Information Sciences 179 (2009) 3659–3672

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Necessary and sufficient conditions for transaction-consistent global checkpoints in a distributed database system

Jiang Wu [a], D. Manivannan [a,*], Bhavani Thuraisingham [b]

[a] Department of Computer Science, University of Kentucky, Lexington, KY 40506, United States
[b] Department of Computer Science, University of Texas at Dallas, United States

A R T I C L E   I N F O

Article history:
Received 14 November 2007
Received in revised form 9 June 2009
Accepted 14 June 2009

Keywords:
Checkpointing
Recovery
Distributed databases

A B S T R A C T

Checkpointing and rollback recovery are well-known techniques for handling failures in distributed systems. The issues related to the design and implementation of efficient checkpointing and recovery techniques for distributed systems have been thoroughly understood. For example, the necessary and sufficient conditions for a set of checkpoints to be part of a consistent global checkpoint has been established for distributed computations. In this paper, we address the analogous question for distributed database systems. In distributed database systems, transaction-consistent global checkpoints are useful not only for recovery from failure but also for audit purposes. If each data item of a distributed database is checkpointed independently by a separate transaction, none of the checkpoints taken may be part of any transaction-consistent global checkpoint. However, allowing individual data items to be checkpointed independently results in non-intrusive checkpointing. In this paper, we establish the necessary and sufficient conditions for the checkpoints of a set of data items to be part of a transaction-consistent global checkpoint of the distributed database. Such conditions can also help in the design and implementation of non-intrusive checkpointing algorithms for distributed database systems.

1. Introduction

It is a common practice to take checkpoint of a database from time to time, and restore the database to the most recent checkpoint when a failure occurs. It is desirable that a global checkpoint of a database records a state of the database which reflects the effect of a set of completed transactions and not the results of any partially executed transactions. Such a checkpoint of the database is called a transaction-consistent global checkpoint [23]. A straightforward way to take a transaction-consistent global checkpoint of a distributed database is to block all newly submitted transactions and wait until all the currently executing transactions finish and then take the checkpoint. Such a checkpoint is guaranteed to be transaction-consistent, but this approach is not practical, since blocking newly-submitted transactions will increase transaction response time which may not be acceptable for the users of the database. Another approach would be to run a read only transaction which would read the entire database and save it to stable storage; the underlying concurrency control algorithm will ensure that the saved state is transaction-consistent. This would be inefficient especially in the presence of long-living transactions. A more efficient way would be to save (checkpoint) the state of each data item independently and periodically without blocking any transaction. However if each data item is checkpointed independently and periodically, some checkpoints of some data items may not be part of any transaction-consistent global checkpoint of the database and hence are useless.

* Corresponding author. Tel.: +1 859 257 9234; fax: +1 859 323 3740.
E-mail addresses: jwu6@cs.uky.edu (J. Wu), mani@cs.uky.edu, manivann@cs.uky.edu (D. Manivannan), bhavani.thuraisingham@utdallas.edu (B. Thuraisingham).

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.06.016

In this paper, we address this issue and establish the necessary and sufficient conditions for a checkpoint of a data item (or the checkpoints of a set of data items) to be part of a transaction-consistent global checkpoint of the database. This result would be useful for constructing a transaction-consistent global checkpoint incrementally from the checkpoints of each individual data item. By applying this condition, we can start from an useful checkpoint of any data item and then incrementally add checkpoints of other data items until we get a transaction-consistent checkpoint of the database.

1.1. Motivation and objectives

In a distributed system, to minimize the lost computation due to failures, the state of the processes involved in a distributed computation are periodically saved (checkpointed). When one or more processes involved in the distributed computation fails, the processes are restarted from a previously saved consistent global checkpoint. When processes are independently checkpointed, the checkpoints taken may not be part of any consistent global checkpoint and hence are useless [21]. Netzer and Xu [21] introduced the notion of zigzag paths between checkpoints of processes involved in a distributed computation and established the necessary and sufficient conditions for a given checkpoint of a process to be part of a consistent global checkpoint (i.e., useful). They proved that a checkpoint of a process is useful if and only if there is no zigzag path from that checkpoint to itself. Several checkpointing algorithms have been proposed for distributed systems [2,18,9,8,19,3,17].

Checkpointing is also an established technique for handling failures in database systems. Many of the checkpointing schemes proposed in the literature for distributed database systems are intrusive to different extent. Some of these are discussed in Section 2 and Section 3. Non-intrusive checkpointing algorithms under which transactions do not have to be blocked when checkpoints are taken are desirable [30]. If each data item in a distributed database is checkpointed by an independent transaction periodically, it is quite possible that none of the checkpoints taken is part of any transaction-consistent global checkpoint of the database. Motivated by the work of Netzer and Xu for distributed computations [21], in this paper, we establish the necessary and sufficient conditions for a given checkpoint of a data item (or checkpoints of a set of data items) to be part of a transaction consistent global checkpoint.

1.2. Organization of the paper

The remainder of this paper is organized as follows. In Section 2 we introduce the background required for understanding the paper. Section 3 discusses related works. In Section 4 we present the necessary and sufficient conditions for a set of checkpoints of a set of data items to be part of a transaction consistent global checkpoint and prove its correctness; we also discuss the applications of our work. Section 5 concludes the paper.

2. Background

2.1. System model

We consider a model of distributed database system similar to the model in [23]. In this model, a *distributed database system* consists of a set of data items residing at various sites. Sites can exchange information via messages transmitted on a communication network, which is assumed to be reliable. The data items of the database are accessed by transactions and the transactions are controlled by transaction managers (TM) that reside at these sites. The TM is responsible for the proper scheduling of transactions using appropriate concurrency control algorithms in such a way that the integrity of the database is maintained. In addition, the data items at each site are controlled by a data manager (DM). Each DM is responsible for controlling access to data items at its site. Each data item is checkpointed by a local transaction periodically. Before a transaction takes a checkpoint of a data item it obtains an exclusive lock on the data item so no other transaction can be accessing that data item while it is checkpointed. The state of a data item changes when a transaction accesses that data item for a write operation. In order to guarantee the integrity and efficiency of transaction processing, the following four properties, referred to as **ACID** [27], must be maintained.

- **A**tomicity: Each transaction is executed in its entirety, or not at all executed.
- **C**onsistency preservation: Execution of a transaction in isolation (that is, with no other transaction execute concurrently) preserves the consistency of the database.
- **I**solation: Even though multiple transactions may execute concurrently, the system guarantees that for every pair of transactions T_i and T_j, it appears to T_i that either T_j finished execution before T_i started, or T_j started execution after T_i finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.
- **D**urability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

In order to maintain ACID requirements and achieve maximum performance, a proper schedule of transactions need to be arranged in which the operations of various transactions are interleaved as much as possible. Given a schedule, a directed graph, referred to as precedence graph [28] or serialization graph [27], can be constructed to illustrate the procedure of