Engineering Applications of Artificial Intelligence 39 (2015) 55–79

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A new approach to active rule scheduling

Abbas Rasoolzadegan [a,*], Rohollah Alesheykh [b], Mohammad Reza Meybodi [c]

[a] Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Faculty of Engineering, Payame Noor University (PNU), Iran
[c] Department of Computer Engineering and IT, Amirkabir University of Technology, Iran

ARTICLE INFO

Article history:
Received 13 July 2014
Received in revised form
13 November 2014
Accepted 14 November 2014
Available online 16 December 2014

Keywords:
Active database management systems
Probability estimation
Active rule scheduling
Learning automata

ABSTRACT

Active database systems (ADSs) react automatically to the occurrence of predefined events by defining a set of active rules. One of the main modules of an ADS is the rule scheduler, which has a significant impact on the effectiveness and efficiency of ADSs. During the rule scheduling process, the rule scheduler is responsible for choosing one of the activated or ready-to-be-executed rules to evaluate its condition section or execute its action section, respectively. This process continues until there is no rule to be evaluated or executed. In this research, we evaluate and compare existing rule scheduling approaches in a laboratory environment based on a three-tier architecture. There are criteria used for the evaluation and comparison of rule scheduling approaches: Average Response Time, Throughput, Response Time Standard Deviation, Time Overhead per Transaction, and CPU Utilization. The three first criteria are used to evaluate the effectiveness, and the latter two criteria are used to evaluate the efficiency of rule scheduling approaches. In this paper, a new approach, referred to as E_X-SJF$_{EsTLA}$, is proposed to improve the rule scheduling process, using a learning automaton. In our laboratory environment, E_X-SJF$_{EsTLA}$ is compared with those rule scheduling approaches that are unconstrained as E_X-SJF$_{EsTLA}$ is. Unconstrained scheduling approaches serially schedule the rules that do not have any priorities or deadlines. The results of experiments revealed that the proposed approach improved the rule scheduling process according to the evaluation criteria.

1. Introduction

Common (Traditional) database systems are often of a passive nature. This means that operations such as querying, updating, inserting, deleting, and reporting are performed only in the event that users request them. Database Management Systems, abbreviated to DBMSs, cannot automatically react when various events occur. Many applications such as real-time expert systems (Hangos et al., 2001; Farias et al., 2009), warehousing programs, the automation of processes, and complex financial calculations in stock markets need automatic control for handling events that have occurred. Active Database Systems (Kasbon et al., 2012), abbreviated to ADSs, meet the requirements of such applications by defining ECA (Event-Condition-Action) rules, referred to as active rules. An ECA rule has three main sections: Event, Condition, and Action. An ECA rule for the context of buying and selling stocks (Zong et al., 2007) is defined as follows:

DEFINE *LowRisk*	*This active rule is named "LowRisk"*
ON *Stock.UpdatePrice*	*Event section*
IF *(Stock.policy = Low_risk) and (Stock.price < Stock.initprice * e); (0 < e < 1)*	*Condition section*
DO *Stock.Buy*	*Action section*

In addition to event, condition, and action components, each active rule has two other features: event-condition coupling mode and condition-action coupling mode. When an event E_1 occurs, the event-condition coupling mode of each rule triggered by E_1 determines the time when the condition section of the rule should be evaluated. There are three choices for the event-condition coupling mode of each active rule: *immediate*, *deferred*, and *independent*. If the event-condition coupling mode of an activated rule is immediate, the condition of the rule should be evaluated immediately. If the event-condition coupling mode of an activated rule is deferred, the evaluation of the condition of the rule should

* Corresponding author.
E-mail addresses: rasoolzadegan@um.ac.ir (A. Rasoolzadegan), alesheykh@pnu.ac.ir (R. Alesheykh), meybodi@aut.ac.ir (M.R. Meybodi).

http://dx.doi.org/10.1016/j.engappai.2014.11.005
0952-1976/

be deferred until the execution of the action section of the rule that is being executed is terminated. If the event-condition coupling mode of an activated rule is independent, the condition section of the rule is evaluated only after the condition sections of all activated rules with the immediate and deferred modes have been evaluated. For example, suppose there are some activated rules, waiting to be evaluated. The condition sections of the rules with the independent event-condition coupling mode are not evaluated until the condition sections of the other rules have been evaluated. If the condition section of an activated rule is evaluated to true, the condition-action coupling mode of the rule determines the time when its action section should be executed.

Similarly, there are three choices for the condition-action coupling mode of each rule: immediate, deferred, and independent. If the condition-action coupling mode of a ready-to-be-executed rule (a rule whose condition-section has been evaluated to true) is immediate, its action section must be executed immediately. If the condition-action coupling mode of a ready-to-be-executed rule is deferred, the execution of its action section should be deferred until the execution of the action section of the rule that is being executed is terminated. If the condition-action coupling mode of a ready-to-be-executed rule is independent, the action section of the rule is executed only after the action sections of ready-to-be-executed rules with immediate and deferred modes have been executed.

Events can be classified as primitive or composite events. Primitive events refer to elementary occurrences which are predefined in the system. Primitive events are typically further categorized as database events, temporal events, transaction events, etc. Database events are related to database operations and are further classified into Insert, Delete, and Update. A temporal event can be an absolute point in time, defined by the system clock (e.g., 9:00:00 a.m., April 10, 1988), relative (30 s after event *A* occurred), or periodic (every day at midnight). As the name implies, transaction events are kinds of events related to transactions; for example, the beginning and end of a transaction are events signaled at the beginning and end of the transaction. A composite event is a set of primitive events which are combined using event operators (such as “and”, “or,” and “not”) to form a new event specification.

An ADS processes its active rules to automatically control various events. The rules processing cycle, elaborated on in Section 2, consists of the following steps (Meenakshi and Thiagarasu, 2014):

1) **Event Signaling:** When a primitive event occurs, the primitive event detector signals it. In addition, the composite event detector considers the occurring primitive events to investigate the occurrence of composite events.
2) **Rule Triggering:** After an event is signaled, those ECA rules that correspond to the signaled event are activated. One instance of each activated rule is created, which includes some additional information (such as a timestamp, a deadline, and an execution time) depending on the scheduling mechanism used by the rule scheduler. These instances are buffered to be used in the next step.
3) **Condition Evaluation:** In this step, the rule scheduler sequentially selects the instances created in the previous step, and their conditions are evaluated. If the condition section of an instance is evaluated to true, the instance is added to the ready-to-be-executed buffer.
4) **Transaction Selection:** In this step, the rule scheduler selects the ready-to-be-executed rules in order of priority. A transaction is generated for each ready-to-be-executed rule based on its action section. The transaction is then sent to the execution unit. This step is also called the transaction scheduling phase.
5) **Transaction Execution:** The transactions generated in the previous phase are executed in this step.

ADSs have many applications in controlling and automatic handling of processes in different areas such as the stock exchange organization, portfolio management systems, automatic traffic control systems, all applications with continuous monitoring, and real-time systems as real-time active database systems (Kasbon et al., 2012; Ale and Espil, 2003; Spokoiny and Shahar, 2007; Badia, 2003; Qiao et al., 2007). An ADS plays the role of an automatic controller by defining and executing several active rules. During the running of the system, several activated rules with different event-condition and condition-action coupling modes are waiting to be evaluated or executed.

There is an important question here: why is it important to have an effective and efficient rule scheduling algorithm in ADSs? To answer this question, consider a stock exchange system that uses an ADS. This ADS has tens of thousands of users, and the number of active rules in their rule-bases reaches hundreds of thousands of rules. For example, suppose there are n users in the electronic management system of the stock exchange. Each user has defined a different number $(m_1, \ldots, m_n)$ of active rules. The definition of these rules is illustrated in Table 16 (see Appendix). After each update operation (i.e. edit, insert, or delete) in the electronic management system of the stock exchange, thousands of rules are activated. Suppose that in a stock exchange system with tens of thousands of users, n' number of users ($n' < n$ and n' is rather big; for example, n' is about two or three thousands), define some rules triggered by each update on the price of Mercedes Benz. When the price of this stock is updated, in a moment, the corresponding rules are activated. At this moment, the stock price of the Volvo Company may be updated as well. And this may cause thousands of other rules to be activated. Such scenarios may continue. There are several similar scenarios that may happen during the running of the electronic management system of the stock exchange. It is obvious that during the running of these kinds of systems in the real-world, many activated rules wait to be evaluated and executed. And in these situations, even milliseconds matter. If a rule, defined by user *X*, is not executed at the desired moment, the goal of a user *X* is not satisfied. This goal may be “buying a stock A” or “selling a stock B,” so user *X* does not obtain his/her expected benefit. Thus, the importance of having an effective and efficient rule scheduling mechanism is quite evident.

So far many rule scheduling mechanisms have been introduced such as Random (Adaikkalavan and Chakravarthy, 2012), FCFS (First Come, First Served) (Sarkar and Debnath, 2012), and E_x-SJF (Extended Shortest Job First) (Rasoolzadegan et al., 2008), presented in Section 3. There are also some criteria for evaluating the effectiveness (performance) and efficiency of rule scheduling approaches such as Average Response Time, Throughput, and CPU Utilization, presented in Section 2.

The scheduling of active rules is one of the main research topics in ADSs (Rasoolzadegan, 2007; Ceri et al., 2003; Jin, 2009; Jin et al., 2007; Meenakshi and Thiagarasu, 2014; Meenakshi and Thiagarasu, 2014; Saravanapandi Solairajan et al., 2013; Narang et al., 2013). There have been several attempts in this area. The focus is mainly on the unconstrained rule scheduling approaches - such as the various versions of E_x-SJF - rather than on those introduced for constrained rule scheduling approaches - such as the “Static Priority” approach. Constrained rule scheduling approaches are used when rules have some restrictions (or constraints) such as deadlines or priorities. For example, in safety-critical and high-integrity systems – such as the automatic control system of an airplane – priorities are assigned to rules according to their importance. Among all activated or ready-to-be-executed rules, the rule that has the highest priority is selected to be evaluated or executed first. The priority of rules is determined by rule developers before the running of the system. In real-time systems, each rule has its own deadline. Therefore, initially, it is necessary to define the deadline of each rule accurately. The deadline of a rule is the greatest amount of time that the execution of the action section of the