Data & Knowledge Engineering 66 (2008) 223–242

Contents lists available at ScienceDirect

Data & Knowledge Engineering

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

A self-adaptable scheduler for synchronizing transactions in dynamically configurable environments

Maristela Holanda [a,*], Angelo Brayner [b,1], Sergio Fialho [c,2]

[a] Catholic University of Brasília, Department of Computer Science, 72030-170 Brasília, Brazil
[b] University of Fortaleza, Department of Computer Science, 60811-341 Fortaleza, Brazil
[c] Federal University of Rio Grande do Norte, Department of Computer Engineering, 59072-970 Natal, RN, Brazil

ARTICLE INFO

Article history:
Received 27 October 2007
Received in revised form 28 February 2008
Accepted 28 February 2008
Available online 18 March 2008

Keywords:
Database
Concurrency control
Scheduler
Transaction and MDBC

ABSTRACT

In a database system, the scheduler has the goal of synchronizing operations belonging to several concurrent transactions. The scheduler implements a concurrency control protocol, which may have either conservative or aggressive behavior. Existing database systems have schedulers with static behavior. This paper presents a self-adaptable scheduler, called Intelligent Transaction Scheduler (ITS), which has the ability of dynamically changing its behavior to adapt itself to the characteristics of the computing environment, without any human interference by using an expert system based on fuzzy logic. ITS can implement different correctness criteria, such as classical (syntactic) serializability and semantic serializability.

1. Introduction

Users interact with database systems by means of application programs. A transaction is an abstraction that represents a sequence of database operations resulting from the execution of an application program [28]. A database system should synchronize the execution of concurrent transactions to guarantee database consistency. The component in the database system responsible for synchronizing the operations of concurrent transactions is the scheduler. The scheduler synchronizes operations belonging to different transactions by means of a concurrency control protocol.

Concurrency control protocols may present aggressive or conservative behavior [28]. An aggressive concurrency control protocol tends to avoid delaying operations, and tries to schedule them immediately. However, such a strategy may lead to a situation in which there is no possibility of yielding a correct execution of all active transactions. In this case, operations of one or more transactions should be rejected. A conservative protocol, on the other hand, tends to delay operations in order to synchronize them correctly.

Additionally, aggressive concurrency protocols can be categorized in two sub-classes: pessimistic and optimistic. A scheduler implementing a pessimistic protocol decides to accept or reject the operation as soon as it receives the operation. On the other hand, a scheduler using an optimistic protocol immediately schedules the operation. After a given period of time, the scheduler verifies the correctness of the schedule it has already produced. If the schedule is correct, the scheduler continues synchronizing operations. Otherwise, the scheduler must abort some transactions.

* Corresponding author. Tel.: +55 81 33569000.
E-mail addresses: mholanda@ucb.br (M. Holanda), brayner@unifor.br (A. Brayner), fialho@pop-rn.rnp.br (S. Fialho).
[1] Tel.: +55 85 43773268.
[2] Tel.: +55 84 32153171.

0169-023X/$ - see front matter Crown Copyright © 2008 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2008.02.004

The concept of serializability [18] has been used since the 1970s by concurrency control protocols as a correctness criterion to ensure database consistency. The Two Phase Locking Protocol (2PL) is the most widely used conservative protocol based on serializability. Among the protocols with aggressive behavior that use a pessimistic approach, the Timestamp Ordering (TO), and the Serialization Graph Testing (SGT), are noteworthy. A scheduler implementing an aggressive concurrency control protocol is called an aggressive scheduler, whereas a scheduler implementing a conservative concurrency control protocol is called a conservative scheduler.

A Mobile Ad hoc NETwork (MANET) [16] is a wireless dynamic network, in which nodes can join or leave the network dynamically. Some database applications that use MANET technology are: industrial and commercial applications involving cooperative mobile data exchange [16]; military networks [21,26]; vehicular ad hoc networks [9]; city services, such as the taxi company system presented in [12]. MANETs allow users carrying portable devices to access database services regardless of their physical location or movement patterns. Thus, a dynamic collection of autonomous mobile databases, called mobile database community (MDBC) [3], can be formed, where the mobile databases reside in nodes (Mobile Units – MU) of a MANET. Since new participants may join an MDBC or a participant may transiently disconnect from the MDBC (due to communication disruptions or to save power), an MDBC is characterized as a dynamically configurable environment. Observe that the concept of MDBC models a dynamically configurable multidatabase [6] whose members (local databases) can move across different locations.

As already mentioned, mobile units may join and leave an MDBC at anytime. In such a dynamic environment, the use of a fixed concurrency protocol (e.g., using only strict 2PL) to generate correct schedules may not be the most efficient strategy at given moments over a period of time. This is because a scheduler with static behavior is not able to capture modifications in the context it is being executed. As shown in Section 4, the arrival of a new mobile unit in an MDBC drastically modifies the scenario in which transactions are being executed. For example, consider that the new mobile unit submits transactions with several write operations and a scheduler implementing a conservative protocol (e.g., strict 2PL) is being used. In such a scenario, the conflicting operation rate may increase (see Section 4), which may induce low throughput rates (low degree of parallelism).

This paper presents the Intelligent Transaction Scheduler (ITS), characterized by having a hybrid behavior (conservative or aggressive). The gist of this approach is to provide a scheduler that can automatically identify and react to the modifications in the computational environment in which it is inserted, by adapting its behavior (from aggressive to conservative and vice versa), without human interference. In other words, ITS is context-aware. It is shown in Section 4 that ITS achieves a reasonable throughput, while generating correct global schedules in dynamically configurable environments. Furthermore, the ITS can be adjusted to use correctness criteria other than classical serializability, since that criterion is very restrictive for controlling concurrency in dynamically configurable environments, such as an MDBC. Therefore, we claim that an adaptive scheduler is quite appropriate for controlling concurrency in dynamically configurable environments, whose configuration (topology) may change dynamically.

It is worthwhile to note that although the ITS has been developed for use in dynamically configurable environments, it can also be used to synchronize operations of concurrent transactions in any other distributed environment with static topology as well.

The remainder of this paper is organized as follows: Section 2 presents the architecture of the proposed scheduler (ITS); Section 3 presents the theoretical support which proves the correctness of ITS' self-adaptability (i.e., ITS produces correct schedules); Section 4 describes aspects of ITS' implementation, its use in MDBCs and an analysis of the obtained results; Section 5 describes some of the existing related proposals in the literature; and finally, Section 6 concludes this paper.

2. ITS – intelligent transaction scheduler

The ITS [24,25] is an intelligent scheduler and its architecture is composed of two modules, as shown in Fig. 1: the *Analyzer*, a component that makes decisions related to the most appropriate scheduler behavior according to characteristics of the computing environment during a given time period; and the *Scheduler*, the component that executes the concurrency control protocols selected by the Analyzer.

2.1. Analyzer

The Analyzer is an Expert System – ES based on fuzzy logic [19]. The Analyzer module presented in Fig. 1 illustrates the basic components of the ES: *Knowledge Database*, which stores the specialized knowledge used in decision making; *Context*, which keeps the values of the variables used to solve the problem; *Inference Engine*, which infers knowledge by means of the rules stored in the *Knowledge Database* and the variable values stored in the *Context*; and *Interface*, which controls the dialogue between the user (Scheduler) and the ES. The *Interface* receives the information sent by the Scheduler, stores it in the *Context* and sends the information about the decision made internally back to the Scheduler.

The ITS' Analyzer was developed using the process presented in [7], which involves five steps. Each one of these steps is described as follows.

2.1.1. Specification of the problem and identification of input and output fuzzy variables

The problem is to choose the most appropriate scheduler behavior, either aggressive or conservative, according to a number of computing environment characteristics. The most appropriate behavior should reduce the delays caused by the synchronization of the operations by the scheduler, while keeping the aborted transaction rate within the lowest possible range.