Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 62 (2007) 547–577
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Performance analysis of long-lived cooperative transactions in active DBMS

Prithwish Kangsabanik *, D.S. Yadav, R. Mall, A.K. Majumdar

Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India

Received 22 July 2006; accepted 10 November 2006
Available online 14 December 2006

Abstract

Active database management systems (ADBMS) are used in different application domains and especially for *cooperative* and *long duration* activity management. This paper deals with performance analysis of long-lived cooperative transaction processing in an ADBMS. We first briefly discuss NP-QuadLock – a concurrency control scheme for cooperative and long durational transactions in ADBMS. A restricted version of NP-QuadLock named 2L-QuadLock has been used for simulation. We have modeled such an ADBMS supporting 2L-QuadLock scheme by a queuing model. The failure of the transactions running in such systems has been modeled by a failure recovery model. We have simulated this model for a transaction processing system serving long-lived and cooperative transactions. We also discuss some important emerging application scenarios, where the proposed cooperative complex transaction mechanism can be used (e.g. 3G-service environment, ubiquitous computing environment, feature composition in intelligent network environment, multi-site and multi-domain web-services).

An important objective of our work is to analyze *quantitatively* (a) the performance penalty on the system due to the partial abort, the number of locks held by a transaction, the number of states of the transactions, and (b) the gain in the performance of the system with the cooperation semantics proposed in 2L-QuadLock concurrency control mechanism. We have analyzed the effect of various parameters such as partial abort rate, cooperation rate, number of locks held by a transaction, multiprogramming level, on the performance metrics such as average service time, average saga length and the degree of compensation. Later, we characterize the application scenarios based on some important simulation parameters, and discuss the application performance needs for each of the application scenarios. The required performance parameters that need to be used for these application scenarios and the corresponding performance results using 2L-QuadLock are also discussed.

Keywords: Active database; Concurrency control; Object oriented systems; Performance analysis; Queuing model

* Corresponding author. Present address: UBS AG, Postfach 8098, Zurich, Switzerland. Tel.: +41 79 653 1965.
E-mail addresses: prithwishk@yahoo.com, prithwish.kangsabanik@ubs.com, prithwish.kangsabanik@ieee.org (P. Kangsabanik), rajib@cse.iitkgp.ernet.in (R. Mall), akmj@cse.iitkgp.ernet.in (A.K. Majumdar).

0169-023X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2006.11.001

1. Introduction

Traditional database systems are called *passive* because they execute transactions or query only when they are explicitly requested by the user or the application program. In contrast, active database systems are capable of executing transactions whenever some events occur. The occurrence of these events may take place due to changes in the database states. In active DBMS (ADBMS), transactions have the ability to monitor the database objects and to take specific actions when a particular event occurs in the database objects. This is specified by means of Event–Condition–Action rules or ECA rules. These events may be generated due to a transaction's own object operation or due to the object operations generated by other transactions running in parallel. For some advanced database applications, it is necessary to monitor the conditions defined on the states of the database. As the conditions are satisfied, appropriate actions are invoked subject to the timing constraints. Examples of such applications, which are currently being developed using active databases, are hospital information system [1], Trading and stock control [2], Workflow management [3].

Increasing concurrency among transactions and maintaining the consistency of the database are two conflicting goals. This problem becomes more acute in ADBMS in the presence of long durational and cooperative transactions. The traditional notion of serializability as a correctness criterion turns out to be a bottleneck for long durational and cooperative transactions. In order to overcome this bottleneck, different kinds of transaction models e.g. Saga [4,5], nested transaction [6], cooperative transactions [7] have been developed. These transaction models generally use relaxed notion of correctness i.e., ACID (Atomicity, Consistency, Isolation and Durability) properties.

1.1. Long durational transactions and sagas

A transaction, which takes relatively long time to complete its execution even without any interference from other transactions, is known as Long Duration Transaction or Long-Lived transaction (LLT). Since an LLT takes a long time to finish its computation, it is more vulnerable to failures [8]. In addition, if an LLT retains all the locks acquired by it until it commits, a large amount of work has to be undone in the event of abort. This makes the recovery procedure more complicated and costly affair. An LLT imposes a relatively long waiting time for short transactions waiting for the locks acquired by the LLT. Thus, an LLT imposes serious constraints on the overall performance of the system.

A Saga [4,5] is a long durational transaction model that can be expressed as a series of subtransactions, called component transactions, which can be interleaved with other concurrently executing component transactions. If any activity needs to be aborted or partially aborted, a compensatory action may be taken for committed component transactions. These transactions are known as compensating transactions. Sagas provide high degree of concurrency by releasing the locks at the commit of the component transactions. A component transaction is executed atomically and it retains the locks until commit, following the two-phase locking protocol. Since Sagas are also LLTs, the overhead required to restore data consistency in case of failures cannot be avoided. However, Sagas support forward progress of activities.

Using an ADBMS, we can model long-lived transactions, where a transaction can be viewed as a collection of subtransactions. Execution dependencies among the subtransactions are not only defined in terms of significant events e.g. (Begin, Commit, Abort) of subtransactions but also in terms of monitored data base object events.

An ADBMS transaction may interact with other concurrent transactions by making its changes in the database accessible to other concurrently running transactions. Thus, a transaction processing system requires a controlled cooperation among these concurrent transactions. This means that before the commit of a transaction, the effects of execution of the transaction on the database will be visible only to those transactions that can cooperate with it.

In our earlier work, the execution of long durational and cooperative transactions has been investigated in [9,10], where a concurrency control scheme named NP-QuadLock has been proposed. NP-QuadLock exploits the semantics of the transactions to achieve better cooperation and concurrency among the ADBMS transactions. It gives a suitable criterion for "correctness" of execution of ADBMS transactions in the presence of inter-transactional events and detached mode ECA rules.