UDGAM VIGYATI, Volume 2, 2015, (November)
ISSN 2455-2488
Page No. 196- 201

"ANALYSIS OF CHECK POINTING IN RECOVERY OF DATA IN DATA BASE MANAGEMENT SYSTEM"

Ms. Sarita Sharma
Department of Computer Science,
Govt. Holkar Science College,
DAVV, Indore,

Asst. Prof. Rakesh Gaherwal
Idyllic Institute of Management,
Rau, Indore

ABSTRACT
We present a significant and cost-effective technique Check pointing to recover the data lost during system crash or hardware failure that generally occurs in Data Base Management System (DBMS).The objective of the method is the creation of a vigorous, fault-tolerant system. The method executes its objective by enabling the processing of new transactions to begin even before restart recovery is completed in case of transient failures. It sustains fine granularity (eg. record) locking with semantically rich lock modes and operation logging, partial rollbacks, write-ahead logging, and the steal and no force buffer management sagacity. The overhead incurred by this method during normal transaction processing is trifling. We require very few changes to an existing DBMS in order to support our method. With the implementation of this method several recovery feedback and algorithmic diversity, systems that endure software design faults could possibly be built.
Keywords -DBMS, Transaction, Recovery, ACID, Log based Recovery technique, Shadow Paging, Domino effect

---

INTRODUCTION

The database is a collection of related data in an organized manner. This is the best way of storing the data [1].Database management systems are used to enable developers to create a database, fill it with information and create ways to query and change that information without having to worry about the technical aspects of data storage and retrieval [2].When we perform transaction on a database then sometimes we face the problems related to data loss due to system crash or any other hardware or software problems. The transaction is the fundamental activity of a DBMS and an area of concern for the reviewer. Transactions maintain consistency constraints or controls determined for an application. This consistency must be maintained at all times, even during a transaction failure. Concurrent processing must also be protected against adverse effects during a transaction failure [3]. In that situation we need such technique which recover our lost data efficiently. DBMS provide us two basic techniques Log based recovery [4] and shadow paging [5]. Recovery is needed to achieve the basic properties (ACID properties) [6] of a transaction. When Log based and shadow paging recovery techniques are used then sometimes we face the problems in searching the entire log which is time consuming and unnecessarily redo transaction which have already output then updates to the database. Hence we need a streamline recovery procedure by periodically performing check pointing. This method outputs all log records currently residing in main memory onto stable storage, all modified buffer blocks to the disk and writes a log record <checkpoint> onto stable storage[7]. It is a common technique to execute a program or system

UDGAM VIGYATI, Volume 2, 2015, (November)
ISSN 2455-2488
Page No. 196- 201

with fault enduring characteristics and perform the transaction processing successfully. It is used to recover the data lost during transaction or after failed transaction. The important features of checkpoint is saving and restoration of the system state. By saving the current state of the system periodically or before critical code sections, it provides the baseline information needed for the restoration of lost state in the event of a system failure. By the use of check pointing, the state of the entire system taking as a snapshot is saved to non-volatile storage medium. Hence it is clear that the cost of a checkpoint may vary with the amount of state required to be saved and the bandwidth available to the storage mechanism being used to save the state.

If there is the case of system failure, the internal state of the system can be restored, and it surely continues its service from the point at which its state was last saved. To perform the transaction in this condition, checkpoint involves the failed task or system to be restarted, and provide some parameters to indicate the recovering state. So we can say that complexity of the process, amount of state and bandwidth to the storage device will take more time from a second to many seconds. Due to this reason a process may take a useless checkpoint that will never be part of a global consistent state. Furthermore, each process maintains multiple checkpoints and has to periodically invoke a garbage collection algorithm to reclaim the checkpoints that are no longer useful. Besides, this method is not suitable for applications with frequent output commits because these require global coordination to compute the recovery line. Determining a consistent global checkpoint may involve lot of overhead, especially in large systems, and the processes may have to be restarted from the beginning due to the non-existence of a consistent global checkpoint other than the initial state[8].In this paper we have analysed what the recovery technique Check pointing is, when, where and how it can be used to recover the lost data effectively.

The rest of this paper is organized as follows: In section II, We have explained the types of check pointing. In section III, we have discussed where to use check pointing? In section IV we have described the related work. In section V, we have explained why to use check pointing? In section VI, we have described how to calculate the checkpoints. In section VII suggestion and recommendation are given and finally in section VIII we have concluded the recovery techniques through conclusion and future study.

TYPES OF CHECK POINTING

There are following types of check pointing:

1. Disk based check pointing
2. Disk less check pointing
3. Double check pointing

1. **Disk Based Check pointing:** This type of check pointing saves the state of the computation during a transaction periodically to a stable storage, which is not subject to failures. When a failure occurs the computation is restarted from one of these previously saved states. According to the type of coordination between different processes while taking checkpoints, checkpoint-based methods can be broadly classified into three categories:
    a. Uncoordinated check pointing or asynchronous check pointing
    b. Coordinated check pointing or synchronous check pointing
    c. Communication-induced or Quasi-Synchronous or Hybrid Check pointing

UDGAM VIGYATI, Volume 2, 2015, (November)
ISSN 2455-2488
Page No. 196- 201

2. **Diskless Check pointing:** This technique is used for distributed system with memory and processor redundancy. Two extra processors are used for storing parity as well as standby in this check pointing. Process migration feature has ability to save a process image. The process can be recommenced on the new node without having to kill the entire application and start it over again. It has memory or disk space .Whenever there is a failure then to restore the process image, a new processor is used in replace of the crashed processor. It needs a pool of standby processors for multiple unexpected failures [9].

3. **Double Check pointing:** This type of check pointing needs relatively small memory footprint on very large number of processors to handle fault at a time, each checkpoint data would be stored to two different locations to make certain the availability of one checkpoint. If one is lost then other two can be used which are having identical checkpoints. It can be saved either in the memory or local disk of two processors. These are double in-memory checkpointing and double in-disk check pointing schemes. This scheme stores checkpoint in a distributed fashion to avoid the network restricted access to the central server [10].

    i. Double In-memory Check pointing: In this check pointing each process stores its data to memory of two different processors. It has faster memory accessing capability, low checkpoint overhead and faster restart to achieve better performance than disk-based checkpoint.

    ii. Double In-disk Check pointing: It is useful for applications with very big memory footprint where checkpoints are stored on local scratch disk instead of in processor memory [11].

WHERE TO USE CHECK POINTING

When we use log based recovery scheme then every time logs are prepared which are stored on stable storage. If there is any crash or failure in the database then entire logs are searched which is a wearisome job. Check pointing is used with log based recovery technique to reduce the time of searching the entire log. In case of any crash or failure the searching is done after check points. Check pointing is also used whenever we may face the problems of stable storage full.

RELATED WORK

1. C.Mohan has described the recovery technique check pointing as cost effective for the recovery of data during a failure in his paper "A Cost-Effective Method for Providing Improved Data Availability during DBMS Restart Recovery after a Failure". The results of this paper have been extended to the remote backup context in [MoTO93].
2. Akanshika has analysed different methods of rollback recovery techniques and compare their performance in his paper "Analysis of Rollback Recovery Techniques in Distributed Database Management System".

3. Li Xiao-lei has developed a method in his paper "Research On Data Backup And Recovery Technology In SCADA System" which improved the history data processing method in the real-

UDGAM VIGYATI, Volume 2, 2015, (November)
ISSN 2455-2488
Page No. 196- 201

time database of SCADA (Supervisory control and data acquisition systems). It can achieve the goal of preserving recovering and deleting the entities of real-time database 、 graphics and the results of state estimation, which being provided for the EMS (Energy Management System) to do research on the calculation of historical data.

4. Greg Bronevetsky has reported in his paper, "Recent Advances in Checkpoint/Recovery Systems", his work on the family of C3 systems for (almost) fully automatic check pointing for scientific applications.

WHY TO USE CHECK POINTING

Checkpoints reduce the processed log during a full recovery of a database. Uncoordinated check pointing makes recovery easier and simpler from failure Coordinated check pointing is not prone to Domino effect since every process upon failure always restarts from the most recent checkpoint. Unlike asynchronous check pointing, the system does not preserve any useless checkpoints. Each process has to preserve only one permanent checkpoint on stable storage. Because of these advantages we should use the check pointing in recovery of the data from a database during failure.

HOW TO CALCULATE CHECKPOINTS[7]

During recovery we need to consider only the most recent transaction Ti that started before the check point, and transactions that started after Ti.

1. Scan backwards from end of log to find the most recent <checkpoint> record
2. Continue scanning backwards till a record <Ti start> is found.
3. Need only consider the part of log following above start record. Earlier part of log can be ignored during recovery, and can be erased whenever desired.
4. For all transactions (starting from Ti or later) with no <Ti commit>, execute undo (Ti). (Done only in case of immediate modification.)
5. Scanning forward in the log, for all transactions starting from Ti or later with a <Ti commit>, execute redo (Ti).

If recovery is to be performed in case of concurrent transactions then following steps are followed:

- Checkpoints are performed as before, except that the checkpoint log record is now of the form <checkpoint L>where L is the list of transactions active at the time of the checkpoint.
- We assume no updates are in progress while the checkpoint is carried out.
- When the system recovers from a crash, it first does the following:
- Initialize undo-list and redo-list to empty
- Scan the log backwards from the end, stopping when the first<checkpoint L> record is found. For each record found during the backward scan:
- if the record is <Ti commit>, add Ti to redo-list
- if the record is <Ti start>, then if Ti is not in redo-list, add Ti to undo-list
- For every Ti in L, if Ti is not in redo-list, add Ti to undo-list
- At this point undo-list consists of incomplete transactions which must be undone, and redo-list consists of finished transactions that must be redone.
- Recovery now continues as follows:

UDGAM VIGYATI, Volume 2, 2015, (November)
ISSN 2455-2488
Page No. 196- 201

1. Scan log backwards from most recent record, stopping when < Ti start> records have been encountered for every Ti in undo-list.
    - During the scan, perform undo for each log record that belongs to a transaction in undo-list.
2. Locate the most recent <checkpoint L> record.
3. Scan log forwards from the <checkpoint L> record till the end of the log.
    - During the scan, perform redo for each log record that belongs to a transaction on redo-list

Example of Recovery
<T0start>
<T0, A, 0, 10>
<T0commit>
<T1start>/* Scan at step 1 comes up to here */
<T1, B, 0, 10>
<T2start>
<T2, C, 0, 10>
<T2, C, 10, 20>
<check point {T1, T2}>
<T3start>
<T3, A, 10, 20>
<T3, D, 0, 10>
<T3commit>

SUGGESTIONS AND RECOMMENDATIONS

After analyzing the check pointing recovery technique in detail we have found that it is much better to use Check pointing recovery technique for the recovery of data in case of system or program failure. It reduces the overhead of storage and there is no need to be garbage collected. To save the checkpoints individually, it is convenient to use Uncoordinated Check Pointing otherwise use Coordinated Check pointing. To prevent from domino effect we use Communication Induced Check pointing. Diskless Check pointing improves performance in distributed / parallel applications and process migration save process image. Double check pointing uses small memory footprint on large number of processors. But all these check pointing also have some drawbacks for not having suitable in some cases such as Uncoordinated Check Pointing is not suitable due to having domino effect, wastage memory, unbounded & complex garbage collection. Coordinated Check pointing has consistent checkpoint and large latency for saving the checkpoints storage. Communication Induced Check pointing is deteriorated parallel performance & requires standby processors. Diskless Check pointing has communication bottleneck. Double check pointing depends on a central reliable storage and required additional hardware. It is recommended that checkpoint should not use in broad application like mobile networking etc. After the analysis of the all types of check pointing it is suggested that if we want to recover the data in a database after a failure then first make sure that which type of problem and application it is and then use related check pointing technique avoiding future problems.

UDGAM VIGYATI, Volume 2, 2015, (November)
ISSN 2455-2488
Page No. 196- 201

CONCLUSION

Checkpoint-recovery is a simple and effective method to add fault tolerance to a system, especially when the system is designed with it in mind. It may not be ideal for embedded applications, especially those with real-time requirements due to recovery overhead, but work on this is in progress. It may not be effective for tolerating software design faults. It has trouble when state includes more than just values in memory and registers - for instance disk accesses, and producer-consumer problems. It is concluded that Check Pointing recovery technique is much easier and sufficient for the recovery of data in small scale applications.

REFERENCES

1. Akanshika, "Analysis of Rollback Recovery Techniques in Distributed Database Management System", International Journal of Modern Engineering Research (IJMER) Vol. 3, Issue 3, pp. 1353-1356 ISSN: 22496645, May- June ,2013
2. Article, "What Is a Database Management System?" by Morgan Davis , November, 2014
3. Article, "4-06-60 DBMS Recovery Procedures" by Fredrick Gallegos and Daniel Manson
4. "Log-Based Recovery for Nested Transactions", by J. Eliot and B. Moss, Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003
5. Lecture 5 given by Jacob Bower based on lectures of James Jacobson on November,2001
6. Fundamentals of Database Systems by ElmasriNavathe
7. Database System Concepts ©Silberschatz, Korth and Sudarshan
8. "Checkpoint-Recovery" by John Devale, Carnegie Mellon University, 18-849b Dependable Embedded Systems Spring 1999
9. ParthaSarathi Mandel, Krishnendu Mukhopadhaya,"Performance analysis of different check pointing and recovery schemes using stochastic model" Journal of Parallel and Distributed Computing,66(1),pp. 99-107, January 2006
10. Y. Manable, "A Distributed Consistent Global Checkpoint Algorithm with minimum number of Checkpoints" , Technical Report of IEICE , COMP97-6 April,1997
11. S.Monnet, C.Morin,R.Badrinath,"Hybrid check pointing for Parallel Applications in Cluster Federations" , In 4thIEEE/ACM International Symposium on cluster Computing and the Grid , Chicago , IL , USA , pp.773-782,April 2004