpoints that could result in Zcycle and useless checkpoints. A model is set up to detect the possibility that such patterns could be forming within in the system, according to some heuristic. A checkpoint is usually forced to prevent the undesirable pattern from occurring. The decision to force a checkpoint is done locally using the information piggybacked on the application messages. Therefore, under this style of check pointing it is possible that multiple processes detect the potential for inconsistent checkpoints and independently force local checkpoints to prevent the formation of undesirable patterns that may never actually materialize or that could be prevented by a single forced checkpoint. thus, model based check pointing always errs on the conservative side by taking more forced checkpoints than is probably necessary, because without explicit coordination, no process has complete information about the global state.

Index based protocol: index based CIC protocols guarantee, through forced checkpoints if necessary, that (1) if there are two checkpoints Ci,m and Cj,n such that Ci,m> Cj,n then timestamp of Cj,n >= timestamp of Ci,m, where ts(c) is the timestamp associated with checkpoint c; (2) consecutive local checkpoints of a process have increasing timestamps. The time stamps are piggybacked on application messages to help receivers decide when they should force a checkpoint. Protocol forces a processes to take a checkpoint upon receiving a message with piggy backed index greater than the local index, and guarantees that the checkpoints having same index at different processes from a consistent state.

IV. comparison

Different rollback recovery protocols offer different tradeoffs with respect to performance overhead latency of output commit, storage overhead ease of garbage collection, simplicity of recovery freedom from domino effect, freedom from orphan processes and extent of rollback. Table 1 summarize the different variation of rollback recovery protocols. Since garbage collection and recovery both involve calculating a recovery line, they can be performed by simple procedures under coordinate checkpoints. Coordinate check pointing can have unbounded rollbacks, and a process may need to retain up to N checkpoints if the optimal garbage collection algorithm is used.

	Uncoordinated check pointing	Coordinated check pointing	Communication induced Check pointing
checkpoint	Several	1	Several
Domino effect	Possible	No	No
Orphan process	Possible	No	No
Rollback Extent	unbounded	Last global checkpoint	Possibly several Checkpoint
Recovery Protocols	Distributed	Distributed	Distributed
Output Commit	Not possible	Global coordination required	Global coordination required

Table 1: A comparison between rollback recovery protocols

References

[1] R Guohong Cao and Mukesh Singhl" On Coordinated Checkpointing in Distributed Systems." vol. 9, no. 12, December 1998.
[2] Elmootazbellah N. Elnozahy and James S. Plank "Check pointing for PetaScale Systems: A Look into the Future of Practical Rollback Recovery." IEEE transactions on dependable and secure computing, vol. 1, no. 2, apriljune 2004.
[3] E.N.Elnozahy "A survey of rollback recovery protocols in message passing system" 2000.
[4] Bharat bhargava "Independent checkpoint and concurrent rollback for recovery in distributed system–an optimistic approach" 1997.
[5] Roberlo baldom "A communication induced check pointing protocol that insures rollback dependency traceability" 2004.
[6] David B. Johnson "Recovery in distributed system using optimistic message logging and checkpointing".1990